Universidade da Beira Interior
Departamento de Informática

LIFE – A Mobile App for Immediate First-Aid and Emergency Location Sharing

Elaborado por:

Xavier José André Tacanho

Orientador:

Professor Doutor Tiago M. C. Simões

8 de julho de 2025

Agradecimentos

É com grande satisfação e respeito que expresso o meu profundo agradecimento a todos os que, de forma direta ou indireta, contribuíram para a concretização deste projeto. Em primeiro lugar, gostaria de manifestar a minha profunda gratidão ao meu orientador, Tiago M.C. Simões, pela orientação exemplar, dedicação e apoio contínuo ao longo de todo o processo de desenvolvimento deste trabalho. O seu conhecimento, a sua experiência e a sua disponibilidade foram fundamentais para o enriquecimento deste estudo, sendo as suas valiosas sugestões e críticas construtivas essenciais para o aprimoramento e a qualidade final deste projeto. Agradeço igualmente à Universidade da Beira Interior, situada na cidade da Covilhã, pela formação sólida e pelo ambiente académico propício ao desenvolvimento de investigação, que me permitiu atingir os objetivos propostos. A todos os docentes e colaboradores da instituição que, ao longo do meu percurso académico, contribuíram para o meu crescimento intelectual, as minhas mais sinceras palavras de reconhecimento. Aos meus colegas de curso, que constantemente partilharam os seus conhecimentos, experiências e ideias, sou também profundamente grato. As discussões académicas e as trocas de experiências foram de extrema importância para o enriquecimento deste trabalho e para a minha evolução enquanto estudante e profissional. À minha família, expresso um agradecimento especial, pela compreensão, paciência e apoio incondicional durante todo o processo de realização deste trabalho. A vossa presença constante, tanto em momentos de dificuldades como de êxito, foi fundamental para a elaboração deste projeto. Sem o vosso apoio emocional, nunca teria sido possível alcançar este resultado. Agradeço, igualmente, a todos os que, garantidamente, colaboraram de forma direta ou indireta com este projeto, seja mediante apoio logístico, disponibilidade para esclarecimentos ou contribuição com informações de relevância para o estudo. A todos esses, o meu reconhecimento. Por fim, expresso os meus agradecimentos a todas as entidades externas, organismos e instituições que, com a sua colaboração, permitiram a recolha de dados e informações cruciais para o desenvolvimento deste trabalho. A todos os que, de alguma forma, contribuíram para a realização deste projeto, o meu profundo e sincero agradecimento.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

RGPD	Regulamento Geral sobre a Proteção de Dados
PSAP	*Public Safety Answering Points*
XML	*Extensible Markup Language*
AML	*Advanced Mobile Location*
PHP	*Hypertext Preprocessor*
IoT	*Internet of Things*
PK	*Primary Key*
FK	*Forgein Key*

Capítulo

1

Introdução

1.1 Enquadramento

Em emergências, a precisão na comunicação da localização da vítima é um fator crítico para a eficácia do socorro. No contexto atual em Portugal, os serviços de emergência dependem essencialmente da descrição verbal fornecida pelo próprio utilizador, o que pode originar atrasos significativos, sobretudo em zonas remotas, autoestradas ou locais desconhecidos.

O projeto **LIFE** surge como uma resposta tecnológica a essa limitação, propondo o desenvolvimento de uma aplicação móvel que combina **orientações imediatas de primeiros socorros** com a **partilha automática e em tempo real da localização exata do utilizador**. Utilizando tecnologias como **geolocalização por GPS** e **comunicação em *cloud***, a aplicação visa reduzir os tempos de resposta e aumentar a precisão das operações de emergência.

Este trabalho foi realizado no âmbito da unidade curricular de Projeto Final de Curso do Mestrado em Informática Web, Móvel e na Nuvem da Universidade da Beira Interior (UBI), integrando competências nas áreas de desenvolvimento móvel, engenharia de software, e comunicação em tempo real.

Com vista à transparência e à continuidade do desenvolvimento, o código-fonte do projeto **LIFE** foi disponibilizado publicamente no *GitHub*. Toda a implementação — desde a interface móvel até à comunicação com o servidor — pode ser consultada em:
LIFE-A-Mobile-App-for-Immediate-First-Aid-and-Emergency-Location-Sharing

1.2 Motivação

A principal motivação para o desenvolvimento da aplicação **"LIFE"** resulta da necessidade urgente de melhorar a eficácia e rapidez da resposta a emergências em Portugal. Atualmente, os serviços de emergência dependem, em grande medida, da descrição verbal do local fornecido pela vítima ou testemunha, o que pode originar atrasos críticos — sobretudo em zonas remotas, autoestradas ou locais pouco familiares.

Esta fragilidade no processo de localização motivou a criação de uma solução tecnológica que pudesse colmatar essa lacuna. O **projeto "LIFE"** visa proporcionar uma resposta concreta a este problema, por meio de uma aplicação móvel que conjuga duas funcionalidades fundamentais:

- **Orientações imediatas de primeiros socorros**, adaptadas a diferentes cenários clínicos;
- **Partilha automática e em tempo real da localização exata do utilizador**, tanto com os serviços de emergência como com contactos pessoais de confiança.

Esta abordagem permite agilizar o processo de socorro, reduzindo significativamente os tempos de resposta e, potencialmente, salvando vidas.

Além da sua relevância prática e impacto social evidente, o projeto representa também um desafio pessoal e académico. O interesse em desenvolver uma solução com utilidade real na sociedade permitiu-me aplicar e consolidar conhecimentos nas áreas do desenvolvimento mobile, *cloud computing*, design de interfaces intuitivas e integração com serviços de localização por GPS.

O facto de o projeto poder ser integrado com os **serviços de emergência nacionais**, e estar alinhado com normas europeias como o ***AML (Advanced Mobile Location)***, reforça o seu valor enquanto iniciativa de **utilidade pública e inovação tecnológica**.

1.3 Objetivos

O projeto "**LIFE**" tem como objetivos principais:

- Reduzir o tempo de resposta das equipas de emergência através da partilha automática e precisa da localização do utilizador em causa;
- Disponibilizar instruções de primeiros socorros para cenários médicos críticos enquanto se aguarda por ajuda profissional;

- Criar uma aplicação intuitiva e acessível, mesmo sob condições de stress
- Assegurar transmissão de dados segura e em tempo real, através de integração com plataformas em cloud;
- Avaliar a eficácia da aplicação na redução de tempos de resposta e melhoria do acesso a cuidados imediatos.

1.4 Organização do Documento

De modo a refletir o trabalho feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento;
2. O segundo capítulo – **Background e Estado de Arte** – analisa soluções tecnológicas existentes e fundamentos técnicos relacionados com geolocalização, aplicações de emergência e primeiros socorros;
3. O terceiro capítulo — **Engenharia de Software** -– descreve os requisitos funcionais e não funcionais do sistema, os casos de uso previstos e o modelo de dados adotado, nomeadamente o modelo entidade-relacionamento e o esquema relacional da base de dados;
4. O quarto capítulo — **Tecnologias e Ferramentas** -– apresenta os ambientes de desenvolvimento e tecnologias utilizadas na construção da aplicação;
5. O quinto capítulo — **Implementação e Testes** -– detalha a arquitetura da aplicação, os componentes desenvolvidos e os testes realizados;
6. O sexto capítulo –- **Demonstração e Validação** -– mostra capturas de ecrã reais da aplicação **LIFE** em funcionamento, contextualizando cada funcionamento e demonstrando a sua execução correta;
7. O quinto capítulo — **Conclusões e Trabalho Futuro** -– sintetiza os principais resultados alcançados, identifica as limitações do trabalho realizado e propõe possíveis melhorias e desenvolvimentos futuros;
8. Por fim, é apresentada a bibliografia utilizada ao longo do relatório, com as fontes consultadas para fundamentar teoricamente o projeto.

Capítulo

2

Background e Estado da Arte

2.1 Introdução

Para compreender os requisitos que o projeto precisa, é necessário um estudo mais aprofundado acerca das técnicas que já foram elaboradas, as tecnologias atualmente em uso e o seu modo de funcionamento. Esta secção de estado de arte pretende elucidar mais acerca do tema de geolocalização e as boas práticas que acompanham a sua implementação, tendo sempre em mente a segurança e privacidade dos utilizadores. Para tal, serão apresentadas aplicações já empregues no mercado, explorando como podem afetar o projeto **LIFE** no seu desenvolvimento.

2.2 Tecnologias de Geolocalização e AML

Uma das inovações que tem entrado de rompante e a revolucionar o atendimento de chamadas de emergências é a utilização do ***Advanced Mobile Location (Advanced Mobile Location (AML))***.

Esta tecnologia tem sido crucial para melhorar a precisão e a rapidez na transmissão de dados da localização dos utilizadores, com especial foco no contexto de chamadas de emergências para o número de emergência, como o 112.

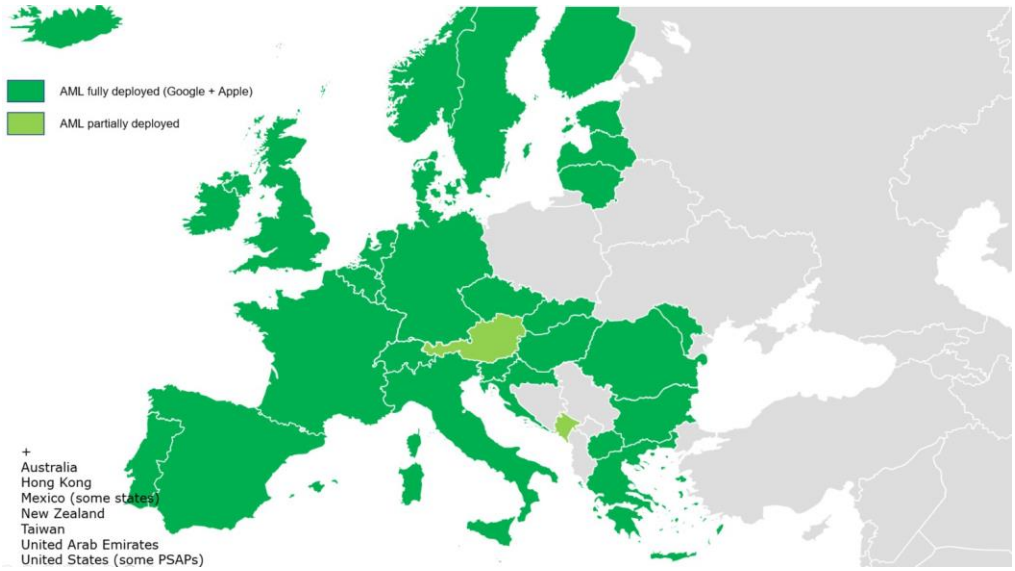

Figura 2.1: Disponibilidade do **Advanced Mobile Location (AML)** na Europa e em alguns países fora do continente

O mapa presente na Figura 2.1ilustra, a verde-escuro, os países europeus onde o ***AML*** se encontra plenamente implementado (abrangendo tanto dispositivos ***Google*** como ***Apple***). A verde-claro, surgem os países que já iniciaram a adoção parcial da tecnologia, prevendo-se uma expansão progressiva do serviço. Além disso, alguns territórios fora da Europa, como a Austrália, Hong Kong, México (em alguns estados), Nova Zelândia, Taiwan, Emirados Árabes Unidos e certos estados dos Estados Unidos (designadamente onde os ***Public Safety Answering Points*** **(PSAP)** – suportam a tecnologia), também contam com a implementação do ***AML*** ou estão em processo de integração.

Esta cobertura crescente reforça a eficácia do atendimento de emergências em diversas regiões, permitindo o envio automático e preciso da localização durante chamadas de socorro.

2.2.1 Funcionamento e Implementação

Imaginemos que, ao fazer uma chamada de emergência, nos encontramos num território desconhecido, sem termos referências visuais tais como nomes de rua ou pontos de referência (isto é, como se estivéssemos numa região sem referências geográficas claras). Nesta situação, seria impossível descrever com precisão a nossa geolocalização. É aqui que surge o valor do ***Advanced Mobile Location (AML)***.

Quando uma pessoa efetua uma chamada de emergência, o dispositivo móvel, de forma automática e em segundo plano sem que o seu utilizador dê conta, ativa os seus sistemas de localização. Mesmo que a pessoa não saiba onde se encontra, o ***AML*** junta informações provenientes de várias fontes para determinar a sua posição exata.

Estas fontes incluem:

- **GPS:** Fornece a posição geográfica com elevada precisão, sendo especialmente eficaz em ambientes exteriores.
- **Redes Wi-Fi:** Complementam o sinal de ***GPS***, sobretudo em zonas urbanas ou interiores, onde o sinal de satélite poderá ser menos fiável.
- **Torres de Telefonia Móvel:** Servem de apoio para estimar a localização em áreas onde os sinais de ***GPS*** e ***Wi-Fi*** possam ser insuficientes.

Desta maneira, mesmo que o utilizador se encontre "perdido", o ***AML*** assegura que a localização possa ser identificada com rapidez e precisão, enviando-a automaticamente para os serviços de emergência por meio de um SMS ou de pacotes de dados. Esta funcionalidade elimina a necessidade de o utilizador descrever verbalmente a sua localização, reduzindo significativamente o tempo de resposta e minimizando potenciais erros de comunicação.

2.2.2 Vantagens em Relação aos Sistemas Tradicionais

Os métodos tradicionais de "partilha" de localização dependiam de descrições verbais por parte do utilizador, o que frequentemente resultava em imprecisões e atrasos. Em contraste, o ***AML*** elimina a necessidade de intervenção manual e verbal, minimizando desta maneira os erros que poderiam ser tomados pelo utilizador, proporcionando assim uma resposta mais célere e eficaz por parte dos serviços de emergência.

Além do referido, o ***Google Emergency Location Service (ELS)*** representa uma solução complementar, permitindo que os dispositivos Android utilizem múltiplas fontes de dados para melhorar ainda mais a precisão e fiabilidade no âmbito do envio automático da localização durante chamadas de emergência. Tais sistemas reforçam a capacidade dos serviços de emergência de obter informações precisas, independentemente das condições ambientais.

2.2.3 Desafios e Considerações Técnicas

Embora a implementação do ***AML*** represente um avanço significativo, existem desafios que devem ser tidos em conta:

- **Precisão de Ambientes Urbanos:** Em áreas densamente construídas, a precisão do ***GPS*** pode ser comprometida. A integração com Wi-Fi e dados das torres de telefonia ajuda em cota parte a minimizar este problema.
- **Privacidade e Segurança:** O envio automático da localização vem levantar questões de privacidade. Embora o ***AML*** esteja desenhado para que este seja somente ativo aquando da duração das chamadas de emergência, garantindo assim que os dados sejam transmitidos segura e somente para as entidades competentes.
- **Compatibilidade de Dispositivos:** Nem todos os dispositivos suportam o protocolo ***AML***. Contudo, com a crescente adesão de *smartphones* modernos, como ***Androids*** quanto ***iOS,*** ampliando a sua implementação em diversos mercados, incluindo Portugal, onde o ***AML*** está conforme as regulamentações europeias.

2.2.4 Impacto na Eficiência dos Serviços de Emergência

A adoção do protocolo ***AML*** tem um impacto direto na capacidade dos serviços de emergência de responderem com maior rapidez e precisão. Ao receberem a localização exata do utilizador em questão, quase instantaneamente, os operadores podem assim orientar as equipas de socorro de uma forma mais eficaz, contribuindo desta forma para a diminuição dos tempos de resposta e deste modo conseguir-se salvaguardar vidas.

Sucintamente, a integração de tecnologias de geolocalização avançadas, como o ***Advanced Mobile Location (AML)*** representa uma evolução crítica na forma como as emergências são geridas. Ao conseguirmos combinar diversas fontes de localização e fazendo a automatização do envio da localização para as entidades competentes, permite-nos então fazer uma base mais sólida para se conseguir obter uma resposta mais célere e eficaz dos serviços de emergência, constituindo um bom enquadramento para o desenvolvimento de aplicações como a **LIFE**.

2.3 Aplicações Móveis de Emergência

Existem diversas aplicações que têm sido desenvolvidas com o intuito de melhorar a resposta em emergências. Entre as mais relevantes, destacam-se:

- **NoonLight:** Anteriormente conhecida como **SafeTrek**, esta aplicação permitia que o utilizador se solicitasse assistência de emergência com um

simples toque, partilhando automaticamente a sua localização em tempo real com as entidades competentes.

- **ProCiv Madeira:** Aplicação oficial do Serviço Regional de Proteção Civil da Madeira, que integra funcionalidades de geolocalização automática e envio de alertas em tempo real, facilitando a resposta a incidentes na região.
- **Cruz Vermelha Portuguesa:** Estas aplicações oferecem conteúdos educativos relativos aos primeiros socorros, como, em algumas implementações, também integram funcionalidades de geolocalização para apoiar a ação dos serviços de emergência.

Estas aplicações demonstram que a convergência entre a partilha automática de localização e a disponibilização de conteúdos de primeiros socorros pode otimizar significativamente o tempo de resposta e a eficácia do atendimento pré-hospitalar, podendo assim diminuir o risco de vida do utilizador.

2.4 Integração de Orientações de Primeiros Socorros

A integração de instruções de primeiros socorros constitui uma componente inovadora no domínio das aplicações de emergência. Em situações críticas, a rapidez e a clareza das orientações podem fazer a diferença entre uma intervenção eficaz e um atraso que comprometa a integridade física do acidentado.

Assim, o projeto **LIFE** propõe disponibilizar conteúdos educativos de maneira a visar a orientação do utilizador, de forma clara e imediata, para a execução de procedimentos de socorro em diversas emergências.

2.4.1 Abordagem e Funcionalidades

- **Conteúdos Educativos Detalhados:** A aplicação disponibilizará instruções passo a passo para a realização de procedimentos de primeiros socorros, adaptadas a diferentes cenários clínicos, tais como paragens cardíacas, hemorragias e acidentes de viação. Este conteúdo é elaborado com base em protocolos reconhecidos e atualizado conforme as diretrizes das entidades de saúde competentes.
- **Interface Intuitiva:** O design da interface foi concebido para permitir um acesso rápido e sem complicações às instruções, mesmo sob condições de stress elevado. A disposição dos conteúdos e o uso de recursos

visuais (como ilustrações e vídeos explicativos) auxiliam o utilizador a compreender rapidamente as ações necessárias.

- **Contextualização das Emergências:** As instruções são apresentadas de forma contextualizada, permitindo ao utilizador identificar rapidamente qual o procedimento adequado para a situação específica que se encontra a enfrentar. Esta abordagem visa reduzir o tempo de decisão e facilitar a execução correta dos passos indicados.

2.5 Análise Comparativa

A Tabela 2.1 apresenta as principais características de cada funcionalidade:

Tecnologia	Facilidade de Uso	Alta Performance	Suporte Comunitário	Segurança e Privacidade	Escalabilidade
Advanced Mobile Location (AML)	✓	✓	✓	✓	✓
Google Emergency Location Service (ELS)	✓	✓	✓	✓	✓
Cruz Vermelha Portuguesa APP	✓		✓	✓	
ProCiv Madeira APP	✓	✓		✓	✓
iOS Core Location	✓	✓	✓	✓	✓

Tabela 2.1: Tabela Comparativa de Funcionalidades

2.6 Conclusões

Em suma, o levantamento do estado da arte revela que a integração de tecnologias avançadas de geolocalização, como o **Advanced Mobile Location (AML)**, com a disponibilização de instruções de primeiros socorros, representa um avanço crucial na resposta a emergências. As soluções atualmente implementadas demonstram que a automação no envio da localização — evitando

a imprecisão inerente às descrições verbais — é fundamental para reduzir os tempos de resposta dos serviços de emergência.

Adicionalmente, as aplicações móveis analisadas, como o ***NoonLight***, a **ProCiv Madeira** e as iniciativas da **Cruz Vermelha Portuguesa**, evidenciam que a combinação de conteúdos educativos de primeiros socorros com funcionalidades de partilha automática de localização não só melhora a eficácia do atendimento pré-hospitalar, como também confere ao utilizador maior autonomia e segurança em situações críticas. A interface intuitiva e a contextualização das instruções permitem uma rápida identificação e execução dos procedimentos necessários, minimizando erros e atrasos.

Desta forma, a convergência destas tecnologias e práticas estabelece uma base sólida para o desenvolvimento do projeto **LIFE**, posicionando-o de forma inovadora e alinhada com as melhores práticas internacionais. Este estado da arte não só orienta a implementação de um sistema que visa salvar vidas, como também abre portas para futuras integrações com tecnologias emergentes, como dispositivos ***Internet of Things*** **(IoT)** e algoritmos de inteligência artificial, potencializando ainda mais a resposta a emergências.

Capítulo

3

Engenharia de Software

3.1 Introdução

Neste capítulo, vamos apresentar detalhadamente os requisitos e os casos de uso do projeto **LIFE**. Aqui definimos não só o que a aplicação deve fazer (requisitos funcionais), mas também os aspetos de qualidade que precisa de garantir (requisitos não funcionais). Além disso, ilustramos através dos casos de uso como os utilizadores irão interagir com o sistema, fornecendo uma visão prática de como o **LIFE** vai contribuir para agilizar o atendimento em emergências e promover a aprendizagem em primeiros socorros.

3.2 Requisitos Funcionais

Para o projeto **LIFE** cumprir o seu objetivo de oferecer uma resposta rápida em emergências e de facilitar o acesso a conteúdos educativos de primeiros socorros, a aplicação deve atender a vários requisitos funcionais.

Em termos mais simples, isto significa que a interface deverá ser o mais intuitiva possível, permitindo assim ao utilizador registar-se, atualizar o seu perfil, acionar uma emergência com um simples toque (que efetuará a partilha automática da sua localização) e consultar orientações práticas e interativas. Estes requisitos garantem que a app é útil e eficaz na resolução de situações críticas de emergência.

- **RF01:** Registo e Login: Permitir que os utilizadores se registem e autentiquem credenciais seguras (e.g. email e password);

- **RF02:** Gestão de Perfil: Possibilidade de atualizar os dados pessoais, incluindo informações elementares, contactos de emergência e morada (partilhada entre vários utilizadores);
- **RF03:** Chamada de Emergência: O utilizador poderá acionar o botão de emergência, que automaticamente envia a sua localização (utilizando tecnologias semelhantes à **AML**) para os serviços de emergência e contactos pré-definidos;
- **RF04:** Histórico de Chamadas: Registar todas as ocorrências de emergências, armazenando data, hora, local e estado da chamada;
- **RF05:** Acesso Rápido: Disponibilizar instruções de primeiros socorros com conteúdos interativos (textos, imagens e vídeos) para situações como paragem cardíaca, hemorragias, etc;
- **RF06:** Contextualização: As instruções devem ser apresentadas de forma contextualizada, facilitando a identificação do procedimento adequado à situação;
- **RF07:** Módulos de Treino: Disponibilizar cursos e conteúdos teóricos de primeiros socorros;
- **RF08:** Histórico de Formação: Registar a participação dos utilizadores nos cursos e os resultados dos quizes;
- **RF09:** Notificações e Alertas: Enviar notificações para os contactos de emergência e para os serviços competentes quando uma emergência for reportada;
- **RF10:** *Logs* de Atividades: Manter um histórico detalhado das interações(acessos, emergências reportadas, cursos realizados) para análise e auditoria.

3.3 Requisitos Não Funcionais

Para além das funcionalidades, é crucial que o sistema **LIFE** seja confiável, seguro e eficiente. Os requisitos não funcionais definem padrões de qualidade que a aplicação deve cumprir. Isto inclui operar com baixa latência, oferecer alta disponibilidade, proteger os dados sensíveis dos utilizadores por meio de encriptação robusta e ser compatível com diversos dispositivos móveis. Em suma, estes requisitos asseguram que a experiência do utilizador seja consistente, segura e de alto desempenho, mesmo sob condições adversas.

- **RNF01:** Baixa Latência: O sistema deve responder rapidamente, especialmente em emergências;
- **RNF02:** Escalabilidade: A infraestrutura deve suportar um número crescente de utilizadores e acessos simultâneos;
- **RNF03:** Proteção de Dados: Garantir que os dados sensíveis (como a localização, dados pessoais, histórico médico) sejam armazenados e transmitidos de forma encriptada, consoante o **Regulamento Geral sobre a Proteção de Dados (RGPD)**;
- **RNF04:** Autenticação Segura: Utilizar métodos robustos de autenticação e armazenamento seguro de senhas (ex.: hash com salt);
- **RNF05:** Integridade e Confidencialidade: Assegurar que somente entidades autorizadas tenham acesso aos dados críticos;
- **RNF06:** Interface Intuitiva: A app deve ter uma interface simples e clara, especialmente importante em momentos de stress, com fácil acesso às funções de emergência e orientação;
- **RNF07:** Compatibilidade Multiplataforma: Suporte a diferentes dispositivos móveis (Android e iOS) e a variados tamanhos de ecrã;
- **RNF08:** Alta Disponibilidade: O sistema deve estar disponível 24 sobre 7, especialmente em situações críticas de emergência;
- **RNF09:** Tolerância a Falhas: Implementação de mecanismos de backup e redundância para evitar perda de dados e assegurar a continuidade do serviço;
- **RNF10:** Modularidade: Arquitetura deve permitir fácil manutenção e a possibilidade de futuras expansões (ex.: Integração com novos sensores ou dispositivos ***IoT***);
- **RNF11:** Manutenção e Evolução: Garantir que todo o sistema está bem documentado para facilitar a manutenção e atualizações futuras.

3.4 Diagramas de Casos de Uso

Os casos de uso são cenários práticos que demonstram como os utilizadores irão interagir com o sistema **LIFE**. Nesta secção, descrevemos situações reais — desde o registo e autenticação do utilizador até à ativação de chamadas de emergência e consulta de instruções de primeiros socorros, bem

como a realização de cursos e quizes. Estes exemplos ajudam a visualizar o fluxo de operações que o sistema deverá suportar, evidenciando como cada funcionalidade contribui para uma resposta rápida em emergências e para a promoção do conhecimento em primeiros socorros.

1. **Registo e Autenticação:**

   Imagine que, ao abrir a aplicação pela primeira vez, o utilizador encara uma interface intuitiva que o convida a criar uma conta. Ele preenche os seus dados básicos -- nome, data de nascimento, e uma senha segura -- e, num instante, tem acesso à sua área pessoal. Este cenário abrange tanto o registo como a autenticação, garantindo que o acesso à app seja simples e seguro desde o início. Nas Figuras 3.1 e 3.2 podemos verificar essa aplicação.

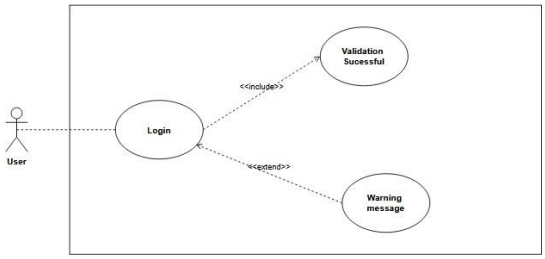

Figura 3.1: Caso de Uso: Login

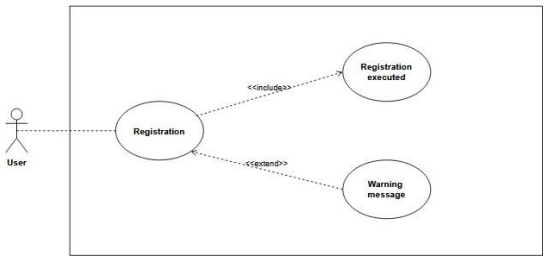

Figura 3.2: Caso de Uso: Registo

2. **Ativação de Chamada de Emergência:**

   Consideremos uma situação em que o utilizador se encontra em perigo e precisa de ajuda imediata. Ao acionar o botão de emergência, a app envia automaticamente a sua localização exata (através de tecnologias como o ***Advanced Mobile Location AML***) para os serviços de emergência e para os contactos pré-definidos. Este fluxo irá garantir uma resposta rápida, mesmo quando o utilizador não consegue comunicar verbalmente a sua posição. Na Figura 3.3 podemos verificar o que fora descrito acima.

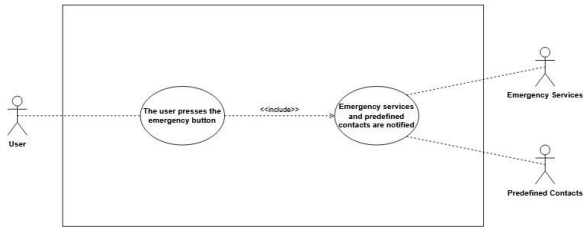

Figura 3.3: Caso de Uso: Emergência

3. **Consulta de Instruções de Primeiros Socorros:**

   Durante uma emergência, cada segundo conta. O utilizador pode aceder, de forma rápida e intuitiva, a uma biblioteca de orientações práticas de primeiros socorros -- desde procedimentos para hemorragias até instruções para suporte básico de vida. Este cenário demonstra como o sistema pode orientar o utilizador em situações críticas de emergência, fornecendo passos claros e visuais para a intervenção eficaz. Na Figura 3.4 podemos verificar o que fora descrito acima.

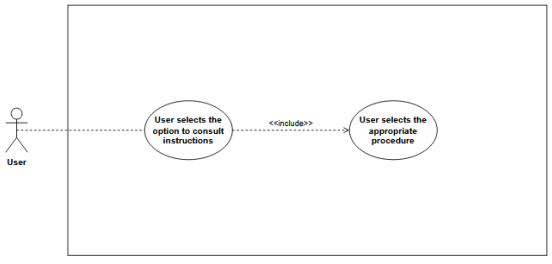

Figura 3.4: Caso de Uso: Procedimentos

4. **Formação e Avaliação:**

   O **LIFE** não se limita a intervenções emergências; este também promove a aprendizagem. Imaginemos que o utilizador decide melhorar os seus conhecimentos de primeiros socorros. Este terá a oportunidade de poder participar em cursos interativos e realizar quizes para testar o seu conhecimento, visualizando o seu desempenho e histórico de forma acessível. Este fluxo incentiva a formação continua e a preparação para situações reais. Na Figura 3.5 podemos verificar o que fora descrito acima.

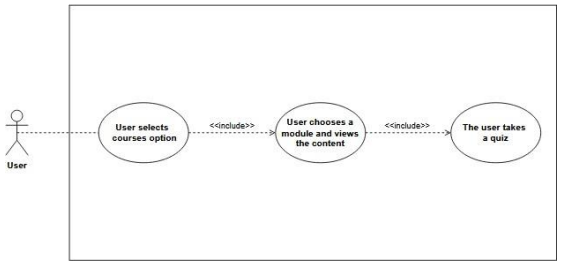

Figura 3.5: Caso de Uso: Teste de Conhecimentos

5. **Gestão de Perfil e Histórico:**

   O utilizador tem a possibilidade de gerir o seu perfil, atualizar os seus dados pessoais, e consultar o histórico de emergências e formações realizadas. Assim, ele pode acompanhar a sua evolução e garantir que as informações estão sempre atualizadas, contribuindo para um sistema mais dinâmico e personalizado. Na Figura 3.6 podemos verificar o que fora descrito acima.

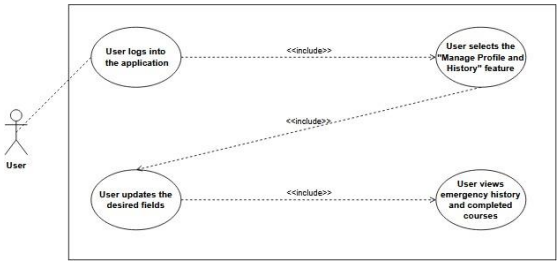

Figura 3.6: Caso de Uso: Atualização de Perfil

3.5 Diagramas de Sequência

Um dos fluxos mais relevantes da aplicação é o do **botão SOS**, que permite ao utilizador comunicar rapidamente uma emergência e partilhar a sua localização atual.

O fluxo inicia-se quando o utilizador pressiona o botão, desencadeando a obtenção da sua localização geográfica através do *FusedLocationProviderClient*. Após a geocodificação (conversão da latitude e longitude em morada textual), é enviado um pedido HTTP POST para o *endpoint* em PHP, contendo os dados da emergência.

O middleware recebe os dados, executa duas instruções SQL INSERT (uma para a tabela *emergency* e outra para *emergency_localization*), e devolve uma resposta JSON de confirmação. Com base nessa resposta, a aplicação abre diretamente o marcador de chamadas com o número de emergência 112.

A Figura **3.7** ilustra este fluxo com um diagrama de sequência, detalhando a comunicação entre os principais intervenientes.

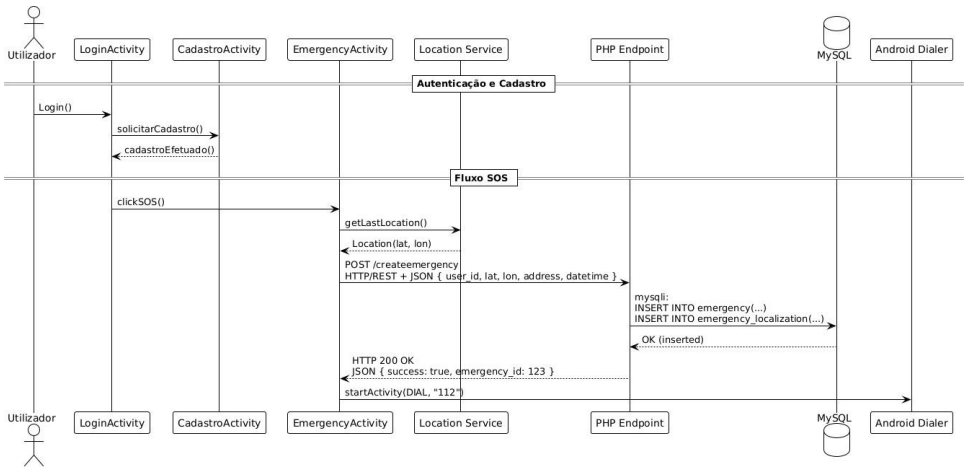

Figura 3.7: Diagrama de sequência do fluxo SOS

3.6 Modelação da Base de Dados

Nesta secção iremos descrever a conceção e a estrutura da base de dados que suporta o projeto **LIFE**. Começamos por apresentar o **modelo Entidade-Relacionamento**, mostrando como cada entidade (e respetivos atributos) se relacionam entre si para garantir a cobertura das funcionalidades principais, como a gestão de utilizadores, contactos, emergências, históricos médicos e formações de primeiros socorros. Em seguida, detalhamos o **Esquema Re-**

lacional, onde cada entidade do **modelo Entidade-Relacionamento** fora traduzida para tabelas, com chaves primárias e estrangeiras definidas para manter a consistência e a integridade dos dados.

3.6.1 Modelo Entidade-Relacionamento

O **modelo Entidade-Relacionamento** é a representação conceptual da base de dados, onde definimos as entidades, os atributos essenciais e os relacionamentos que interligam cada componente do sistema. Neste modelo que apresentamos na figura 3.8, foi possível identificar as seguintes entidades principais:

- **User:**

  Armazena os dados básicos do utilizador, como o nome, data de nascimento, o hash da password e a data de criação de perfil;

- **Address:**

  Regista as informações de morada. Adotando um design que permita que vários utilizadores possam partilhar o mesmo endereço, refletindo cenários reais (ex.: o caso de uma família);

- **User_Address:**

  Funciona como tabela de ligação entre os utilizadores e as moradas, permitindo relacionamentos muitos para muitos (N:M);

- **Contacts:**

  Guarda os Contactos associados a cada utilizador, permitindo a ligação entre os dados pessoais e as informações de contacto;

- **User_Contacts:**

  Funciona como tabela de ligação entre os utilizadores e os contactos, permitindo relacionamentos de muitos para muitos (N:M);

- **Emergency:**

  Contém os registos de emergências, incluindo dados como a data, descrição e localização, facilitando o envio automático de alertas;

- **Medical_History:**

  Documenta o histórico do utilizador, como condições clínicas, alergias e medicações;

- **Courses:**

  Armazena os cursos disponíveis de primeiros socorros;

- **User_Courses:**

  Funciona como uma tabela de ligação entre os utilizadores e os cursos, permitindo relacionamentos muitos para muitos (N : M).

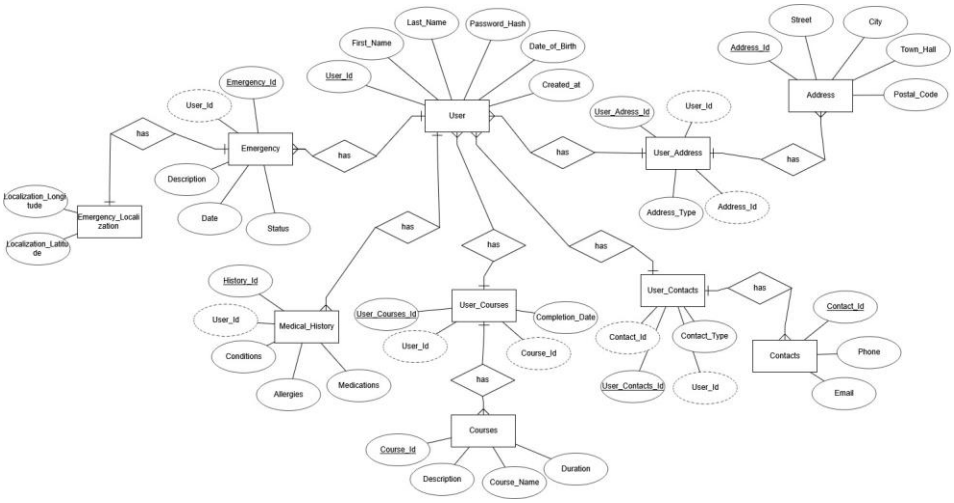

Figura 3.8: Modelo Entidade Relacionamento

Na figura 3.8 representativa do **modelo Entidade-Relacionamento**, nota-se uma organização clara e lógica, com as entidades e os seus atributos bem definidos. A decisão de separar os dados de endereço e contactos do perfil principal do utilizador evidência um design mais pensado para evitar redundâncias e promover a normalização.

Adicionalmente, a utilização de uma tabela de ligação para os cursos (***User_Formations***) demonstra uma abordagem escalável, que permite a evolução futura sem comprometer a integridade dos dados.

3.6.2 Esquema Relacional

O **Esquema Relacional** é a concretização do **modelo ER**, onde cada entidade é transformada numa tabela com colunas especificas e restrições de integridade definidas. Neste esquema, cada tabela apresenta:

- **Chaves Primarias (*Primary Key* (PK)):**

  Para identificar univocamente cada registo.

- **Chaves Estrangeiras (*Forgein Key* (FK)):**

  Para estabelecer e reforçar as relações entre as tabelas.

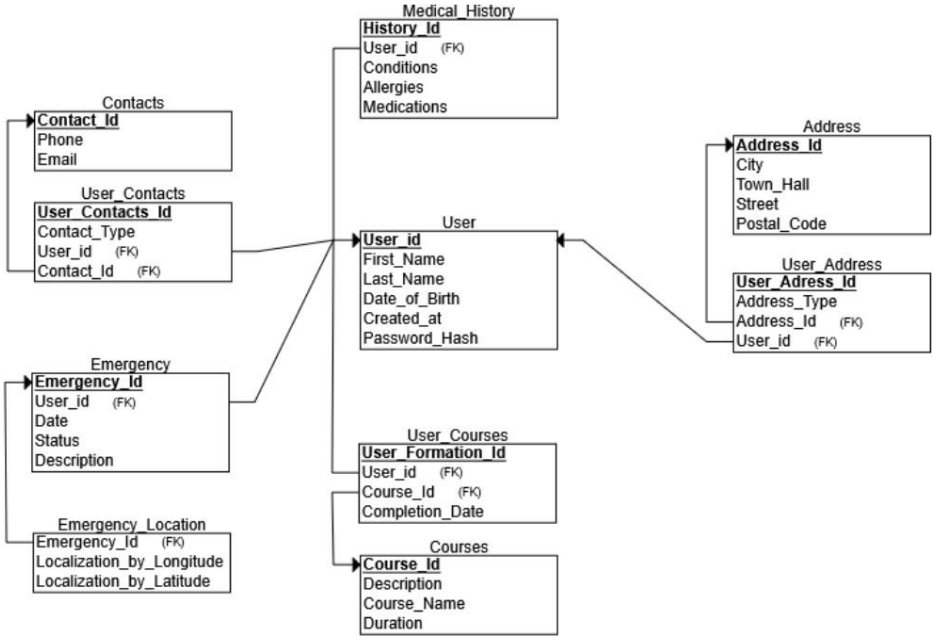

Figura 3.9: Esquema Relacional

Por exemplo:

- A tabela ***User*** contém atributos essenciais como *User_Id*, *First_Name*, *Last_Name*, *Date_of_Birth* e *Created_at*.
- A tabela ***Address*** possui uma **PK** (*Address_Id*) e uma **FK** (*User_Id*).
- A tabela ***Contacts*** inclui *Contact_Id* como **PK** e User_Id como **FK**, estabelecendo a relação 1 : N entres *Users* e *Contacts*.
- Similarmente, as tabelas ***Emergency*** e ***Medical_History*** contêm **FK´s** para vincular cada registo ao respetivo utilizador.
- As tabelas de ligação ***User_Courses***, ***User_Contacts*** e ***User_Address*** permitem relacionar múltiplos utilizadores com múltiplos cursos, múltiplos contactos e múltiplas moradas mediante uma **PK** composta ou uma coluna de identificação própria.

Na figura **3.9** representativa do **Esquema Relacional**, a estrutura apresentada evidencia um design robusto e alinhado com as melhores práticas de engenharia de bases de dados. Cada Tabela está definida, com as chaves primárias e estrangeiras devidamente indicadas, facilitando a manutenção da integridade referencial.

A segmentação das informações em tabelas distintas -- por exemplo, a separação dos contactos e dos endereços -- contribui para uma melhor segmentação dos dados e reduz a redundância, garantindo assim que o sistema seja eficiente e de fácil manutenção. A abordagem modular adotada no **Esquema Relacional** também permita futuras expansões, como a inclusão de novos atributos ou tabelas, sem comprometer a estrutura global.

3.7 Conclusões

Neste capítulo, foram definidos os principais alicerces técnicos do projeto **"LIFE"** no âmbito da Engenharia de Software. Através da especificação dos requisitos funcionais e não funcionais, da descrição dos casos de uso e da modelação da base de dados, ficou evidente que o sistema foi concebido para responder de forma rápida e eficaz a emergências, promovendo simultaneamente a formação prática em primeiros socorros.

Os requisitos funcionais asseguram funcionalidades essenciais como o registo, a autenticação, a gestão de perfil, o acionamento de emergências com partilha automática da localização e o acesso a conteúdos educativos. Em paralelo, os requisitos não funcionais reforçam a preocupações com desempenho, segurança, escalabilidade e compatibilidade multiplataforma. Os casos de uso apresentados ilustram interações realistas tanto em contextos críticos como em momentos de aprendizagem.

A base de dados foi estruturada com base num **modelo Entidade-Relacionamento** claro e um **esquema relacional** robusto, com entidades bem definidas e integridade referencial garantida. Destaca-se ainda a inclusão da entidade *Emergency_Localization*, que permite registar múltiplos pontos de localização durante uma emergência, aumentando o rigor e utilidade dos dados recolhidos.

De uma maneira geral, este capítulo comprova que o projeto **"LIFE"** assenta numa base sólida de engenharia de software, construída segundo boas práticas de desenvolvimento e com foco na fiabilidade, modularidade e extensibilidade do sistema. Esta fundamentação técnica não só suporta a implementação atual, como também facilita a evolução futura da aplicação, permitindo a integração de novas funcionalidades e tecnologias emergentes.

Capítulo

4

Tecnologias e Ferramentas

4.1 Introdução

No presente capítulo são apresentadas as principais tecnologias e ferramentas utilizadas no desenvolvimento deste projeto. Para cada item, descreve-se em que consiste, a sua finalidade geral e como foi integrada na arquitetura do "LIFE".

4.2 Descrição das Tecnologias e Ferramentas Utilizadas

4.2.1 Android Studio

Android Studio é a *IDE* oficial da *Google* para desenvolvimento de aplicações Android, baseada no *IntelliJ IDEA*. Oferece editores visuais de layout, emuladores de diversos dispositivos, ferramentas de *profiling* e integração com o sistema de *build Gradle*. Foi o ambiente onde toda a aplicação móvel foi concebida, testada e empacotada em APK, permitindo ciclos rápidos de compilação e depuração. Poderão ser consultadas mais informações na página oficial [1].

4.2.2 Java

Java é uma plataforma de programação orientada a objetos, reconhecida pela portabilidade e robustez. Neste projeto, foi usada para implementar a lógica no *backend* (classes *DAO*, serviços de autenticação e registo de emergências) e também faz parte do código da aplicação *Android*, correndo sobre

a *Java Virtual Machine (JVM)* do *SDK* do *Android*. A vasta disponibilidade de bibliotecas e a gestão automática de memória facilitaram a construção de componentes modulares e seguros. Poderão ser consultadas mais informações na página oficial [2].

4.2.3 XML

Extensible Markup Language (*Extensible Markup Language* (*XML*)) é uma linguagem de marcação usada para representar dados de forma hierárquica e legível. No "**LIFE**", todos os layouts da interface *Android* (ficheiros *.*XML*) foram definidos em *XML*, separando a camada de apresentação da lógica de programação. Esta abordagem declarativa simplifica alterações ao design e a manutenção das views. Poderão ser consultadas mais informações na página oficial [3].

4.2.4 Visual Studio Code

O *Visual Studio Code* é um editor de código desenvolvido pela Microsoft para os sistemas operativos *Windows*, *macOs* e *Linux*, sendo excelente para o desenvolvimento de diversas aplicações e revela ser extremamente eficaz. Apresenta uma UI apelativa, organizada e de fácil utilização para o programador. Uma das grandes vantagens que apresenta é que permite a instalação de diversas extensões que auxiliam o programador na escrita e no teste do código, possuindo ainda uma linha de comandos embutida que agiliza a compilação e os testes. Poderão ser consultadas mais informações na página oficial [4].

4.2.5 PHP

Hypertext Preprocessor (PHP) (Hypertext Preprocessor) é uma linguagem de *scripting* de propósito geral, amplamente usada em aplicações web. No middleware deste projeto, escreveram-se *endpoints REST* em *PHP* para registar utilizadores, autenticar sessões (JWT) e gravar emergências com as respetivas localizações no *MySQL*. A sintaxe simples e a integração nativa com servidores Apache (via *WampServer*) aceleraram o desenvolvimento do *backend*. Poderão ser consultadas mais informações na página oficial [5].

4.2.6 PostMan

Postman é um ambiente de testes para *APIs REST*, que permite construir coleções de requisições HTTP, parametrizar cabeçalhos e *payloads JSON*, e validar respostas interactivamente. Utilizámo-lo para verificar manualmente

cada *endpoint* PHP (createuser.PHP, login.PHP, createemergency.PHP), simulando vários cenários de sucesso e erro antes da integração com a app Android. Poderão ser consultadas mais informações na página oficial [6].

4.2.7 WampServer

WampServer é um pacote para Windows que integra Apache, PHP e MySQL num único instalador, criando um servidor local ("localhost") de forma simples. Foi a plataforma escolhida para hospedar o middleware e a base de dados durante o desenvolvimento, permitindo importar o dump SQL, editar db_config.PHP e testar os scripts PHP sem necessidade de configurações complexas. Poderão ser consultadas mais informações na página oficial [7].

4.2.8 MySQL

MySQL é um sistema de gestão de base de dados relacional open source. Neste projeto, armazenou-se toda a informação persistente: utilizadores, contactos, emergências, localizações e histórico de cursos. O MySQL Workbench serviu para modelar o esquema ER, gerar o DDL e exportar/importar dados de teste, garantindo a integridade referencial e performance em consultas. Poderão ser consultadas mais informações na página oficial [8].

4.2.9 GitHub

GitHub é uma plataforma de hospedagem de código com suporte a Git, *issues*, *pull requests* e integração contínua. Hospedou-se o repositório do "LIFE", permitindo o versionamento colaborativo entre elementos da equipa, controlo de *branches*, revisão de código e documentação do progresso. A utilização de "*Commits*" atómicos e "*Pull Requests*" assegurou rastreabilidade e qualidade no desenvolvimento. Poderão ser consultadas mais informações na página oficial [9].

Capítulo

5

Implementação e Testes

5.1 Introdução

Nesta secção é descrito o processo de implementação da aplicação LIFE, desde a definição da arquitetura até à integração dos diversos componentes que constituem o sistema de forma clara e acessível. O principal objetivo foi garantir uma resposta eficaz e imediata em emergências, através da partilha automática da localização do utilizador e da disponibilização de orientações de primeiros socorros, de forma clara e acessível.

A aplicação foi desenvolvida utilizando *Android Studio*, com recurso às linguagens Java e XML para a construção da interface e da lógica da aplicação. A camada de comunicação entre a aplicação móvel e a base de dados foi assegurada por um middleware desenvolvido em PHP, que permitiu o intercâmbio seguro e eficiente de dados entre o cliente (app) e o servidor.

A base de dados *MySQL* foi utilizada para armazenar de forma estruturada os dados dos utilizadores, contactos de emergência, históricos de utilização, entre outros registos relevantes. Esta arquitetura cliente-servidor proporcionou uma separação clara entre o *frontend*, o *backend* e a camada de dados, facilitando a manutenção e futura escalabilidade da aplicação.

Adicionalmente, foram realizados testes para validar a robustez e o correto funcionamento da aplicação, com especial foco nas funcionalidades críticas como o botão de emergência, o envio automático da localização em tempo real, a visualização de procedimentos de primeiros socorros e a gestão do perfil do utilizador.

Estes testes foram fundamentais para garantir a fiabilidade da aplicação, mesmo em contextos de elevada urgência e stress.

5.2 Arquitetura Geral do Sistema

O sistema "**LIFE**" segue uma arquitetura de três camadas composta por:

- A aplicação móvel **Android**;
- Um ***middleware*** em **PHP** executado num servidor local via WampServerer;
- E uma **base de dados *MySQL*** onde se armazenam os dados persistentes.

A comunicação entre a app e o middleware é realizada via chamadas HTTP RESTful, com os dados trocados em formato JSON. O middleware age como uma ponte entre o cliente e a base de dados, tratando a autenticação do utilizador, o registo de emergências e a consulta de dados do utilizador.

A figura **5.1** apresenta a arquitetura de alto nível, destacando os principais componentes e o fluxo de informação entre eles.

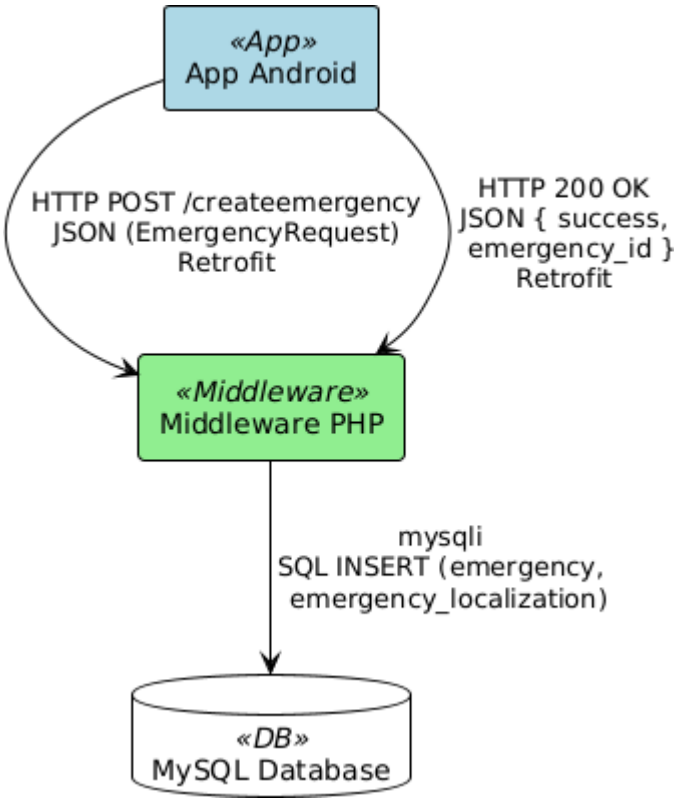

Figura 5.1: Arquitetura de alto nível do sistema LIFE

5.3 Descrição dos Componentes do *Frontend*

A aplicação LIFE foi desenvolvida na plataforma Android Studio, recorrendo às linguagens Java e XML, para disponibilizar uma solução móvel intuitiva, fiável e eficaz para emergências. Esta secção apresenta os principais componentes técnicos da aplicação, bem como as funcionalidades implementadas do lado do cliente (Android), descrevendo o seu funcionamento e integração com os restantes elementos do sistema.

A interface gráfica foi desenhada para proporcionar uma experiência de utilização rápida e clara, mesmo em contextos de stress. Funcionalidades críticas, como o botão de emergência (SOS), foram destacadas na interface para permitir uma interação imediata.

A aplicação comunica com o servidor através da biblioteca *Retrofit*, utilizando o protocolo HTTP e o formato JSON para envio e receção de dados. A geolocalização é obtida com recurso à API de localização da Google, sendo posteriormente tratada e enviada para o servidor PHP. O identificador do utilizador autenticado é armazenado localmente com recurso à classe *SharedPreferences*, permitindo manter a sessão ativa entre utilizações.

Nas subsecções seguintes são descritos os principais módulos da aplicação, com destaque para as funcionalidades implementadas, os trechos de código mais relevantes e o papel de cada componente na lógica da aplicação.

5.3.1 Autenticação e Registo

O sistema LIFE inclui um módulo de autenticação baseado em nome de utilizador e palavra-passe. A funcionalidade de registo permite ao utilizador criar uma conta mediante um formulário, sendo obrigatória a introdução de dados como nome, e-mail, data de nascimento, nome de utilizador e palavra-passe.

Para garantir a segurança e a integridade dos dados, foram aplicadas diversas validações no lado cliente, complementadas por verificações no servidor. Após um login bem-sucedido, é mantida uma sessão persistente do utilizador, assegurando que as funcionalidades críticas (como o botão SOS) sejam corretamente associadas à conta ativa.

Abaixo apresentam-se os principais mecanismos aplicados, acompanhados dos respetivos trechos de código:

5.3.1.1 Validações

Para garantir a integridade e a segurança dos dados introduzidos no processo de registo, foram implementadas duas validações essenciais no lado do

cliente: uma para a robustez da **password** e outra para o formato do **e-mail**.

Validação de Password: A password fornecida pelo utilizador é sujeita a uma verificação de robustez utilizando uma expressão regular que impõe os seguintes critérios mínimos:

- Mínimo de 8 caracteres;
- Pelo menos uma letra maiúscula;
- Pelo menos um número;
- Pelo menos um símbolo especial (como !, @, $, etc.).

Os seguintes excertos de código **5.1** e 5.2 demonstra a implementação desta validação no Android:

```
String regex = "^(?=.*[A-Z])(?=.*\\d)(?=.*[@$\!\%\*\?\\&])[A-Za-z\\d\@\$
    \!\%\*\?\\&]{8,}$";
if (!pwd.matches(regex)) {
    Toast.makeText(this,
        "Password fraca: precisa de >= 8 carateres, 1 maiuscula, 1
            numero e 1 simbolo",
        Toast.LENGTH_LONG).show();
    return;
}
```

Excerto de Código 5.1: Validação de password forte no registo

Validação do E-mail: Para evitar erros de envio e reforçar a integridade da informação, é também verificado o formato do e-mail introduzido, com base na classe Patterns.EMAIL_ADDRESS do Android SDK. Somente endereços com estrutura válida são aceites e enviados ao servidor.

```
if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
    Toast.makeText(this,
        "E-mail invalido: insira um endereco no formato correto", Toast.
            LENGTH_SHORT).show();
    return;
}
```

Excerto de Código 5.2: Validação do e-mail no cliente Android

Esta validação é complementada no lado do servidor PHP, utilizando a função filter_var() com o filtro FILTER_VALIDATE_EMAIL, garantindo assim coerência e proteção em ambas as camadas da aplicação.

5.3.1.2 Gestão de Sessões e Credenciais

A autenticação e gestão de sessões na aplicação LIFE foram concebidas para garantir segurança, persistência e uma boa experiência de utilizador. Para isso, foram combinadas práticas modernas de envio de credenciais com mecanismos locais de persistência de sessão.

A comunicação entre o cliente Android e o servidor PHP é feita através da biblioteca *Retrofit*, utilizando pedidos HTTP com *payload* em formato JSON. Após a submissão do nome de utilizador e da password, o servidor valida as credenciais e devolve uma resposta contendo o identificador do utilizador e um *token* JWT. O excerto 5.3 ilustra o envio das credenciais com *Retrofit*:

```
JsonObject body = new JsonObject();
body.addProperty("username", username);
body.addProperty("password", password);

api.loginUser(body).enqueue(new Callback<JsonObject>() {
    @Override
    public void onResponse(Call<JsonObject> call, Response<JsonObject>
        resp) {
        if (resp.isSuccessful() && resp.body().get("success").
            getAsBoolean()) {
            int userId = resp.body().get("user_id").getAsInt();
            // guardar user_id...
        }
    }
});
```

Excerto de Código 5.3: Envio de credenciais via *Retrofit*

Após uma autenticação bem-sucedida, o user_id é armazenado localmente através da classe SharedPreferences. Esta estratégia permite que a aplicação reconheça o utilizador em sessões futuras sem necessidade de um novo login, mantendo o estado da sessão de forma segura e eficiente. No excerto 5.4 podemos verificar como tal é devidamente aplicado.

```
getSharedPreferences("my_app_preferences", MODE_PRIVATE)
    .edit()
    .putInt("user_id", userId)
    .apply();
```

Excerto de Código 5.4: Armazenamento de sessão do utilizador

O identificador guardado é utilizado para associar todas as ações relevantes — como o envio de pedidos de emergência ou a visualização do histórico — à conta correta na base de dados. Este mecanismo simplifica o fluxo de navegação do utilizador e contribui para uma experiência contínua e segura.

5.3.2 Modelos de Dados JSON e Integração

Na aplicação "**LIFE**" foi utilizado um ficheiro JSON para estruturar e apresentar as informações relativas aos tópicos de primeiros socorros, como os procedimentos de **RCP** (Reanimação Cardiopulmonar). Cada tópico está organizado de forma hierárquica, permitindo uma navegação intuitiva e modular na aplicação Android.

Cada tópico é representado por um objeto com os seguintes campos principais:

- ***id*:** Identificador único do tópico;
- ***title*:** Título do tópico apresentado na interface da aplicação;
- ***subtopics* (opcional):** Lista de subtópicos relacionados, onde cada um contém:
  - ***id*:** Identificador único do subtópico;
  - ***title*:** Título do subtópico apresentado ao utilizador;
  - ***steps*:** Lista de instruções passo-a-passo, onde cada passo contém:
    - ***instruction*:** Texto principal da instrução;
    - ***substeps*:** Lista de ações complementares ou detalhes adicionais.

O excerto 5.5 exemplifica a estrutura definida para o tópico **RCP em Adulto**, demonstrando a organização interna em passos e sub passos:

```
{
  "id": "adulto",
  "title": "Adulto",
  "steps": [
    {
      "instruction": "Verifique se existe perigo",
      "substeps": [
        "Antes de se aproximar assegure-se da sua segurana, da vitima e
            dos espectadores."
      ]
    },
    {
      "instruction": "Verifique a resposta da vitima",
      "substeps": [
        "Use um comando em voz alta e um aperto FIRME nos ombros para
            obter resposta",
        "LIGAR 112 CASO NAO OBTENHA RESPOSTA"
      ]
```

```
    },
    {
      "instruction": "Limpe e abra as vias aereas",
      "substeps": [
        "Verifique se a boca esta limpa",
        "Abras as vias aereas com inclinacao da cabeca e elevacao do
            queixo",
        "Olhe, escute e sinta a respiracao normal"
      ]
    }
  ]
}
```

Excerto de Código 5.5: Instruções iniciais para RCP em Adulto

5.3.3 Descrição das Interfaces

A aplicação "**LIFE**" utiliza uma arquitetura baseada em fragmentos para organizar e apresentar os conteúdos de forma modular e intuitiva. Esta abordagem permite uma navegação fluida entre diferentes níveis de informação, desde a visualização geral dos tópicos até aos pormenores específicos de cada procedimento.

5.3.3.1 Interface Principal

O **PrimeirosSocorrosFragment** serve como ponto de entrada para os módulos de primeiros socorros, apresentando uma grelha organizada de tópicos disponíveis. Este fragmento é o principal responsável por:

- Carregar dinamicamente os tópicos através do **TopicRepository**;
- Organizar os tópicos em duas categorias:
  - Comuns (os primeiros 6);
  - outros sintomas.
- Gerir a transição para os pormenores específicos de cada tópico.

O seguinte excerto de código 5.6 mostra a inicialização do fragmento e carregamento dos tópicos de forma dinâmica:

```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
    ViewGroup container, @Nullable Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_primeiros_socorros,
        container, false);
```

```
    // Inicializacao dos componentes da interface
    btnEmergency = view.findViewById(R.id.btnEmergency);
    gridCommon = view.findViewById(R.id.gridCommon);
    gridAll = view.findViewById(R.id.gridAll);

    // Carregamento dos topicos atraves do repositorio
    TopicRepository.init(getContext());
    List<Topic> allTopics = TopicRepository.getAllTopics();

    // Separacao em topicos comuns e outros
    List<Topic> common = allTopics.subList(0, Math.min(6, allTopics.size
        ()));
    List<Topic> others = allTopics.size() > 6 ? allTopics.subList(6,
        allTopics.size()) : new ArrayList<>();

    // Adicao dos botoes as respectivas grelhas
    addButtonsToGrid(gridCommon, common);
    addButtonsToGrid(gridAll, others);

    return view;
}
```

Excerto de Código 5.6: Inicialização do fragmento e carregamento dos tópicos

5.3.3.2 Interface dos botões Dinâmicos

A funcionalidade addButtonsToGrid() é responsável pela criação dinâmica de botões para cada tópico, configurando as suas propriedades visuais e comportamentos. O seguinte excerto 5.7 demonstra esta implementação:

```
private void addButtonsToGrid(GridLayout grid, List<Topic> topics) {
    grid.removeAllViews();
    for (Topic t : topics) {
        Button btn = new Button(getContext());
        btn.setText(t.title);

        // Configuracao do layout para distribuicao equitativa
        GridLayout.LayoutParams params = new GridLayout.LayoutParams();
        params.width = 0;
        params.height = ViewGroup.LayoutParams.WRAP_CONTENT;
        params.columnSpec = GridLayout.spec(GridLayout.UNDEFINED, 1f);
        btn.setLayoutParams(params);

        // Configuracao do listener para navegacao
        btn.setOnClickListener(v -> {
            Bundle args = new Bundle();
            args.putString("topicId", t.id);
```

```
                DetailFragment fragment = new DetailFragment();
                fragment.setArguments(args);
                getParentFragmentManager()
                        .beginTransaction()
                        .replace(R.id.fragment_container, fragment)
                        .addToBackStack(null)
                        .commit();
            });

            grid.addView(btn);
        }
    }
```

Excerto de Código 5.7: Criação dinâmica de botões

5.3.3.3 Interface de Visualização de Eventos

O **DetailFragment** é responsável pela apresentação detalhada de cada tópico de primeiros socorros. Este fragmento adapta-se dinamicamente ao conteúdo, podendo apresentar:

- Tópicos com subtópicos (como RCP para adultos, crianças e bebés);
- Tópicos diretos com instruções passo-a-passo;
- Interface expansível para visualização hierárquica das instruções

O seguinte excerto 5.8 mostra a lógica de inicialização e adaptação do conteúdo:

```
@Override
public View onCreateView(LayoutInflater inf, ViewGroup c, Bundle
    savedInstanceState) {
    TopicRepository.init(getContext());
    View v = inf.inflate(R.layout.fragment_detail, c, false);
    LinearLayout ll = v.findViewById(R.id.llContent);

    // Obtencao do topico atraves do ID passado como argumento
    String topicId = getArguments().getString("topicId");
    Topic t = TopicRepository.findById(topicId);

    // Verificacao se o topico tem sub-topicos
    if (t.subtopics != null && !t.subtopics.isEmpty()) {
        // Criacao de botoes para cada sub-topico
        for (Topic.Subtopic s : t.subtopics) {
            Button btnSub = new Button(getContext());
            btnSub.setText(s.title);
            ll.addView(btnSub);
```

```
            btnSub.setOnClickListener(_v -> showSteps(ll, s.steps));
        }
    } else {
        // Apresentacao directa das instrucoes
        showSteps(ll, t.steps);
    }
    return v;
}
```

Excerto de Código 5.8: Lógica de inicialização e adaptação do conteúdo

5.3.3.4 Interface Hierárquica das Instruções

A funcionalidade **showSteps()** é fundamental para a apresentação clara e organizada das instruções de primeiros socorros. Esta implementação cria uma interface hierárquica onde cada instrução principal pode ter sub instruções detalhadas.

O seguinte excerto **5.9** demonstra a implementação desta funcionalidade:

```
@Override
public View onCreateView(LayoutInflater inf, ViewGroup c, Bundle
    savedInstanceState) {
    TopicRepository.init(getContext());
    View v = inf.inflate(R.layout.fragment_detail, c, false);
    LinearLayout ll = v.findViewById(R.id.llContent);

    // Obtencao do topico atraves do ID passado como argumento
    String topicId = getArguments().getString("topicId");
    Topic t = TopicRepository.findById(topicId);

    // Verificacao se o topico tem sub-topicos
    if (t.subtopics != null && !t.subtopics.isEmpty()) {
        // Criacao de botoes para cada sub-topico
        for (Topic.Subtopic s : t.subtopics) {
            Button btnSub = new Button(getContext());
            btnSub.setText(s.title);
            ll.addView(btnSub);
            btnSub.setOnClickListener(_v -> showSteps(ll, s.steps));
        }
    } else {
        // Apresentacao directa das instrucoes
        showSteps(ll, t.steps);
    }
    return v;
}
```

Excerto de Código 5.9: Implementação da funcionalidade de instruções e sub instruções

5.3.3.5 Interface de Integração

A classe **TopicRepository** serve como camada de abstração para acesso aos dados dos tópicos de primeiros socorros. Esta implementação permite:

- Carregamento dos dados JSON de forma assíncrona;
- Armazenamento temporário dos tópicos em memória para melhor desempenho;
- Pesquisa eficiente por ID de tópico;
- Isolamento da lógica de acesso aos dados

A integração entre os fragmentos e o repositório assegura uma separação clara entre a lógica de apresentação e a gestão de dados, facilitando a manutenção e futuras extensões da aplicação.

5.3.3.6 Interface de Emergência

A funcionalidade do botão SOS foi desenhada para que o utilizador, com um único toque, possa:

- Obter a sua localização atual;
- Ver a morada correspondente;
- Ver a sua geolocalização (Longitude e Latitude);
- Submeter uma emergência para o servidor;
- Abrir o marcador de chamada para o número europeu de emergência — **112**.

A geolocalização é obtida através da API **FusedLocationProviderClient**, com recurso à classe **Geocoder** para conversão de coordenadas em morada textual (geocodificação inversa). Caso o utilizador não tenha permissões, a app solicita-as de forma dinâmica com ActivityResultLauncher.

Após a recolha da localização, é criado um objeto do tipo EmergencyRequest, enviado para o servidor através da API *Retrofit*. Se o servidor responder com sucesso, é automaticamente iniciado o marcador de chamada para o 112.

5.3.4 Descrição das Integrações com o Google Maps

Foi utilizada a **API** do ***Google Maps*** para:

- Visualizar a localização atual do utilizador um *MapView*;
- Mostrar marcadores com coordenadas e nomes de ruas;
- Ajudar a confirmar visualmente o local de onde a emergência foi reportada.

Para isso, foi necessário gerar uma chave de API (API Key) através da *Google Cloud Platform* e ativar os seguintes serviços:

- ***Maps SDK for Android***;
- ***Geocoding API***.

A chave foi associada ao ficheiro google_maps_api.xml e incluída no projeto.

Esta chave é necessária para que o mapa carregue corretamente, bem como para que a conversão de coordenadas em morada funcione através do *Geocoder*.

5.4 Descrição dos Componentes do *Backend*

Nesta secção descreve-se a organização, as responsabilidades e as principais características do middleware em *PHP* que serve de ponte entre a aplicação *Android* e a base de dados *MySQL*. Para facilitar a leitura, a estrutura foi dividida em subcapítulos, cada um focado num aspeto essencial.

5.4.1 Visão Geral da Estrutura e Dependências

O código do middleware encontra-se na pasta life_api, cuja hierarquia lógica é a seguinte:

- vendor/: bibliotecas externas geridas pelo *Composer* (por exemplo, textttfirebase/php-jwt);
- composer.json e composer.lock : definição e travamento de versões de dependências;
- db_config.php : configuração única da ligação à base de dados *MySQL*;

- create_user.php, login.php, create_emergency.php : endpoints REST independentes, cada um responsável por uma funcionalidade (registo, login, emergência)

O uso do *Composer* e do *autoloader* (require 'vendor/autoload.php') permite integrar rapidamente bibliotecas externas, mantendo o código limpo e modular.

5.4.2 Configuração do Sistema de Gestão de Bases de Dados

O ficheiro db_config.php centraliza a criação da ligação *MySQL*, definindo *host*, utilizador, base de dados e conjunto de caracteres UTF-8. Em caso de falha, retorna HTTP_500 e uma mensagem *JSON* de erro.

```php
<?php
// db_config.php
$mysqli = new mysqli('127.0.0.1','root','','life');
if ($mysqli->connect_errno) {
    http_response_code(500);
    echo json_encode(['error' => 'DB connection failed']);
    exit;
}
$mysqli->set_charset('utf8');
?>
```

Excerto de Código 5.10: Configuração da ligação à base de dados

Estas poucas linhas referidas acima no excerto de código 5.10 garantem que todas as operações subsequentes (nos vários *endpoints*) partilham a mesma configuração de acesso à base de dados e respondem de forma consistente a erros de ligação.

5.4.3 Descrição das Rotas

Esta secção apresenta o funcionamento das principais rotas (*endpoints*) da *API REST* desenvolvida para o projeto LIFE, responsáveis pela comunicação entre o cliente *Android* e o servidor *PHP*/*MySQL*. Cada rota foi implementada para tratar pedidos HTTP do tipo POST, garantindo validação dos dados recebidos, resposta em formato JSON e proteção contra vulnerabilidades comuns como *SQL injection*.

Através destas rotas, é possível realizar operações fundamentais como o registo e autenticação de utilizadores, bem como o envio de pedidos de emergência com geolocalização associada. Os exemplos apresentados incluem excertos de código reais, ilustrando a lógica de validação, sanitização, inserção na base de dados e geração de *tokens* JWT. Esta abordagem modular e segura

assegura a fiabilidade do *backend* e o correto suporte às funcionalidades críticas da aplicação.

5.4.3.1 Registo de Utilizadores

Este script processa pedidos *HTTP POST* com *payload JSON* contendo campos como first_name, last_name, email, username, password e date_of_birth. Primeiro, valida se todos os campos obrigatórios estão presentes e se o e-mail tem formato válido. Depois, verifica unicidade de nome de utilizador e e-mail. A password é *hasheada* com password_hash() (*BCRYPT*) antes de ser inserida na tabela *user* usando *prepared statements*. Em caso de sucesso, retorna um *JSON* com success:true e o novo user_id; em caso de erro, um *JSON* com *error* e o código *HTTP* adequado. No excerto 5.11 podemos verificar o *header* do create_user.php

1. **Cabeçalho e inclusão das dependências:**

```
header('Content-Type: application/json');
require 'db_config.php';
```

Excerto de Código 5.11: *Header* do create_user.php

2. **Leitura e decodificação do *payload JSON***
   Lemos todo o corpo da requisição e tentamos convertê-lo num array PHP. Se falhar, devolvemos um erro 400 (Bad Request):

```
$data = json_decode(file_get_contents('php://input'), true);
if (!$data) {
    http_response_code(400);
    echo json_encode(['error' => 'Invalid JSON']);
    exit;
}
```

Excerto de Código 5.12: Leitura e decodificação do payload JSON

Este passo protege representado no excerto de código 5.12 contra pedidos mal formatados e evita avisos ou erros posteriores ao acederem a $data.

3. **Sanitização e validação de campos obrigatórios**
   Extraímos cada campo esperado, escapando *strings* para prevenir SQL *injection*, e confirmamos que nenhum está vazio:

```
$first = $mysqli->real_escape_string($data['first_name'] ?? '');

$dob = $mysqli->real_escape_string($data['date_of_birth'] ?? '')
    ;
```

```
if (!$first || !$last || !$email || !$user || !$plain || !$dob) {
    http_response_code(422);
    echo json_encode(['error'=>'Missing fields']);
    exit;
}
```

Excerto de Código 5.13: Sanitização e validação de campos obrigatórios

Neste excerto de código 5.13 asseguramos que todos os campos que o registo exige estão presentes e limpos.

4. **Verificação de unicidade (username/email)**

   Antes de inserir, conferimos se já existe um utilizador com o mesmo nome ou e-mail e, em caso afirmativo, devolvemos 409 (Conflict):

```
$res = $mysqli->query(
    "SELECT user_id FROM user WHERE username='$user' OR email='
        $email'"
);
if ($res && $res->num_rows > 0) {
    http_response_code(409);
    echo json_encode(['error' => 'Username or email already exists
        ']);
    exit;
}
```

Excerto de Código 5.14: Verificação de unicidade

Este bloqueio, apresentado no excerto de código 5.14, no lado servidor previne duplicações e garante consistência.

5. **Hash da password e inserção**

   Geramos um hash seguro com *Bcrypt* e preparamos o *prepared statement* para inserir o novo registo:

```
$hash = password_hash($plain, PASSWORD_BCRYPT);

$stmt = $mysqli->prepare(
  "INSERT INTO `user`
   (first_name, last_name, date_of_birth, username, email,
       password_hash)
   VALUES (?, ?, ?, ?, ?, ?)"
);
$stmt->bind_param('ssssss', $first, $last, $dob, $user, $email,
    $hash);
$stmt->execute();
```

Excerto de Código 5.15: Hash da password e inserção

O uso de *prepared statements*, referido no excerto de código 5.15, elimina o risco de *SQL injection* e a password nunca fica armazenada em texto claro.

6. **Resposta ao cliente**

   Se o execute() correr sem erros, devolvemos success:true com o novo user_id; caso contrário, um erro 500 com detalhe:

```
if ($stmt->execute()) {
    echo json_encode(['success'=>true, 'user_id'=>$stmt->insert_id
        ]);
} else {
    http_response_code(500);
    echo json_encode([
      'error'=>'Insert failed',
      'details' => $stmt->error
    ]);
}
$stmt->close();
$mysqli->close();
```

Excerto de Código 5.16: Resposta ao cliente

Assim, com este excerto de código 5.16, o cliente *Android* pode reagir de forma apropriada, mostrando sucesso ou mensagem de erro ao utilizador.

5.4.3.2 Autenticação de Utilizadores

Recebe via *POST* o *username* e *password* em *JSON*. Procura no campo password_hash correspondente ao nome de utilizador, e compara com password_verify().
Se as credenciais forem válidas, gera um *token JWT* com *payload* contendo emissor, data de expiração (1 hora) e sub igual ao user_id. Retorna um *JSON* com success:true, user_id e o *token*. Em caso de falha, devolve um *HTTP* 401 com o campo error.

1. **Cabeçalho e dependências**

   O excerto 5.17 configura resposta *JSON* e carrega a ligação à Base de Dados e o *autoloader* do *Composer* para *JWT*:

```
header('Content-Type: application/json');
require 'db_config.php';
```

```
require    'vendor/autoload.php';

use Firebase\JWT\JWT;
```

Excerto de Código 5.17: Cabeçalho e dependências

2. **Leitura e validação dos campos**
   O excerto 5.18 decodifica o *JSON* de entrada e assegura que *username* e *password* não estejam vazios:

```
$data = json_decode(file_get_contents('php://input'), true);
$user = $mysqli->real_escape_string($data['username'] ?? '');
$plain = $data['password'] ?? '';

if (!$user || !$plain) {
    http_response_code(422);
    echo json_encode(['error'=>'Missing fields']);
    exit;
}
```

Excerto de Código 5.18: Leitura e validação dos campos

3. **Recuperação do hash e verificação da password**
   O excerto 5.19 obtém o password_hash e compara com password_verify(), retornando 401 em caso de credenciais inválidas:

```
$stmt = $mysqli->prepare("SELECT user_id, password_hash FROM `user`
    WHERE username=?");
$stmt->bind_param('s', $user);
$stmt->execute();
$stmt->store_result();

if ($stmt->num_rows === 0) {
    http_response_code(401);
    echo json_encode(['error'=>'User not found']);
    exit;
}

$stmt->bind_result($uid, $hash);
$stmt->fetch();

if (!password_verify($plain, $hash)) {
    http_response_code(401);
    echo json_encode(['error'=>'Invalid credentials']);
    exit;
}
```

Excerto de Código 5.19: Recuperação do *hash* e verificação da *password*

4. **Geração e envio do *token JWT***
   O excerto 5.20 cria um *payload* com emissor, emissão, expiração (1 Hora) e sub igual a user_id, e devolve-o em JSON:

```
$now        = time() ;
$payload    = [
        'iss'  =>  'teu_dominio.com',
        'iat' => $now,
        'exp'  => $now + 3600,
        'sub' => $uid
];
$jwt    = JWT::encode( $payload ,    'chave_super_secreta_123!' ,    'HS256' ) ;

echo   json_encode ( [
        'success'  =>  true ,
        'user_id'  =>  $uid ,
        'token'      =>  $jwt
]) ;
```

Excerto de Código 5.20: Geração e envio do token JWT

5.4.3.3 Gestão de Eventos de Emergência

Processa pedidos *POST JSON* com user_id, description, status, latitude, latitude e street_address. Valida os campos obrigatórios e garante que o estado (status) está entre os valores permitidos. Insere primeiro na tabela emergency (registando data e descrição) e de seguida na tabela emergency_localization (com latitude, longitude e morada), usando sempre *prepared statements*. Responde com um JSON detalhado que inclui emergency_id, coordenadas e descrição.

1. **Cabeçalho e configuração**
   O ficheiro começa por definir o tipo de resposta como JSON e incluir a configuração de acesso à base de dados, como refere o excerto 5.21:

```
<?php
header( 'Content-Type: application/json' );
require  'db_config.php';
?>
```

Excerto de Código 5.21: create_emergency.php – Cabeçalho e configuração

2. **Leitura e validação do JSON**
   Lê todo o corpo da requisição e converte-o num *array* PHP. Se a conversão falhar (*payload* inválido ou vazio), devolve um erro 400, tal como descrito no excerto 5.22:

```
$data = json_decode(file_get_contents('php://input'), true);
if (!$data) {
    http_response_code(400);
    echo json_encode(['error'=>'Invalid JSON']);
    exit;
}
```

Excerto de Código 5.22: create_emergency.php – Leitura e validação do JSON

3. **Limpeza e verificação de campos obrigatórios**

   Extraímos e limpamos (real_escape_string) os campos críticos: user_id, *description*, latitude e longitude. Se algum dos valores essenciais estiver a faltar ou inválido, devolvemos um erro 422, tal como descrito no excerto 5.23:

```
$userId      = intval($data['user_id'] ?? 0);
$description = $mysqli->real_escape_string($data['description'] ??
    '');
$lat         = isset($data['latitude']) ? doubleval($data['
    latitude']) : null;
$lon         = isset($data['longitude']) ? doubleval($data['
    longitude']) : null;

if (!$userId || !$description || $lat === null || $lon === null) {
    http_response_code(422);
    echo json_encode(['error'=>'Missing fields']);
    exit;
}
```

Excerto de Código 5.23: create_emergency.php – Sanitização e verificação de campos

4. **Inserção na tabela emergency**

   Preparamos e executamos um *prepared statement* para inserir o registo de emergência (associado ao user_id) na tabela emergency, capturando em seguida o insert_id gerado,tal como descrito no excerto 5.24:

```
$stmt = $mysqli->prepare("
    INSERT INTO emergency (user_id, date, description, status)
    VALUES (?, NOW(), ?, ?)
");
$stmt->bind_param('iss', $userId, $description, $status);
$stmt->execute();
$emergencyId = $stmt->insert_id;
$stmt->close();
```

Excerto de Código 5.24: create_emergency.php – Inserção na tabela emergency

5. **Inserção na tabela emergency_localization**

   Com o emergency_id obtido, preparamos outro *prepared statement* para registar as coordenadas e a morada textual na tabela emergency_localization,tal como descrito no excerto 5.25:

```
$stmt2 = $mysqli->prepare("
    INSERT INTO emergency_localization
      (emergency_id, latitude, longitude, street_address)
    VALUES (?, ?, ?, ?)
");
$stmt2->bind_param('idds', $emergencyId, $lat, $lon, $street);
$stmt2->execute();
$stmt2->close();
```

Excerto de Código 5.25: create_emergency.php – Inserção na tabela emergency_localization

6. **Resposta JSON de sucesso**

   Finalmente, devolvemos ao cliente um JSON detalhado com todos os dados da emergência registada, incluindo emergency_id, coordenadas e descrição, tal como descrito no excerto 5.26:

```
echo json_encode([
    'success'        => true,
    'emergency_id'   => $emergencyId,
    'latitude'       => $lat,
    'longitude'      => $lon,
    'street_address'=> $street,
    'status'         => $status,
    'description'    => $description
]);
$mysqli->close();
?>
```

Excerto de Código 5.26: create_emergency.php – Resposta JSON de sucesso

5.5 Testes e Validação

Esta secção apresenta os procedimentos adotados para a verificação da robustez, estabilidade e conformidade funcional da aplicação LIFE. Os testes realizados visaram avaliar o cumprimento dos requisitos definidos na fase de Engenharia de Software, bem como validar o comportamento do sistema em diferentes cenários simulados. A abordagem adotada baseou-se em testes manuais exploratórios, conduzidos num ambiente controlado com recurso a emulador Android e servidor local, com especial enfoque nas funcionalidades

essenciais como o registo de utilizadores, autenticação, e gestão de emergências com localização em tempo real.

Serão descritos, de forma sistemática, o ambiente de testes utilizado, a análise do cumprimento dos requisitos funcionais e não funcionais, e os resultados obtidos através da observação direta do sistema durante a sua execução. As evidências recolhidas, sob a forma de tabelas e capturas de dados reais, sustentam a avaliação da fiabilidade e da consistência da aplicação desenvolvida.

5.5.1 Descrição do Ambiente de Testes

Os testes foram realizados em emulador *Pixel 8 Pro* (API 35) no *Android Studio*, conectado ao servidor local *WampServer*. Apesar de não ter sido testado em dispositivo físico, o emulador permitiu validar todas as funcionalidades core.

5.5.2 Validação dos Requisitos do Sistema

Nesta secção apresenta-se a verificação do cumprimento dos Requisitos Funcionais (RF) e Não-Funcionais (RNF), definidos na fase de Engenharia de Software (Capítulo 3.2).

As Tabelas 5.1 e 5.2 apresentam o cumprimento, respetivamente, dos Requisitos Funcionais (RF) e dos Requisitos Não Funcionais (RNF), indicando para cada um o seu estado de implementação e eventuais observações relevantes.

Tabela 5.1: Cumprimento dos Requisitos Funcionais

Código	Descrição	Estado
RF01	Registo e Login com credenciais seguras	Cumprido
RF02	Gestão de perfil e contactos de emergência	Não cumprido
RF03	Botão de emergência com envio de localização	Cumprido
RF04	Histórico de chamadas de emergência	Parcialmente cumprido
RF05	Instruções de primeiros socorros interativas	Cumprido
RF06	Contextualização dos procedimentos	Cumprido
RF07	Módulos de treino e conteúdos teóricos	Não cumprido
RF08	Histórico de formação do utilizador	Não cumprido
RF09	Notificações para serviços e contactos	Não cumprido
RF10	*Logs* detalhados de atividade	Não cumprido

Tabela 5.2: Cumprimento dos Requisitos Não Funcionais

Código	Descrição	Estado
RNF01	Baixa latência nas respostas do sistema	Cumprido
RNF02	Escalabilidade da infraestrutura	Não cumprido
RNF03	Proteção de dados sensíveis (ex. *RGPD*)	Parcialmente cumprido
RNF04	Autenticação segura (hash de passwords)	Cumprido
RNF05	Confidencialidade e integridade dos dados	Parcialmente cumprido
RNF06	Interface intuitiva e simples	Cumprido
RNF07	Compatibilidade multiplataforma (*Android*/*iOS*)	Não cumprido
RNF08	Alta disponibilidade do sistema	Não cumprido
RNF09	Tolerância a falhas e redundância	Não cumprido
RNF10	Modularidade e possibilidade de expansão	Cumprido
RNF11	Manutenção facilitada e documentação clara	Cumprido

Com o objetivo de validar o correto funcionamento da aplicação LIFE, foram realizados testes funcionais e exploratórios centrados nas funcionalidades principais do sistema. Para além da verificação formal dos requisitos definidos nas Tabelas 5.1 e 5.2, foram conduzidos testes manuais com especial enfoque na criação de emergências, envio de localização GPS em tempo real e persistência dos dados em base de dados.

Durante os testes, foram validadas as seguintes operações:

- Envio de um pedido de emergência através do botão SOS;
- Obtenção e geocodificação da localização *GPS* atual do utilizador;
- Armazenamento correto dos dados nas tabelas *emergency* e emergency_localization;
- Receção de resposta *JSON* positiva por parte do middleware *PHP*.

As Figuras 5.2 e 5.3 apresentam exemplos reais dos dados persistidos nas respetivas tabelas da base de dados life:

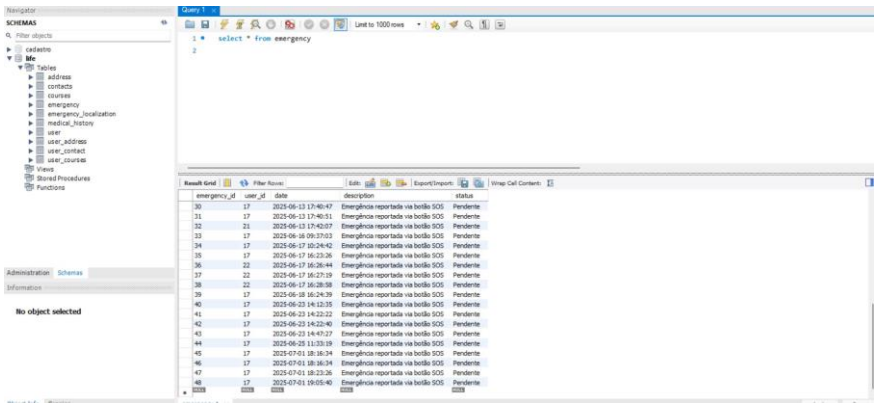

Figura 5.2: Registos da tabela *emergency* após envio de *SOS*

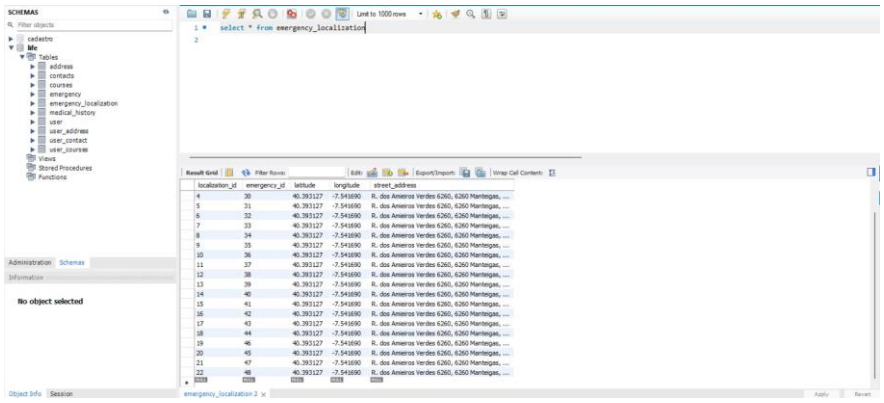

Figura 5.3: Dados de localização armazenados na emergency_localization

Para além das emergências, também foram efetuados múltiplos testes de registo de utilizadores, com diferentes combinações de dados — incluindo nomes fictícios e valores simulados como "asd" ou "ola" — com o intuito de validar o *endpoint* create_user.php, o correto *hashing* das passwords, e a inserção adequada na base de dados. A Figura 5.4 ilustra os registos criados na tabela *user*, confirmando a presença dos campos esperados, como password_hash e created_at.

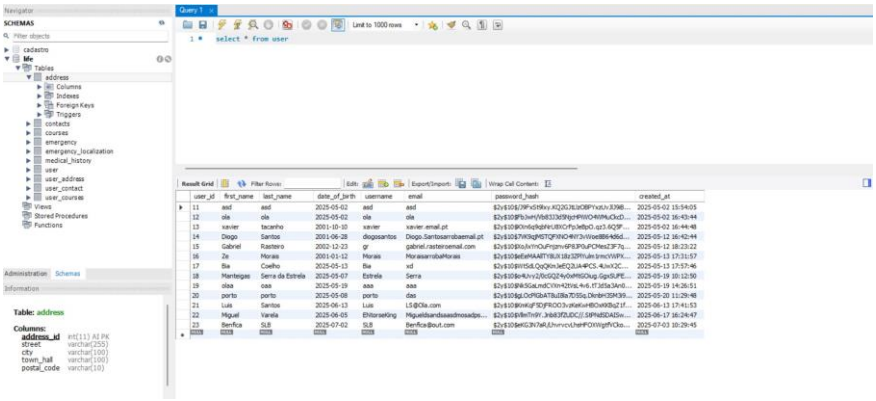

Figura 5.4: Registos de utilizadores criados durante os testes

Apesar de não terem sido realizados testes automatizados nem testes em dispositivos físicos, os testes manuais efetuados no emulador Android demonstraram a estabilidade da aplicação, a consistência na comunicação entre cliente e servidor, e o correto funcionamento do fluxo principal da aplicação.

5.6 Conclusões

Os testes manuais realizados ao longo do desenvolvimento permitiram validar as funcionalidades principais da aplicação **"LIFE"** em ambiente controlado. Apesar de não terem sido utilizados dispositivos físicos ou *frameworks* de testes automatizados, a execução em emulador demonstrou que os componentes críticos — como o registo de utilizadores, a autenticação, o envio de emergências e o armazenamento de localizações — funcionam de forma consistente e sem erros.

A análise direta da base de dados confirmou a persistência correta dos dados relevantes (utilizadores, emergências, localizações), assegurando que os *endpoints* em *PHP* estão a cumprir a sua função. Embora algumas funcionalidades planeadas não tenham sido implementadas nesta fase, o núcleo do sistema — centrado na resposta a emergências e na consulta de primeiros socorros — encontra-se estável e funcional.

Este processo de validação serviu também para identificar pontos de melhoria e requisitos ainda não cumpridos, que poderão ser endereçados em fases futuras do projeto.

Capítulo

6

Demonstração e Validação

6.1 Introdução

Este capítulo apresenta capturas de ecrã da aplicação **LIFE**, com o objetivo de demonstrar visualmente as principais funcionalidades implementadas e validar a sua execução. As imagens foram recolhidas durante sessões de testes realizadas num emulador *Android* configurado para simular o uso da aplicação em situações reais. Cada figura é acompanhada de uma breve descrição contextual, permitindo uma apreciação clara do funcionamento da interface e das interações com o utilizador.

6.2 Registo e Autenticação de Utilizadores

A aplicação **LIFE** permite o registo e autenticação de utilizadores através de uma interface simples e funcional, garantindo segurança e controlo de acesso às funcionalidades principais. O processo é acompanhado de validações visuais e comunicação com o servidor para garantir consistência dos dados.

As Figuras 6.1 e 6.2 mostra os ecrãs iniciais de registo e login, onde os utilizadores introduzem os seus dados pessoais e credenciais.

Figura 6.1: Formulário de registo inicial

Figura 6.2: Ecrã de login com campos preenchidos

O sistema realiza validações dos campos obrigatórios e apresenta mensagens de erro quando estes estão em falta (Figuras 6.3 e 6.4). Também são verificadas a estrutura do e-mail e a robustez da password.

Figura 6.3: Aviso de campos obrigatórios em falta

Figura 6.4: Erro apresentado para e-mail inválido

A aplicação aplica regras de segurança para palavras-passe. As Figuras 6.5 e 6.6 apresentam mensagens associadas à verificação da robustez e da coincidência da password com a confirmação.

Figura 6.5: Validação da robustez da password

Figura 6.6: Aviso de passwords não coincidentes

Para facilitar a introdução da data de nascimento, é apresentado um seletor de data (Figura 6.7).

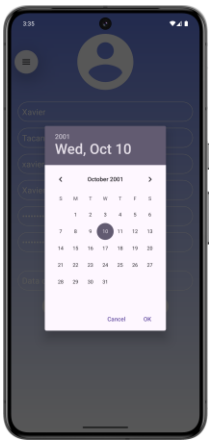

Figura 6.7: Seleção da data de nascimento com *DatePicker*

A aplicação também fornece mensagens claras em caso de erro de autenticação. A Figura 6.8 apresenta um exemplo em que o utilizador insere credenciais inválidas ou inexistentes.

Figura 6.8: Mensagem de erro ao tentar autenticar com credenciais inválidas

6.3 Eventos de Emergência

Após a autenticação, o utilizador é direcionado para a página principal da aplicação (Figura 6.9).

Figura 6.9: Página principal da aplicação após login

Ao pressionar o botão de emergência, o utilizador acede ao ecrã com mapa e botão SOS (Figura 6.10).

Figura 6.10: Ecrã com botão SOS e mapa com localização GPS

Após ativação do SOS, é apresentada uma confirmação (Figura 6.11).

Figura 6.11: Mensagem de confirmação após envio do pedido de emergência

6.4 Instruções de Primeiros Socorros

A Figura 6.12 apresenta os tópicos iniciais de primeiros socorros.

Figura 6.12: Ecrã com a lista de tópicos de primeiros socorros

Ao selecionar um tópico, são apresentados os subtópicos (Figura 6.13).

Figura 6.13: Subtópicos disponíveis dentro da categoria RCP

A Figura 6.14 ilustra a apresentação passo-a-passo das instruções.

Figura 6.14: Passos da RCP para adulto, com instruções passo-a-passo

6.5 Conclusão

Com base nas imagens e testes apresentados ao longo deste capítulo, é possível afirmar que a aplicação **LIFE** cumpre com sucesso os principais fluxos definidos nos requisitos funcionais. As funcionalidades de registo e autenticação, envio de pedidos de emergência com localização em tempo real, e a consulta estruturada de instruções de primeiros socorros demonstraram estabilidade e coerência no seu funcionamento.

As interações com o servidor *PHP* e a base de dados *MySQL* também foram validadas, comprovando que os dados são corretamente processados e armazenados. As mensagens visuais e comportamentos da interface foram consistentes com os objetivos do projeto, mesmo em situações simuladas de erro ou ausência de permissões.

Apesar de não estarem ainda implementadas todas as funcionalidades previstas, como a gestão de perfil ou os módulos de treino, os testes demonstram que o núcleo funcional da aplicação está consolidado, servindo como base sólida para futuras extensões.

Capítulo

7

Conclusões e Trabalho Futuro

7.1 Conclusões Principais

O desenvolvimento da aplicação **LIFE** permitiu a consolidação de competências práticas em várias áreas da Engenharia Informática, com especial destaque para o desenvolvimento mobile, comunicação cliente-servidor e persistência de dados em base de dados relacional.

O principal objetivo — disponibilizar uma aplicação funcional e intuitiva para resposta a emergências — foi concretizado com sucesso. A aplicação permite:

- Efetuar registos e autenticação segura de utilizadores;
- Acionar rapidamente um pedido de emergência (botão *SOS*);
- Obter e partilhar automaticamente a localização geográfica do utilizador;
- Armazenar os dados relevantes na base de dados *MySQL*;
- Apresentar instruções de primeiros socorros de forma clara e estruturada.

A arquitetura modular (*frontend Android*, *middleware PHP*, base de dados *MySQL*) revelou-se eficaz e escalável. A comunicação via REST e JSON foi implementada com sucesso, com tratamento de erros e validação de dados no cliente e no servidor.

Apesar de algumas funcionalidades previstas inicialmente não terem sido totalmente concretizadas (ex. notificações, histórico detalhado ou módulos

de formação), a aplicação apresenta um núcleo robusto e coerente, validado por testes funcionais em ambiente de emulador.

7.2 Trabalho Futuro

Apesar dos objetivos principais terem sido cumpridos, existem vários aspetos que podem ser melhorados ou expandidos numa futura versão do sistema. Seguem-se as propostas mais relevantes:

- **Testes em Dispositivos Reais**: O projeto foi testado apenas em emulador. A validação em smartphones físicos é essencial para avaliar o desempenho real, compatibilidade com diferentes versões do Android e qualidade da geolocalização em campo;
- **Notificações em Tempo Real**: Envio de alertas automáticos para contactos de emergência e/ou serviços através de *SMS*, notificações *push* ou integração com *APIs* externas;
- **Gestão de Perfil Completa**: Permitir a edição e visualização completa dos dados pessoais, moradas e contactos de emergência diretamente na aplicação;
- **Histórico e Exportação**: Adicionar um histórico consultável de emergências e formação, com possibilidade de exportar dados (PDF, CSV) para fins médicos ou de auditoria;
- **Integração de Módulos Educativos**: Implementar quizes e conteúdos multimédia sobre primeiros socorros, reforçando a vertente formativa da aplicação;
- **Reforço de Segurança**: Explorar autenticação biométrica, *tokens* de sessão ou mecanismos adicionais de encriptação para aumentar a segurança dos dados dos utilizadores;
- **Disponibilização na *Cloud***: Migrar o servidor local (*WAMP*) para uma infraestrutura *cloud*, garantindo maior disponibilidade, *backups* e redundância.
- **Versão para *iOS***: Expandir a aplicação para a plataforma *iOS*, alcançando um público mais alargado.
- **Utilização noutros Contextos**: A arquitetura do projeto **LIFE** pode ser adaptada para outras áreas, como assistência a idosos, monitorização

de pacientes ou resposta a situações de violência doméstica, onde a partilha imediata de localização e informação pode salvar vidas.

Bibliografia

[1] Google LLC. Android studio – the official ide for android development, 2025. Acedido a 23 de junho de 2025.

[2] Oracle Corporation. Java platform, standard edition, 2025. Acedido a 23 de junho de 2025.

[3] World Wide Web Consortium (W3C). Extensible markup language (xml) 1.0, 2025. Acedido a 24 de junho de 2025.

[4] Microsoft. Visual studio code – code editing. redefined, 2025. Acedido a 25 de junho de 2025.

[5] The PHP Group. Php: Hypertext preprocessor, 2025. Acedido a 27 de junho de 2025.

[6] Inc. Postman. Postman – api development environment, 2025. Acedido a 30 de junho de 2025.

[7] WampServer. Wampserver – windows web development environment, 2025. Acedido a 30 de junho de 2025.

[8] Oracle Corporation. Mysql – the world's most popular open source database, 2025. Acedido a 27 de junho de 2025.

[9] Inc. GitHub. Github – where the world builds software, 2025. Acedido a 26 de junho de 2025.