



INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

DEPARTAMENTO DE ENGENHARIA ELETRÓNICA E DE  
TELECOMUNICAÇÕES E COMPUTADORES (DEETC)

---

LEIM

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA  
UNIDADE CURRICULAR DE PROJETO

---

## Talk.id - Communication App



Miguel Leitão (46309)

Rui Correia (48594)

Tiago Figueiredo (49154)

### *Orientadores*

---

*Professor* Paulo Trigo

*Professora* Helga Henriques

*Professora* Joana Portela

*Professora* Ana Frade

---

*Julho, 2024*



# Resumo

Desenvolver uma aplicação móvel, utilizando a *framework Flutter* e linguagem *Dart*, com comunicação a uma *web app*, desenvolvida com a *framework Django* e linguagem *Python*. O objetivo é proporcionar aos pacientes que passaram pelo processo de traqueostomia uma forma de comunicar fácil e eficiente com os profissionais de saúde, para que estes tenham uma melhor compreensão sobre como e de que forma podem ajudar o paciente.

Cada paciente tem acesso à aplicação móvel, onde podem comunicar presencialmente, frente a frente, através da ferramenta *text-to-speech* e perguntas/respostas predefinidas para comunicação rápida sem que o paciente tenha a necessidade de escrever. O processo de comunicação também pode ser efetuado remotamente, na ausência de enfermeiro perto do paciente, através do envio das necessidades do paciente para uma *API REST*, incorporada na *web app*, com acesso exclusivo aos profissionais de saúde, que guarda na base de dados *PostgreSQL* as mensagens/necessidades enviadas pelos pacientes nesses pedidos REST. As páginas da web visualizadas pelos enfermeiros vão obter os dados à base de dados onde também podem ser atualizadas em tempo real caso novos dados sejam inseridos na base de dados.

As diversas funcionalidades implementadas na aplicação móvel foram facilmente testadas e validadas porque o *Flutter* possui as funcionalidades Hot Restart e/ou Hot Reload que permite que, sem ter que parar a execução da app, atualizar a mesma com as novas funcionalidades implementadas e assim testar rapidamente. Após cada implementação de uma nova funcionalidade de ambas as apps, estas eram testadas simulando um contexto de utilização real, como se fosse um paciente e um enfermeiro a interagirem.

A ideia mais relevante deste projeto foi, precisamente, tornar a comunicação remota entre paciente e profissional de saúde possível, para que os profissionais de saúde não necessitem de estar próximos do paciente, permitindo uma melhor alocação de tempo e recursos para melhorar eficiência num ambiente onde tal é uma necessidade.

Este projeto despertou o nosso interesse pela possibilidade de ajudar estas pessoas que se encontram sem forma de comunicar e expressar o que querem

ou o que sentem às pessoas ao seu redor.

**Palavras-Chave:** Aplicação móvel; Web App; Paciente; Profissionais de saúde; Comunicação

# Abstract

Mobile app development, using *Flutter* framework [3] with *Dart* [2] as the programming language, with the ability to communicate with a web app, made with *Django* framework and *Python* as the programming language. The goal is to provide patients who have undergone the procedure of tracheostomy a way to easily and efficiently communicate with the health professionals, so that they have a better understanding regarding how they can help the patient.

Each patient has access to the mobile app, which they can use to communicate personally, face to face, with the health professionals, using the help of a *text-to-speech* tool and predefined questions/answers for quick chatting, so that the patient doesn't have the need to type on the keyboard. The communication can happen remotely too, in the absence of a nurse near the patient, by sending the patient needs to a *REST API*, embedded in the *web app*, with exclusive access to health professionals, which saves in the *PostgreSQL* database the messages/needs sent by patients in the REST requests. The web pages viewed by nurses will get the data to the database where it can also be updated in real time if new data is entered into the database.

The various functionalities implemented in the mobile application were easily tested and validated because *Flutter* has Hot Restart and/or Hot Reload functionalities that allow, without having to stop running the app, update it with the new implemented features and thus test quickly. After each implementation of a new functionality of both apps, they were tested simulating a real usage context, as if it were a patient and a nurse interacting.

The most relevant purpose of this project is, precisely, making the remote communication between patient and health professional possible so the health professional doesn't have to be administrating the patient at all times in person, allowing better allocation of time and resources to improve efficiency in an environment where such is a necessity.

This project awakened our interests by showing the possibility to help these people that find themselves without any means of expressing what they want or what they feel to the people around them.

**Key-words:** Mobile app; Web App; Patient; Health professionals; Communication

# Índice

<b>Resumo</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Índice</b>	<b>v</b>
<b>Lista de Tabelas</b>	<b>vii</b>
<b>Lista de Figuras</b>	<b>ix</b>
<b>1 Introdução</b>	<b>1</b>
<b>2 Tecnologias de Suporte</b>	<b>3</b>
<b>3 Modelo Proposto</b>	<b>5</b>
3.1 Requisitos . . . . .	5
3.2 Fundamentos . . . . .	9
3.3 Abordagem . . . . .	13
3.3.1 Arquitetura do Sistema . . . . .	13
3.3.2 Esboços iniciais da aplicação . . . . .	14
3.3.3 Interface do utilizador . . . . .	16
3.3.4 Comunicação . . . . .	18
<b>4 Implementação do Modelo</b>	<b>21</b>
4.1 Aplicação . . . . .	21
4.1.1 Interface do Utilizador . . . . .	21
4.1.2 Comunicação . . . . .	23
4.1.3 Autenticação . . . . .	28
4.2 Web App . . . . .	29

4.3	Logotipo . . . . .	31
<b>5</b>	<b>Validação e Testes</b>	<b>33</b>
5.1	Validação das interfaces do utilizador . . . . .	33
5.1.1	Aplicação . . . . .	33
5.1.2	Web App . . . . .	39
5.2	Teste de Registo de um utilizador . . . . .	46
5.3	Comunicação entre Aplicação e Web app . . . . .	47
<b>6</b>	<b>Conclusões e Trabalho Futuro</b>	<b>51</b>
	<b>Bibliografia</b>	<b>53</b>



# Lista de Tabelas

3.1	Requisitos Funcionais . . . . .	6
3.2	Requisitos Não Funcionais . . . . .	7



# Lista de Figuras

3.1	Diagrama dos Casos de Utilização . . . . .	9
3.2	<i>Flutter</i> . . . . .	11
3.3	Arquitetura <i>Django</i> . . . . .	12
3.4	Diagrama de Arquitetura do Sistema . . . . .	14
3.5	<i>Sketch</i> Página Principal . . . . .	15
3.6	<i>Sketch</i> Página de uma categoria . . . . .	15
3.7	Página Inicial da Aplicação . . . . .	16
3.8	Página da categoria “Problemas” . . . . .	17
3.9	Página para indicar o local e nível da dor no corpo . . . . .	18
4.1	Página Perguntas/Respostas . . . . .	23
4.2	Página Sim/Não . . . . .	24
4.3	Página Conversas Rápidas . . . . .	25
4.4	<i>Pop Up TTS</i> . . . . .	26
4.5	Escrita no <i>Pop up TTS</i> através do teclado . . . . .	26
4.6	PWA instalação . . . . .	30
4.7	Logo Talk.id . . . . .	31
5.1	Página Principal da Aplicação (tema escuro) . . . . .	35
5.2	Opções da categoria Dificuldade em Respirar . . . . .	36
5.3	Página Comichão . . . . .	37
5.4	Página Dor . . . . .	38
5.5	Página Problemas . . . . .	38
5.6	Página Login <i>Web App</i> . . . . .	39
5.7	Página Principal do <i>Web App</i> . . . . .	40
5.8	Página Administração Django . . . . .	41
5.9	Página de Registo . . . . .	43

5.10	Página <i>Report</i> . . . . .	44
5.11	Manage Accounts . . . . .	45
5.12	Mensagem de sucesso no registo . . . . .	46
5.13	Utilizador registado na base de dados <i>PostgreSQL</i> . . . . .	46
5.14	Mensagem de erro no registo . . . . .	46
5.15	Mensagem Enviada para o Servidor . . . . .	47
5.16	Notificação . . . . .	48
5.17	Mensagens Recebidas na Web App . . . . .	48
5.18	Mensagem Detalhada . . . . .	49

# Capítulo 1

## Introdução

Este projeto consiste na criação de uma aplicação móvel e uma web app cujo objetivo é possibilitar uma forma de comunicação eficiente e fácil entre pacientes, que passaram pelo procedimento de traqueostomia [4], e profissionais de saúde. A traqueostomia consiste num procedimento cirúrgico, consistindo num pequeno orifício que é feito na garganta, sobre a região da traqueia, onde é inserido um tubo para facilitar a entrada de ar nos pulmões, quando a via normal da respiração fica obstruída ou congestionada. Neste procedimento, a respiração passa a fazer-se exclusiva e definitivamente pelo orifício da traqueostomia, deixando o paciente sem a habilidade de falar durante algum tempo.

A aplicação móvel foi desenvolvida para ser suportada num telemóvel ou tablet, sendo o tablet o principal dispositivo em mente, uma vez que este possui maiores dimensões e, por isso, é mais prático e fácil de se usar nesta situação. Foi utilizada a *framework Flutter* [3] com linguagem *Dart* [2] para desenvolver a aplicação.

A aplicação possui diversas categorias de possíveis temas de comunicação para a facilitar e para que o paciente consiga encontrar o que procura e comunicar rapidamente. Estas categorias são: Perguntas/Respostas, Necessidades e Problemas - e dentro destas categorias o paciente pode encontrar vários pedidos e formas de comunicação para se expressar. O paciente tem sempre acesso à ferramenta *text-to-speech* disponível nos idiomas português e inglês para poder comunicar livremente (esta opção está disponível em todas as páginas da aplicação). Adicionalmente a aplicação possui uma “campainha” S.O.S, que toca do lado dos enfermeiros na receção desta mensagem, para

poder chamar o profissional de saúde em caso de emergência. Cada vez que o paciente clica numa opção de comunicação é enviada uma mensagem para a *web app* dos profissionais de saúde.

Neste projeto foi criado, juntamente com a aplicação, uma *web app* para possibilitar uma forma de comunicação remota, proporcionando aos profissionais de saúde maior flexibilidade na atenção dada ao paciente, não precisando de estar constantemente próximos deste. A *web app* foi desenvolvida utilizando a *framework Django* com linguagem *Python*, para o backend, e *HTML*, *JavaScript* e *CSS*, para o frontend. Na *web app* os profissionais também têm a opção de fazer o registo dos dados do paciente, sendo estes armazenados numa base de dados *PostgreSQL*. Este registo é essencial para que os pacientes possam ser devidamente identificados no envio de mensagens, pois ao iniciar a aplicação pela primeira vez, o paciente (com a ajuda do profissional de saúde) deve iniciar sessão com os dados correspondentes. A *web app* é exclusiva aos profissionais de saúde tendo estes de estar registados na base de dados também, para poderem terem acesso à *web app*.

A principal motivação na realização deste projeto é ajudar estas pessoas que se encontram numa situação precária sem a possibilidade de expressar aquilo que sentem, querem e precisam.

O desenvolvimento deste projeto, possibilitou a criação de uma aplicação móvel e *web app* que satisfazem todos os requerimentos e trazem benefícios para a comunicação entre paciente e profissional de saúde.

## Capítulo 2

# Tecnologias de Suporte

Os fundamentos tecnológicos da aplicação móvel desenvolvida com *Flutter* baseiam-se em várias características e ferramentas oferecidas pela *framework*. *Flutter* é uma *framework open source* criada pela *Google* para a construção de aplicações nativas de alta *performance* a partir de um único *source code*, utilizando o motor gráfico *Skia* para renderizar na tela do ecrã diretamente, o que garante desempenho nativo em múltiplas plataformas. A linguagem de programação utilizada é *Dart*, também desenvolvida pela *Google*, sendo esta otimizada para a construção de interfaces de utilizador e oferece recursos poderosos como tipagem estática opcional, um sistema robusto de gerenciamento de *packages* e suporte para programação assíncrona, facilitando a escrita de código limpo e eficiente.

*Widgets* são os blocos de construção fundamentais do *Flutter*, descritos como *StatelessWidget* ou *StatefulWidget*, que definem como a interface de utilizador deve ser apresentada e como deve responder a mudanças dinâmicas. Bibliotecas como *material.dart* e *cupertino.dart* fornecem componentes visuais que seguem as diretrizes de *design* do *Material Design* e do *iOS*, permitindo que os desenvolvedores criem interfaces nativas e consistentes. Outras bibliotecas, como *animation.dart* e *gestures.dart*, oferecem suporte para animações e interações gestuais, aprimorando a interatividade e a fluidez da aplicação.

O *Flutter* também providencia uma biblioteca *text-to-speech*, *flutter\_tts.dart*, e uma biblioteca para permitir que a aplicação suporte vários idiomas, neste caso dois (português e inglês), *flutter\_gen/gen\_l10n/app\_localizations.dart*.

Além disso, o *Flutter* possui um ecossistema robusto com diversas ferra-

mentas e bibliotecas de terceiros que facilitam o desenvolvimento, depuração e testes. Ferramentas como o *Flutter DevTools* e o *Hot Reload* aceleram o ciclo de desenvolvimento, permitindo iterações rápidas e correção de *bugs* em tempo real. A comunidade ativa contribui com *packages* e *plugins* que ampliam as funcionalidades padrão do *Flutter*, tornando-o uma solução versátil para diversos tipos de projetos.



# Capítulo 3

## Modelo Proposto

Este capítulo visa apresentar o modelo proposto para o desenvolvimento da aplicação móvel juntamente com a *web app*. O objetivo é criar uma aplicação e web app *user-friendly*, para que a sua utilização seja fácil, eficiente e que melhore a comunicação entre paciente e profissional de saúde.

### 3.1 Requisitos

Os requisitos ajudam a clarificar as funcionalidades essenciais do projeto, de forma a que este cumpra os objetivos e necessidades a atender aos utilizadores. Na seguinte tabela podem se observar os principais requisitos funcionais:

Requisito	Descrição	Categoria
R1	O sistema tem que possuir uma aplicação móvel capaz de comunicar	Evidente
R2	O sistema tem que ter uma web app para gerir os pacientes e receber mensagens da aplicação	Evidente
R3	O sistema tem que comunicar entre Aplicações e Servidor	Invisível
R4	O sistema tem que ter uma ferramenta <i>text-to-speech</i>	Evidente
R5	O sistema tem que fazer registo de utilizadores na base de dados <i>PostgreSQL</i>	Evidente
R6	O sistema tem que ter uma interface de utilizador (UI) intuitiva e atrativa	Adorno

Tabela 3.1: Requisitos Funcionais

As funcionalidades, características e aspetos que garantem o funcionamento e qualidade de todo o projeto e que não são visíveis diretamente por parte do utilizador designam-se requisitos não funcionais, sendo alguns desses os seguintes:

Requisito	Descrição
R1	<i>Performance</i> da aplicação: a aplicação deve estar otimizada para expressar o que o paciente quer sem nenhum precalço, seja presencialmente ou remotamente, não podendo haver <i>delay</i> .
R2	Gestão de mensagens: a <i>web app</i> deve receber todas as mensagens que o paciente enviar para o servidor sem qualquer tipo de <i>delay</i>
R3	Gestão de utilizadores: todos os utilizadores devem estar registados na base de dados com as devidas permissões, sendo que o registo só pode ser feito na <i>web app</i> por utilizadores que sejam profissionais de saúde ou administradores.
R4	Usabilidade: tanto a aplicação como a <i>web app</i> deve ser intuitivo e de fácil utilização.
R5	Identificação de pacientes: o paciente deve ter uma sessão iniciada durante o uso da aplicação com os dados correspondentes a este, para que seja identificado no envio de mensagens para o servidor e posteriormente do servidor para a <i>web app</i> .

Tabela 3.2: Requisitos Não Funcionais

Para listar as interações entre os utilizadores e o sistema recorreu-se ao modelo de casos de utilização que, tal como o nome, refere todos os casos de utilização para estas interações, sendo alguns destes os seguintes:

- Registo de um profissional de saúde: O profissional de saúde é registado na base de dados, a partir da *web app*, por um profissional de saúde já registado ou pelo *Super User*.
- Registo de um paciente: Um profissional de saúde faz o registo dos dados do paciente para que este possa aceder à aplicação com as suas devidas informações.
- Iniciar a aplicação: O paciente inicia a aplicação, efetuando um único início de sessão para que as mensagens enviadas estejam identificadas com o nome do paciente que as enviou, e pode realizar qualquer tipo de comunicação que desejar.
- Selecionar categoria de comunicação: O paciente seleciona, das diferentes categorias aquela que contém o que o paciente deseja expressar.
- Comunicação via *text-to-speech*: O paciente escreve no campo de escrita da componente *text-to-speech* e seleciona “*Play*” para converter o texto em som.
- Comunicação via seleção de um botão: O paciente seleciona o botão que expressa um caso específico, enviando uma mensagem para o servidor que posteriormente é enviada para a *web app*.
- Visualização de mensagens: Na *web app*, os profissionais de saúde recebem uma notificação que uma mensagem foi enviada por um paciente, a qual podem abrir e visualizar os detalhes da mensagem.

Para representar os casos de utilização fez-se um diagrama que descreve as funcionalidades propostas para o sistema que será projetado, sendo uma excelente ferramenta para o levantamento dos requisitos funcionais do sistema.

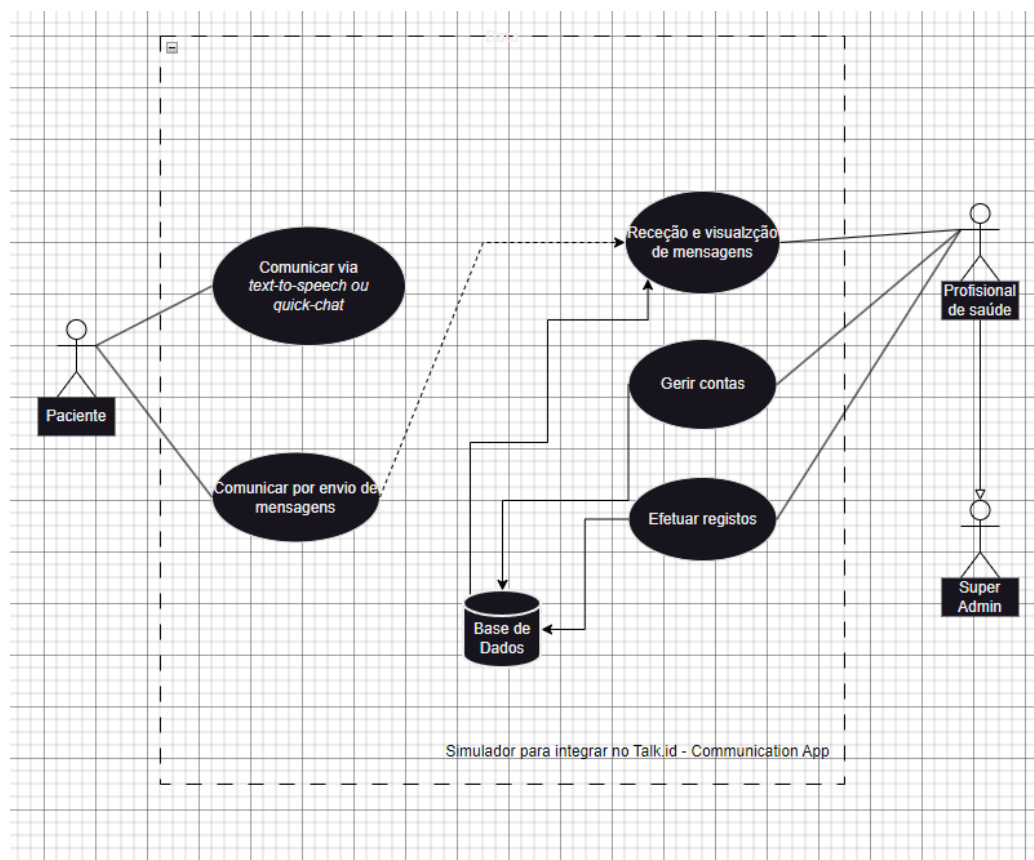


Figura 3.1: Diagrama dos Casos de Utilização

## 3.2 Fundamentos

Este capítulo contém uma visão geral sobre projetos e estudos relevantes que proporcionaram uma grande ajuda e compreensão no desenvolvimento deste projeto. Os principais estudos foram a aprendizagem das *frameworks* *Flutter*, *Django* e *PostgreSQL*, tendo em conta que no início do desenvolvimento do projeto nenhum dos elementos do grupo tinha trabalhado com nenhuma destas *frameworks*.

1. O *Flutter* é uma estrutura de desenvolvimento *open-source* criada pela Google para criar interfaces de utilizador multiplataforma. Permite desenvolver aplicações para dispositivos móveis, *web* e *desktop* usando um único conjunto de *source code*. Utiliza a linguagem de programação *Dart* e destaca-se pela sua performance rápida, facilidade de uso e capacidade de proporcionar uma experiência consistente em várias plataformas [3].
2. *Django* é uma *framework* de desenvolvimento *web open source*, escrito em *Python*. Facilita a criação rápida de aplicações *web* robustas e escaláveis, seguindo o princípio do design padrão *MVC* (*Model-View-Controller*) ou *MVT* (*Model-View-Template*, em *Django*). *Django* é conhecido pela sua simplicidade, eficiência e segurança, oferecendo uma vasta gama de funcionalidades prontas para uso, como autenticação de utilizador, administração de conteúdo, *ORM* (*Object-Relational Mapping*), entre outros. É amplamente utilizado e possui uma comunidade ativa que contribui para o seu crescimento e evolução contínua [6].
3. O *PostgreSQL* é um sistema de gestão de base de dados relacional *open source*, conhecido pela sua robustez, confiabilidade e conformidade com padrões *ANSI SQL*. Oferece suporte a consultas complexas, transações *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade), integridade referencial e uma ampla gama de tipos de dados, incluindo tipos personalizados e extensíveis. O *PostgreSQL* é altamente escalável e adequado para aplicações que exigem alta disponibilidade, segurança e desempenho. Além disso, possui uma comunidade ativa que contribui para o seu desenvolvimento e suporte contínuos [7].

Com a junção destes componentes foi possível a criação de uma aplicação móvel que comunica com uma *web app*, para permitir uma melhor comunicação entre profissionais de saúde e pacientes que não conseguem falar.



Figura 3.2: *Flutter*

Para visualizar a aplicação foi utilizado um emulador *Android Pixel Tablet API 34* para simular um Tablet e *Pixel 3* para simular um telemóvel.

Para o desenvolvimento da web app foi utilizada a arquitetura *Django*. O *Django* é baseado na arquitetura *MVT* (*Model-View-Template*) que possui as seguintes três partes:

- **Modelo:** O modelo atuará como interface dos dados. É responsável pela manutenção dos dados. É a estrutura lógica de dados por trás de toda a *web app* e é representada por uma base de dados (geralmente base de dados relacionais como *MySQL*, *Postgres*).
- **Visualização:** A visualização é a interface do utilizador que é vista no *browser* ao renderizar um *site*. É representado por ficheiros *HTML/CS-S/Javascript* e *Jinja*.
- **Template:** O *template* consiste em partes estáticas da saída *HTML* desejada, como também uma sintaxe especial que descreve como o conteúdo dinâmico será inserido.

Esta arquitetura pode ser observada na Figura 3.3.

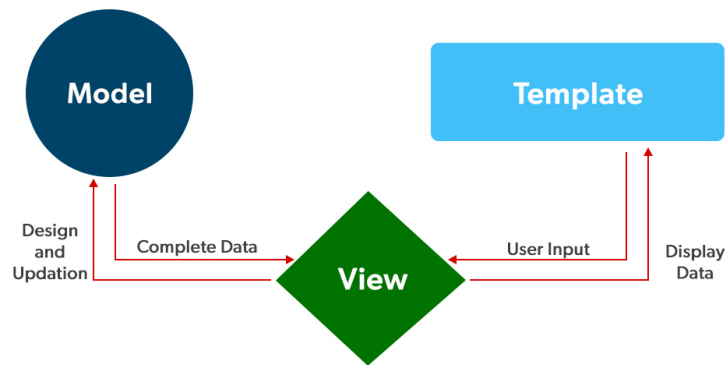


Figura 3.3: Arquitetura *Django*

Para o armazenamento de dados no decorrer da aplicação e *web app* foi utilizado o *PostgreSQL*. A escolha deste sistema foi baseada na sua robustez, fiabilidade e conformidade com padrões *ANSI SQL*. Suporta grandes volumes de dados, tipos de dados personalizados, extensões como *PostGIS*, e alta segurança com métodos avançados de autenticação e cifração. É altamente escalável. No projeto, o *PostgreSQL* foi utilizado para garantir o



armazenamento de dados, autenticação dos utilizadores e as suas devidas permissões seja na utilização da aplicação móvel ou na *web app*.

Para a comunicação entre aplicação móvel e *web app* é utilizado uma *API RESTful* [1], arquitetura para programação de interfaces de aplicações que usa pedidos *HTTP* para aceder e usar dados.

Uma *API RESTful* melhora a escalabilidade, permitindo pedidos independentes e que respostas sejam armazenadas em cache temporariamente no lado do cliente, melhorando o desempenho e eficiência. A simplicidade de uso, com métodos *HTTP* intuitivos, e ferramentas de documentação automática como o *Swagger*, tornam o desenvolvimento e a manutenção mais fáceis. A *API* é flexível e extensível, suportando a atualização modular e a adição de novas funcionalidades sem alterar a base existente. Além disso, oferece segurança robusta com autenticação, autorização e uso de *HTTPS*. A eficiência é garantida com o uso de *JSON* para a troca de dados, e a reutilização de código é promovida, centralizando a lógica de negócio no servidor.

## 3.3 Abordagem

Nesta subsecção estão enumerados, em detalhe, os contributos centrais no desenvolvimento do projeto que promoveram a sua diferenciação.

### 3.3.1 Arquitetura do Sistema

O sistema possui quatro principais componentes: Servidor, Base de Dados, *Mobile App* e *Web App*. O servidor será a componente responsável por lidar com os pedidos *HTTP* e disponibiliza várias funcionalidades para os “clientes”. A base de dados é responsável por armazenar toda a informação necessária para o funcionamento das apps, como: dados dos utilizadores, necessidades/mensagens, entre outros. A mobile app funciona como um elemento que permite comunicar com os profissionais de saúde através do envio de mensagens via *HTTP* para o servidor, usando *API REST*, que guarda a informação das mesmas, para que a web app esteja atualizada com a informação pretendida. A web app serve de consulta e manuseamento dos dados armazenados na base de dados, de onde são obtidos para disponibilizar a informação no browser e é também atualizada em “tempo real” caso nova informação seja adicionada.

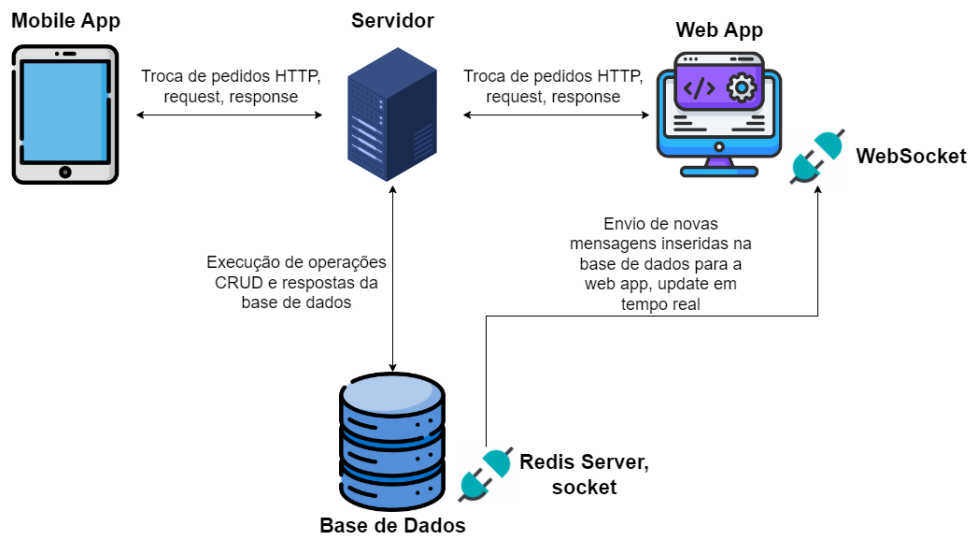
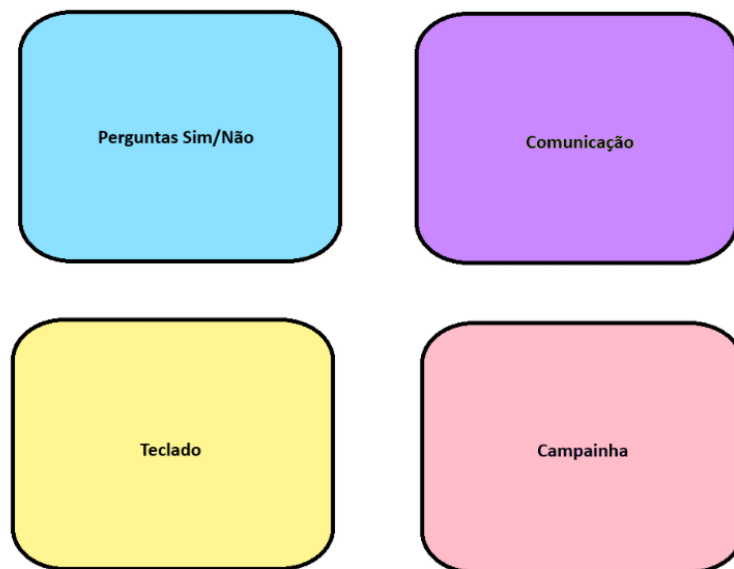
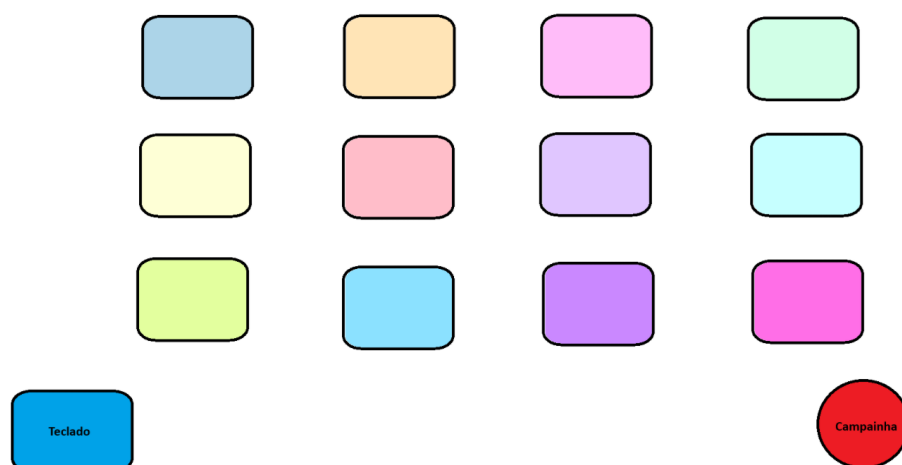


Figura 3.4: Diagrama de Arquitetura do Sistema

### 3.3.2 Esboços iniciais da aplicação

A criação de esboços é essencial no desenvolvimento de uma aplicação móvel, pois permitem visualizar e planejar as interfaces e funcionalidades. Estes desenhos simples ajudam a explorar diferentes *layouts* e interações rapidamente, facilitando ajustes antes da codificação. Nas Figuras 3.5 e 3.6 estão apresentados os esboços iniciais da aplicação móvel, interface gráfica da página principal e da página de uma categoria.

Figura 3.5: *Sketch* Página PrincipalFigura 3.6: *Sketch* Página de uma categoria

### 3.3.3 Interface do utilizador

Após explorarmos várias opções, como por exemplo, *Android Studio*, para o desenvolvimento da aplicação móvel, concluímos que o *Flutter* é a ferramenta ideal para projetar a nossa aplicação com uma interface de utilizador (UI) intuitiva, rica e atraente, e devido ao uso de *widgets* altamente personalizáveis. O desempenho das aplicações com *Flutter* é comparável ao das nativas, garantindo uma experiência de utilizador fluida e responsiva.

A UI inclui todas as categorias de comunicação, num formato grande e explícito para que o utilizador saiba onde pode encontrar aquilo que deseja expressar. Ao selecionar uma das categorias, é redirecionado para a listagem de expressões daquela categoria. As categorias estão divididas em “Perguntas/Respostas”, “Necessidades” e “Problemas”, e, tal como os nomes indicam, cada categoria engloba um tema de comunicação ao qual as expressões envolvidas correspondem.

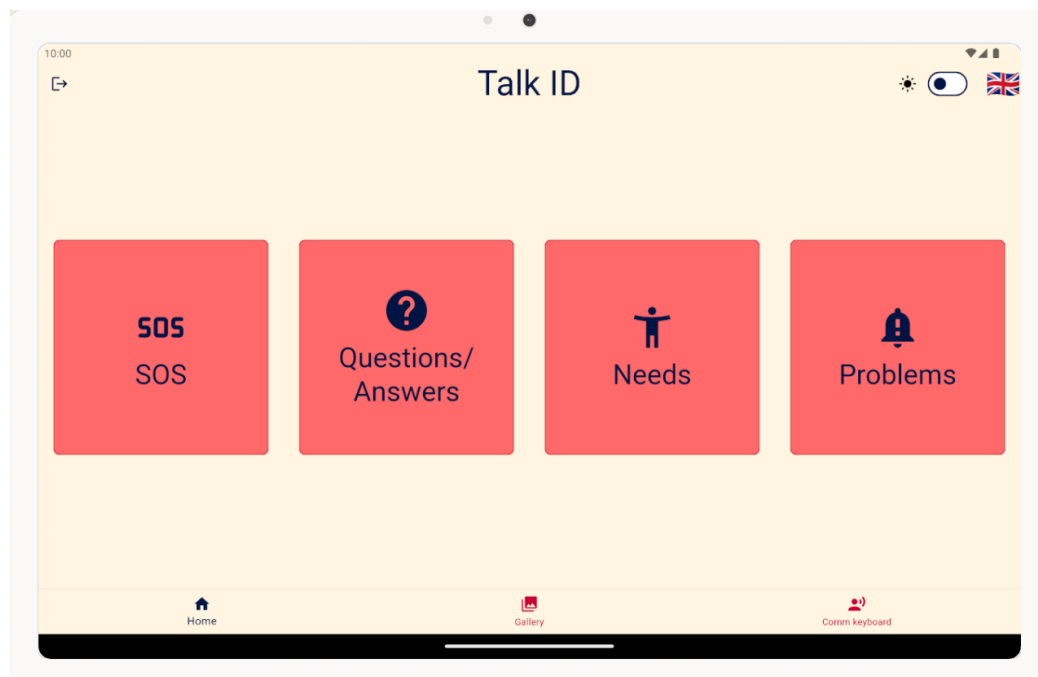


Figura 3.7: Página Inicial da Aplicação

Por exemplo: se o utilizador sentir dor no lado esquerdo do peito, deve selecionar a categoria dos problemas que irá redirecioná-lo para a página da categoria, mostrando, entre outras, a opção da dor, que deve ser selecionada

pelo utilizador nesta situação. Ao seleccionar a opção da dor, o utilizador é direccionado para a página da dor onde deve indicar que sente dor no lado esquerdo do peito com o nível e raio da dor. O utilizador pode, então, enviar mensagem para o servidor, que será enviada para a *web app* dos profissionais de saúde. Contudo, caso haja um profissional de saúde por perto, frente a frente, e atento ao paciente, este poderá simplesmente utilizar a ferramenta *text-to-speech*.

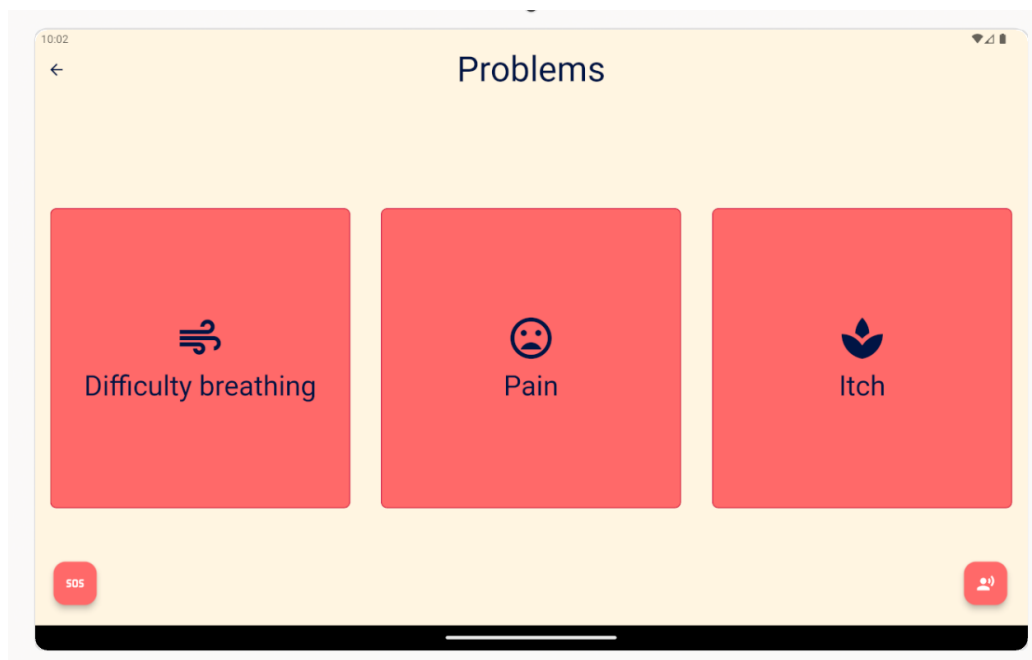


Figura 3.8: Página da categoria “Problemas”

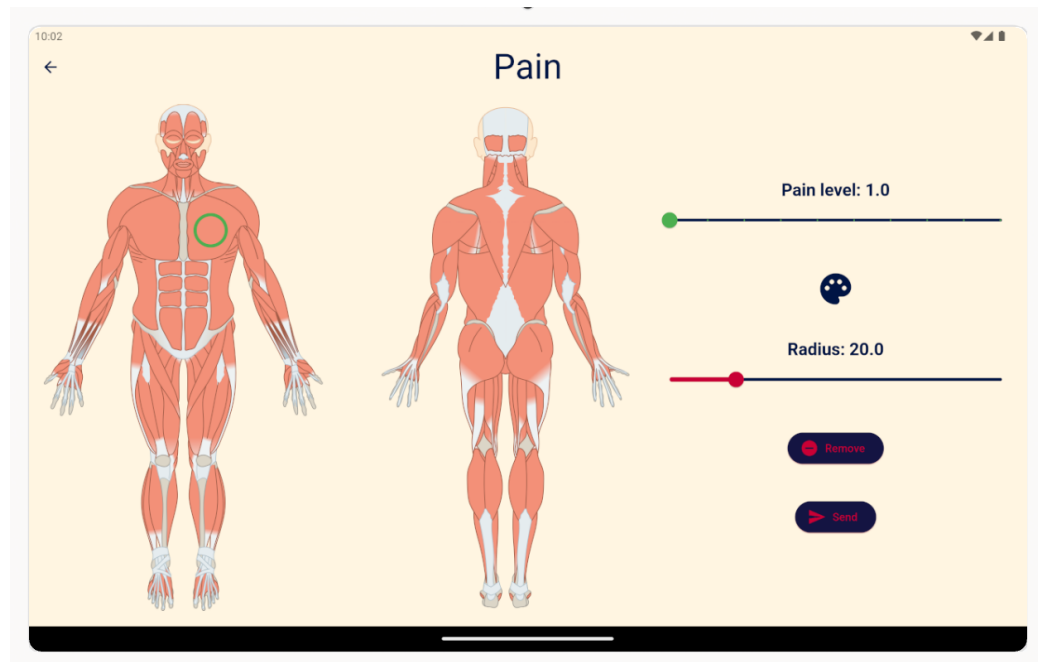


Figura 3.9: Página para indicar o local e nível da dor no corpo

### 3.3.4 Comunicação

O foco deste projeto é a comunicação, e para tal foi desenvolvida uma aplicação que possibilita a comunicação entre paciente e profissionais de saúde de forma presencial ou remota. Como foi referido na subsecção 3.3.3, as formas de comunicação estão divididas em categorias, promovendo a organização e facilidade para o paciente encontrar o que deseja. Contudo, existem algumas categorias que servem apenas para comunicação presencial e outras para tanto presencial, frente a frente, como remota, ausência de enfermeiro por perto.

A categoria para comunicação presencial é a categoria “Perguntas/Respostas”, esta categoria tem como objetivo, tal como o nome indica, fazer perguntas ou responder de forma rápida, estando dividida em duas subcategorias: “Sim/Não” para responder a perguntas de “sim” e “não”, e “*Quick chat*” que contém perguntas, respostas e outras expressões pré-definidas que o utilizador pode seleccionar para comunicar rapidamente.

No caso de o paciente (utilizador) necessitar de expressar algo mais elaborado, tem como alternativa a ferramenta “*text-to-speech*” (*TTS*), que permite

converter algo que o utilizador escreva em som e reproduzir. Esta funcionalidade encontra-se em todas as páginas da aplicação excepto na página para indicar a dor e na página para indicar a comichão, para facilitar a comunicação e uso da aplicação.

O resto das categorias existentes na aplicação servem para comunicar presencialmente e remotamente, sendo que uma mensagem é enviada para o servidor sempre que o utilizador selecciona uma opção destas categorias.

Esta mensagem posteriormente é enviada para a *web app* onde os profissionais são alertados através de uma notificação *push notification* a indicar a nova mensagem. Para a implementação da push notification foi criado um “*service worker*” onde o utilizador ao clicar no botão da campainha que aparece na página do browser, este subscreve o serviço obtendo a subscrição com os dados necessários para receber as notificações, como por exemplo, o endpoint, uma chave pública (p256dh) e um token de autenticação, estes guardados na base de dados, com isto sempre que é recebida uma mensagem de SOS é enviada uma notificação para os enfermeiros que têm o serviço subscrito. A campainha de emergência ou S.O.S está presente em todas páginas da aplicação tal como a ferramenta *TTS*, para que o paciente possa chamar um profissional de saúde a qualquer altura em caso de emergência. A mensagem também alerta os profissionais de saúde através de um alarme visual(piscar a vermelho) e para além do alarme visual, também emite um alarme sonoro, caso a mensagem seja do tipo S.O.S. Ao aceder à mensagem na *web app* é apresentado ao profissional de saúde os detalhes da mensagem:

- nome da categoria;
- nome da seleção;
- nome do paciente que enviou a mensagem;
- data e hora;
- figura do corpo humano (frente e costas) para indicar o local da dor ou comichão caso a mensagem esteja relacionada com um dessas duas possibilidades;
- opção para os profissionais de saúde introduzirem a solução que vão adotar para responder ao pedido do paciente.





# Capítulo 4

## Implementação do Modelo

Com base no capítulo 3 Modelo Proposto, passou-se à implementação do modelo, que passou pela implementação da Aplicação móvel e da *web app* e posteriormente a comunicação entre estes.

### 4.1 Aplicação

Nesta secção estão apresentadas as etapas da implementação da aplicação móvel.

#### 4.1.1 Interface do Utilizador

Tal como foi referido no Capítulo 3, secção 3.3, é essencial que as interfaces do utilizador, na aplicação, sejam grandes e bem explícitas, para facilitar a sua utilização tendo em conta a situação precária em que o utilizador se encontra. O uso das bibliotecas do Flutter permitiu a implementação dos *esboços* apresentados no capítulo 3, secção 3.3 e promoveu a implementação de interfaces que não só satisfazem estes requisitos, mas que também são visualmente atrativas e ricas, virtudes que proporcionam uma melhor experiência de utilizador.

As bibliotecas do Flutter foram utilizadas para implementar a aplicação de modo a garantir experiência de utilizador eficiente e acessível foram:

- Construção da interface gráfica: *flutter/material.dart*;
- Formatação de datas: *intl*;

- Suporte a múltiplos idiomas: *flutter\_gen/gen\_l10n/app\_localizations.dart*.

A navegação entre as diferentes partes da aplicação é gerida pelo *Navigator*, que proporciona transições suaves e manutenção do estado entre páginas.

O uso de *provider* e *ThemeManager* permite alternar o tema da aplicação entre temas claro e escuro, melhorando a experiência de utilizador.

A aplicação utiliza a classe *BoxData* para estruturar os dados dos botões, associando ícones e textos relevantes. Esta abordagem modular facilita a manutenção e expansão do código. O uso de listas de objetos *BoxData* permite a criação dinâmica dos botões, mantendo o código limpo.

A aplicação é responsiva, ajustando a disposição dos botões conforme a orientação do dispositivo usando *OrientationBuilder*. Funções como *calculateMargin* e *calculateIconSize* ajustam dinamicamente o tamanho e a margem dos ícones, assegurando uma interface consistente.

O suporte de múltiplos idiomas é gerido pela biblioteca do Flutter: *flutter\_gen/gen\_l10n/app\_localizations.dart*, tendo sido utilizado o idioma do nosso país, português, e o idioma mais falado no mundo, inglês, facilitando o uso por utilizadores de diferentes nacionalidades.

### 4.1.2 Comunicação

#### Presencial

A **comunicação presencial** refere-se à presença de um profissional de saúde ou de outra pessoa com a qual o paciente comunica através da aplicação, selecionando aquilo que tenciona expressar. Categorias como “Perguntas/-Respostas” e ferramenta *TTS* (*Text-To-Speech*) foram implementadas com o objetivo de permitir ao paciente participar numa conversa pessoalmente, e por isso, as mensagens não são enviadas para o servidor. Contudo todas as categorias podem ser utilizadas para comunicação presencial, mas como estas duas categorias só realizam comunicação presencial, serão explicadas para este contexto.



Figura 4.1: Página Perguntas/Respostas

Ao selecionar a categoria “Perguntas/Respostas”, o utilizador é direcionado para uma página onde se encontram duas opções para temas de conversa (Figura 4.1):

- “Sim/Não”: Se esta opção for selecionada, o utilizador é direcionado para uma página com dois botões indicados como “Sim” e “Não”, com o propósito de responder a perguntas de sim e não. A classe *Yes-NoAnswersPage* implementa esta funcionalidade, utilizando a biblioteca *flutter\_gen* para internacionalização dos textos, garantindo que a

aplicação possa ser facilmente adaptada a diferentes idiomas (mencionado na secção anterior). A interface integra funcionalidades de *TTS* através da classe *TabletComKeyboardPage* para fornecer *feedback* de áudio quando as opções são seleccionadas.



Figura 4.2: Página Sim/Não

- “Conversa Rápida”: Se esta opção for seleccionada, o utilizador é direccionado para uma página com diferentes categorias onde se encontram listas de perguntas/respostas rápidas que o utilizador pode seleccionar numa conversa. O utilizador pode adicionar perguntas/respostas a cada uma das listas, ou até mesmo adicionar uma nova categoria para conversa rápida. A classe *QuickChatPage* gerencia estas listas de saudações, respostas, perguntas personalizadas e categorias adicionadas pelo utilizador, armazenando esses dados localmente usando a biblioteca *shared\_preferences*. Diálogos como *\_showGreetDialog*, *\_showAnswersDialog* e *\_showQuestionsDialog* permitem ao utilizador gerenciar as listas personalizadas.



Figura 4.3: Página Conversas Rápidas

A funcionalidade *TTS* (*Text-To-Speech*) desempenha um papel crucial na aplicação, pois permite ao utilizador reproduzir o texto introduzido através de som, dando flexibilidade ao paciente para comunicar algo mais elaborado.

A funcionalidade *TTS* está incorporada na aplicação utilizando a classe *TabletComKeyboardPage*, que contém o método estático *speak*. Este método é responsável por transformar o texto introduzido pelo utilizador em áudio, utilizando a biblioteca do Flutter *flutter\_tts*. A implementação garante que qualquer texto exibido na interface pode ser convertido em áudio quando necessário. O *TTS* possui dois idiomas para a reprodução do texto introduzido, em formato de som, sendo estes os idiomas implementados na aplicação (português e inglês) através da biblioteca Flutter *flutter\_gen/gen\_l10n/app\_localizations.dart*. Todas as páginas da aplicação têm o *TTS* disponível em formato *pop-up*.

Figura 4.4: *Pop Up TTS*Figura 4.5: Escrita no *Pop up TTS* através do teclado

## Remota

A **comunicação remota**, tem como objetivo permitir ao utilizador comunicar com um profissional de saúde quando nenhum se encontra em proximidade para que possa comunicar presencialmente.

A solução elaborada para este problema foi o envio de mensagens para um servidor quando o paciente seleciona uma opção de comunicação, que posteriormente é enviada para a *web app* desenvolvida para os profissionais

de saúde receberem estas mensagens e gerirem as contas dos pacientes (informação elaborada na secção 4.2).

As categorias “Necessidades” e “Problema” são as que se enquadram neste contexto, bem como o botão de S.O.S.. Apesar destas categorias mandarem mensagens para o servidor também podem ser utilizadas para comunicação presencial.

Um dos principais componentes desenvolvidos é a classe *Utils*, que centraliza métodos essenciais para enviar mensagens de diferentes tipos para o servidor.

A implementação da comunicação através do envio de mensagens envolveu a utilização da biblioteca *http* do Dart, que facilita pedidos *HTTP* para interação com *APIs* externas, neste caso *RESTful API*. Esta escolha tecnológica foi fundamentada na necessidade de comunicação eficiente e segura entre o dispositivo móvel e o servidor, garantindo assim que as mensagens do utilizador sejam enviadas e recebidas de forma rápida e confiável.

Na classe *Utils*, foram desenvolvidos métodos assíncronos para diferentes tipos de mensagens, tais como *sendMessage*, *sendMessageCoordinates* e *sendSOS*. Cada um destes métodos é responsável por construir corretamente o corpo da mensagem *JSON*, conforme esperado pela *RESTful API* do servidor, e enviar esses dados através de pedidos *POST*.

A comunicação no servidor é gerenciada através do *Django Channels*, utilizando *Redis* como *backend* para *WebSocket* [5]. Quando uma mensagem é inserida com sucesso na base de dados *PostgreSQL*, o servidor *Django* envia uma notificação via *WebSocket* para todos os utilizadores e enfermeiros conectados. Isto permite que o conteúdo *HTML* da página da *web app* seja atualizado dinamicamente com *JavaScript*, garantindo que todos recebam as atualizações em tempo real.

O método *sendMessage* é responsável por enviar uma mensagem genérica para o servidor. Este método recebe como parâmetros o tipo de mensagem (*msgType*), informações adicionais (*msgInfo*) e opcionalmente um nível (*level*). O *msgType* identifica o tipo específico de mensagem a ser enviada, e cada tipo está armazenado na base de dados *PostgreSQL*, correspondendo a uma das seguintes categorias:

1. Mensagem do tipo S.O.S
2. Mensagem do tipo Necessidades

### 3. Mensagem do tipo Problemas

O *msgInfo* contém os detalhes da mensagem, enquanto o *level* pode ser utilizado para indicar a intensidade da dor ou comichão do paciente na mensagem, se aplicável. Estes parâmetros são então convertidos para *JSON* e enviados ao servidor através de um pedido *POST HTTP*. Quando a mensagem é recebida e processada pelo servidor, uma notificação é enviada via *WebSocket* para atualizar a interface do utilizador em tempo real.

Já o método *sendMessageCoordinates* é utilizado para enviar mensagens que incluem coordenadas específicas da zona da dor ou comichão, caso a mensagem esteja relacionada com uma destas opções. O método recebe como parâmetros o *ID* da mensagem (*msgID*), a largura (*imgWidth*) e altura (*imgHeight*) da imagem de referência, e uma lista de objetos *CustomCircle* que representam, em formato de círculos, a localização da dor ou comichão. O método transforma os dados em *JSON* e envia para o servidor, possibilitando o armazenamento e processamento das coordenadas recebidas. Após o processamento, o servidor envia uma notificação via *WebSocket* para atualizar as coordenadas na interface do utilizador (profissional de saúde).

Por fim, o método *sendSOS* é uma funcionalidade específica que envia uma mensagem de S.O.S para o servidor. Quando acionado, é chamado o método *sendMessage* com parâmetros predefinidos para indicar a emergência. Após o envio bem-sucedido da mensagem, a resposta do servidor é verificada para garantir que a comunicação foi efetuada com sucesso, e uma mensagem de confirmação é exibida ao utilizador. A notificação via *WebSocket* também é enviada para garantir que todos os utilizadores conectados sejam informados da emergência em tempo real. A opção para enviar uma mensagem do tipo S.O.S encontra-se em todas as páginas da aplicação.

#### 4.1.3 Autenticação

Para garantir que as mensagens recebidas na web app estejam devidamente identificadas pelo paciente que as enviou, é efetuada uma única autenticação quando o paciente acede, pela primeira vez, à aplicação a partir da página de *login*.

A página de *login*, denominada *TabletLoginPage*, é uma *StatefulWidget*, que permite manter e gerir o estado do formulário de *login*. Os principais



componentes desta página incluem um formulário de entrada de dados, controladores de texto para capturar o nome de utilizador e a palavra-passe, e uma instância de *FlutterSecureStorage* para armazenar o *token* de autenticação de forma segura.

Quando a página é inicializada, o método *initState* chama a função *\_checkLoginStatus* para verificar se o utilizador já possui um *token* de autenticação armazenado. Caso exista um *token* válido, o utilizador é redirecionado automaticamente para a página principal da aplicação, *TabletHomePage*, utilizando o *Navigator* do Flutter.

O formulário de *login* é criado utilizando o *widget Form*, que está envolvido por um *Padding* para fornecer espaçamento adequado. O formulário contém dois campos de entrada de texto, um para o nome de utilizador e outro para a palavra-passe, ambos com verificações para garantir que não estão vazios antes de permitir o envio.

Ao pressionar o botão de *login*, a função *onPressed* verifica se o formulário é válido. Se for, é feita uma chamada assíncrona à função *login*, que envia um pedido *HTTP POST* para o servidor, contendo o nome de utilizador e a palavra-passe no corpo do pedido, utilizando a biblioteca *http* do Dart. O servidor responde com um *token* de autenticação se as credenciais forem válidas, ou seja, se o utilizador estiver registado na base de dados PostgreSQL.

Se a resposta do servidor indicar sucesso (código de *status* 200), o *token* de autenticação é extraído da resposta *JSON* e armazenado de forma segura utilizando *FlutterSecureStorage*. Em seguida, o utilizador é redirecionado para a página principal da aplicação. Caso contrário, é apresentada uma mensagem de falha na autenticação e o utilizador permanece na página de *login*.

O utilizador pode efetuar *logout* na página principal da aplicação.

## 4.2 Web App

Em adição à aplicação móvel, foi desenvolvido uma *web app* em *Django*, para os profissionais de saúde, com o objetivo de gerir as contas dos pacientes e as mensagens que estes enviam. De modo a facilitar a utilização, a *web app* está disponível sob a forma de *app* com uso da tecnologia *PWA* (*Progressive Web App*), não tendo que aceder através do browser e introduzir o URL para

utilizar a web app. Para a criação desta, foi usado um package do django, *django-pwa*, onde, para o uso do mesmo, foi criado e configurado um *service worker* para proceder à instalação da app. Ao entrar na página *index.html*, é gerado um ficheiro *manifest.json* para que possa ser feita a instalação da app.

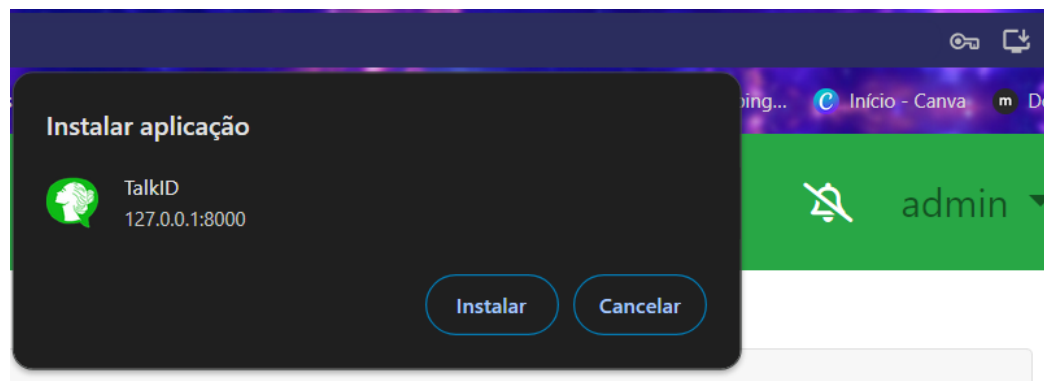


Figura 4.6: PWA instalação

Utilizando a *framework Django*, foram implementadas funcionalidades que permitem o registo, autenticação e gestão de contas de utilizadores. Semelhante à aplicação móvel, é necessário realizar um processo de autenticação para utilizar as funcionalidades da web app, com validação dos dados da autenticação através da base de dados *PostgreSQL*.

O sistema permite que os pacientes enviem mensagens, incluindo mensagens de S.O.S e coordenadas específicas associadas a zonas de dor ou comichão, etc. Estas mensagens são armazenadas numa base de dados *PostgreSQL* e distribuídas em “tempo real” para os profissionais de saúde através de *WebSockets*, configurando o *CHANNEL\_LAYERS* das settings do projeto com um servidor *Redis* para a comunicação em canais e lidar com o envio e receção de mensagens, assim sendo torna mais rápido a leitura dos pedidos dos pacientes o que pode fazer a diferença e melhora a experiência do utilizador com código HTML “gerado dinamicamente”. A web app foi concebida para ser intuitiva e de fácil utilização, proporcionando uma interface clara e acessível para a monitorização e resposta rápida às necessidades dos pacientes.

## 4.3 Logotipo

A criação de um logotipo para a aplicação e *web app* oferece benefícios estratégicos e funcionais significativos. Um logotipo eficaz atua como a face pública da marca, proporcionando uma representação visual instantaneamente reconhecível e transmitindo os valores e a missão da empresa. Ajudando a construir uma conexão emocional com o público, fomentando a lealdade do cliente e reforçando a consistência da marca em todos os pontos de contato.

Além disso, um logotipo distintivo diferencia a marca da concorrência, tornando-a mais atrativa e memorável. Do ponto de vista funcional, facilita a comunicação visual, permitindo que os consumidores entendam rapidamente a essência da marca. Este investimento traz retornos a longo prazo, com o potencial de se tornar um ativo duradouro que evolui juntamente com a empresa. Em resumo, um logotipo bem concebido é essencial para a construção de identidade, fortalecimento da marca, diferenciação competitiva e eficácia comunicativa, contribuindo significativamente para o sucesso e crescimento sustentável da empresa.[8]



Figura 4.7: Logo Talk.id



# Capítulo 5

## Validação e Testes

Neste capítulo, são apresentadas as validações e os testes realizados para garantir a qualidade, a fiabilidade e a funcionalidade da aplicação móvel e da web app desenvolvidos. O processo de teste incluiu testes unitários, de integração e de sistema, assegurando a correção dos componentes individuais, a interação adequada entre módulos e o comportamento global do sistema em condições reais de uso.

Foram também efetuados testes de usabilidade e desempenho para garantir uma interface intuitiva e responsiva, bem como a capacidade do sistema de lidar com as cargas de trabalho esperadas. Testes de segurança identificaram e mitigaram vulnerabilidades potenciais, protegendo os dados dos utilizadores.

### 5.1 Validação das interfaces do utilizador

Nesta secção são apresentadas as interfaces do utilizador, tanto na aplicação móvel como na web app.

#### 5.1.1 Aplicação

O utilizador, no primeiro acesso, depara-se com a página de *login* para fazer a autenticação (única), sendo direcionado para a página principal da aplicação assim que o *login* for efetuado. A página principal fornece várias opções com diferentes funcionalidades, com resultados diferentes, tais como:

- Enviar mensagem S.O.S;

- Ir para a página Perguntas/Respostas (*Questions/Answers*)
- Ir para a página Necessidades (*Needs*)
- Ir para a página Problemas (*Problems*)
- Aceder à galeria (*Galery*)
- Aceder ao teclado *TTS*
- Alterar o tema da aplicação (claro/escuro)
- Alterar idioma (português/inglês)
- Efetuar *logout*

Se o utilizador decidir navegar para a página dos problemas, seleccionando o botão "Problemas", irá deparar-se com três temas de comunicação, e adicionalmente a ferramenta *TTS* e a campanha de emergência (S.O.S):

- Dificuldade em respirar (*Difficulty breathing*), onde pode seleccionar a causa da dificuldade, e será enviada a mensagem com a opção que seleccionar;

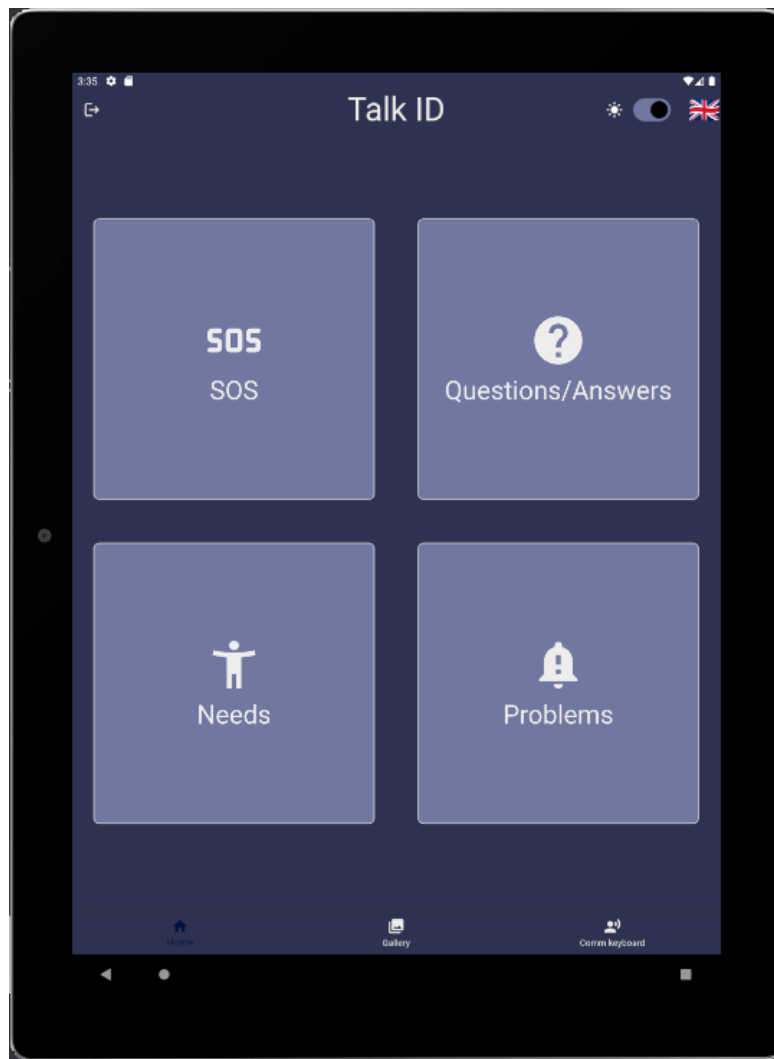


Figura 5.1: Página Principal da Aplicação (tema escuro)

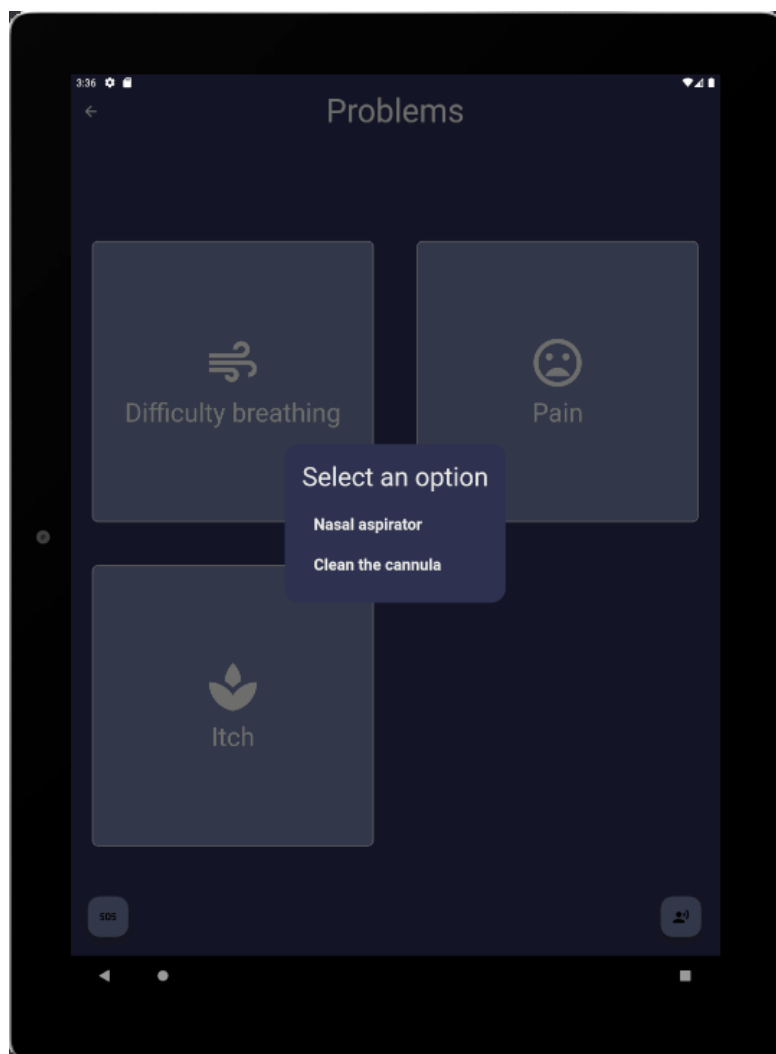


Figura 5.2: Opções da categoria Dificuldade em Respirar

- Comichão (*Itch*), direciona o utilizador para a página Comichão, onde pode indicar o local da comichão no corpo e será enviada uma mensagem para o servidor com essa informação;



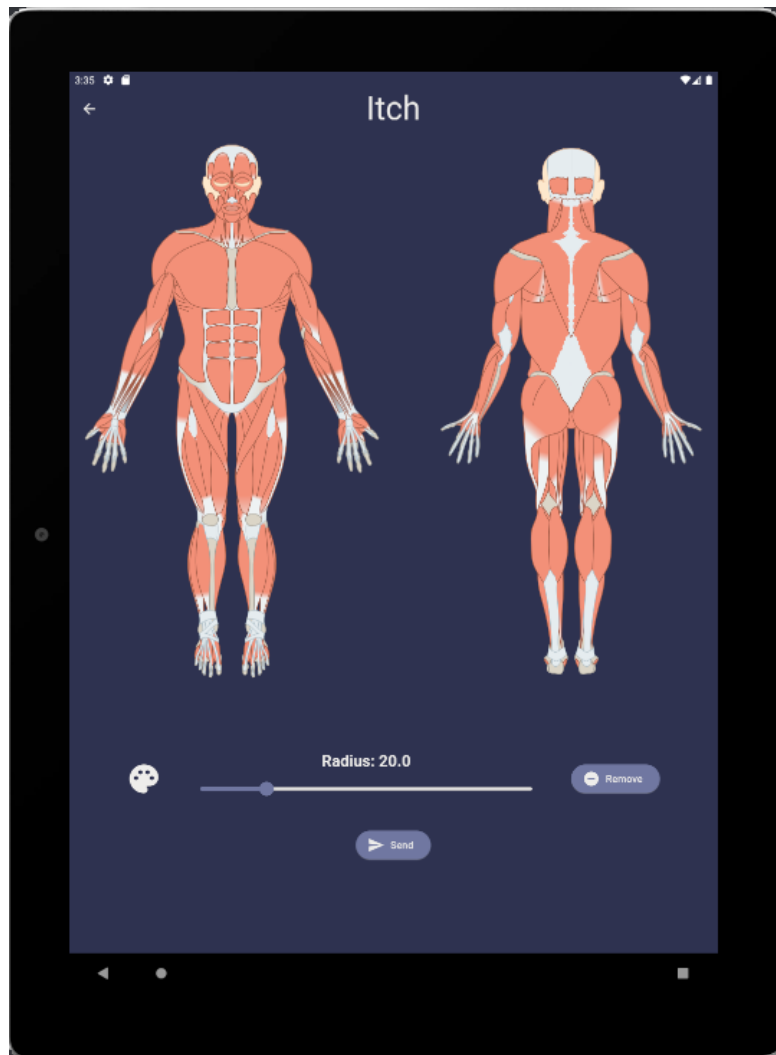


Figura 5.3: Página Comichão

- Dor (*Pain*), direciona o utilizador para página Dor, onde pode indicar o local da dor no corpo e será enviada uma mensagem para o servidor com essa informação.

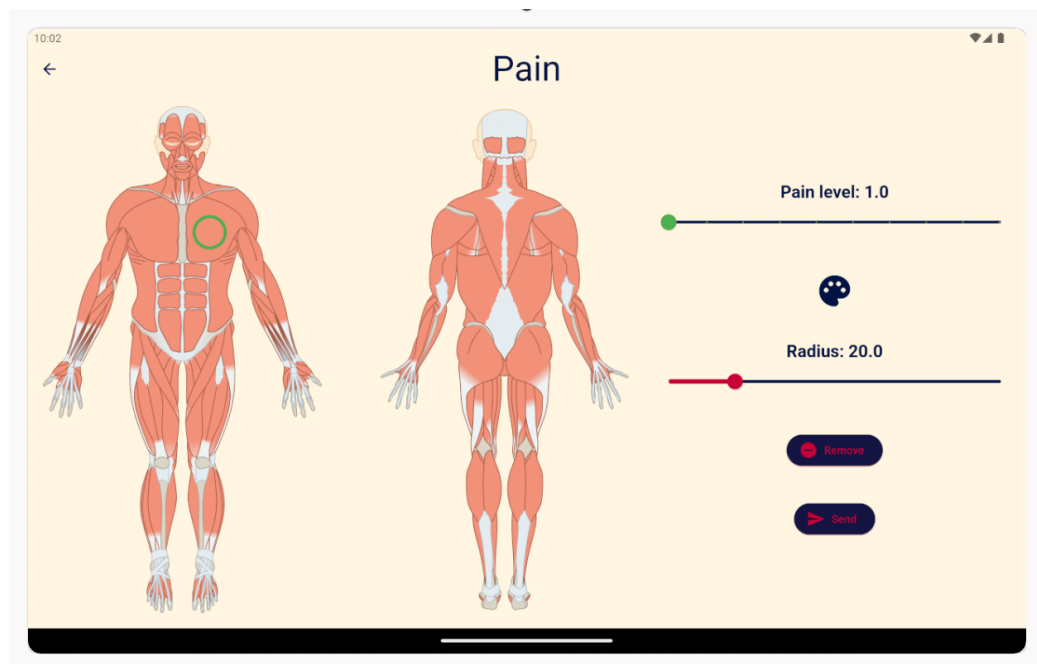


Figura 5.4: Página Dor

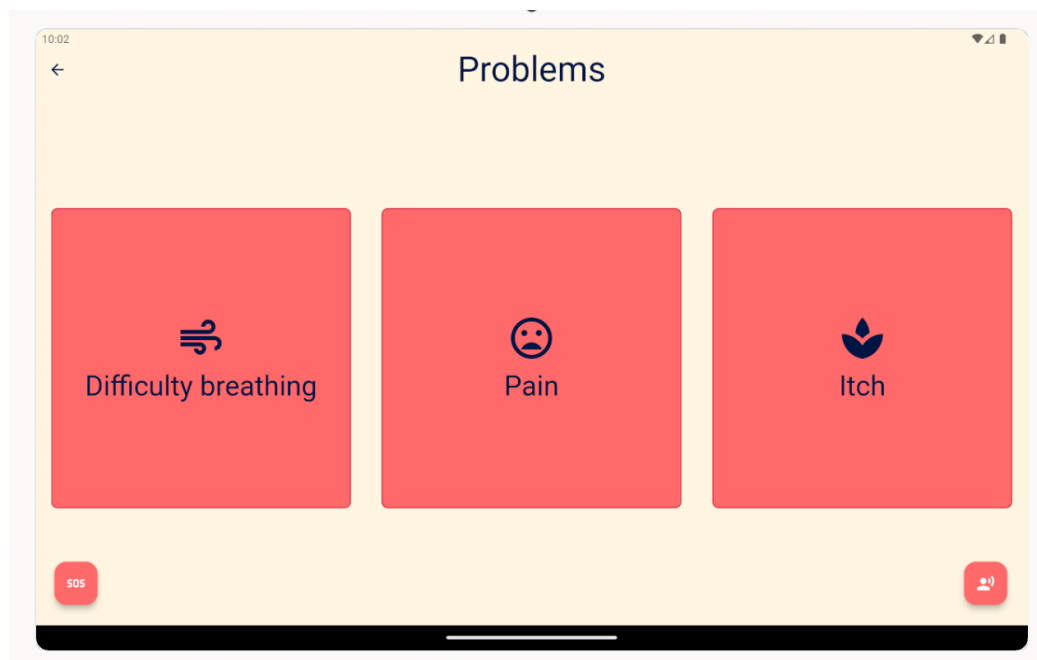
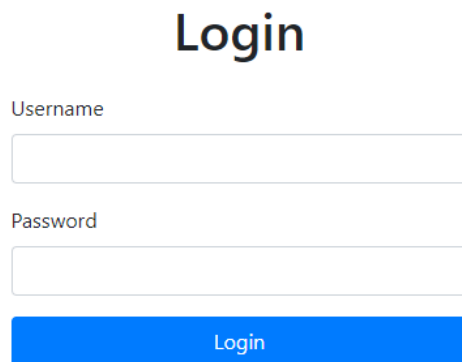


Figura 5.5: Página Problemas

### 5.1.2 Web App

A primeira página que aparece quando o utilizador (profissional de saúde) quando acede à *web app* é a página *Login* para que faça a devida autenticação, validando os dados introduzidos através dos utilizadores registados na base de dados *PostgreSQL*.



The image shows a login form titled "Login" in a large, bold, black font. Below the title, there are two input fields. The first field is labeled "Username" in a small, gray font, and the second field is labeled "Password" in a small, gray font. Both fields are empty and have a light gray border. Below the password field, there is a blue button with the text "Login" in white, centered on the button.

Figura 5.6: Página Login *Web App*

Após ser efetuado o *login* o utilizador é direcionado para a página principal da *web app*, sendo esta a página onde as mensagens do paciente são entregues. Esta escolha fundamenta-se na premissa de que a comunicação eficaz e imediata entre pacientes e profissionais de saúde é crucial para a qualidade do atendimento e a prontidão na resposta às necessidades dos pacientes.

A página de receção de mensagens centraliza todas as comunicações enviadas pelos pacientes, permitindo que os profissionais de saúde acessem rapidamente às informações vitais e respondam de forma eficiente. Esta funcionalidade foi priorizada porque possibilita uma monitorização contínua do estado dos pacientes, facilitando a identificação e a resposta a situações de emergência ou a necessidades específicas de forma ágil. Esta página contém uma pesquisa por filtros para as mensagens, dando a possibilidade ao utilizador de pesquisar por mensagens específicas, a partir do tipo da mensagem, período, paciente que enviou a mensagem, etc. Se o utilizador seleccionar uma mensagem, uma versão detalhada sobre esta aparece, onde o utilizador pode visualizar todos os detalhes da mensagem e até mesmo indicar a solução que vai exercer no cuidado do paciente que a enviou.

Todas as páginas da *web app* contêm uma barra de navegação (*navbar*) para que o utilizador possa silenciar ou ativar as notificações sonoras, e um botão *dropdown* para realizar a navegação entre as diferentes páginas.

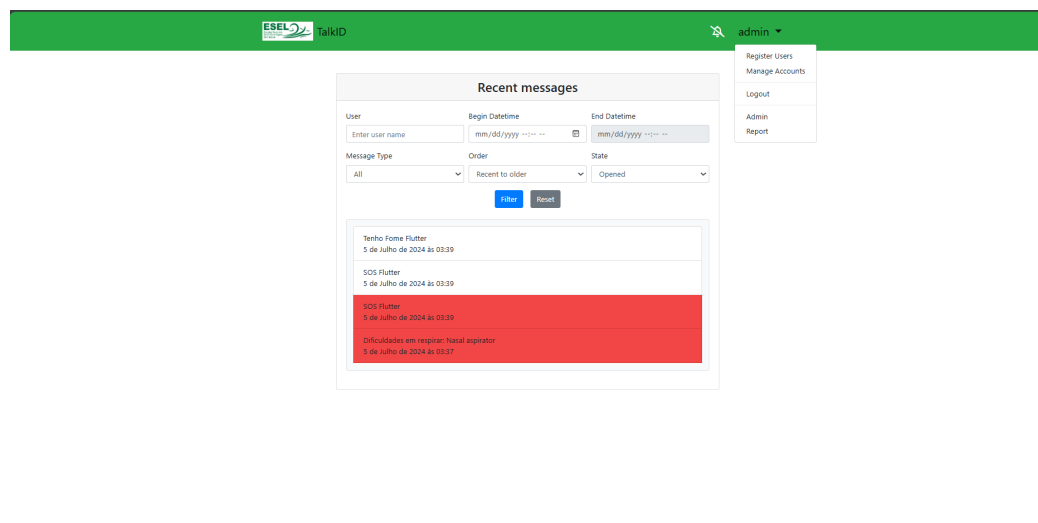


Figura 5.7: Página Principal do *Web App*

A *web app* é gerenciada inicialmente por um *Super User* podendo este aceder a qualquer funcionalidade da *web app* e até mesmo alterar dados da base de dados e predefinições da *web app* através da página “Administração do Django”.

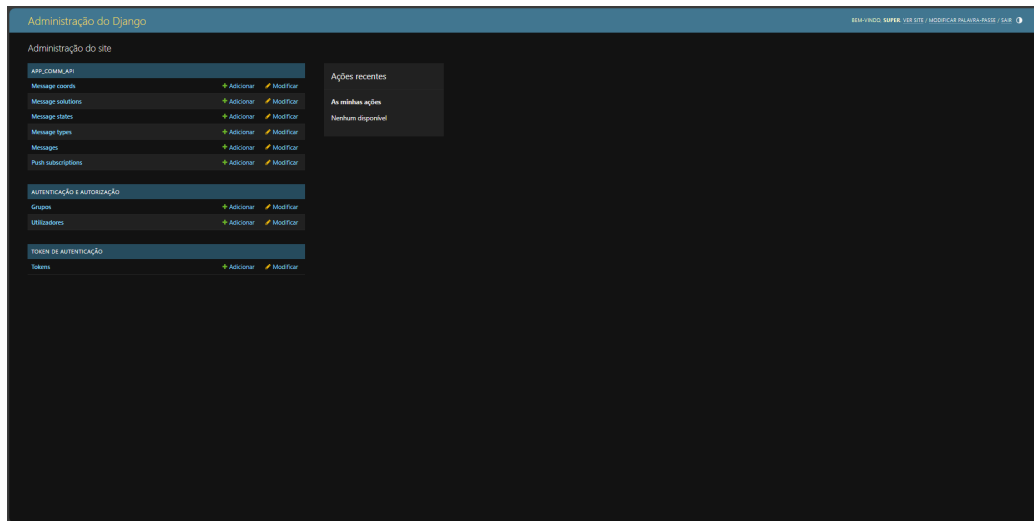

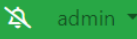


Figura 5.8: Página Administração Django

A página de registo permite que os profissionais de saúde registem novos utilizadores na base dados *PostgreSQL*. No registo de um novo utilizador é lhe atribuído um *role* que dita as suas permissões, existindo três tipos:

1. ***Admin***: Contém todas as permissões, podendo aceder a qualquer funcionalidade e registar qualquer tipo de utilizador;
2. ***Nurse***: Contém permissões para aceder e utilizar as funcionalidade da *web app* e aplicação móvel, e pode registar outros utilizadores do mesmo tipo e utilizadores do tipo *Patient*;
3. ***Patient***: Contém permissões para aceder e utilizar as funcionalidades da aplicação móvel.

O registo de um utilizador contém condições e regras para a introdução de dados, explicitas em cada área de introdução de texto, e só se faz o registo se estas forem cumpridas.

 Talk.id 

### Register user

Username:  
  
Required. 3 to 16 characters long. Letters and digits only.

Email:  
  
Required. Inform a valid email address.

First name:  
  
Required. 30 characters or fewer.

Last name:  
  
Required. 30 characters or fewer.

Password:  
  

- Your password must contain at least 8 characters.
- Your password can't be too similar to the other fields.
- Your password can't be a commonly used password.
- Your password needs to contain numbers and characters.
- Your password needs to contain at least 1 digit.
- Your password needs to contain at least 1 letter.
- Your password needs to contain at least 1 symbol.

Password confirmation:  
  
Enter the same password as before, for verification.

Figura 5.9: Página de Registo

A *web app* proporciona aos utilizadores uma página *Report*, onde os profissionais de saúde podem aceder a relatórios detalhados sobre solução para uma mensagem enviada por um paciente. Através de uma pesquisa por filtros semelhante à da página principal, a página vai buscar à base de dados *PostgreSQL* aquilo que o utilizador procura. Esta funcionalidade é essencial para documentar as ações tomadas e assegurar que todas as interações e intervenções estão devidamente registadas.

The screenshot displays the 'User Report' interface. At the top, there's a green navigation bar with the 'ESELO' logo and 'TalkID' text. On the right, a user profile 'admin' is shown. The main content area features a 'User Report' form. This form includes a 'Select User' dropdown set to 'paciente', and two date pickers for 'Begin Datetime' and 'End Datetime'. Below these are 'Get' and 'Clear' buttons. The report results are organized into three sections, each with a teal header: 'Type: Tenho' (1 de Junho de 2024 às 00:00), 'Type: SOS' (1 de Junho de 2024 às 00:00), and 'Type: SOS' (1 de Junho de 2024 às 00:00). The 'Type: Tenho' section lists 'Tenho Fome Flutter Solutions' and contains two entries: 'dei-lhe água' and 'dei-lhe um pacote de bolachas', both attributed to 'Super Admin (admin)'. The 'Type: SOS' section lists 'SOS Flutter' and contains one entry: 'dei-lhe um comprimido', also attributed to 'Super Admin (admin)'.

Figura 5.10: Página *Report*



Os utilizadores ainda têm a opção de gerir as contas de outros utilizadores, através da página *Manage Accounts*, tendo em conta as suas permissões anteriormente mencionadas.

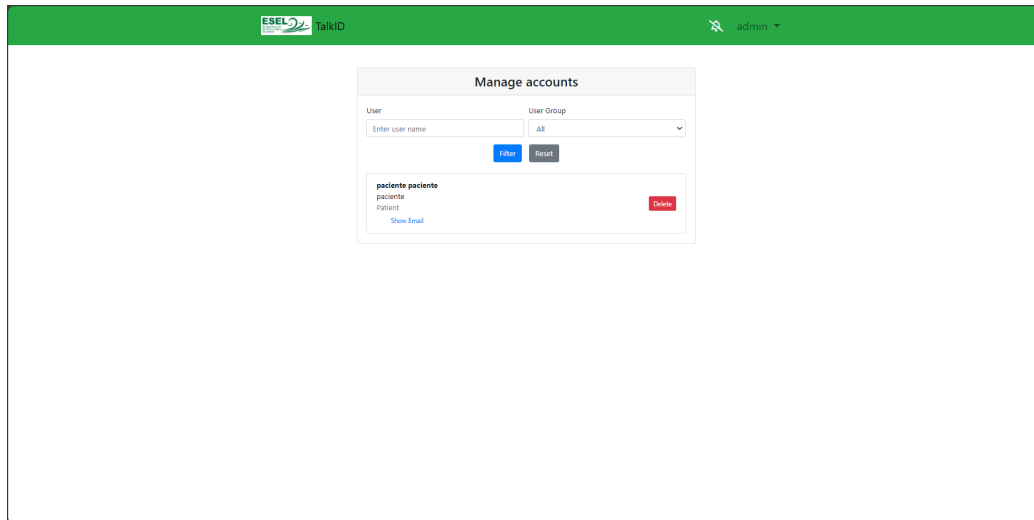


Figura 5.11: Manage Accounts

Se todos os dados introduzidos cumprem as regras de introdução de dados no registo, Figura 5.9, aparece a seguinte mensagem:

[illegible]

Caso as regras de introdução de dados não tivessem sido cumpridas, por exemplo, se a palavra-passe fosse muito curta ou se a confirmação desta estivesse errada aparece a seguinte mensagem:



## 5.3 Comunicação entre Aplicação e Web app

Como mencionado no capítulo 4, secção 4.1, subsecção 4.1.2, a comunicação entre aplicação e web app é efetuada através de pedidos HTTP para a API REST que guarda na base de dados e atualiza as páginas da web app, caso seja necessário. O paciente consegue enviar mensagens para o servidor através da seleção de uma opção de comunicação, mostrando ao utilizador uma mensagem *pop up* a indicar o sucesso no envio da mensagem. A Figura 5.15 representa uma mensagem enviada pelo paciente quando selecionou a opção “*Hunger*”.



Figura 5.15: Mensagem Enviada para o Servidor

Após o envio da mensagem para o servidor através da aplicação, o servidor envia a mensagem para a *web app*, que notifica o profissional de saúde, através de uma *push notification*, que recebeu uma mensagem nova.

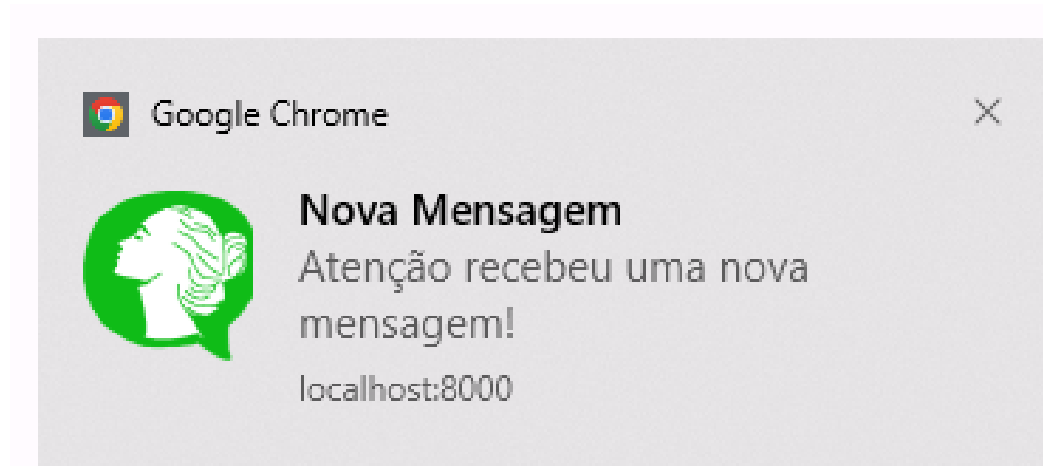


Figura 5.16: Notificação

Na *web app* as mensagens novas aparecem a piscar a vermelho enquanto o utilizador não interagir com elas, e no caso da mensagem ser do tipo S.O.S é reproduzido um alarme.

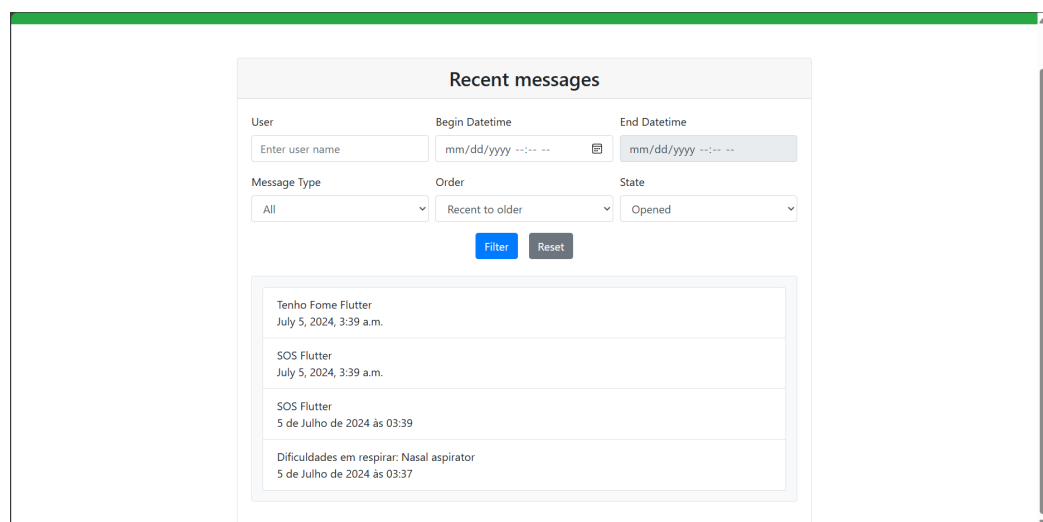


Figura 5.17: Mensagens Recebidas na Web App

O utilizador ao aceder à mensagem consegue visualizar os seus detalhes e, opcionalmente, indicar a solução que irá exercer no cuidado do paciente para a situação descrita, sendo esta guardada na base de dados *PostgreSQL*.

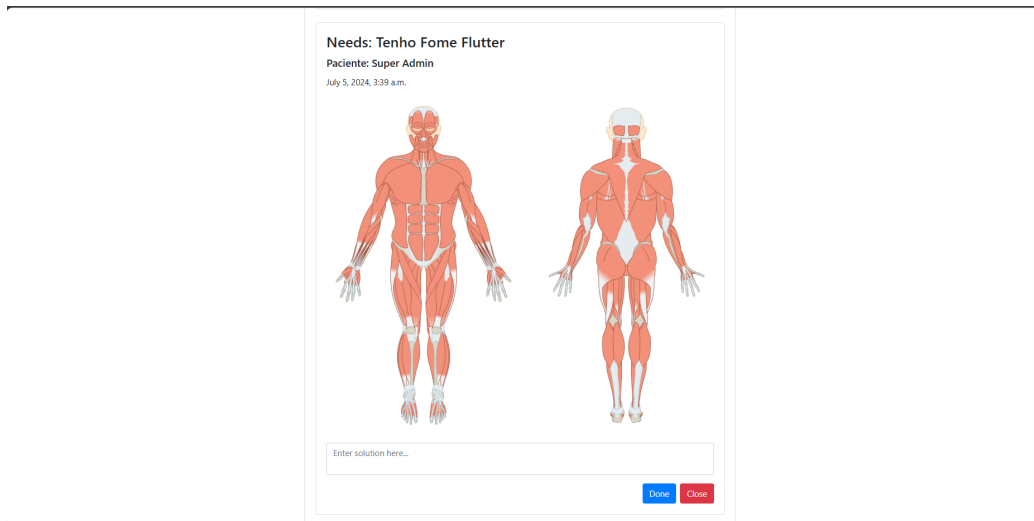


Figura 5.18: Mensagem Detalhada



## Capítulo 6

# Conclusões e Trabalho Futuro

O desenvolvimento do projeto “Talk.id - Communication App” tem como objetivo proporcionar uma significativa contribuição para a melhoria da comunicação entre pacientes submetidos a traqueostomia e profissionais de saúde. Ao longo deste trabalho, abordamos temas cruciais como a criação de interfaces de utilizador intuitivas, a integração de tecnologias modernas como Flutter e Django, e a implementação de sistemas de comunicação. Através deste projeto, é demonstrada a importância de uma solução tecnológica inovadora e eficiente para resolver desafios específicos de comunicação no ambiente hospitalar.

A aplicação móvel e a *web app* desenvolvidas neste projeto foram concebidas com o objetivo de fornecer uma plataforma de comunicação, que não só melhora a experiência dos pacientes, mas também facilita o trabalho dos profissionais de saúde. A arquitetura modular do sistema, a escolha criteriosa das tecnologias utilizadas e a atenção ao design intuitivo garantiram a criação de uma solução passível de uso.

Os principais pontos a serem destacados para futuros leitores e possíveis aplicadores deste trabalho são:

- **Inovação na Comunicação:** Este projeto demonstra como a tecnologia pode ser aplicada de forma inovadora para resolver problemas de comunicação específicos, especialmente em contextos médicos sensíveis. A aplicação de técnicas de *text-to-speech* e a organização de categorias de comunicação são exemplos claros disso.
- **Interoperabilidade e Integração:** A interação eficiente entre a aplicação

móvel e a *web app*, utilizando uma API REST em conjunto com sockets para comunicarem e atualizarem as informações, ilustra a importância de criar sistemas integrados e interoperáveis. Esta arquitetura pode servir de modelo para outros projetos que necessitem de uma comunicação fluida entre diferentes plataformas.

- Design Centrado no Utilizador: A elaboração de *mockups* e a constante adaptação do design para atender às necessidades dos utilizadores finais realçam a importância do design centrado no utilizador. Este enfoque garante que a solução seja intuitiva e fácil de usar, uma consideração essencial em qualquer desenvolvimento de software, especialmente em contextos médicos.
- Escalabilidade e Segurança: A utilização do PostgreSQL como base de dados e a utilização do protocolo *HTTP* para a comunicação entre a aplicação e o servidor sublinham a necessidade de sistemas escaláveis e seguros. Este projeto serve como um exemplo de como abordar estes aspectos desde o início do desenvolvimento.

A experiência adquirida durante o desenvolvimento deste projeto reforça a importância de uma abordagem integrada e centrada no utilizador para resolver problemas complexos de comunicação. Esperamos que as lições e as técnicas apresentadas aqui inspirem futuros projetos e pesquisas, não só no campo da saúde, mas também em outras áreas onde a comunicação eficiente e segura é crucial.

Por fim, este projeto destaca a capacidade da tecnologia de transformar positivamente as experiências humanas e melhorar a qualidade de vida. O “Talk.id - Communication App” é um testemunho do poder da inovação tecnológica quando aplicada com sensibilidade e foco nas necessidades reais dos utilizadores. Futuros leitores e desenvolvedores podem encontrar neste trabalho um ponto de partida sólido e inspirador para suas próprias iniciativas, contribuindo para um mundo onde a comunicação eficaz e acessível é uma realidade para todos.



# Bibliografia

- [1] Stephen J. Bigelow e Alexander S. Gillis. *What is a RESTful API?* 2024. URL: <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API#:~:text=A%20RESTful%20API%20is%20an,deleting%20operations%20related%20to%20resources..>
- [2] Google. *Dart*. 2024. URL: <https://dart.dev/>.
- [3] Google. *Flutter*. 2024. URL: <https://docs.flutter.dev>.
- [4] Manuel Reis. *O que é a traqueostomia?* 2021. URL: <https://www.tuasaude.com/como-cuidar-de-uma-pessoa-com-traqueostomia/>.
- [5] Elijah Samson. *WebSockets-Based APIs with Django (Real-Time communication made easy)*. 2023. URL: <https://medium.com/django-unleashed/websockets-based-apis-with-django-real-time-communication-made-easy-2122b49720bf>.
- [6] Adrian Holovaty e Simon Willison. *Django*. 2024. URL: <https://www.djangoproject.com>.
- [7] Michael Stonebraker. *PostgreSQL*. 2024. URL: <https://www.postgresql.org>.
- [8] Alexander Westgarth. *The Importance Of Having The Right Logo*. 2018. URL: <https://www.forbes.com/sites/theyec/2018/11/30/the-importance-of-having-the-right-logo/>.