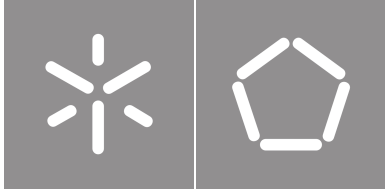
Universidade do Minho
Escola de Engenharia

Carlos Miguel Azevedo Magalhães

Implementação de serviços Confort@Home

Outubro de 2022



Universidade do Minho
Escola de Engenharia

Carlos Miguel Azevedo Magalhães

Implementação de serviços Confort@Home

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efetuado sob a orientação de:
José Manuel Ferreira Machado

Outubro de 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Agradecimentos

Em primeiro lugar quero deixar um agradecimento à minha família, em especial aos meus pais, por todo o apoio e sacrifício que fizeram ao longo destes anos, por me proporcionarem todas as condições para a conclusão deste percurso e pelo apoio moral, amor e dedicação.

À minha namorada e amigos por toda a compreensão e motivação, por me acompanharem ao longo destes anos, tornando o meu percurso académico inesquecível.

Quero agradecer ao Professor José Machado pela orientação e sugestões que melhoraram esta dissertação.

Por fim, quero agradecer à empresa *Altice Labs*, em especial à Telma Mota e à equipa do *SmartAL*, pela sugestão deste tema de dissertação, pela confiança depositada e por todo o apoio e conhecimento transmitido durante a realização deste trabalho.

DECLARAÇÃO DE INTEGRIDADE

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

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

_____, _____

(Local) (Data)

_____

(Carlos Miguel Azevedo Magalhães)

Resumo

Implementação de serviços Confort@Home

A área de *eHealth/Assisted Living* tem vindo a crescer com a necessidade de melhorar a qualidade de vida de pessoas que precisam de acompanhamento médico permanente (e.g., doença crónica) ou ocasional (e.g., pós-operatório, pandemia), de forma presencial ou à distância. No entanto, a tendência é manter as pessoas em casa o máximo tempo possível e não nas instituições, evitando o perigo de contaminação, precavendo a rutura dos serviços de saúde, e garantindo simultaneamente maior conforto e bem-estar ao utente. Contudo, estas pessoas precisam de ser acompanhadas e ter acesso a serviços de saúde de qualidade, tal como se estivessem nas instituições. Os serviços de telemedicina e telemonitorização servem este propósito, permitindo a cuidadores formais e/ou informais estar em contacto com os seus doentes, acompanhá-los remotamente em tempo real e prevenir situações de maior risco, agindo rapidamente em casos de urgência. No âmbito específico da telemonitorização, esta pode ser clínica e/ou não clínica, mais focada no controlo de sinais vitais ou apenas no acompanhamento da pessoa com base noutro tipo de informação pessoal (e.g., atividade, sono, localização), respetivamente. No caso particular desta dissertação, o objetivo é desenhar, especificar e implementar um serviço que ofereça conforto e segurança usando essencialmente informação não clínica e assegure o acompanhamento remoto de pessoas de idade avançada que vivam ou passem muito tempo sozinhas. A necessidade de criação de uma ferramenta deste tipo deve-se ao facto de, cada vez mais, se pretender apostar na prevenção e segurança das pessoas, sobretudo das pessoas de idade que se encontram em situações de alguma fragilidade (e.g., por doença, por isolamento), mas simultaneamente tentar passar um sentimento de companhia, mesmo à distância, procurando preservar ao máximo a sua autonomia. Neste contexto, o projeto pretende explorar cenários mistos de telemonitorização, através de informação proveniente de sensores ligados à casa e dispositivos *wearables*, criando o conceito de *Confort@Home*, ou seja, um conceito de saúde e bem-estar centrado na pessoa, na sua casa e no ambiente circundante. A oferta de serviços a considerar incluirá informação de localização dentro e fora de casa, dados adicionais provenientes de dispositivos a selecionar e notificações/alertas; na lógica da aplicação serão definidas as situações e condições que espelhem os possíveis riscos e falta de segurança do idoso, e enviados alertas para o seu cuidador/familiar, de forma a facilitar a vida de ambos no dia-a-dia. Como demonstração do conceito, será desenvolvida uma das vertentes do *Confort@Home* que se materializa numa aplicação *Web* fornecida em ambiente *cloud* que permita a cuidadores informais monitorizar os seus familiares/dependentes com base na sua localização.

Palavras-chave: Smart Home, SmartAL, eHealth, Assisted Living, Telemonitorização

Abstract

Confort@Home services implementation

The area of *eHealth/Assisted Living* has been growing with the need to improve the quality of life of people who need permanent (e.g., chronic disease) or occasional (e.g., post-operative, pandemic) medical monitoring, in person or remotely. However, the trend is to keep people at home for as long as possible, rather than in institutions, avoiding the danger of contamination, preventing the breakdown of health services, and at the same time ensuring greater comfort and well-being for the user. Nevertheless, these people need to be accompanied and have access to quality health services, just as if they were in the institutions. Telemedicine and telemonitoring services serve this purpose, allowing formal and/or informal caregivers to stay in touch with their patients, to monitor them remotely in real time, and to prevent higher risk situations by acting quickly in cases of emergency. In the specific scope of telemonitoring, this monitoring may be clinical and/or non-clinical, i.e., more focused on monitoring vital signs or only on monitoring the person based on other types of personal information (e.g., activity, sleep, location), respectively. In the particular case of this dissertation, the goal is to design, specify and implement a service that offers comfort and security using essentially non-clinical information and ensures remote monitoring of elderly people who live or spend a lot of time alone. The need to create a tool of this type is due to the fact that, increasingly, we want to focus on the prevention and safety of people, especially the elderly who are in situations of some fragility (e.g., due to illness, isolation), but simultaneously trying to pass on a sense of companionship, even at a distance, trying to preserve their autonomy as much as possible. In this context, the project aims to explore mixed telemonitoring scenarios, through information from sensors connected to the house and wearable devices, creating the concept of *Confort@Home*, that is, a concept of health and well-being centered on the person, their home and the surrounding environment. The service offering to be considered will include location information inside and outside the home, additional data from devices to be selected, and notifications/alerts; in the application logic, situations and conditions that mirror possible risks and lack of safety of the elderly person will be defined, and alerts will be sent to the caregiver/family member, in order to facilitate the life of both on a daily basis. As a demonstration of the concept, one of the aspects of the *Confort@Home* will be developed, materialized in an web application provided in an cloud environment that will allow informal caregivers to monitor their relatives/dependents based on their location.

Keywords: Smart Home, SmartAL, eHealth, Assisted Living, Telemonitoring

Índice

Índice de Figuras

Índice de Tabelas

Siglas

AMQP Advanced Message Queuing Protocol 43

B2B Business to Businnes 16
B2B2C Business to Business to Consumer 16
BPM Batidas por Minuto 13

CET Centro de Estudos de Telecomunicações 2
CHUCB Centro Hospitalar Universitário Cova da Beira 14
CHULC Centro Hospitalar Universitário Lisboa Central 14
CNTS Centro Nacional do TekSaúde 14, 15

DECO Defesa do Consumidor 6
DPOC Doença Pumonar Obstrutiva Crónica 14
DSR Design Science Research 2, 79

ER Entidade-Relacionamento 34

GPS Global Positioning System 4, 10, 16, 17, 19, 20, 23, 31, 33, 36

HDS Hospital Distrital de Santarém 14

IA Inteligência Artificial 2
ID Investigação e Desenvolvimento 2
IoT Internet of Things 15

ML Machine Learning 2

NPM Node Package Manager 42

ONU Organização das Nações Unidas 5
ORM Object-relational mapping 42

SGBD Sistema de Gestão de Base de Dados 43
SmartAL Smart Assisted Living 1
SNS Serviço Nacional de Saúde 6, 14
SPMS Serviços Partilhados do Ministério da Saúde 14, 15
SQL Structured Query Language 43

TERI Telemonitorização de Doentes em Risco 14
TIC Tecnologias da Informação e Comunicação 2

UE União Europeia 5

1 Introdução

Neste capítulo introdutório, será contextualizado o tema da dissertação, assim como a metodologia utilizada para a sua realização, e será feita uma breve apresentação da empresa que lançou o desafio. De seguida, serão apresentados os objetivos e resultados esperados e, por fim, um pequeno resumo da restante estrutura do documento.

1.1 Contexto e Motivação

A área de *eHealth/Assisted Living* tem vindo a crescer com a necessidade de melhorar a qualidade de vida de pessoas. A pandemia catapultou a necessidade de exercer telemedicina e prestar cuidados de saúde à distância. No entanto, para o fazer é necessário recorrer à tecnologia e as pessoas mais idosas apresentam ainda dificuldades em lidar com novos terminais e aplicações devido à sua natural falta de literacia digital. Esta realidade realça o papel do cuidador informal, do vizinho ou do familiar que, dada a proximidade física e/ou emocional, possa ajudar e acompanhar mais de perto estas pessoas. É certo que a maioria dos idosos, a partir de determinada altura, passa a viver em instituições, mas pretende-se cada vez mais prolongar a sua estadia em casa, assegurando a segurança e conforto necessários, pois está provado que é essa a preferência da maior parte da população sénior [17]. No entanto, persiste a preocupação do isolamento e da falta de acesso a bons cuidados de saúde. Neste sentido, é necessário assegurar o mais possível a autonomia dos idosos e garantir o seu bem-estar, implementando ferramentas que permitam acompanhar à distância esta população. O conceito “envelhecer em casa” ganha cada vez mais tração, mas é necessário reforçar a telemonitorização para melhorar a qualidade dos serviços de saúde. Usufruindo do ambiente ”casa e área circundante”, pode-se cruzar informação proveniente da telemonitorização clínica (já recolhida pela aplicação da *Altice Labs SmartAL* [18]), com informação adicional ligada à telemonitorização não clínica (e.g., localização) e à domótica (e.g., sensores de deteção de quedas, de abertura de portas, de movimento, de qualidade do ar e câmaras), criando cenários interessantes de ”casa inteligente”aplicados especificamente à área da saúde e do bem-estar. Em particular, para os idosos (ou para outras pessoas em situações de risco ou que sofram de patologias leves - e.g., demência ligeira, é importante continuar a garantir a sua autonomia conforme vão perdendo capacidades,

mas em segurança, ou seja, criando mecanismos, de preferência não intrusivos, que permitam o acompanhamento remoto por parte de cuidadores informais (i.e., familiares, amigos, outros), assim como, em casos mais graves, de cuidadores formais (i.e., profissionais de saúde). Neste sentido, serviços que permitam identificar a normalidade dentro das rotinas diárias da pessoa e, por oposição, prever e identificar situações de risco são da maior importância. Em específico, através da informação de localização, quer dentro de casa quer nas imediações, deteção de movimento, assim como outra correlacionada (i.e., quedas, "qualidade" dos passos, etc.), será possível construir serviços que realmente contribuam para uma maior tranquilidade e segurança, tanto do utente como dos que o rodeiam.

1.2 Apresentação da Empresa

A *Altice Labs* é uma empresa tecnológica de inovação do grupo *Altice*, fundada em 1950 como Centro de Estudos de Telecomunicações (CET). Tem como objetivo entregar aos seus clientes serviços e produtos avançados de alta qualidade, personalizados e diferenciados, fatores que a destacam no mercado altamente competitivo. Para tal, conta com um ecossistema de inovação forte e dinâmico, com foco na investigação, desenvolvimento e inovação de novas soluções e tecnologias em TIC, essencialmente em 4 áreas de negócio: conectividade, sistemas de suporte às operações, controlo de rede e plataformas de serviços e aplicações [19].

Como parte da estratégia sustentada de liderança tecnológica, a empresa integra projetos em colaboração com vários tipos de instituições, tais como universidades, institutos de Investigação e Desenvolvimento (ID), fornecedores e clientes. São diversas as áreas de ID em que se envolve, com especial incidência, atualmente, nas de *Inteligência Artificial (IA) & Machine Learning (ML)*, *Cloud*, *Smart Living* e Segurança & Privacidade, por serem atuais e desafiantes e por acreditar que poderão no imediato criar diferenciação e valor no mercado [19].

Em particular na área da saúde e bem-estar, a *Altice Labs* desenvolveu dois produtos: a) o *Medigraf*, uma solução de teleconsulta que opera em vários hospitais desde 1998, com capacidade de assegurar, para além da comunicação áudio-vídeo entre profissionais de saúde especializados residentes em hospitais centrais e outros não especializados de instituições mais remotas, a transmissão remota e em tempo real de sinais de aparelhos clínicos (e.g., ecógrafos) e b) o *SmartAL*, uma solução de telemonitorização clínica com capacidade de coletar dados de vários dispositivos clínicos e não clínicos, assim como de questionários e outras fontes de informação relevantes para o doente.

1.3 Metodologia de Investigação

Este trabalho foi realizado segundo a metodologia *Design Science Research (DSR)*, que é bastante utilizada para desenvolver soluções para problemas práticos de engenharia [25]. O principal objetivo desta

metodologia é desenvolver conhecimento específico, de forma a conceber a verdadeira solução para o problema em questão.

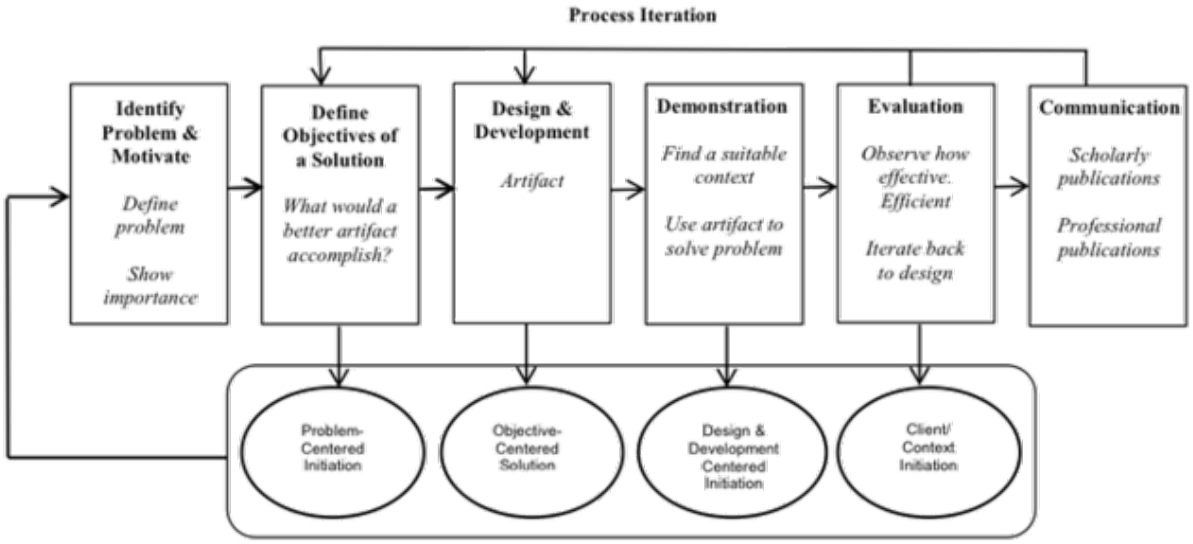


Figura 1: Design Science Research Methodology

Como se pode observar pela figura 1, esta metodologia está divida em seis etapas, sendo que Peffers et al. afirmam que as pesquisas não têm de começar sempre a partir do primeiro passo (ou seja, a identificação), mas que na sua maioria passam por todos os passos de uma forma ou outra, deslocando-se para fora a partir do ponto de entrada da pesquisa [25]. As etapas são as seguintes:

- **Identificação do Problema e Motivação** - Nesta etapa define-se o problema e justifica-se o valor de uma solução;
- **Definição dos Objetivos da Solução** - Tendo em conta o estado atual do problema e as soluções já existentes para o mesmo, são definidos os objetivos para o artefacto a desenvolver;
- **Design e Desenvolvimento** - Nesta etapa o artefacto começa a ser construído, definindo a sua arquitetura, o modelo, métodos etc.;
- **Demonstração** - Utiliza-se o artefacto desenvolvido para resolver uma ou mais instâncias do problema, recorrendo a simulações, experiências, casos de estudo real entre outros;
- **Avaliação** - É efetuada a avaliação da solução, comparando os resultados com os objetivos inicialmente definidos. Mediante os resultados obtidos pode-se voltar ao Design e Desenvolvimento do artefacto, ou então avançar para o próximo passo e deixar indicações para trabalho futuro;
- **Comunicação** - Por fim é realizada a comunicação do problema, do artefacto desenvolvido, assim como da sua utilidade e eficácia em relação a outras soluções.

1.4 Objetivos e Resultados Esperados

Seguindo então a metodologia indicada no capítulo anterior, e após identificar na introdução o problema e a motivação, avança-se para a definição dos objetivos.

O projeto pretende explorar cenários mistos de telemonitorização com informação proveniente de dispositivos GPS, dando início ao conceito de *Confort@Home*, um conceito abrangente de oferta de serviços de saúde e bem-estar que pretende acompanhar e facilitar a vida do utilizador no seu dia-a-dia. Por exemplo, a deteção de uma queda ou a ausência de movimento são indicadores alarmantes para quem é suposto estar casa em telemonitorização clínica, assim como o distanciamento exagerado da casa para quem tem mobilidade reduzida. Estes são apenas alguns exemplos, mas considerar informação adicional à de telemonitorização clínica (já coletada pela solução *SmartAL* da *Altice Labs*) ajudará com certeza no futuro a reforçar a prevenção, permitindo sinalizar situações anómalas complementares aos profissionais de saúde e familiares. No caso particular desta dissertação, o foco estará no desenvolvimento de um serviço denominado *Teleloc* e na análise de situações de potencial risco à volta de casa, tendo em conta dispositivos (*wearables*) dotados de localização GPS.

1.5 Estrutura do Documento

No primeiro e presente capítulo, foi apresentado o contexto em que a dissertação foi realizada, as motivações para a sua realização, a metodologia utilizada e uma breve apresentação da *Altice Labs*, empresa que propôs o projeto.

No segundo capítulo, serão apresentados conceitos base relativos à temática abordada na dissertação, começando pelo enquadramento sobre a população mais idosa e como esta vive em Portugal, e sobre a necessidade de manter a autonomia destas pessoas e, ao mesmo tempo, garantir o seu acompanhamento regular para lhes oferecer mais segurança. Ainda neste contexto, são também apresentados os conceitos de telemonitorização e serviços de localização.

No terceiro capítulo, é feita uma descrição dos objetivos e requisitos identificados no desenvolvimento da solução. São também apresentados alguns casos de uso e *storyboards*, para uma melhor compreensão da solução, e por fim, definem-se a arquitetura funcional, o modelo de dados e os diagramas de sequência, de forma a detalhar os serviços a implementar.

No quarto capítulo, é apresentado todo o processo de desenvolvimento da aplicação, como foi implementada e as ferramentas utilizadas, assim como os diferentes microsserviços e como estes funcionam na aplicação como um todo. É também feita uma apresentação da interface do utilizador e como foi realizado o *deployment* da aplicação na *cloud*.

No último e quinto capítulo, são apresentadas as considerações finais, onde se avalia o trabalho efetuado, os desafios que surgiram e como estes foram ultrapassados e também as perspetivas de trabalho futuro, identificando as melhorias e funcionalidades que poderão ser implementadas.

2

Estado da Arte

Este capítulo apresenta os conceitos base inerentes à temática abordada nesta dissertação. Começa dando enquadramento sobre o envelhecimento da população, como os idosos vivem em Portugal e sobre a necessidade de os manter nos seus lugares de conforto com acompanhamento regular. Posteriormente apresentam-se alguns serviços de telemonitorização, e por fim, é feita uma breve análise sobre serviços de localização.

2.1 Enquadramento

Progressivamente tem-se assistido a um envelhecimento da população mundial. De acordo com a Organização das Nações Unidas (ONU) o envelhecimento da população está prestes a tornar-se numa das transformações sociais mais significativas do século XXI, estimando-se que o número de idosos, com 60 ou mais anos, duplique até 2050 e triplique até 2100, passando de 962 milhões em 2017 para 2,1 mil milhões em 2050 e 3,1 mil milhões em 2100 [24].

Um estudo realizado pelo Gabinete de Estatística da União Europeia — Eurostat, com dados extraídos em junho de 2021, revela que Portugal é o 4.º país com a população mais envelhecida da União Europeia, estando apenas à sua frente Grécia, Finlândia e Itália. Em 2010 a percentagem de população idosa fixava-se nos 18,3% e em 10 anos esta percentagem aumentou 3,8%, passando a representar 22,1% da população nacional (cerca de 2.299 mil idosos), estando este valor acima da média europeia que se fixava nos 20,6%. De acordo com o mesmo estudo, prevê-se que este valor aumente cerca de 10,7%, representando as pessoas com 65 anos ou mais, cerca de 31,3% da população da União Europeia UE. Em Portugal, estima-se que este valor represente 32,2% da população nacional [7].

Se por um lado, Portugal é um dos países com a população mais idosa, por outro é o país com menor percentagem de população idosa a viver sozinha, com apenas 21,2% [6], o que representa cerca de 488mil idosos a viverem sozinhos, um valor bem abaixo da média europeia (31,4%). A partir destes dados, pode-se concluir que, apesar de o país ter uma das populações mais idosas da UE, grande parte desta faixa etária não vive sozinha, o que também não significa que vivam no seu seio familiar, podendo apenas ser um indicador de que grande parte vive em hospitais, lares, centros de dia etc. Aliás, através

de dados revelados pelo Serviço Nacional de Saúde (SNS) [29] em agosto 2020, existem em Portugal 2526 estruturas residenciais para idosos, com capacidade para apenas 99 234 utentes. Portanto, tendo em conta os valores apresentados e a recente situação pandémica, é possível afirmar que não existem infraestruturas suficientes para acompanhar convenientemente os idosos em Portugal, nem parecem existir grandes condições para criar muitas mais, para além das que já prestam apoio à população considerada ativa (entre 15 e 64 anos).

Pode ser necessário acompanhar um paciente idoso por vários motivos, seja por doença crónica, doença ocasional, recuperação (e.g., pós-operatório, convalescença), ou mesmo por prevenção (e.g., confinamento profilático, isolamento). Mediante o seu quadro clínico, o acompanhamento necessário ter-se-á de adaptar ao paciente que tanto pode conseguir autonomamente realizar certas tarefas do dia-a-dia, como encontrar-se debilitado e muito dependente de terceiros. Perante as estatísticas já apresentadas, conclui-se que um paciente que não padeça de um quadro clínico grave, mas que necessite, no entanto, de acompanhamento, pode não conseguir receber o apoio necessário, pelo facto de não existirem infraestruturas/condições suficientes para prestar esse apoio. De acordo com um estudo lançado pela DECO/Proteste (DECO) em 2020 [4], um idoso pode esperar até seis meses para conseguir entrar num lar e ter o apoio que necessita. Para além de uma demorada lista de espera, o apoio recebido numa instituição onde é admitido pode não favorecer o seu quadro clínico; para reforçar esta ideia, o mesmo estudo indica que muitos dos lares apenas se limitam a garantir um espaço para viver, refeições e cuidados básicos de saúde, salientando a ausência de atividades estimulantes, tanto do ponto de vista físico, como mental, o que tem impacto direto na qualidade de vida do idoso. No entanto, o agravamento do quadro clínico de uma pessoa num lar pode não se dever exclusivamente à falta de condições, mas depender também do próprio idoso. De acordo com Laranjeira Tereso et. al [34], à medida que as pessoas envelhecem vão-se tornando mais sensíveis ao meio em que vivem, diminuindo as suas capacidades de adaptação, e quando são deslocadas de sua casa para estas instituições, dificilmente se consegue ir de encontro às suas necessidades e garantir um nível mínimo de felicidade, pelo menos no imediato, devido às alterações profundas nas rotinas a que antes estavam habituadas. Por todas estas razões, a tendência é manter os utentes no seu ambiente de conforto o máximo tempo possível; a pandemia só veio fortalecer esta ideia.

Para além disso, os acontecimentos do último ano e meio levam a afirmar que é necessário repensar como é feito o acompanhamento da saúde e reforçar a ideia que se deve apostar na prevenção e na priorização dos meios tecnológicos, em substituição dos tradicionais. No caso específico da COVID-19, verifica-se que, quando ocorre uma infeção, o paciente, ao cumprir quarentena, fica isolado e, por conseguinte, limitado nas visitas e na monitorização do seu estado de saúde. Isto não acontece apenas no domicílio, mas também nas instituições onde os pacientes se encontram. A situação pode ainda agravar-se quando o utente tem de se deslocar às instituições para solucionar algum tipo de problema, pois corre o risco de contrair outras infeções, piorando assim o seu quadro clínico. O objetivo é manter os idosos no seu conforto sempre que possível, e acompanhá-los preventivamente à distância, em situações de isolamento, doença, recuperação ou qualquer outro risco. De qualquer forma, a telemonitorização e o

acompanhamento remoto podem ser aplicados a qualquer pessoa e em qualquer condição clínica, mas de facto os idosos, por todas as razões já apresentadas requerem particular atenção.

A par das crianças (-14 anos), a população idosa (+65 anos) é naturalmente a faixa etária mais vulnerável e menos autónoma na realização das suas atividades diárias, e obviamente a sua falta de autonomia agrava-se quando padece de algum problema de saúde. Olhando para a situação mais recente, e de acordo com um estudo do Barómetro Covid-19 da Escola Nacional de Saúde Pública [28], a idade é um fator de risco para uma pessoa infetada, estando mais suscetível a ter doença grave, necessitar de internamento, cuidados intensivos ou até mesmo morrer. De acordo com o mesmo estudo, uma pessoa entre os 70 e 79 anos tem 10,4 vezes mais probabilidade de ser internada nos cuidados intensivos do que uma pessoa entre os 0 e os 50 anos, e no caso de uma pessoa entre os 80 e os 89 anos, este risco é de 5,7 vezes maior do que na população não idosa (em situação idêntica de vacinação). A idade é também fator de risco acrescido no aparecimento de doenças do foro neuro-cognitivo que são, na sua grande maioria, degenerativas (e.g., *Parkinson*, *Alzheimer*). Por exemplo, a associação Alzheimer Portugal [26], refere que a doença é relativamente rara abaixo dos 60 anos, mas a partir dos 65 a sua incidência duplica e a probabilidade de uma pessoa com mais de 80 anos ter algum tipo de deterioração cognitiva de causa degenerativa é de 30%.

Estes e outros estudos semelhantes leva-nos a concluir o que já se sabe por mera experiência e observação - que a idade é um fator determinante no agravamento de sintomas e no aparecimento de novas doenças quer do foro físico, quer do cognitivo. Assim sendo, pode-se afirmar que o acompanhamento preventivo remoto é prioritário no caso dos idosos, independentemente do seu quadro clínico, pois apesar de poderem não apresentar ainda um estado de saúde preocupante, podem encontrar-se numa qualquer outra condição ocasional ou permanente (e.g., isolamento) que exija atenção por parte de profissionais de saúde e/ou de familiares. Deste modo, utilizando a monitorização remota, o idoso pode viver sozinho, ou até isolado e sem familiares por perto, e simultaneamente, garantir alguma autonomia e segurança, permanecendo na sua residência com o conforto necessário para manter o seu equilíbrio mental e físico, evitando a inconveniência de ter de mudar toda a sua rotina para uma instituição. A telemonitorização permite aos cuidadores verificarem o estado de saúde e/ou bem-estar do idoso, com a frequência que desejarem, mantendo-se eles próprios mais confiantes sobre a qualidade dos cuidados prestados.

Como já foi referido, a telemonitorização não é obviamente aplicável só aos idosos, apesar de, por todas as razões já apresentadas, constituírem o que se acredita ser o público-alvo de eleição para, nos tempos mais próximos, se generalizar o acompanhamento remoto preventivo. Neste momento, já muitas pessoas jovens usam dispositivos que as ajudam a monitorizar o seu estado de saúde, estando ou não doentes. Portanto, é de esperar que as pessoas idosas (que de qualquer forma serão os "jovens" de amanhã) venham, de forma generalizada, a usufruir de dispositivos semelhantes ou de caráter mais clínico para o mesmo fim, podendo ser acompanhadas por familiares e/ou profissionais nessa jornada, para sua própria segurança e tranquilidade.

Em suma, mediante um público diversificado, mas que provavelmente irá abranger maioritariamente idosos e adultos em idades avançadas, ir-se-á com certeza verificar uma crescente adaptação de toda a

sociedade à telessaúde, permitindo a redução da saturação dos serviços de saúde primários em hospitais, lares e outras instituições e, por conseguinte, concedendo aos profissionais de saúde mais tempo de qualidade para poderem prestar mais apoio aos casos efetivamente críticos. Manter os utentes em casa, com acompanhamento remoto, beneficiará todas as partes envolvidas, estado, instituições, profissionais de saúde e familiares, mas sobretudo os próprios, pois a manutenção num ambiente que lhes é à partida mais favorável, psicológica e emocionalmente, estimulará a sua mais rápida recuperação, ao invés de agravar o seu quadro clínico, para condições eventualmente sem retorno nos casos mais sensíveis.

No entanto, atualmente a implementação no terreno de um serviço de telemonitorização pode não ser linear, devido à natural resistência à novidade e à baixa literacia digital, por parte dos utentes (sobretudo dos mais idosos), mas também de familiares e de alguns profissionais de saúde, pelo que é necessário formar e familiarizar utentes e profissionais com este tipo de acompanhamento e, sobretudo, com as ferramentas e aplicações de suporte, para que a sua utilização passe a ser uma *commodity* e a constar das rotinas diárias de ambos. A título de exemplo, espera-se que a medição de sinais vitais decorrentes da telemonitorização ocorra com frequência e dentro da máxima normalidade, pois a partir dela espera-se poder ajudar o doente a prevenir situações mais complicadas e a ganhar mais consciência e autonomia para participar mais ativamente na manutenção da sua saúde e nos próprios processos de tratamento. No entanto, como a telemonitorização não se aplica apenas a quadros clínicos que necessitam de acompanhamento profissional, pelo menos de forma permanente, pode ser apenas usada como forma de acompanhamento preventivo, e nestes casos, os cuidadores informais (sobretudo os mais novos) podem ser os principais agentes de mudança; acredita-se que mais facilmente poderão incentivar os idosos a usar a tecnologia e a mostrar as mais-valias do acompanhamento remoto para lhes garantir mais conforto, segurança e salvaguardar ocorrências que resultem em problemas mais graves.

De acordo com um artigo publicado no *Journal of Medical Internet Research* por Norman e Skinner, literacia pode definir-se, de forma genérica, pela "capacidade de obter, processar e entender a informação e os serviços necessários para tomar as decisões mais adequadas". No caso da saúde digital, ou *eHealth*, esta literacia traduz-se na capacidade de uma pessoa obter, processar e entender informação básica sobre saúde, através da tecnologia [21]. Um estudo realizado por Cruz Miranda Dídia em 2013 [3], onde foram inquiridas 408 pessoas, verificou-se que pessoas com doenças crónicas, com idade avançada e/ou baixa escolaridade, possuem naturalmente menor literacia digital em saúde. Associa-se a este facto o nível de analfabetismo que, segundo os dados dos Censos 2011, ainda abrange cerca de 5,22% da população portuguesa, o que se traduz em cerca de 551 mil portugueses [16]. Segundo o mesmo estudo, a maioria destas pessoas são idosas e vivem no interior, consequentemente mais isoladas. Assim, assumindo que um dos principais objetivos das sociedades evoluídas é garantir apoio e qualidade de saúde a todos, mas sobretudo a estas pessoas mais frágeis, enfrenta-se um grande desafio, pois são as mesmas que apresentam maior grau de analfabetismo, e consequentemente baixa ou nenhuma literacia tecnológica. É, portanto, necessária a contribuição de todos para formar e apoiar a adaptação e inclusão destas pessoas - familiares, amigos, cuidadores, governos, instituições e profissionais de saúde devem cooperar para que estas pessoas se integrem o melhor possível nos novos sistemas de saúde, cada vez mais digitalizados. A

necessidade deste trabalho conjunto, que será necessariamente gradual, é urgente, pois acredita-se que a telessaúde ajudará as pessoas a preservarem durante mais tempo a sua autonomia e a prolongarem a sua qualidade de vida nos seus locais de conforto, evitando mudanças prematuras e, muitas vezes, indesejadas.

No entanto, apesar de as soluções de telemonitorização serem centradas no paciente e no seu bem-estar, também é necessário ter em conta o envolvimento e as dificuldades de quem faz o acompanhamento, ou seja, dos cuidadores formais e informais, onde se enquadram, respetivamente, os profissionais de saúde e os familiares. Estes podem levantar o mesmo problema em termos de literacia digital. Não obstante a maior probabilidade de os utentes terem menos conhecimento tecnológico do que quem está a cuidar deles, também existem casos em que os próprios cuidadores têm dificuldades, pelo que também é necessário formar estas pessoas. Por outro lado, apesar de, naturalmente, os profissionais de saúde apresentarem uma maior literacia em saúde, podem não estar suficientemente "digitalizados", o que implica também a necessidade de alguma formação no sentido de se familiarizarem com as soluções mais atuais de telemedicina e telessaúde, sendo que neste caso se espera um processo de adaptação mais ágil.

Em suma, o processo de criação de um serviço de telemonitorização, deverá ser inicialmente pensado e centrado no doente, pois será ele o principal interessado. Contudo, deve-se considerar com igual atenção o papel dos cuidadores, sobretudo dos informais, pois serão eles em muitos casos, e também no caso particular a desenhar no âmbito desta dissertação, a dar apoio direto ao doente. É necessário, portanto, considerar ambos na especificação e desenvolvimento da solução, tentando antever as possíveis dificuldades que possam vir a ter. Em termos de usabilidade, após criar a solução para estes dois tipos de utilizadores, será mais fácil no futuro estendê-la aos cuidadores formais.

Um serviço de telemonitorização clínica deve conseguir retratar, com a maior precisão possível, o estado de saúde em que se encontra o paciente, de forma a poder prestar auxílio em caso de necessidade e evitar o espoletar de condições agravadas. Para que isto aconteça, deve-se considerar conectar um conjunto de dispositivos que permitam recolher a informação necessária da pessoa e integrá-la de forma automática na solução de telemonitorização. Os dispositivos podem ser de uso próprio ou instalados no ambiente circundante. Com o avanço tecnológico as pessoas usam cada vez mais dispositivos que lhes permitem ter algum controlo sobre a sua vida e a sua saúde, e têm cada vez mais dispositivos inteligentes em casa que lhes possibilitam obter diferentes tipos de informação sobre o que está a acontecer na casa e, por consequência, ter mais segurança. Neste contexto, é interessante cruzar informação proveniente de todos os dispositivos que rodeiam a pessoa, clínicos e não clínicos, para se poder ter um retrato mais holístico da situação.

A título de exemplo, enumeram-se aqui tipos de informação que se podem extrair de um ambiente tecnológico caseiro, rico em dispositivos inteligentes:

- Estado e atividade do residente: em que divisão se encontra, se está a dormir ou acordado, se se encontra em movimento, etc.;

- Consumos energéticos da casa: total ou por dispositivo; a partir destes desta informação também de pode inferir estados de presença, sobre se a pessoa se encontra ou não em casa.
- Informação de portas e/ou janelas abertas ou fechadas: permitindo detetar invasões indesejadas e de forma indireta estados de espírito.

Complementarmente, a partir dispositivos clínicos e/ou usáveis (*wearables*), é possível obter informação sobre os sinais vitais e outras atividades diárias, como por exemplo: número de batimentos cardíacos, número de passos, tempo de sono, peso, temperatura, nível de oxigénio, localização, etc.

Individualmente, por dispositivo, a informação pode ser correta e precisa, mas nunca retrata a situação no seu todo. Só interpretando e combinando dados provenientes de várias fontes permite construir uma visão mais alargada da situação e, por conseguinte, do estado de saúde e bem-estar de uma determinada pessoa; informação proveniente de vários dispositivos, quando devidamente processada e correlacionada, acrescenta valor e conhecimento aos dados coletados por cada sensor, permitindo recomendar e sugerir ações preventivas e corretivas.

Analisando, por exemplo, os coletes que os jogadores de futebol profissionais atualmente usam que recolhem informação GPS e quantificam o esforço desenvolvido durante um jogo ou treino; ao mesmo tempo, permitem analisar a frequência cardíaca, distância percorrida, *sprints*, impactos e mudanças de direção. De acordo com José Soares, fisiologista, estes dispositivos, que na sua forma mais simples são um conta-quilómetros e um conta-rotações, geram informação que após analisada, permite ao treinador e à sua equipa técnica, perceber se os jogadores estão bem e a cumprir com as metas estabelecidas [22]. No entanto, estes dados não devem ser analisados só na vertente, física, técnica ou tática, devem ser vistos no seu conjunto, de forma a capitalizar o conhecimento proveniente da informação recolhida e aplicá-lo no melhoramento da performance dos jogadores. Este é apenas um exemplo, mas na realidade, todos os serviços de telemonitorização têm o mesmo objetivo - melhorar processos, tratamentos e condições físicas e/ou mentais, assim como prevenir e alertar, em caso de agravamento ou emergência. Pelo que ao cruzar informação proveniente de diferentes dispositivos, sejam estes *wearables*, pessoais ou instalados na habitação (e.g., sinais vitais, presença, movimento, localização), é possível realizar uma análise mais profunda e detalhada da situação e, por conseguinte, tomar decisões mais informadas e acertadas.

Contudo, como foi referido anteriormente, a população idosa não tem, no geral, grandes conhecimentos tecnológicos, pode até não possuir dispositivos inteligentes na sua habitação e a simples utilização de um *smartphone* pode ser complicada. Para estes casos, existem dispositivos de fácil utilização (e.g., botão de pânico, cinto, relógio) que permitem recolher alguma informação básica, mas que pode melhorar a segurança de quem vive sozinho e dar mais tranquilidade a quem cuida. Para além disso, devido à sua simplicidade permitem também suavizar a curva de aprendizagem e adaptação das pessoas às novas tecnologias.

Por estas razões, optou-se por criar uma solução de telemonitorização que permita acompanhar pessoas à distância, recolhendo essencialmente informação de localização e, dependendo do dispositivo, sinais vitais básicos. Apesar de simples, esta solução deve abranger uma gama de funcionalidades que

permita a cuidadores informais (e.g., familiares) acompanhar pessoas idosas ou com alguma fragilidade física ou mental, pouco digitalizadas, que vivam isoladas ou que passem bastante tempo sozinhas. Assim, este trabalho tem em vista a criação de serviços focados na deteção de localização e na identificação de situações de risco daí decorrentes. Destinam-se a idosos, sobretudo aos que sofram de doenças físicas algo restritivas ou doenças mentais degenerativas leves. Nos casos em que exibam algumas dificuldades de locomoção ou comportamentos erráticos, é esperado que a solução a desenvolver possibilite, com base na informação recolhida, fazer uma "vigilância" eficiente, podendo assim minimizar o efeito negativo em situações de risco como quedas, desaparecimentos, desorientações, pânico, etc.

O conceito *Confort@Home* abrange cenários complementares, *indoor* e *outdoor*.

No caso *indoor*, consideraram-se sensores não invasivos, de simples instalação na residência para fornecer informação complementar sobre o idoso. Pretende-se desta forma, monitorizar a sua atividade para distinguir situações que se encontram dentro do normal/habitual, de eventos que correspondem a desvios ao considerado normal e que podem ser considerados de risco. Os dispositivos a considerar podem ser: lâmpadas e/ou tomadas inteligentes que para além de cumprirem a sua função e permitirem verificar consumos e frequência de utilização, poderão também ser usados para inferir estados de presença e ausência; fechaduras inteligentes poderão ser usadas também para reforçar este tipo de inferência, para além executarem a função de deteção de intrusões; detetores de movimento que podem ser instalados em várias divisões da casa. Combinando informação proveniente de todos estes dispositivos será possível ter uma ideia de diferentes situações de potencial risco, como por exemplo se não se detetar movimento no quarto ao acordar e deitar, se a lâmpada da mesinha de cabeceira não for ligada antes de dormir, etc. Se se basear esta análise em rotinas, será ainda mais precisa a inferência do risco.

No cenário *outdoor*, usar-se-ão essencialmente dispositivos equipados com GPS para monitorizar a localização do utente no exterior. Entre eles, serão considerados relógios, cintos e botões. O familiar ou o próprio utente poderá adquirir o que for lhe for mais conveniente, pois para além da localização poderão dar informação adicional, como por exemplo sinais vitais básicos (no caso do relógio). Podem também ser considerados outros dispositivos, como por exemplo, o de deteção de quedas, que pode ser usado quer no interior quer no exterior da casa. Este dispositivo em particular, permite detetar uma queda e espoletar automaticamente uma chamada para números de emergência previamente configurados. Caso não aconteça de forma automática, é possível manualmente iniciar a chamada pressionando o botão de que o dispositivo dispõe. Dispositivos com GPS permitem distinguir lugares e trajetos de visita comum, de lugares e trajetos estranhos e, em consequência, agir em situações de risco. Em casos de utentes, por exemplo, com demência ou outro tipo de patologia do género que perdem a orientação por várias horas e por vezes têm dificuldade em encontrar o caminho de regresso a casa, este tipo de dispositivos pode ser muito útil. Para aumentar a segurança do utente, ir-se-á criar uma cerca virtual, em que, a partir do momento em que é detetado que saiu da zona habitual ou da zona definida como permitida/segura para circular, é ativado um aviso e enviado para o cuidador. Dado o tempo previsto da dissertação, este será o principal foco do trabalho de desenvolvimento a realizar.

2.2 Serviços de Telemonitorização

A telemonitorização ajuda os pacientes a sentirem-se mais acompanhados, seguros, e ao mesmo tempo, mais autónomos e participativos no seu próprio processo de tratamento. Quanto aos cuidadores, ajuda-os a prestar melhores cuidados e a sentirem-se mais produtivos e confiantes, pois podem estar remotamente em qualquer lugar a seguir os seus doentes, sendo notificados só nos casos mais urgentes.

No entanto, para obter real entendimento sobre o estado de saúde de um paciente, é necessário selecionar um conjunto relevante de medições/indicadores que permitam tirar conclusões acertadas, assim como criar algoritmos que ajudem a obter conhecimento adicional, criando correlações, inferências e recomendações adequadas [20]. Para que tal aconteça, existe na base da telemonitorização clínica a recolha de dados, em particular de sinais vitais que, de forma direta ou indireta, avaliam as funções básicas do corpo humano e providenciam informação crucial sobre o bem-estar geral. Estes dados permitem aos profissionais avaliar o estado de saúde do paciente e tirar conclusões sobre algumas disfunções e/ou doenças [20]. Conseguem, deste modo, aconselhar e avançar para a realização de exames, se necessário, assim como sugerir tratamentos ou alterações ao estilo de vida. Os sinais vitais mais comuns e abrangentes, ou seja, que permitem despistar um maior número de situações anómalas são: a temperatura corporal, a pressão arterial, o ritmo cardíaco e/ou pulsação, e complementarmente, o peso e a altura.

`Temperatura Corporal`

A temperatura normal do corpo humano não varia muito. Ainda assim, varia conforme o sexo, a atividade física, a comida, os fluídos ingeridos e, no caso das mulheres, com o ciclo menstrual. De acordo com o SNS, os valores de temperatura corporal são considerados normais quando se encontram entre 36°C e 37,5°C [31].

A temperatura corporal é controlada pela região do cérebro chamada hipotálamo; quando há um vírus, por exemplo, o sistema imunitário envia informação para esta parte do cérebro, com o intuito de aumentar a temperatura corporal e enfraquecer o vírus [31]. A temperatura corporal é considerada anormal quando se tem febre (alta temperatura), ou seja, acima dos 37,5°C, ou hipotermia (baixa temperatura), abaixo dos 36°C. A variação da temperatura corporal de forma anómala e inesperada, é motivo de preocupação, pelo que a medição regular deste sinal vital é crucial, pois permite tomar ações preventivas e/ou tratar a condição em causa.

`Pressão Arterial`

A pressão arterial é a força que o sangue exerce sobre as paredes das artérias durante a sua circulação [32]. Cada vez que o coração bate, é bombeado sangue para as artérias, resultando num aumento da pressão à medida que o coração contrai e numa redução à medida que relaxa. A pressão é calculada em duas medidas, sistólica (ou "máxima") que se refere à pressão na artéria quando o coração contrai e bombeia sangue para o resto do corpo e a diastólica (ou "mínima") que diz respeito à pressão na artéria quando o coração relaxa.

Genericamente, a pressão arterial pode ser considerada normal quando os valores sistólica/diastólica se encontram abaixo de 120/80, mas a avaliação mais fina depende de outros fatores, como a idade e condições clínicas associadas. De qualquer forma, considera-se hipertensão no Estado 1 quando a sistólica se encontra entre 130 e 139 e a diastólica entre 80 e 89, e no Estado 2 quando se encontram acima de 140/90, respetivamente. A unidade de medida da pressão arterial, tanto na sistólica como na diastólica, é mmHg (milímetros de mercúrio) e corresponde objetivamente à altura de uma coluna de mercúrio de um milímetro num dispositivo chamado manómetro, que é elevada pela pressão do sangue.

Os valores da pressão arterial podem variar durante o dia devido a esforços físicos ou emocionais e com o avançar da idade tendem a aumentar. De qualquer forma, é necessário monitorizá-los, pois quando estes atingem valores que mediante um padrão verificado anteriormente são considerados anómalos, podem indicar algum problema de coração ou mesmo iminência de ataque cardíaco. A monitorização regular da pressão arterial permite avaliar se é necessário fazer ajustes ao estilo de vida e/ou prosseguir para tratamento médico/medicação.

Ritmo Cardíaco

O ritmo ou frequência cardíaca é a velocidade do ciclo cardíaco, medido pelo número de contrações do coração por minuto (BPM). O valor pode variar de acordo com as necessidades físicas do organismo, incluindo a necessidade de absorção de oxigénio e de excreção de dióxido de carbono. É normalmente igual ou próxima da pulsação arterial, medida em qualquer ponto periférico. Pode ser alterada pelo exercício físico, sono, ansiedade, stress, doença ou ingestão de estupefacientes.

O valor considerado normal em adultos saudáveis varia entre 60 e 100 batidas por minuto, em repouso. Durante o sono, a frequência cardíaca desce para valores entre 40 e 50 batidas por minuto. Taquicardia corresponde a uma alta frequência cardíaca definida como acima de 100 batidas por minuto e arritmia é quando o coração não bate a uma frequência regular. Anomalias na frequência cardíaca geralmente indicam doença. A medição do ritmo cardíaco pode ser efetuada manualmente, recorrendo a dois dedos no pulso ou pescoço, ou usando dispositivos como oxímetros ou relógios que em poucos segundos indicam o valor recolhido. Deve ser medido regularmente, pois valores fora do normal podem implicar medidas clínicas para evitar situações mais sérias de doenças cardiovasculares ou outro tipo de problemas do foro cardíaco.

Peso e Altura

Apesar de não serem considerados sinais vitais, como os anteriormente descritos, são os indicadores físicos que complementarmente permitem avaliar o estado de saúde de uma pessoa. Variações bruscas indicam normalmente problemas sérios de saúde. Uma variação na altura pode indicar perda de densidade óssea e aumentam o risco de desenvolver osteoporose com o avançar da idade. O aumento ou diminuição do peso pode indicar uma vasta gama de problemas médicos subjacentes (como a doença da tiroide) a maus hábitos de vida. Saber o peso e altura de um paciente, em comparação com os valores normais para a idade, permite calcular outros indicadores (e.g., massa corporal) e ajuda a tomar decisões

mais acertadas relativas ao seu bem-estar.

De uma forma geral, pode-se verificar que todos estes valores são úteis para indicar se há alguma anomalia no estado de saúde do paciente. Ao efetuar uma monitorização regular destas medidas, pode-se determinar uma média, dentro da qual os valores para aquele paciente podem ser considerados normais, e delinear limites para quando estes valores forem ultrapassados. Deste modo, pacientes e cuidadores serão notificados quando alguma coisa não estiver bem e podem agir em conformidade de forma a evitar o agravamento do quadro clínico. Apesar de um só sinal vital poder per si dar bastante informação sobre o estado de saúde, cruzando e correlacionando várias medições obtém-se um quadro mais completo e preciso, até porque muitos dos sinais acima mencionados estão relacionados entre si, como é o caso da pressão arterial e do ritmo cardíaco - por exemplo, quando o ritmo cardíaco aumenta a pressão arterial também tende a aumentar.

O mercado dos serviços de telemonitorização está ainda em fase embrionária, o que se deve essencialmente a alguma inércia por parte das instituições, falta de pessoal especializado e alocação de tempo para treino. Com o aparecimento da pandemia e a consequente necessidade de prestar serviços de saúde remotamente, tem-se verificado uma crescente utilização da telessaúde, nomeadamente de teleconsulta e da telemonitorização. Atualmente, em Portugal, já existem alguns projetos em curso nesta área, como por exemplo, o projeto de telemonitorização do Hospital Distrital de Santarém (HDS), o do Centro Hospitalar Universitário Cova da Beira (CHUCB), o do Serviços Partilhados do Ministério da Saúde/Centro Nacional do TekSaúde (SPMS/CNTS) e o da Altice Portugal com o Centro Hospitalar Universitário Lisboa Central (CHULC). Estes são alguns dos mencionados publicamente, mas existem muitos mais a iniciar ou já em curso.

Projeto do Hospital Distrital de Santarém (HDS)

O projeto do HDS foi iniciado pelo serviço de pneumologia e consiste num serviço de telemonitorização domiciliária prestado a casos de doença pulmonar obstrutiva crónica (DPOC) [30].À semelhança do que já foi explicado como sendo a base de um serviço de telemonitorização, este consiste no acompanhamento remoto de pacientes quando estes se encontram no conforto da sua casa. De acordo com a notícia divulgada pelo SNS, este projeto está a ser testado em cinco pacientes, sendo recolhidos parâmetros fisiológicos, designadamente a frequência cardíaca, a saturação de oxigénio, a temperatura e o número de passos que o doente dá por dia. Os valores são registados pelo paciente e supervisionados pelos profissionais de saúde. Para facilitar, existe um conjunto de limites que geram alertas caso sejam ultrapassados, e em consequência o utente é contactado.

Projeto do Centro Hospitalar Universitário Cova da Beira (CHUCB)

Este projeto, denominado Telemonitorização de Doentes em Risco - TERI, é uma solução focada em pessoas clinicamente consideradas em situação de risco, quer se encontrem em regime de urgência ou internamento, ou, em hospitalização domiciliária. Tal é alcançado através da monitorização constante de

sinais vitais e da implementação de sistemas de alerta em caso de descompensação, o que permitirá tornar os processos mais eficientes e a atuação das equipas mais célere, para maior conveniência e segurança dos utentes [1].

`Serviço da SPMS/CNTS`

Esta solução já se encontra disponível no mercado para utilização dos utentes. Trata-se de uma solução de telemonitorização que pretende dar resposta às necessidades de acompanhamento digital dos utentes do Serviço Nacional de Saúde, quando estes não se encontram nas unidades de saúde. Denominada *Telemonit SNS 24*, a aplicação móvel permite aos utentes aceder ao um plano de monitorização clínica proposto por um profissional de saúde, podendo registar sinais vitais ou outras medições biométricas, adicionados manualmente ou através de equipamentos ligados ao telemóvel. É uma solução indicada para situações de pós-consulta, alta de internamento, alta de um episódio de urgência ou para qualquer outro tipo de situação resultante de contacto com um profissional de saúde. Neste momento, está apenas disponível para utentes com insuficiência cardíaca congestiva, doença pulmonar obstrutiva crónica e recuperados pós-COVID-19. Esta aplicação está limitada à recomendação de um profissional de saúde, pelo que não é permitida a sua utilização de forma individual ou por cuidadores informais [33].

`Smart AL`

SmartAL é a solução desenvolvida pela *Altice Labs* com vista a telemonitorizar pessoas que necessitem de acompanhamento remoto por parte de profissionais de saúde e/ou cuidadores informais. O serviço visa simplificar a vida de ambos, quer do ponto de vista da saúde quer socialmente.

O *SmartAL* oferece um conjunto de tecnologias, dispositivos e serviços de apoio, que permitem acompanhar, em tempo real, doentes crónicos, idosos e pessoas em convalescença, hospitalização domiciliária, confinamento, pós-internamento, etc., pois permite a telemonitorização de sinais vitais, teleconsultas e suporte de outras atividades relacionadas com a saúde, o bem-estar e a segurança. É uma solução personalizável e adaptável às necessidades de cada cliente, permitindo ao utilizador final usufruir de uma maior independência, autonomia e dignidade, fazendo-o sentir-se seguro dentro e fora de casa. Para além de permitir a monitorização de diferentes sinais vitais provenientes de dispositivos clínicos e/ou *wearables*, torna possível cruzar estes dados com outros complementares recolhidos através de ficheiros, imagens ou mesmo dispositivos de IoT/domótica.

É uma solução disponível em diferentes tipos de interfaces (PC, Tablet, Smartphone, TV, etc.). A aplicação TV (possível disponibilizar a clientes MEO) permite a utentes pouco digitalizados e que não disponham de outro tipo de interfaces aceder de forma simples ao serviço, recolher medições, receber alertas e lembretes e ter acesso a questionários e vídeos. A aplicação móvel, com funções mais evoluídas, está disponível para vários perfis (utente, cuidador formal, informal, administrativo, etc.) e permite recolher dados de vários dispositivos, enviá-los para a *Cloud* para cruzar com limites pré-definidos e disparar notificações e alarmes, sempre que se justificar. Por último, a aplicação *web* que é naturalmente a mais completa está também disponível para vários perfis, mas é mais usada essencialmente pelos cuidadores

formais, administrativos e administradores, pois permite-lhes todo tipo de ações, como por exemplo: criar instituições, adicionar utilizadores, definir tarefas e planos de medição, incluir novos questionários e vídeos e, no caso dos profissionais de saúde, ter acesso ao mais importante, ou seja à informação médica do utente (doenças, alergias, prescrições, exames) e à monitorização em tempo real de toda a informação recebida.

Em suma, das várias funcionalidades disponíveis no *SmartAL* destacam-se então as seguintes:

- Gestão de atividades do dia-a-dia;
- Agendamento de tarefas e planos;
- Monitorização dos sinais vitais;
- Alertas, notificações e lembretes;
- Questionários para avaliação social e de saúde;
- Teleconsulta, chat e transferência de ficheiros;
- Tutoriais em vídeo;
- Relatórios estatísticos;
- Automação da casa;
- Proteção de dados pessoais e informação sensível.

O *SmartAL* é um serviço dirigido tanto a idosos como a crianças; na realidade a qualquer pessoa que necessite de cuidados no seu dia-a-dia, seja por motivo de doença crónica, temporária ou qualquer outro que a coloque em situação de fragilidade (e.g., idade, isolamento, confinamento). É indicado para instituições do tipo lar, hospital, casa de repouso, etc., mas também pode ser usado num contexto mais abrangente, como por exemplo ao serviço de câmaras municipais ou no âmbito do serviço nacional de saúde [18]. O produto disponível no mercado segue um modelo *B2B* e *B2B2C*, ou seja, é vendido às instituições que depois disponibilizam aos seus utentes conforme as suas necessidades.

2.3 Serviços de Localização

Os serviços de localização GPS já estão estabelecidos no mercado há vários anos, com permissão para uso civil desde 1995. Os primeiros telemóveis com GPS surgiram em 1999 e, com este serviço permitem fornecer informações sobre a localização exata de pontos de interesse, através do uso de satélites [5].

Desde então, a tecnologia GPS tem-se tornado cada vez mais barata, passando por um processo rápido de evolução. Começou com a criação de dispositivos GPS em veículos para orientação dos condutores, e a partir daí foi aumentando a sua utilização até à situação atual, em que a tecnologia é usada

em praticamente todos os dispositivos eletrónicos, pois muitas das aplicações comummente utilizadas necessitam de informação da localização para garantirem o seu funcionamento. A evolução da tecnologia GPS resultou numa variedade de utilização em diversas áreas: comunicações, segurança, agricultura, transportes etc. Atualmente, existem inúmeros serviços que permitem localizar animais, pessoas ou objetos, através do uso de dispositivos equipados com GPS. Também na área da saúde e do bem-estar existem dispositivos GPS que permitem localizar pacientes em tempo real e acompanhar à distância o seu desempenho, dando-lhes mais segurança e evitando, se possível, visitas desnecessárias às instituições de saúde. O objetivo é ajudar a prevenir situações de risco e prestar auxílio com a maior brevidade possível. Através da utilização de serviços de localização, podem-se desbloquear uma série de problemas e prever determinadas situações desagradáveis, tendo em conta a possibilidade de estabelecer de limites e regras, assim como analisar as rotinas e os percursos executados que permitam distinguir anomalias da atividade considerada regular.

Para tornar o acompanhamento mais eficiente, pode-se cruzar a localização com dados complementares obtidos pelos dispositivos GPS, ou por outros quaisquer dispositivos já instalados em casa ou utilizados pelo paciente e que forneçam informação útil, como por exemplo os a atividade e os sinais vitais (obtidos através de dispositivos clínicos ou *wearables*); contudo, por constrangimentos temporais não se irá analisar esse cruzamento de informação no trabalho de implementação. No entanto, quanto mais informação e conhecimento efetivo se obtiver do utilizador, mais rapidamente será possível ao familiar/cuidador informal chegar a uma conclusão sobre o está a acontecer, em caso de risco. Por exemplo, o paciente utiliza um *smartwatch* que fornece periodicamente sinais vitais (batimentos cardíacos, saturação do oxigénio, etc.), mas o cuidador constatou que está há bastante tempo sem receber este tipo de dados. Ao verificar a localização do seu paciente, percebe que este se encontra a caminhar dentro de uma zona segura e que tem na aplicação uma notificação sobre falta de bateria do dispositivo; contacta o paciente que rapidamente compra novas baterias, ou pede-lhe para recolher a casa e aguardar pela sua visita. Do ponto de vista de segurança e do bem-estar, todo o tipo de informação pode ser útil para proteger o paciente, mesmo noutros cenários mais caseiros. Por exemplo, em situações em que o familiar/cuidador saiba que o paciente vive sozinho e está a dormir (por informação proveniente de um relógio, por exemplo), mas através dos sensores espalhados pela casa é detetado movimento, e/ou que uma porta ou uma janela se abriu, pode ligar ao paciente para verificar que está tudo bem, ou optar por rapidamente contactar a segurança do prédio por se poder tratar de uma invasão. Estes são apenas alguns exemplos da aplicação e importância de recolher informação complementar à de saúde, e da necessidade de a cruzar com outros dados para se poder tirar conclusões mais precisas e salvaguardar a segurança daqueles que são mais vulneráveis.

3

Modelação e Arquitetura

O desenvolvimento da solução atravessou várias etapas, sendo que, a primeira e a mais importante foi definir o que se pretendia alcançar, quais objetivos e a quem se destinava. Esta fase é crucial, pois permite definir bem o propósito do trabalho, balizar o esforço e antecipar alguns problemas. De seguida, foram identificados os requisitos, com base em casos de uso e *storyboards*, para uma melhor compreensão da solução. Por último, definiu-se a arquitetura funcional e o correspondente modelo de dados, e com a ajuda de diagramas de sequência detalhou-se a lógica dos serviços.

3.1 Descrição da Solução

A solução desenvolvida, denominada aqui por *Teleloc*, consiste numa ferramenta de acompanhamento remoto que explora cenários de telelocalização, com base em informação proveniente de dispositivos GPS. A aplicação possibilita ao utilizador (neste caso a um cuidador informal) visualizar a localização em tempo real dos seus dependentes (e.g., pais), criar cercas virtuais e adicionar locais considerados seguros que se traduzem em locais que o dependente visita com alguma regularidade. Todas estas funcionalidades devem ser configuráveis pelo utilizador, de modo a tirar o melhor proveito possível da aplicação e ajustá-la ao seu caso específico. Para usufruir das referidas funcionalidades, os utilizadores têm de efetuar o registo na plataforma, realizando uma subscrição e adquirir um dispositivo compatível com a solução. Um utilizador não autenticado apenas tem acesso à opção de autenticação ou registo, caso não possua ainda uma conta e pretenda telemonitorizar alguém que tenha ao seu cuidado.

Genericamente, uma aplicação de telemonitorização serve para monitorizar uma ou várias pessoas à distância, utilizando um equipamento de monitorização (e.g., telemóvel, computador) e um dispositivo que emita informação sobre a pessoa a ser monitorizada (e.g., *smartwatch*, botão). No caso particular da aplicação *Teleloc*, não será a pessoa que detém a conta o "monitorizado", mas sim o cuidador. Portanto, a solução deve permitir ao utilizador/cuidador adicionar "dependentes", ou seja as pessoas a monitorizar, assim como informações consideradas necessárias relativamente a essas mesmas pessoas. Para além disso, a solução deve possibilitar o registo de dispositivos e a associação destes aos seus dependentes, de forma a identificar univocamente a quem pertence a informação obtida.

Como se trata de uma aplicação de telelocalização, a mesma deve dar a possibilidade ao utilizador/cuidador de visualizar a localização dos seus dependentes, a partir da informação proveniente dos dispositivos GPS (ou de qualquer outro sistema de localização). Esta deve ser fornecida em tempo real ou num curto intervalo de tempo, de forma a poder analisar a informação o mais rapidamente possível e evitar situações mais graves de risco. Para tal, a aplicação deve contar com uma funcionalidade que exiba um mapa e indique a localização dos diferentes dispositivos. Para além da localização em tempo real, a aplicação deve permitir ao utilizador definir um intervalo de tempo e verificar o histórico das localizações dos dispositivos, para verificar se ocorreu alguma anomalia durante aquele período.

Quando um utilizador recorre a este tipo de soluções pressupõe-se que pretende telemonitorizar uma pessoa com algum tipo de fragilidade física e/ou cognitiva. No entanto, ver em tempo real a localização pode não ser suficiente nem muito prático, pois o utilizador não vai estar constantemente com a aplicação aberta para poder verificar se o dependente está em lugares de seguros ou de risco. Como tal, é importante que a aplicação disponibilize a criação de cercas virtuais e lugares seguros. Relativamente às cercas virtuais, deve ser permitido ao utilizador desenhar no mapa áreas de conforto, usando formas geométricas (e.g., círculos, retângulos, polígonos), e atribuir uma ou mais a cada um dos seus dependentes. Assim, cada vez que o mesmo saia dessa área delimitada, será espoletado um aviso para o cuidador, podendo ele mesmo confirmar no mapa a irregularidade detetada. A mesma lógica deve ser usada para a criação de lugares seguros. No entanto, em vez de ser delimitado a uma área, é assinalada uma localização específica no mapa. Normalmente assinala-se os locais frequentados pelo dependente com alguma regularidade (e.g., café, farmácia), e assim o cuidador será avisado cada vez que o dependente entre ou saia de um destes locais ou permaneça mais tempo em qualquer outro lugar não considerado de "conforto". A disponibilização deste tipo de funcionalidades traz a vantagem de permitir ao cuidador ser informado sobre situações possivelmente anómalas, sem que tenha de estar permanentemente em vigilância na aplicação. Em qualquer altura pode apagar ou registar novos lugares seguros.

Para além de notificações associadas às cercas e aos locais seguros, existem também alarmes ativados diretamente pelos dispositivos (i.e., quedas ou pressionando o botão de SOS) e outros complementares de uso frequente (e.g., falta de bateria). É necessário que a aplicação disponibilize notificações e alarmes com diferentes níveis de severidade, de forma a informar o cuidador com a maior brevidade possível sobre o nível de segurança e bem-estar dos seus dependentes. Assim, sempre que um dispositivo fique sem bateria, seja pressionado o botão de SOS, seja detetada uma queda, uma saída de uma cerca virtual ou de um lugar seguro, será espoletada uma notificação no sistema, e o cuidador será informado do caso em questão, associando-se uma cor à notificação que varia consoante a severidade. Deve também ser possível ao cuidador verificar o histórico de notificações de cada dependente.

Para todas estas funcionalidades da aplicação funcionarem com a maior eficácia possível, a informação dos intervenientes deve estar o mais atualizada possível. Para tal, deve ser permitido ao cuidador a atualização da sua informação pessoal, assim como a dos dependentes e a dos dispositivos associados. Poderá também atualizar as condições da sua subscrição e as definições de segurança da sua conta.

3.2 Casos de Uso e Story Boards

Os casos de uso servem para capturar, modelar e especificar os requisitos de um sistema [2]. Por sua vez, um requisito é uma funcionalidade ou característica considerada relevante na ótica do utilizador e representa o comportamento esperado do sistema [23]. Os casos de uso são normalmente especificados em texto, onde é descrita a função do caso em questão. No entanto, por vezes são também apresentados sob a forma de diagramas em que estão representados vários casos de uso, conjuntamente com os diferentes atores.

De acordo com a solução descrita anteriormente, foram criados casos de uso para especificar os requisitos da aplicação. Cada caso de uso possui na sua estrutura um título, o público-alvo/pré-requisitos, uma breve descrição onde é dado o contexto da funcionalidade a especificar, e por fim um cenário que representa uma situação real.

3.2.1 Especificação de Casos de Uso

3.2.1.1 Macro Use Case

Tabela 1: Teleloc

Público-Alvo	Qualquer pessoa com idosos (ou pessoas com outras limitações) ao seu cuidado
Descrição	Em geral, os mais idosos, como qualquer outra pessoa se não gravemente doente, gostam de se deslocar, sair para passear e visitar os seus locais preferidos, para fazer compras, almoçar, ou simplesmente conviver com os seus vizinhos e amigos. Normalmente, isto não é motivo de preocupação, mas com o avançar da idade, as pessoas desenvolvem alguns problemas e enfermidades, tanto físicas como mentais, que podem limitar o seu desempenho habitual e pôr em risco a sua saúde. Nestes casos, a adesão a um serviço de telemonitorização pode ser a solução ideal para dar segurança e paz de espírito, tanto aos mais velhos como aos seus cuidadores.
Cenário	Rute tem 77 anos, é viúva e sofre de *Alzheimer*. Vive com a filha mais velha Sara, o seu marido e os dois filhos. No entanto, passa a maior parte do dia sozinha. Foi diagnosticada com a doença há cerca de 2 anos, e apesar de se lembrar da maior parte das coisas, começa a ter episódios cada vez mais frequentes; perde-se quando vai ao minimercado que fica no final da rua, ou simplesmente esquece-se do que ia fazer. A Sara, para se sentir mais descansada enquanto está ausente no trabalho, decide aderir a um serviço de telemonitorização que possua funcionalidades de localização para, durante o dia, poder ocasionalmente verificar o estado da sua mãe.

3.2.1.2 Micro Use Case - Criação de conta e Plano de Subscrição

Tabela 2: Criação de conta e Plano de Subscrição

Target	Cuidadores
Descrição	Qualquer indivíduo tem a possibilidade de aceder à aplicação, criando uma conta e preenchendo os dados necessários. Depois, pode subscrever o serviço desejado (neste caso o *Teleloc*) e escolher o seu plano. O cuidador escolhe o plano de subscrição mediante o número de dependentes que quer monitorizar e compra os dispositivos. A título de exemplo pensou-se em dois planos: o plano *Standard* que incorpora entre 1 a 2 dependentes e o *Plus* que oferece a possibilidade de incorporar mais dependentes (3 a 5). O valor de cada plano poderá ser debitado mensalmente ou anualmente.

3.2.1.3 Micro Use Case - Adicionar um dependente

Tabela 3: Adicionar um dependente

Pré-Requisitos	Cuidadores que criaram conta na plataforma
Descrição	Após devidamente autenticado na plataforma o cuidador pode iniciar a configuração da sua conta, começando por adicionar os dependentes que vai querer monitorizar durante o tempo que subscrever o produto.
Cenário	Após se autenticar na aplicação a Sara vai adicionar as informações da sua mãe, Rute, de forma a poder iniciar o processo de telemonitorização.

3.2.1.4 Micro Use Case - Atribuir um dispositivo a um dependente

Tabela 4: Atribuir um dispositivo a um dependente

Pré-Requisito	Cuidadores que criaram conta na plataforma, subscreveram um plano e compraram os dispositivos
Descrição	Quando um novo dispositivo é adicionado à aplicação, é possível atribuí-lo a um dependente específico. O cuidador pode posteriormente atribuí-lo a outro dependente ou até eliminá-lo da sua conta.
Cenário	A Sara compra um dispositivo compatível com a solução e adiciona os dados do mesmo na aplicação. Após isso, atribui o dispositivo à sua mãe, de forma a obter toda a informação que este recolhe e poder visualizá-la na interface como relativa à Rute.

3.2.1.5 Micro Use Case - Monitorização da Localização em Ambiente Exterior

Tabela 5: Monitorização da Localização em Ambiente Exterior

Pré-Requisitos	Cuidadores que criaram conta na plataforma, subscreveram um plano, compraram os dispositivos, registaram-nos na *App* e associaram-nos aos dependentes
Descrição	O cuidador tem a possibilidade de poder visualizar num mapa os dependentes associados à sua conta simultaneamente e em tempo real. No entanto, caso pretenda ver apenas um dependente basta clicar em cima da sua foto e é automaticamente redirecionado para a sua localização. A localização exterior pode ser detetada por qualquer dispositivo com GPS/*Wi-fi*, de que são exemplo *smartbands, smartwatches*, cintos e botões. Por exemplo, um cinto com um sensor GPS pode ser muito útil e não tão intrusivo para localizar um idoso do sexo masculino com alguns pequenos problemas de desorientação ou outros problemas de mobilidade. Utilizando GPS e/ou triangulação *Wi-fi*, estes tipos de dispositivos permitem identificar e visualizar com melhor precisão os percursos e locais que a pessoa costuma visitar.
Cenário	Após configurar a aplicação (adicionar dependentes e dispositivos associados a esses dependentes) a Sara pode aceder à aplicação a qualquer hora do dia e verificar a localização dos dispositivos que adicionou. A Sara vai trabalhar todo os dias e por norma almoça sempre no trabalho. Durante a sua pausa de almoço aproveita para visitar a aplicação e verificar se a sua mãe se encontra em casa ou se foi dar uma pequena volta pela vizinhança.

3.2.1.6 Micro Use Case - Criar e/ou editar Cercas Virtuais

Tabela 6: Criar e/ou editar Cercas Virtuais

Pré-Requisitos	Cuidadores que criaram conta na plataforma, subscreveram um plano, compraram os dispositivos, registaram-nos na *App* e associaram-nos aos dependentes
Descrição	O cuidador tem a possibilidade de criar várias cercas virtuais para cada dependente. A cada cerca criada é atribuída a uma localização específica e será apresentada ao utilizador com a forma geométrica pretendida (círculo, quadrado ou polígono). Após criada fica automaticamente ativa. Quando ativa, são enviadas notificações ao cuidador, sempre que o respetivo dependente sair das imediações dessa cerca. É também possível editar uma cerca existente (local, nome e forma geométrica) ou até mesmo eliminá-la.
Cenário	O Sr. Carlos sofre de demência leve e passa grande parte do seu tempo sozinho em casa. Um dia precisou de lenços de papel porque estava com gripe. Por isso, decidiu ir ao minimercado perto de casa. Assim que deixou a zona designada como ”casa”, o seu filho Rafael recebeu um alerta de saída da sua zona de “conforto”. Mas antes de entrar em contacto com o pai, monitorizou o seu movimento e percebeu que se dirigia para o minimercado (registado também como lugar seguro). Assim que comprou o que precisava, o Sr. Carlos voltou para casa. O filho ficou aliviado porque percebeu que estava a regressar a casa e que tudo estava bem, tendo posteriormente recebido um alerta a indicar que o pai tinha voltado a entrar na zona ”casa”.

3.2.1.7 Micro Use Case - Criar e/ou editar Lugares Seguros

Tabela 7: Criar e/ou editar Lugares Seguros

Pré-Requisitos	Cuidadores que criaram conta na plataforma, subscreveram um plano, compraram os dispositivos, registaram-nos na *App* e associaram-nos aos dependentes
Descrição	O cuidador tem a possibilidade de atribuir locais seguros a cada dependente. Cada local seguro está designado por uma localização específica, ficando ativo quando criado, sendo possível desativar posteriormente. Quando ativo, são enviadas notificações ao utilizador assim que o dependente atribuído se encontrar no local seguro. Por fim, é possível editar um local existente ou até mesmo eliminá-lo.
Cenário	O Sr. Carlos sofre de demência leve e passa grande parte do seu tempo sozinho em casa. Um dia, decidiu ir ao café ter com os seus amigos. O seu filho estava no trabalho e não viu imediatamente a notificação de saída da cerca virtual que tinha definida para a casa do pai. Entretanto o Sr. Carlos chegou ao café e esta localização estava definida como lugar seguro, pois costuma ir ao café todas as tardes encontrar-se com os seus amigos. Nesta altura o filho já estava com mais disponibilidade e recebeu a notificação que o pai estava num lugar seguro por isso continuou o seu trabalho normalmente. No final da tarde o Sr. Carlos decidiu regressar a casa, pelo que o sistema alertou o filho que ele tinha saído do lugar seguro, passando a poder monitorizar o seu percurso até casa.

3.2.1.8 Micro Use Case - Deteção de Queda/SOS

Tabela 8: Deteção de Queda/SOS

Pré-Requisitos	Cuidadores que criaram conta na plataforma, subscreveram um plano, compraram os dispositivos, registaram-nos na *App* e associaram-nos aos dependentes
Descrição	As quedas podem ser detetadas por botões/sensores. Estes dispositivos, são capazes de detetar quando ocorre uma queda e desencadear chamadas de emergência para números pré-configurados. Além disso, se, por qualquer alteração, o equipamento estiver a funcionar mal e não conseguir detetar automaticamente a queda, o utilizador pode sempre enviar um sinal de emergência premindo o botão SOS. Caso nenhum número esteja configurado, o dispositivo emite um sinal que é enviado diretamente para a aplicação e apresentado sobre a forma de notificação, de forma a informar o seu cuidador da ocorrência.
Cenários	1. O Sr. Joaquim gosta de caminhar durante a tarde e normalmente caminha cerca de 7 km por dia. Um dia, foi dar um passeio e tropeçou numa rocha, acabando por cair. A sua filha Maria logo recebeu um alerta de que o seu pai tinha caído, e foi à aplicação para descobrir onde ele estava, utilizando o GPS incorporado no aparelho. Graças a isso, ela conseguiu encontrá-lo e enviar ajuda para o local. 2. A Sra. Rita, tem 79 anos e decidiu sair para fazer compras no mini-mercado perto da sua casa. Ao regressar a casa, ela escorregou e acabou por cair no chão. Por alguma razão, o detetor de queda não estava a funcionar corretamente, e não contactou a sua filha. Assim que se apercebeu do mau funcionamento, a Sra. Rita premiu o Botão SOS e contactou a sua filha.

3.2.1.9 Micro Use Case - Notificações

Tabela 9: Notificações

Pré-Requisitos	Cuidadores que criaram conta na plataforma, subscreveram um plano, compraram os dispositivos, registaram-nos na *App* e associaram-nos aos dependentes
Descrição	As notificações serão apresentadas ao cuidador com diferentes níveis de severidade. As de maior gravidade serão representadas a vermelho (e.g., quando houver uma queda registada), a laranja, as de gravidade média (e.g., quando alguém sai da cerca virtual), e por fim a verde, as de menor gravidade (e.g., quando alguém entra na cerca virtual ou se encontra num local seguro).
Cenários	Durante o dia, a Sara não está constantemente no telemóvel ou computador o que a impede de estar a par de tudo o que se passa com a sua mãe que está com inicio de demência. No entanto, recebe várias notificações com informação que lhe permite tomar ações para o bem da sua mãe. Ainda ontem, a Sara recebeu uma notificação que o dispositivo que a sua mãe usa estava com pouca bateria, então, antes que recebesse uma notificação a indicar que o mesmo tinha ficado sem bateria e desligado, entrou em contacto com a mãe e pediu-lhe que pusesse o dispositivo a carregar.

3.3 Requisitos

Com o intuito de identificar os requisitos necessários para a elaboração do projeto, foram realizadas reuniões com a equipa da empresa *Altice Labs*, de forma a ter uma perspetiva mais pormenorizada das funcionalidades desejadas. Desta forma, foi possível identificar os requisitos chave para serem a base da implementação do projeto, assim como os requisitos facultativos, permitindo uma redução na probabilidade de insucesso da aplicação.

3.3.1 Requisitos Funcionais

Com a análise da solução pretendida, e depois de algum trabalho de investigação, foram alcançados os seguintes requisitos funcionais, que devem permitir ao utilizador:

- Registar na aplicação, fornecendo o seu nome, email, contacto, número de identificação fiscal e a palavra-passe que lhe irá dar permissão para entrar na plataforma;
- Alterar os seus dados pessoais;
- Alterar os seus dados de acesso;
- Autenticar na aplicação, indicando o seu email e palavra-passe;
- Subscrever a um plano para uso das funcionalidades da aplicação;

- Adicionar dependentes a seu cuidado, fornecendo dados sobre os mesmos;
- Alterar dados dos dependentes
- Adicionar dispositivos que serão usados para obter dados;
- Associar dispositivos a dependentes;
- Alterar dados dos dispositivos e/ou alterar dependente associado;
- Visualizar localização dos diferentes dependentes no mapa;
- Desenhar cercas virtuais com diferentes formas geométricas(círculo, quadrado e polígono);
- Alterar dados das cercas virtuais;
- Remover cercas virtuais;
- Atribuir cercas virtuais a dependentes;
- Assinalar locais seguros no mapa;
- Alterar dados de lugares seguros;
- Remover lugares seguros;
- Atribuir lugares seguros a dependentes;
- Receber notificações que informem sobre os diferentes eventos espoletados pelos dispositivos (bateria, desconexão, conexão, entrada e saída de cercas virtuais, entrada e saída de locais seguros e avisos SOS);
- Obter um histórico de localizações de um dependente num determinado intervalo de tempo.

3.3.2 Requisitos Não Funcionais

- A aplicação tem de estar disponível na língua portuguesa e inglesa;
- A aplicação não deve ser ofensiva em termos religiosos, étnicos ou sexuais;
- O sistema deve suportar o registo de, pelo menos, 100 utilizadores por ano;
- O sistema deve estar operacional pelo menos 361 dias por ano (99%)
- O sistema deve apresentar resposta a qualquer pedido do utilizador em menos de 2 segundos, desprezando atrasos na rede;

- Cada página deve ser totalmente carregada em menos de 2 segundos, desprezando atrasos na rede;
- Qualquer atualização na aplicação deve ser realizada nos períodos de menor utilização;
- O sistema deve prevenir a introdução de dados errados;
- O sistema deve prevenir o acesso indevido aos dados pessoais;
- A aplicação deve ser suportada pelos browsers: *Safari, Mozilla Firefox, Google Chrome, Opera* e *Microsoft* ;
- A aplicação deve garantir a privacidade dos dados dos utilizadores;
- A aplicação deve processar e lançar alertas em menos de 5 segundos;
- Deve possuir uma interface intuitiva e responsiva.

3.4 Arquitetura Funcional

A *Altice Labs* já possui uma solução de telemonitorização, o *SmartAL*; uma solução focada em telemonitorizar pessoas que necessitem de acompanhamento remoto por parte de profissionais de saúde e/ou cuidadores informais. O *SmartAL* não possui serviços de localização implementados, daí o surgimento deste trabalho, com vista a desenvolver uma aplicação que permita telelocalizar pacientes. No entanto, o *SmartAL* já é uma solução com algum avanço do ponto de vista tecnológico, pelo que muitos serviços-base de uma aplicação *web* já se encontravam implementados, sendo apenas necessário alinhá-los com a solução descrita na secção 3.1 e os seus casos de uso e requisitos. Deste modo, foi alcançada a arquitetura funcional apresentada na figura 2.

Na figura 2 é possível ver os vários serviços que irão permitir a implementação deste serviço de telelocalização. Os serviços a azul são os serviços já implementados por parte da equipa da *Altice Labs*; serviços utilizados pelo *SmartAL* e que serão também utilizados pelo *Teleloc*. A verde estão assinalados os microsserviços a implementar para ser possível criar uma aplicação de telelocalização. De seguida serão apresentadas as funcionalidades dos serviços presentes nesta arquitetura; no entanto, só no capítulo 4.2 serão detalhadas todas as características de implementação desses serviços, com foco nos microsserviços *Outdoor* e *Geofences*.

- **API Gateway** - Serve de "ponte"entre a interface gráfica/clientes externos e os diferentes microsserviços. Este serviço recebe os pedidos dos clientes e da interface e encaminha-os para os diferentes microsserviços que compõem a aplicação. Após receber a resposta dos microsserviços, o *gateway* devolve a informação a quem efetuou o pedido;

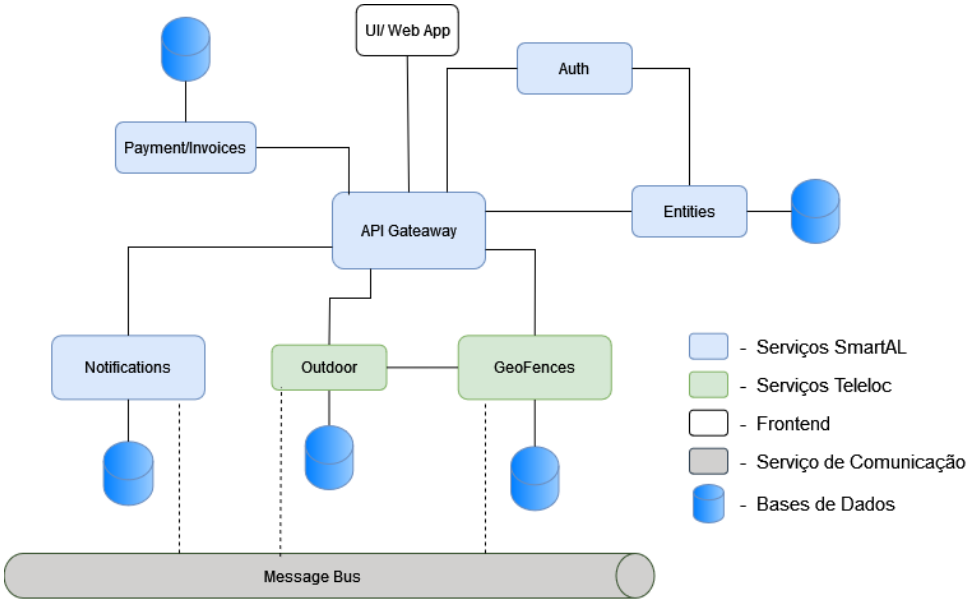


Figura 2: Arquitetura Funcional

- **Auth** - Utiliza o *Keycloak* como *identity provider*[1] (IdP) e é responsável pela autenticação dos utilizadores na aplicação;

- **Entities** - Serviço responsável por gerir os utilizadores e perfis do sistema. No caso desta solução as entidades serão apenas duas, o Cuidador ("Caregiver") e o Dependente ("Dependent"). Apenas o Cuidador tem permissões de autenticação, o Dependente é um perfil sem funcionalidades ativas, servindo apenas para associar os dados de dependentes ao Cuidador. Este perfil, foi implementado para no futuro ser mais fácil de atribuir aos dependentes um uso mais ativo na aplicação, caso se deseje;

- **Payment/Invoices** - Serviço responsável por efetuar e gerir todos os pagamentos feitos na solução e por fazer a ponte entre a plataforma e uma API de faturação externa ;

- **Message Bus** - Serviço para permitir a comunicação assíncrona entre alguns dos serviços da rede, através de um padrão de publicação-subscrição;

- **Notifications** - Serviço responsável por gerir as notificações do sistema. Os diferentes microsserviços publicam notificações no *Message Bus* e o serviço de notificações está à escuta nos diferentes canais de forma a obter e armazenar a informação do que foi publicado. Sempre que é requisitado, este serviço espoleta *push notifications* para o *browser*, e envia SMS e emails;

- **Geofences** - Serviço responsável por gerir a informação das cercas virtuais e a sua lógica de serviço. Dado que as cercas virtuais e as suas verificações exigem alguma complexidade, este

[1] Sistema que guarda e gere as informações das entidades (utilizadores reais e aplicacionais) de determinado sistema.

serviço foi separado do serviço *Outdoor* de forma a simplificar o sistema. Este armazena a informação de todas as cercas virtuais da aplicação e, mediante as posições que recebe dos diferentes dependentes, verifica as suas posições relativas às diferentes cercas;

- **Outdoor** - Serviço responsável por gerir a informação relacionada com a localização exterior e a sua lógica de serviço. Este serviço gere os dispositivos, as localizações e os lugares seguros do sistema. Como o serviço de *Geofences* é um serviço frequentemente utilizado pelo *Outdoor*, os pedidos entre eles são efetuados diretamente, sem a necessidade de passar pela *API Gateway*.

3.5 Diagramas de Sequência

Dado ser uma aplicação assente em microsserviços, a lógica e a informação passada entre eles deve ser simples e de fácil compreensão para uma boa implementação. Para isso foram utilizados diagramas de sequência, que permitiram perceber como a informação iria navegar desde a execução de um pedido até à resposta ao cliente. Os diagramas de sequência são diagramas em UML que representam a sequência de processos num *software*. Assim, é possível visualizar de forma simples e lógica os diferentes métodos e classes e como estes surgem no sistema e colaboram entre si. Os diagramas de sequência são representados por

- **Linhas de vida:** linhas verticais que representam diferentes processos ou objetos;
- **Linhas horizontais:** mensagens ou pedidos trocados entre os diferentes processos;
- **Atores:** entidades externas que interagem com o sistema e executam pedidos. Normalmente enviam a primeira mensagem que inicia toda a sequência.

De forma a entender o funcionamento do sistema e a comunicação entre os diferentes serviços, serão apresentados alguns diagramas de sequência, que demonstram a lógica das principais funcionalidades da aplicação.

3.5.1 Login

Quando um utilizador acede à plataforma, será apresentada a habitual interface da página principal. Aqui é realizada a primeira ação por parte do utilizador, em que tem de escolher se pretende aceder, caso tenha conta, ou, caso não tenha, se se quer registar.

Neste exemplo, representado na figura 3, assume-se que o utilizador já possui uma conta no sistema e seleciona a opção de *login*, demonstrando à interface a intenção de se autenticar, sendo-lhe apresentado o formulário para inserir as suas credenciais. Após preencher e enviar o formulário, o serviço de *front-end* irá comunicar com a *API Gateway* para aceder ao serviço de autenticação ("Auth"). Este, por sua vez, valida os dados inseridos com recurso ao *Keycloak*. Caso estejam corretos, irá realizar a autenticação,

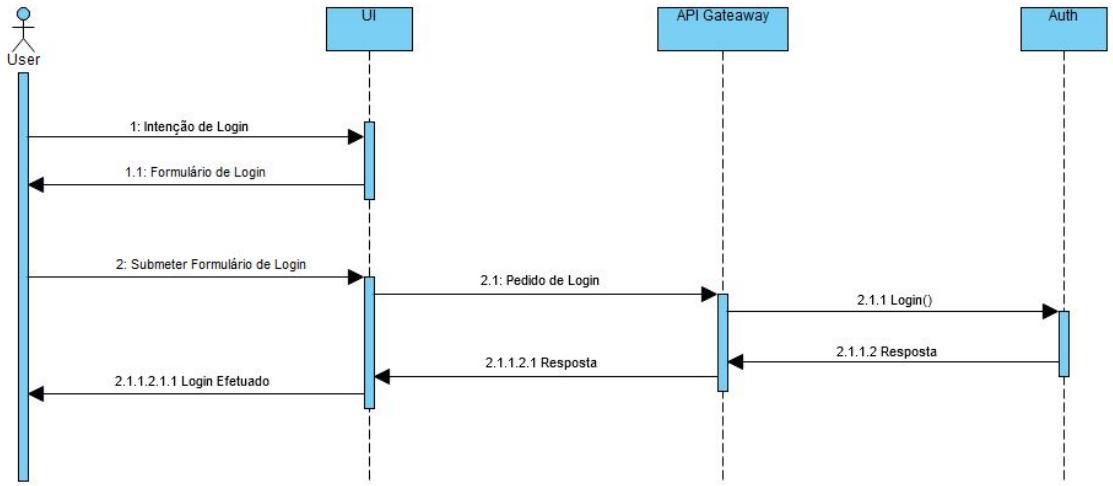


Figura 3: Diagrama de Sequência: Login

gerar um *token* e devolvê-lo à *API Gateway* para devolver a resposta de "*login* efetuado"à interface do utilizador. Caso os dados não estejam corretos, a sequência é semelhante, com a única diferença de que a autenticação não é efetuada, o *token* não é gerado, e a resposta que é apresentada ao utilizador é de insucesso.

3.5.2 Criar Dependente

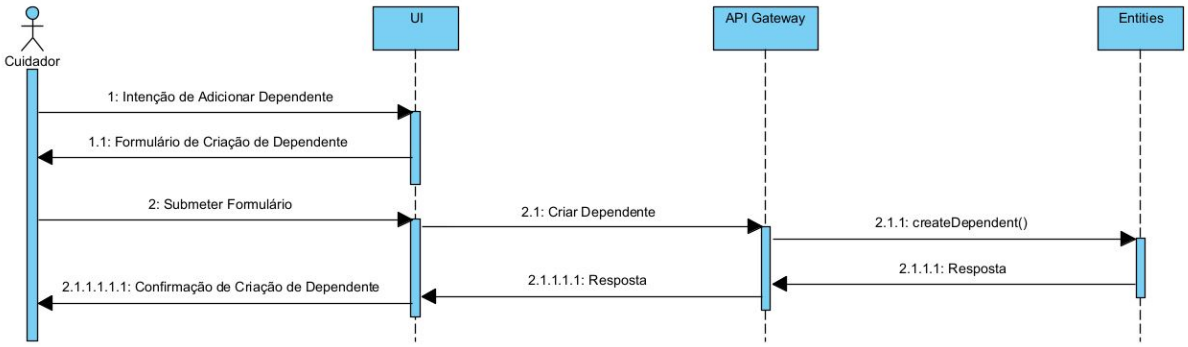


Figura 4: Diagrama de Sequência: Criar Dependente

Após autenticado na aplicação, o cuidador pode realizar várias ações. Assumindo que esta é a primeira interação com a aplicação, o utilizador não terá quaisquer perfis de dependentes adicionados, pelo que é essencial criá-los para poder posteriormente adicionar um dispositivo e receber posições. Para tal, o utilizador irá indicar a intenção de adicionar um dependente e a interface responsabilizar-se-á por apresentar o formulário que permite o envio dos dados do dependente.

Seguidamente, o utilizador envia o formulário de criação e o serviço de *front-end* faz um pedido à *API Gateway* para a criação de um dependente. Este, por sua vez, encaminha o pedido para o microsserviço responsável (neste caso é o *Entities*), que irá executar a função de criação de dependente e adicionar à sua base de dados um novo dependente, associado ao cuidador que efetuou o pedido. Consequentemente, será devolvida a resposta de sucesso ao utilizador, fazendo o percurso inverso ao da realização do pedido. À semelhança do exemplo anterior, caso haja algum problema na criação do dependente, como dados incorretos ou erro por parte do serviço, será devolvida ao utilizador uma resposta a indicar o sucedido.

3.5.3 Criar Dispositivo

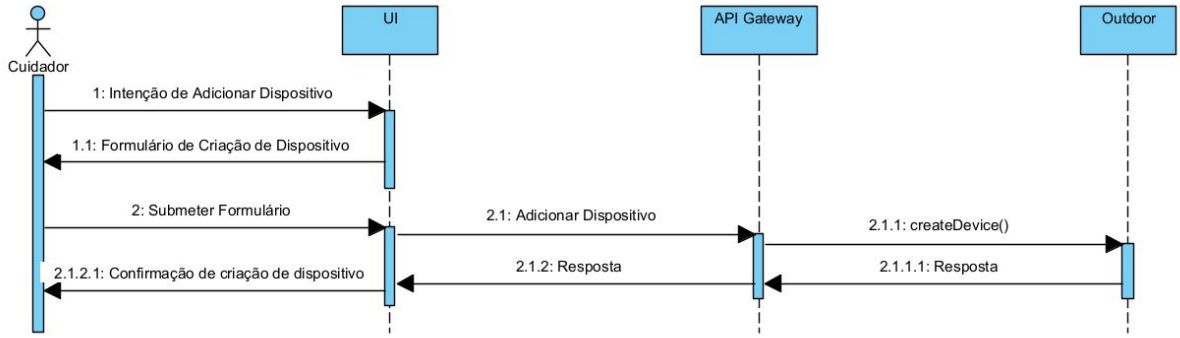


Figura 5: Diagrama de Sequência: Criar Dispositivo

Para se poder receber e apresentar a localização dos dependentes na aplicação, é necessário que o cuidador adicione um dispositivo e atribua o mesmo ao dependente. À semelhança dos exemplos anteriores, o utilizador irá indicar na interface que quer adicionar um dispositivo e ser-lhe-á mostrado o formulário para a criação do mesmo. Após o seu envio, o *front-end* indicará à *API Gateway* o pedido de criação de dispositivo e este, por sua vez, encaminhará para o serviço *Outdoor*, onde será criado e persistido na base de dados. Por fim, a resposta será apresentada ao utilizador, caso o pedido tenha sido realizado com sucesso, ou a mensagem de erro em caso contrário.

3.5.4 Visualizar Localização dos Dependentes

Após o cuidador criar um dependente e ter associado um dispositivo, o sistema começará a receber informações da localização GPS que o dispositivo equipado pelo dependente emite e a guardá-las na base de dados. Cada localização estará associada a um dispositivo, que estará associado a um dependente. Tendo um ou vários dependentes, a lógica para visualizar a localização dos dependentes é a mesma - o cuidador indicará na interface que quer visualizar o mapa, será feito um pedido à *API Gateway*, que encaminhará o pedido de obter a localização de todos os dependentes para o serviço *Outdoor* (que é

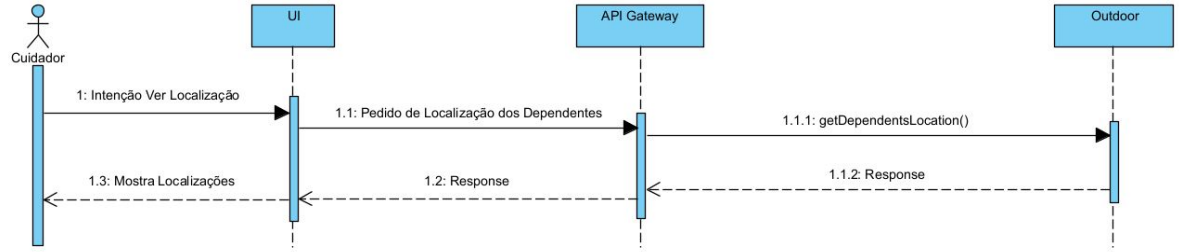


Figura 6: Diagrama de Sequência: Visualizar Localização dos Dependentes

responsável por gerir a informação relacionada com localizações *outdoor*). Este retornará a localização mais recente de todos os dependentes à *API Gateway* e esta será apresentada no mapa ao utilizador.

3.5.5 Criar Cerca Virtual

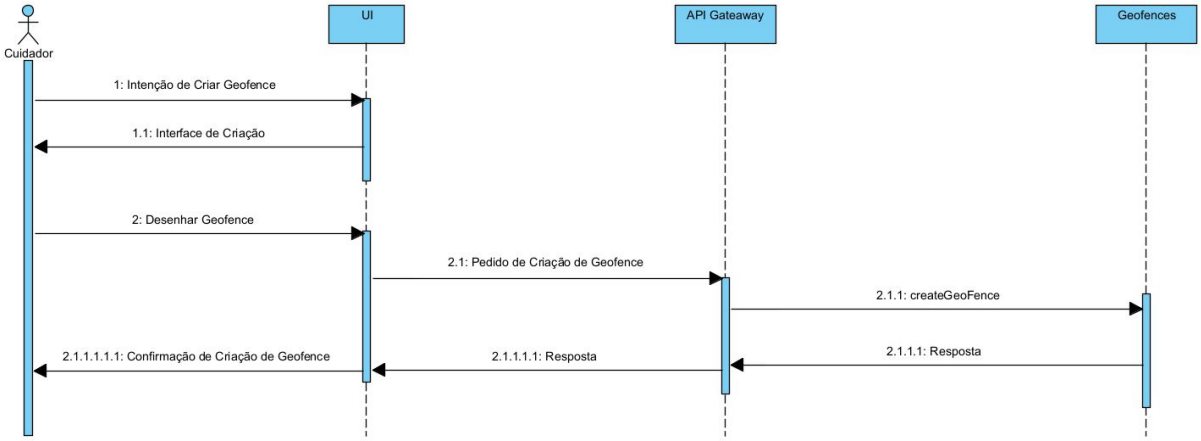


Figura 7: Diagrama de Sequência: Criar Cerca Virtual

Para além de visualizar a localização dos dependentes, o cuidador poderá também querer criar cercas virtuais para ser notificado sempre que o dependente sai ou entra de uma determinada área. Para tal, o utilizador seleciona a opção de criar cerca virtual, em que lhe é apresentada a interface de criação da cerca. O cuidador desenha a área que pretende ser verificada (círculo, retângulo ou polígono) e envia o pedido de criação. A *API Gateway* recebe o pedido e encaminha o mesmo para o serviço *Geofences*, serviço responsável pela gestão das cercas virtuais e que irá persistir os dados de criação da cerca. Após isso, será retornada pelos diferentes serviços a resposta de confirmação de criação de sucesso, ou a resposta com erro de criação, caso tenha acontecido algum erro.

Para a criação de um lugar seguro, a lógica da sequência é muito semelhante à criação da cerca virtual; no entanto, a interface de criação é diferente, pois neste caso quer-se apenas assinalar uma posição no mapa, ao invés de desenhar uma área. O pedido efetuado pela *API Gateway* também será a um serviço diferente - em vez de ser ao serviço *Geofences*, será ao serviço *Outdoor* que, para além de ser

responsável por gerir os dados e pedidos de dispositivos e localizações, também é responsável por gerir os lugares seguros. Excetuando estes dois casos, a sequência de criação de um lugar seguro é igual à de criação de uma cerca virtual.

3.5.6 Verificar Localização dos Dependentes

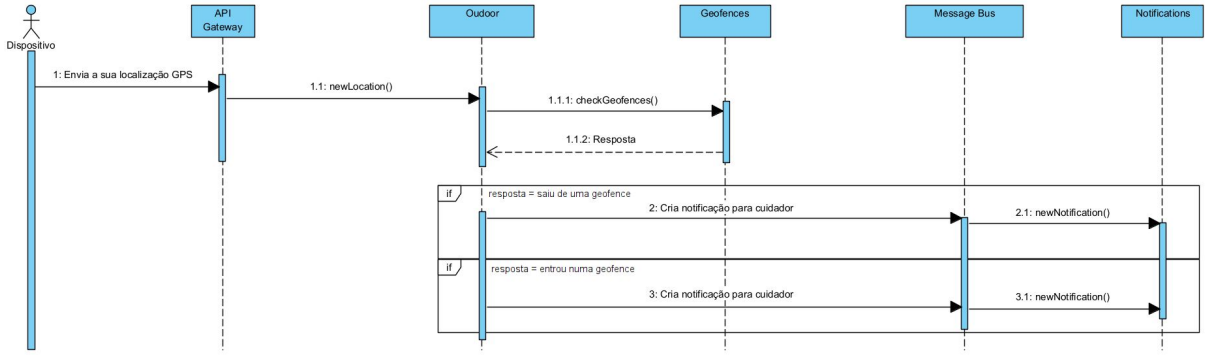


Figura 8: Diagrama de Sequência: Verificar Localização dos Dependentes

Após o utilizador criar os dependentes, associar dispositivos e desenhar cercas virtuais, como foi descrito nas secções anteriores, o sistema encontra-se capaz de verificar a localização dos dependentes à medida que vai recebendo as posições GPS por parte dos dispositivos. No exemplo apresentado no diagrama da figura 8, a *API Gateway* recebe regularmente a posição GPS enviada pelos dispositivos e encaminha-a para o serviço *Outdoor*. Este serviço, por sua vez, regista a localização do dispositivo para ser apresentada ao utilizador mais tarde, caso pretenda, e comunica diretamente com o serviço de *Geofences* para confrontar a posição recebida com as cercas ativas, sem passar pela *API Gateway*. O serviço *Geofences* irá fazer a verificação com base na posição GPS recebida e enviará a resposta de volta para o *Outdoor*, indicando se o dependente se encontra dentro de uma cerca virtual, se saiu de uma, ou se entrou numa após algum tempo. Caso a informação recebida seja de que o dependente se encontra dentro de uma cerca virtual, não será espoletado nenhum alerta para o cuidador, pois tudo se encontra dentro da normalidade. No entanto, caso seja verificado que o dependente se encontrava numa cerca virtual e saiu, o serviço *Outdoor* vai criar uma mensagem e enviar para o *Message Bus*, que irá comunicar diretamente com o serviço *Notifications*, que, por sua vez, irá criar a notificação a ser enviada para o utilizador. Caso seja verificado que o dependente esteve algum tempo fora de uma cerca mas entretanto já retornou à área "autorizada", a lógica anterior repete-se, com diferença no conteúdo da mensagem a apresentar ao utilizador, que indica que o dependente retornou à cerca virtual.

A sequência descrita na figura 8 também se aplica aos lugares seguros, mas, como foi referido anteriormente na criação de cercas virtuais e lugares seguros, a diferença encontra-se na comunicação entre os serviços *Outdoor* e *Geofences* (que não existe), e na verificação que é feita. Neste caso, a

verificação é feita no próprio serviço *Outdoor* e, caso se verifique que o dependente saiu de um lugar seguro, a sequência continua para o *Message Bus* e posteriormente para o *Notifications*.

3.6 Modelo de Dados

No âmbito da modelação da aplicação, é necessário desenvolver um modelo de dados, de forma a servir a lógica atrás apresentada. Uma aplicação que permita ao utilizador a inserção, edição ou remoção de dados, precisa de armazenar os mesmos. Este armazenamento pode ser realizado com recurso a bases de dados, podendo estas ser relacionais ou não-relacionais.

Dada a natureza do projeto, do tipo de dados guardados e a forma como são processados, uma base de dados relacional é a mais adequada para a implementação da solução, associando ao facto de os serviços já implementados pela *Altice Labs* recorrerem também a bases de dados relacionais. Posto isto, foi necessário desenvolver o modelo Entidade-Relacionamento (ER), caracterizado como um modelo de dados que permite ter uma visão estruturada da base de dados, identificando as principais partes envolvidas, os diferentes objetos (entidades), as suas características (atributos) e os relacionamentos entre si.

O primeiro passo na realização de um modelo de dados é definir quais as tabelas que serão criadas. Após identificar as tabelas, é necessário definir a chave primária de cada uma, que permite identificar univocamente cada registo presente em cada tabela. Normalmente as chaves primárias utilizadas são ID únicos gerados ou, no caso de o objeto já possuir um campo que o permita identificar univocamente, esse mesmo atributo. De seguida, são definidos os atributos, que representam a informação concreta dos objetos, as suas características, e indicam o seu tipo. Para terminar, é necessário criar os relacionamentos entre as tabelas – tabelas dependentes de outras, em que um ou mais dos seus atributos são uma referência a atributos de outras tabelas, necessitam da explicitação dessas relações através das designadas "chaves estrangeiras".

Este método para a realização do modelo de dados foi utilizado para a criação dos modelos que se seguem. Visto que esta solução recorre a microsserviços, foi criado um modelo de dados para cada serviço, uma vez que cada serviço é independente dos outros, pelo que deve ter a sua base de dados também independente. Para tal serão descritos os modelos de dados criados, indicando as tabelas a serem criadas para cada um, os atributos e relacionamentos.

3.6.1 Entities

Na figura 9 pode-se ver o modelo de dados do serviço *Entities*. Este microsserviço já estava implementado por parte da *Altice Labs*, gerindo todas as entidades da solução de telemonitorização *SmartAL*. Dado que a presente solução será um complemento ao *SmartAL*, ficou decidido reutilizar este microsserviço, pois já tinha toda a lógica implementada, tendo sido apenas necessário alterar o modelo de dados para

acrescentar as tabelas que guardam a informação dos dependentes e dos cuidadores e as relações entre si.

Figura 9: Modelo de Dados: Entities

- **Dependent** - Armazena os dados dos dependentes. Possui um identificador (id), que consiste numa string unívoca *(UUID)*[2]. Tem relação com a tabela *Caregiver*, sendo que um dependente apenas pode ter um *Caregiver*, podendo um *Caregiver* ter vários dependentes. Possui também relação com a tabela *Profile_image*, sendo esta de 1 para 1; isto é, um dependente só tem uma *Profile_image* e vice-versa;

- **Caregiver** - Armazena os dados dos cuidadores. Tem o mesmo tipo de identificador da tabela *Dependent* e a mesma relação com a tabela *Profile_image*;

[2]Universal Unique Identifier

- **Entity_type** - Tabela responsável por identificar o tipo de classe de um utilizador apenas através do seu ID. Possui referência para os ID dos utilizadores, podendo um utilizador ter apenas um tipo;

- **Profile_image** - Armazena os dados das imagens de perfil das entidades. Possui também um identificador único, referenciado pelas tabelas das entidades. A relação é de 1 para 1, tendo cada entidade uma só imagem de perfil.

3.6.2 Outdoor

O serviço *Outdoor* ao contrário do serviço *Entities*, por exemplo, foi criado especificamente para este projeto, dado que é um serviço de funcionalidades que não estavam implementadas no âmbito do *SmartAL*, tendo sido necessário criar um modelo que satisfizesse as funcionalidades que se pretendiam.

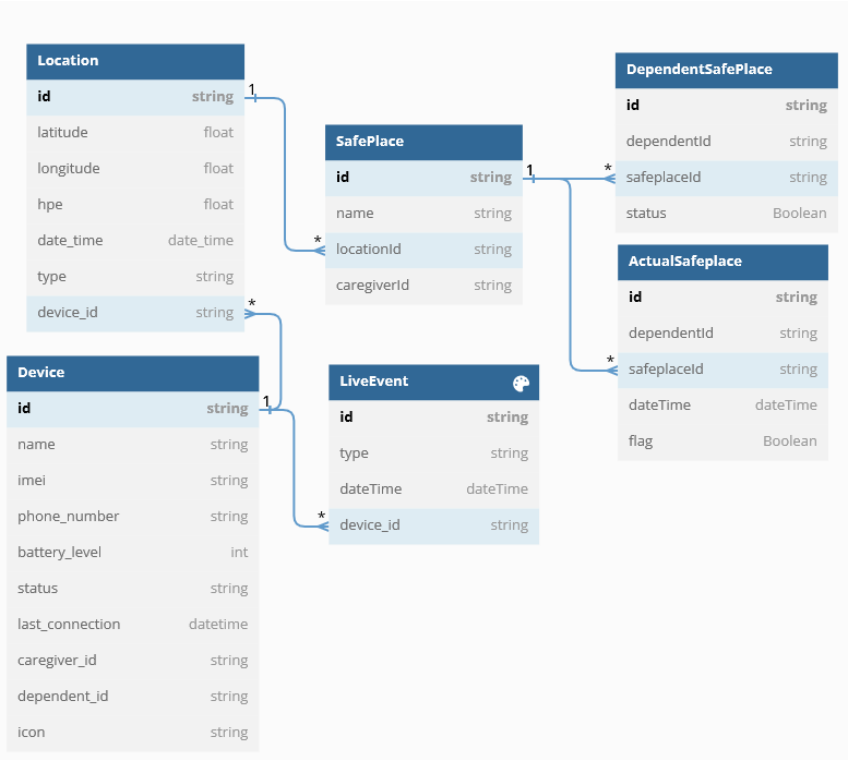


Figura 10: Modelo de Dados: Outdoor

- **Device** - Tabela responsável por armazenar os dados dos dispositivos de localização;

- **Location** - Armazena os dados das localizações emitidas pelos dispositivos. São de notar dois atributos, o *hpe* e o *type*: o *hpe* indica a precisão da localização obtida em percentagem, e o *type* indica como foi obtida a localização (GPS, *Wi-fi* etc.). Uma localização também pode ter associado o ID do dispositivo que a emitiu, no entanto, pode haver situações em que se queira criar localizações sem estarem associadas a dispositivos;

- **Safeplace** - Armazena os dados dos lugares seguros criados. Estes possuem uma localização, daí o relacionamento com a tabela *Location*;
- **ActualSafeplace** - Esta tabela é responsável por manter atualizada a informação do lugar seguro em que cada dependente se encontra atualmente; e serve para se poder fazer uma verificação em tempo real e espoletar alertas caso o dependente saia de um lugar seguro. O atributo *flag* tem como função saber se o cuidador já foi alertado ou não;
- **DependentSafeplace** - Tabela que persiste as relações entre um lugar seguro e um dependente, podendo um lugar seguro estar associado a vários dependentes. O atributo *status* representa se o lugar seguro se encontra ativo ou não;
- **LiveEvent** - Os dispositivos fornecem vários dados, desde informações de posição, bateria, conexão, etc. Posto isto, a tabela *LiveEvent* armazena os dados de todos os eventos enviados pelos dispositivos para a aplicação, sendo também guardado o tipo e o evento recebidos.

3.6.3 Geofences

Outro serviço criado especificamente para a solução apresentada neste documento foi o serviço *Geofences*. Como foi referido, é o serviço responsável por gerir a informação das cercas virtuais e persistir a mesma. Na figura 11 está representado o modelo de dados deste microsserviço.

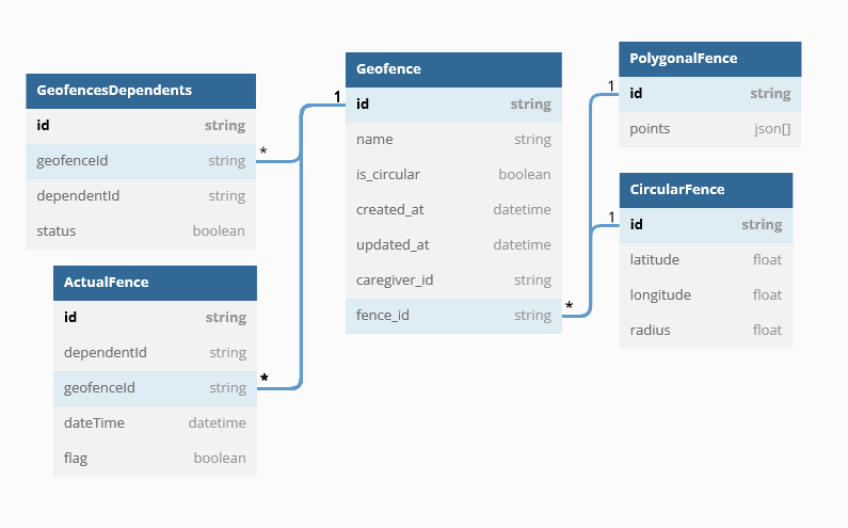


Figura 11: Modelo de Dados: Geofences

- **Geofence** - Armazena os dados informativos das cercas virtuais. O atributo *is_circular* permite saber se a cerca é circular ou poligonal, para que a aplicação saiba a que tabela tem de ir buscar os dados de localização. Tem relacionamentos com as tabelas *PolygonalFences* e *CircularFence*, explicadas de seguida;

- **PolygonalFence** - Armazena os dados da localização dos vértices das cercas poligonais no atributo *points*;
- **CircularFence** - Armazena os dados da localização de uma cerca circular, como o seu centro e o raio;
- **ActualFence** - Tal como no serviço *Outdoor* para os lugares seguros, esta tabela armazena a informação da cerca virtual em que o dependente atualmente se encontra. O atributo *flag* tem a mesma função – saber se o cuidador já foi avisado no caso de o dependente ter saído da cerca;
- **GeofencesDependents** - Tabela que persiste as relações entre uma cerca virtual e um dependente, podendo uma cerca virtual estar associada a vários dependentes. O atributo *status* indica se a cerca virtual se encontra ativa ou não.

3.6.4 Notifications

A figura 12 representa o modelo de dados do serviço *Notifications*, outro serviço que foi reaproveitado do SmartAL. Este serviço é utilizado para armazenar os *tokens* e as notificações dos utilizadores. Este serviço recorre ao serviço *RabbitMQ*, por motivos explicados adiante. Para enviar as *push notifications*, utiliza-se o *Firebase*[3], gerando um *Token* quando um utilizador se autentica na aplicação e armazená-lo para poder depois aceder às notificações.

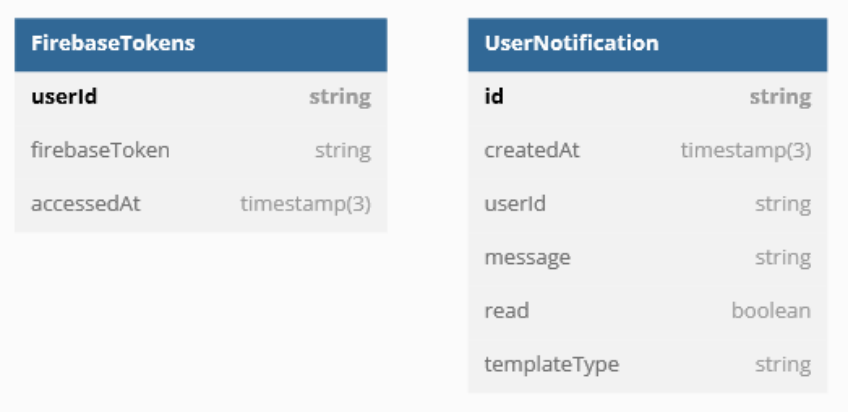


Figura 12: Modelo de Dados: Notifications

- **FirebaseTokens** - Responsável por armazenar os *Tokens* dos utilizadores. Cada entrada possui o identificador único *userId*, que se refere a um utilizador, o atributo *firebaseToken* e a *timestamp* em que ele foi gerado.
- **UserNotification** - Armazena as notificações dos utilizadores. Cada notificação contém um ID, a data de criação, o ID do utilizador a que se refere, o conteúdo da notificação, a indicação se já foi lida e o atributo *templateType*, que indica como deve ser apresentada ao utilizador na interface.

[3]https://firebase.google.com/?hl=pt

4

Implementação da Aplicação

Neste capítulo será apresentado o desenvolvimento da plataforma, ou seja, como foi implementada a modelação descrita no capítulo anterior. Inicialmente, na secção 4.1 será retratada a arquitetura tecnológica, isto é, as ferramentas utilizadas, de que forma foram utilizadas, e como se relacionaram. Para complementar a descrição da implementação, na secção 4.2 são aprofundados os diferentes microsserviços e como cada um foi desenvolvido, sendo também apresentada a interface do utilizador e as suas características. Para terminar, é explicado como foi feito o *deployment* da aplicação e a que tipo de validação foi sujeita.

4.1 Arquitetura Tecnológica

De modo a proporcionar um melhor entendimento de como a solução está dividida, será exposta nesta secção a sua arquitetura tecnológica, que consiste em várias ferramentas e tecnologias comunicantes entre si. Esta ilustração pretende explicar o fluxo do sistema, as camadas e ferramentas envolvidas, e a razão da escolha dessas mesmas ferramentas.

Como foi explicado anteriormente, esta solução assenta em microsserviços, pelo que o desenvolvimento de uma arquitetura tecnológica não é tão linear como numa situação que consiste apenas em camadas de apresentação, de modelo de negócios e de dados. Certos serviços já se encontravam implementados por parte da *Altice Labs*, pelo que não terão as suas tecnologias explicadas, por não se tratar de algo realizado no âmbito deste trabalho.

Na arquitetura apresentada na figura 13, é possível ver as várias tecnologias e ferramentas utilizadas no desenvolvimento desta solução. Inicialmente serão apresentadas as tecnologias utilizadas no desenvolvimento, tais como *React*[1] para o *front-end*, *Node.js*[2] para o *back-end* e *PostgreSQL*[3] para a base de dados. De seguida, serão brevemente abordadas as ferramentas de apoio ao desenvolvimento, como

[1] https://reactjs.org/
[2] https://nodejs.org
[3] https://www.postgresql.org/

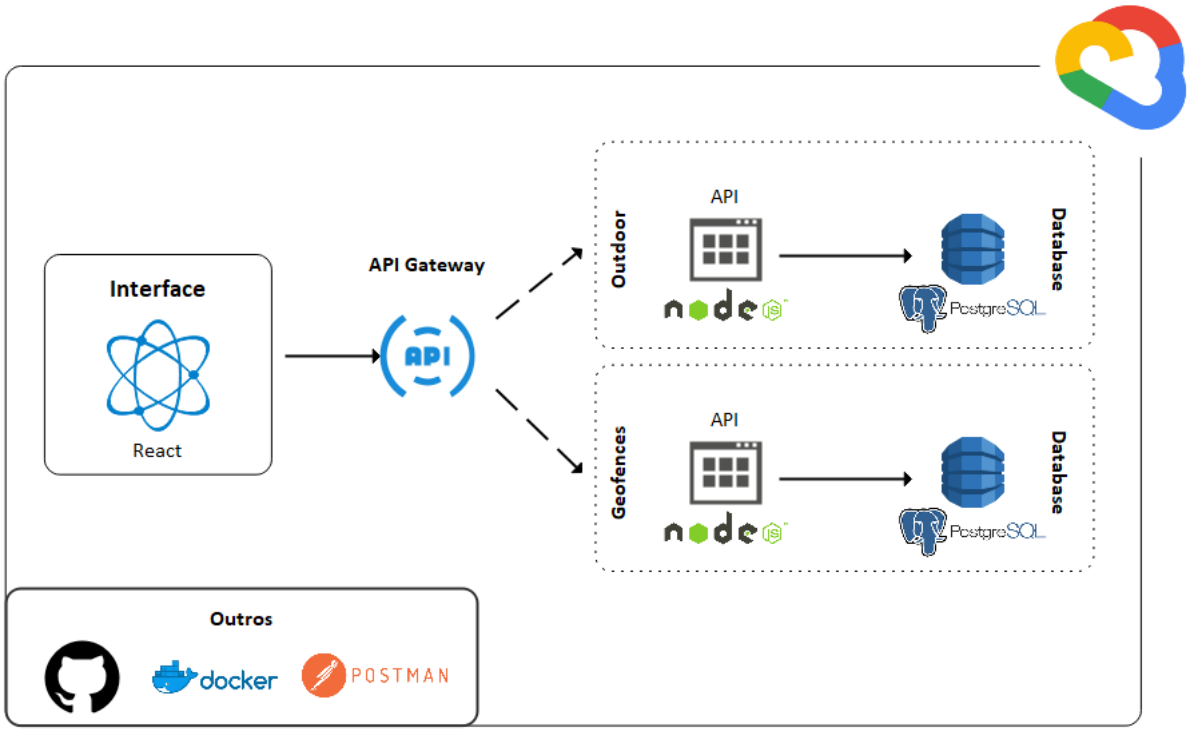


Figura 13: Arquitetura Tecnológica

Docker[4] para criação e gestão de *containers*, o *Github*[5] para controlo de versões e o *Postman*[6] para fazer pedidos às API's e testar as mesmas. Por fim será explicado o porquê da utilização da *Google Cloud*[7] para fazer o *deployment* da aplicação, assim como as suas vantagens e desvantagens.

4.1.1 React

O *React* é uma biblioteca *JavaScript open-source* para criar interfaces para os utilizadores em aplicações web, criado e mantido pela empresa *Meta* [27]. Dado que o *front-end* do *SmartAL* já se encontrava desenvolvido em *React*, e de forma a reutilizar certos componentes e estilos, tirando assim o máximo proveito das suas funcionalidades, a interface nesta solução foi também desenvolvida em *React*. O *React* organiza-se em componentes, em que cada um contém pequenos elementos da UI que, unidos, formam a página final. Esta abordagem permite a atualização e gestão dos dados de forma independente, sendo as atualizações aos dados feitas estritamente onde estão colocados, resultado do conceito de "Virtual DOM". Esta capacidade permite ao *React* ter uma excelente performance e ser fácil de escalar, uma vez que o código está todo separado em pedaços que funcionam de forma independente.

[4] https://www.docker.com/
[5] https://github.com/
[6] https://www.postman.com/
[7] https://cloud.google.com

Uma das particularidades do *React* é o facto de usar o *JSX* para escrever código *HTML* diretamente com o *JavaScript*. Esta funcionalidade traz flexibilidade na ligação dos dados com a estrutura da página web. Relativamente a esta ligação, ela é *one-way data flow*, ou seja, os dados dos componentes não são afetados pelas mudanças nos dados dos descendentes, permitindo um código mais estável e controlável em projetos complexos.

Por fim, esta *framework* é a mais popular dos últimos anos, tendo crescimentos exponenciais a nível de funcionalidades e atualizações, e de integração em diversos projetos, que resultou numa vasta comunidade que disponibiliza *plug-ins* e torna a aprendizagem mais fácil [9]. De seguida serão apresentadas as vantagens e desvantagens do *React*.

Vantagens

- Fácil de aprender e utilizar, qualquer desenvolvedor com *background* em *JavaScript* consegue perceber facilmente e começar a desenvolver utilizando *React*;
- *Virtual DOM* – resulta numa performance otimizada, já que apenas são atualizados os elementos necessários;
- *One-way data flow* – dados dos componentes não são afetados pela alteração dos descendentes;
- Baseado em componentes reutilizáveis que se interligam, mas funcionam independentemente;
- Otimizado para motores de busca;
- Diversos *plug-ins* disponíveis – grande suporte da comunidade, o que também torna a aprendizagem fácil;
- *Cross-platform* – muitas *frameworks* de desenvolvimento *web/mobile* utilizam o *React*;
- Ferramentas de testagem incluídas - *Jest*.

Desvantagens

- Dada a rápida evolução do *React* (o que pode parecer uma vantagem), os desenvolvedores têm de estar constantemente a atualizar-se com novos processos e mecânicas da biblioteca;
- Fraca documentação – a comunidade será o melhor meio de obter conhecimento das potencialidades do *React*.

4.1.2 Node.js

No que diz respeito às *API*, ambas foram desenvolvidas em *Node.js*, um ambiente em *JavaScript* que permite a execução de código do lado do servidor [11]. É uma tecnologia com excelente performance

e baixo consumo de memória, o que faz com que tenha ganho cada vez mais popularidade. Outra vantagem que influenciou a escolha do *Node.js* foi a sua flexibilidade e escalabilidade, graças ao seu gestor de pacotes e bibliotecas (*Node Package Manager (NPM)*), que contém inúmeros componentes reutilizáveis, e também ao facto de ser possível escalar não só horizontalmente, adicionando mais nós, como verticalmente, adicionando mais recursos. Com estas características, o *Node* é cada vez mais utilizado, o que resulta num grande apoio da comunidade, sendo partilhado o conhecimento por todos os utilizadores desta tecnologia. No entanto, a utilização do *Node* tem algumas desvantagens, dado que, devido à vasta quantidade de bibliotecas, alguns recursos podem não estar suficientemente preparados para ambientes de produção, o que muitas vezes impossibilita o seu uso.

Apesar de o *Node.js* ser um ambiente em *JavaScript*, como foi referido, também é possível utilizar o mesmo com *TypeScript*. *TypeScript*[8] é uma linguagem de programação *open-source* que é um superconjunto de *JavaScript*, isto é, adiciona funcionalidades próprias às já existentes do *JavaScript*, neste caso *static typing*. O *typing* consiste em atribuir tipos a objetos, fazendo com que a aplicação esteja melhor documentada e exista um maior controlo sobre o código, uma vez que o compilador de *TypeScript* valida se as operações estão corretas – cada vez que objetos, funções, etc. são chamados no decorrer do projeto, os seus campos têm de estar de acordo com os tipos definidos, sendo detetados os erros mais rapidamente. Apesar de inicialmente parecer mais complexo que o *JavaScript*, tem vindo a ganhar popularidade entre a comunidade, pelo que se optou por utilizar *TypeScript* neste projeto ao invés do *JavaScript*.

4.1.3 Prisma

Para fazer a comunicação com a base de dados utilizou-se *Object-relational mapping (ORM)*, que é um mecanismo que encaminha, acede e manipula os dados de forma agnóstica em relação à base de dados utilizada, não sendo necessário executar comandos em *SQL*, utilizando ao invés uma interface de programação que faz todo o trabalho de persistência dos dados [8]. Por exemplo, com o uso de um *ORM* com uma linguagem de programação orientada a objetos, este mecanismo irá converter as classes em tabelas na base de dados, incluindo os seus atributos e as relações entre si.

Para o desenvolvimento desta solução, utilizou-se o *Prisma*[9], um *ORM open-source* para aplicações com *back-end* em *Node.js* e *Typescript*. Está dividido em três camadas:

- **Prisma Client** - camada em que é gerado um construtor para definir e executar as consultas;
- **Prisma Migrate** - sistema de migração responsável por converter as classes em tabelas;
- **Prisma Studio** - interface para o utilizador visualizar e editar os dados.

O facto de ser um *ORM* recente faz com que não exista ainda muita documentação por parte da comunidade, o que por vezes pode dificultar a solucionar certos problemas, no entanto, dado a sua simplicidade e fácil utilização, foi o escolhido para a execução deste trabalho.

[8] https://www.typescriptlang.org/
[9] https://www.prisma.io/

4.1.4 PostgreSQL

O *PostgreSQL* é um Sistema de Gestão de Base de Dados (SGBD) *open-source* e relacional, que utiliza *Structured Query Language (SQL)* para o tratamento dos dados, combinando com muitas outras funcionalidades que permitem guardar e escalar dados de variada complexidade, de *data types* avançados e com uma excelente performance [14]. Suporta muitos tipos de dados, alguns que são mais comuns noutros SGBD, como *string, integer, float, boolean* etc., mas também dados documentais como *JSON, XML* e até formas geométricas. A sua robustez, alta disponibilidade, tolerância a falhas, ser *open-source*, requerer baixa manutenção e ser suportado em todos os sistemas operativos faz com que seja uma ferramenta muitas vezes escolhida para o desenvolvimento de soluções. Para a sua escolha para este projeto, aliou-se o facto dos microsserviços do *SmartAL* também utilizarem todos este SGBD, e assim facilitar o *deployment* das bases de dados dos diferentes serviços.

4.1.5 RabbitMQ

O *RabbitMQ* é um *software open-source* para mensagens, que implementa o protocolo *Advanced Message Queuing Protocol (AMQP)*, isto é, permite a criação de *queues* (canais) nos quais as aplicações se conectam e trocam mensagens [15]. Permite lidar com o tráfego de mensagens de forma rápida e confiável e é compatível com diversas linguagens de programação. Após existir um canal definido, é criada uma mensagem por uma aplicação que se conecta ao canal, mensagem essa que permanece na *queue* até que outra aplicação a receba. Os conceitos mais importantes para perceber o funcionamento do *RabbitMQ* são os seguintes:

- **Producer** - aplicação que envia a mensagem;
- **Consumer** - aplicação que recebe a mensagem;
- **Queue** - canal que transmite e guarda as mensagens;
- **Exchange** - agente responsável por encaminhar as mensagens para a *queue*, dependendo das regras definidas;
- **Routing Key** - atributo anexado ao *header* das mensagens, que indica como a mesma deve ser tratada pelo *exchange*.

4.1.6 Docker

Docker é uma plataforma que permite virtualizar sistemas operativos, na forma de *containers*. É utilizado para automatizar o desenvolvimento de aplicações, pois utiliza *containers* portáteis, isto é, componentes do *Docker* que contêm toda a informação de uma aplicação ou serviço, podendo ser transferido para outro computador ou *cloud* e ser executado sem ser necessário configurar toda a aplicação novamente.

O *Docker* tem como objetivo o isolamento, criando múltiplos ambientes no mesmo servidor, isolados dos restantes, no entanto, possibilitando a comunicação entre eles. Para criar um ambiente *Docker* é necessário criar um *dockerfile*, um ficheiro com as instruções para a criação de imagens, ou seja, os comandos a serem executados, os ficheiros a serem utilizados e as variáveis de ambiente necessárias. As imagens, por sua vez, são os ficheiros que têm as instruções para criar *containers*. Enquanto um *dockerfile* tem as instruções para criação de várias imagens, as suas variáveis de ambiente, canais de comunicação etc., as imagens são como que um *snapshot*, que, quando executado cria os *containers*. Os *containers* são os referidos ambientes isolados, que são vários neste projeto, pois cada serviço será um *container*. Mais uma vez, a escolha do *Docker* neste trabalho recaiu sobre o facto da equipa da *Altice Labs* já usar esta tecnologia no desenvolvimento; assim, ao criar os *containers* e utilizá-los para desenvolver o projeto, o processo de *deployment* foi facilitado, pois apenas foi necessário carregar e executar as imagens na *Google Cloud*.

4.1.7 Github

O *Github* é uma plataforma para controlo de versões e colaboração entre desenvolvedores e equipas, permitindo a várias pessoas trabalharem no mesmo projeto em simultâneo. Consiste na criação de repositórios, que se traduzem em projetos, que contêm todos os ficheiros dos mesmos, permitindo criar várias versões do mesmo projeto para várias pessoas poderem contribuir em simultâneo, sendo possível fazer um *merge* (fusão) dessas versões para obter um produto mais complexo. A escolha desta plataforma deveu-se ao facto de ser a plataforma utilizada pela *Altice Labs* para o controlo de versões de outros projetos, como por exemplo o *SmartAL*, com o qual a presente solução possui serviços em comum.

4.1.8 Postman

O *Postman* é um *API Client open-source* usado para facilitar o desenvolvimento de *API*. Com esta ferramenta é possível realizar pedidos *HTTP*, sendo possível catalogá-los, facilitando a documentação da *API*. Ao criar um pedido é possível especificar o seu tipo (*GET, POST, PUT, DELETE*, etc.), personalizar a informação enviada e receber as respostas. Com o *Postman* é possível realizar testes às *API* de forma simplificada e partilhar os pedidos com outros membros de equipa, melhorando assim a colaboração e evolução do projeto.

4.1.9 Google Cloud

Google Cloud é uma plataforma que fornece recursos de *cloud computing*, permitindo, entre outras funcionalidades, processar e armazenar dados, ajudar no desenvolvimento, correr testes e instalar aplicações. Dada a sua infraestrutura completa e as diversas funcionalidades que fornece, a *Google Cloud* permite a empresas gerir os seus dados e aplicações de uma forma mais simples, eficaz e com segurança.

Este serviço foi utilizado no *deployment* deste projeto, dado que a *Altice Labs* possui uma conta, não havendo necessidade de subscrever outra ferramenta do género, centralizando assim todos os microsserviços. Apesar de esta ser a razão principal da escolha da *Google Cloud*, o autor salienta as seguintes características, que tornam esta uma escolha acertada para serviço de *cloud*:

- Alta produtividade - consegue gerir vários dados de vários utilizadores ao mesmo tempo sem comprometer o serviço;
- Facilidade de colaboração - múltiplos utilizadores podem trabalhar no mesmo projeto ao mesmo tempo;
- Abstração - dado ser um serviço na *cloud* e aí serem processados todos os dados, os utilizadores podem trabalhar/contribuir a partir de qualquer lado sem se preocuparem com compatibilidade de hardware/software;
- Escalável - é altamente escalável e usa escalamento automático para ajustar o uso de máquinas virtuais conforme a variação de acessos à aplicação;
- Fiabilidade - dispõe de alta fiabilidade; caso um serviço vá abaixo o sistema imediatamente utiliza um secundário, sem que qualquer interrupção seja notada pelos utilizadores.

4.2 Implementação de Microsserviços

Nesta secção serão apresentados os microsserviços e como estes foram implementados. De uma forma mais detalhada, serão abordados os microsserviços *Outdoor* e *Geofences*, que são os que foram criados no âmbito deste projeto; no entanto, serão também abordadas as alterações feitas a outros serviços já implementados por parte da equipa do *SmartAL*, como é o caso do serviço *Notifications*. De notar que nesta secção apenas serão apresentados os microsserviços que constituem a camada de negócio, isto é, o *back-end*, deixando a camada de apresentação, *front-end*, para a próxima secção.

4.2.1 Notifications

O serviço de notificações é um serviço que, como foi dito anteriormente, já se encontrava implementado pela equipa da *Altice Labs*. No entanto, este foi desenhado com vista a ser utilizado exclusivamente pelo *SmartAL*, pelo que foram necessárias fazer alterações para ser também utilizado pelo *Teleloc*. Serão essas as alterações descritas nesta secção, sendo apenas explicadas as funcionalidades que são necessárias para a presente solução.

Na figura 14 é possível visualizar a estrutura do serviço *Notifications*, e de seguida será explicado como este se encontra organizado assim como uma explicação dos ficheiros mais relevantes.

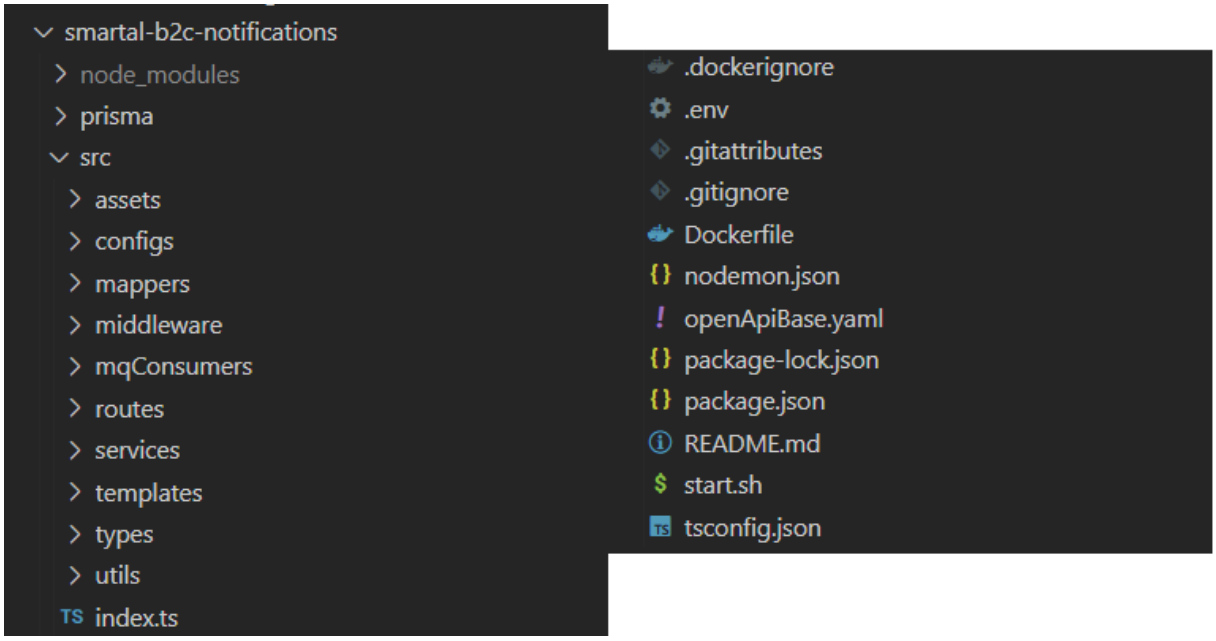


Figura 14: Estrutura do serviço Notifications

prisma

Nesta pasta estão os ficheiros de configuração do *ORM Prisma*, nomeadamente o ficheiro *schema.prisma*. Este ficheiro contém a conexão à base de dados, com o *URL* de ligação a ser passado numa variável de ambiente, e os *models* que se vão traduzir em tabelas na base de dados.

configs

Nesta pasta estão ficheiros com configurações do projeto como variáveis globais, portas de comunicação, etc.

mappers, middleware

Estas diretorias contêm ficheiros com funções e ferramentas que auxiliam o restante código. Na pasta mappers encontra-se o ficheiro *DateTimeFormater.ts*, que é responsável por formatar uma data recebida de acordo com a linguagem e fuso horário que recebe como argumentos quando é chamada. Já na pasta *middleware* está o ficheiro *ExpressAuthTokenValidator.ts*, que é responsável por fazer receber e validar o *token*, verificando se se encontra válido ou se há um erro na autenticação.

types

Como é utilizado *TypeScript*, é necessário definir os tipos a serem utilizados. Nesta pasta encontram-se ficheiros com os tipos necessários para a escrita do código. Na figura 15 vê-se a estrutura de uma notificação (neste caso, de um aviso sobre cercas). A notificação é composta pelo ID da cerca, pelo ID do cuidador a ser notificado, pelo ID do dependente em questão, pelo tipo da notificação (que neste caso é se o dependente entrou ou saiu da cerca virtual) e pela severidade da notificação - alta, média ou baixa. O último campo é o texto a ser adicionado posteriormente à notificação para ser apresentada ao utilizador.

```
22  export interface GeofenceStatus {
23    geofenceId: string;
24    caregiverId: string;
25    dependentId: string;
26    type: string;
27    severity: string;
28    severityText?: string;
29  }
```

Figura 15: Estrutura de uma notificação de *geofences*

mqConsumers

Como foi mencionado na secção 4.1.5, utiliza-se o *RabbitMQ* para fazer a troca das mensagens entre os serviços. Para tal é necessário implementar funções que consumam as diferentes mensagens recebidas. Nesta pasta encontra-se o ficheiro *TelelocStatusConsumer.ts*, onde estão definidas as funções que recebem as notificações e as configuram para serem guardadas na base de dados e enviadas ao utilizador. Cada função é referente a uma *routing key*, um elemento análogo a um endereço para o qual a mensagem é enviada na *queue*, existindo uma para cada tipo de notificação. Após receber uma mensagem a função respetiva vai buscar toda a informação necessária para construir uma notificação para poder ser apresentada ao utilizador.

routes

Contém os ficheiros com as rotas a serem chamadas pela *API Gateway* para obter as notificações.

- **GET /api/userNotifications/history** - retorna todas as notificações de um utilizador, sendo necessário enviar *queryParams* com o ID do utilizador;
- **GET /api/userNotifications/unread** - retorna todas as notificações não lidas de um utilizador;
- **PUT /api/userNotifications/read/notificationId** - atualiza o estado de uma notificação não lida para lida.

services

Nesta pasta encontram-se os ficheiros de serviço, isto é, ficheiros que realizam a comunicação com outras ferramentas e tecnologias, como é a base de dados e o *RabbitMQ*. Nesta pasta encontra-se o ficheiro *EntityService.ts*, que comunica com o serviço das entidades para obter informação dos cuidadores e dependentes. O *MessageQueueService.ts* estabelece a comunicação com o serviço *RabbitMQ*, conectando-se aos diferentes *exchanges* onde vão ser recebidas as mensagens. O ficheiro *UserNotificationService.ts* é responsável por persistir os dados das notificações na base de dados, com recurso ao *Prisma*.

Adicionalmentem, existem ficheiros comuns a uma aplicação em *Node.js*:
index.ts - ficheiro com a implementação geral da *API*;

.env - responsável pelas variáveis de ambiente;
Dockerfile - ficheiro responsável por gerar a imagem *Docker*;
package.json - ficheiro com as dependências a serem instaladas e respetivas versões;
tsconfig.json - ficheiro de configuração do *TypeScript*.

4.2.2 Outdoor

Um dos microsserviços especificamente implementados para este trabalho é o serviço *Outdoor*, que é responsável por gerir os dispositivos, as localizações e os lugares seguros dos dependentes. O desenvolvimento deste microsserviço que, como referido, foi realizado em *Node.js*, começou pela definição do modelo de dados; de seguida, criou-se uma rota e respetiva lógica de negócio para satisfazer cada requisito e, por fim, implementou-se a persistência das alterações na base de dados. Este processo foi realizado ao longo de todo o desenvolvimento, ciclicamente até se obter o número de rotas que satisfizesse os requisitos definidos na modelação do projeto.

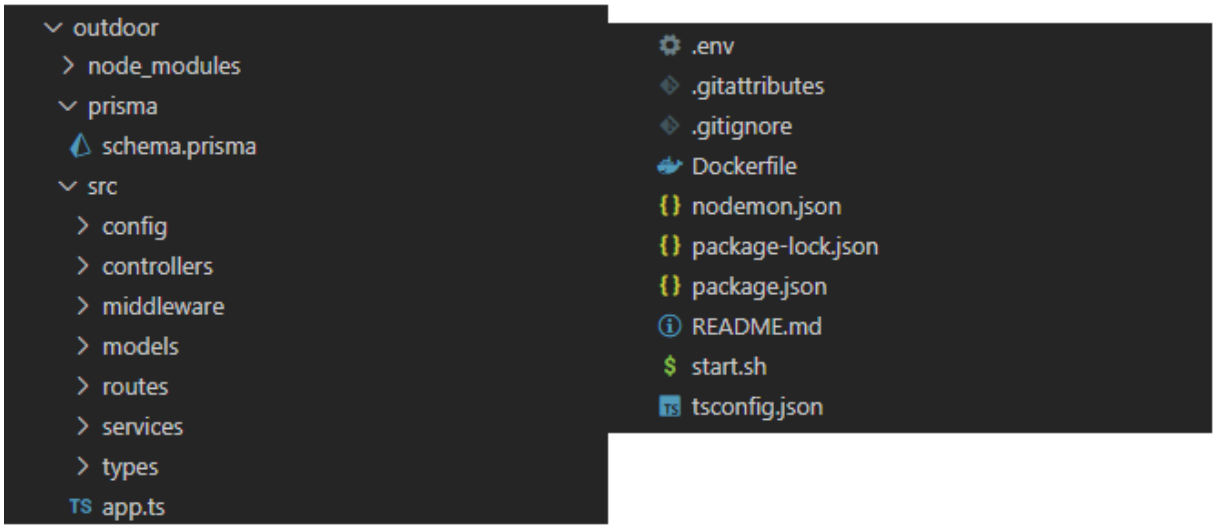


Figura 16: Estrutura do serviço Outdoor

Na figura 16 é possível observar a estrutura do repositório do serviço *Outdoor*, pelo que será explicado em detalhe de seguida.

prisma

À semelhança do serviço *Notifications*, e dado que ambos usam o *Prisma ORM*, na pasta *prisma* encontram-se o ficheiro *prisma.schema*, que contém a conexão à base de dados (sendo também o *URL* de ligação fornecido pela variável de ambiente) e o diretório *models*, que contém as classes que se vão traduzir em tabelas na base de dados. Os modelos a serem criados são os que foram representados na figura 10, na secção 3.6, sobre o modelo de dados da aplicação.

configs

À semelhança do serviço anteriormente apresentado, a pasta *configs* contém ficheiros com configurações do projeto, como variáveis globais, portas de comunicação e a *public key* do *JWT*[10].

middleware

Esta diretoria contém como o nome indica, ficheiros de *middleware*, que servem para realizar validações em certos casos, possuindo os ficheiros *DeviceValidator.ts*, *ExpressAuthTokenValidator.ts* e *LocationValidator.ts*.

```
export const creationSchema = [
  body("imei")
    .not()
    .isEmpty()
    .withMessage("IMEI Missing")
    .bail()
    .custom(async (imei) => {
      const device = await DeviceService.findByImei(imei);
      if (device) {
        throw new Error();
      }
    })
    .withMessage("IMEI already registed"),
  body("name").not().isEmpty().withMessage("Name Missing").bail(),
  body("phone_number")
    .not()
    .isEmpty()
    .withMessage("Phone Number Missing")
    .bail()
    .custom(async (phone_number) => {
      const device = await DeviceService.findByPhone(phone_number);
      if (device) {
        throw new Error();
      }
    })
    .withMessage("Phone Number already registed"),
];
```

Figura 17: DeviceValidator.ts

Na figura 17 pode-se ver o ficheiro *DeviceValidator.ts*. Este *middleware* vai ser chamado aquando da criação de um novo device, para fazer a verificação de certos campos. Primeiro, verifica se o campo IMEI[11] se encontra vazio e, caso não se encontre, verifica se já está registado na base de dados. De seguida, verifica se o nome ou número de telemóvel estão em falta e, caso não estejam, se já se encontram na base de dados. Em todos os casos, se for detetado algum erro, é devolvida essa informação.

[10] JSON Web Token, um padrão para transmitir credenciais de forma segura – https://jwt.io
[11] International Mobile Equipment Identity – número que identifica univocamente um dispositivo móvel

```
export const existLocationSchema = [
  check("locationId")
    .custom(async (locationId) => {
      const location = await LocationService.findById(locationId);
      if (typeof location === "undefined") {
        throw new Error();
      }
    })
    .withMessage("Location doesn't exist on database"),
];
```

Figura 18: LocationValidator.ts

Na figura 18 é apresentada a função que valida a localização. Aquando da criação de um lugar seguro, é necessário este estar associado a uma localização existente na base de dados; para isso, este validador confirma se o ID da localização passada na criação do lugar seguro existe na base de dados, caso contrário espoleta um erro.

```
if (route === "/newLiveEvent") {
  let token = req.headers.authorization;
  if (!token) return res.status(403).send("Authentication error");
  let key = process.env.NEKI_API_KEY;
  if (token === key) {
    next();
  } else {
    return res.status(403).send("Invalid API KEY");
  }
} else {
  let token = req.headers.authorization?.split(" ")[1];
  if (!token) return res.status(403).send("Authentication error");
  jwt.verify(
    token,
    constants.JWT_PUBLIC_KEY,
    { algorithms: ["RS256"] },
    (err, decoded) => {
      if (err) {
        return res.status(403).send("Invalid Token");
      }
      req.decodedToken = decoded;
      if (!(req.decodedToken.realm_access.roles[3] === "CAREGIVER")) {
        return res.status(403).send("You are not a caregiver");
      }
      log.info(`Request Validated`);
      next();
    }
  );
}
```

Figura 19: ExpressAuthTokenValidator.ts

A validação do *token*, apresentada na figura 19, é feita em duas situações diferentes. A primeira é realizada quando a rota pedida é a */newLiveEvent*, explicada mais à frente neste documento. Neste caso, o token que é recebido no *header* *Authorization* tem de ser igual a um guardado em variável de ambiente para ter acesso à rota. No segundo caso, trata-se da validação realizada para todas as outras rotas, em que o token recebido no mesmo *header* é validado pela função *jwt.verify* de acordo com a *public key* definida.

models e types

Como foi referido, foi utilizado *TypeScript* na realização deste projeto, pelo que é necessário definir os tipos a serem utilizados nas diferentes situações para uma maior eficácia e aproveitamento das funcionalidades desta linguagem. Nesta pasta encontra-se o ficheiro *outdoorModel.ts*, que define os tipos (classes)

que são utilizadas ao longo do projeto. Estes tipos vão ser um reflexo do modelo de dados definido, à semelhança do que acontece na definição dos modelos no *schema.prisma*. Na figura 20 pode-se verificar o exemplo de um tipo definido em *TypeScript*, neste caso da classe *Device*. Na pasta *types* também estão definidos alguns tipos a serem usados no desenvolvimento do projeto; no entanto, são referentes a tipos que não são específicos do serviço *Outdoor*, como é o caso dos que definem as entidades *Dependent* e *Caregiver*.

```typescript
export interface Device {
  id: string;
  name: string;
  imei: string;
  phone_number: string;
  battery_level: number | null;
  status: string;
  last_connection: Date | null;
  caregiver_id: string;
  dependent_id: string | null;
  icon: string;
}
```

Figura 20: Classe Device em TypeScript

routes

As *routes* estão divididas em três ficheiros: o *DeviceRouter.ts*, que define as rotas relativas à gestão dos dispotitivos, o *LocationRouter.ts*, que define as rotas relativas à gestão das localizações e o *SafePlaceRouter.ts*, que define as rotas relativas à gestão dos lugares seguros.

DeviceRouter.ts

- **POST /createDevice** - criar dispositivo; os dados do mesmo são passados no *body* do pedido;
- **PUT /editDevice/:id** - editar o dispositivo, cujo ID é um parâmetro e os dados a alterar são enviados no *body*;
- **DELETE /removeDevice/:id** - remover o dispositivo, cujo ID é indicado no parâmetro;
- **GET /device/:id** - obter toda a informação de um dispositivo através do seu ID;
- **GET /dependentDevices/:id** - obter uma lista com a informação dos dispositivos de um dependente, indicando o seu ID;
- **GET /devices** - obter todos os dispositivos associados ao cuidador que realiza o pedido.

LocationRouter.ts

- **POST /setLocation** - criar uma localização; a latitude e longitude da mesma são passadas no *body* do pedido;

- **POST /getLocationHistory/:id** - devolver o histórico de localizações de um dependente, cujo ID é um parâmetro, num intervalo de tempo enviado no *body*;
- **GET /getLocationInfo/:locationId** - devolver a informação de uma localização através do seu ID;
- **GET /getRecentLocation/:id** - devolver a localização mais recente de um dependente através do seu ID;
- **GET /getDependentsLocation** - obter a localização mais recente de todos os dependentes associados ao cuidador que realiza o pedido.

SafePlaceRouter.ts

- **POST /createSafeplace** - cria um *safe place* em que os dados são passados no *body*;
- **POST /setDependentSafeplace** - atribui um novo *safe place* a um dependente, o ID de ambos é enviado no *body*;
- **PUT /editSafeplace/:id** - edita a informação de um *safe place* através do seu ID, os campos a serem alterados são enviados no *body*;
- **PUT /changeSafeplaceDependentStatus/:dependentId/safeplace/:safeplaceId** - altera o status de um *safe place* de um dependente entre ativo e não-ativo;
- **DELETE /removeSafeplace/:id** - apaga o *safe place* com o ID referido;
- **DELETE /removeSafeplaceDependent** - apaga a relação entre o *safe place* e o dependente, ambos os ID são enviados no *body*;
- **GET /getDependentSafeplaces/:id** - obtém uma lista com a informação de todos os *safe places* de um dependente;
- **GET /getSafeplaceDependents/:id** - obtém uma lista com a informação de todos os dependentes associados a um *safe place*;
- **GET /getSafeplaces** - obtém todos os *safe places* do cuidador que fez o pedido;
- **GET /getSafeplace/:id** - obtém informação do *safe place* com o ID em parâmetro.

Na figura 21 pode-se ver o exemplo de uma rota implementada, neste caso a rota usada para criar um dispositivo. Trata-se de um pedido *POST* com o caminho */createDevice*. Esta rota recebe no *body* os dados necessários para a criação de um dispositivo no sistema, pelo que é necessário fazer uma validação dos campos obrigatórios. Como foi referido na descrição do ficheiro *DeviceValidator.ts*, a rota vai validar junto do *creationSchema* e do *validateRequestSchema* se os campos se encontram em conformidade e

```
63  app.post(
64    "/createDevice",
65    creationSchema,
66    validateRequestSchema,
67    (req: Request, res: Response) => {
68      DeviceController.createDevice(req, req.decodedToken.sub)
69        .then((device) => {
70          if (typeof device === "undefined") throw "Couldn't Create Device";
71          log.info(`Device with id ${device.id} was created`);
72          return res.status(201).send(device);
73        })
74        .catch((error) => {
75          log.error(`Error creating device: ${error}`);
76          return res.status(400).send(error.toString());
77        });
78    }
79  );
```

Figura 21: Rota POST /createDevice

só depois proceder à criação do dispositivo. Quando a rota é chamada e o *schema* validado, é chamada a função do *Controller* que vai executar a criação do dispositivo e encaminhar os dados para o *Service* para serem persistidos na base de dados (estas camadas da aplicação serão explicadas em seguida, utilizando o mesmo exemplo). Após criado e persistido o dispositivo, é devolvida uma resposta à função no *DeviceRouter.ts*, que a verifica e devolve com sucesso (ou com erro, caso não tenha sido possível criar o dispositivo).

Este fluxo é semelhante nas restantes rotas, variando apenas o tipo de resposta dada, conforme o que a rota é suposto devolver, influenciando também assim os erros a apresentar.

Existe também neste serviço a rota apresentada na descrição do ficheiro *ExpressAuthTokenValidator.ts*. A rota */newLiveEvent* é utilizada pelos dispositivos para enviarem a informação para a aplicação. Nos testes deste projeto, foram utilizados dispositivos de localização de uma empresa, a *Neki* [10], que forneceu alguns no âmbito de uma parceria com a *Altice Labs*. Cada vez que o dispositivo emite uma informação (e.g., localização, queda, bateria, conexão), a *Neki* envia a informação diretamente para a API. No entanto, por não serem um utilizador normal, já que não se encontram autenticados para este efeito, criou-se esta rota, em que apenas é necessária uma *API key* para a utilizar. Cada vez que é enviado um novo *liveEvent* é feita a verificação se a *API key* é válida, seguindo depois o mesmo processo que as restantes rotas, passando para a função do *Controller* e deste para o *Service*.

controllers

Os *controllers*, tal como as *routes*, estão divididos em três ficheiros: *DeviceController.ts*, *LocationController.ts* e *SafePlaceController.ts*, que executam funções mais complexas no serviço e atuam como intermediários entre os pedidos (rotas) e as funções de persistência de dados (*service*). Como foi dito anteriormente, cada vez que uma rota é chamada, esta encaminha para o respetivo *controller* o pedido para este ser tratado - as rotas servem apenas para receber e enviar conteúdo do utilizador. Ao utilizar ficheiros separados com as funções (*Controllers*), diminui-se a extensividade do código, não ficando tudo no mesmo ficheiro e assim sendo mais compreensível e visualmente mais apelativo. Permite também reutilizar código, implementando uma função apenas num ficheiro (ao invés de a implementar em diversos locais) e invocando-a as vezes que forem necessárias, sendo assim possível, por exemplo, uma

função do *LocationController.ts* chamar diretamente uma função do *DeviceController.ts* (pois necessita da informação de um dispositivo), em vez de passar novamente por uma rota.

```
48  static async createDevice(req: Request, caregiverId: string) {
49    try {
50      var new_device: Device = {
51        id: uuidv4(),
52        name: req.body.name,
53        imei: req.body.imei,
54        phone_number: req.body.phone_number,
55        status: "unpaired",
56        caregiver_id: caregiverId,
57        icon: req.body.icon,
58        battery_level: null,
59        last_connection: null,
60        dependent_id: null,
61      };
62      const device = await DeviceService.createDevice(new_device);
63      return device;
64    } catch (err) {
65      log.error(`Error creating Device: ${err}`);
66      if (err instanceof Error) throw err;
67    }
68  }
```

Figura 22: Função do Controller createDevice

Na figura 22 pode-se observar o exemplo da função do *DeviceController.ts*, que é chamada pela rota *POST /createDevice* e, como o nome indica, é utilizada na criação de um dispositivo, recebendo a informação do *request* e o ID do cuidador. Primeiramente é utilizado um bloco *try/catch*, em que uma exceção/erro na lógica executada no bloco *try* é tratada no bloco *catch*. No bloco *try*, vai ser criado um objeto do tipo *Device*, e gerado um *UUID* com recurso à biblioteca *uuid*. Todos os outros campos encontram-se no *body* do *request*, pelo que são atribuídos aos respetivos atributos, criando assim o objeto com a informação correta. Como, aquando da criação do objeto, ele não fica logo atribuído a um dependente, o seu *status* é inicialmente *unpaired* e a referência para o dependente fica a *null*. O facto de ainda não ter sido recebido nenhum evento por parte do dispositivo implica que o *battery_level* e *last_connection* sejam também *null*, sendo atualizados no primeiro *Live Event*. Após o objeto ser instanciado, vai ser enviado para o *DeviceService*, que é responsável por persistir os dados na base de dados, o que será explicado posteriormente. Após receber a resposta por parte do *service*, é possível devolver à *route* o objeto criado para ser devolvido ao utilizador. Caso a execução do bloco resulte num erro, é espoletado o bloco *catch*, que é responsável por registar o erro nos *logs* e criar uma mensagem de erro que será apresentada ao utilizador.

Uma das funções mais importantes neste serviço encontra-se no *DeviceController.ts*, a *newLiveEvent*, que trata da lógica dos eventos. O *request* de um *live event* contém sempre um tipo (e.g., *position*, *battery_alert*, *reconnect_alert*, *disconnection_alert* e *sos_alert*), a hora do evento e o IMEI do dispositivo a que corresponde, que é denominado aqui por *device_id*. Inicialmente, nesta função é verificado o tipo de evento recebido. Caso seja do tipo *position*, ou seja, informação de uma nova posição do dispositivo,

é criada uma nova *location* a partir desses dados e atualizada a base de dados. Em seguida, é feita uma verificação dessa posição em relação às cercas virtuais e aos lugares seguros.

No caso das cercas, como são geridas por um serviço diferente, utiliza-se o *axios*[12] para fazer um pedido de verificação da localização do dependente recebida em relação às cercas que tem associadas. A figura 23 demonstra esse pedido, em que é enviado o ID do dependente a que o dispositivo está associado e a posição recebida (esta verificação será explicada na secção 4.2.3, que é dedicada ao serviço *Geofences*). Se este serviço devolver a informação de que há um alerta por o dependente ter entrado ou saído de uma cerca, é criada uma mensagem para o serviço *RabbitMQ*, apresentado anteriormente na secção 4.1.5.

```
145         await axios({
146           method: "POST",
147           url: `${constants.REACT_APP_GEOFENCES_API}/internal/checkLocationGeofences`,
148
149           data: {
150             dependentId: device.dependent_id,
151             position: {
152               latitude: location.latitude,
153               longitude: location.longitude,
154             },
155           },
156           headers: {
157             "Content-Type": "application/json",
158           },
159         })
160           .then((res) => {
161             if (device.dependent_id === null) throw "Device has no dependent";
162             if (res.data !== "") {
163               MessageQueueService.sendGeofenceStatus({
164                 caregiverId: device.caregiver_id,
165                 dependentId: device.dependent_id,
166                 geofenceId: res.data.geofenceId,
167                 type: res.data.type,
168                 severity: res.data.severity,
169               });
170             }
171           })
```

Figura 23: Verificação das Geofences na função newLiveEvent

```
MessageQueueService.sendGeofenceStatus({
  caregiverId: device.caregiver_id,
  dependentId: device.dependent_id,
  geofenceId: res.data.geofenceId,
  type: res.data.type,
  severity: res.data.severity,
});
```

Figura 24: Envio de alerta sobre cercas-virtuais para o RabbitMQ

Na figura 24 está demonstrada a criação dessa mensagem, que é constituída pelo ID do cuidador, do dependente e da cerca em questão, o tipo de alerta (i.e., se entrou ou saiu da cerca) e a gravidade do alerta. Em seguida é feita a mesma verificação, mas para o caso dos lugares seguros, seguindo os mesmos passos, criando no fim também uma mensagem para o *RabbitMQ*, se for o caso, com os campos adequados.

[12] https://axios-http.com/

No caso dos eventos em que é recebida uma atualização da bateria, os dispositivos são atualizados no campo *battery_level* e, caso o mesmo se encontre abaixo dos 20%, é enviado um alerta ao cuidador a indicar bateria baixa. Nos restantes tipos de eventos não é necessário haver nenhum tipo de verificação, sendo apenas necessário enviar os alertas com o aviso correspondente. Na figura 25 é possível verificar a criação de uma mensagem sobre um alerta SOS que corresponde a uma queda.

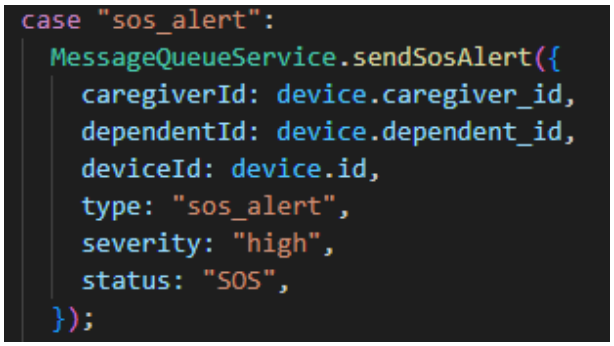


Figura 25: Envio de alerta sobre queda para o RabbitMQ

services

Na última camada dos microsserviços encontram-se os *services*, ficheiros responsáveis por fazer a persistência na base de dados, utilizando o *Prisma ORM*. Os ficheiros nesta diretoria são *DeviceService.ts*, *LocationService.ts* e *SafePlaceService.ts*. Nos *services* encontram-se as funções que são chamadas pelos *controllers* para persistir os dados, pelo que não possuem muita lógica, apenas recebendo dados ou campos e realizando operações CRUD (*create, read, update, delete*).

Nas operações de criação, apenas é necessário indicar a tabela que se pretende criar e os dados a persistir.

Nas operações de leitura, pode-se obter a informação pretendida de várias maneiras:

- findUnique ()- obtém-se apenas uma entrada da tabela, indicando um identificador único;
- findMany() - obtêm-se várias entradas, que correspondam a um filtro;
- findFirst() - obtém-se a primeira entrada encontrada na tabela que corresponda ao critério indicado.

Nestas operações também é possível selecionar apenas alguns campos a serem devolvidos com a *query select*.

As operações de atualização são semelhantes às de criação, com a exceção de que é necessário indicar as condições que devem ser respeitadas para uma ou mais entradas serem atualizadas com a respetiva informação. Por fim, existe a operação de apagar, em que é possível apagar um ou mais registos que respeitem os critérios definidos.

Na figura 26, é apresentado o exemplo da função utilizada para persistir os dados de um novo dispositivo em que é recebido um objeto com os dados do mesmo, passados no campo *data* da operação

create do *prisma* para a tabela *device*. Neste caso também é utilizado um bloco *try/catch*, pelo que, se for espoletado algum erro, este é devolvido ao *controller*.

```
52  static async createDevice(device: Device) {
53    log.info(
54      `Creating Device ${device.name} from caregiver ${device.caregiver_id}`
55    );
56    try {
57      return await prisma.device.create({
58        data: {
59          id: device.id,
60          name: device.name,
61          imei: device.imei,
62          phone_number: device.phone_number,
63          battery_level: device.battery_level,
64          status: device.status,
65          last_connection: device.last_connection,
66          caregiver_id: device.caregiver_id,
67          dependent_id: device.dependent_id,
68          icon: device.icon,
69        },
70      });
71    } catch (e) {
72      log.error(`Error creating device: ${JSON.stringify(e)}`);
73      if (e instanceof Error) throw e;
74    }
75  }
```

Figura 26: Função do Service createDevice

Neste serviço existem também ficheiros que servem de configuração do projeto, assim como a sua configuração para um *Docker Container*:

app.ts - ficheiro com a implementação geral da *API*;

.env - responsável pelas variáveis de ambiente;

Dockerfile - ficheiro responsável por gerar a imagem *Docker*;

package.json - ficheiro com as dependências a serem instaladas, assim como as suas versões;

tsconfig.json - ficheiro de configuração do *TypeScript*.

4.2.3 Geofences

O segundo e último microsserviço implementado especificamente para a aplicação *Teleloc* foi o serviço *Geofences*, que gere toda a informação relativa às cercas virtuais, assim como a sua lógica de serviço. À semelhança do serviço *Outdoor*, inicialmente foi implementado o modelo de dados e criadas as classes e de seguida implementaram-se as rotas, a lógica de cada uma e a lógica de persistência dos dados tratados por cada rota.

Na figura 27 é possível observar a estrutura do repositório do serviço *Geofences*. Muitos dos ficheiros que se encontram neste repositório têm o mesmo propósito de ficheiros que foram descritos nos serviços anteriores. As diretorias *prisma*, *config*, *middleware* e *types* possuem ficheiros com as mesmas funções que as suas homónimas nos outros serviços, servindo respetivamente para fazer a ligação com a base de dados e criar as tabelas, definir variáveis globais e portas de comunicação, validar dados ao longo da

aplicação e definir os tipos (estruturas de dados/interfaces) a serem utilizadas pelo projeto.

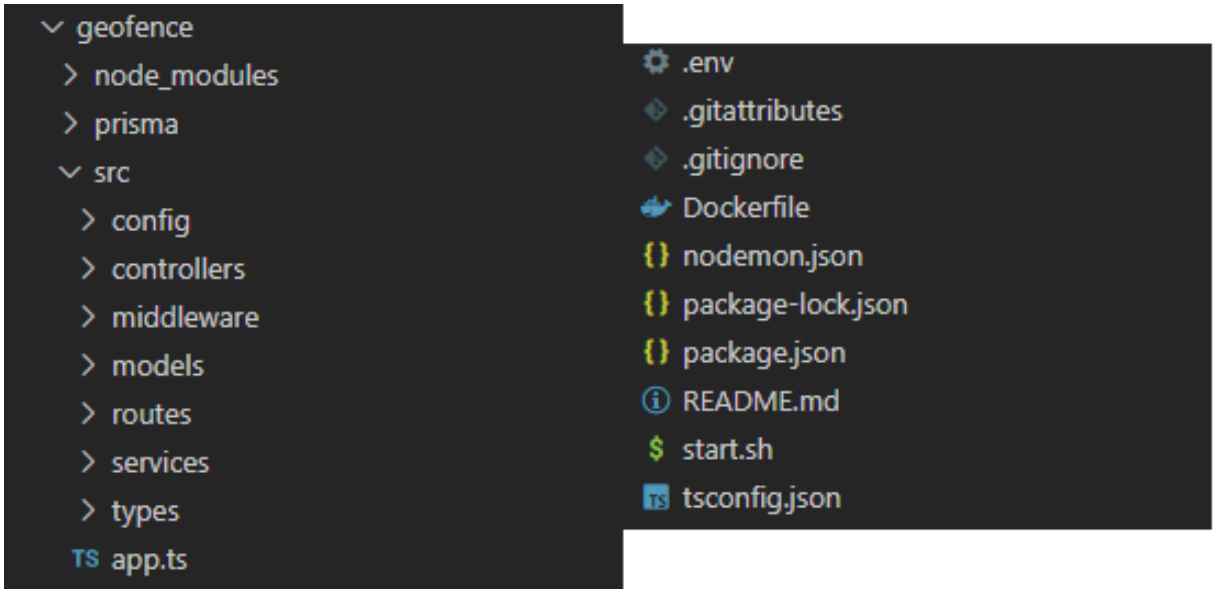


Figura 27: Estrutura do serviço Geofences

models

Nesta pasta encontra-se o ficheiro *geofencesModel.ts*, que define os tipos (classes) que são utilizados ao longo do projeto. Estes tipos vão ser um reflexo do modelo de dados definido, à semelhança do que acontece na definição dos modelos no *schema.prisma*. Na figura 28 pode-se ver o exemplo de um tipo definido em *TypeScript*, neste caso da classe *Geofence*.

```
1  export interface Geofence {
2    id: string;
3    name: string;
4    is_circular: boolean;
5    created_at: Date | null;
6    updated_at: Date | null;
7    caregiver_id: string;
8    fence_id: string;
9  }
```

Figura 28: Classe Geofence em TypeScript

routes

Dado que este serviço é mais pequeno e com menos funcionalidades geridas, em comparação ao *Outdoor*, possui apenas um ficheiro de rotas, o *GeofencesRouter.ts*, que define todas as rotas necessárias para a gestão das cercas virtuais.

GeofencesRouter.ts

- **POST /createGeofence** - criar uma cerca virtual, em que os dados são recebidos no *body* do pedido;
- **POST /setDependentGeofence** - atribuir uma cerca a um dependente, em que os ID são enviados no *body*;
- **POST /removeDependentGeofence** - remover a relação entre uma cerca e um dependente, em que os ID são enviados no *body* do pedido;
- **PUT /editGeofence/:id** - editar uma cerca, com dados enviados no *body* do pedido e o ID passado em parâmetro;
- **PUT /changeGeofenceDependentStatus/:dependentId/geofence/:geofenceId** - alterar o estado de uma cerca entre ativa/inativa relativamente a um dependente, em que os ID são enviados como parâmetros;
- **DELETE /removeGeofence/:id** - apagar uma cerca, indicando o seu ID;
- **GET /getGeofence/:id** - obter todos os dados de uma cerca através do seu ID;
- **GET /geofences** - obter todas as cercas associadas ao cuidador que efetua o pedido;
- **GET /getDependentGeofences/:id** - obter todas as cercas de um dependente através do seu ID;
- **GET /getDependentActiveGeofences/:id** - obter todas as cercas ativas de um dependente através do seu ID;
- **GET /getFenceData/:id** - obter os dados de localização de uma cerca através do seu ID;
- **GET /getGeofenceDependents/:id** - obter os dependentes de uma cerca através do seu ID.

Na figura 29 pode-se ver a rota para a criação de uma cerca virtual implementada. Esta rota é um pedido *POST* com o caminho */createGeofence*; recebe no *body* os dados necessários para a criação da cerca, pelo que é necessário fazer uma validação dos campos obrigatórios (mais simples do que na criação de um dispositivo, pois apenas é verificado o campo *name*). À semelhança do serviço *Outdoor*, a rota faz esta validação através das funções *creationSchema* e *validateRequestSchema*. Após tudo validado, é chamada a função do *Controller* que vai executar a criação da cerca e encaminhar os dados para o Service para serem persistidos na base de dados. Após criada a cerca e persistidos os dados, é devolvida uma resposta à função no *GeofencesRouter.ts*, que verifica a resposta e devolve a mesma com sucesso (ou com erro, caso não tenha sido possível criar a cerca).

```
export default function (app: Express) {
  app.post(
    "/createGeofence",
    creationSchema,
    validateRequestSchema,
    (req: Request, res: Response) => {
      GeofencesController.createGeofence(req, req.decodedToken.sub)
        .then((geofence) => {
          if (typeof geofence === "undefined") throw "Couldn't Create geofence";
          log.info(`Geofence with id ${geofence.id} was created`);
          return res.status(201).send(geofence);
        })
        .catch((error) => {
          log.error(`Error creating device: ${error}`);
          return res.status(400).send(error.toString());
        });
    }
  );
```

Figura 29: Rota POST /createGeofence

Existe também neste ficheiro a rota *POST /internal/checkLocationGeofences*, uma rota que é utilizada pelo serviço *Outdoor* para fazer a validação das posições recebidas relativamente às cercas de cada dependente. Cada vez que é recebido um novo *liveEvent* no serviço *Outdoor*, como já foi verificado, é feito um pedido por parte desse mesmo serviço para esta rota, indicando no *body* o ID do dependente cuja posição verificar e a localização em questão, sendo a verificação feita no *controller*.

controllers

O *GeofencesController.ts* executa as funções lógicas no serviço. Cada vez que uma rota é chamada, o pedido é encaminhado para a respetiva função no *controller* para ser tratado.

```
static async createGeofence(req: Request, caregiverId: string) {
  try {
    let fence: CircularFence | PolygonalFence;
    if (req.body.is_circular) {
      fence = {
        id: uuidv4(),
        latitude: req.body.data.latitude,
        longitude: req.body.data.longitude,
        radius: req.body.data.radius,
      } as CircularFence;
    } else {
      let points: {
        latitude: number;
        longitude: number;
      }[] = [];
      for (let i = 0; i < req.body.data.length; i++) {
        points.push(req.body.data[i]);
      }
      fence = {
        id: uuidv4(),
        points: points,
      } as PolygonalFence;
    }
    const new_geofence: Geofence = {
      id: uuidv4(),
      name: req.body.name,
      is_circular: req.body.is_circular,
      created_at: new Date(),
      updated_at: null,
      caregiver_id: caregiverId,
      fence_id: fence.id,
    };
    const geofence = await GeofencesService.createGeofence(
      new_geofence,
      fence
    );
    return geofence;
  } catch (err) {
    log.error(`Error creating geofence: ${err}`);
    if (err instanceof Error) throw err;
  }
}
```

Figura 30: Função do Controller createGeofence

Na figura 30 pode-se observar o exemplo da função do *GeofencesController.ts*, que é chamada pela rota *POST /createGeofence*, utilizada na criação de uma cerca virtual, recebendo a informação do pedido e o ID do cuidador. Mais uma vez, é utilizado um bloco *try/catch*, em que é executada lógica no *try* ou, caso seja espoletada uma exceção/erro, é tratada no *catch*. No bloco *try*, vai ser criado um objeto do tipo *CircularFence* ou *PolygonalFence*. Este objeto possui os dados de localização da cerca e tem uma estrutura diferente, consoante a cerca for circular ou poligonal (dependendo do atributo *is_circular* que é recebido no *body*). Após verificada se é circular ou não, atribuem-se os dados ao objeto conforme o seu tipo, sendo gerado um UUID com recurso à biblioteca *uuid* em ambos os casos. Caso seja circular, o objeto vai conter a latitude e longitude do centro da circunferência e o raio da mesma. Se for poligonal, vai ser criado um atributo *points* que contem um *array* com os pontos (latitude e longitude) dos vértices do polígono. Todos os outros campos encontram-se no *body* do pedido e são atribuídos aos respetivos atributos. Após isso é criado um objeto do tipo *Geofence*, em que são acrescentados os restantes dados e a referência para o ID do objeto *fence* que contém os dados da localização. Posteriormente é feito o pedido ao *GeofencesService*, para onde é enviado o objeto para os dados serem persistidos. Após recebida a resposta por parte do *service*, é possível devolver à *route* o objeto criado para ser devolvido ao utilizador. Caso a execução do bloco resulte num erro, é espoletado o bloco *catch*, que é responsável por registar o sucedido nos *logs* e enviar ao utilizar a mensagem sobre erro em questão.

No *GeofencesController.ts* existe a função *checkLocationGeofences*, que é chamada pela rota *POST /internal/checkLocationGeofences* quando é necessário verificar uma posição recebida em relação às cercas do dependente. A função vai receber o pedido como parâmetro, para ser possível aceder ao *body* e obter os dados necessários para a verificação, que são a localização e o dependente que se encontra na mesma. Inicialmente vai ser criado um *array* de objetos que vão indicar a posição do dependente em relação às várias cercas (*in* ou *out*) e que serão adicionados à medida que a verificação é feita para cada uma das cercas ativas. Em seguida, é verificado se o dependente, na atualização anterior a esta posição, se encontrava dentro de uma cerca, sendo feito um pedido ao *Service* para obter informação da tabela *ActualFence* (explicada no modelo de dados na secção 3.6). Finalmente, são obtidas as cercas do dependente – caso não tenha nenhuma, a função termina neste passo, pois não há nenhuma verificação a fazer; caso contrário continua e será feita uma verificação para cada cerca, sendo diferentes as verificações nas cercas circulares das poligonais.

Na figura 31 é apresentada a função que faz a verificação da posição numa cerca circular. Primeiramente são obtidos os dados de localização da cerca (ponto do centro e raio). Para fazer esta verificação foi utilizada a fórmula de *haversine*, uma fórmula utilizada para calcular a distância entre dois pontos numa esfera (neste caso no planeta Terra). Para utilizar a fórmula são definidas as seguintes variáveis e equações:

- **R** - raio da Terra em metros;
- **dx** - latitude da localização recebida em radianos;

- **dy** - latitude do centro da cerca em radianos;
- **rx** - diferença entre a latitude do centro da cerca e a latitude do ponto recebido em radianos;
- **ry** - diferença entre a longitude do centro da cerca e a longitude do ponto recebido em radianos.

$$a = \sin(rx/2)^2 + \cos(dx) \cdot \cos(dy) + \sin(ry/2)^2 \tag{4.1}$$

$$c = 2 \cdot atan2(a, 1 - a) \tag{4.2}$$

$$d = R \cdot c \tag{4.3}$$

As equações 4.1 e 4.2 são a fórmula de *haversine* dividida para simplificar a sua implementação. Através da resolução destas duas equações obtém-se a variável *c*, que representa o ângulo central, um arco entre os dois pontos. Após obtido o ângulo central, o mesmo é multiplicado pelo raio da Terra, o que se vai traduzir na distância entre os dois pontos, em metros. Posteriormente este valor é comparado com o raio da cerca e, caso seja menor, significa que o ponto se encontra dentro da cerca, caso contrário está fora da cerca.

```
363 async function checkCircular(
364   point: { latitude: number; longitude: number },
365   fence: Geofence
366 ): Promise<boolean> {
367   const fence_values = (await GeofencesService.getFenceData(
368     fence.fence_id,
369     fence.is_circular
370   )) as CircularFence;
371   if (typeof fence_values === "undefined") throw "Error getting fence data";
372
373   const R = 6371e3; // metres
374   const dx = (point.latitude * Math.PI) / 180;
375   const dy = (fence_values.latitude * Math.PI) / 180;
376   const rx = ((fence_values.latitude - point.latitude) * Math.PI) / 180;
377   const ry = ((fence_values.longitude - point.longitude) * Math.PI) / 180;
378   const a =
379     Math.sin(rx / 2) * Math.sin(rx / 2) +
380     Math.cos(dx) * Math.cos(dy) * Math.sin(ry / 2) * Math.sin(ry / 2);
381   const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
382   const d = R * c; // in metres
383   const r = fence_values.radius;
384   if (d <= r) {
385     return true;
386   } else {
387     return false;
388   }
389 }
```

Figura 31: Verificação da Posição numa Cerca circular

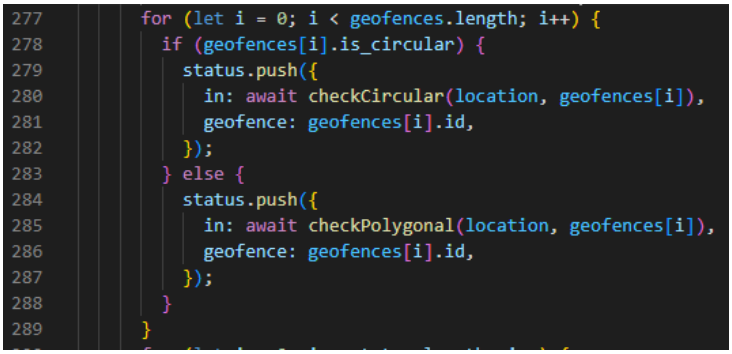
```
async function checkPolygonal(
  point: { latitude: number; longitude: number },
  fence: Geofence
): Promise<boolean> {
  const fence_values = (await GeofencesService.getFenceData(
    fence.fence_id,
    fence.is_circular
  )) as PolygonalFence;
  if (typeof fence_values === "undefined") throw "Error getting fence data";
  let points: number[][] = [];
  fence_values.points.forEach((point) => {
    points.push([point.latitude, point.longitude]);
  });

  var n = points.length,
    is_in = false,
    x = point.latitude,
    y = point.longitude,
    x1,
    x2,
    y1,
    y2;

  for (var i = 0; i < n - 1; ++i) {
    x1 = points[i][0];
    x2 = points[i + 1][0];
    y1 = points[i][1];
    y2 = points[i + 1][1];

    if (y < y1 != y < y2 && x < ((x2 - x1) * (y - y1)) / (y2 - y1) + x1) {
      is_in = !is_in;
    }
  }

  return is_in;
}
```

Figura 32: Verificação da Posição numa Cerca poligonal

A verificação do ponto relativamente a uma cerca poligonal inicia da mesma forma, sendo obtidos os pontos que constituem os vértices do polígono. Após isso, inicia-se a verificação que se pode ver implementada na figura 32. É criada uma matriz com os valores da longitude e latitude dos vértices e serão percorridos todos os valores da matriz, sendo a mesma realizada aos pares de forma a ser formada uma "linha" entre os dois pontos e verificar a localização obtida em relação a esta "linha". Após feita a verificação, é devolvida à função a variável *is_in*, que pode ser verdadeira ou falsa, consoante a localização do ponto em relação ao polígono (dentro ou fora, respetivamente).

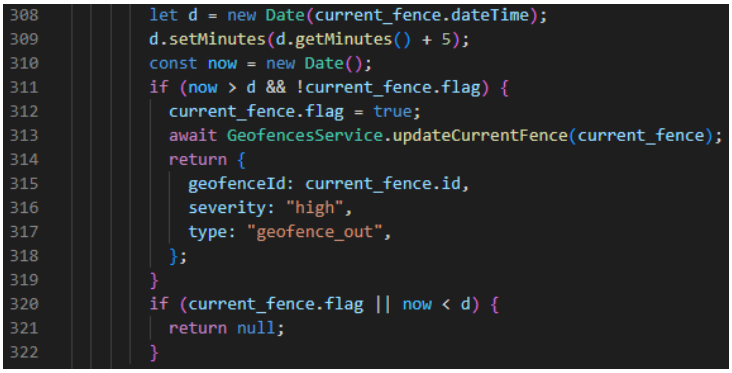
Cada vez que uma destas verificações é feita, o resultado é devolvido à função principal e é adicionado ao objeto o ID da cerca de que foi feita a verificação e o resultado, como é possível verificar na figura 33.

Após criado este objeto, o mesmo é percorrido de forma a verificar se houve uma atualização na cerca em que o dependente se encontra atualmente. Se saiu de uma cerca em que se encontrava, é criada uma resposta com um alerta para ser enviado ao utilizador como notificação.



```
277     for (let i = 0; i < geofences.length; i++) {
278       if (geofences[i].is_circular) {
279         status.push({
280           in: await checkCircular(location, geofences[i]),
281           geofence: geofences[i].id,
282         });
283       } else {
284         status.push({
285           in: await checkPolygonal(location, geofences[i]),
286           geofence: geofences[i].id,
287         });
288       }
289     }
```

Figura 33: *Loop* de verificação de posição



```
308       let d = new Date(current_fence.dateTime);
309       d.setMinutes(d.getMinutes() + 5);
310       const now = new Date();
311       if (now > d && !current_fence.flag) {
312         current_fence.flag = true;
313         await GeofencesService.updateCurrentFence(current_fence);
314         return {
315           geofenceId: current_fence.id,
316           severity: "high",
317           type: "geofence_out",
318         };
319       }
320       if (current_fence.flag || now < d) {
321         return null;
322       }
```

Figura 34: Resposta de alerta de saída de uma cerca

Na figura 34 é possível visualizar a criação de uma notificação no caso de o dependente não se encontrar em nenhuma cerca. Inicialmente é obtida a data em que está a ser feita a verificação e a data em que o dependente esteve na cerca pela última vez. Comparando estes dois tempos, e caso a diferença seja superior a 5 minutos e a *flag* indique (por ter o valor falso) que não foi ainda enviada uma notificação ao cuidador, esta mesma é atualizada para verdadeiro e é criada uma notificação de severidade alta a indicar que o dependente saiu da cerca em que se encontrava. Neste caso foi utilizado uma margem de 5 minutos, para evitar situações de erro que por vezes ocorrem na obtenção das localizações, pois um dependente pode-se encontrar no limiar da cerca e assim estariam constantemente a ser enviadas notificações ao utilizador no caso de as localizações por vezes saírem um pouco do limite. Caso a *flag* esteja a verdadeiro (indicando que o cuidador já foi avisado) ou ainda não tenham passados 5 minutos, nenhuma ação é realizada.

services

Dada a necessidade de persistência dos dados relativos às cercas, como a informação das suas localizações, a cerca em que o dependente se encontra atualmente, relações, etc., é utilizado, à semelhança dos serviços anteriores, o ficheiro *GeofencesService.ts*, que realiza as operações CRUD de conexão à base de dados que foram apresentadas anteriormente.

Na figura 35 é apresentado o exemplo da função utilizada para persistir os dados de uma nova cerca em que são recebidos dois objetos: um relativo às informações da cerca – nome, se é circular, data de criação, cuidador e ID da cerca – que guarda os dados de localização, e essa mesma cerca com os dados

de localização. Dado haver uma dependência entre a tabela *Geofence* e *Fence*, é necessário persistir os dados nesta primeiro, pois a primeira possui a referência para o ID de uma entrada desta tabela. De seguida, utilizando a operação *Create*, são persistidos os dados com a informação genérica da cerca. À semelhança dos exemplos anteriores, esta implementação encontra-se num bloco *try/catch*, pelo que, em caso de sucesso, a informação da criação da cerca é devolvida ao utilizador, caso contrário é devolvido o erro.

```
18  static async createGeofence(
19    geofence: Geofence,
20    fence: PolygonalFence | CircularFence
21  ) {
22    log.info(
23      `Creating Geofence ${geofence.name} from caregiver ${geofence.caregiver_id}`
24    );
25    try {
26      this.createFence(fence);
27      return await prisma.geofence.create({
28        data: {
29          id: geofence.id,
30          name: geofence.name,
31          is_circular: geofence.is_circular,
32          created_at: geofence.created_at,
33          caregiver_id: geofence.caregiver_id,
34          fence_id: geofence.fence_id,
35        },
36      });
37    } catch (e) {
38      log.error(`Error creating device: ${JSON.stringify(e)}`);
39      if (e instanceof Error) throw e;
40    }
41  }
```

Figura 35: Função do Service createGeofence

Neste serviço também se encontram os ficheiros de configuração de projeto, com as mesmas funções que foram descritas no serviço *Outdoor* (4.2.2), aplicadas a este cenário.

4.3 Interface do Utilizador

Como foi referido anteriormente na apresentação das tecnologias utilizadas (secção 4.1), o *front-end* da aplicação *Teleloc* foi desenvolvido em *React*. O desenvolvimento utilizando esta tecnologia careceu de alguma aprendizagem, sendo necessário familiarizar com certos conceitos, como é o caso dos elementos, componentes, *props*, eventos, *hooks*, etc.; no entanto, como esta solução será um complemento da aplicação já desenvolvida, *SmartAL*, foi possível reutilizar a grande maioria das componentes e estilos já desenvolvidos pela equipa da *Altice Labs*, o que simplificou e agilizou todo o processo de desenvolvimento, sendo apenas necessário aplicar ao presente caso de uso, realizando as adaptações necessárias e construindo as interfaces pretendidas. Em seguida, é realizada uma demonstração da aplicação desenvolvida, com foco na interface implementada, explicando como podem ser utilizadas as funcionalidades requeridas.

Na página inicial, o utilizador depara-se com o formulário de *login*, que lhe permite aceder ao conteúdo da plataforma. Para tal o utilizador tem de inserir o seu email e palavra-passe para iniciar sessão e poder

utilizar as funcionalidades do sistema, como se pode verificar na figura 36. Ao clicar em entrar, as credenciais do utilizador são verificadas e, caso sejam válidas, será redirecionado para a página inicial; caso contrário será apresentada uma mensagem de erro.

Figura 36: Aplicação - Página de Login

Caso o utilizador não possua ainda uma conta, pode criar uma, clicando em "Criar conta", sendo redirecionado para o formulário apresentado na figura 37. Aqui, o utilizador deve inserir o email que pretende utilizar para iniciar sessão, assim como a senha/palavra-passe de acesso. Nesta etapa será verificado se o email que pretende registar já se encontra em uso, avisando assim em caso de conflito; caso contrário, após clicar no botão "Criar Conta", a mesma será criada com sucesso.

Figura 37: Aplicação - Página de Registo

Ao iniciar sessão pela primeira vez, dado que no registo apenas foi indicado o email e palavra-passe, será pedido ao utilizador para completar as suas informações registadas, acrescentando alguns dados necessários. Na figura 38, é possível visualizar a mensagem que é apresentada ao utilizador, a informar da necessidade de completar o registo.

Figura 38: Aplicação - Primeiro Login

Na figura 39 é apresentado o formulário para completar o perfil do cuidador. Aqui é possível inserir o nome, o título que deseja ser tratado, a data de nascimento, o sexo e o número de contribuinte para ser apresentado na fatura. Após preenchidos estes campos o utilizador pode carregar no botão "Próximo" e usufruir da totalidade da aplicação.

Figura 39: Aplicação - Dados Pessoais

Após iniciar sessão o utilizador deparar-se-á com a página inicial apresentada na figura 40. No caso de ser um utilizador recente não terá dependentes nem dispositivos associados. Para tal, o cuidador deve

criar primeiro os dependentes, carregando no botão "Adicionar Dependente"no canto superior direito (1) ou então no ícone apresentado no centro do ecrã (2). Após clicar numa destas opções, será encaminhado para o formulário de criação de dependentes.

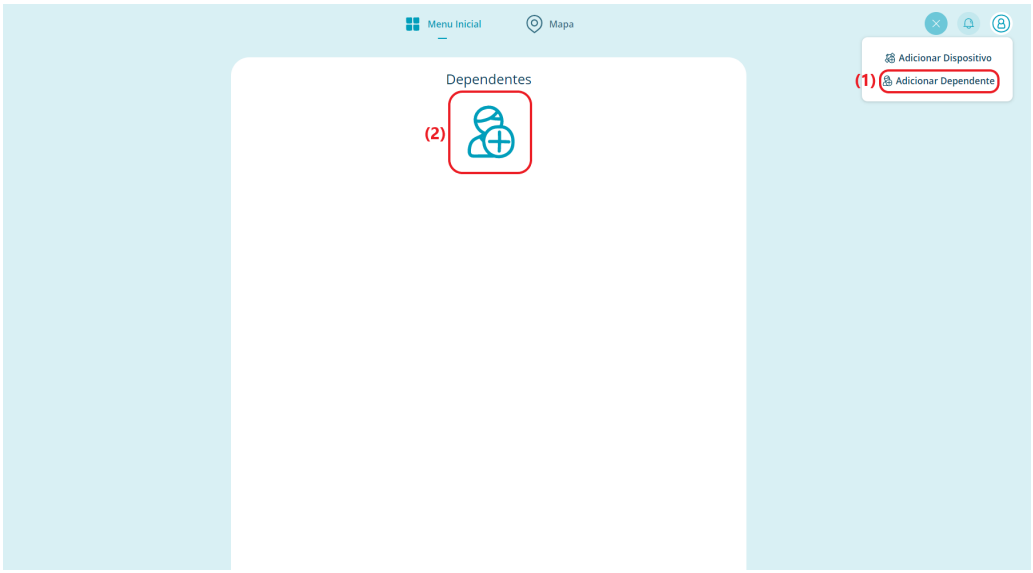


Figura 40: Aplicação - Página Principal sem Dependentes

Para a criação de um dependente o formulário é o que se encontra na figura 41. É possível adicionar uma fotografia de perfil para o dependente, assim como os dados de identificação como nome, data de nascimento, sexo, contacto, email, morada, código postal, cidade ou país e deixar algumas notas como informação adicional. Após preenchidos todos os campos, o botão "Adicionar Dependente" fica disponível para ser carregado e ser enviado o pedido de criação do dependente. À semelhança dos casos anteriores, caso haja algum problema na criação será apresentada uma mensagem de erro, caso contrário, o utilizador é encaminhado para a página principal.

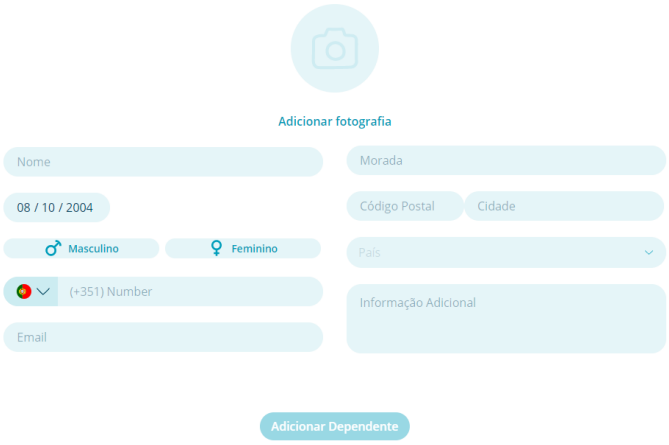


Figura 41: Aplicação - Adicionar Dependente

Após possuir um ou mais dependentes, o cuidador deve atribuir-lhes dispositivos para poder receber localizações e apresentá-las no mapa para cada um dos dependentes. Na figura 42 é apresentado o formulário para adicionar um dispositivo, onde é possível selecionar o tipo (foram criados quatro tipos, mediante os dispositivos fornecidos pela *Neki*), o contacto do cartão SIM inserido no dispositivo (para ser possível entrar em contacto caso necessário) e o IMEI do dispositivo e associar a um dependente. Este último passo não é obrigatório, pelo que é possível adicionar um dispositivo e este não ficar emparelhado a nenhum dependente, sendo isto feito posteriormente.

Figura 42: Aplicação - Adicionar Dispositivo

Após ter dependentes associados, a página inicial de um cuidador será semelhante à apresentada na figura 43, podendo ter mais ou menos dependentes a seu cargo. Um cuidador também pode a qualquer altura aceder às suas definições de conta, carregando em "Gerir Conta" no canto superior direito (1), verificar os dispositivos que tem adicionados (2) ou terminar sessão (3). As funcionalidades "Definições" e "Ajuda" encontram-se desativadas de momento, pois não são funcionalidades essenciais para o funcionamento da plataforma e serão adicionadas posteriormente, quando forem implementadas no *SmartAL*.

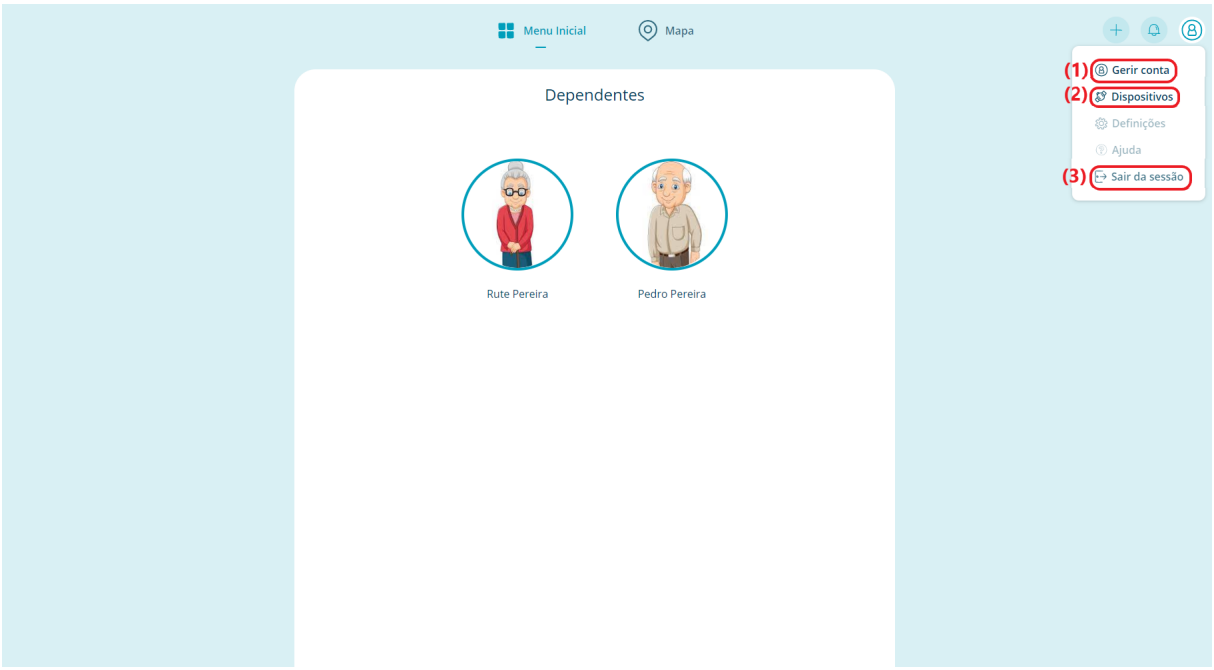


Figura 43: Aplicação - Página Principal com Dependentes

Ao selecionar "Gerir Conta", o utilizador será encaminhado para um formulário com os seus dados, que foram adicionados no primeiro *login*, assim como outros campos que pode adicionar opcionalmente. Na figura 44 é possível visualizar o formulário para editar os dados da conta. Caso o utilizador esteja satisfeito com as alterações efetuadas deve clicar no botão "Guardar" para persistir as alterações.

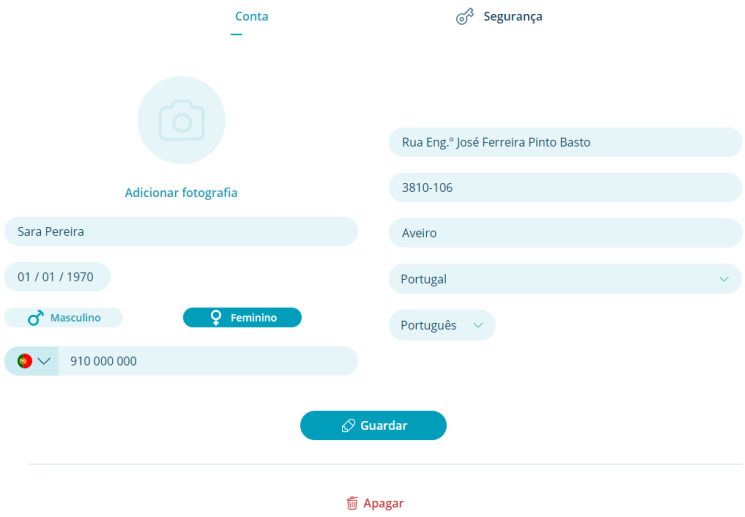


Figura 44: Aplicação - Página Conta Cuidador

Outra funcionalidade que é apresentada ao selecionar "Gerir conta", relacionada com a segurança da conta, é alterar a senha de acesso. Para tal, o utilizador deve inserir a senha atual, a nova senha, e confirmar a mesma, cumprindo os requisitos de segurança indicados.

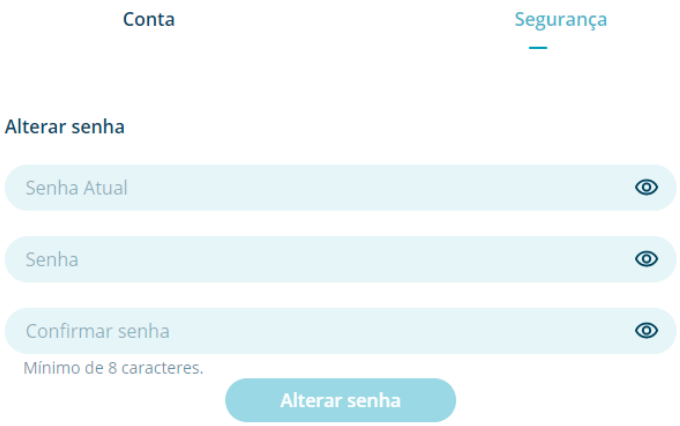


Figura 45: Aplicação - Página Alterar password

Na opção "Dispositivos", é apresentada uma página em que contém a informação relativa aos dispositivos que o cuidador tem associados. Na figura 46, é possível verificar os dispositivos emparelhados, sendo possível saber que dispositivos existem e a que dependentes estão associados. No caso de ter dispositivos não emparelhados, estes são apresentados do lado direito do ecrã, com a respetiva informação. Em ambos os casos o utilizador pode editar as informações dos dispositivos ou eliminá-los.

Figura 46: Aplicação - Página de Dispositivos do Cuidador

Também é possível visualizar informações relativas a um dependente, clicando na sua imagem na página principal. Após realizada essa ação, será apresentada a interface da figura 47. Aqui o cuidador pode também editar os dados relativos aos dependentes, persistir as alterações com o botão "Guardar", ou então eliminar o dependente. Também é possível selecionar no separador superior (1) outras funcionalidades relativas aos dependentes.

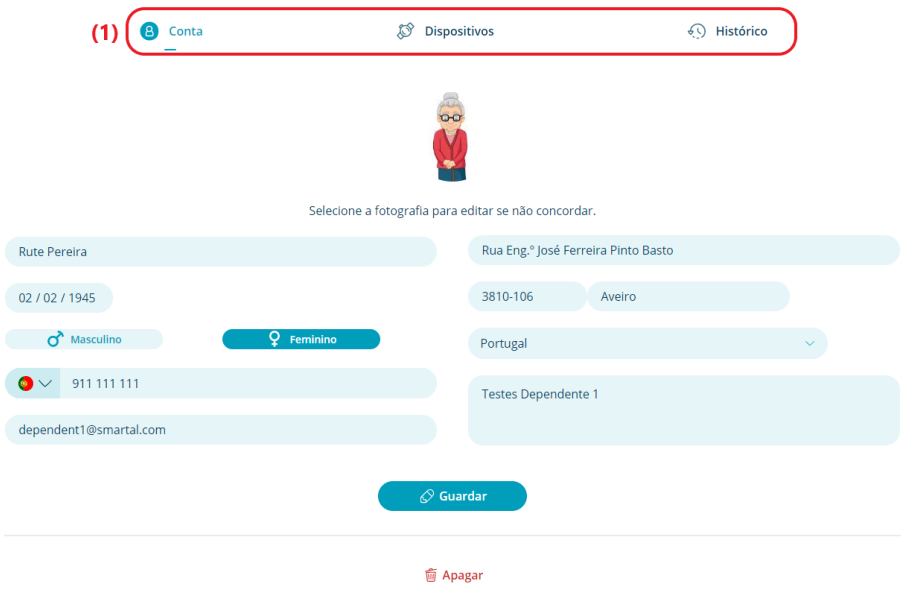


Figura 47: Aplicação - Página de Dados de um Dependente

Ao selecionar o separador "Dispositivos", serão apresentados os dispositivos associados ao dependente em questão, com a possibilidade e editar, eliminar ou então adicionar mais dispositivos, como se pode verificar na figura 48.

Figura 48: Aplicação - Página de Dispositivos de um Dependente

No separador histórico, é possível obter um histórico das notificações recebidas relativas a um dependente. A figura 49 mostra esta funcionalidade, com as notificações recebidas relativas a um dependente criado. Estas serão apresentadas ordenadas por dia e hora, sendo possível pesquisar pelo conteúdo de forma a filtrar as notificações exibidas.

No separador histórico também é possível alterar no menu *dropdown* (1) para o histórico de localizações, onde se pode definir um intervalo de tempo e ver a sua representação no mapa. Na figura 50 é possível observar um exemplo, onde se definiu um intervalo de tempo e foi desenhado no mapa o percurso realizado por um dependente nesse período.

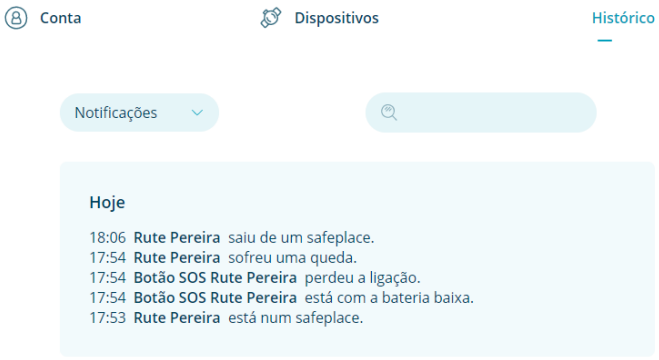


Figura 49: Aplicação - Página de Histórico de Notificações de um Dependente

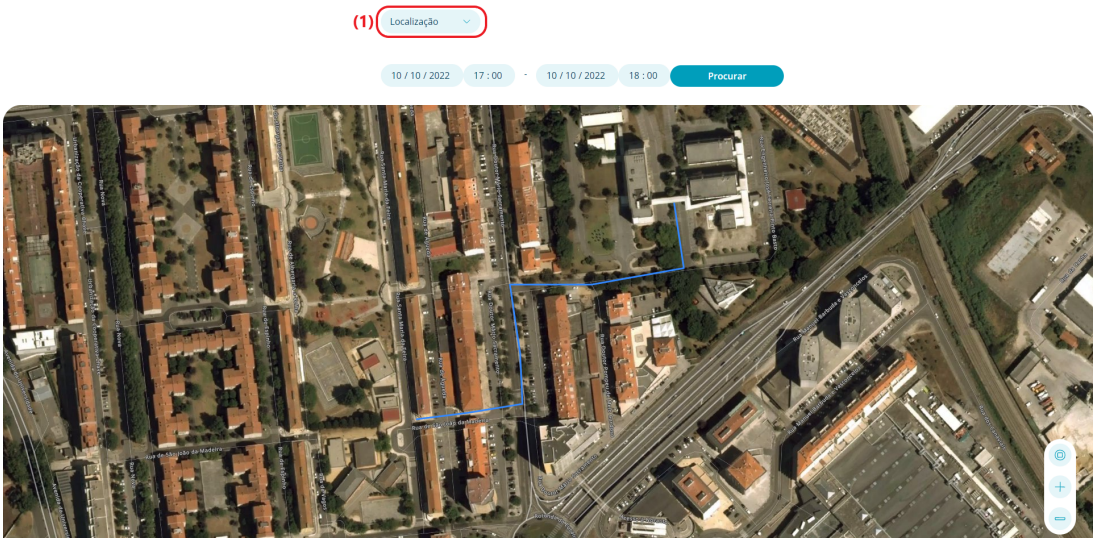


Figura 50: Aplicação - Página de Histórico de Localizações de um Dependente

Uma das opções da página principal é a visualização de um mapa onde se encontram as posições atuais dos dependentes. Esta interface engloba as principais funcionalidades da aplicação, onde é possível visualizar a posição dos dependentes e criar cercas virtuais e lugares seguros.

Na figura 51 é apresentada a interface com o mapa sem qualquer cerca ou lugar seguro definido. No canto superior esquerdo encontra-se um filtro (1), onde se pode ativar e desativar a visualização de cercas virtuais ou lugares seguros, para simplificar a visualização e realização de ações quando o mapa apresentar demasiados elementos para uma compreensão rápida. Por baixo dos botões de filtro encontram-se duas caixas (2) com informações sobre cada um dos dependentes; a cada dependente é atribuída uma cor, que se reflete depois na cor dos elementos relativos a cada um (cercas e lugares seguros). Nesta caixa também é possível centrar o mapa num dos dependentes e adicionar cercas ou lugares habituais. No canto superior direito (3) encontra-se um menu que serve para a edição das cercas virtuais. Quando selecionado, pode-se selecionar uma cerca e editá-la ou removê-la. No canto inferior direito (4) encontram-se os controlos genéricos comuns aos mapas *on-line*, como *zoom in/out* e centrar na localização atual.

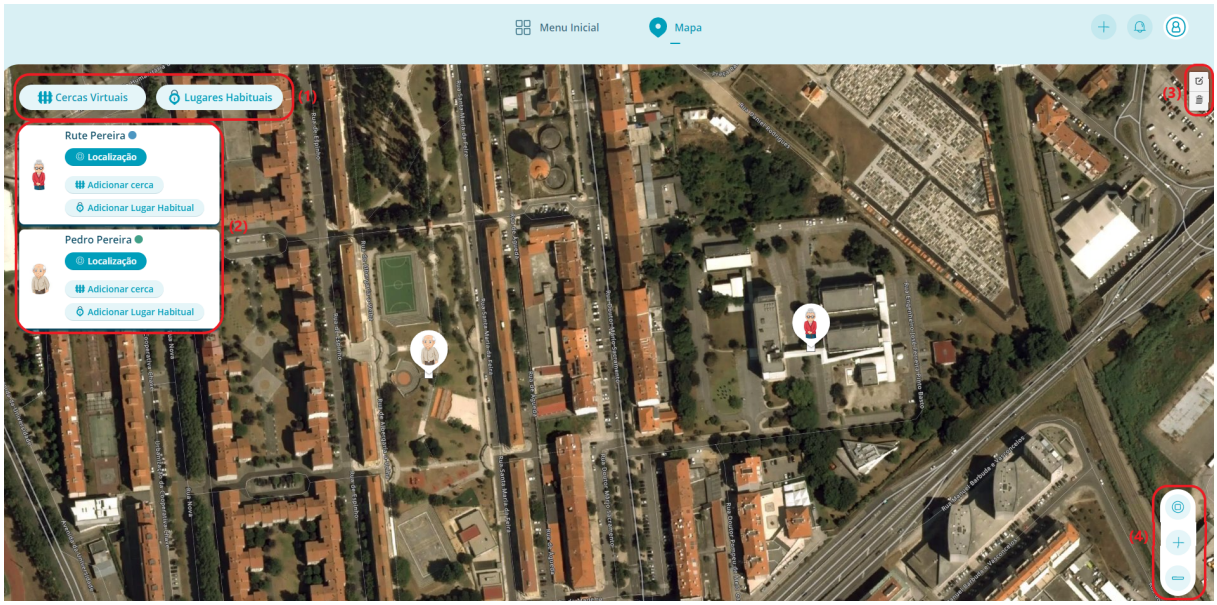


Figura 51: Aplicação - Página de Mapa sem Cercas/Lugares seguros

Na figura 52 é possível observar a mesma página, mas com elementos atribuídos aos dependentes. Como se pode verificar, a dependente criada para esta demonstração, Rute Pereira, possui um lugar habitual denominado "Altice Labs", cuja localização é visível no mapa com a cor atribuída à dependente. Já no caso do segundo dependente criado para o exemplo, Pedro Pereira, é possível verificar que tem uma cerca virtual definida, assim como o nome da mesma ("parque") e que se trata de uma circunferência numa zona de lazer. Ambos os elementos podem ser ativados ou desativados, através do *toggle* (1) à frente do nome do elemento. Esta ativação/desativação reflete-se no estado do elemento, para, quando é feita uma verificação da posição relativa aos elementos de um dependente, o sistema saber quais os elementos que deve verificar e quais não o necessitam por estarem desativados.

Figura 52: Aplicação - Página de Mapa com Cercas/Lugares seguros

Caso o cuidador queira definir uma nova cerca ou lugar habitual para um dependente, deve clicar nos botões indicados para cada uma dessas funcionalidades. No caso dos lugares habituais, após clicar no botão "Adicionar Lugar Habitual", bastará selecionar a posição no mapa onde deseja alocar o elemento. No caso das cercas virtuais, é ativado um menu, que na figura 52 pode ser visualizado no canto superior direito (2), onde o cuidador poderá selecionar a forma que deseja desenhar a cerca. Após concluídas ambas as ações, é ativada uma caixa de diálogo, como a que pode ser observada na figura 53, onde o cuidador deve inserir o nome que deseja atribuir à cerca ou lugar habitual e clicar no botão "Adicionar" para confirmar a sua intenção. Após isso, o elemento fica disponível para ser visualizado no mapa, junto dos restantes.

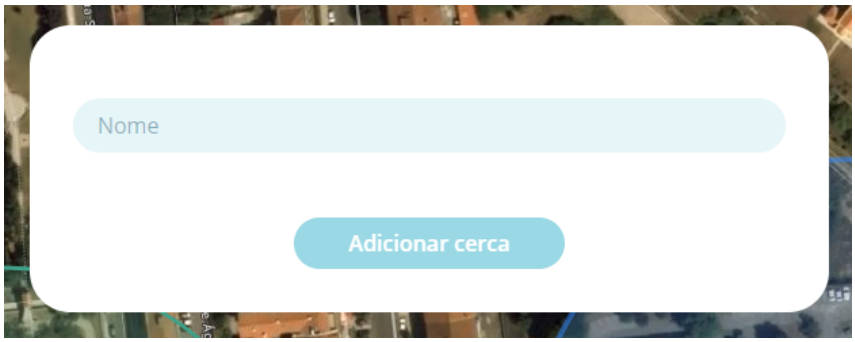


Figura 53: Aplicação - Caixa de Diálogo para criação de uma cerca

Para editar ou eliminar um elemento, basta clicar sobre o mesmo; é aberto um menu que permite efetuar alterações, como a localização ou o nome, ou então proceder à eliminação. Na figura 54 é possível visualizar o menu que permite gerir um lugar habitual.

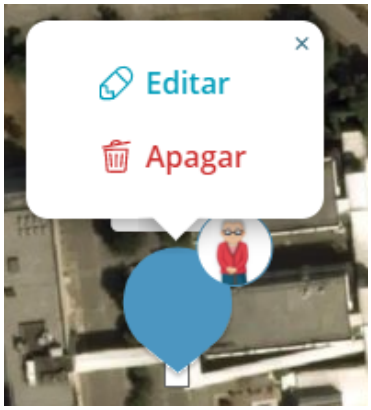


Figura 54: Aplicação - Caixa de Diálogo para editar um Lugar Habitual

Dado que o cuidador pode ter a seu encargo mais do que um dependente, é possível aceder na página principal às notificações de todos os dependentes (em vez de um a um, como na figura 49), carregando no ícone do sino no canto superior direito, onde é indicado o número de notificações por ler. Aqui serão apresentadas as notificações de todos os dependentes, indicando o grau de severidade das mesmas, com a possibilidade de ir para o mapa para verificar a situação do dependente em questão, como é possível observar na figura 55.

Figura 55: Aplicação - Pop-up para notificações

4.4 Deployment

Para facilitar a gestão de todos os microsserviços referidos acima, num ambiente *cloud*, foi escolhido o *Kubernetes*[13]. Esta solução de orquestração de *containers* controla o lançamento das imagens *Docker*, o seu ciclo de vida e, caso necessário, permite escalar serviços para dar resposta a um aumento de carga, lançando várias instâncias da mesma imagem para processamento em paralelo, podendo até fazê-lo de forma automática e dinâmica. Outra vantagem do uso de *Kubernetes* é sua adoção pelas *clouds* públicas mais conhecidas, como é o caso da *Google Cloud*, as quais disponibilizam ambientes geridos pelas próprias, o que reduz o tempo de inicialização e configuração do *cluster Kubernetes* [13].

Para escolher qual *cloud* contratar, a equipa de desenvolvimento do *SmartAL* fez um estudo comparativo sobre os 3 grandes *cloud providers: Google Cloud, Amazon Web Services* e *Microsoft Azure*. Concluiu-se que não existia um vencedor claro, todos têm um excelente catálogo de serviços de qualidade, com preços competitivos. A escolha final acabou por incidir na *Google Cloud*, devido a outros fatores externos, que já foram referidos na secção 4.1.9.

Foi instanciado um *cluster* no *Google Kubernetes Engine*, composto por 3 máquinas virtuais, num total de 48 vCPU e 48 GB RAM. Cada microsserviço é lançado neste ambiente, com o *Kubernetes* a fazer o balanceamento de carga pelas 3 máquinas, de forma a distribuí-la uniformemente.

[13]https://kubernetes.io/

As imagens *Docker* são guardadas no *GitHub Package Registry*, podendo até ser geradas automaticamente, sempre que se lança uma versão, através de *GitHub Actions*.

A configuração destes serviços no *Kubernetes* é feita através de ficheiros *yaml*, que contêm o URL do *GitHub Package Registry* onde reside a imagem *Docker* do serviço, além de outros parâmetros de configuração específicos do serviço. O *Kubernetes* age ao detetar mudanças nestes ficheiros *yaml* – por exemplo, ao mudar o URL da imagem, é lançado um "pod"[14] com a nova imagem, que irá substituir o existente assim que acabe de iniciar, minimizando desta forma o *downtime* (o *pod* contendo a imagem antiga permanece ativo até o novo estar pronto).

[14]Unidade mínima que o *Kubernetes* orquestra; tipicamente um conjunto de um ou mais *containers Docker*

5

Conclusões e Trabalho Futuro

Neste capítulo são apresentadas as conclusões obtidas na realização desta dissertação, os desafios e dificuldades encontrados e como estes foram ultrapassados, sendo realizada uma análise sobre todo o projeto. Também serão discutidos aspetos a serem melhorados e funcionalidades que poderiam ser implementadas num trabalho futuro.

5.1 Conclusões

Esta dissertação teve como objetivo o desenvolvimento de uma aplicação de telelocalização, que permitisse a cuidadores formais ou informais visualizar em tempo real, delinear limites e verificar o estado dos seus dependentes através do simples uso de uma aplicação.

Inicialmente foi necessário realizar um estudo sobre as áreas de *eHealth* e *Assisted Living*, de forma a entender a situação atual – social e tecnológica – nestas áreas e perceber de que forma seria possível a implementação de uma solução deste tipo e qual a melhor maneira de o realizar. Foi realizado também um estudo sobre serviços de telemonitorização que se encontram já disponíveis, assim como sobre dispositivos de localização, de forma a identificar os requisitos necessários e desafios que poderiam surgir no desenvolvimento da solução.

Dada a metodologia utilizada no desenvolvimento deste projeto *DSR*, após o problema identificado e feito o estudo da área de atuação e definidos os objetivos pelo qual se basearia o desenvolvimento, procedeu-se ao design e desenvolvimento, onde se definiu o modelo de dados e a arquitetura a utilizar, e se pôs em prática a implementação da aplicação em si.

Foi na fase inicial da implementação que começaram a surgir as primeiras dificuldades. Inicialmente, surgiu o problema de como obter a localização de dispositivos, pelo que, durante o processo, vários foram testados, de forma a perceber qual fornecia melhores informações e melhor fiabilidade para o sistema. No entanto, após vários testes, vários problemas surgiram, ora na configuração dos dispositivos, que não eram compatíveis ou adaptáveis para uma API, ora na informação que estes forneciam, que não era útil para o caso em questão. Este desafio foi ultrapassado através da realização de uma parceria entre a *Altice Labs* e a *Neki*, que cedeu dispositivos que enviavam eventos com localização para a API

desenvolvida neste projeto, e assim permitiu realizar o desenvolvimento com a utilização de dados de localização reais obtidos pelos dispositivos de teste.

Ultrapassado o primeiro desafio, surgiu um segundo, que seria a implementação da aplicação. O facto de ser o primeiro projeto realizado em ambiente laboral e a utilização de tecnologias novas, nomeadamente de *front-end*, criou uma barreira inicial que resultou num arranque mais lento; no entanto, após a prática e apoio por parte da restante equipa, este problema foi facilmente ultrapassado, sendo que posteriormente o desenvolvimento foi mais simples e rápido.

Durante o desenvolvimento da aplicação, já numa fase final, a mesma passou pelo processo de demonstração e avaliação, de forma a aplicar o artefacto desenvolvido a uma simulação ou experiência e comparar com os requisitos pretendidos. Esta etapa foi realizada numa apresentação ao vivo da aplicação no aniversário da empresa *Altice Labs*, em que soluções de telemonitorização e *eHealth* foram o destaque do evento [12]. A aplicação *Teleloc* teve um papel de destaque, tendo sido possível testar o seu uso e apontar as qualidades e defeitos identificados por uma maior quantidade de utilizadores do que até então, com diversos graus de experiência, e assim melhorar a solução.

Nesta solução é possível destacar a interface gráfica, que cumpre com princípios de usabilidade que permitem ao utilizador menos experiente realizar facilmente as ações pretendidas de forma simples e rápida. De notar também a resposta dada por toda a parte lógica, que cumpre com os requisitos delineados, garantindo a execução de todas as funções de forma fiável e com tempo de resposta baixo.

Por último, é possível afirmar que a aplicação desenvolvida é um produto que irá contribuir ativamente para a proteção e melhoria da qualidade de vida da população a que se destina, garantido um maior acompanhamento por parte dos cuidadores e consequentemente uma maior segurança. Assim, o autor conclui que a realização desta dissertação foi positiva, contribuindo para uma evolução nas áreas de *eHealth* e *Assisted Living*, para a *Altice Labs* e para a sociedade no geral.

5.2 Trabalho Futuro

Os objetivos traçados para a execução deste projeto foram alcançados; no entanto, existe a possibilidade de implementar melhorias ou funcionalidades a acrescentar. De seguida serão apresentadas algumas sugestões para um trabalho futuro na melhoria da aplicação *Teleloc*:

- Melhoria de responsividade da aplicação para dispositivos *mobile*;
- Aprimorar as funções de verificação de posicionamento em relação às cercas virtuais;
- Incluir cenários de deteção de rotinas, com recurso a algoritmos de *machine learning*, de forma a prever eventuais situações de perigo para os dependentes;
- Implementar o perfil de aplicação para os dependentes, para que os mesmos também possam utilizar a aplicação em certos cenários;

Bibliografia

[1] C. H. U. C. da Beira. *TERI - Telemonitorização Doentes em Risco*. [Online; acedido em 23/11/2021]. url: http://www.chcbeira.min-saude.pt/investigacao-no-chcb/projetos/projetos-ue/projeto-teri/ (acedido em 23/11/2021) (ver p. 15).

[2] K. Bittner, I. Spence e I. Jacobson. *Use Case Modeling*. Addison-Wesley object technology series. Addison Wesley, 2003. isbn: 9780201709131. url: https://books.google.pt/books?id=zvxfXvEcQjUC (ver p. 21).

[3] D. M. Cruz. "Literacia em ehealth dos portugueses: estudo exploratório". Em: (fev. de 2013). url: http://hdl.handle.net/10400.6/2942 (ver p. 8).

[4] Deco. *Idosos em lares perderam vitalidade durante a quarentena*. [Online; acedido em 01/12/2021]. url: https://www.deco.proteste.pt/familia-consumo/orcamento-familiar/noticias/idosos-em-lares-perderam-vitalidade-durante-a-quarentena (acedido em 01/12/2021) (ver p. 6).

[5] B. Escalada. *A história do GPS: O aparelho que revolucionou o montanhismo*. [Online; acedido em 28/01/2022]. url: https://blogdescalada.com/historia-do-gps/ (acedido em 28/01/2022) (ver p. 16).

[6] Eurostat. *Number of persons by sex, age groups, household composition and working status (1 000)*. [Online; acedido em 27/11/2021]. url: https://ec.europa.eu/eurostat/databrowser/view/LFST_HHINDWS__custom_1070805/bookmark/table?lang=en&bookmarkId=7362c1ab-c97e-425f-a51e-29e085585173 (acedido em 27/11/2021) (ver p. 5).

[7] Eurostat. *Population structure and ageing*. [Online; acedido em 29/11/2021]. url: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Population_structure_and_ageing (acedido em 29/11/2021) (ver p. 5).

[8] hibernate.org. *What is Object/Relational Mapping?* 2022. url: https://hibernate.org/orm/what-is-an-orm/ (acedido em 21/09/2022) (ver p. 42).

[9] https://applemagazine.com. *Why You Should Use ReactJS*. 2022. url: https://applemagazine.com/why-you-should-use-reactjs/54556 (acedido em 20/10/2022) (ver p. 41).

[10] https://neki.pt/. *Neki*. 2022. url: https://neki.pt/ (acedido em 21/09/2022) (ver p. 53).

[11] https://nodejs.org. *About Node.js*. 2022. url: https://nodejs.org/en/about/ (acedido em 21/09/2022) (ver p. 41).

[12] https://telecom.pt. *A Saúde do futuro é made in Portugal, made by Altice Labs*. 2022. url: https://www.telecom.pt/pt-pt/media/comunicados/Paginas/2022/maio/A-Sa%C3%BAde-do-futuro-%C3%A9-made-in-Portugal,-made-by-Altice-Labs-.aspx (acedido em 31/05/2022) (ver p. 80).

[13] https://www.ibm.com. *Top 7 Benefits of Kubernetes*. 2022. url: https://www.ibm.com/cloud/blog/top-7-benefits-of-kubernetes (acedido em 21/09/2022) (ver p. 76).

[14] https://www.infoworld.com. *PostgreSQL benefits and challenges: A snapshot*. 2021. url: https://www.infoworld.com/article/3619531/postgresql-benefits-and-challenges-a-snapshot.html (acedido em 21/09/2022) (ver p. 43).

[15] https://www.rabbitmq.com/. *RabbitMQ*. 2022. url: https://www.rabbitmq.com/ (acedido em 21/09/2022) (ver p. 43).

[16] INE. *Taxa de analfabetismo (%) por Local de residência (à data dos Censos 2011) e Sexo; Decenal*. [Online; acedido em 10/11/2021]. url: https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_indicadores&indOcorrCod=0006731&contexto=bd&selTab=tab2&xlang=PT (acedido em 10/11/2021) (ver p. 8).

[17] iSalus. *How Remote Patient Monitoring Helps Seniors Age in Place*. [Online; acedido em 11/08/2022]. url: https://isalushealthcare.com/blog/how-remote-patient-monitoring-helps-seniors-age-in-place/ (acedido em 11/08/2022) (ver p. 1).

[18] A. Labs. *eHealth Smart Assisted Living*. [Online; acedido em 23/11/2021]. url: https://www.alticelabs.com/products/ehealth-smart-assisted-living/ (acedido em 23/11/2021) (ver pp. 1, 16).

[19] A. Labs. *Site Altice Labs*. [Online; acedido em 30/12/2021]. url: https://www.alticelabs.com/pt/ (acedido em 30/12/2021) (ver p. 2).

[20] S. Moola et al. "Vital signs to monitor hospital patients: a systematic review." Em: *JBI library of systematic reviews* 6 4 Suppl (2008), pp. 1–11 (ver p. 12).

[21] C. D. Norman e H. A. Skinner. "eHealth Literacy: Essential Skills for Consumer Health in a Networked World". Em: *J Med Internet Res* 8.2 (jun. de 2006), e9. issn: 1438-8871. doi: 10.2196/jmir.8.2.e9. url: http://www.ncbi.nlm.nih.gov/pubmed/16867972 (ver p. 8).

[22] J. de Notícias. *Coletes melhoram exibição dos craques do futebol*. [Online; acedido em 15/11/2021]. url: https://www.jn.pt/desporto/coletes-melhoram-exibicao-dos-craques-9292447.html (acedido em 15/11/2021) (ver p. 10).

[23] H. O'Neil e M. Nunes. *FUNDAMENTAL DE UML*. 7ª ed. Accessed: 2021-03-29. FCA - Editora de informática, Lda., 2011 (ver p. 21).

[24] Onu. *Envelhecimento ONU*. [Online; acedido em 29/11/2021]. url: https://unric.org/pt/envelhecimento/ (acedido em 29/11/2021) (ver p. 5).

[25] K. Peffers et al. "The design science research process: A model for producing and presenting information systems research". Em: *Proceedings of First International Conference on Design Science Research in Information Systems and Technology DESRIST* (fev. de 2006) (ver pp. 2, 3).

[26] A. Portugal. *Alzheimer. E se um dia lhe disserem que...* [Online; acedido em 05/12/2021]. url: https://alzheimerportugal.org/pt/news_text-78-11-1085-alzheimer-e-se-um-dia-lhe-disserem-que (acedido em 05/12/2021) (ver p. 7).

[27] Reactjs.org. *React - A JavaScript library for building user interfaces*. 2019. url: https://reactjs.org/ (acedido em 21/09/2022) (ver p. 40).

[28] E. N. de Saúde Pública. *Covid-19: idade é maior factor de risco para internamento ou morte*. [Online; acedido em 05/12/2021]. url: https://barometro-covid-19.ensp.unl.pt/covid-19-idade-e-maior-factor-de-risco-para-internamento-ou-morte/ (acedido em 05/12/2021) (ver p. 7).

[29] SNS. *Estruturas Residenciais para idosos*. [Online; acedido em 01/12/2021]. url: https://www.sns.gov.pt/noticias/2020/08/12/estruturas-residenciais-para-idosos/ (acedido em 01/12/2021) (ver p. 6).

[30] SNS. *Hospital Distrital de Santarém arranca projeto para doentes com DPOC*. [Online; acedido em 23/11/2021]. url: https://www.sns.gov.pt/noticias/2021/03/01/telemonitorizacao-domiciliaria/ (acedido em 23/11/2021) (ver p. 14).

[31] SNS24. *Febre*. [Online; acedido em 23/11/2021]. url: https://www.sns24.gov.pt/tema/sintomas/febre/#sec-0 (acedido em 23/11/2021) (ver p. 12).

[32] SNS24. *Hipertensão*. [Online; acedido em 23/11/2021]. url: https://www.sns24.gov.pt/tema/doencas-do-coracao/hipertensao-arterial/#sec-1 (acedido em 23/11/2021) (ver p. 12).

[33] SNS24. *Utilizar a aplicação móvel Telemonit SNS 24*. [Online; acedido em 23/11/2021]. url: https://www.sns24.gov.pt/servico/utilizar-a-aplicacao-movel-telemonit-sns-24/#sec-0 (acedido em 23/11/2021) (ver p. 15).

[34] S. Tereso e M. Moreira. "Envelhecer Sozinho - Um Estudo de Caso No Interior de Portugal". Em: *Egitania Sciencia* 28 (2020), pp. 25–41. issn: 1646-884 (ver p. 6).