DEPARTAMENTO DE
INFORMÁTICA

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

BEATRIZ MARGARIDA RODRIGUES DOS SANTOS

Licenciatura em Engenharia Informática

PARTILHA DE DADOS DURANTE VIDEOCONFERÊNCIA EM APP MÓVEL

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Setembro, 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

PARTILHA DE DADOS DURANTE VIDEOCONFERÊNCIA EM APP MÓVEL

BEATRIZ MARGARIDA RODRIGUES DOS SANTOS

Licenciatura em Engenharia Informática

Orientador: Engenheiro Roberto Silva
Gestor de Projeto, Opensoft - Soluções Informáticas, A.A.

Coorientadora: Fernanda Barbosa
Professora Auxiliar, Universidade NOVA de Lisboa

Júri

Presidente: Doutora Teresa Isabel Lopes Romão
Professora Associada, Universidade NOVA de Lisboa

Vogais: Doutor Vítor Manuel Alves Duarte
Professor Auxiliar, Universidade NOVA de Lisboa

Engenheiro Roberto Silva
Gestor de Projeto, Opensoft - Soluções Informáticas, A.A.

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Setembro, 2024

Partilha de dados durante videoconferência em App móvel

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v7.1.10) [1].

Resumo

A evolução tecnológica e a crescente procura por eficiência numa sociedade digital têm pressionado os serviços públicos burocráticos e tradicionais, como a emissão de documentos de identificação ou a assinatura de documentos oficiais, a adaptarem-se, garantindo maior acessibilidade, comodidade e rapidez. Estes serviços, que requerem autenticação robusta, enfrentam desafios significativos devido à necessidade de presença física do cidadão, o que implica deslocações longas e obrigatórias, além de processos frequentemente complexos e demorados.

Este projeto de tese insere-se no âmbito do **idfyme**, um produto da Opensoft que visa ultrapassar essas barreiras físicas através da implementação de uma solução digital que permite a realização remota destes serviços, garantindo uma autenticação forte e segura, e otimizando tanto a experiência do utilizador como a eficiência dos processos. Inicialmente, o **idfyme** foi lançado em versão *web* e para dispositivos móveis Android, oferecendo funcionalidades como videoconferência, assinatura masnuscrita de documentos digitais e extração e validação biométrica que incluem fotografias faciais, impressões digitais e assinaturas manuscritas digitais.

Apesar do sucesso do lançamento para Android, a falta de suporte para dispositivos iOS limitava significativamente o alcance do produto. Para superar esta limitação, o objetivo desta tese foi expandir o **idfyme**, garantindo suporte também para iOS. Após uma análise aprofundada, decidiu-se desenvolver uma nova plataforma móvel, mas agora multiplataforma, recorrendo à *framework* Flutter. Esta escolha permitiu integrar as versões Android e iOS num único projeto com código partilhado, simplificando os processos de manutenção e atualização, além de assegurar uma experiência de utilizador mais consistente e otimizada em ambos os sistemas operativos.

Palavras-chave: Serviços Remotos, Dados Biométricos, Videoconferência, Assinatura de Documentos, Aplicação Móvel, Multiplataforma, Flutter

Abstract

Technological evolution and the growing demand for efficiency in a digital society have pressured traditional bureaucratic public services, such as issuing identification documents or signing official documents, to adapt by ensuring greater accessibility, convenience, and speed. These services, which require robust authentication, face significant challenges due to the need for the citizen to be physically present, which implies long and obligatory journeys, as well as often complex and time-consuming processes.

This thesis project is part of **idfyme**, an Opensoft product that aims to overcome these physical barriers by implementing a digital solution that allows these services to be carried out remotely and securely, guaranteeing strong authentication and optimizing both the user experience and process efficiency. Initially, **idfyme** was launched in a web version and for Android mobile devices, offering features such as video conferencing, manual signing of digital documents and biometric extraction and validation including facial photographs, fingerprints and digital handwritten signatures.

Despite the success of the Android release, the lack of support for iOS devices significantly limited the product's reach. To overcome this limitation, the aim of this thesis was to expand **idfyme** by guaranteeing support for iOS as well. After an in-depth analysis, it was decided to develop a new mobile platform, but now cross-platform, using the Flutter framework. This choice made it possible to integrate the Android and iOS versions into a single project with shared code, simplifying maintenance and update processes, as well as ensuring a more consistent and optimized user experience on both operating systems.

Keywords: Remote Services, Biometric Data, Video Conferencing, Signing Documents, Mobile Application, Cross-platform, Flutter

Índice

Índice de Figuras

Índice de Tabelas

Siglas

AOT Ahead-of-Time *(pp. 11, 17)*
API Application Programming Interface *(pp. 8, 12, 14, 15, 18, 20, 22, 24, 26, 31, 41, 48, 49, 54, 57, 64, 82, 111)*
ASCII American Standard Code for Information Interchange *(p. 77)*

DAO Data Access Objects *(pp. 24, 25, 33)*
DOM Document Object Model *(p. 12)*

GDPR Regulamento Geral sobre a Proteção de Dados *(p. 49)*
GIF Graphics Interchange Format *(p. 74)*

HTML HyperText Markup Language *(pp. 12, 52, 108)*
HTTP Hypertext Transfer Protocol *(pp. 19, 45, 49–52, 98, 101, 110)*

ICE Interactive Connectivity Establishment *(p. 68)*
ID Identifier *(pp. 50, 52, 65–67)*
IMI Imposto Municipal sobre Imóveis *(p. 60)*
iOS iPhone Operating System *(pp. 2–4, 7–12, 14–17, 22–24, 26, 27, 32, 34, 35, 38–40, 51, 57, 58, 82, 107, 109, 110, 112, 113)*
IP Internet Protocol *(pp. 18, 19, 68)*

JPEG Joint Photographic Experts Group *(pp. 22, 74, 86, 87)*
JSON JavaScript Object Notation *(pp. 49, 52, 54, 101)*
JSX JavaScript XML *(p. 12)*
JVM Java Virtual Machine *(p. 14)*

KMM Kotlin Multiplatform Mobile *(pp. 14–16, 26)*

ML Kit Machine Learning Kit *(pp. 20, 21, 26, 32, 82, 111)*

MVVM Model-View-ViewModel *(pp. 40, 41, 98, 101, 110)*

NAT Network Address Translation *(pp. 18, 19, 26)*

P2P Peer-to-peer *(pp. 17, 18, 64, 67–69, 81, 111)*
PDF Portable Document Format *(pp. 21–23, 32, 45, 90, 91, 95, 96, 99, 105, 112)*
PIN Personal Identification Number *(pp. 2, 26, 35, 36, 39, 50, 56, 57, 87, 97, 99, 110, 112, 113)*
PNG Portable Network Graphics *(pp. 22, 74, 76, 86, 87, 96)*

RAM Random Access Memory *(p. 24)*

SDK Software Development Kit *(pp. 10, 15, 17, 22, 26, 27)*
SQL Structured Query Language *(pp. 25, 27, 33)*
SRTP Secure RTP *(p. 17)*
STUN Session Traversal Utilities for NAT *(pp. 18, 19, 26, 32, 63, 64, 67, 99, 111)*

TCP Transmission Control Protocol *(p. 18)*
TURN Traversal Using Relays around NAT *(pp. 18, 19, 26, 32, 63, 64, 67, 99, 111)*

UI User Interface *(pp. 5, 9, 10, 12, 15, 37, 38, 40–43, 104, 110)*
URI Uniform Resource Identifier *(p. 49)*

WebRTC Web Real-Time Communication *(pp. 17–20, 26, 32, 63, 67, 99, 111)*
WiFi Wireless Fidelity *(p. 18)*

XML eXtensible Markup Language *(pp. 12, 52)*

1

Introdução

Este capítulo tem como objetivo fornecer uma introdução ao tema desta tese e aos conceitos básicos necessários para a sua compreensão. Serão abordados a motivação e o contexto que impulsionaram a realização desta pesquisa, o seu enquadramento no projeto em que está inserida, o problema que motivou o seu desenvolvimento, os objetivos que pretende alcançar e as contribuições que proporciona. Por fim, será apresentada uma breve descrição da organização do documento.

1.1 Motivação e Contexto

As últimas décadas têm sido marcadas pela rápida evolução tecnológica e pelas mudanças no comportamento social. Isto tem impulsionado serviços que requerem uma autenticação robusta do cliente, como os serviços públicos legais historicamente moldados por procedimentos tradicionais e burocráticos, a enfrentarem desafios significativos para acompanhar e adaptar as demandas crescentes dos cidadãos.

A forma como as pessoas vivem, trabalham e interagem com as instituições mudou drasticamente. Os clientes estão cada vez mais ansiosos para que estes serviços sejam fáceis de usar, eficientes, cómodos e adaptados à dinâmica da vida moderna. O acesso eficiente a procedimentos complexos, como a emissão de documentos oficiais, serviços notariais, ou a assinatura de documentação legal tornou-se uma necessidade premente.

A burocracia tradicional e os métodos convencionais de prestação destes serviços constituem obstáculos que transformam a obtenção dos mesmos num desafio para o cidadão. Estes desafios são ainda acentuados pela necessidade de uma forte autenticação do cidadão, pela presença de barreiras geográficas e pela sua complexidade burocrática. A sua realização muitas vezes exige deslocações consideráveis, agendas apertadas, exigências físicas imperativas e tempo de espera em filas intermináveis [2].

A busca incessante de eficiência, acessibilidade e adequação às expectativas crescentes de uma população cada vez mais conectada digitalmente motiva a adaptação para uma forma remota e online destes serviços proporcionando uma oportunidade de ampliar significativamente a sua acessibilidade e rapidez. Ao eliminar barreiras geográficas e

reduzir a dependência de procedimentos presenciais, a população pode ter acesso a serviços essenciais de maneira mais competente e conveniente, otimizando a eficiência operacional dos órgãos que os prestam.

Não apenas se tornou imperativa a adaptação para um formato remoto e online, mas também móvel. O telemóvel, constantemente presente em todos os momentos da vida quotidiana, emergiu como o meio mais conveniente, prático, útil e eficiente para dar resposta às demandas exigentes destes serviços [3].

A transição para serviços móveis, ajustados a ambos os sistemas operativos que lideram o mercado, o Android e o iPhone Operating System (iOS) [4], impulsiona a inclusão digital, proporcionando uma alternativa mais acessível para diversos cidadãos que podem não ter acesso a um computador ou enfrentam desafios na sua gestão. Isso deve-se ao facto de o uso de uma aplicação móvel ser consideravelmente mais simples e direto do que aceder a um *website*, além de proporcionar facilidades extras [5], como a presença de câmara, microfone e ecrã tátil incorporados. Este benefício estende-se também aos cidadãos que possuam exclusivamente um dos sistemas operativos (Android ou iOS) garantindo assim a sua integração na solução proposta e permitindo que desfrutem da mesma praticidade e funcionalidade.

A segurança e autenticação também são aspetos que podem ser aprimorados com a implementação de serviços móveis. Mecanismos de segurança avançados, como autenticação de dois fatores (autenticação que exige duas etapas distintas, como uma senha/Personal Identification Number (PIN) e um código ou *token* gerado para o dispositivo móvel) e tecnologias biométricas disponíveis nos dispositivos móveis, podem oferecer uma camada adicional de proteção para transações legais e informações sensíveis [6].

1.2 Enquadramento

Esta tese está integrada no projeto **idfyme** desenvolvido pela Opensoft, cujo objetivo é ultrapassar as referidas limitações físicas através da implementação eficiente destes serviços remotamente.

O **idfyme**[1] é uma solução de *software* para atendimento através de canais digitais, totalmente personalizável conforme as necessidades de cada organização. A plataforma pode ser utilizada para facilitar a comunicação direta entre funcionários e clientes através de videoconferências, tanto para o esclarecimento de dúvidas como para a gestão de situações mais complexas, como a extração de dados para a emissão de documentos de identificação. Estas situações envolvem o envio e recolha de informações sensíveis, incluindo dados biométricos, como assinaturas manuais e fotografias de rosto e de documentos oficiais. O **idfyme** disponibiliza ainda funcionalidades avançadas de extração biométrica (como reconhecimento facial, assinaturas manuais e impressões digitais) e de validação e assinatura de documentos oficiais.

[1] https://idfyme.com/

A plataforma garante assim o mesmo nível de segurança e fiabilidade do atendimento presencial, enquanto melhora a experiência do utilizador e otimiza os recursos do prestador de serviços, promovendo uma relação de proximidade e mantendo o contacto personalizado com o cliente.

No entanto, o produto possui apenas uma versão *web* e móvel Android, não havendo nenhuma versão para o sistema operativo iOS.

1.3 Problema

O problema reside no facto de o produto **idfyme** não estar disponível para o sistema operativo iOS, o que limita a oferta dos seus serviços a um segmento significativo da população que utiliza exclusivamente dispositivos iOS. Considerando que o mercado de sistemas operativos móveis é amplamente dominado por Android, com uma quota de 71.6%, e iOS, com 27.5%, é evidente que a ausência de uma versão compatível com ambos afeta significativamente o alcance da aplicação [4].

Embora exista uma versão *web* do produto acessível a ambos os tipos de utilizadores, esta apresenta limitações funcionais e técnicas. Muitas das funcionalidades essenciais do **idfyme**, como a extração biométrica e a assinatura de documentos, não estão incluídas na versão *web*, uma vez que os computadores frequentemente carecem de acessibilidade fácil a componentes críticos, como câmara, microfone e ecrã tátil. Além disso, a versão *web* não oferece o mesmo nível de segurança que uma aplicação móvel, particularmente no que respeita à autenticação biométrica, que é crucial para garantir a integridade e a segurança dos dados no produto.

Desta forma, é imperativo que o **idfyme** desenvolva uma solução móvel robusta, compatível tanto com iOS como com Android, garantindo não apenas a inclusão de uma ampla faixa de utilizadores, mas também a implementação eficiente e segura das funcionalidades essenciais. Ao alargar a sua presença para os dois sistemas dominantes, o **idfyme** pode aumentar significativamente a sua base de utilizadores, expandindo o seu alcance e popularidade no mercado.

Um desafio adicional surge na eficiência técnica da solução, que exige uma implementação simultaneamente eficaz e otimizada para ambos os sistemas operativos Android e iOS, mantendo a coerência e fluidez entre as duas plataformas. Para garantir uma experiência de utilização consistente e fluida, é essencial que a integração das funcionalidades seja eficiente e harmoniosa, assegurando que não existam discrepâncias de desempenho ou diferenças na utilização entre as duas versões. A solução deve ser desenhada de forma a preservar a coesão e a uniformidade entre as implementações, proporcionando uma experiência homogénea e intuitiva, sem comprometer a usabilidade ou o desempenho em qualquer um dos sistemas operativos.

1.4 Objetivos

O objetivo central da tese é a criação de uma versão móvel do produto **idfyme** para o sistema operativo iOS, assegurando a coerência entre esta versão e a versão Android já desenvolvida. Para tal, a versão iOS deverá preservar todas as funcionalidades já implementadas na versão **idfyme** para Android, adaptando-as ao ecossistema iOS de forma eficaz e otimizada.

A solução visa então implementar as diversas funcionalidades que compõem o produto, incluindo uma comunicação online segura e fluida com agendamento e realização de videoconferências. Além disso, a solução deve incluir a validação e extração de dados biométricos, tanto durante as videoconferências como fora delas. Isto abrange a captura de assinaturas digitais manuscritas, fotografias de rosto e documentos oficiais. A solução deve também garantir a validação e extração de assinaturas manuais em documentos, assegurando a sua autenticidade e validade legal. Adicionalmente, tem de garantir a segurança e a privacidade dos dados trocados através da implementação de mecanismos já existentes no produto ou da introdução de novos recursos de segurança.

1.5 Contribuições

A principal contribuição desta tese é o desenvolvimento de uma versão do produto **idfyme** para o sistema operativo iOS, o que expande significativamente o alcance da aplicação para um segmento relevante da população que utiliza exclusivamente dispositivos iOS. Esta inclusão é especialmente importante, já que permite que o **idfyme** atinja mais 27.5% do mercado de sistemas operativos móveis [4], promovendo uma maior inclusão digital e garantindo que uma parcela significativa dos utilizadores possa beneficiar dos serviços oferecidos.

Além disso, a versão para iOS preserva e aprimora as funcionalidades existentes na versão Android, incluindo a comunicação segura em videoconferência e a validação e extração de dados biométricos, como assinaturas digitais manuscritas e fotografias da face. A implementação de mecanismos de segurança avançados garante a proteção dos dados sensíveis e a autenticidade dos documentos assinados digitalmente, alinhando-se aos padrões de segurança e privacidade exigidos.

A solução também contribui para a melhoria da eficiência operacional dos prestadores de serviços ao permitir atendimentos remotos e simplificar processos burocráticos. Esta abordagem reduz significativamente o tempo e os custos associados ao atendimento presencial, resultando numa gestão mais eficiente dos recursos disponíveis e promovendo uma alocação mais eficaz desses recursos.

Com a solução disponível em ambos os sistemas operativos, os cidadãos têm acesso facilitado a serviços essenciais de forma mais conveniente, eliminando a necessidade de deslocamentos físicos e reduzindo a dependência de procedimentos presenciais. Esta

melhoria promove a inclusão e a acessibilidade, beneficiando a população em geral e fortalecendo a presença e a eficácia do **idfyme** no mercado.

1.6 Organização

O documento está organizado de acordo com a seguinte estrutura:

- **Introdução**- Este capítulo estabelece o contexto e a motivação subjacentes ao trabalho desenvolvido, enquadrando-o no produto em que se insere. Além disso, clarifica o problema que motivou a sua criação, delineando os seus objetivos e as suas respetivas contribuições.

- **Descrição e Estado da Arte**- Neste capítulo, realiza-se uma análise tecnológica relevante para o projeto, destacando as ferramentas, bibliotecas e *plugins* externos, *frameworks* e conceitos que fundamentam a implementação proposta. Esta exploração foi conduzida por meio de uma revisão de literatura para identificar os desafios enfrentados e as vantagens encontradas nas tecnologias selecionadas. O objetivo foi construir uma base sólida, assente numa compreensão aprofundada de estudos realizados, para justificar de maneira robusta as escolhas tecnológicas feitas para o desenvolvimento do projeto.

- **idfyme - Transformação Total do Atendimento**- Neste capítulo, aborda-se os aspetos mais relevantes do produto **idfyme**, desenvolvido anteriormente. Procede-se com a análise do mesmo, utilizando-o como contexto fundamental para situar o trabalho a ser realizado.

- **Descrição e Análise da Solução**- Neste capítulo, apresenta-se a solução proposta para resolver o problema estabelecido. São exploradas as versões compatíveis e o fluxo da aplicação através de um diagrama, oferecendo uma compreensão visual e detalhada de como a proposta se desenrola, integrando as funcionalidades exigidas pelo produto **idfyme**. Adicionalmente, realiza-se uma análise do seu *design*, tendo em consideração a sua implementação multiplataforma.

- **Implementação da Solução** - Neste capítulo, detalham-se as implementações realizadas na solução e os respetivos constrangimentos, explicando e ilustrando os fluxos das funcionalidades através de gráficos, bem como as User Interface (UI)s da aplicação móvel, com recurso a figuras. As implementações, assim como as metodologias, bibliotecas e *plugins* utilizados, são discutidas em detalhe, abordando cada um dos projetos envolvidos na solução: *idfyme-flutter* e *idfyme-plugins*.

- **Qualidade Técnica da Solução** - Neste capítulo, apresentam-se todas as técnicas e práticas de desenvolvimento implementadas para assegurar a qualidade e a segurança da solução. São abordadas as técnicas de desenvolvimento, os ambientes

de desenvolvimento utilizados, os testes efetuados, a automatização de tarefas e a gestão eficiente das *builds* realizadas.

- **Conclusão** - Neste capítulo, apresentam-se as conclusões finais do trabalho desenvolvido, resumindo as implementações, decisões e limitações encontradas. Reflete-se também sobre possíveis trabalhos futuros que poderão adicionar valor à solução, assim como as suas contribuições finais.

2

Descrição e Estado da Arte

Este capítulo aborda algumas das metodologias, tecnologias e ferramentas relevantes para o desenvolvimento da aplicação proposta, com foco na integração do produto para o sistema operativo iOS. A seleção das tecnologias a utilizar no desenvolvimento da solução desempenhou um papel crucial na eficiência, desempenho e interoperabilidade da aplicação. Com base numa revisão de trabalhos relacionados, a escolha foi sustentada por estudos e projetos que abordam os desafios e as soluções para as tecnologias em consideração.

Foi adicionalmente conduzida uma análise abrangente para compreender claramente as vantagens e desvantagens de cada tecnologia. Esta avaliação aprofundada teve como objetivo construir um conhecimento sólido, proporcionando orientação para as decisões e estratégias que se realizaram durante o desenvolvimento da aplicação.

2.1 Estudo de Viabilidade: Desenvolvimento Multiplataforma e Nativo

O desenvolvimento da aplicação móvel para iOS pode ser realizado de duas formas distintas: através de um desenvolvimento nativo ou de um desenvolvimento multiplataforma. Cada um destes desenvolvimentos apresentam vantagens e desvantagens, que devem ser cuidadosamente consideradas para garantir a escolha mais adequada ao projeto e produto em consideração.

2.1.1 Desenvolvimento Nativo

O desenvolvimento nativo para iOS envolve a criação de uma aplicação específica para o sistema operativo iOS, utilizando ferramentas e linguagens próprias, como Swift ou Objective-C, e aproveitando o ambiente de desenvolvimento Xcode[1].

Uma das principais vantagens desta abordagem é a performance otimizada que ela oferece. Aplicações nativas geralmente proporcionam um desempenho superior e acesso

[1] https://developer.apple.com/xcode/

direto a Application Programming Interface (API)s e funcionalidades específicas do sistema operativo, resultando numa experiência de utilizador mais fluída e responsiva. Além disso, as aplicações nativas podem integrar-se completamente com o sistema do dispositivo móvel, utilizando todos os recursos e serviços nativos da plataforma, o que melhora a profundidade da integração e a experiência do utilizador. Outro benefício é que a solução pode se concentrar exclusivamente no desenvolvimento no ambiente iOS, sem a necessidade de se ter que desenvolver uma nova versão para Android, uma vez que esta já está desenvolvida e em funcionamento no produto [7].

Contudo, desenvolver e manter uma aplicação nativa para iOS requer recursos adicionais, uma vez que é necessário criar e manter uma versão separada da aplicação para cada plataforma, o que pode aumentar os custos e o tempo de desenvolvimento. Além disso, a duplicação de esforços e código entre as versões para iOS e Android pode levar a inconsistências e desafios adicionais na manutenção. A necessidade de manter as mesmas funcionalidades e a experiência do utilizador em ambas as plataformas pode também exigir um esforço adicional para garantir a consistência entre as versões e uma interface uniforme [7].

2.1.2 Desenvolvimento Multiplataforma

Por outro lado, o desenvolvimento multiplataforma envolve a criação de uma única aplicação compatível com ambos os sistemas operativos Android e iOS utilizando *frameworks* e ferramentas próprias como o Flutter[2] ou o React Native[3].

O principal benefício desta abordagem é a capacidade de partilhar uma parte significativa, ou mesmo a totalidade, do código entre ambas as plataformas. Isto pode reduzir significativamente o tempo e os custos associados ao desenvolvimento e à manutenção. Com uma base de código única, o desenvolvimento tende a ser mais rápido, pois alterações e atualizações precisam de ser feitas apenas uma única vez [7].

Esta abordagem também permite uma maior flexibilidade e escalabilidade para futuras atualizações e expansões, uma vez que as mudanças podem ser implementadas de forma consistente e ao mesmo tempo em ambas as plataformas. Além disso, uma aplicação multiplataforma facilita muito mais a obtenção de uma experiência de utilizador consistente em ambas as plataformas, tornando mais simples a implementação das funcionalidades e a manutenção de uma interface uniforme [7].

No entanto, pode haver uma perda de desempenho em comparação com aplicações nativas, especialmente para funcionalidades que exigem acesso intenso aos recursos do dispositivo. As limitações na integração com APIs e funcionalidades específicas de cada sistema operativo podem exigir trabalho adicional para superar as diferenças entre as plataformas. Além disso, problemas específicos de uma das plataformas podem ser

[2] https://flutter.dev/
[3] https://reactnative.dev/

mais difíceis de depurar e resolver, uma vez que o código é executado num ambiente intermédio [7].

Outra dificuldade é a diferença no fluxo de utilização entre aplicações móveis no Android e no iOS, o que pode complicar a criação de um *design* e UI único que seja igualmente intuitiva e familiar para os utilizadores de ambas as plataformas [7].

Adicionalmente, nesta abordagem é necessário desenvolver uma nova versão para Android levando à descontinuação da versão atual. Isto implica considerar o funcionamento e a integração da aplicação tanto no ambiente Android como no iOS, o que aumenta a complexidade e o esforço necessário para a implementação da solução. Outra dificuldade surge quando é necessário integrar funcionalidades nativamente na aplicação multiplataforma, o que adiciona código nativo ao código partilhado, aumentando assim a complexidade e a dificuldade do desenvolvimento [7].

2.1.3 Análise Comparativa

Após uma análise detalhada das vantagens e desvantagens do desenvolvimento nativo para iOS e multiplataforma para Android e iOS, foi elaborada a tabela 2.1, com o objetivo de apresentar uma comparação clara e simplificada entre as características das duas abordagens, facilitando assim a escolha da solução mais adequada para o projeto.

Características	**Nativa**	**Multiplataforma**
Desempenho da Aplicação	+	-
Reutilização de código	- -	++
Melhor Tempo e Custos de Desenvolvimento	- -	++
Flexibilidade e Manutenção Futura	- -	++
Experiência do Utilizador	+	-
Consistência da Aplicação e *Design*	-	+

Tabela 2.1: Análise comparativa do desenvolvimento nativo e multiplataforma para a solução

Após uma análise das vantagens e desvantagens mostradas na tabela, decidiu-se adotar um desenvolvimento multiplataforma para a solução. Esta escolha foi fundamentada na capacidade desta abordagem conseguir atender às características mais relevantes para o projeto. Optar por este desenvolvimento reduz significativamente o tempo e os custos associados ao desenvolvimento e à manutenção, ao mesmo tempo que proporciona maior flexibilidade, manutenção e escalabilidade para futuras atualizações e expansões. Além disso, o desenvolvimento multiplataforma garante uma maior consistência entre as versões para Android e iOS, facilitando a criação de uma interface uniforme e oferecendo uma experiência de utilizador mais coesa e consistente.

Desta forma, o desenvolvimento multiplataforma permite oferecer um desenvolvimento significativamente mais ágil e uma experiência de utilizador mais coesa em comparação com a alternativa considerada do desenvolvimento nativo.

Apesar dos desafios, como a possível perda de desempenho e a complexidade adicional associada à experiência do utilizador e ao desenvolvimento de uma nova versão para Android, os benefícios do desenvolvimento multiplataforma superam estes obstáculos. A perda de desempenho não é significativa o suficiente para justificar a escolha de uma abordagem nativa [7]. Além disso, a complexidade adicional pode ser gerida de forma eficaz com uma abordagem cuidadosa e organizada no desenvolvimento da solução, considerando o tempo disponível, juntamente com uma boa arquitetura de código e um *design* bem planeado.

Assim, o desenvolvimento multiplataforma foi considerada a mais adequada para o desenvolvimento da aplicação proposta, oferecendo uma solução eficaz, simples e eficiente, justificando a decisão com base no equilíbrio entre as vantagens proporcionadas e as dificuldades a serem geridas.

2.2 *Framework* de Desenvolvimento Multiplataforma

Esta secção procurou explorar três *frameworks* de desenvolvimento multiplataforma que se considerou na implementação da solução, oferecendo uma análise abrangente das suas vantagens e desvantagens. Com esta abordagem, foi possível alcançar uma economia significativa de tempo e recursos, ao escrever uma base de código única que podia ser utilizada tanto em dispositivos iOS quanto Android. Esta abordagem unificada não apenas reduziu os custos de desenvolvimento, mas também facilitou a manutenção e as atualizações, proporcionando uma solução mais reduzida e consistente [8].

Esta secção explora então estes aspetos, destacando as nuances e peculiaridades de cada *framework* para permitir uma tomada de decisão informada e correta.

2.2.1 Flutter

O Flutter[4] é uma *framework* de desenvolvimento de interface do utilizador, UI, de código aberto (*open-source*), criada pela Google. Ela é projetada para o desenvolvimento rápido de aplicações nativas para várias plataformas (*cross-platform*) como iOS, Android, *web* e *desktop* a partir de um único código-fonte. Possui ainda um Software Development Kit (SDK) e uma biblioteca de UI baseada em *widgets*, elementos visuais e funcionais, como botões, campos de texto e *layouts*. A linguagem de programação integrada do Flutter é o Dart, uma linguagem de programação reativa (programação assíncrona e automática de mudanças), orientada a objetos, com uma sintaxe semelhante ao JavaScript mas mais eficiente [9].

[4] https://flutter.dev/

Estas características do Flutter permitem ter várias vantagens como:

- ***widgets*:** Contém uma extensa coleção de *widgets* pré-criados que permite que os programadores criem facilmente interfaces de utilizador apelativas e responsivas. Além disso, é bastante fácil criar novos componentes personalizáveis, a partir da estrutura por camadas dos *widgets*, originando implementações inovadoras [10].

- **Temas integrados:** Contém os temas integrados *Material* e *Cupertino* que são conjuntos de estilos visuais e comportamentais que seguem as diretrizes de *design* do Material Design (Google) e do iOS (Apple), respectivamente, oferecendo uma experiência de utilizador consistente com a estética típica do Android e do iOS. Desta forma, pode-se implementar configurações dinâmicas para alternar entre temas Android e iOS, independentemente do sistema operativo da plataforma real em que a aplicação está a ser executada [11].

- **Recarregamento Rápido (*Hot Reloading*):** Permite que os programadores vejam imediatamente as mudanças no código refletidas na aplicação, simuladores e emuladores, sem a necessidade de executar novamente a aplicação. Isso agiliza o ciclo de desenvolvimento e facilita a experimentação [11].

- **Alto Desempenho:** As aplicações Flutter são compiladas seguindo uma abordagem de compilação Ahead-of-Time (AOT) para gerar código nativo, proporcionando desempenho próximo ao de aplicações desenvolvidas nativamente. O processo de compilação do Flutter envolve a transformação do código-fonte Dart em código nativo específico da plataforma de destino eliminando a necessidade de escrever uma camada de ponte entre o código Dart e o código nativo [11].

- **Renderização própria:** Possui o seu próprio mecanismo de renderização baseado na biblioteca gráfica Skia. Esse mecanismo possibilita a criação de uma representação visual da interface do utilizador de maneira independente, sem depender de componentes nativos do sistema operativo, de *browsers* da *web* ou de *web views* (componentes que permitem incorporar conteúdo da *web* diretamente em aplicações). Isto resulta numa renderização rápida e eficiente garantindo que as aplicações criadas com o Flutter tenham excelente desempenho, mesmo em dispositivos com recursos limitados ou *hardware* mais antigo [10].

No entanto o Flutter também possui algumas desvantagens como:

- **Integração com Recursos Nativos Complexos:** Integração com recursos nativos complexos ou específicos de uma plataforma pode ser mais difícil no Flutter em comparação com o desenvolvimento nativo ou de outras *framework* que oferecem uma camada de acesso mais direto aos recursos nativos.

- **Curva de aprendizagem:** A curva de aprendizagem do Flutter pode ser acentuada para programadores que não estão familiarizados com Dart ou conceitos de programação reativa.

2.2.2 React Native

React Native[5] é uma *framework* popular de desenvolvimento móvel de código aberto que permite aos programadores criar aplicações nativas para iOS e Android utilizando o mesmo código base.

Utiliza JavaScript [12], uma linguagem de programação amplamente utilizada para o desenvolvimento *web*, como linguagem de programação principal e a biblioteca React para construir aplicações móveis nativas para iOS e Android. O React [13] é uma biblioteca JavaScript que simplifica a construção de interfaces de utilizador dinâmicas e reativas utilizando uma arquitetura baseada em componentes. Estes são blocos de construção reutilizáveis para elementos de UI.

A linguagem utiliza JavaScript XML (JSX), uma extensão de sintaxe para JavaScript, que permite descrever a aparência da interface de utilizador de forma declarativa tornando o código mais legível e intuitivo. Ao contrário do JavaScript puro, onde a criação de elementos de interface envolve a manipulação direta do Document Object Model (DOM), o JSX fornece uma maneira mais expressiva e legível de definir a hierarquia de componentes. Com o JSX, os programadores podem escrever código que se assemelha ao HyperText Markup Language (HTML) ou ao eXtensible Markup Language (XML), mas que é traduzido para chamadas de funções JavaScript durante o processo de *transpilation* (fase de compilação onde o código numa linguagem de programação é traduzido para código noutra linguagem de nível semelhante) [14].

O React Native apresenta as seguintes vantagens:

- **Renderização nativa:** Capacidade de converter os componentes React Native em elementos visuais nativos específicos da plataforma, como *Views* para Android ou *Views* de UI para iOS. Isto é possível graças à camada de abstração conhecida como "ponte"(*bridge*), que atua como uma interface entre o código JavaScript executado pelo React Native e as API nativas dos sistemas operativos Android e iOS [15].
- **Interface do utilizador (UI) Simples:** Cria a interface do utilizador usando React JavaScript baseado em componentes, resultando num programa mais responsivo e rápido. O React Native utiliza ainda componentes nativos para uma aparência e experiência autênticas, representando elementos de interface nativos em JavaScript, contribuindo para um desempenho eficiente e para uma experiência do utilizador aprimorada [15].

[5] https://reactnative.dev/

- **Atualização Rápida (*Fast Refresh*):** Permite que os programadores visualizem e apliquem alterações do código em tempo real sem perder o estado da aplicação e ter de reiniciar completamente a aplicação. O objetivo é tornar o processo de desenvolvimento mais suave, atualizando apenas os componentes que foram alterados.

- **Recarregamento Rápido (*Hot Reloading*):** Os programadores podem ver o impacto imediato das alterações de código durante o desenvolvimento sem recompilar a aplicação inteira. Esta funcionalidade acelera o processo de desenvolvimento e produtividade, fornecendo *feedback* imediato sobre as alterações do código [15].

- **Velocidade e Desempenho:** A diferença de velocidade entre o código nativo e o JavaScript é quase invisível ao olho humano utilizando componentes nativos [15], resultando num desempenho próximo ao de uma aplicação nativa, proporcionando uma experiência de utilizador suave e responsiva.

- **Comunidade e Ecossistema Amplos:** Por ser um dos líder de mercado, possui uma comunidade grande e ativa, contribuindo para um vasto ecossistema de bibliotecas, módulos e ferramentas que podem ser aproveitados para diversas funcionalidades.

- **Expo:** Conjunto de ferramentas e serviços para construir aplicações React Native, simplificando o processo de desenvolvimento ao fornecer um ambiente de desenvolvimento e de serviços.

O React Native apresenta as seguintes desvantagens:

- **Difícil Modificar Dados:** Modificar dados numa aplicação React pode ser complexo devido à natureza unidirecional do fluxo de dados. A manipulação dos dados depende da gestão adequada do estado, uma vez que React utiliza um modelo de fluxo de dados *top-down*, onde o estado é normalmente gerido no nível superior da árvore de componentes e passado para os componentes filhos através de *props*. Esta abordagem garante previsibilidade, mas pode tornar-se desafiadora quando lidamos com estados complexos ou quando múltiplos componentes necessitam de partilhar e modificar o mesmo conjunto de dados. [15]

- **Mau Desempenho em Interações Complexas:** A performance do React Native revela-se insatisfatória em contextos que requerem um *design* da interface do utilizador mais elaborado e complexo com animações e interações intensivas. Esta limitação é atribuída ao conceito da "ponte", onde todos os módulos nativos necessitam de ser conectados à componente JavaScript da aplicação. Em casos de interações excessivas, o programa pode deteriorar-se, tornando-se inutilizável e apresentando lentidão como resultado [15].

- **Dependência do Ecossistema:** O ecossistema do React Native está intrinsecamente ligado a bibliotecas de terceiros, cuja qualidade e manutenção podem apresentar variações, gerando desafios ao contar com soluções externas.

2.2.3 Kotlin Multiplatform Mobile (KMM)

Kotlin Multiplatform Mobile (KMM)[6] é uma tecnologia desenvolvida pela JetBrains que permite aos programadores partilhar código entre plataformas móveis usando a linguagem de programação Kotlin. Esta linguagem foi desenhada para ser concisa, expressiva e compatível com o Java, completamente suportada pela Java Virtual Machine (JVM) [16].

O KMM não compromete a experiência do programador, pois permite a integração fácil com as API nativas de cada plataforma através do uso de três módulos principais [16]:

- **Módulo Compartilhado:** Núcleo do KMM onde se escreve o código que posteriormente será partilhado entre as plataformas. Este módulo geralmente contém a lógica de negócios, algoritmos e manipulação de dados que são comuns a ambas as plataforma sendo escrito uma vez e podendo ser utilizado tanto no Android quanto no iOS.

- **Módulo Android:** Contém o código específico para a plataforma Android permitindo uma integração suave com o seu ecossistema. Ele pode incluir componentes de interface do utilizador específicos do Android, interações com API do Android e outras características que são exclusivas dessa plataforma.

- **Módulo iOS:** Similar ao módulo Android, este módulo contém o código específico para a plataforma iOS, facilitando a interação com as API nativas do iOS. Ele abrange a implementação de interfaces de utilizador específicas do iOS, interações com APIs do iOS e outras funcionalidades que são exclusivas para o desenvolvimento iOS.

O Kotlin Multiplatform Mobile apresenta, portanto, as seguintes vantagens:

- **Produtividade Aprimorada:** Ao utilizar o KMM, os programadores podem aproveitar a produtividade oferecida pela linguagem Kotlin e pela interoperabilidade com o Java resultando num desenvolvimento mais rápido e eficiente, já que muitas tarefas podem ser realizadas de maneira mais concisa. Esta característica manifesta-se não apenas na criação de novos projetos, mas também na facilidade de migrar um projeto anteriormente desenvolvido nativamente para Android em Java para uma abordagem unificada multiplataforma com a tecnologia KMM [17].

[6] https://www.jetbrains.com/kotlin-multiplatform/

- **Integração Nativa:** O KMM permite a integração nativa com as APIs específicas de cada plataforma (Android e iOS). Isto significa que os programadores podem aceder a recursos nativos e utilizar bibliotecas específicas de cada plataforma, mantendo ao mesmo tempo uma base de código comum oferecendo então uma flexibilidade para lidar com casos em que componentes específicos para cada plataforma são necessários [16].

- **Compatibilidade com Ferramentas de Desenvolvimento Padrão:** O KMM é compatível com as ferramentas de desenvolvimento padrão para Android e iOS, como o Android Studio[7] e o Xcode[8] facilitando a integração em fluxos de trabalho existentes, não exigindo a adoção de novas ferramentas complexas.

- **Recurso de Carregamento Preguiçoso (*Lazy-loading*):** O carregamento preguiçoso é um padrão de design e recurso frequentemente utilizado no desenvolvimento de *software*, especialmente no contexto do carregamento de recursos ou dados a pedido. No carregamento preguiçoso, os recursos são carregados ou inicializados apenas quando são realmente necessários, em vez de antecipadamente, melhorando o desempenho e reduzindo o uso de memória [16].

- **Utilização dos *framework* de UI Mais Recentes:** Os programadores que trabalham com o Kotlin Multiplatform Mobile têm a flexibilidade de aproveitar as *framework* de UI mais recentes fornecidas por cada plataforma. Por exemplo, no iOS, o SwiftUI pode ser utilizado para construir interfaces de utilizador modernas e declarativas, e o Jetpack Compose pode ser utilizado no Android. Isto permite que os programadores permaneçam atualizados com as últimas ferramentas e tirem vantagem dos avanços específicos de cada plataforma [18].

- **Acesso aos SDKs do Android e iOS:** O KMM não impõe restrições de acesso aos SDK específicos de cada plataforma, tendo acesso fácil e direto aos SDK do Android e iOS. Isto permite a integração de funcionalidades nativas, significando que qualquer funcionalidade ou API fornecida pelo Android ou iOS pode ser utilizada diretamente no código Kotlin compartilhado [18].

No entanto, o Kotlin Multiplatform Mobile apresenta também as seguintes desvantagens:

- ***Framework* recente:** Como o KMM é uma *framework* relativamente nova (a versão estável foi lançada em 2021), pode ter suporte limitado de bibliotecas e ferramentas, como *plugins* e integrações de terceiros em comparação com *framework* mais estabelecidas. Isto pode levar os programadores a ter que construir determinadas funcionalidades do zero, a encontrar soluções alternativas para requisitos específicos

[7] https://developer.android.com/studio?hl=pt-br
[8] https://developer.apple.com/xcode/

ou a confiar em pacotes menos estabelecidos, aumentando potencialmente o tempo de desenvolvimento.

- **Possíveis Mudanças Futuras:** Como o KMM é recente ainda está em evolução, podendo levar a mudanças em futuras versões. Mudanças significativas na estrutura ou nas práticas recomendadas podem exigir ajustes em projetos existentes, o que pode ser um desafio.

2.2.4 Análise Comparativa

Após analisar as vantagens e desvantagens das diversas *framework* introduzidas neste capítulo, foi elaborada a tabela 2.2 com o objetivo de oferecer uma comparação mais concisa e simplificada das características que cada um oferece ou não.

Características	Flutter	React Native	Kotlin Multiplatform Mobile (KMM)
Estrutura Simples com Componentes	✓	✓	×
Temas Integrados	✓	×	×
Recarregamento Rápido (*Hot Reloading*)	✓	✓	×
Atualização Rápida (*Fast Refresh*)	✓	✓	×
Renderização Própria	✓	×	×
Renderização e Integração Nativa	✓	✓	✓
Desempenho Próximo ao Nativo	✓	✓	✓
Comunidade e Ecossistema Amplos	✓	✓	×
Compatibilidade com Ferramentas de Desenvolvimento Padrão para Android e iOS	×	×	✓
Produtividade Aprimorada de Migração para Multiplataforma	×	×	✓
Menor Dificuldade de Implementação Nativa	×	✓	✓
Menor Curva de Aprendizagem	×	×	✓
Fácil Modificar Dados	×	×	✓
Bom Desempenho em Interações Complexas	✓	×	✓

Tabela 2.2: Análise comparativa das *framework* de desenvolvimento multiplataforma

Com base nas características apresentadas nas subsecções anteriores e na tabela 2.2, o Flutter destaca-se como a escolha preferencial para o desenvolvimento eficiente e consistente de uma aplicação móvel para Android e iOS. O Flutter incorpora a maioria das

características essenciais e desejadas, e as ausências identificadas não parecem representar problemas significativos.

Destacam-se as suas extensas coleções de *widgets* pré-criados, permitindo a criação de interfaces de utilizador apelativas, responsivas e facilmente personalizáveis. Além disso, a capacidade de alternar entre temas Android e iOS, juntamente com a compilação AOT para gerar código nativo, proporciona um desempenho próximo ao de aplicações nativos, assegurando uma experiência consistente.

O recurso *Hot Reload* acelera significativamente o ciclo de desenvolvimento, permitindo visualizar instantaneamente as mudanças no código. Notavelmente, o Flutter possui ainda o seu próprio mecanismo de renderização, resultando numa renderização rápida e eficiente para garantir o excelente desempenho das aplicações criadas.

Conforme evidenciado por Mehmet Isitan e Murat Koklu no seu estudo abrangendo diversos fatores relacionados a várias *framework* multiplataforma [19], tanto o Flutter quanto o React Native são responsáveis pela maior popularidade e apoio da comunidade de programadores nos últimos anos. No entanto, o Flutter não apenas superou o React Native, como também mantém uma clara vantagem em termos de desempenho.

Face ao apresentado, e tomando em conta que a empresa assim o desejava, a escolha da *framework* de desenvolvimento multiplataforma recaiu sobre o Flutter.

2.3 Videoconferência

Para a implementação da videoconferência, considerou-se explorar opções como Twilio Video SDK[9], Agora SDK[10] ou Zoom Video SDK[11], uma vez que oferecem conjuntos abrangentes de ferramentas, bibliotecas, documentação e recursos (SDK) que simplificam o desenvolvimento desta funcionalidade. No entanto, devido às complexidades adicionais de segurança associadas ao uso destes SDK no ambiente empresarial, bem como às dificuldades inerentes de personalização da solução conforme desejado, a análise recaiu sobre a tecnologia Web Real-Time Communication (WebRTC).

O WebRTC[12] é uma tecnologia *open-source* que viabiliza a comunicação em tempo real diretamente entre *browsers* da *web* e aplicações móveis (Android e iOS). Desenvolvido para suportar videoconferências e outras formas de comunicação em tempo real, o WebRTC destaca-se por permitir a comunicação ponto a ponto (Peer-to-peer (P2P)) [20] sem depender de servidores intermediários, permitindo aos utilizadores partilharem e transmitirem conteúdo como vídeo, áudio e mensagens, diretamente entre si, sem a dependência de um servidor central para coordenação da comunicação. Amplamente suportado por *browsers* modernos, como o Google Chrome, Mozilla Firefox e Microsoft Edge, o WebRTC integra-se facilmente em diversas plataformas e utiliza o protocolo Secure RTP (SRTP) [21] para

[9] https://www.twilio.com/pt-br
[10] https://www.agora.io/en/
[11] https://developers.zoom.us/docs/video-sdk/
[12] https://webrtc.org/?hl=pt-br

encriptar e autenticar voz e vídeo. Oferece ainda benefícios notáveis em redes Wireless Fidelity (WiFi), prevenindo a escuta e a gravação não autorizadas de voz e vídeo.

O WebRTC emprega também APIs essenciais para adquirir e comunicar dados em fluxo contínuo. A API `MediaStream` permite aceder aos dados da câmara e ao microfone dos dispositivos, a `RTCPeerConnection` é responsável por estabelecer e coordenar a conexão direta ponto a ponto (P2P) entre os *browsers*, lidando com a codificação, descodificação e questões como latência e congestionamento de rede, e a `RTCDataChannel` representa um canal bidirecional genérico para transferência de dados, possibilitando partilhar arquivos, troca de mensagens de texto e outras formas de comunicação de dados [20].

Na arquitetura da *framework* WebRTC, as conexões entre dispositivos (utilizadores) são estabelecidas através de endereços Internet Protocol (IP) [22], sendo necessário o IP de cada dispositivo para a sua identificação na rede. A complexidade do seu uso surge devido ao facto de todas as máquinas possuírem um IP interno (para comunicação dentro da rede) e um externo (para comunicação fora da rede), ambos traduzidos pelo processo de Network Address Translation (NAT) [23]. Para contornar esses processos de NAT e superar barreiras de redes com regras restritas de firewall (uma componente de segurança e monitorização de rede que atua como uma barreira entre uma rede privada interna e redes externas), tornou-se necessário o uso de servidores Session Traversal Utilities for NAT (STUN) e Traversal Using Relays around NAT (TURN) [24].

No entanto, o WebRTC, por si só, não lida com o estabelecimento da conexão ou a negociação de parâmetros de comunicação iniciais. Em vez disso, delega essas responsabilidades para um mecanismo externo conhecido como servidor de sinalização.

Nas próximas subseções, será apresentado o servidor de sinalização, integrado por meio do uso do Socket.IO baseado no protocolo WebSockets. Para complementar essa solução, também será abordado o uso adicional de servidores STUN e TURN.

2.3.1 Servidor de Sinalização e WebSockets

O protocolo WebSockets[13] é um protocolo de comunicação que estabelece um canal *full-duplex* (os dados podem ser transmitidos em ambas as direções simultaneamente) por meio de uma única ligação Transmission Control Protocol (TCP) de longa duração [25], desempenhando com eficácia o papel de intermediário na troca de mensagens de sinalização entre o navegador (cliente) e o servidor de sinalização. Este servidor de sinalização já foi previamente implementado no produto, sendo responsável por coordenar o estabelecimento e o encerramento das conexões P2P entre os utilizadores, bem como na troca de documentos adicionais durante a mesma.

A eficiência do WebSocket em proporcionar comunicação bidirecional e em tempo real é fundamental para o contexto de videoconferências, reduzindo significativamente a latência e a sobrecarga na rede, tornando a experiência do utilizador mais fluída e responsiva. Ao

[13] `https://websocket.org/`

contrário de abordagens convencionais como o Hypertext Transfer Protocol (HTTP), os WebSockets eliminam a necessidade constante de enviar pedidos extras para estabelecer conexões e transferir dados, otimizando assim a eficiência da comunicação. Além disso, a capacidade de manter uma conexão persistente contribui para uma troca contínua de mensagens de sinalização, sendo especialmente crucial durante o estabelecimento e a manutenção de uma sessão WebRTC.

Além disso, a natureza bidirecional dos WebSockets e a sua capacidade de suportar eventos assíncronos torna-os ideais para aplicações interativas, proporcionando uma experiência de utilizador mais dinâmica e adaptável.

Com o objetivo de aprimorar a eficiência da comunicação e a sua implementação, avaliou-se também o uso do Socket.IO[14] [26], uma biblioteca JavaScript que simplifica a comunicação bidirecional e em tempo real entre utilizadores e servidores. Esta biblioteca é construída sobre o protocolo WebSockets, podendo utilizá-lo como o seu protocolo principal de transporte. No entanto, destaca-se por oferecer uma abstração sobre o WebSockets que ultrapassa a simples eficiência. Ela é projetada não apenas na otimização dos recursos do WebSockets, mas também na maximização da flexibilidade, facilidade de implementação e manutenção, proporcionando funcionalidades melhoradas. Isto permite os programadores desenvolverem aplicações em tempo real sem se preocuparem com detalhes de rede de baixo nível, simplificando o processo de criação e oferecendo uma experiência de desenvolvimento mais intuitiva e eficaz. Além disso, incorpora mecanismos de *fallback* (contingência) para outros protocolos de transporte, caso os WebSockets não sejam suportados ou sejam restritos num ambiente específico. Estes mecanismos garantem que o Socket.IO se adapte a diferentes cenários, assegurando a continuidade da comunicação em tempo real.

Desta forma, o Socket.IO, construido em cima de WebSockets, pode desempenhar um papel crucial na aplicação móvel, simplificando o estabelecimento da conexão inicial entre dois utilizadores para a realização de uma videoconferência. Adicionalmente, poderá assumir eficientemente a responsabilidade de coordenar a extração dos dados biométricos solicitados durante as videoconferências, por meio de eventos acionados por um dos participantes do processo, devido à sua natureza de comunicação bidirecional e em tempo real.

2.3.2 Servidores STUN e TURN

O STUN é utilizado para descobrir o endereço IP público e a porta de um dispositivo situado atrás de um NAT, viabilizando a comunicação direta entre utilizadores. Já o TURN age como intermediário quando a comunicação direta é prejudicada por políticas restritivas de NAT ou firewalls, funcionando como um servidor intermediário para reencaminhar dados entre os utilizadores.

[14] https://socket.io/

A configuração destes servidores é integrada ao processo de estabelecimento da conexão WebRTC, assegurando uma conectividade eficaz em ambientes de rede diversos, sendo crucial para possibilitar comunicações em tempo real em cenários complexos de rede, como em videoconferências.

2.4 Dados Biométricos

A aplicação móvel desempenha um papel crucial na extração de dados biométricos, incluindo fotografias, assinaturas manuais digitais e impressões digitais, requerendo, por isso, a integração de diversas tecnologias e bibliotecas para realizar esta tarefa de maneira otimizada.

Esta secção destaca então as bibliotecas e tecnologias consideradas para realizar esta extração, bem como, nos casos das fotografias, conduzir a uma validação prévia dos dados. A validação das fotografias na aplicação móvel é essencial, pois é fundamental garantir imediatamente que elas atendem aos requisitos necessários, caso seja necessário submeter uma nova fotografia.

2.4.1 Fotografias

As fotografias podem incluir documentos, mas também retratos faciais, tornando essencial a utilização de uma tecnologia versátil que permita validar eficazmente a imagem. Este processo é fundamental para garantir a autenticidade e conformidade das fotografias, uma vez que a verificação prévia assegura a precisão e integridade dos dados biométricos fornecidos. Além disso, melhora a eficiência da extração, reduzindo o tempo necessário para esta funcionalidade, uma vez que a imagem não precisa de ser enviada para o servidor para validação.

O Firebase Machine Learning Kit (ML Kit)[15] é uma tecnologia que permite realizar este tipo de validação, integrando-se facilmente na *framework* Flutter. Este é um componente do conjunto Firebase[16] oferecido pela Google, que permite aos programadores de aplicações móveis integrar facilmente funcionalidades de aprendizagem automática nas suas aplicações. O ML Kit fornece um conjunto de APIs prontas a utilizar e modelos pré-treinados, abrangendo várias tarefas de aprendizagem automática [27]. Dentro deste *kit* destacam-se os seguintes componentes:

- **firebase-ml-vision**[17]**:** biblioteca que fornece um conjunto abrangente de APIs para reconhecimento e processamento de imagens, incluindo reconhecimento de texto, leitura de código de barras, rotulagem de imagens, deteção de rosto e reconhecimento de pontos de referência.

[15] https://firebase.google.com/docs/ml-kit?hl=pt
[16] https://firebase.google.com/?hl=pt
[17] https://pub.dev/documentation/firebase_ml_vision/latest/

- **firebase-ml-vision-face-model**[18]**:** é um módulo especializado do ML Kit, especificamente concebido para tarefas relacionadas com rostos. Oferece funcionalidades para deteção de faces, reconhecimento de marcas faciais e outras características de análise facial.

Além do Firebase ML Kit, é possível aproveitar um sensor incorporado nos dispositivos móveis para obter informações sobre o movimento e a orientação do aparelho, o acelerómetro [28]. O acelerómetro mede a aceleração linear ao longo de três eixos, fornecendo informações sobre variações de velocidade e direção em cada um dos eixos (x, y e z). É utilizado em funcionalidades como a rotação do ecrã, detetando alterações no seu movimento.

2.4.2 Assinatura manuscrita digital

Para incorporar a funcionalidade de extração e assinatura de documentos Portable Document Format (PDF) usando o Flutter, várias alternativas foram consideradas ao explorar as ferramentas disponíveis da *framework*[19]. Entre as opções avaliadas, destacam-se duas alternativas específicas discutidas nas subsecções seguintes usando duas abordagens distintas.

2.4.2.1 Bibliotecas e *Plugins*

Após um estudo aprofundado das bibliotecas e *plugins* disponíveis para Flutter, concluiu-se que a *framework* não dispõe de uma única biblioteca ou *plugin* que permita tanto a renderização e exibição quanto a manipulação e edição de documentos PDF. Como resultado, foi necessário adotar uma abordagem que utiliza várias componentes distintas para atender a essas necessidades específicas.

A biblioteca `syncfusion-flutter-pdf`[20] foi selecionada para a manipulação e edição de ficheiros PDF. Esta biblioteca permite criar, ler e editar documentos PDF sem depender de plataformas externas, oferecendo uma gama completa de funcionalidades. Além disso, suporta a adição de texto formatado, imagens, formas, tabelas, listas, cabeçalhos e rodapés, proporcionando uma flexibilidade significativa para personalizar documentos PDF de acordo com as necessidades do projeto, que será a edição da imagem da assinatura extraída no documento PDF.

Para exibir o PDF na aplicação Flutter, será necessário utilizar tecnologias projetadas especificamente para renderizar conteúdo PDF. Entre as mais destacadas, devido à sua popularidade, estão:

[18] https://pub.dev/documentation/firebase_ml_vision/latest/
[19] https://pub.dev/
[20] https://pub.dev/packages/syncfusion_flutter_pdf

- **flutter-pdfview**[21]**:** Este *plugin* fornece um *widget* Flutter para renderizar documentos PDF dentro de aplicações. Ele utiliza recursos nativos de renderização de PDF tanto no Android quanto no iOS, oferecendo uma experiência de visualização de PDF específica para cada plataforma e com bom desempenho. Os programadores podem incorporar o *widget* `PDFView` para exibir arquivos PDF existentes, fornecendo funcionalidades como gestos de deslizar e navegação de página.

- **native-pdf-view / pdfx** [22]**:** Este *plugin* oferece funcionalidades muito semelhantes ao *plugin* anterior, proporcionando renderização nativa e exibição de documentos PDF tanto em Android quanto em iOS. Ele apresenta duas API distintas: uma para renderização, a `renderer`, e outra para visualização, a `viewer`. Assim como o `flutter-pdfview`, a API de visualização também fornece um conjunto de *widgets* e controladores que facilitam a exibição de documentos PDF.

O *plugin* `flutter-pdfview` é considerado o mais conceituado e amplamente adotado pela comunidade de programadores, para além do seu desempenho superior em comparação com o *plugin* `pdfx`. Além disso, o `flutter-pdfview` destaca-se pela sua estabilidade e por ter uma presença consistente de manutenção, o que contribui para uma experiência mais confiável e suportada ao longo do tempo [29].

Em semelhança com a renderização e edição de documentos, a biblioteca `syncfusion-flutter-pdf` e o *plugin* `flutter-pdfview` também não foram concebidos para fornecer funcionalidades explícitas para a extração de assinaturas. No entanto, é possível implementar a realização de uma assinatura digital manuscrita utilizando outro *plugin* do Flutter, o `signature`[23]. Este *plugin* inclui um *widget* denominado `Signature`, que permite aos utilizadores desenhar ou inserir assinaturas manuais no ecrã do dispositivo móvel. A assinatura pode ser, então, extraída e salva como uma imagem nos formatos Portable Network Graphics (PNG) ou Joint Photographic Experts Group (JPEG). Ao integrar este *widget* com a biblioteca `syncfusion-flutter-pdf`, é então possível capturar a assinatura de um utilizador e incorporá-la num documento PDF.

2.4.2.2 Pspdfkit

O PSPDFKit[24] é um SDK comercial amplamente adotado, projetado para simplificar a incorporação de funcionalidades avançadas de PDF em diversas aplicações. Este SDK oferece suporte a várias plataformas, incluindo iOS e Android, proporcionando recursos abrangentes para a visualização, anotação, edição e assinatura manual digital de documentos PDF. Frequentemente utilizado por programadores, o PSPDFKit possibilita aprimorar as capacidades das aplicações que lidam intensivamente com tarefas relacionadas a PDF

[21] https://pub.dev/packages/flutter_pdfview
[22] https://pub.dev/packages/native_pdf_view
[23] https://pub.dev/packages/signature
[24] https://pspdfkit.com/

garantindo uma experiência contínua e consistente em diferentes plataformas e dispositivos. Recentemente, o PSPDFKit disponibilizou um pacote específico para o Flutter, chamado `pspdfkit-flutter`[25], do qual o estudo desta alternativa se insere.

No entanto, a escolha da solução a adotar recaiu sobre a utilização das bibliotecas `syncfusion-flutter-pdf` e dos *plugins* `flutter-pdfview` e `signature`, devido à sua ampla adoção e suporte pela comunidade de programadores [29], bem como à sua eficácia e desempenho comprovados na renderização e manipulação de documentos PDF em aplicações móveis Flutter. A integração destas tecnologias proporciona uma solução robusta e eficiente para a manipulação de documentos PDF, oferecendo um elevado grau de personalização e flexibilidade. Estes aspetos são fundamentais para implementar a funcionalidade desejada, algo que a outra abordagem não proporciona.

2.4.3 Impressões digitais

A versão original do produto **idfyme** inclui a funcionalidade de extração de impressões digitais durante a extração biométrica do utilizador. No entanto só a possui na versão Android, pois não foram identificados leitores de impressões digitais compatíveis no sistema operativo iOS para uso no produto.

Na versão Android da aplicação, usa-se o leitor de impressões digitais `Digital Persona U.are.U 4500` representado na figura 2.1, para o qual se incorpora a biblioteca `asia.kanopi.tools:fingerscan`[26]. Esta biblioteca desempenha um papel fundamental na leitura e transformação da imagem original da impressão digital, capturada pelo utilizador no dispositivo mencionado, para um formato de lista de *bytes* manipulável pela aplicação.

Figura 2.1: Leitor de impressões digitais `DigitalPersona U.are.U 4500`

Para além disso, foi ainda usada a biblioteca `androidx.biometric`[27], que faz parte do AndroidX [30], um conjunto de bibliotecas disponibilizado pela Google para simplificar e aprimorar o desenvolvimento de aplicações Android. O `androidx.biometric` é uma

[25] https://pub.dev/packages/pspdfkit_flutter
[26] https://github.com/shodgson/uareu
[27] https://developer.android.com/reference/androidx/biometric/package-summary

biblioteca que oferece, de maneira consistente e segura, a autenticação biométrica para as impressões digitais na plataforma Android [31].

No entanto, devido ao tempo limitado para o desenvolvimento da solução e ao objetivo principal de garantir a sua adaptação multiplataforma, decidiu-se em conjunto com a empresa não implementar esta funcionalidade nesta versão da solução. A sua inclusão será avaliada numa fase posterior, permitindo garantir a compatibilidade e eficácia de todas as funcionalidades desta fase em ambos os sistemas operativos. Além disso, possibilitará que, no futuro, já possa existir um leitor para a versão iOS que possa ser utilizado.

2.5 Base de Dados Local

Uma aplicação móvel deve ter a capacidade de utilizar uma base de dados local em memória (na Random Access Memory (RAM) do dispositivo) para armazenar dados de sessão. Isto proporciona uma acessibilidade rápida aos dados, garantindo persistência temporária e resultando em melhorias significativas no desempenho da aplicação. Esta funcionalidade é particularmente crucial para dados de sessão, pois permite a rápida recuperação e atualização desses dados durante a interação do utilizador com a aplicação.

A implementação desta funcionalidade irá ocorrer num ambiente multiplataforma utilizando o Flutter, visando otimizar o desenvolvimento e proporcionar uma solução mais integrada e eficaz. Este contexto motivou a pesquisa por bibliotecas e *plugins* específicos do Flutter, que serão apresentados a seguir, com o objetivo de integrar esta componente de maneira eficiente.

Primeiramente, foi explorada a viabilidade da biblioteca `floor`[28]. Esta biblioteca representa uma abstração do SQLite, desenvolvida para tornar mais acessíveis as operações nas bases de dados locais dentro de aplicações Flutter. O SQLite [32], sendo um sistema de gestão de base de dados relacional, diferencia-se de sistemas convencionais pela sua autonomia, dispensando a necessidade de um servidor separado e armazenando os dados num único ficheiro no sistema de arquivos do dispositivo móvel. O `floor` foi então projetado com o propósito de simplificar a manipulação da base de dados SQLite em aplicações multiplataforma, oferecendo uma API mais robusta e amigável em comparação com a manipulação direta do SQLite. Este aspeto simplifica significativamente a criação, consulta e manipulação dos dados, proporcionando uma experiência de desenvolvimento mais eficiente e intuitiva.

A biblioteca também incorpora o conceito de Data Access Objects (DAO), um padrão de *design* que oferece uma abordagem simplificada e consistente para interagir com fontes de dados, como bases de dados ou APIs. Esta metodologia utiliza anotações específicas

[28] `https://pub.dev/packages/floor`

aplicadas em classes e métodos, permitindo a definição clara de operações comuns em bases de dados, como inserção, recuperação, atualização e eliminação. O DAO proporciona uma estrutura organizada e uniforme para o acesso aos dados, contribuindo para a clareza do código e simplificando as tarefas relacionadas à manipulação de informações em aplicações Flutter.

A utilização desta biblioteca oferece por isso uma vantagem de viabilizar o armazenamento estruturado de dados, assemelhando-se a um sistema de base de dados, simplificando e aprimorando a implementação desta funcionalidade. Além disso, apresenta benefícios como a nulidade segura (*null-safe*), segurança para erros de tipos de dados, reatividade, leveza, foco em Structured Query Language (SQL), simplicidade e ausência de custos adicionais de implementação.

Além disso, analisou-se o uso do *plugin* `shared-preferences`[29] do Flutter para armazenar e recuperar persistentemente pequenas quantidades de dados chave-valor no dispositivo móvel. É particularmente útil para cenários onde é necessário manter as preferências do utilizador, definições, ou quaisquer outros dados que devam persistir entre lançamentos de novas versões da aplicação. O *plugin* abstrai os detalhes de armazenamento subjacentes, facilitando a interação com as preferências partilhadas de uma forma independente da plataforma. O uso deste *plugin* envolve métodos simples e intuitivos para definir e recuperar valores, e oferece um mecanismo de *fallback* (contingência) para fornecer valores padrão no caso de uma preferência solicitada não ser encontrada. Assim, este *plugin* simplifica a gestão do armazenamento local de dados em aplicações Flutter, melhorando a experiência do utilizador ao permitir que a aplicação se lembre e aplique configurações específicas do utilizador em diferentes sessões.

No entanto, após uma análise comparativa com o `floor` optou-se por utilizar esta biblioteca em vez do `shared-preferences` para a gestão de dados na aplicação devido à natureza e complexidade dos dados a serem armazenados.

A biblioteca `floor` oferece uma abstração sobre o SQLite, proporcionando uma solução mais robusta e escalável para manipulação de bases de dados relacionais. Isto é particularmente vantajoso quando se necessita de operações complexas, como consultas avançadas e transações, que o `shared-preferences` não suporta, uma vez que este é mais adequado para o armazenamento de pequenas quantidades de dados chave-valor, como preferências e configurações do utilizador. O `floor` ainda permite uma organização mais estruturada dos dados, com suporte para operações como inserção, atualização e eliminação em tabelas, o que facilita o desenvolvimento e manutenção de aplicações que exigem um armazenamento de dados mais complexo e relacional esperado ter na solução.

[29] https://pub.dev/packages/shared_preferences

Por fim, estudou-se a possibilidade do uso adicional do *plugin* `flutter-secure-storage`[30] do Flutter, de modo a proporcionar um armazenamento seguro para informações sensíveis da aplicação, como o PIN de acesso à aplicação e *tokens* de autenticação do utilizador. Este *plugin* concentra-se em melhorar a proteção de dados, fornecendo uma solução de armazenamento seguro e criptográfico para dados confidenciais, como *tokens* de autenticação ou chaves de API. Aproveitando os mecanismos de armazenamento seguro específicos de várias plataformas, incluindo o iOS Keychain[31] e o Android Keystore[32], o `flutter-secure-storage` garante que os dados armazenados permaneçam protegidos contra acessos não autorizados numa aplicação multiplataforma.

2.6 Síntese

A implementação da aplicação móvel proposta, com as suas diversas funcionalidades, exigiu uma cuidadosa seleção e análise de tecnologias, bibliotecas e *plugins*. Optou-se pelo desenvolvimento multiplataforma, pois, em comparação com o desenvolvimento nativo, oferece maior eficiência e produtividade, além de assegurar uma maior consistência e uniformidade entre as plataformas Android e iOS.

Dentro das opções disponíveis, a *framework* Flutter foi escolhida em vez do React Native e do KMM. Esta escolha foi baseada na sua capacidade de fornecer uma experiência de desenvolvimento mais rápida e eficiente, bem como na sua ampla adoção e suporte pela comunidade de programadores. A *framework* Flutter destaca-se pela sua abordagem de desenvolvimento centrada no programador, oferecendo uma vasta gama de ferramentas, bibliotecas e *plugins* que simplificam a implementação de funcionalidades complexas e personalizadas.

Para a funcionalidade de videoconferência, foram avaliadas a tecnologia WebRTC em conjunto com o protocolo WebSockets, utilizando a biblioteca Socket.IO para simplificar a sua implementação. Esta abordagem foi considerada mais flexível e personalizável em comparação com soluções como Twilio Video SDK, Agora SDK, ou Zoom Video SDK, que, embora sejam opções robustas e prontas para uso, apresentam limitações em termos de personalização e segurança.

Para garantir a estabilidade e segurança das comunicações, também foi considerada a implementação de um servidor de sinalização, essencial para a inicialização e gestão das conexões WebRTC entre os pares. Além disso, foram incluídos servidores STUN e TURN para facilitar a passagem das NATs e firewalls, garantindo a comunicação eficiente entre os dispositivos em redes complexas ou restritas.

Para a análise e validação de fotografias, contemplou-se a utilização do Firebase ML

[30] https://pub.dev/packages/flutter-secure-storage
[31] https://developer.apple.com/documentation/security/keychain_services/
[32] https://developer.android.com/privacy-and-security/keystore?hl=pt-br

Kit, destacando-se pelas suas robustas capacidades de aprendizagem automática. Esta integração foi ainda complementada pela consideração do uso do acelerómetro, sensor incorporado nos dispositivos móveis responsável por avaliar a rotação do ecrã.

No caso das assinaturas digitais manuais, foram estudadas e analisadas duas abordagens diferentes: uma utilizando várias bibliotecas e *plugins* do Flutter, e outra utilizando o SDK comercial PSPDFKit. A primeira abordagem, que inclui a biblioteca `syncfusion-flutter-pdf` e os *plugins* `flutter-pdfview` e `signature`, foi considerada a mais adequada devido à sua ampla adoção e suporte pela comunidade de programadores, bem como à sua eficácia, desempenho e elevado grau de personalização e flexibilidade.

Para a extração de impressões digitais, conclui-se que a funcionalidade não será implementada nesta versão da solução devido ao facto de não ter sido identificado um leitor de impressões digitais compatível com o sistema operativo iOS. Implementar esta funcionalidade apenas para um dos sistemas operativos não faria sentido, uma vez que a solução tem como foco a sua característica multiplataforma.

No que diz respeito à implementação da base de dados local, armazenada em memória no dispositivo móvel, foram estudadas diversas tecnologias da *framework* Flutter. Neste contexto, a biblioteca `floor` foi considerada a mais adequada para uma gestão eficiente dos dados em memória, tirando proveito do mapeamento e da utilização de SQL, em detrimento do uso do *plugin* `shared-preferences`, que é mais indicado para o armazenamento persistente de pequenos volumes de dados. Além disso, para garantir o tratamento seguro de informações sensíveis na aplicação, foi também estudado o uso do *plugin* `flutter-secure-storage`.

3

idfyme - Transformação Total do Atendimento

Este capítulo apresenta e contextualiza o produto **idfyme**, desenvolvido pela Opensoft, que serviu como alicerce para a solução desenvolvida. Será abordada uma descrição mais técnica das suas funcionalidades, estrutura e componentes, apoiada por gráficos complementares. Adicionalmente, serão detalhadas as tecnologias implementadas no produto.

3.1 Descrição do Produto

O produto **idfyme** é uma solução de *software* totalmente personalizável, concebida para se adaptar de forma flexível às necessidades específicas de cada organização. Com uma abordagem remota e digital, o **idfyme** elimina a necessidade de presença física do cliente em serviços legais que requerem autenticação robusta, utilizando três processos principais: videoconferência com e sem extração biométrica, extração de dados biométricos e assinatura manuscrita digital de documentos.

O produto **idfyme** inclui uma plataforma *web* e uma aplicação móvel Android, ambas totalmente personalizáveis e preparadas para ajustes futuros, de forma a responder a novas exigências. Estas soluções de *software* integram funcionalidades como a assinatura manuscrita digital de documentos, a recolha de dados para a posterior emissão de documentos de identificação (como passaportes, cartões de identidade e certidões de nascimento), a extração biométrica de vários dados (fotografias, assinatura e impressões digitais), e a comunicação direta entre clientes e funcionários por meio de videoconferência.

De forma a viabilizar estas funcionalidades, a plataforma oferece os seguintes recursos:

- **Histórico de Processos:** Histórico de atividades de forma a que o utilizador consiga ver e/ou aceder aos processos que já realizou ou tem por realizar;
- **Videoconferência:** Participação, armazenamento e agendamento de videoconferências entre funcionários e clientes, com ou sem guias integrados que orientam as ações a serem seguidas pelos funcionários para a extração de determinados dados

biométricos do cliente, como a obtenção de fotos da face e do documento de identificação, a extração da assinatura digital manuscrita e das impressões digitais, durante a prestação de serviços, como para a emissão de documentos de identificação.

Além disso, nas videoconferências com guias, é gerado um *token*, validado o ambiente do cidadão e colocadas perguntas pessoais, que são confirmadas por ambas as partes envolvidas, tanto pelo cliente como pelo funcionário, no início da sessão. Esta medida visa reforçar ainda mais a segurança no processo de autenticação do cliente neste serviço.

- **Extração Biométrica:** Extração e validação pelo funcionário de dados biométricos do utilizador, incluindo assinatura manual digital, impressões digitais e foto da face.
- **Assinatura de Documentos:** Assinatura manual digital de documentos com a assinatura manuscrita do utilizador, posicionada num local escolhido pelo próprio no documento, assim como a visualização dos documentos assinados ou pendentes de assinatura e a sua validação pelo funcionário.

3.2 Componentes do Produto

A figura 3.1 ilustra uma visão geral dos componentes que compõem a aplicação móvel Android e *web* do produto.

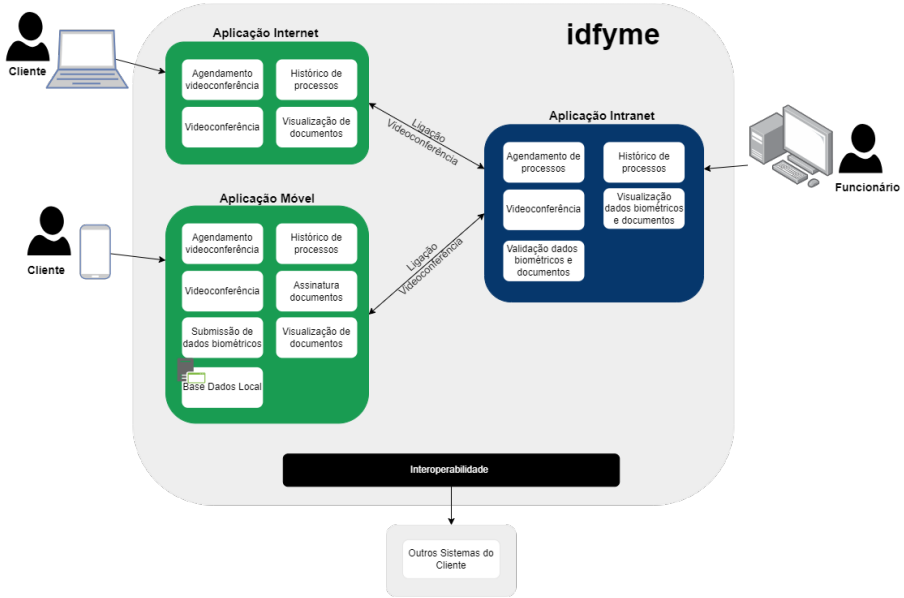


Figura 3.1: Visão geral dos componentes da aplicação móvel Android e *web* **idfyme**

O produto tem os seguintes componentes:

- **Aplicação Intranet-** Aplicação *web* que contém a parte da rede privada de atividades e comunicação utilizada internamente pelos funcionários de uma organização. Ela

permite a criação e partilha segura dos processos e documentos, proporcionando um ambiente controlado e protegido. Os funcionários podem vizualizar e validar dados biométricos e documentos assinados pelos clientes, participar em videoconferências normais ou com fim a extração de dados biométricos do cliente (seguindo um *script* / guia integrado), agendar os diversos processos para clientes (como recolha de dados biométricos, videoconferências e assinatura de documentos) e consultar o histórico de todos os processos realizados e por realizar na aplicação.

- **Aplicação Internet-** Aplicação *web* destinada ao uso público, clientes, acessível por meio da *World Wide Web* através de um navegador *web*. Aqui os clientes podem agendar e participar em videoconferências normais sem troca de dados biométricos, visualizar documentos assinados ou por assinar e aceder aos processos já realizados e por realizar.

- **Aplicação Móvel-** Aplicação móvel Android na qual o cliente pode realizar as funcionalidades da aplicação Internet com a adição de poder assinar manuscritamente documentos digitalmente, submeter dados biométricos como fotografias, assinatura manuscrita digital e impressões digitais, e realizar videoconferências para a extração de dados biométricos.

  Esta componente possui ainda um módulo de base de dados local que armazena informações diretamente na memória do dispositivo móvel, sendo responsável por guardar os dados de sessão dos utilizadores. Esta abordagem é escolhida principalmente devido à rápida acessibilidade dos dados, à persistência temporária e ao notável aprimoramento no desempenho da aplicação. Este método revela-se especialmente crucial para dados de sessão, os quais necessitam de ser recuperados e atualizados de forma ágil durante a interação do utilizador com a aplicação móvel.

O produto mantém-se ainda aberto à possibilidade de interoperabilidade com outros sistemas relevantes para o cliente, como, por exemplo, a utilização da Chave Móvel Digital ou recursos avançados de deteção de vivacidade facial (*face liveness*).

3.3 Componentes da Infraestrutura do Produto

A figura 3.2 apresenta uma visão geral da estrutura física do produto mostrando os componentes da sua infraestrutura, tanto da sua aplição móvel como *web*. A **Aplicação Intranet**, a **Aplicação Internet** e a **Aplicação Móvel** interagem, assim, diretamente com as estruturas:

- **Servidor *Backend-*** Componente crucial responsável pela gestão e processamento de dados, pela lógica de negócios e pela comunicação com outros serviços (módulos de apoio) e base de dados. O servidor *backend* funciona como o lado do servidor da

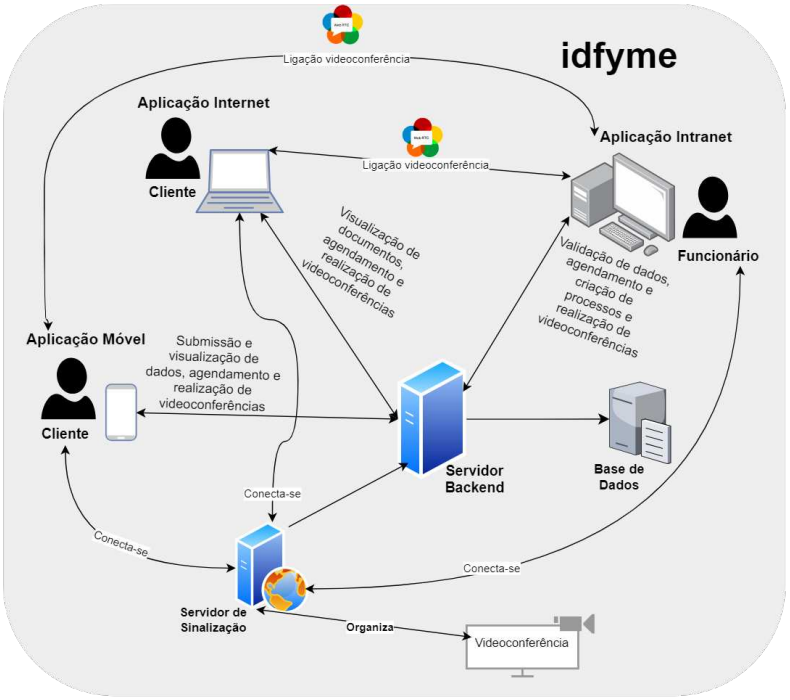


Figura 3.2: Visão geral da estrutura física da aplicação móvel e *web* **idfyme**

aplicação, tratando da facilitação da comunicação entre o *frontend* da aplicação, a base de dados e o servidor de sinalização por meio da sua API.

- **Servidor de Sinalização-** O servidor de sinalização desempenha um papel fundamental na orquestração e coordenação dos participantes para a realização das videoconferências. Ele atua como intermediário, coordenando a troca de informações de controlo necessárias para as estabelecer ou terminar, e para a troca de documentos e dados adicionais durante as mesmas. Desta forma, as aplicações Intranet, Internet e Móvel estabelecem uma conexão inicial com o servidor de sinalização para iniciar uma sessão de videoconferência e realizar a troca de dados e documentos. Posteriormente, estabelecem uma conexão ponto-a-ponto (*peer-to-peer*) diretamente entre si, otimizando o uso de recursos e da rede sem ter de envolver este servidor.

A implementação das aplicações **Intranet**, **Internet** e **Móvel**, bem como a sua integração com as estruturas anteriores, está distribuída em três projetos distintos:

- ***idfyme-plugins:*** Projeto que contém a aplicação *web* (**Aplicação Internet e Intranet**) e o *backend* do produto **idfyme**, sendo o responsável por toda a lógica de negócio do produto e a comunicação com a base de dados.

- ***idfyme-android:*** Projeto que contém a aplicação android (**Aplicação Móvel**) já implementada.

- ***idfyme-flutter:*** Projeto que contém a aplicação Flutter implementada que constitui a solução (**Aplicação Móvel**), destinada a substituir o projeto **idfyme-android** por uma aplicação móvel multiplataforma para Android e iOS.

3.4 Tecnologias Utilizadas

A aplicação *web* foi desenvolvida por base no Liferay[1]. O Liferay [33] é uma plataforma de experiência digital de código aberto concebida para ajudar as organizações a criar e gerir experiências digitais envolventes para os seus utilizadores. Com um forte foco no fornecimento de uma plataforma unificada para gestão de conteúdos *web*, colaboração e funcionalidades de portal empresarial, o Liferay permite às empresas criar e implementar *websites*, *intranets* e *extranets*. A sua arquitetura modular permite que os programadores ampliem e personalizem a funcionalidade através de *plugins* e extensões, proporcionando flexibilidade e escalabilidade.

A aplicação móvel Android e a aplicação *web* foram ainda desenvolvidas usando Java 8.

Para as funcionalidades do produto descritas na secção 3.1, foram adotadas diversas tecnologias a referir de seguida, a maioria das quais está detalhada no capítulo 2.

Videoconferência

Para a implementação da videoconferência destaca-se a utilização do WebRTC com o suporte dos servidores STUN e TURN para estabelecer a conexão ponto a ponto entre os utilizadores, possibilitando a transmissão de vídeo e áudio nas videoconferências. Além disso, contou-se com o auxílio do Socket.IO sobre o protocolo WebSockets para a integração do servidor de sinalização, responsável pela coordenação da conexão inicial e encerramento das sessões, bem como pela transmissão de dados e arquivos durante as videoconferências.

Dados Biométricos

A validação das fotografias foi realizada por meio do Firebase ML Kit e pelo uso do acelerómetro dos dispositivos móveis.

Para a assinatura manuscrita digital, utilizaram-se as bibliotecas Java `AndroidPdfViewer`[2] e `iText`[3] para manipulação eficiente de PDFs. A primeira destina-se à renderização eficiente de documentos PDF no Android, oferecendo suporte a animações, gestos, *zoom* e duplo toque. A segunda, por sua vez, é voltada para a criação, extração e manipulação dinâmica de documentos PDF, simplificando tarefas como adição de texto, imagens, tabelas e *links*, além de suportar assinaturas manuais digitais.

[1] https://www.liferay.com/
[2] https://github.com/barteksc/AndroidPdfViewer
[3] https://itextpdf.com/

Para a extração das impressões digitais, utilizou-se o dispositivo `DigitalPersona U.are.U 4500`, com o auxilio da biblioteca `asia.kanopi.tools:fingerscan` para a sua conversão para formato digital, e da biblioteca Java `androidx.biometric` para a sua implementação.

Base de Dados

No que diz respeito ao armazenamento de dados, adotou-se uma abordagem local para a aplicação móvel Android, utilizando as bibliotecas `Room`[4] e `Shared Preferences`[5]. O `Room` é uma poderosa biblioteca de persistência que simplifica a interação com bases de dados SQLite no Android. Ela fornece uma camada de abstração eficiente, permitindo a definição de entidades e operações de acesso a dados por meio de DAO facilitando a manipulação de dados estruturados na aplicação [34]. Já o `SharedPreferences` é uma solução leve e simples para armazenar pequenas quantidades de dados, como configurações e preferências do utilizador, em pares de chave-valor. Esta abordagem é ideal para informações simples que não requerem uma estrutura de base de dados completa como dados de sessão.

Além disso, foi utilizada uma base de dados externa em SQL, empregada para armazenar dados de forma mais abrangente na aplicação. Adicionalmente, optou-se por utilizar o Amazon S3 (*Simple Storage Service*)[6], um serviço de armazenamento baseado na nuvem fornecido pela Amazon Web Services (AWS)[7], para armazenar mais facilmente e eficientemente os vídeos e áudios obtidos nas videoconferências.

[4] https://developer.android.com/training/data-storage/room?hl=pt-br
[5] https://developer.android.com/reference/android/content/SharedPreferences
[6] https://aws.amazon.com/pt/s3/
[7] https://aws.amazon.com/pt/

4

Descrição e Análise da Solução

Esta tese aborda o problema da limitada inclusão digital do produto **idfyme**, que, ao dispor apenas de uma versão para Android, deixava 27,5% do mercado fora [4], composto por utilizadores de iOS. Para resolver esta questão, foi elaborada e implementada uma solução que, após um estudo e análise detalhada, resultou no desenvolvimento de uma aplicação móvel multiplataforma integrando versões para iOS e Android do produto num único projeto.

A solução visou então consolidar as funcionalidades do **idfyme** numa única aplicação que operasse em ambos os sistemas operativos, aproveitando as vantagens de uma base de código unificada. Entre as principais vantagens desta abordagem destacam-se a redução de custos e tempo de desenvolvimento e manutenção, a maior flexibilidade na implementação e a consistência na experiência do utilizador.

4.1 Descrição da Solução

O objetivo da solução foi então desenvolver uma versão móvel do produto **idfyme** para iOS e implementá-la de forma multiplataforma com a versão já existente para Android. Assim sendo, foi necessário identificar, de forma a manter, todas as funcionalidades já existentes no produto, que incluiem:

- **Autenticação do utilizador** perante dados de *login* e registo;
- **Realização do ecrã inicial** com a listagem e filtragem de todos os processos associados ao utilizador, permintindo a consulta do histórico dos processos;
- **Realização e agendamento de videoconferências** com a possibilidade de extração de dados biométricos durante a sua realização, através do seguimento de um *script* / guia. Esta extração deve incluir um processo de validação do ambiente do cidadão e a realização de perguntas pessoais, bem como a verificação de um *token* recebido, para garantir uma segurança mais robusta. Adicionalmente, deve também incluir a extração e envio da assinatura manuscrita digital e de fotografias da face e dos documentos de identificação.

- **Recolha de dados biométricos** que incluiem foto da face e assinatura manual;
- **Assinatura de documentos digitalmente** através de uma assinatura manuscrita digital num local escolhido pelo utilizador no documento;

4.2 Análise das Versões Suportadas na Solução

A solução desenvolvida precisou ainda de estabelecer restrições quanto às versões suportadas dos sistemas operativos iOS e Android, dado que oferecer suporte a todas as versões tornaria o processo de desenvolvimento excessivamente complexo, extenso e dispendioso. Assim, para o sistema operativo iOS, a aplicação móvel foi projetada para ser compatível com a versão 13.0 ou superior, enquanto para Android, o suporte foi direcionado para a versão 6.0 ou superior.

Estas restrições foram cuidadosamente definidas com base numa análise de mercado que considerou a compatibilidade com as versões mais recentes dos sistemas operativos. O objetivo foi garantir que a aplicação fosse acessível ao maior número possível de utilizadores, mantendo uma compatibilidade de quase 100% com as versões selecionadas [4], sem comprometer o seu desenvolvimento e custos. Este enfoque permitiu equilibrar a inovação tecnológica com a viabilidade técnica, garantindo que a aplicação atendesse às necessidades de uma ampla base de utilizadores, sem comprometer a eficiência e a qualidade do desenvolvimento. Desta forma, o projeto conseguiu conciliar inovação com sustentabilidade [35].

4.3 Fluxo da Solução

O diagrama 4.1 representa o fluxo da solução da aplicação móvel oferecendo uma visão mais nítida da sequência de interações e das diversas funcionalidades disponíveis na aplicação. Esta representação detalhada destaca as etapas e opções disponíveis ao longo do percurso do utilizador na aplicação.

Para facilitar a compreensão dos estados de cada processo, estes foram divididos em quatro cores distintas, semelhantes às utilizadas na aplicação móvel:

- **Azul:** Representa uma ação pendente para o utilizador realizar.
- **Amarelo:** Representa uma ação pendente para o funcionário realizar.
- **Verde:** Representa que o processo já foi validado e concluído.
- **Vermelho:** Representa falha, erro ou recusa no processo.

O utilizador inicia a sua atividade na aplicação móvel ao efetuar o *login* ou criar uma conta. Neste processo, é necessário criar ou inserir um PIN de 4 a 8 dígitos, que serve como

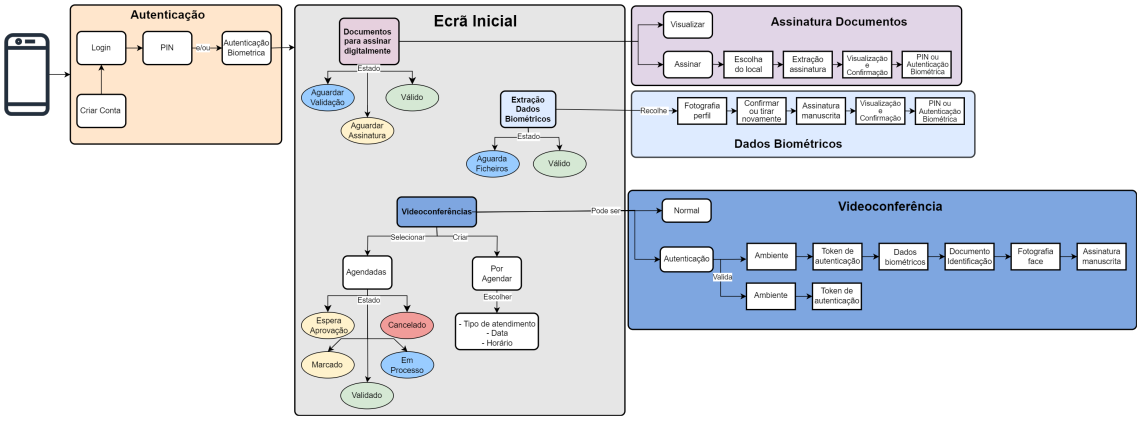


Figura 4.1: Solução do fluxo da aplicação móvel **idfyme**

uma camada adicional de segurança. Após a primeira autenticação, o procedimento é simplificado nas subsequentes, exigindo apenas o PIN, não requerendo um *login*. Também é possivel substituir o uso do PIN por uma autenticação biométrica integrada do dispositivo móvel, como a impressão digital ou o reconhecimento facial, se o utilizador assim o preferir.

Posteriormente, o utilizador é redirecionado para o ecrã inicial da aplicação. Aqui, o utilizador pode visualizar todos os processos associados a ele, concluidos ou por concluir, e desencadeiar várias ações, que incluem:

- **Assinar Documentos:**

  Visualização de documentos pendentes de assinatura ou já assinados (aprovados ou em fase de aprovação), com a possibilidade de assinar aqueles que ainda estão por assinar. Aqui, o utilizador segue a seguinte ordem: escolha do local para colocar a assinatura manuscrita digital no documento, extração da assinatura, visualização do documento assinado com a possibilidade de repetir a assinatura, cancelar o processo ou submeter o documento, e, por fim, a inserção do PIN ou dos dados biométricos (dependendo do método escolhido durante o *login*) para a aprovação da submissão.

- **Dados Biométricos:**

  Submissão de dados biométricos, onde o utilizador segue a seguinte ordem: captura de uma fotografia da face, confirmação da fotografia capturada (ou possibilidade de tirar uma nova), extração da assinatura manuscrita, visualização dos dados extraídos com a possibilidade de repetir a sua extração, cancelar o processo ou submeter os dados, e, por fim, a inserção do PIN ou dos dados biométricos (dependendo do método escolhido durante o *login*) para a aprovação da submissão.

- **Videoconferências:**

  Visualização das videoconferências já agendadas, ainda por realizar, e as já realizadas, assim como participar em videoconferências e agendar novas videoconferências, escolhendo o tipo de atendimento desejado, a data e o horário pretendido.

Existem dois tipos de videoconferências:

- **Normais:** Incluiem uma conversa normal entre cliente e funcionário destinada à realização de serviços específicos como por exemplo o esclarecimento de dúvidas;
- **Autenticação:** Videoconferências dedicadas à autenticação do cliente através da extração de vários dados biométricos para a emissão ou renovação de documentos de identificação. Esta abordagem é suportada por um *script*/guias que orienta o funcionário na execução da extração dos dados dos clientes.

  Aqui, podem ser realizados dois serviços (guias) distintos para a emissão e renovação dos documentos. Ambos incluem a validação do ambiente do cidadão e a verificação de um *token* recebido para garantir uma segurança mais robusta. O segundo tipo de serviço, além dessas validações, oferece ainda uma verificação biométrica mais completa, que envolve perguntas de identificação pessoal, extração de fotografias do documento de identificação, captura de uma fotografia do rosto e a extração da assinatura digital manual.

4.4 Análise do *Design* da Interface

Para garantir que a aplicação implementada estivesse alinhada com os requisitos do cliente e da empresa, conduziu-se ainda uma análise preliminar detalhada. Este processo garantiu que a UI fosse intuitiva e simples para os utilizadores, promovendo coerência e fluidez em toda a aplicação móvel. Além disso, foi crucial garantir que o fluxo das páginas fosse acessível e intuitivo, proporcionando uma experiência de navegação fluida e eficiente [36].

Durante o estudo prévio, foram cuidadosamente considerados aspectos de *design* como fontes, cores, ícones e outros elementos da interface, garantindo que fossem consistentes e estivessem em conformidade com o planeado para o produto **idfyme**. Isto foi fundamental para manter a identidade visual da marca e para garantir uma experiência coesa e agradável para os utilizadores finais.

Para aumentar a clareza e intuitividade da aplicação foram ainda consideradas cores e ícones distintos associados a cada processo e usados de forma concistente em toda a aplicação.

- **Azul escuro e ícone de câmera:** Processos de videoconferência.
- **Roxo e ícone de documento com caneta:** Processos de assinatura de documentos.
- **Azul Claro e ícone com impressão digital:** Processos de dados biométricos.

O mesmo critério foi aplicado aos estados dos processos, com cores específicas designadas para cada estado:

- **Azul:** Processos pedentes de ações por parte do funcionário.
- **Amarelo:** Processos pedentes de ações por parte do utilizador
- **Verde:** Processos já validados e conluídos.
- **Vermelho:** Processos rejeitados ou cancelados.

Também foi levado em consideração um *design* simples que fosse compatível tanto com o estilo do Android quanto do iOS. Dado os seus distintos sistemas operativos, o fluxo de uma aplicação móvel no Android apresenta diferenças em comparação com uma aplicação móvel no iOS. Para lidar com esta disparidade, um dos aspectos considerados foi a inclusão de uma seta de retorno às páginas anteriores, uma vez que as aplicações móveis no iOS não possuem botões integrados dedicados para essa funcionalidade [37], ao contrário do Android.

Além disso, priorizou-se a criação de um *design* elegante, simples e consistente, características essenciais para uma UI no iOS, mas támbem prioritário num *design* para Android [38]. Para isso, as páginas foram desenvolvidas com base em elementos simples, uniformes e retângulares, mantendo o foco nas funcionalidades principais da aplicação [36].

4.5 Adequação da Solução

A solução desenvolvida nesta tese respondeu então ao problema do produto **idfyme** criando uma versão móvel para iOS e integrando-a de forma unificada e multiplataforma com a versão já existente para Android. A implementação desta versão unificada (partilham o mesmo projeto com a mesma base de código) do **idfyme** para os sistemas operativos Android e iOS abordou eficazmente os desafios iniciais relacionados com a falta de inclusão digital do produto e da fragmentação entre ambas as plataformas. Esta abordagem permitiu oferecer uma experiência de utilizador coesa e consistente, independentemente do dispositivo utilizado. Ao consolidar todas as funcionalidades numa base de código única, não só se otimizou o tempo e os recursos necessários para futuras manutenções e atualizações, como também se garantiu uma experiência mais uniforme e fluida para os utilizadores.

Desta forma, a solução implementada integrou todas as funcionalidades já disponibilizadas pelo produto, respondendo às necessidades de autenticação remota para a realização de serviços oficiais. Ela proporciona comunicação direta e em tempo real entre funcionários e clientes através de videoconferências, tanto para esclarecimento de dúvidas quanto para a extração de dados biométricos necessários à emissão de documentos de identificação. Adicionalmente, a solução permite a recolha de dados biométricos, como fotos da face e assinaturas manuscritas digitais, bem como a assinatura de documentos digitais, com a opção de o utilizador escolher o local da assinatura no documento.

Adicionalmente, a introdução de medidas de segurança robustas, como o uso de um PIN de 4 a 8 dígitos, da autenticação biométrica e do uso de validações de ambiente e de *token* antes da extração biométrica nas videoconferências, contribuiu para elevar o nível de confiança e proteção nos processos realizados na aplicação. Adicionalmente, a implementação de um *design* responsivo e intuitivo adaptado a ambos os sistemas operativos também garantiu uma experiência de utilizador eficiente.

Além disso, a aplicação foi desenvolvida com compatibilidade limitada às versões mais recentes e usadas dos sistemas operativos iOS (13.0 ou superior) e Android (6.0 ou superior), de modo a equilibrar a complexidade do desenvolvimento com a viabilidade técnica e a acessibilidade dos utilizadores.

Assim, a solução apresentada neste capítulo não só cumpre com os objetivos iniciais de otimização e expansão da plataforma **idfyme**, como também se posiciona como uma ferramenta capaz de responder às necessidades emergentes de autenticação remota e digital, facilitando o acesso e a interação dos utilizadores com os serviços prestados. O sucesso desta implementação abre portas para a continuidade do desenvolvimento de funcionalidades adicionais, mantendo sempre como prioridade a acessibilidade, segurança, manutenção e eficiência da aplicação.

5

Implementação da Solução

Como mencionado anteriormente na secção 3.3, o produto **idfyme** é dividido em três projetos distintos: o *idfyme-android* que contém a aplicação móvel Android já desenvolvida anteriomente no produto, o *idfyme-flutter* que contém a solução móvel multiplataforma, e o *idfyme-plugins* que engloba toda a lógica do *backend* do produto já desenvolvido.

Neste capítulo, são detalhadas as implementações gerais realizadas em cada um dos projetos envolvidos na solução: o *idfyme-flutter*, a aplicação móvel que constitui a solução, e o *idfyme-plugins*, que interage diretamente com ela. São também explicadas as tecnologias e ferramentas utilizadas, bem como abordadas as dificuldades encontradas durante o processo de implementação e as técnicas adotadas para as superar.

5.1 Projeto *idfyme-flutter*

A solução desta tese é composta pelo projeto *idfyme-flutter*, responsável pelo desenvolvimento da aplicação móvel multiplataforma em Flutter. Este projeto visa substituir a versão anterior do produto, disponível apenas para Android, integrando também a versão para iOS num único projeto com base de código partilhada entre ambos os sistemas operativos.

Esta solução em Flutter foi desenvolvida com base na arquitetura Model-View-ViewModel (MVVM) com a adição de uma camada de serviço extra, os **Services**, de modo a tornar a solução ainda mais modular, promovendo a sua clareza, escalabilidade, manutenção, flexibilidade e fiabilidade.

O padrão MVVM, ilustrado na figura 5.1, é uma arquitetura de desenvolvimento de *software* que divide a lógica de funcionamento em três componentes distintos: o **Model**, responsável por representar e encapsular os dados da aplicação; a **View**, encarregada de exibir a UI e interagir com o utilizador; e o **ViewModel**, que atua como intermediário, facilitando a comunicação e partilha de dados entre o Model e a View [39].

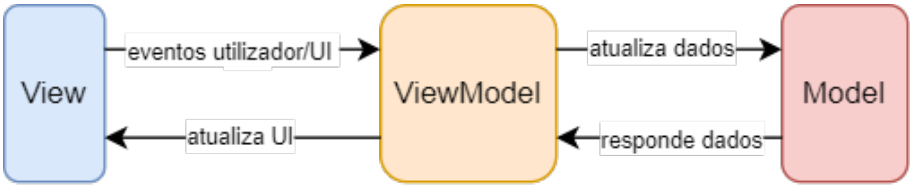


Figura 5.1: Diagrama da arquitetura MVVM

Assim, a arquitetura da solução implementada, ilustrada na figura 5.2, é composta por estas diferentes camadas interconectadas. As **Views** representam a UI, sendo compostas por *widgets* que apresentam os dados no ecrã e lidam com as interações do utilizador. Os **Models** encapsulam os dados da aplicação, assegurando a sua integridade e consistência. Os **ViewModels** agem como intermediários entre as Views e os Models, coordenando a lógica de apresentação dos dados exibidos nas Views e oferecendo capacidades de ligação para manipulação dos dados encapsulados nos Models. Por fim, os **Services**, a camada de serviço extra implementada, abstraiem a comunicação da aplicação para com sistemas externos ou APIs, cuidando da recuperação e manipulação dos dados externos para a aplicação.

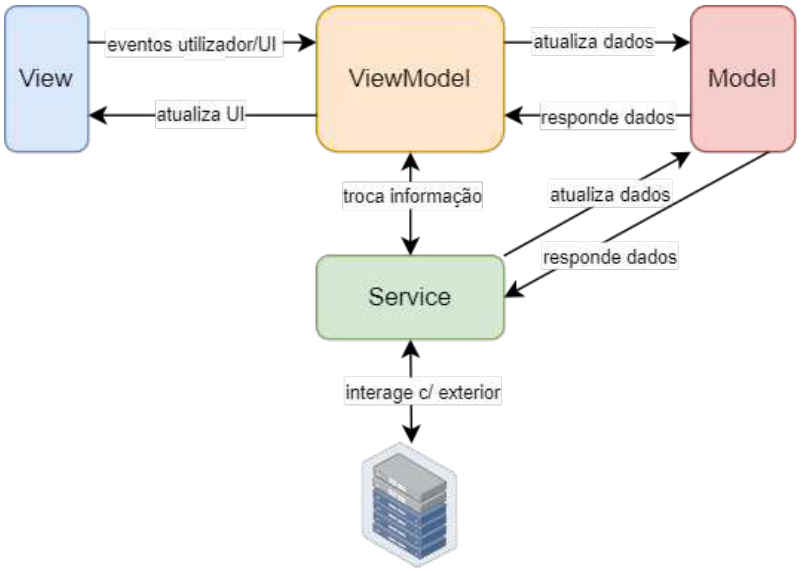


Figura 5.2: Diagrama da arquitetura MVVM com Services implementada na solução

O padrão MVVM com a adição dos Services oferece uma série de vantagens no desenvolvimento da solução. Ele promove uma separação clara das preocupações (*separation of concerns*), através do Model, View, ViewModel e Service, cada um a representar funcionalidades distintas da lógica da aplicação. Esta separação melhora a capacidade de manutenção e teste, permitindo o desenvolvimento independente em aspectos específicos da aplicação. Além disso, torna a solução mais modular e escalável, já que os componentes podem ser substituídos ou estendidos facilmente sem afetar a restante aplicação. Este padrão também fomenta a reutilização do código, uma vez que os Services podem ser

partilhados por vários ViewModels, resultando num desenvolvimento mais eficiente e numa manutenção mais fácil das bases do código [39].

De seguida serão descritas as implementações de algumas características relevantes deste projeto.

5.1.1 Internacionalização

De modo a tornar a aplicação internacional foi usado a classe `AppLocalizations` do pacote `flutter-localization`[1]. Esta classe fornece acesso a *strings* (sequência de caracteres usada para representar texto) definidas em ficheiros locais que são acedidos pelas diferentes classes para vários elementos da UI, como etiquetas de texto, botões e mensagens de sucesso ou erro. Cada ficheiro contém as mesmas *strings* traduzidas para um idioma específico, permitindo assim que a aplicação consiga adaptar automaticamente o idioma preferido do utilizador.

Para a implementação da solução foram realizados dois ficheiros distintos, um para o idioma inglês e outro para o idioma português, sendo o idioma português, por decisão da empresa, o idioma padrão da aplicação.

Esta implementação permite que a aplicação ajuste dinamicamente o seu conteúdo para corresponder às preferências de idioma de cada utilizador, proporcionando uma experiência mais inclusiva. Além disso, simplifica o processo de gestão e organização de recursos localizados, facilitando a manutenção e atualização do suporte linguístico da aplicação móvel ao longo do tempo [40].

5.1.2 Tema

Para garantir uma experiência do utilizador consistente e agradável, foi desenvolvida uma classe exclusiva destinada a conter um tema personalizado para a aplicação móvel, seguindo as diretrizes de *design* do produto **idfyme**.

Esta abordagem visa aprimorar a experiência do utilizador e reforçar a identidade da marca, proporcionando uma aparência visual coesa. Possibilita por isso a manutenção de uma identidade visual consistente em toda a aplicação, por meio da definição de cores personalizadas, tipografia e outros elementos de *design*. Além disso, uma classe de tema personalizada promove a reutilização e a modularidade do código, ao centralizar a configuração das propriedades visuais num único sítio. Isto simplifica o processo de atualização do visual da aplicação, pois as alterações podem ser feitas num único local, em vez de precisar de serem replicadas em diversas ocorrências ao longo da aplicação [41].

Esta abordagem otimiza então o desenvolvimento e a manutenção da solução, garantindo uma experiência de utilizador mais consistente e coesa [36].

[1] https://pub.dev/packages/flutter_localization

5.1.3 Design Responsivo

Outro aspeto desenvolvido de forma a garantir uma experiência do utilizador consistente e agradável foi a implementação de uma UI que se adaptasse eficazmente aos diferentes tamanhos de ecrã dos dispositivos móveis e diferentes orientações (vertical e horizontal) como ilustrado nas figuras 5.3 e 5.4, respetivamente. Para tal, foram aplicadas práticas de *design* responsivo, com especial foco na utilização de *widgets* expansíveis, textos que se ajustam dinamicamente e páginas deslizáveis verticalmente. Desta forma, garantiu-se que a aplicação móvel mantivesse a sua usabilidade e legibilidade, independentemente do dispositivo utilizado [36].

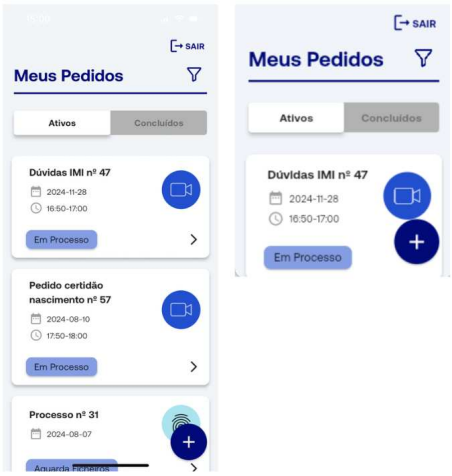


Figura 5.3: Página inicial em dois tamanhos de ecrãs distintos

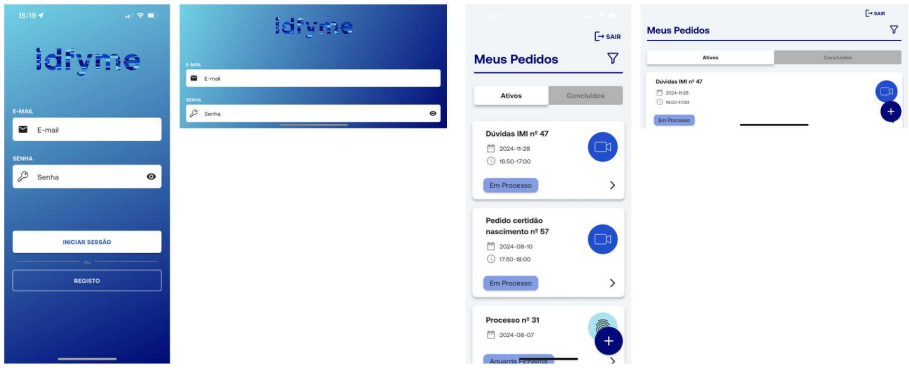


Figura 5.4: Páginas da aplicação móvel em orientações vertical e horizontal

5.1.4 Feedback da Aplicação

Para melhorar a experiência do utilizador, foi implementado um *feedback* visual claro em resposta às suas ações. Esta abordagem incluiu o uso de rótulos informativos, barras de notificação e *popups* com mensagens de sucesso ou erro. Além disso, foram realizadas alterações visuais, como a mudança de cor dos campos de entrada em caso de erros, conforme ilustrado na figura 5.5. Estas práticas foram adotadas para fornecer uma orientação eficaz

e um *feedback* claro durante a interação com a aplicação, contribuindo significativamente para uma experiência mais intuitiva e informativa [36].

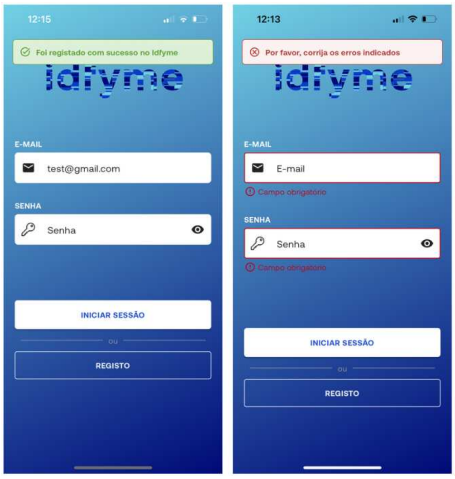


Figura 5.5: Páginas de autenticação do utilizador com mensagens de sucesso de registo e de erro no *login*

5.1.5 Estado da Aplicação

De forma a controlar o comportamento da aplicação de uma forma mais eficiente e organizada foi também implementado o uso de `Providers`, com a complementariação do uso de `ChangeNotifierProvider` e `MultiProvider`, no projeto *idfyme-flutter*.

`Provider`[2] [42] é uma biblioteca do Flutter que ajuda na gestão do estado da aplicação, facilitando a partilha de dados entre diferentes *widgets*. Ela permite expor um valor a todos os *widgets* na árvore de *widgets* abaixo dele, permitindo que estes recebam as mudanças neste valor e se reconstruam de acordo com essa informação.

O `ChangeNotifierProvider` é um tipo específico de `Provider` que funciona bem com classes que estendem o `ChangeNotifier`. O ChangeNotifier[3] é uma classe fornecida pelo Flutter que permite notificar os seus ouvintes (outras classes) quando o objeto for alterado. Assim, o `ChangeNotifierProvider` é usado para expor o estado de um objeto `ChangeNotifier` aos seus descendentes. Sempre que o objeto `ChangeNotifier` for alterado, ele notifica os seus ouvintes. Estes ouvintes são classes cuja lógica depende do estado desse objeto e que precisam de ser atualizadas quando ele muda de estado. Isto faz com que os *widgets* dependentes sejam reconstruídos conforme a nova lógica dos ouvintes, mantendo a aplicação atualizada de forma assíncrona, sem necessitar que o utilizador faça *refresh* da página.

Esta abordagem foi então implementada na classe raiz da aplicação (`main`), para que todas as classes que precisavam de notificar os seus descendentes sobre alterações fossem

[2] https://pub.dev/packages/provider
[3] https://api.flutter.dev/flutter/foundation/ChangeNotifier-class.html

capazes de o fazer. Para lidar com a necessidade de usar vários `ChangeNotifierProviders` em diferentes partes da aplicação, foi igualmente adotado o `MultiProvider`, que permite combinar diversos `Providers` num único *widget*, neste caso, na `main`.

Ao estabelecer esta camada de serviço de atribuição de `Providers` no nível mais alto da aplicação para as diferentes classes que o requerem, conseguiu-se organizar melhor o código e simplificar sua gestão, assim como a gestão do estado da aplicação, de forma mais eficiente. Isto ainda garantiu que os *widgets* fossem reconstruídos automaticamente sempre que o estado da aplicação fosse alterado, sem a necessidade de uma implementação mais detalhada e profunda e com tempos de espera menores para o utilizador [42].

5.1.6 *Loading*

Para melhorar a experiência do utilizador e fornecer *feedback* visual durante o carregamento de dados, foi implementado um indicador de `loading` na solução. Este indicador é exibido sempre que a aplicação processa um pedido HTTP ao projeto *idfyme-plugins*, sendo gerido através da classe `LoadingState`, que implementa o `ChangeNotifier`, explicado anteriormente.

A classe `LoadingState` altera o seu estado para `Loading` ao iniciar um carregamento, com base num enumerado que contém os valores `Loading`, `Sucesso`, `Erro` e `Nenhum`. Esta mudança de estado notifica os seus ouvintes, neste caso, as páginas que realizam estes pedidos HTTP, para que sejam reconstruídas e exibam o indicador de carregamento. Quando a resposta é recebida, o estado volta a `Nenhum`, e as páginas são novamente notificadas para serem reconstruídas com os dados obtidos, removendo o indicador de carregamento.

Além disto, a `LoadingState` permite notificar se ocorreram erros durante o processo, alterando o estado para `Erro`, ou para `Sucesso`, caso tudo tenha corrido como esperado.

Desta forma, esta classe foi amplamente utilizada em toda a aplicação, nomeadamente na busca dos processos na página inicial, no carregamento de documentos PDF para visualização durante os processos de assinatura de documentos, e na submissão e validação dos dados de extração biométrica e assinatura de documentos, conforme ilustrado na figura 5.6.

Esta abordagem trouxe várias vantagens significativas. Em primeiro lugar, melhorou a experiência do utilizador ao fornecer *feedback* visual claro durante o carregamento de dados, evitando situações de incerteza ou confusão sobre o estado da aplicação. Além disso, garantiu-se uma atualização eficiente da interface, permitindo que apenas as páginas relevantes fossem reconstruídas, economizando recursos e melhorando o desempenho. A possibilidade de exibir mensagens de erro ou sucesso também facilitou a comunicação de problemas ou confirmações, tornando a aplicação mais intuitiva e responsiva às ações do utilizador [36].

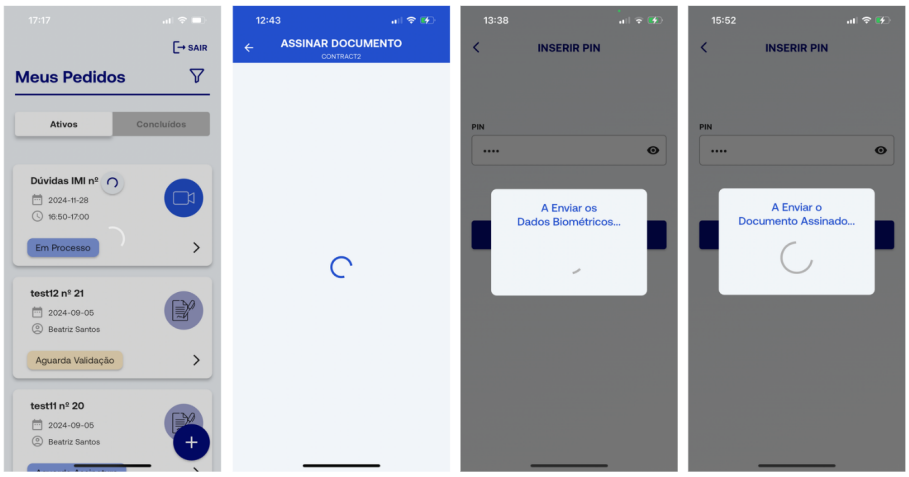


Figura 5.6: Páginas com o indicador de carregamento implementado na solução

5.1.7 Service Locator

Para gerir a criação e a recuperação de objetos (serviços) na aplicação, foi implementado o padrão de *design* Service Locator (localizador de serviços) numa classe dedicada, através do uso do pacote Flutter `get-it`[4]. Este padrão oferece uma abordagem centralizada para aceder a serviços sem a necessidade de os instanciar diretamente ou passá-los através de construtores, simplificando assim a gestão e a obtenção das dependências na aplicação [43].

Com o `get-it`, as dependências podem ser registadas utilizando métodos como `registerFactory`, `registerSingleton`, ou `registerLazySingleton`. O `registerFactory` é usado para gerar uma nova instância de uma dependência sempre que é requisitada, sendo útil para objetos que não devem ser partilhados entre diferentes partes da aplicação. O `registerSingleton` cria uma única instância de uma dependência sendo reutilizada sempre que é solicitada, garantindo assim a sua consistência na aplicação. Por fim, o `registerLazySingleton` cria uma única instância apenas quando ela é solicitada pela primeira vez, otimizando o desempenho ao atrasar a inicialização até ser mesmo necessária.

Ao utilizar o Service Locator, a solução conseguiu dissociar os diferentes serviços, facilitando a sua manutenção e testabilidade. Além disso, ele promoveu a reutilização e a modularidade, permitindo que os serviços fossem facilmente trocados ou substituídos em qualquer parte do código, sem a complexidade de os gerir manualmente e individualmente [43].

5.1.8 Navegação e Rotas

A navegação entre as diferentes páginas da aplicação móvel foi implementada através da classe `NavigationService`, para gerir a navegação, e do método `generateRoute` para gerir as rotas de navegação.

[4] https://pub.dev/packages/get_it

A classe `NavigationService` encapsula a lógica de navegação da aplicação por meio da biblioteca `material`[5] do Flutter. Ela oferece acesso ao estado de navegação da aplicação em qualquer parte do código, permitindo a navegação entre diferentes rotas de forma flexível. Além disso, possibilita o uso de parâmetros opcionais, como limpar a pilha de navegação, retroceder um número específico de páginas atrás, verificar qual página está no topo da pilha (a página atualmente exibida), ou até tornar a barra superior da aplicação transparente.

Por sua vez, o método `generateRoute` define como gerar rotas dinamicamente para as diferentes páginas da aplicação, também utilizando a biblioteca `material`[6]. Ele também inclui o tratamento de erros para rotas que não são definidas, garantindo uma experiência de navegação mais robusta e livre de problemas.

Estas abordagens combinadas proporcionaram uma estrutura sólida para a navegação dentro da solução, facilitando a organização e o controlo das transições entre as diferentes páginas. Ao utilizar estas estruturas a aplicação consegue responder de forma dinâmica e eficiente aos pedidos de navegação do utilizador, proporcionando uma experiência mais fluida [44].

5.1.9 Notificador de Orientação

Algumas funcionalidades e páginas da aplicação requerem uma orientação específica do ecrã do dispositivo móvel. Por exemplo, a página de extração da assinatura necessita que o ecrã esteja na horizontal, e a página da escolha do local para assinar o documento requer que o ecrã esteja na vertical. Devido a estas exigências, tornou-se necessário impor uma orientação específica para cada uma destas páginas, utilizando a função `setPreferredOrientation` da biblioteca `services`[7] do Flutter, que recebe e aplica a orientação pretendida no dispositivo móvel.

O desafio surge quando o utilizador navega destas páginas para outras, que já não possuem estas restrições de orientação. Nestes casos, as restrições de orientação aplicadas anteriormente permanecem, mesmo que já não sejam necessárias, afetando a usabilidade.

Para resolver este problema, foi implementado um sistema de notificação de orientação, que utiliza o `ValueNotifier` da biblioteca `foundation`[8] do Flutter para cada uma das páginas envolvidas, tanto as que precedem como as que sucedem as páginas com restrições de orientação.

As páginas que precisam de modificar uma orientação previamente imposta subscrevem-se então a este notificador, mantendo-se atentas a eventuais alterações. Assim que o utilizador navega para uma destas páginas, detetado pelo `NavigationService` conforme explicado anteriormente, o notificador `ValueNotifier` é informado, e a página

[5] https://api.flutter.dev/flutter/material/material-library.html
[6] https://api.flutter.dev/flutter/material/material-library.html
[7] https://api.flutter.dev/flutter/services/services-library.html
[8] https://api.flutter.dev/flutter/foundation/foundation-library.html

correspondente é notificada da mudança. Após ser notificada, a página ajusta automaticamente a sua orientação para quebrar as restrinções impostas e retorna as orientações normais.

Esta abordagem oferece várias vantagens como garantir uma experiência de utilizador mais fluída, uma vez que a orientação do ecrã é ajustada de forma dinâmica e automática conforme as necessidades de cada página. Além disso, ao se utilizar um sistema de notificação de orientação, o código torna-se mais modular e organizado, já que cada página lida com as suas próprias restrições de forma independente. Esta independência reduz o acoplamento entre as páginas, facilitando a manutenção e futuras expansões da aplicação [36].

5.1.10 Permissões da Aplicação

Algumas funcionalidades, como a videoconferência e a extração biométrica, requerem permissões específicas do dispositivo móvel, nomeadamente o acesso à câmara e ao microfone. Para gerir estas permissões, foi implementada uma função que, antes de o utilizador aceder a estas funcionalidades, verifica se as permissões necessárias foram concedidas ou, no caso de ser a primeira vez a acede-las, faz o seu respetivo pedido.

Esta função utiliza o *plugin* do Flutter `permission-handler`[9], que oferece uma API multiplataforma para solicitar e verificar o estado das permissões nos dispositivos móveis.

Se o utilizador não conceder as permissões necessárias e tentar aceder às funcionalidades, a aplicação exibe uma mensagem de erro a informá-lo da necessidade de autorizar as permissões para utilizar a funcionalidade, como ilustrado na figura 5.7.

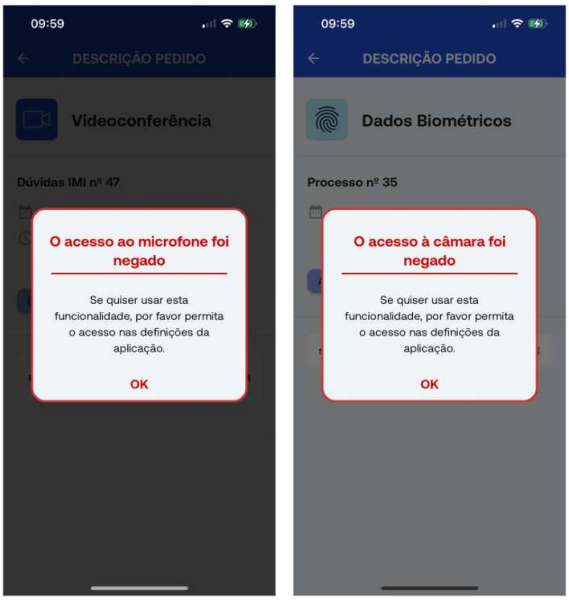


Figura 5.7: *Popup* com a mensagem de erro das permissões

[9] https://pub.dev/packages/permission_handler

Esta abordagem assegura assim que a aplicação acede apenas às funcionalidades previamente autorizadas pelo utilizador, respeitando as suas escolhas individuais no que diz respeito à privacidade e evitando acessos não autorizados a recursos sensíveis do dispositivo. Ao proteger o acesso a estas funcionalidades, a solução implementada reforça a segurança dos dados pessoais do utilizador, minimizando potenciais vulnerabilidades associadas ao uso indevido ou não consentido de dados sensíveis [45].

Adicionalmente, esta estratégia promove a transparência, uma vez que o utilizador é notificado claramente sobre as permissões requeridas e os motivos pelos quais são necessárias. Assim, garante-se o cumprimento das melhores práticas de privacidade de dados e regulamentações vigentes na União Europeia, como o Regulamento Geral sobre a Proteção de Dados (GDPR)[10], assegurando que a aplicação cumpre os requisitos legais e éticos para a proteção da privacidade e segurança dos dados dos utilizadores [45].

5.1.11 Comunicação com API Rest

Para a implementação dos pedidos HTTP na aplicação móvel, foi desenvolvida uma camada de serviço base (`ApiService`) que lida com a gestão dos pedidos HTTP para a comunição entre a aplicação móvel e a API do projeto *idfyme-plugins*. Esta camada é responsável por criar os pedidos HTTP bem como processar e manipular as respostas recebidas usando a biblioteca do Flutter `http`[11] e `io`[12]. Isto inclui a conversão dos dados recebidos no formato JavaScript Object Notation (JSON), através do uso das bibliotecas `Japx`[13] e `convert`[14], para um formato que se alinhe com a lógica da aplicação, viabilizando a sua utilização de forma eficaz, como, por exemplo, na adaptação dos dados para o seu armazenamento na base de dado local do dispositivo móvel. Além disso, esta camada também aborda a gestão dos erros e exceções a serem lançadas após a chegada da resposta do pedido.

Foi também desenvolvida uma camada de serviço adicional, a `ApiRequest`, responsável pela construção da informação dos pedidos HTTP, como os seus parâmetros, cabeçalhos, dados e Uniform Resource Identifier (URI), com suporte da biblioteca `core`[15]. Posteriormente, esta camada é utilizada pelo serviço `ApiService`, que a usa para construir a estrutura do pedido `HTTP` (compondo-o como sendo um pedido `GET`, `PUT`, `POST` ou `DELETE`) para depois ser enviado.

Portanto, estas duas camadas de serviços adicionais permitiram a reutilização do código, simplificando a construção dos pedidos e promovendo a manutenção e a evolução da aplicação [41].

[10] https://eur-lex.europa.eu/eli/reg/2016/679/oj
[11] https://pub.dev/packages/http
[12] https://pub.dev/packages/io
[13] https://pub.dev/packages/japx
[14] https://api.flutter.dev/flutter/dart-convert/dart-convert-library.html
[15] https://api.flutter.dev/flutter/dart-core/dart-core-library.html

5.1.12 Exceções

Para implementar o tratamento de exceções na solução, foram criadas exceções específicas para cada código de erro que podia ser lançada no produto **idfyme**. Cada uma destas exceções é então uma extensão de uma exceção básica criada, denominada `NetworkException`, que inclui um código de erro, uma mensagem de erro com título e detalhes, e um estado do erro.

Assim quando ocorre um erro num pedido HTTP na aplicação móvel, é lançada uma exceção correspondente. Essa exceção, através de um serviço dedicado denominado `ApiExceptionService`, executa diversas ações, como exibir notificações de erro ao utilizador para informá-lo sobre o problema encontrado, ou redirecionar o utilizador para a página de *login* caso a sua sessão tenha expirado, entre outras possibilidades.

Desta forma foi possível fornecer *feedbacks* claros e detalhados ao utilizador, informando-o do erro ocorrido e as medidas necessárias para o resolver. Esta abordagem contribuiu para uma experiência do utilizador aprimorada e uma maior eficiência e usabilidade do código promovendo a sua reutilização eficiente [46].

5.1.13 Logs

Para facilitar a depuração e o diagnóstico de erros na solução, foi ainda implementado um sistema de registo de eventos (`LoggerService`) que implementa o registo e exibição de mensagens de depuração, informações, avisos e erros (Logs). Este sistema é baseado no pacote `logger`[16] do Flutter, que fornece métodos para registar mensagens de diferentes níveis de gravidade, como `verbose`, `debug`, `info`, `warning`, `error` e `critical`.

Isto permitiu uma monitorização e análise abrangentes do comportamento da aplicação em tempo de execução, que ajudou na identificação de problemas durante as fases de desenvolvimento e teste e tornou mais eficiente a implementação da solução [47].

5.1.14 Base de Dados Local

Na aplicação móvel, foram implementados dois tipos de bases de dados locais: uma destinada a armazenar informações sensíveis do utilizador, como o *token* de acesso e de atualização, o PIN e a senha de acesso do *login*; e outra para guardar outras informações mais estruturadas do utilizador, como o seu Identifier (ID), e-mail, nome, estado de *login*, tipo de utilizador (cliente ou funcionário), se tem autenticação biométrica, entre outras informações relevantes.

Este processo otimiza o desempenho da aplicação móvel por disponibilizar o acesso da aplicação a dados frequentemente utilizados no seu fluxo [48], como por exemplo, proporcionando a conveniência de evitar que o utilizador faça *login* de cada vez que abra a aplicação móvel.

[16] `https://pub.dev/packages/logger`

Para a base de dados que guarda informações sensíveis do utilizador, foi utilizado o *plugin* `flutter-secure-storage`[17] do Flutter, que oferece uma solução de armazenamento seguro e criptográfico para informações sensíveis. Este *plugin* permite armazenar dados de forma segura nos dispositivos móveis, protegendo-os contra acessos não autorizados e adulterações. Além disso, o `flutter-secure-storage` oferece métodos para armazenar e recuperar informações de forma eficiente para iOS e Android, garantindo a segurança e a privacidade dos dados do utilizador.

Para a base de dados que guarda informações estruturadas do utilizador, foi utilizado a biblioteca `floor`[18] do Flutter, que fornece uma camada de abstração SQLite fácil de utilizar e eficiente. O `floor` permite definir modelos de dados e interagir com a base de dados local dos dispositivos móveis utilizando paradigmas de programação orientados a objetos de uma forma simples e intuitiva.

Ao implementar as bases de dados locais a partir destes componentes do Flutter, a solução foi capaz de ser desenvolvida com um armazenamento mais robusto e com uma maior segurança dos dados, seguindo as melhores práticas de gestão de dados e proteção da privacidade. Isto resultou num aumento de desempenho, proporcionando uma experiência de utilizador mais rápida e eficiente. Ela também é capaz de operar em modo *offline*, uma vez que está localizada dentro dos dispositivos móveis, proporcionando a vantagem de garantir a segurança da autenticação mesmo quando não exista conexão à internet. Estes benefícios contribuiram para a qualidade e a usabilidade geral da solução desenvolvida [48].

5.2 Projeto *idfyme-plugins*

O projeto *idfyme-plugins* engloba toda a lógica do *backend* do produto sendo o responsável por lidar com os pedidos HTTP [49] recebidos tanto da aplicação móvel quanto da aplicação *web* e fazer a sua lógica de negócios comunicando com a base de dados e gerindo a lógica do servidor de sinalização.

Sempre que a aplicação móvel precisa de criar, obter, modificar ou eliminar dados, envia pedidos HTTP para os *endpoints* criados no projeto *idfyme-plugins*. Estes *endpoints* processam os pedidos e mandam uma resposta à aplicação móvel. Esta resposta pode ser uma mensagem de sucesso ou de erro, dependendo do resultado do pedido. Além disso, é associado a cada resposta um código do seu estado, que pode ser [49]:

- Respostas Informativas (100 - 199)
- Respostas de sucesso (200 - 299)
- Mensagens de redirecionamento (300 - 399)

[17] https://pub.dev/packages/flutter_secure_storage
[18] https://pub.dev/packages/floor

- Respostas de erro do cliente (400 - 499)
- Respostas de erro do servidor (500 - 599)

Os pedidos HTTP do produto seguem o formato JSON, um formato leve e simples para troca de dados entre um servidor e aplicação baseado num subconjunto da linguagem de programação JavaScript [50].

A utilização de JSON nos pedidos e respostas HTTP, especialmente quando mensagens de erro são fatores essenciais, oferece inumeras vantagens em comparação com outros formatos, como o XML ou o HTML. A principal vantagem é o facto de ser mais leve e legível, o que facilita a análise e a compreensão tanto por parte dos programadores como dos sistemas. A sua simplicidade também facilita o desenvolvimento, uma vez que os erros podem ser rapidamente identificados e resolvidos. Além disso, a estrutura hierárquica do JSON permite a inclusão de objectos de dados complexos, fornecendo informações detalhadas sobre erros, como códigos de erro, títulos, descrições e contexto adicional, que podem ajudar na resolução dos erros. Além disso, o JSON é amplamente suportado em diferentes linguagens de programação e plataformas, garantindo a interoperabilidade e a compatibilidade com várias aplicações e sistemas [50].

A implementação das funcionalidades na solução incluiu igualmente a criação de novos *endpoints* específicos no projeto *idfyme-plugins*, os quais serão detalhados nas secções seguintes.

5.2.1 *Endpoints* criados

Para a implementação das funcionalidades da aplicação móvel, foram criados três novos *endpoints*, considerados como uma segunda versão do projeto:

- *.../o/biocid-rest/service/api/biocid/rest-api/v2/users/register* - Realiza o registo do utilizador no produto **idfyme** recebendo os dados inseridos na página de Registo da aplicação móvel.
- *.../o/biocid-rest/service/api/biocid/rest-api/v2/users/user-info* - Após receber o *token* de acesso e o *token* de atualização (*refresh-token*) do utilizador, este *endpoint* disponibiliza informações essenciais do utilizador, como o seu ID, primeiro e último nome, e se é cliente ou funcionário. Estes dados são então armazenados na base de dados local dos dispositivos móveis que acedem à aplicação móvel.
- *.../o/biocid-rest/service/api/biocid/rest-api/v2/video-conference/add-vc* - Adiciona uma nova sessão de videoconferência para o utilizador registado no momento, após receber o *token* de acesso e as informações da nova videoconferência.

Foi necessário desenvolver um novo *endpoint* para o registo do utilizador, uma vez que a aplicação móvel necessitava de um tipo específico de resposta de erros, do qual o *endpoint* existente utilizado pela aplicação *web* e Android antiga, não suportava.

Foi também necessário criar o *endpoint user-info* para que a aplicação móvel pudesse aceder aos dados do utilizador e armazená-los na base de dados local dos dispositivos. Este *endpoint* não havia sido implementado anteriormente, já que a aplicação *web* não fazia uso desta base de dados local.

Para implementar o *endpoint* de criação de videoconferências na aplicação móvel, foi necessário desenvolver uma nova versão do existente, uma vez que o anterior não possuía uma lógica adequada para a solução, incluindo validação e verificação de dados inseridos pelo utilizador limitadas, não sendo por isso seguras.

Os restantes *endpoints* foram utilizados sem quaisquer alterações, uma vez que a aplicação móvel criada implementava funcionalidades já desenvolvidas pela aplicação *web* e pela antiga aplicação móvel Android, não sendo necessárias modificações ou otimizações adicionais.

5.2.2 Gestão das Mensagens de Erro

Grande parte da lógica de validação dos dados inseridos e submetidos pelo utilizador está integrada no projeto *idfyme-plugins*, aproveitando as vantagens de centralizar esta lógica nos *endpoints*. Desta forma, foi essencial implementar esta validação no projeto para os novos *endpoints* criados, incluindo a criação de mensagens de sucesso e mensagens de erro específicas para cada situação em que existiam dados inseridos de maneira incorreta.

No entanto, existem algumas validações mais básicas a serem realizadas no lado do cliente (projeto *idfyme-flutter*), com o objetivo de aliviar a carga no servidor ao fazer uma filtragem inicial mais simples [51]. Exemplos dessas validações incluem verificar se os campos de introdução de dados estão corretamente preenchidos de uma forma mais geral, ou seja, se não estão vazios, se não possuem caracteres inválidos, e se o email e data têm um formato correto.

Realizar uma parte da validação dos dados no lado do servidor, especificamente no projeto *idfyme-plugins*, ofereceu várias vantagens significativas ao produto em comparação com a validação exclusiva no lado do cliente (projeto *idfyme-flutter*). Ao realizar a validação na última camada de acesso à base de dados (servidor), garantiu-se uma maior segurança em relação aos dados a serem guardados. Esta abordagem também permitiu centralizar a lógica de validação num único local, simplificando a manutenção e a evolução da solução, garantindo o mesmo nível de validação tanto para a aplicação *web* quanto móvel que partilham os mesmos *endpoints*. Além disso, proporcionou uma experiência do utilizador aprimorada com mensagens de erro mais detalhadas e específicas para cada tipo de erro cometido. A validação no servidor também previne tentativas de burlar as verificações realizadas no lado do cliente, reduzindo potenciais riscos à segurança do sistema [51].

Anteriormente, o projeto *idfyme* não possuía mensagens de erro específicas para cada situação de inserção incorreta de dados, apenas uma mensagem genérica de erro. Esta abordagem não era eficaz, pois o *frontend* não conseguia fornecer informações precisas sobre o erro ao utilizador, dificultando a compreensão e resolução do problema.

Portanto, foi implementada uma nova exceção no projeto *idfyme-plugins*, a `BadRequestErrorApiException`, que é lançada sempre que os dados inseridos pelo utilizador na aplicação forem incorretos. Esta exceção estende uma classe genérica JSON API (`RestApiErrorException`), implementada de raiz, que contém uma lista de erros. Cada erro segue um modelo, também desenvolvido de raiz, num formato JSON predefinido comum. Os erros, como o exemplo apresentado na figura 5.8, aparecem então numa lista em JSON cada um contendo o código do erro, o parâmetro de entrada que o causou, um título e uma mensagem detalhada e a localização do erro.

```
{"errors": [
    {
        "status": "400",
        "code": "nome",
        "title": "Validacao de dados falhada",
        "detail": "Obrigatorio",
        "source": {
            "pointer": "apelido",
            "parameter": "apelido"
        }
    },
    {
        "status": "400",
        "code": "dataNascimento",
        "title": "Validacao de dados falhada",
        "detail": "A data introduzida tem de ser anterior a data atual.",
        "source": {
            "pointer": "dataNascimento",
            "parameter": "dataNascimento"
        }
    }
],
"data": null,
"jsonapi": {
    "version": "2.0.0"
}}
```

Figura 5.8: Exemplo de uma lista de erros de resposta de um pedido ao *endpoint* de registo.

A exceção é então lançada no projeto *idfyme-plugins* sempre que os dados inseridos pelo utilizador estão incorretos, seja devido a campos vazios, caracteres inválidos, duplicidade de dados na base de dados, datas inválidas, entre outros. Assim, num pedido que envolve vários campos de dados inseridos pelo utilizador, como no registo demonstrado na figura 5.9, a aplicação móvel não precisa de lidar com a validação dos campos diretamente. Basta enviar o formulário e, em seguida, lidar com a resposta, seja de sucesso ou falha, e tratar da lista de erros em JSON recebida. Esta lista é então convertida numa mensagem para o utilizador, informando-o sobre as correções necessárias a fazer.

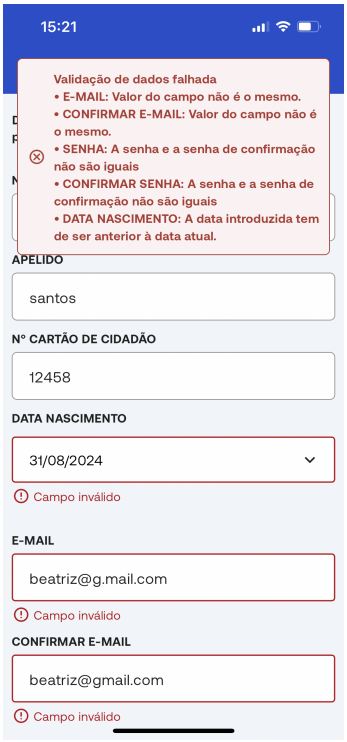


Figura 5.9: Mensagem de erros na página de registo

A implementação de um formato padrão para mensagens de erro traz vantagens para a solução como a promoção da clareza e da consistência na comunicação dos erros, tanto para o lado dos programadores para a sua melhor compreensão e identificação, como para o lado dos utilizadores da aplicação através de *feedbacks* mais claros e detalhados do erro cometido. Isto facilita em grande parte a interpretação e o tratamento destes erros pelos utilizadores, garantindo uma experiência mais consistente e fiável e contribuindo para uma melhor usabilidade e satisfação geral do produto [52].

5.3 Funcionalidades da Solução Desenvolvida

Como dito anteriormente, a solução móvel desenvolvida compreende o desenvolvimento de várias funcionalidades, já integradas e usadas na aplicação *idfyme-plugins*:

- **Autenticação do utilizador** perante dados de *login* e registo;
- **Realização do ecrã inicial da listagem dos processos** concluídos e por concluir, permitindo a consulta do histórico dos mesmos;
- **Realização e agendamento de videoconferências** com e sem extração biométrica;
- **Extração de dados biométricos** que incluem foto da face e assinatura manual;
- **Assinatura de documentos digitalmente** através de uma assinatura manual digital num local escolhido pelo utilizador no documento;

Nas secções seguintes, serão discutidas as implementações detalhadas dessas funcionalidades na solução.

5.3.1 Autenticação do Utilizador

A autenticação do utilizador é um procedimento vital no contexto da solução uma vez que é esta a responsável por verificar a identidade do utilizador. Este processo desempenha um papel fundamental na garantia da segurança e confidencialidade dos dados dos utilizadores, além de proteger a integridade do sistema como um todo [53].

A autenticação implementada, mostrada na figura 5.10, compreende quatro funcionalidades distintas:

- *Login*
- Registo
- PIN
- Autenticação Biométrica

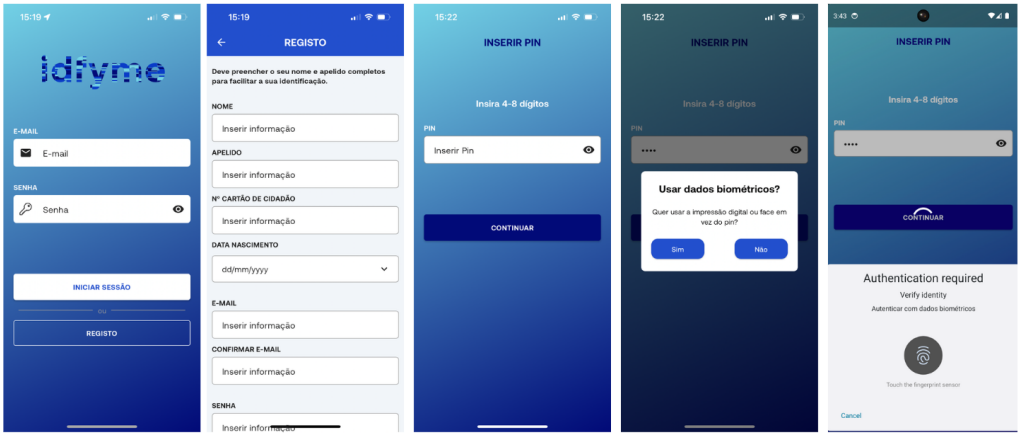


Figura 5.10: Páginas de *login*, Registo, PIN e Dados Biométricos da solução

5.3.1.1 Fluxo da Autenticação

O fluxo de autenticação do utilizador consiste em várias etapas, como ilustrado no diagrama de atividades apresentado na figura 5.11. Este diagrama delineia o processo de registo e entrada do utilizador na aplicação móvel, desde o *login* até à autenticação biométrica final, passando pelo registo de conta e pela criação ou inserção do PIN. Cada uma dessas etapas é destacada no diagrama de atividades por meio do uso de diferentes cores de modo a facilitar a compreensão e visualização dos diferentes fluxos dentro do processo de autenticação do utilizador. A cor laranja é utilizada para representar o *login*,

a cor azul para o registo, a cor cinzenta para o PIN, e a cor vermelha para a autenticação biométrica.

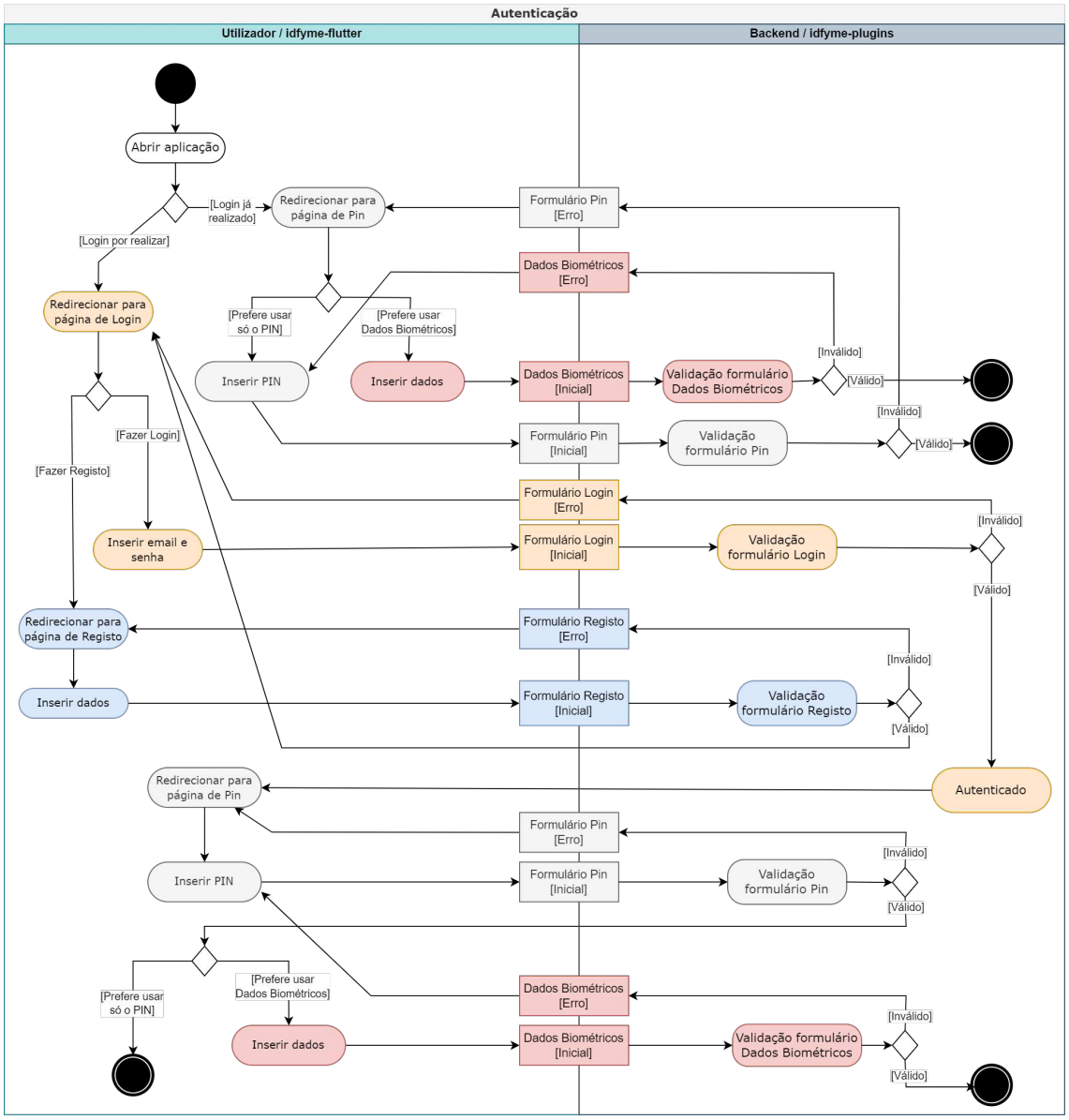


Figura 5.11: Diagrama de atividades da autenticação do utilizador

5.3.1.2 Autenticação Biométrica

Para a implementação da autenticação biométrica na autenticação do utilizador foi usado o *plugin* `local-auth`[19] do Flutter. Este *plugin* aproveita as capacidades de autenticação nativas das plataformas iOS e Android, garantindo uma experiência de utilizador consistente em todos os dispositivos. Ao abstrair as complexidades das APIs biométricas nativas, o *plugin* `local-auth` facilita significativamente o processo de integração, oferecendo uma

[19] https://pub.dev/packages/local_auth

autenticação biométrica rápida e segura por meio das impressões digitais ou do rosto do utilizador, previamente configurados nos seus dispositivos móveis.

Uma das principais vantagens é a sua capacidade de oferecer uma autenticação biométrica consistente nas diferentes plataformas, Android e iOS. Assim a solução não teve a necessidade de ter implementações específicas para cada plataforma, garantindo uma experiência de utilizador unificada nos dois sistemas operativos distintos. Desta forma a solução foi capaz de fornecer uma experiência de utilizador mais consistente, intuitiva e de fácil utilização, melhorando a segurança global da aplicação e a satisfação do utilizador [53].

5.3.2 Ecrã inicial

O ecrã inicial da aplicação móvel é a primeira página que o utilizador vê após a autenticação. Este ecrã contém uma lista de pedidos pendentes e finalizados do utilizador exibidos a partir da seleção dos botões `Ativos` e `Concluídos`, como ilustrado na figura 5.12. Nesta página, também é possível filtrar os pedidos exibidos por tipo, clicando no botão de filtro, conforme mostrado também na figura. Além disso, é possível fazer *logout* da aplicação através do botão `Sair` no canto superior direito da página e adicionar um novo pedido de videoconferência pelo botão no canto inferior direito com o ícon +.

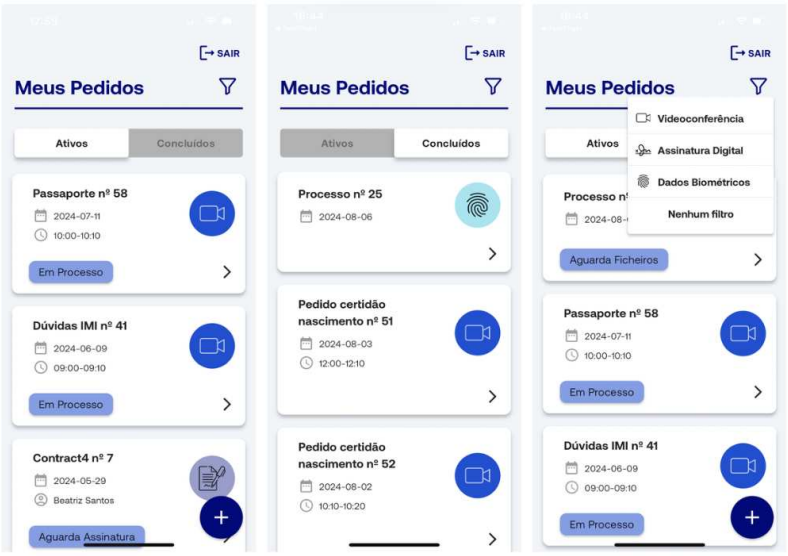


Figura 5.12: Página inicial ao selecionar pedidos ativos, concluídos e ao clicar no filtro, respetivamente

Cada pedido exibido na lista contém informações essenciais, como o tipo de processo, a data e a hora do pedido ou nome do utilizador relacionado ao mesmo, e o estado do pedido. Como mostrado na secção 4.3, cada processo pode possuir vários estados distintos, que variam entre ativos (pendentes de ações por parte do utilizador ou do funcionário) e concluídos (validados, rejeitados ou cancelados). Para aumentar a usabilidade e clareza da aplicação, esses estados são representados por diferentes cores:

- **Azul:** Processos pedentes de ações por parte do utilizador.
- **Amarelo:** Processos pedentes de ações por parte do funcionário
- **Verde:** Processos já validados e concluídos.
- **Vermelho:** Processos rejeitados ou cancelados.

A mesma lógica foi igualamente aplicada aos processos onde cada tipo possui uma cor diferente e um ícone ilustrativo associado:

- **Azul Escuro:** Processos de videoconferência.
- **Roxo:** Processos de assinatura de documentos.
- **Azul Claro:** Processos de dados biométricos.

O utilizador pode, igualmente, clicar em qualquer pedido exibido no ecrã inicial para ser redirecionado à página correspondente, onde poderá visualizar mais detalhes e realizar as operações associadas a cada processo. As figuras 5.13 e 5.14 ilustram estas páginas, sendo que a primeira representa pedidos ainda por concluir e a segunda pedidos já concluídos.

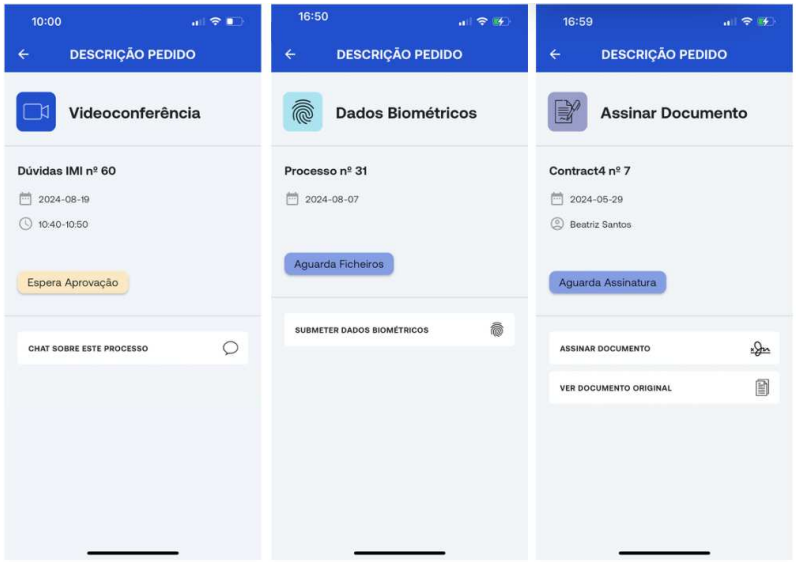


Figura 5.13: Páginas descritivas dos pedidos ativos

Este ecrã também exibe várias mensagens de erro, ilustrados na figura 5.15, para melhorar a usabilidade da aplicação conforme as dificuldades apresentadas, como:

- Falta de conexão à internet;
- Utilizador sem pedidos associados;

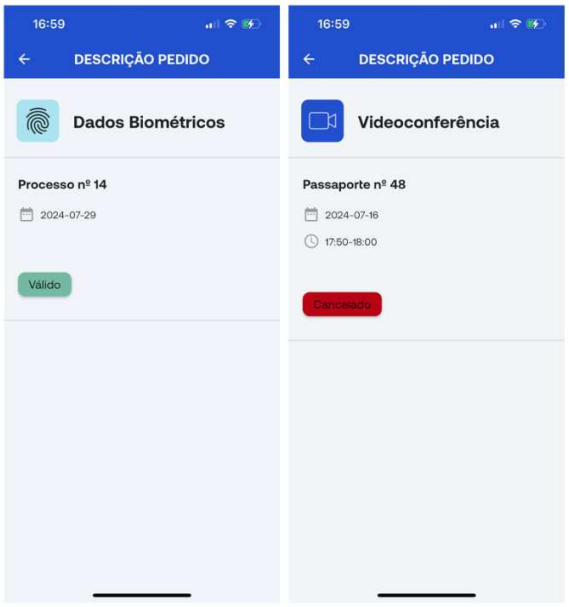


Figura 5.14: Páginas descritivas dos pedidos concluídos

- Utilizador sem pedidos ativos ou concluídos quando está na respetiva página;
- Utilizador que selecionou no filtro um tipo de processo para o qual não possui pedidos.

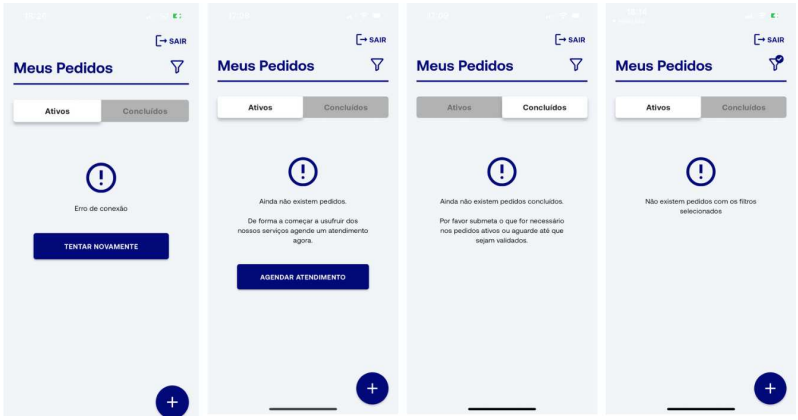


Figura 5.15: Páginas inicias com as várias mensagens de erro possíveis

5.3.3 Agendamento e Realização de Videoconferências

Como referido anteriormente, o produto possui três tipos de processos diferentes, sendo um deles a realização de videoconferências entre o cliente e o funcionário. Essas videoconferências destinam-se a processos que requerem interação direta e autenticação reforçada, como a extração de dados para a criação de um documento de identificação ou emissão de certidões de nascimento e esclarecimento de dúvidas sobre assuntos como Imposto Municipal sobre Imóveis (IMI).

Existem portanto dois tipos de videoconferências: a **Normal** e a de **Autenticação**. As videoconferências de Autenticação são aquelas em que o funcionário segue um *script* / guia pré-definido para extrair dados biométricos, visando a sua autenticação e criação posterior de documentos.

As videoconferências Normais são aquelas que não requerem nenhum *script* específico, sendo conversas normais entre o utilizador e o funcionário para exclarecimento de dúvidas.

Para a realização de uma videoconferência, o utilizador deve ainda, previamente, agendá-la numa data e hora disponíveis no sistema.

5.3.3.1 Agendamento de Videoconferências

O agendamento de videoconferências é uma funcionalidade essencial na aplicação móvel, permitindo ao utilizador marcar uma videoconferência com um funcionário.

Esta funcionalidade é realizada através de um formulário de agendamento que aparece após o utilizador clicar no botão mencionado anteriormente na subsecção 5.3.2 no ecrã inicial. Neste formulário o utilizador pode selecionar o tipo de videoconferência que quer realizar, e a data e a hora desejadas, conforme ilustrado na figura 5.16.

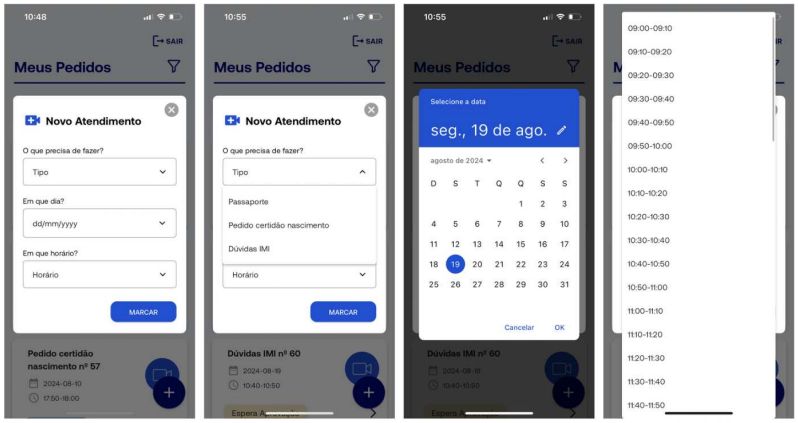


Figura 5.16: Formulário de agendamento de uma videoconferência

Após inserir os dados no formulário, o utilizador pode clicar no botão `Marcar` para enviar o pedido de agendamento. Se ocorrer algum problema ou o pedido contiver erros, uma mensagem de erro é exibida, como ilustrado na figura 5.17. No entanto, se o pedido for efetuado com sucesso, ele é então processado e armazenado na base de dados e o utilizador é redirecionado de volta ao ecrã inicial com uma mensagem de sucesso, como ilustrado na figura 5.18.

5.3.3.2 Realização de Videoconferências

Assim que o utilizador clica num processo de videoconferência na página principal, é redirecionado para a página correspondente, que aparece conforme ilustrado na figura

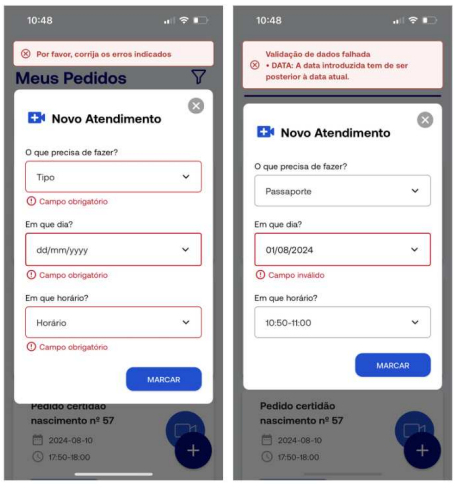


Figura 5.17: Formulário de agendamento de uma videoconferência com os erros possíveis

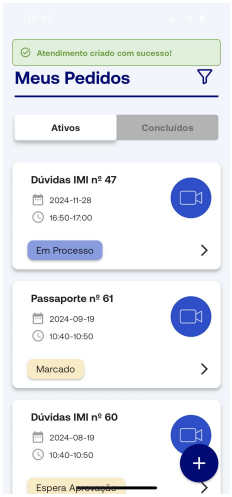


Figura 5.18: Agendamento de uma videoconferência criada com sucesso

5.19, dependendo do estado do processo. Nesta página, é possível ver mais detalhes sobre o processo e realizar as operações relacionadas, que variam conforme o estado do mesmo. Somente as videoconferências nos estados `Marcado` e `Em Processo` possuem a opção de iniciar a videoconferência; as restantes não possuem ações disponíveis.

Ao iniciar uma videoconferência, durante o estabelecimento da conexão entre o utilizador e o funcionário, ou caso o funcionário ainda não tenha entrado na chamada, a página exibe a imagem que está a ser transmitida localmente na chamada captada pela câmara do telemóvel, conforme ilustrado na figura 5.20.

Quando a videoconferência é estabelecida e o funcionário entra na chamada, a página passa a exibir a imagem transmitida pelo funcionário, juntamente com a imagem local do utilizador, que aparece num retângulo no canto superior direito como defeito. Implementou-se também a funcionalidade de mover este retângulo, permitindo que o utilizador posicione a sua imagem onde desejar, para evitar ocultar qualquer informação que o funcionário possa estar a mostrar, conforme ilustrado na figura 5.21.

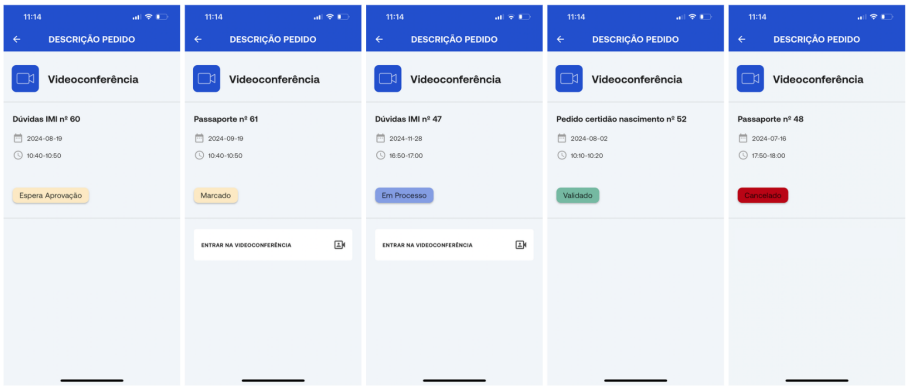


Figura 5.19: Páginas de videoconferência com todos os estados possíveis

Figura 5.20: Página inicial da chamada de videoconferência sem o funcionário

Além disso, estas páginas apresentam ainda quatro botões: um para alternar entre a câmara frontal e traseira, um para desligar a chamada, outro para desligar a câmara e um para desligar o microfone, como ilustrado na figura 5.22.

Como mencionado anteriormente na secção 2.3, foram estudas várias tecnologias para a implementação das videoconferências na aplicação móvel sendo que se obtou por usar o WebRTC com apoio de um servidor de sinalização e de servidores STUN e TURN e o uso do Socket.IO.

Estabelecimento das Videoconferências

O projeto *idfyme-plugins* já possuía a implementação dos servidores STUN e TURN, bem como o do servidor de sinalização, incluindo toda a lógica de conexão entre os *peers*/utilizadores para a realização de videoconferências neste servidor de sinalização, uma vez que essa funcionalidade já estava presente nas aplicações Android e *web*. Assim, a solução implementada na solução consistiu em integrar essas estruturas já existentes,

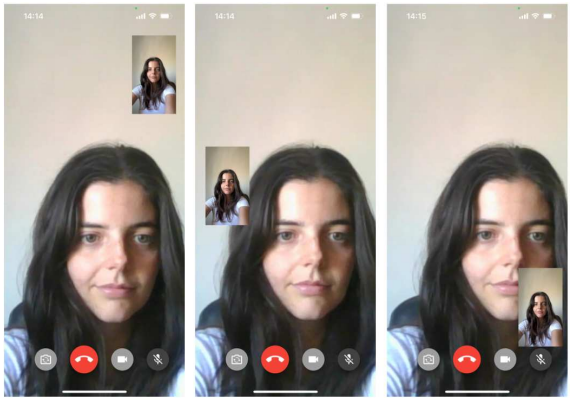


Figura 5.21: Página inicial da chamada de videoconferência com a imagem local movida

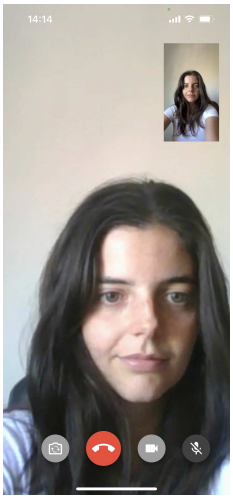


Figura 5.22: Página da videoconferência normal na aplicação móvel

como ilustrado na figura 5.23.

Utilizou-se a biblioteca Flutter `socket-io-client`[20] que forneceu uma interface para a criação de *sockets* de comunicação em tempo real, permitindo a troca eficiente, segura e contínua de mensagens entre a aplicação móvel e o servidor de sinalização para a criação e estabelecimento das ligações de videoconferência.

Esta biblioteca utiliza o Socket.IO, não apenas *sockets* brutos. O Socket.IO é uma biblioteca JavaScript que permite a comunicação em tempo real, bidirecional e baseada em eventos entre *browsers* e servidores. Esta é construída sobre WebSockets, mas fornece uma série de funcionalidades adicionais, como *fallback* para outros protocolos se os WebSockets não estiverem disponíveis, reconexão automática e uma API mais amigável [26].

A partir desta biblioteca, foram então gerados *sockets* que atuaram como canais de comunicação, através dos quais foram trocadas as mensagens de conexão inicial com o servidor de sinalização e com a conexão P2P (que continha os servidores STUN e TURN a serem usados). Estes *sockets* também foram responsáveis pelo envio de ficheiros

[20] `https://pub.dev/packages/socket_io_client`

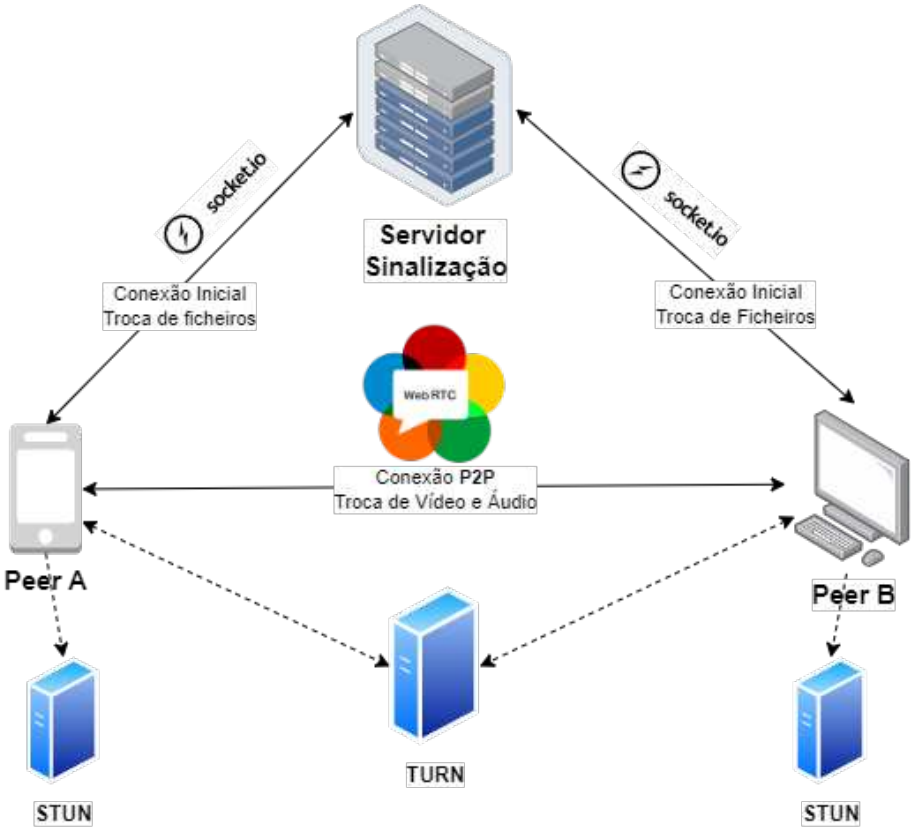


Figura 5.23: Diagrama da implementação da videoconferência

pertinentes durante as videoconferências, como fotos e *tokens* de autenticação.

Conexão ao Servidor de Sinalização

O estabelecimento da conexão para a videoconferência começa então com os *peers*/utilizadores a se conectarem ao servidor de sinalização. Esta conexão é estabelecida por meio da troca de mensagens entre eles e o servidor de sinalização, através de *sockets* inicializados sobre o protocolo WebSocket, como ilustrado na figura 5.24.

A primeira mensagem trocada consiste na conexão do *socket* Socket.IO ao servidor de sinalização, que inclui a autenticação do utilizador com um *token* de acesso obtido através de um pedido ao servidor do produto, juntamente com o ID do utilizador. Este processo garante a segurança e a integridade da conexão.

Em seguida, é trocado um conjunto de mensagens para finalizar esta conexão inicial, que inclui o seguinte:

Peer /utilizador A, que inicia a conexão da videoconferência:

- Emite uma mensagem `createOrJoin` com o seu ID e o identificador da sala à qual se pretende conectar.
- O servidor de sinalização emite uma mensagem de resposta (`created`) para informar

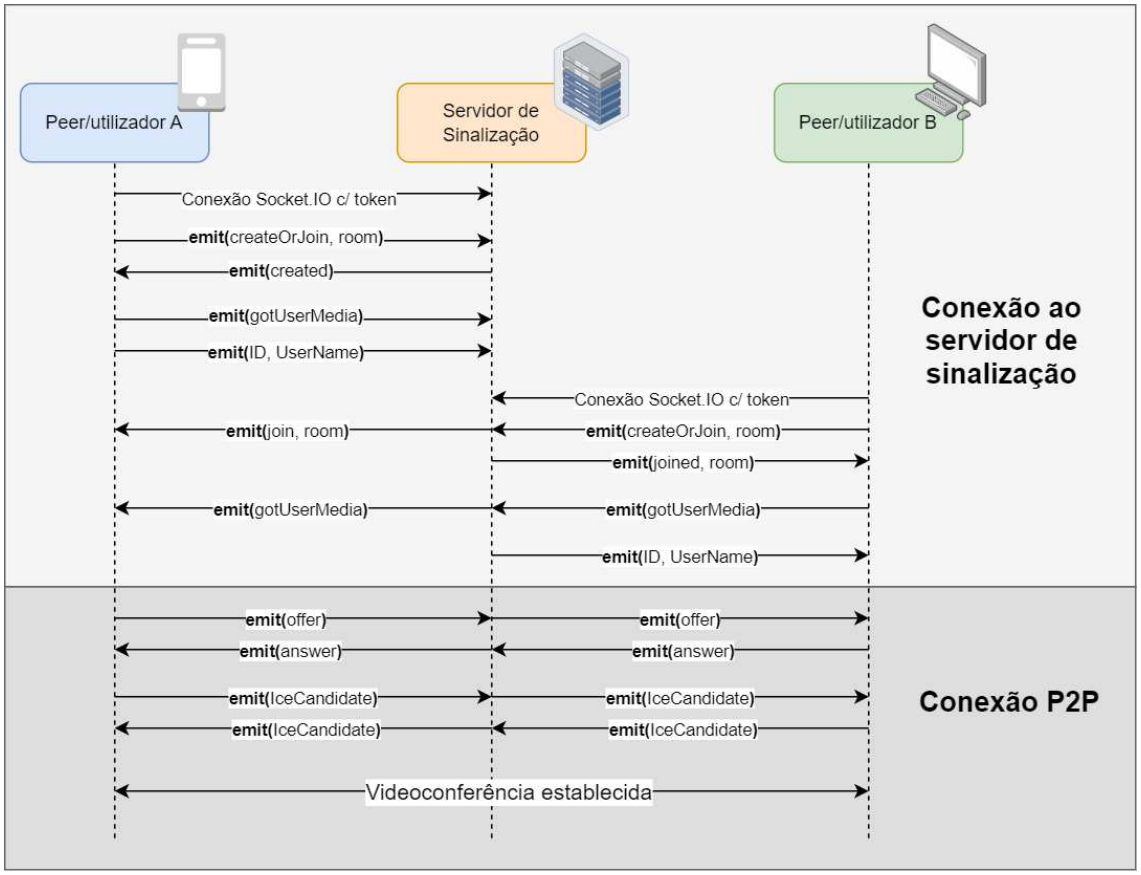


Figura 5.24: Diagrama do fluxo de mensagens de estabelecimento da videoconferência

que a conexão do utilizador ao servidor foi criada com sucesso e que o pedido de criação da videoconferência na sala solicitada foi bem-sucedido.

- O utilizador emite então uma mensagem `gotUserMedia` para informar ao servidor e outros *peers* que está pronto para iniciar a conexão inicial da videoconferência. Além disso, envia uma mensagem com o seu *username* e o seu ID para que possa ser identificado posteriormente durante a chamada com o outro *peer*.

- Após estas trocas de mensagens, o utilizador aguarda que outro *peer* se conecte à mesma videoconferência e informe que quer entrar na mesma sala (*room*). Quando isso acontece, o servidor de sinalização emite uma mensagem de resposta (`join`) para este *peer* A, de modo a informar que ele foi conectado com sucesso à sala.

Peers/utilizadores que entram numa conexão de videoconferência já iniciada (*Peer* B):

- Emite ao servidor de sinalização uma mensagem `createOrJoin` com o seu ID e o identificador da sala à qual se pretende conectar.

- O servidor de sinalização emite uma mensagem de resposta (`joined`) para informar que o conectou à sala que já estava criada devido ao *peer* A.

Conexão P2P

Após os *peers*/utilizadores se conectarem inicialmente ao servidor de sinalização e se vincularem à sala desejada, passa-se à sua conexão P2P de forma a gerar a chamada de videoconferência de vídeo e aúdio. Para isso, foi usado o *plugin* `flutter-webrtc`[21] que fornece uma interface para o WebRTC em Flutter, permitindo a criação de aplicações de videoconferência em tempo real de forma simples e eficiente, garantindo uma experiência de utilizador suave e de alta qualidade. O `flutter-webrtc` oferece suporte para a acessibilidade à câmera e microfone dos dispositivos móveis, à criação e gestão das conexões P2P e à transmissão de áudio e vídeo em tempo real.

Ao utilizar este *plugin*, foi possível implementar a conexão P2P com os servidores STUN e TURN através do método `createPeerConnection()` que os recebe como argumentos, e realizar a transmissão de áudio e vídeo em tempo real entre os *peers*. Na solução, esta transmsissão foi alcançada utilizando as classes `RTCVideoRenderer` para exibir o áudio e vídeo remoto dos outros *peers*, e a classe `MediaStream` para capturar o áudio e vídeo local do utilizador da aplicação móvel.

A conexão P2P inicial é criada ainda através de mensagens intermediadas pelo servidor de sinalização e efetuadas por meio de *sockets*, como ilustrado na figura 5.24, sempre incluindo o ID do *peer* remetente e o ID do *peer* destinatário. Neste processo, ocorre a seguinte troca de mensagens:

- O *peer* B, que não iniciou a chamada, após receber a mensagem do servidor de sinalização `joined` anterior emite uma mensagem `gotUserMedia` para informar ao servidor e aos outros *peers* que está pronto para criar a conexão inicial P2P. Esta mensagem é reencaminhada pelo servidor de sinalização para o *peer* que iniciou a chamada, informando-o de que há outro *peer* pronto para se conectar à videoconferência.

- O *peer* A, que iniciou a chamada, inicia então a troca de mensagens para iniciar o processo de negociação e estabelecimento da conexão P2P para a videoconferência utilizando os servidores STUN e TURN. Esta troca começa com a emissão de uma mensagem de oferta (`offer`).

- O outro *peer* emite uma mensagem de resposta (`answer`) com o seu ID e a sua descrição de conexão.

- Por fim, eles trocam mensagens contendo os seus `ICE Candidates` para finalmente finalizar a conexão P2P.

A mensagem `offer` é enviada por um *peer*/utilizador para iniciar a negociação da conexão P2P. Ela contém uma descrição completa da configuração que o *peer* está a propor para a ligação, incluindo *codecs* de áudio e vídeo, tipos de média, parâmetros de transporte

[21] `https://pub.dev/packages/flutter_webrtc`

e outras informações técnicas necessárias. A `offer` define as capacidades e preferências do *peer* que está a fazer a oferta e serve como base para a configuração da ligação. O *peer* que inicia a ligação cria então uma oferta utilizando o método `createOffer()` e envia-a ao outro *peer* através do *socket* associado.

A mensagem `answer` é enviada em resposta a uma `offer`. Ela contém a descrição da configuração que o *peer* respondente aceita para a ligação, ajustada conforme necessário para ser compatível com a `offer`. A `answer` confirma a aceitação da configuração proposta na `offer`, possivelmente ajustando alguns parâmetros, e completa a negociação inicial da ligação. Assim, o *peer* que recebe a `offer` cria uma resposta utilizando o método `createAnswer()` e envia a `answer` de volta ao *peer* que fez a `offer` através do mesmo canal de sinalização.

As mensagens `candidate` são utilizadas para descobrir e trocar informações sobre os possíveis caminhos de rede que podem ser usados para estabelecer a ligação P2P. Elas contêm detalhes sobre os candidatos e os seus caminhos de rede, incluindo endereços IP e portas que podem ser usados para a comunicação. As `candidate` facilitam assim o processo de Interactive Connectivity Establishment (ICE), permitindo que os *peers* descubram o melhor caminho de rede para estabelecer a ligação direta. Durante o processo de ICE, ambos os *peers* recolhem candidatos utilizando o método `onicecandidate()`. Esses candidatos são então trocados entre os *peers* através do servidor de sinalização via *sockets*, onde depois se testa várias combinações dos candidatos recebidos para encontrar o caminho de rede mais eficiente e estável para usar na conexão.

O processo de negociação para estabelecer a conexão P2P começa então quando o *peer* A invoca o método `createOffer()` e envia a oferta (`offer`) para o *peer* B. O *peer* B recebe então a oferta, utiliza o método `createAnswer()` para gerar e enviar a resposta (`answer`) de volta ao *peer* A. Em seguida, ambos os *peers* recolhem e trocam candidatos (`candidate`) através do servidor de sinalização como intermediário e canais de *sockets*, até encontrar o melhor caminho de rede disponível.

Durante as Videoconferências

Após o estabelecimento inicial da ligação de videoconferência, ambos os intervenientes partilham o áudio e vídeo através da ligação P2P, enquanto que os seus dispositivos permanecem a ouvir, de forma contínua e sem interrupções, às mensagens distribuídas pelo servidor de sinalização através dos *sockets* conectados.

Quando a ligação envolve autenticação e a execução de um *script* para a extração de dados biométricos, estas mensagens notificam os intervenientes sobre ações que precisam de ser tomadas ou canceladas durante a videoconferência. Isto inclui informações sobre dados que devem ser enviados ou que já foram enviados e precisam de validação, encerramento das chamadas e mudança dos critérios da ligação.

Se o funcionário solicitar ou cancelar a extração de dados, uma mensagem é enviada

ao servidor de sinalização, que a reencaminha para o utilizador. A aplicação móvel, então, recebe a mensagem através do seu *socket*, e as páginas correspondentes à extração são exibidas ao utilizador para que possa prosseguir com o processo. Da mesma forma, após a extração e submissão dos dados pelo utilizador, estes são enviados ao servidor de sinalização, que os reencaminha para o funcionário para as suas validações.

A mesma lógica se aplica ao encerramento da chamada por qualquer um dos intervenientes. Se um dos intervenientes encerrar a chamada, mensagens de encerramento são enviadas ao servidor de sinalização, que notifica o outro interveniente sobre o término da chamada. Isto permite que o interveniente que não a desligou tome as ações necessárias em resposta ao encerramento da chamada.

Este processo apresenta a vantagem de ser mais eficiente e consumir menos recursos, uma vez que ambos os *peers* apenas ouvem as mensagens do servidor de sinalização assíncronamente, sem a necessidade de enviar mensagens constantemente para determinar os próximos passos. Assim, o servidor de sinalização assume a responsabilidade de gerir o fluxo da videoconferência e de notificar os intervenientes sobre as ações a serem tomadas na mesma. Esta abordagem garante uma comunicação eficiente e sincronizada entre os intervenientes, minimizando a carga de trabalho e o consumo de recursos das aplicações dos *peers*.

Tipos de Fluxos das Videoconferências

Como dito anteriomente existem dois tipos de videoconferências: a **Normal** e a de **Autenticação**, a serem explicadas de seguida.

Normal

As videoconferências Normais são aquelas que não requerem nenhum *script*/guia específico, sendo conversas normais entre o utilizador e o funcionário para exclarecimento de dúvidas.

Nenhuma implementação especial foi necessária aqui, pois apenas a transmissão em tempo real de áudio e vídeo através da conexão P2P estabelecida anteriormente é suficiente, conforme ilustrado nas figuras 5.25 e 5.26.

Autenticação

As videoconferências de Autenticação são aquelas em que o funcionário segue um *script*/guia pré-definido para extrair dados biométricos do utilizador, visando a sua autenticação e criação posterior de documentos oficiais.

Atualmente existem dois tipos de videoconferências de Autenticação no produto, as para realizar um Passaporte e as para emitir uma Certidão de Nascimento. Cada uma destas videoconferências possui um *script* específico que o funcionário têm de seguir, como ilustrado na figura 5.27.

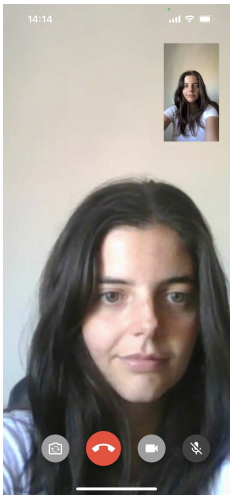


Figura 5.25: Página da chamada de videoconferência normal na aplicação móvel

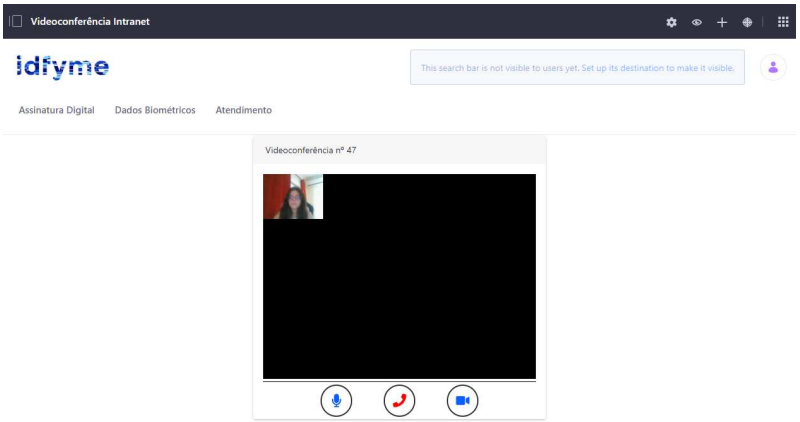


Figura 5.26: Página *web* da chamada de videoconferência Normal na aplicação *web*

A emissão de uma Certidão de Nascimento segue um *script* mais simples, mostrado na figura 5.28, no qual o funcionário precisa apenas de validar o ambiente do utilizador para garantir a sua privacidade e verificar um *token* emitido para assegurar a sua participação ativa e autêntica, ilustrados, respetivamente, nas figuras 5.29 e 5.30.

Por outro lado, a emissão de um Passaporte envolve um *script* mais complexo que inclui os seguintes passos:

- Validar o ambiente do utilizador para garantir a sua privacidade e segurança;
- Emitir um *token* que aparece no ecrã da videoconferência aplicação móvel do utilizador, garantindo assim a participação ativa e autêntica do mesmo e reduzindo significativamente o risco de fraude ou falsificação de identidade;
- Validação de dados biométricos, ilustrado na figura 5.32, onde o funcionário solicita o nome do utilizador e a sua data de nascimento, confirmando as respostas no seu ecrã para garantir a autenticação do utilizador;

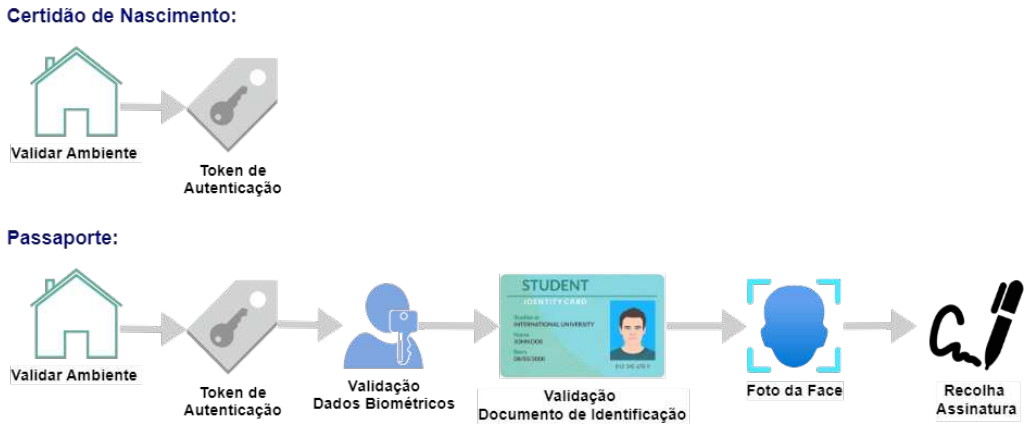


Figura 5.27: Diagrama do fluxo dos processos da videoconferência de autenticação

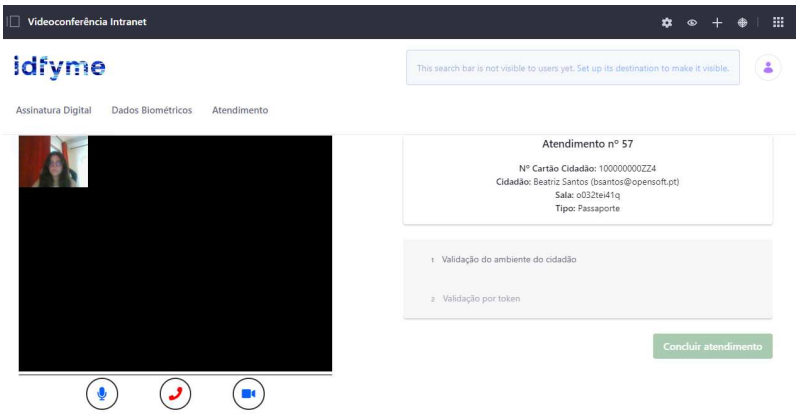


Figura 5.28: Página *web* da vista do funcionário com o *script* da videoconferência de autenticação da certidão de nascimento

- Validação do documento de identificação, pedindo que o utilizador efetue fotografias da frente e verso do documento;
- Recolha da foto da face do utilizador;
- Recolha da assinatura manuscrita do utilizador;

Este *script* é exibido conforme mostrado na figura 5.31. É importante destacar que o funcionário só consegue entrar numa videoconferência de autenticação através da aplicação *web*. Por esse motivo, não foi necessário implementar este fluxo na solução, apenas a videoconferência normal do lado do utilizador na aplicação móvel em resposta aos pedidos deste *script*.

Assim, foram criadas novas páginas para responder aos pedidos de emissão do *token*, ilustrado na figura 5.33, validação do documento de identificação, recolha da foto da face e recolha da assinatura manuscrita do utilizador que serão discutidas mais detalhadamente a seguir.

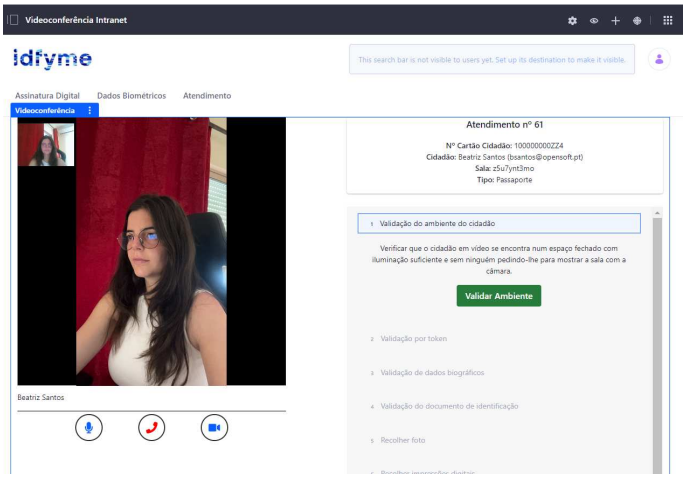


Figura 5.29: Página *web* da vista do funcionário para validar o ambiente do utilizador

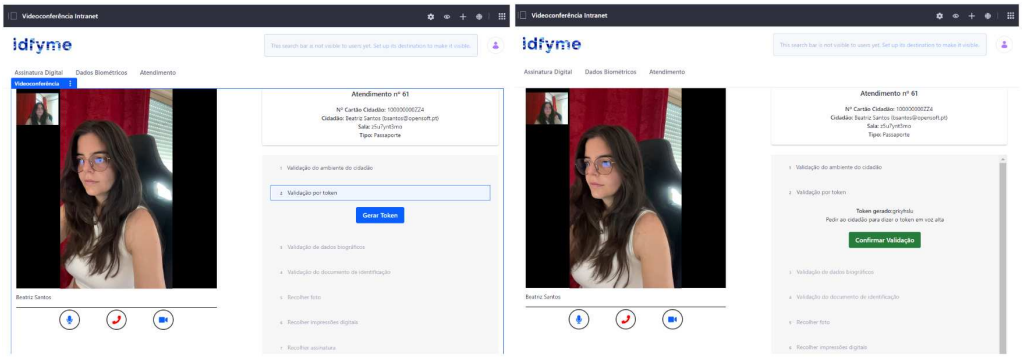


Figura 5.30: Página *web* da vista do funcionário para gerar e validar o *token* de autenticação

Todas estas páginas aparecem automaticamente no ecrã do dispositivo móvel na aplicação móvel do utilizador, conforme os pedidos do funcionário na aplicação *web*. O funcionário é responsável por controlar o fluxo da videoconferência, fazendo o pedido e validando cada fase e fotografia tirada. Caso a qualidade da imagem não seja suficiente ou a informação fornecida esteja incorreta, o funcionário pode recusar e solicitar novamente.

É também importante notar que todos os processos de extração de fotografias e dados biométricos durante a videoconferência podem ser cancelados a qualquer momento pelo funcionário, como ilustrado na figura 5.34. Se o utilizador estiver no meio da extração nas páginas correspondentes a esse processo, será interrompido e redirecionado automaticamente para a página da videoconferência inicial com uma mensagem num *popup* a informar desse cancelamento, como ilustrado na figura 5.35.

A mudança de páginas a pedido do funcionário, através do *script* na *web*, gerou uma complexidade adicional no sistema de navegação desta funcionalidade na aplicação móvel. Quando a aplicação recebia, via *sockets*, mensagens de mudança de página para extração de dados ou cancelamento dos mesmos, era necessário redirecionar o utilizador para a página correspondente. Para lidar com isso, foi implementado o uso do `ChangeNotifier`,

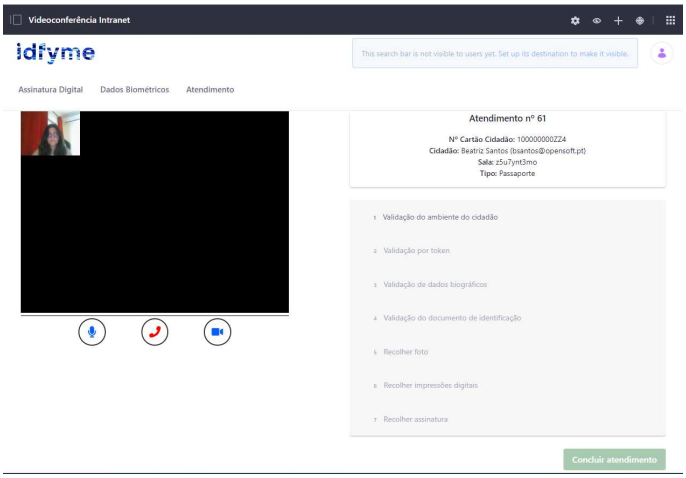


Figura 5.31: Página *web* da vista do funcionário com o *script* da videoconferência de autenticação do passaporte

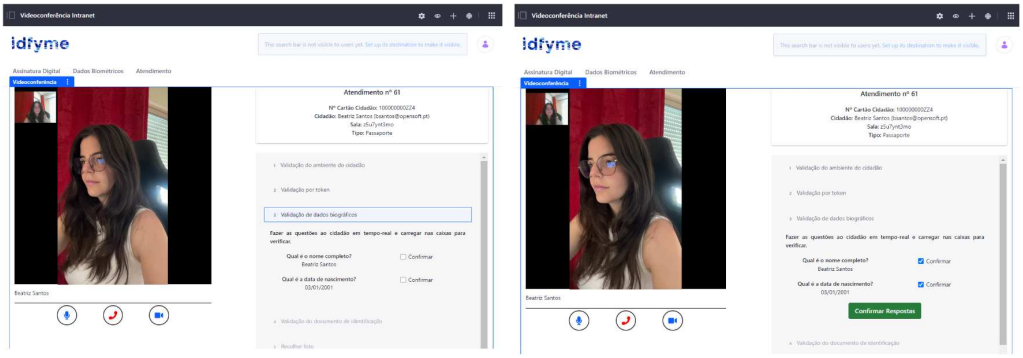


Figura 5.32: Página *web* da vista do funcionário para validar os dados biométricos do utilizador

conforme explicado anteriormente na secção 5.1.5.

Assim, foi criada uma classe `ChangeNotifier` que mantém o estado da página da videoconferência a ser exibida ao utilizador, sendo atualizada sempre que o funcionário envia uma mensagem de mudança de página, recebida via *sockets*. Dessa forma, a classe responsável por exibir a página da videoconferência está subscrita a essa classe, sendo notificada da alteração e redirecionando o utilizador para a página correspondente.

Em caso de encerramento da chamada ou falha de conexão no meio da autenticação, a videoconferência na aplicação móvel é automaticamente encerrada e o utilizador é redirecionado para o ecrã da página da videoconfência com uma mensagem de erro, como ilustrado na Figura 5.36.

Se a videoconferência for concluída com sucesso, após o funcionário clicar no botão `Concluir Atendimento` após a extração e validação de todos os dados, conforme ilustrado na figura 5.37, o utilizador é redirecionado para o ecrã inicial, agora com uma mensagem de sucesso, conforme mostrado na figura 5.38.

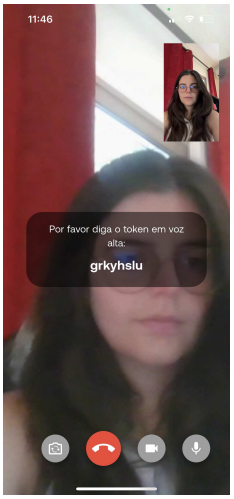


Figura 5.33: *Token* de autenticação emitido para o utilizador na aplicação móvel

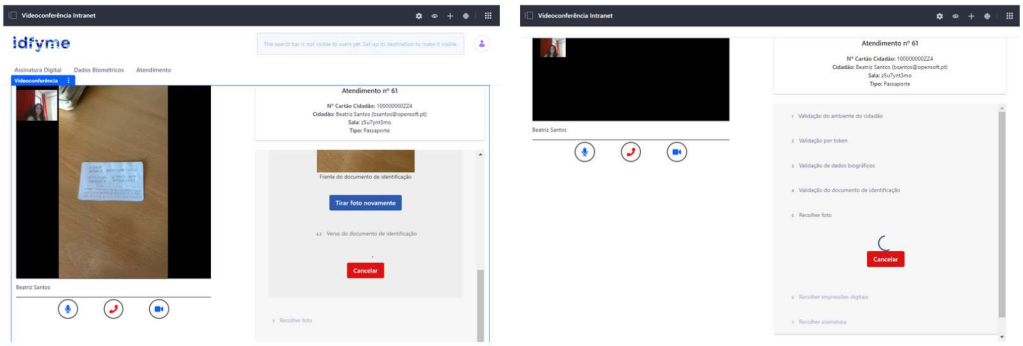


Figura 5.34: Página da *web* vista pelo funcionário para cancelar o processo que já iniciou

Fotografia do documento de identificação

Para a implementação da fotografia do documento de identificação foi usada a biblioteca Flutter `image`[22] que oferece funcionalidades para manipulação de imagens permitindo realizar diversas operações como decodificar formatos de imagem comuns (como JPEG, PNG, Graphics Interchange Format (GIF)), aplicar transformações (como redimensionamento, rotação, recorte), salvar imagens em diferentes formatos, processamento de *pixels*, aplicação de filtros, entre outros.

Assim, quando o funcionário solicita a fotografia do documento de identificação, o utilizador é redirecionado para a página correspondente. A câmera é automaticamente ativada e configurada para a de trás, e o microfone é reativado caso o utilizador o tenha desativado anteriormente. Nesta página, é exibida uma mensagem com as instruções para capturar a fotografia da frente do documento, conforme ilustrado na figura 5.39.

Esta página possui ainda uma barra de progresso que indica o progresso da realização da fotografia ao utilizador tendo quatro fases, que vão mudando consoante as ações do funcionário e envio das fotos, como ilustrado na figura 5.40:

[22] `https://pub.dev/packages/image`

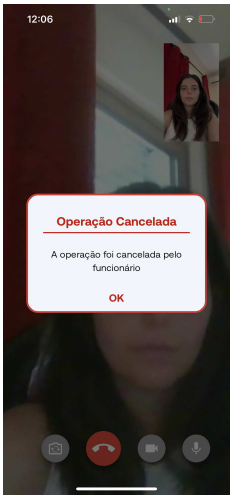


Figura 5.35: *Popup* a informar que a operação foi cancelada pelo funcionário durante a videoconferência de autenticação

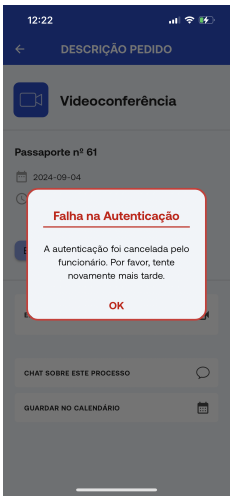


Figura 5.36: *Popup* a informar que a videoconferência de autenticação foi cancelada

- À espera do Documento de Identificação;
- A tirar a foto;
- A enviar a foto;
- Foto enviada com sucesso.

Quando o funcionário verifica que o utilizador posicionou corretamente o cartão de identificação no ecrã e está pronto para tirar a foto, ele clica no botão `Tirar Fotografia` no seu *script*, como apresentado na figura 5.41. Isso envia o pedido à aplicação móvel, que então captura a fotografia diretamente do ecrã do utilizador.

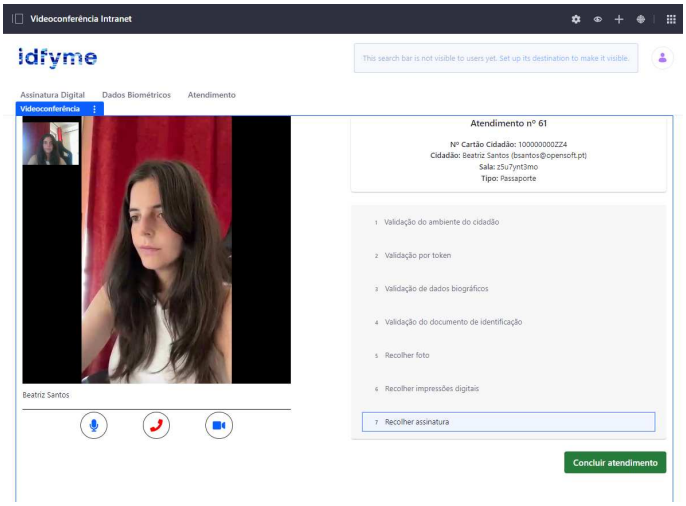


Figura 5.37: Página *web* vista pelo funcionário para concluir a videoconferência de autenticação

Figura 5.38: *Popup* a informar que a videoconferência de autenticação foi terminada com sucesso

Ao ser o funcionário a realizar esta ação evita que o utilizador tenha que manipular ou interagir diretamente com o dispositivo móvel para capturar a fotografia, facilitando o processo assim como a sua precisão, segurança e eficiência.

Esta foto é tirada através da câmera do dispositivo móvel que recupera a faixa de vídeo local emitido na videoconferência naquele instânte e captura assincronamente um fotograma utilizando o método `captureFrame()` do *plugin* `flutter-webrtc`. O fotograma capturado é então descodificado numa imagem utilizando a biblioteca `image`. Se for bem sucedida, a imagem é codificada para o formato PNG e subsequentemente convertida numa cadeia de caracteres codificada em `base64` e enviada para o servidor de sinalização que a reencaminha para o funcionário.

Optou-se pelo formato PNG devido ao seu uso de compressão sem perdas, garantindo que não haja perda de qualidade da imagem durante o processo de compressão. Esta

Figura 5.39: Página da aplicação móvel para a captura da frente do documento de identificação

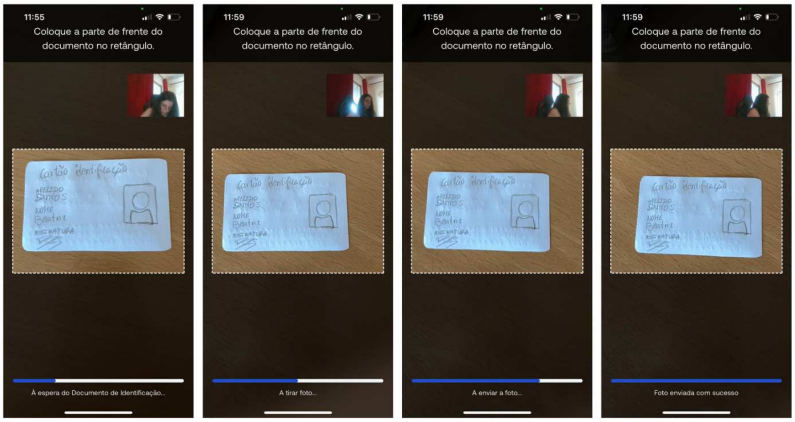


Figura 5.40: Páginas da aplicação móvel com a progressão da barra de progresso quando o funcionário solicita a captura da foto do documento de identificação

escolha é crucial para assegurar que a imagem capturada seja nítida e clara, facilitando ao funcionário verificar os detalhes do documento de identificação com precisão [54].

A codificação em `base64` é essencial para garantir que os dados da imagem, originalmente em formato binário, sejam transmitidos de maneira segura e eficiente através dos *sockets*. Isto ocorre porque a `base64` converte os dados binários numa representação textual composta apenas por caracteres American Standard Code for Information Interchange (ASCII), que são mais seguros e fáceis de manipular, armazenar e transmitir, além de serem universalmente compatíveis com diversas plataformas e tecnologias [55].

O funcionário pode então validar a imagem e solicitar a captura do verso do documento de identificação, como apresentado na figura 5.42, repetindo o processo. Caso a imagem não esteja adequada, ele pode recusá-la e solicitar que seja tirada novamente, como ilustrado na figura 5.43. No final da captura da parte de trás do documento o funcionário pode então validar as imagens como ilustrado na figura 5.44 e concluir o processo de

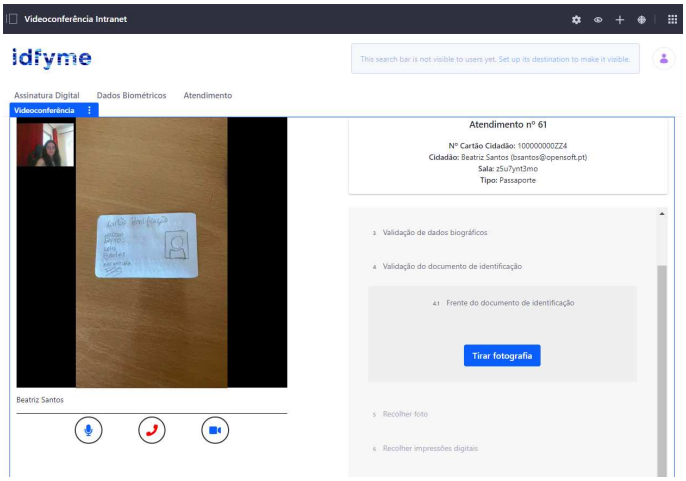


Figura 5.41: Página *web* vista pelo funcionário para solicitar que o utilizador tire a foto

extração do documento de identificação.

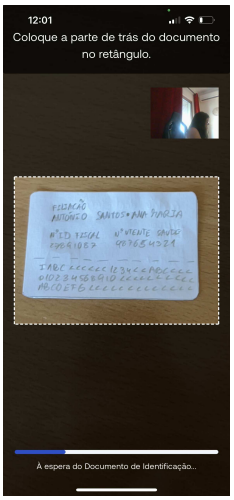


Figura 5.42: Página da aplicação móvel para a captura da parte de trás do documento de identificação

Recolha da foto da face e assinatura manual

Para a recolha da foto da face e da assinatura manual do utilizador, foram utilizadas as mesmas páginas do processo de extração biométrica descritas na subsecção 5.3.4 a seguir.

Quando o funcionário solicita a recolha da foto da face e da assinatura manual, o utilizador é redirecionado para as páginas correspondentes na aplicação móvel. A câmera e o microfone são ativados automaticamente, caso o utilizador os tenha desativado anteriormente, e a câmera é configurada para a frontal, permitindo que o funcionário veja o que o utilizador está a fazer.

No caso da foto da face, o utilizador é levado para as páginas de extração e validação da mesma, conforme ilustrado na figura 5.45. Para a assinatura, o redirecionamento é feito para a página de extração da assinatura, conforme ilustrado na figura 5.46.

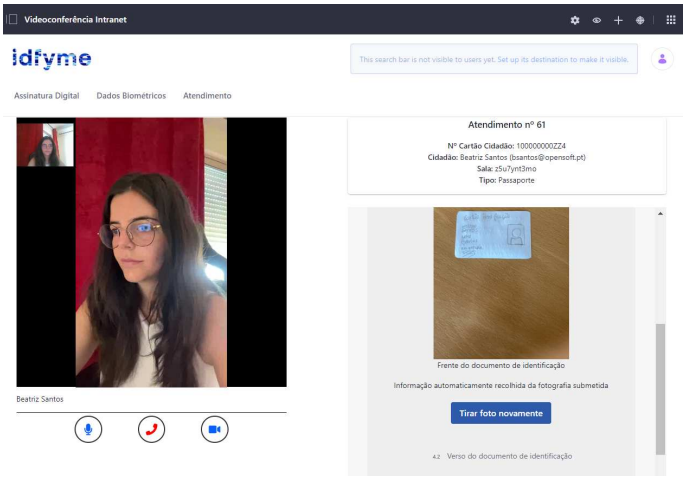


Figura 5.43: Página *web* vista pelo funcionário para solicitar que o utilizador faça a captura do documento novamente

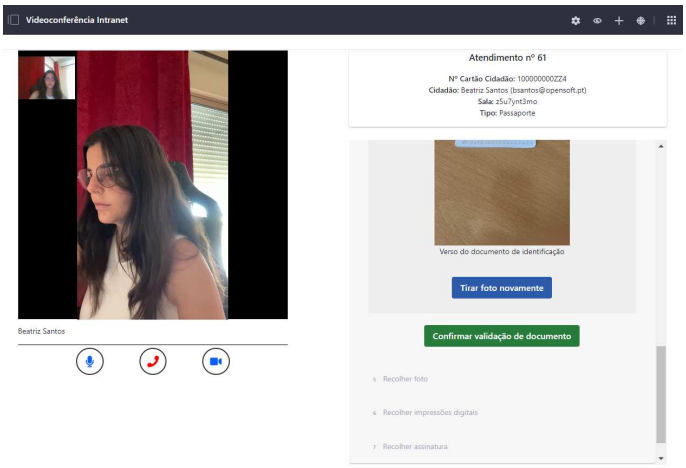


Figura 5.44: Página *web* vista pelo funcionário para validar as imagens do documento de identificação

Após a submissão dos dados pelo utilizador, estes são enviados por meio de *sockets* para o servidor de sinalização, que os reencaminha para o funcionário. As imagens são então exibidas no ecrã do funcionário para a sua validação posterior e conclusão do processo de extração. O funcionário pode então validar as imagens ou solicitar que sejam tiradas novamente, se necessário, como ilustrado nas figuras 5.47 e 5.48.

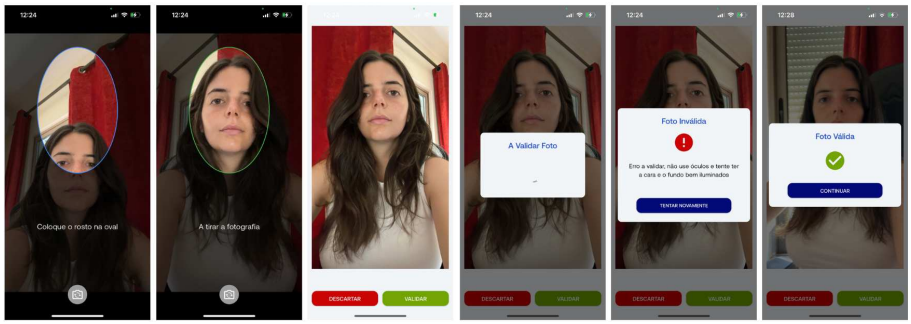


Figura 5.45: Páginas na aplicação móvel do processo de extração e pré validação da fotografia da face

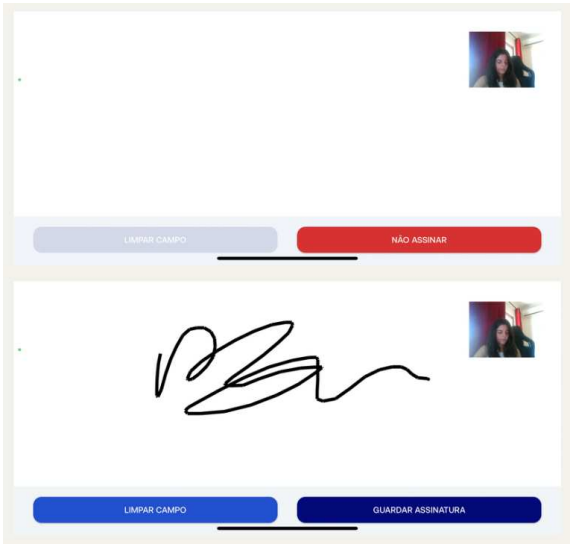


Figura 5.46: Páginas na aplicação móvel do processo de extração da assinatura

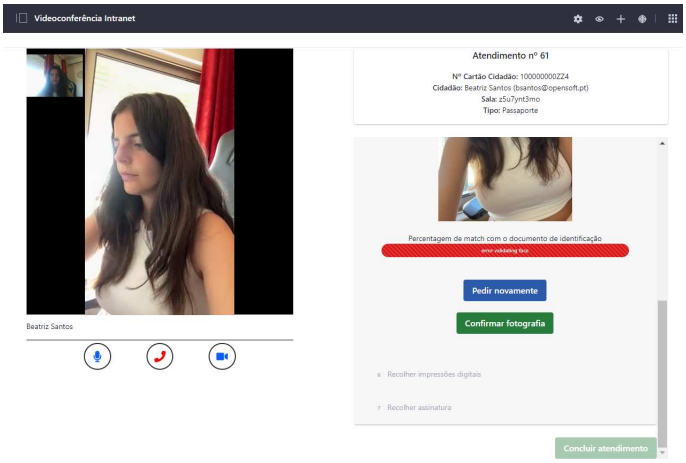


Figura 5.47: Página *web* vista pelo funcionário para validar a foto da face do utilizador e concluir o processo

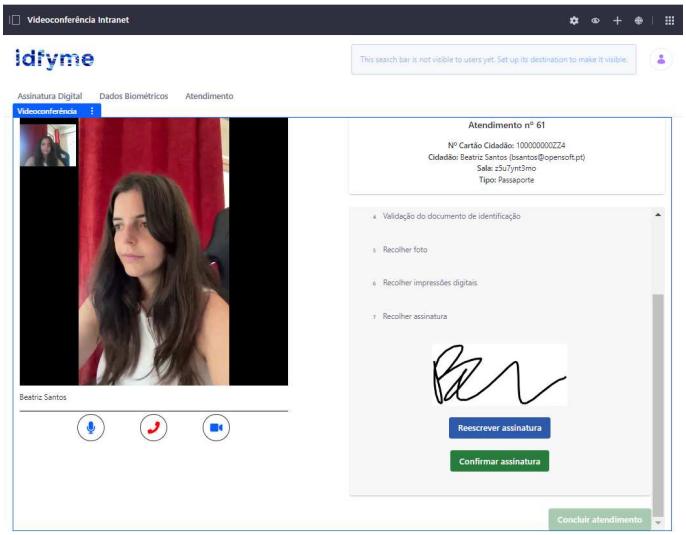


Figura 5.48: Página *web* vista pelo funcionário para validar a assinatura do utilizador e concluir o processo

No caso da recolha da foto da face, a página de extração necessita de acesso à câmera do dispositivo móvel para capturar a imagem. Para permitir esta funcionalidade, a ligação de vídeo da videoconferência precisa de ser interrompida temporariamente. Assim, antes de redirecionar o utilizador para a página de extração da foto da face, remove-se a faixa de mídia local de vídeo da ligação P2P da videoconferência. Em seguida, é feita uma nova oferta (`offer`) para a mesma sala e ligação P2P, mas agora com uma proposta de mídia atualizada que inclui apenas áudio e exclui vídeo, através do servidor de sinalização. Desta forma, ambos os intervenientes continuam a comunicar por áudio enquanto que vídeo é temporariamente desativado para que a fotografia da face seja extraída.

Quando o outro *peer* recebe a oferta reencaminhada pelo servidor de sinalização, aceita-a e gera uma nova resposta (`answer`). A conexão P2P é então restabelecida, mas agora com apenas áudio e sem vídeo.

O mesmo processo é repetido quando o utilizador termina a extração da foto da face e a submete indo novamente para a videoconferência inicial. A faixa de mídia local é restaurada para a nova ligação P2P da videoconferência, reativando tanto o áudio quanto o vídeo, numa nova oferta e resposta.

5.3.4 Extração de Dados Biométricos

Outro processo que a solução permite é a extração de dados biométricos do utilizador, onde as suas páginas estão ilustradas na figura 5.49. Estes dados biométricos incluem uma fotografia do rosto do utilizador, bem como a sua assinatura manuscrita digital.

Assim este processo apresenta o seguinte fluxo:

- Extração e validação da fotografia da face;

- Extração da assinatura manuscrita digitalmente;
- Confirmação e envio dos dados extraídos com autenticação do utilizador;

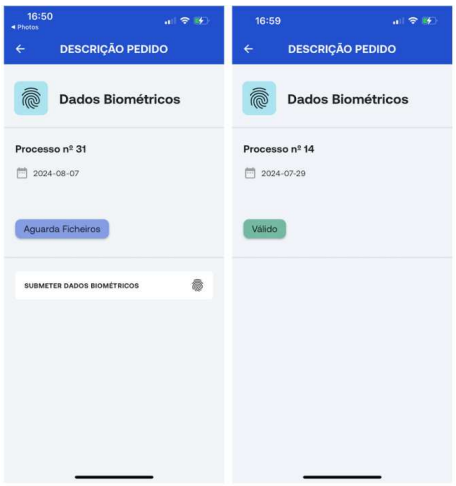


Figura 5.49: Páginas para o processo de extração de dados biométricos

Fotografia da Face

Para extrair uma fotografia do rosto do utilizador, foi necessário assegurar várias condições para garantir a qualidade e a segurança da imagem, conforme exigido para fotografias de rosto usadas em documentos de identificação. Para isso, utilizou-se o *plugin* Flutter `google-mlkit-face-detection`[23], que se baseia no ML Kit[24] da Google. O ML Kit fornece APIs prontas para funcionalidades de inteligência artificial em aplicações móveis, como reconhecimento de texto, rostos, digitalização de códigos de barras e tradução de texto, entre outras.

O *plugin* `google-mlkit-face-detection` no Flutter utiliza canais de plataforma para realizar a deteção de rostos, comunicando com as APIs nativas do ML Kit da Google em Android e iOS. Estes canais de plataforma são um mecanismo de comunicação que permite que o código Dart do Flutter interaja com o código nativo da plataforma, o Java/Kotlin no Android e o Swift/Objective-C no iOS. Portanto, neste *plugin*, nenhum processamento de inteligência artificial é feito diretamente no Flutter. Em vez disso, o *plugin* atua como uma ponte, enviando solicitações do Flutter para a plataforma nativa através do `MethodChannel` no Android e do `FlutterMethodChannel` no iOS. Estes canais permitem então que o Flutter invoque as APIs nativas, que realizam o processamento real usando o ML Kit da Google. Os resultados são então enviados de volta ao Flutter pelos mesmos canais e processados na aplicação.

[23] https://pub.dev/packages/google_mlkit_face_detection
[24] https://developers.google.com/ml-kit?hl=pt-br

Assim, a extração da fotografia do rosto inicia-se com a avaliação da face do utilizador usando este pacote para detectar a sua posição. Em seguida, verifica-se se o rosto está dentro da área oval desenhada no ecrã, para garantir uma centralização correta na fotografia, como apresentado na figura 5.50.

Figura 5.50: Página inicial para a extração da fotografia da face

Esta funcionalidade apresentou ainda um desafio adicional devido à diferença entre as coordenadas geradas na deteção facial pelo *plugin* `google-mlkit-face-detection` e as coordenadas do ecrã real do dispositivo móvel, que são usadas para calcular se o rosto está dentro da oval. Para resolver este problema, foi necessário desenvolver um conjunto adicional de funções que realizassem a conversão das coordenadas geradas pelo *plugin* para as coordenadas do ecrã real, permitindo que fossem corretamente utilizadas.

Além disso, o *plugin* realiza outras verificações, como confirmar se o utilizador tem os olhos abertos, se o rosto está virado para a frente so ecrã e se não está a sorrir. Se alguma destas condições não for satisfeita, são exibidas mensagens de erro e a oval fica vermelha de forma a alertar o utilizador, conforme ilustrado na figura 5.51.

Além disso, realiza-se ainda uma outra validação para verificar se o telemóvel está na posição vertical ou horizontal correta, de acordo com a orientação desejada pelo utilizador para a fotografia, conforme ilustrado na figura 5.52. Para isso, utiliza-se o *plugin* Flutter `sensors-plus`[25], que fornece acesso ao acelerómetro do dispositivo móvel. O acelerómetro mede a aceleração linear ao longo dos três eixos (x, y e z), permitindo observar as alterações de velocidade e direção e verificar a rotação do ecrã [28].

Assim, ao detectar uma alteração na rotação superior a um determinado valor, é possível afirmar com certeza que o ecrã não está alinhado corretamente. Neste caso, a fotografia tirada fica inválida.

Se todas estas verificações forem bem-sucedidas, a oval torna-se verde, como ilustrado

[25] https://pub.dev/packages/sensors_plus

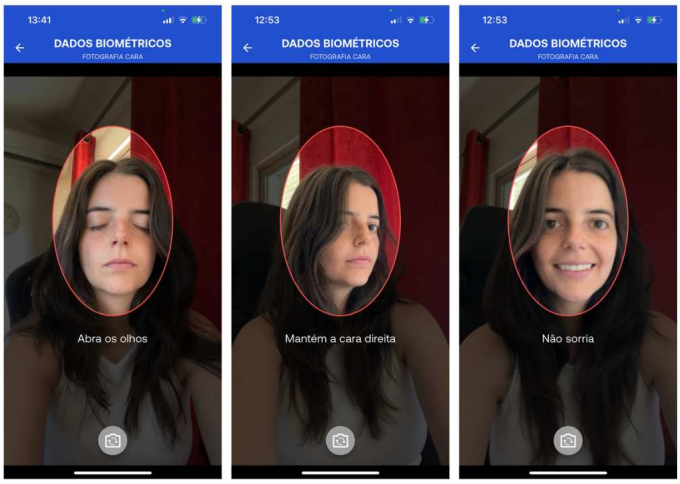


Figura 5.51: Página inicial para a extração da fotografia da face com os erros dos olhos fechados, posição correta da face e sorriso

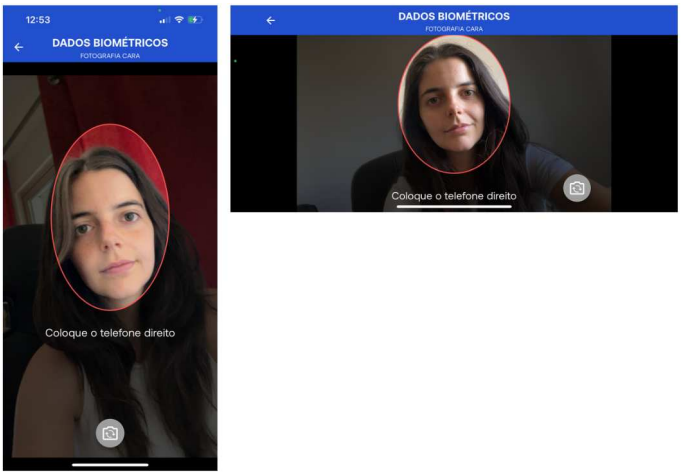


Figura 5.52: Página inicial para a extração da fotografia da face com o erro de orientação do ecrã

na figura 5.53.

Após alguns segundos, se a oval permanecer verde sem erros detectados, a fotografia é tirada e a página de confirmação da foto é exibida ao utilizador, conforme ilustrado na figura 5.54.

Esta página apresenta a fotografia tirada e oferece duas opções ao utilizador: descartar a fotografia, retornando à página anterior para tirar uma nova; ou validar a fotografia e prosseguir para o próximo passo da extração biométrica.

Se a opção de validar a fotografia for escolhida, a imagem é enviada para um *endpoint* específico do projeto *idfyme-plugins*. Este *endpoint* utiliza processos mais detalhados e intensivos de inteligência artificial para avaliar e confirmar a fotografia, já anteriormente implementados no produto. Após a análise e chegada da sua resposta, uma mensagem de sucesso ou erro é exibida ao utilizador apartir de um *popup*, conforme ilustrado na figura 5.55.

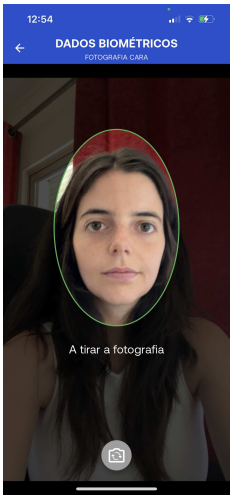


Figura 5.53: Página inicial quando a extrair a fotografia da face

Figura 5.54: Página para visualização da fotografia capturada

Se a fotografia for aceite, o utilizador, ao carregar no botão `Continuar`, é redirecionado para a página seguinte do processo para realizar a assinatura. Caso contrário, é exibida uma mensagem informando o que deu de errado com a fotografia. Nesta situação, ao clicar no botão `Tentar Novamente`, o utilizador é redirecionado para a página anterior para tirar a fotografia novamente.

Esta avaliação preliminar da fotografia no *frontend*, através destes *plugins*, oferece uma grande vantagem na aceleração do processo de captura da imagem. Ao identificar e resolver a maioria dos problemas durante a sua captura inicial, evita-se a necessidade de repetir o processo de extração da fotografia e o seu envio e avaliação para o *backend* várias vezes. Isto não só melhora a eficiência do processo como também aumenta a satisfação do utilizador, reduzindo o tempo total necessário para obter uma fotografia adequada e minimizando a frustração associada a repetições e correções.

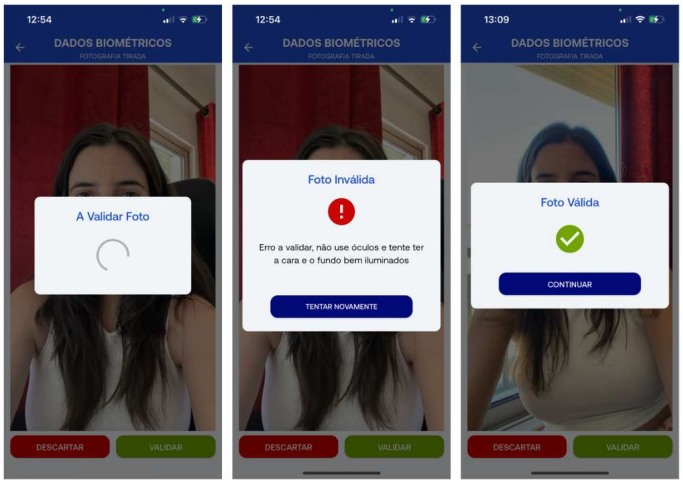


Figura 5.55: *Popup* com a mensagem de sucesso ou erro da validação da fotografia

Para capturar a fotografia, foi utilizada o *plugin* Flutter `camera`[26], que fornece uma interface para a câmara do dispositivo móvel. Esta biblioteca permite tanto a captura de fotografias quanto a visualização em direto da imagem captada pela câmara, permitindo ao utilizador ver o que está a ser capturado antes de tirar a foto.

Além do *plugin* `camera`, foi também utilizada a biblioteca `image`[27], já mencionada anteriormente. Esta biblioteca é responsável por converter a fotografia para o formato JPEG — o único formato autorizado pelo projeto *idfyme-plugins* para a avaliação posterior da imagem — e por codificá-la em formato `base64`, o que melhora a segurança e a eficiência da sua transmissão.

Assinatura Manuscrita Digital

O próximo passo na extração biométrica é a captura da assinatura digital manuscrita do utilizador. Para tal, foi utilizada o *plugin* Flutter `signature`[28], que fornece uma interface para a captura de assinaturas manuais digitalmente diretamente no ecrã do dispositivo móvel, como ilustrado na figura 5.56. Este *plugin* permite ao utilizador desenhar a sua assinatura diretamente no ecrã, utilizando o dedo ou uma caneta para dispositivos móveis. A assinatura desenhada é então capturada como uma imagem no formato PNG.

Esta página possui dois botões: um para limpar o campo da assinatura escrita e outro para submeter a assinatura. Se o campo estiver vazio, o botão de limpar fica desativado e o botão de submeter é utilizado para avançar no processo de extração, considerando que a funcionalidade de extração da assinatura não é obrigatória para utilizadores que sejam funcionários. Caso haja uma assinatura escrita, o botão de submeter torna-se ativo e serve para submeter e guardar a assinatura.

[26] https://pub.dev/packages/camera
[27] https://pub.dev/packages/image
[28] https://pub.dev/packages/signature

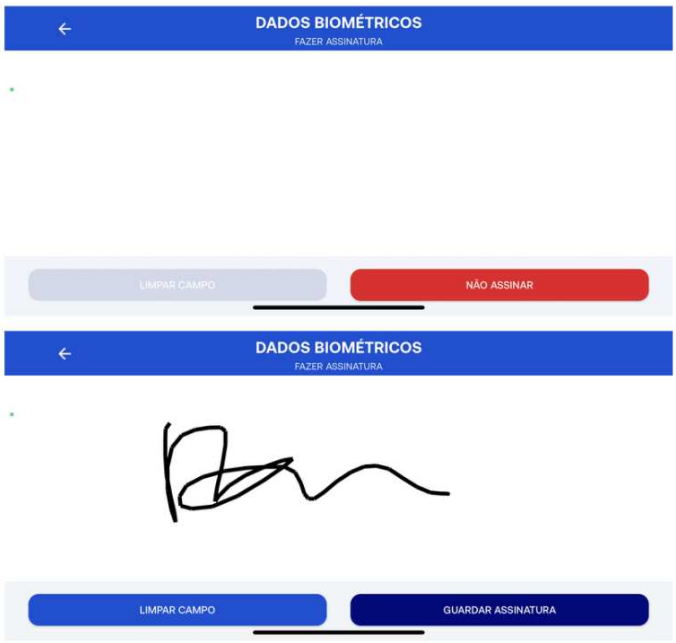


Figura 5.56: Página para a captura da assinatura digital com e sem o campo em branco

Após a submissão da assinatura e utilizando novamente a biblioteca `image`[29], a imagem originalmente no formato PNG é convertida para o formato JPEG. Este processo de conversão é necessário uma vez que o projeto *idfyme-plugins* só aceita imagens neste formato para avaliação posterior. Adicionalmente, a imagem JPEG é ainda codificada em `base64` para garantir uma transmissão e armazenamento mais eficiente e seguro.

Página de Confirmação dos Dados

Com a assinatura submetida ou passada, o utilizador é redirecionado para a página seguinte do processo, onde é exibida uma página de confirmação dos dados já extraídos. Nela, o utilizador pode visualizar a fotografia da face e a assinatura, se a tiver realizado, conforme ilustrado nas figuras 5.57 e 5.58. Nesta página, o utilizador tem a opção de confirmar os dados, repetir cada extração, ou cancelar o processo completamente, eliminando as extrações realizadas e retornando ao menu principal.

Se optar por confirmar os dados, o utilizador é redirecionado para a página de PIN, onde deve inserir o seu PIN, conforme ilustrado na figura 5.59, caso tenha escolhido esta opção de autenticação no *login*. Caso contrário, em vez de introduzir o PIN, será solicitada a autenticação biométrica semelharmente ao *login*. Este passo serve para reforçar a segurança dos dados submetidos e, assim, finalizar o processo.

Se o PIN ou a autenticação biométrica for inserida com sucesso, os dados são enviados através de um *endpoint*, para o projeto *idfyme-plugins* para a sua avaliação e armazenamento. Enquanto a aplicação aguarda a resposta deste pedido, o que pode demorar devido à avaliação posterior dos dados submetidos, é exibida uma mensagem ao utilizador

[29] `https://pub.dev/packages/image`

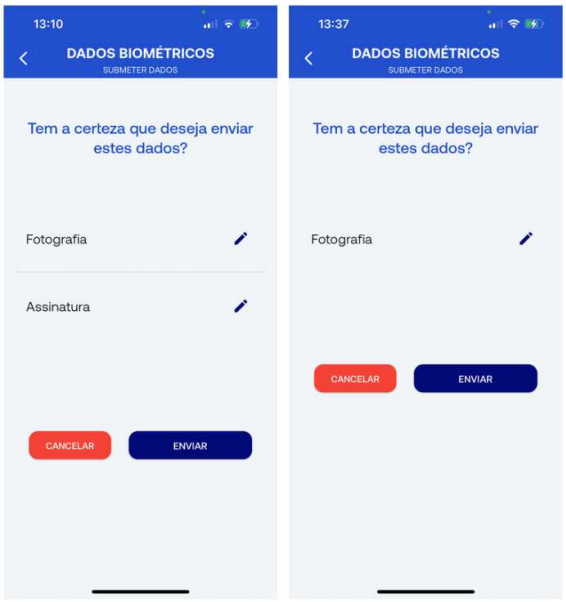


Figura 5.57: Página para a confirmação dos dados extraídos

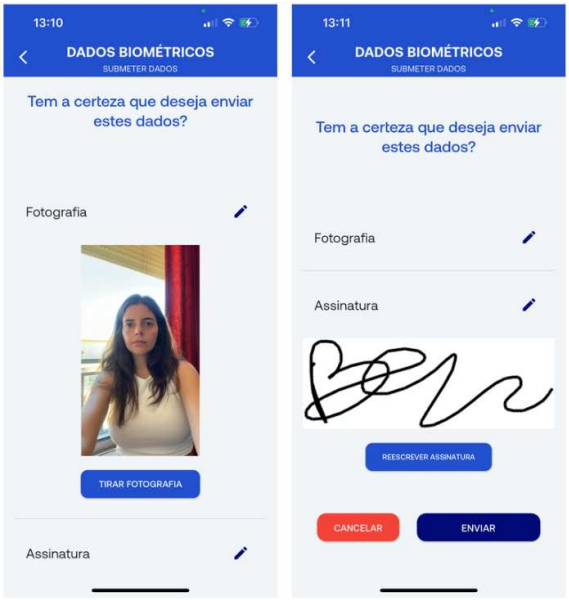


Figura 5.58: Página para a confirmação dos dados extraídos aberto com a opção de repetir a extração visível

informando que a submissão dos dados está a ser processada, conforme ilustrado na Figura 5.60.

Se a avaliação for bem sucedida, o utilizador é redirecionado para o menu inicial, e uma mensagem de sucesso é exibida, conforme mostrado na figura 5.61. Caso contrário, é exibida uma mensagem de erro e o utilizador é redirecionado de volta para a página de confirmação dos dados, conforme ilustrado na figura 5.62.

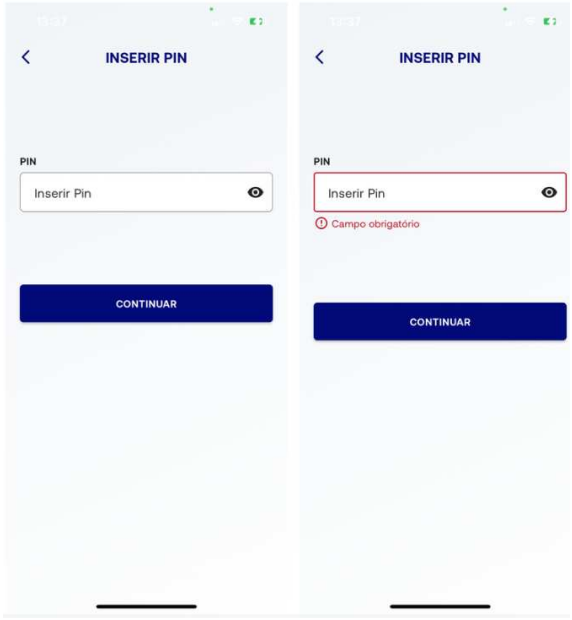


Figura 5.59: Página para a inserção do PIN com ilustração de erros

Figura 5.60: *Popup* a informar que a submissão dos dados está a ser processada

5.3.5 Assinatura de Documentos

O último tipo de processo incluído na solução é o de assinatura de documentos. Este processo permite ao utilizador assinar manualmente documentos digitais, como contratos, autorizações, declarações, entre outros, diretamente na aplicação móvel.

O processo oferece a possibilidade de o utilizador visualizar tanto o documento a assinar como o documento já assinado, conforme o estado do processo, que varia de acordo com os estados ilustrados na figura 5.63. Caso o processo se encontre nos estados de `Aguarda Validação` ou `Validado`, o utilizador poderá visualizar o documento original e o assinado. Se o processo estiver no estado `Aguarda Assinatura`, o utilizador poderá só proceder à assinatura após visualizar o documento.

Figura 5.61: *Popup* a informar que os dados biométricos foram submetidos com sucesso

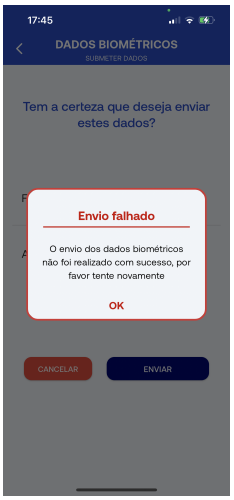


Figura 5.62: *Popup* a informar que os dados biométricos não foram submetidos com sucesso

Desta forma, o processo apresenta dois fluxos distintos: **Visualização de Documentos** e **Assinatura de Documentos**.

Vizualização de Documentos

Após o utilizador carregar nos botões `Ver Documento Original` ou `Ver Documento Assinado` disponíveis na página do processo, a aplicação realiza um pedido `GET` ao projeto *idfyme-plugins* para obter o documento correspondente. Este pedido é efetuado através de um *endpoint* específico do projeto *idfyme-plugins*, que retorna um modelo contendo o documento original e o assinado (caso exista) em formato PDF, codificado em `base64`, de forma a melhorar a segurança e a eficiência da sua transmissão.

Após a receção do documento, a aplicação móvel decodifica-o para o formato PDF e exibe-o ao utilizador.

Se ocorrer algum erro durante o processo de obtenção ou exibição do documento, é

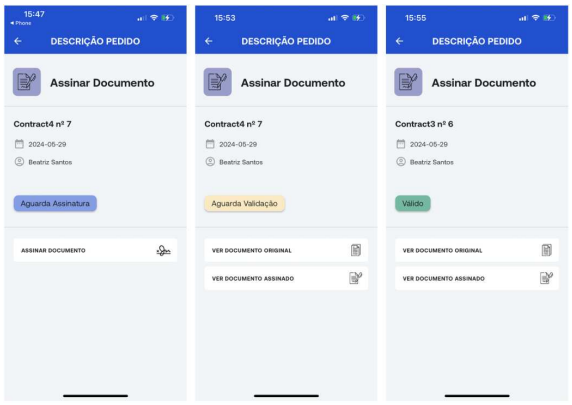


Figura 5.63: Diferentes estados do processo de assinatura de documentos

apresentada uma mensagem de erro ao utilizador, informando-o do problema e sugerindo que tente novamente mais tarde, conforme ilustrado na figura 5.64.

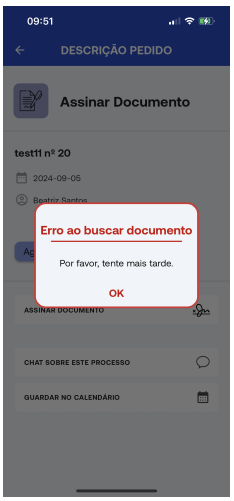


Figura 5.64: *Popup* a informar que houve um erro ao carregar o documento

Para a visualização do documento, foi utilizado o *plugin* Flutter `alh-pdf-view`[30], que permite exibir documentos PDF diretamente na aplicação móvel, além de disponibilizar funcionalidades adicionais, como a exibição do número de páginas do documento, navegação entre páginas, *zoom*, entre outras.

Este *plugin* foi escolhido por se basear no *plugin* de renderização de documentos mais popular e amplamente utilizado no Flutter, o `flutter-pdfview`[31], estudado e analisado no capítulo 2 do Estado da Arte. No entanto, ao contrário do `flutter-pdfview`, este *plugin* oferece funcionalidades adicionais identificadas após a sua análise, que são relevantes para a presente implementação, como a capacidade de manipular o *zoom* do documento.

Assim o utilizador pode visualizar o documento original e o assinado, se existir,

[30] https://pub.dev/packages/alh_pdf_view
[31] https://pub.dev/packages/flutter_pdfview

conforme ilustrado nas figuras 5.65 e 5.66.

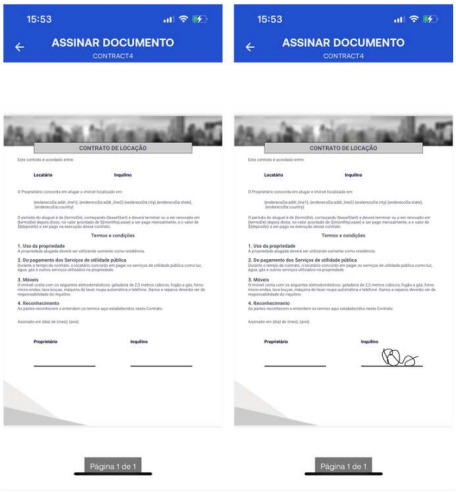


Figura 5.65: Páginas de visualização de um documento original e assinado com 1 página

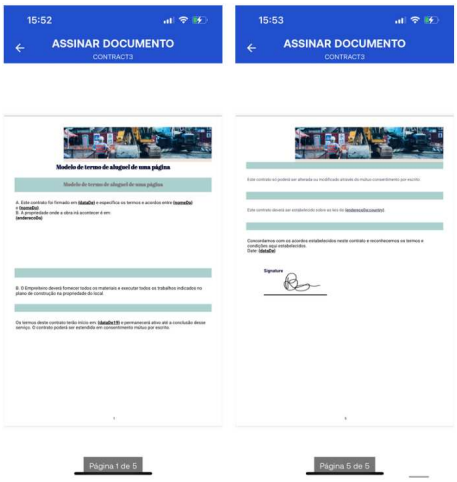


Figura 5.66: Páginas de visualização de um documento original e assinado com múltiplas páginas

Assinatura de Documentos

Para a assinatura de documentos, o processo apresenta um fluxo mais complexo, que se divide nas seguintes etapas:

- Pré-visualização do documento;
- Escolha do local no documento para assinar;
- Extração da assinatura manuscrita;
- Visualização do documento assinado e a sua confirmação;

- Autenticação do utilizador.

Pré-visualização do documento

Quando o utilizador carrega no botão `Assinar Documento` na página principal do processo ilustrada na figura 5.67, é redirecionado para a página de pré-visualização do documento, já explicada anteriormente, onde pode visualizar o documento original antes de o assinar, conforme ilustrado na figura 5.68. A diferença desta página em relação à explicada anteriormente, que apenas permite a visualização, é a adição de um botão que permite ao utilizador prosseguir com a assinatura do documento.

Figura 5.67: Página do processo para assinar documento

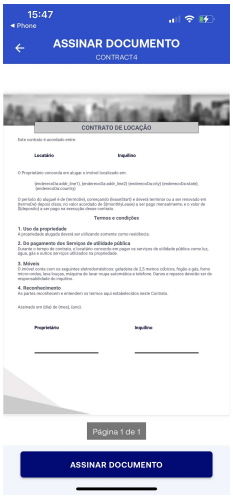


Figura 5.68: Página de visualização do documento por assinar

Escolha do local no documento para assinar

Se o utilizador decidir assinar o documento, carrega no botão `Assinar Documento` e é redirecionado para a página de escolha do local da assinatura, conforme ilustrado na figura 5.69. Nesta página, o documento é exibido com um retângulo móvel, que indica o local onde a assinatura será colocada. O utilizador pode navegar até à página desejada do documento e mover o retângulo para o local pretendido.

Teve-se também o cuidado de garantir que este retângulo não pode ser posicionado fora dos limites do documento. Para isso, foi necessário implementar uma verificação da posição do retângulo no ecrã do utilizador em relação ao tamanho e posição do documento exibido, assegurando que este não ultrapassasse os seus limites.

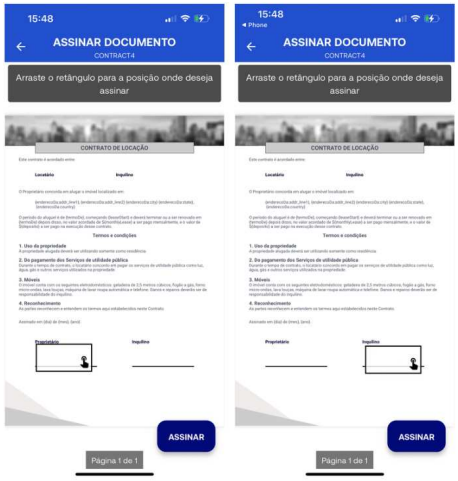


Figura 5.69: Página de escolha do local da assinatura com retângulo móvel

Extração da assinatura manuscrita

Ao pressionar o botão `Assinar`, o utilizador é redirecionado para a página de extração da assinatura, conforme ilustrado na figura 5.70. Esta página é a mesma utilizada para a recolha dos dados biométricos, explicada anteriormente na secção 5.3.4.

Nesta página, o utilizador pode optar por não assinar, clicando no botão `Não Assinar`, que só está ativo enquanto o campo de assinatura estiver vazio, voltando assim à página anterior. Se o utilizador desenhar a sua assinatura, os botões `Limpar` e `Guardar Assinatura` ficam ativos, sendo responsáveis, respetivamente, por limpar o campo de assinatura e por guardar a assinatura, redirecionando o utilizador para a página seguinte de visualização do documento assinado.

Visualização do documento assinado e a sua confirmação

Na página de visualização do documento assinado, ilustrada na figura 5.71, o utilizador tem ainda a opção de cancelar o processo, retornando ao menu principal através do botão `Cancelar`, limpar a assinatura realizada, utilizando o botão `Limpar`, que o redireciona novamente para a página de escolha do local da assinatura, ou confirmar a assinatura, clicando no botão `Continuar`.

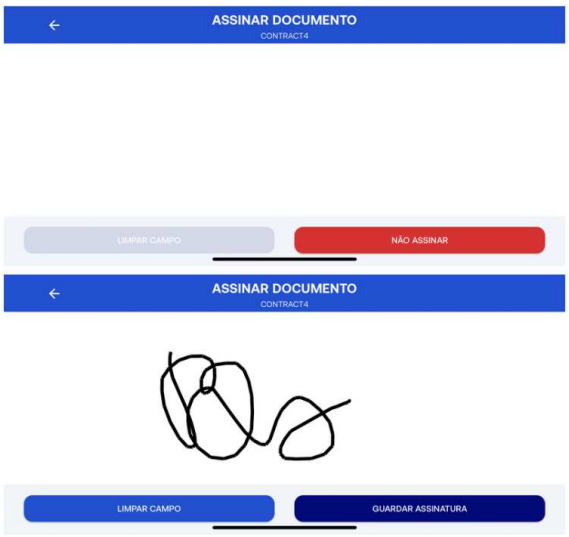


Figura 5.70: Página de extração da assinatura

Nesta página, teve-se também o cuidado de apresentar o documento assinado exatamente na página onde a assinatura foi colocada, para que o utilizador pode-se verificar de forma simples e sem ambiguidades que a assinatura estava no local correto e conforme o desejado.

Figura 5.71: Página de visualização do documento assinado

Após um estudo de várias bibliotecas e *plugins* do Flutter, concluiu-se que esta *framework* não possuia uma biblioteca que realiza-se simultaneamente a renderização/exibição e a manipulação de documentos PDF para a adição de imagens. Assim, foi necessário recorrer a duas abordagens distintas para estas implementações. Para a renderização/exibição, utilizou-se o *plugin* explicado anteriormente, o `alh-pdf-view`[32], e para a manipulação dos

[32] https://pub.dev/packages/alh_pdf_view

documentos foi utilizada a biblioteca `syncfusion-flutter-pdf`[33], que permite criar, ler e editar ficheiros PDF sem dependência adicional de qualquer outra plataforma, oferecendo ainda funcionalidades como a adição de texto formatado, imagens, formas, tabelas, listas, cabeçalhos, rodapés, entre outros.

Após o utilizador confirmar a sua assinatura na página de extração, a imagem desta é extraída no formato PNG com fundo transparente, para que possa ser inserida no documento no local escolhido pelo utilizador anteriormente através do retângulo.

Como a imagem da assinatura devia de ser adicionada na posição e com as dimensões exatas do retângulo pré-definido, foi necessário converter a escala da posição deste retângulo no ecrã do dispositivo móvel do utilizador para a posição correspondente no documento PDF real a ser editado.

Esta conversão adicionou uma complexidade extra à implementação, uma vez que a posição do retângulo variava conforme a dimensão do documento, o tamanho do ecrã do utilizador, o nível de *zoom* aplicado e a orientação do ecrã do dispositivo móvel (horizontal ou vertical). Assim, devido à elevada complexidade associada ao cálculo desta escala, optou-se por simplificar o processo de extração da assinatura, desativando a funcionalidade de *zoom* e restringindo a orientação do ecrã ao modo vertical na página de escolha da localização da assinatura.

O cálculo da escala foi então realizado de forma a converter as coordenadas do retângulo no ecrã do dispositivo móvel para as coordenadas correspondentes no documento real, de modo a determinar a posição exata onde a assinatura seria colocada no documento PDF.

No entanto, surgiu um desafio adicional: como o retângulo era sobreposto e realizado à parte da exibição do documento, o cálculo das coordenadas em relação ao documento exibido teve de ser feito manualmente, sem o apoio da biblioteca `alh-pdf-view`, responsável pela renderização do documento. Foi então necessário desenvolver um conjunto de cálculos que determinassem as coordenadas exatas onde o documento começava e terminava no ecrã, de modo a conseguir calcular as coordenadas do retângulo em relação a este, e não ao ecrã do dispositivo móvel.

Por fim, foi necessário determinar as dimensões reais do documento PDF a ser editado e adicionar a ele a imagem da assinatura.

Para esta edição, utilizou-se a biblioteca `syncfusion-flutter-pdf`, que permitiu a leitura do ficheiro PDF e a sua edição, com a adição da imagem PNG da assinatura na página e local desejados. Através desta biblioteca, foi possível obter as dimensões exatas do documento, e, juntamente com os valores calculados anteriormente, determinar a escala das coordenadas da localização da assinatura no documento vizualizado no ecrã e convertê-las para o tamanho real do documento.

[33] https://pub.dev/packages/syncfusion_flutter_pdf

Autenticação do utilizador

Após a confirmação do documento assinado, o utilizador é redirecionado para a página de introdução do PIN, ilustrada na figura 5.59, com o objetivo de realizar a sua autenticação e reforçar a segurança dos dados. Caso o utilizador tenha optado pela autenticação biométrica durante o *login*, esta será solicitada em vez da introdução do PIN; caso contrário, o utilizador terá de introduzir o PIN manualmente.

Se o PIN for inserido com sucesso, o documento é enviado para validação e armazenamento através de um *endpoint* do projeto *idfyme-plugins*.

Enquanto a aplicação aguarda a resposta do pedido, é exibida uma mensagem a informar que a submissão dos dados está a ser processada, conforme ilustrado na figura 5.72.

Figura 5.72: *Popup* a informar que a submissão dos dados está a ser processada

Caso os dados sejam armazenados com sucesso, o utilizador é redirecionado para o ecrã inicial e uma mensagem de sucesso é exibida, conforme ilustrado na figura 5.73.

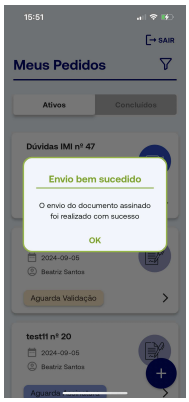


Figura 5.73: *Popup* a informar que o documento assinado foi submetido com sucesso

Caso contrário, uma mensagem de erro é apresentada, e o utilizador é redirecionado de volta para a página de confirmação da assinatura, conforme ilustrado na figura 5.74.

Uma vez que a validação dos documentos é realizada por um funcionário, que aprova ou rejeita a assinatura, o processo é imediatamente atualizado no servidor, e o seu estado

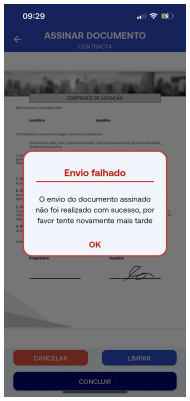


Figura 5.74: *Popup* a informar que o documento assinado não foi submetido com sucesso

muda automaticamente para `Aguarda Validação`. Posteriormente, se o funcionário validar o documento, este passará para o estado `Validado`; caso seja rejeitado, o processo retorna ao estado `Aguarda Assinatura`, e o utilizador terá de realizar novamente a assinatura.

5.4 Síntese

Neste capítulo, foi descrito em detalhe a implementação da solução proposta, incluindo os desafios técnicos enfrentados e as estratégias utilizadas para os superar, no contexto do problema apresentado pelo produto **idfyme**. A solução teve origem no desenvolvimento de uma aplicação multiplataforma em Flutter, representada pelo projeto *idfyme-flutter*. Adicionalmente, foram também descritas as implementações e otimizações realizadas num outro projeto do produto, o *idfyme-plugin*, que interage diretamente com o *idfyme-flutter* sendo o responsável pelo *backend* do produto.

Foram implementadas diversas estratégias no projeto *idfyme-flutter* para melhorar a solução, incluindo a arquitetura MVVM com `Services` dedicados, *design* responsivo, gestão de temas e internacionalização, estados da aplicação, e permissões necessárias ao dispositivo móvel. Também se destacou a gestão modular de classes, navegação dinâmica entre páginas, construção reutilizável dos pedidos HTTP para a comunicação com o servidor (projeto *idfyme-plugin*), gestão das exceções do projeto, *logs* e mensagens detalhadas de erros e de sucesso. A utilização de uma base de dados local assegurou ainda o armazenamento seguro dos dados do utilizador.

No projeto *idfyme-plugin* já desenvolvido anteriormente no produto, focado principalmente na integração com o servidor, foram introduzidas melhorias em *endpoints* e otimizações na gestão de mensagens de erro, garantindo uma experiência de utilizador mais personalizada e eficiente na aplicação móvel.

Foram ainda explicadas em pormenor as funcionalidades da solução, a autenticação de utilizadores, a exibição do ecrã inicial com o histórico de processos, a realização e agendamento de videoconferências, a recolha de dados biométricos e a assinatura

manuscrita de documentos digitais, assim como as suas implementações na aplicação móvel.

A autenticação na aplicação envolveu a implementação do registo e *login* de utilizadores, assim como a adição de um PIN ou de autenticação biométrica para aumentar a segurança e simplificar o processo de voltar a entrar na aplicação. Esta solução permitiu que o utilizador, ao sair da aplicação, não necessitasse de realizar o *login* completo novamente, bastando apenas introduzir o seu PIN ou utilizar a autenticação biométrica, conforme a sua preferência.

O *design* do ecrã inicial foi cuidadosamente planeado, separando os processos em categorias de `Ativos` e `Concluídos` e oferecendo filtros para facilitar a sua navegação. Além disso, foram implementadas diversas mensagens de erro que cobrem potenciais falhas e incertezas. O *layout* simples e intuitivo permitiu ainda que os utilizadores acedessem facilmente a detalhes adicionais sobre cada processo, com diferentes ações possíveis consoante o estado de cada um.

A funcionalidade de videoconferência foi um dos pilares da solução, possibilitando a comunicação remota entre utilizadores e funcionários. Esta funcionalidade inclui o agendamento de videoconferências, permitindo que o utilizador escolhesse o tipo, a data e a hora que pretendia. A integração do servidor de sinalização, *sockets*, e WebRTC garantiu uma transmissão de vídeo e áudio segura e eficiente, enquanto os servidores STUN e TURN asseguraram uma conectividade estável mesmo em redes complexas. Foi estabelecida uma distinção entre videoconferências normais e de autenticação, sendo que as últimas incorporam funcionalidades adicionais, como a extração de dados biométricos e fotografias de documentos. Além disso, possuem uma camada extra de segurança, incluindo a validação do ambiente do utilizador, a utilização de um código de verificação (*token*) e a confirmação biométrica através de perguntas de segurança.

A extração de dados biométricos, como a extração de fotografias de rosto e assinaturas digitais manuscritas, foi implementada de forma eficaz, com a utilização de um *plugin* de inteligência artificial para realizar uma validação preliminar da fotografia. Esta abordagem permitiu que as imagens fossem validadas localmente, aliviando o servidor de processamentos desnecessários.

A assinatura de documentos em formato PDF foi outra funcionalidade essencial. Os utilizadores podem pré-visualizar documentos, selecionar o local exato para assinatura e proceder à assinatura manuscrita diretamente na aplicação. A implementação desta funcionalidade envolveu a integração de duas bibliotecas e *plugins* distintos para renderizar e manipular documentos PDF, o que permitiu processar assinaturas de forma eficiente e garantir a validade das transações realizadas. A extração da assinatura manuscrita digitalmente foi implementada ainda de forma intuitiva e simplificada, de forma a ser desenhada diretamente no ecrã do dispositivo móvel.

Para garantir a segurança da submissão dos dados biométricos e da assinatura dos documentos, implementou-se ainda a necessidade do utilizador inserir o seu PIN ou autenticação biométrica antes de concluir o processo, reforçando assim a proteção destes

dados sensíveis.

Em resumo, a implementação técnica atingiu todos os objetivos propostos, proporcionando uma solução integrada e segura para a realização de tarefas complexas de forma remota. A escolha adequada das tecnologias e o foco na experiência do utilizador garantiram uma aplicação fácil de usar e altamente segura, inspirando confiança nos serviços oferecidos e permitindo a sua escalabilidade futura.

6

Qualidade Técnica da Solução

Para assegurar a maior qualidade e segurança da solução, foram implementadas diversas técnicas e práticas de desenvolvimento. Isto incluiu a realização preliminar de User Stories, o uso de vários ambientes de desenvolvimento para testes e produção, a execução de uma variedade de testes funcionais e de integração e o uso do Fastlane para automatização de tarefas como a execução de testes e respetivos relatórios e a gestão eficiente de *builds*. As próximas secções abordarão em maior detalhe estas implementações e práticas.

Foram também aplicados padrões de *design* para manter a estrutura do código coesa e eficiente. Como mencionado na secção 5.1, a solução foi desenvolvida com base no padrão de *design* MVVM, que promoveu a separação de responsabilidades entre os diferentes componentes da aplicação e facilitou a manutenção e a evolução da solução.

Além disso, foram ainda implementadas técnicas avançadas de gestão de erros, como mencionado na secção 5.2. Isto incluiu a criação de *logs* detalhados para facilitar o diagnóstico de problemas na aplicação móvel e a definição de exceções e mensagens de erro claras. Estas mensagens de erro foram estruturadas num formato JSON, permitindo que a aplicação móvel exibisse *feedbacks* precisos e adaptados às exceções e erros recebidos dos pedidos HTTP realizados. Esta abordagem não apenas aumentou a transparência e usabilidade da aplicação, como também ajudou os utilizadores a compreenderem melhor a natureza dos problemas ocorridos e a tomarem medidas mais adequadas para os resolver diretamente.

6.1 User Stories

O desenvolvimento da solução começou com a elaboração de User Stories, assegurando desta forma um produto final de melhor qualidade.

User Stories (histórias de utilizador) são descrições concisas e informais de características ou funcionalidades para a aplicação na perspetiva dos utilizadores finais ou clientes. Estas seguem um formato simples, incluindo quem é o utilizador, o que pretende realizar e porquê. Estas histórias são um meio eficaz para capturar e comunicar os requisitos do utilizador/cliente de maneira clara e compreensível. Elas concentram-se nas necessidades

e objetivos do mesmo, evitando detalhes técnicos excessivos, permitindo identificar de forma clara e simples o que precisa de ser desenvolvido na solução para satisfazer os requisitos do utilizador [56].

Ao agrupar várias User Stories relacionadas, formam-se épicos, que são blocos de trabalho mais amplos e complexos que representam funcionalidades abrangentes que não podem ser concluídas rapidamente. Estes épicos permitem uma visão geral dos grandes objetivos ou temas do projeto, que posteriormente são desmembrados em User Stories mais específicas e geríveis [56].

As User Stories realizadas estão ilustradas na figura 6.1, organizadas nos diferentes épicos considerados para a solução.

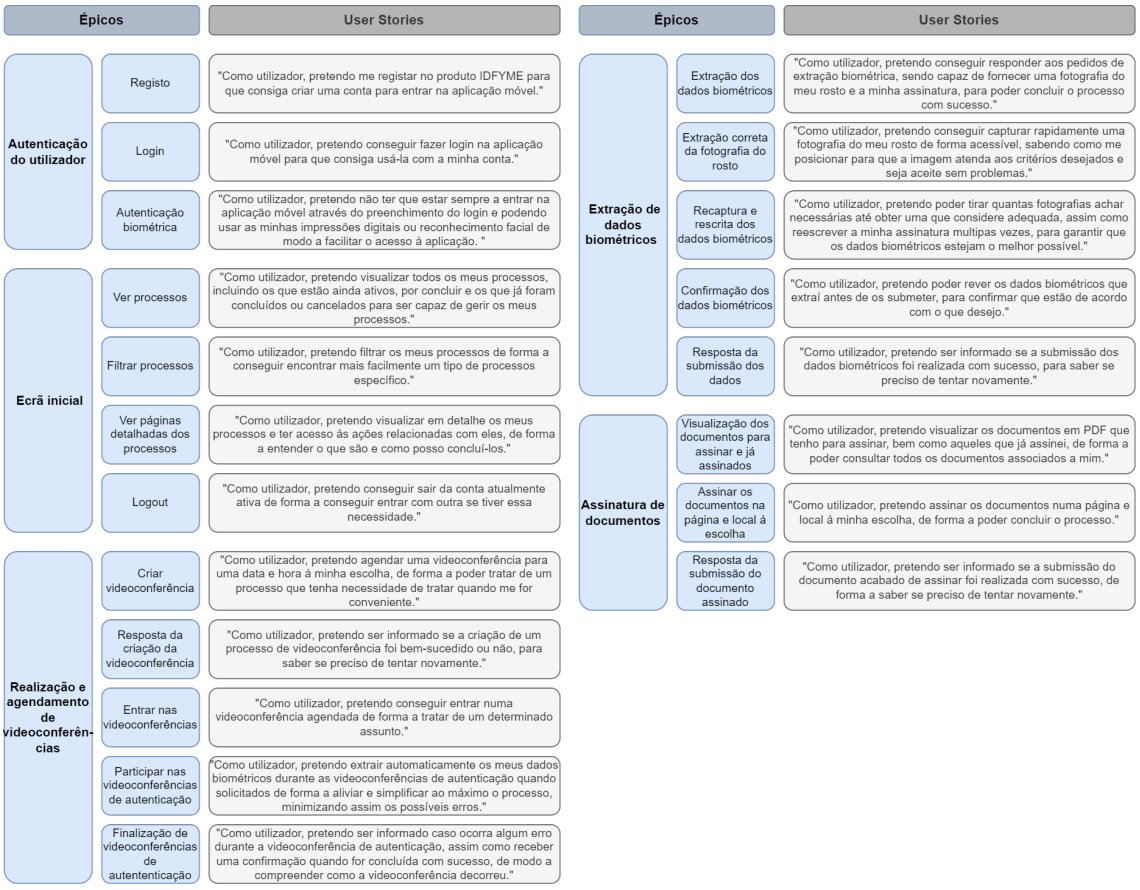


Figura 6.1: User Stories realizadas para o desenvolvimento da solução

As vantagens do seu uso foram múltiplas. As User Stories mantiveram o foco no utilizador/cliente ao descreverem características diretamente relacionadas às suas necessidades, aumentando a sua satisfação e o potencial sucesso do produto. Elas também facilitaram a comunicação com o cliente, promovendo um entendimento partilhado e colaboração eficiente e garantindo que tanto o cliente como os programadores tivessem voz no desenvolvimento do produto. Além disso, elas são flexíveis e adaptáveis, o que permitiu que fosse possível fazer ajustes conforme necessário ao longo do desenvolvimento da

solução. As User Stories também proporcionaram uma maneira tangível de acompanhar o progresso do projeto e identificar problemas precocemente. Por fim, elas foram uma ferramenta valiosa para a priorização e definição de funcionalidades, garantindo que as mais importantes fossem desenvolvidas primeiro e que nenhuma delas faltasse [56].

6.2 Ambientes de Desenvolvimento

O uso de múltiplos ambientes no desenvolvimento na solução foi essencial para garantir a qualidade, estabilidade e eficiência da aplicação. Estes ambientes, denominados debug e desenvolvimento serviram a diferentes propósitos e ajudaram a isolar etapas distintas do ciclo de vida de desenvolvimento da aplicação.

- **Ambiente de Debug:** Este é um ambiente isolado e local do programador, onde ocorre o desenvolvimento e teste inicial da solução em tempo real. Ele fornece *feedback* imediato sobre o comportamento do código, permitindo a rápida identificação e correção de *bugs*. Foi o primeiro ambiente onde a aplicação começou a ser desenvolvida devido à sua flexibilidade e dinamismo, facilitando a experimentação, teste e implementação rápida de novas funcionalidades.
- **Ambiente de Desenvolvimento:** Após a versão no ambiente de debug estar estável, ela é promovida para o ambiente de desenvolvimento. Este ambiente é usado para testar e validar as funcionalidades em condições mais realistas, embora ainda com uma infraestrutura mínima, suficiente para realizar um primeiro nível de testes. Neste estágio, a aplicação e as suas alterações foram experimentadas e testadas por outros membros da empresa, permitindo uma avaliação mais abrangente [57].

O uso destes ambientes distintos permitiu uma transição suave e segura no desenvolvimento inicial da aplicação. Eles ajudaram a isolar problemas e erros em várias fases e em ambientes de diferentes comportamentos, garantindo que a solução fosse cuidadosamente validada. Esta abordagem minimizou então os riscos, os erros, e melhorou a qualidade do produto [57].

6.3 Testes

O uso de testes no desenvolvimento da solução foi uma prática fundamental que trouxe múltiplas vantagens, garantindo a qualidade e a confiabilidade do produto final.

A implementação de uma estratégia abrangente de testes envolve a execução de vários tipos de testes em diferentes fases do desenvolvimento, cada um com seu propósito específico. No entanto, como a solução desenvolvida diz respeito a uma aplicação móvel em Flutter, os testes foram conduzidos de acordo com as melhores práticas recomendadas para este tipo de aplicação. Foram realizados um total de 268 testes, abrangendo testes unitários, testes de integração, testes funcionais e testes de *widget*. Estes testes foram

projetados para avaliar o desempenho das diferentes páginas e componentes da aplicação, tanto de forma individual quanto em conjunto.

Dado que a solução se centrou no desenvolvimento de uma única componente do produto **idfyme**, especificamente numa camada relacionada à interação do cliente com o sistema numa plataforma móvel (o *frontend* correspondente ao projeto *idfyme-flutter*), não fazia sentido realizar testes para a parte funcional do produto que já havia sido previamente desenvolvida (o *backend* correspondente ao projeto *idfyme-plugins*). Por essa razão, os testes não tiveram como foco a verificação da segurança do produto, nem a sua capacidade de desempenho e resposta a ataques ou condições extremas de carga, uma vez que essa parte do projeto já estava concluída e devidamente testada.

Embora o *backend* tenha sido excluído desta fase de testes, a integração da solução com estes serviços já existentes foi cuidadosamente considerada realizando-se alguns testes unitários nos *endpoints* novos desenvolvidos no projeto *idfyme-plugins*.

Testes Unitários

Os testes unitários permitiram verificar se as pequenas partes do código funcionavam como esperado de forma isolada. Ao testar funções ou métodos individuais, foi possível identificar e corrigir erros desde as etapas iniciais do desenvolvimento, o que reduziu significativamente os custos e o tempo necessário para resolver problemas mais adiante. Além disso, com a automatização dos testes unitários também realizada, foi possível rapidamente validar alterações no código sempre que realizadas, garantindo uma base sólida para a solução [58].

Testes Widget

Os testes de *widget* foram uma forma de avaliar a UI da aplicação, composta por *widgets* individuais e agrupados. Estes testes garantiram que os *widgets* funcionavam conforme o esperado, verificando se respondiam corretamente às interações do utilizador e se o estado da aplicação era refletido de maneira adequada na UI [59].

Realizar este tipo de testes foi crucial para assegurar que a apresentação e a funcionalidade das páginas e os seus componentes atendiam aos requisitos, oferecendo uma experiência de usuário consistente e sem erros.

Testes Integração

Os testes de integração foram essenciais para assegurar que os diferentes módulos e componentes do sistema interagiam corretamente entre eles. No desenvolvimento da solução, onde múltiplos componentes precisavam de funcionar em conjunto, foi crucial garantir que a sua integração conjunta não introduziu erros. Estes testes ajudaram a identificar problemas de comunicação entre módulos, como incompatibilidades de interface ou erros de troca de dados, promovendo uma arquitetura de *software* mais robusta e coesa [58].

Testes Funcionais

Os testes funcionais verificaram o comportamento do sistema em relação aos requisitos especificados inicialmente, garantindo que todas as funcionalidades esperadas pelo cliente estivessem implementadas corretamente. Estes testes foram realizados em cenários que simulavam o uso real da aplicação, permitindo validar se a solução atendia às necessidades do utilizador final. Este tipo de testes centrado mais nos utilizadores/clientes finais assegurou que a solução oferecia valor real, melhorando a satisfação e a confiança dos mesmos no produto [58].

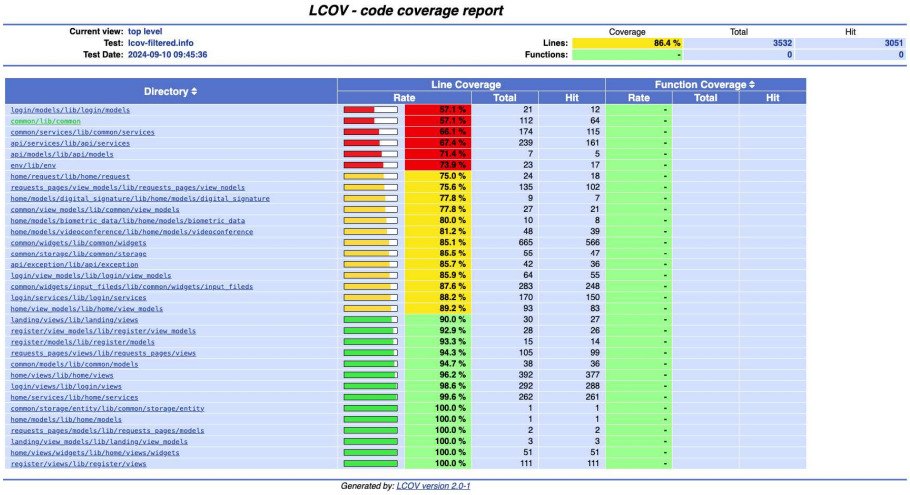
No final do desenvolvimento dos testes, foram ainda gerados gráficos de cobertura a partir do uso do Fastlane a ser explicado de seguida, que permitiram avaliar a eficácia dos testes realizados e identificar áreas do código que ainda não haviam sido cobertas. Estes gráficos foram cruciais para garantir que a solução fosse testada de forma abrangente e que todos os cenários de uso fossem considerados.

No entanto, surgiu um desafio adicional no desenvolvimento de testes relacionados a implementações que envolviam o uso da câmara e microfone do dispositivo assim como com a manipulação de PDFs.

A realização destes testes requeria a criação de testes altamente específicos e complexos, devido à necessidade de simular o comportamento de bibliotecas, *pulgins* e sensores integrados ao dispositivo móvel, como a câmara, o microfone e criadores e manipuladores de documentos PDF. Isto envolvia a criação de objetos simulados personalizados para estas bibliotecas e componentes, a fim de simular de maneira eficaz as interações com o *hardware* do dispositivo, garantindo que os testes pudessem ser executados num ambiente controlado e automatizado. Este processo tornava-se particularmente desafiador, já que era necessário garantir que as respostas simuladas refletissem adequadamente o comportamento do *hardware* real, preservando a precisão e a confiabilidade dos testes.

Assim, dado que o objetivo principal deste projeto era o desenvolvimento da plataforma móvel multiplataforma do produto *idfyme* e considerando o prazo limitado para a sua execução, optou-se por não implementar testes automáticos (testes de integração, funcionais, *widget* e unitários) para estas funcionalidades específicas, recorrendo-se, em vez disso, a testes manuais. O foco foi então direcionado para alcançar uma cobertura máxima destes testes automáticos nas restantes áreas da aplicação. Desta forma, foi alcançada uma cobertura de 86.4%, como ilustrado na figura 6.2, considerando o projeto sem essas classes, e de 56.9%, como ilustrado na figura 6.3, considerando o projeto completo.

Em resumo, este processo de testes meticuloso garantiu que a aplicação móvel fosse robusta e eficiente, pronta para ser lançada ao mercado com a confiança de que as suas funcionalidades principais operariam conforme esperado, proporcionando uma experiência positiva e sem interrupções aos utilizadores. A implementação de uma estratégia abrangente de testes no seu desenvolvimento proporcionou inúmeras vantagens,



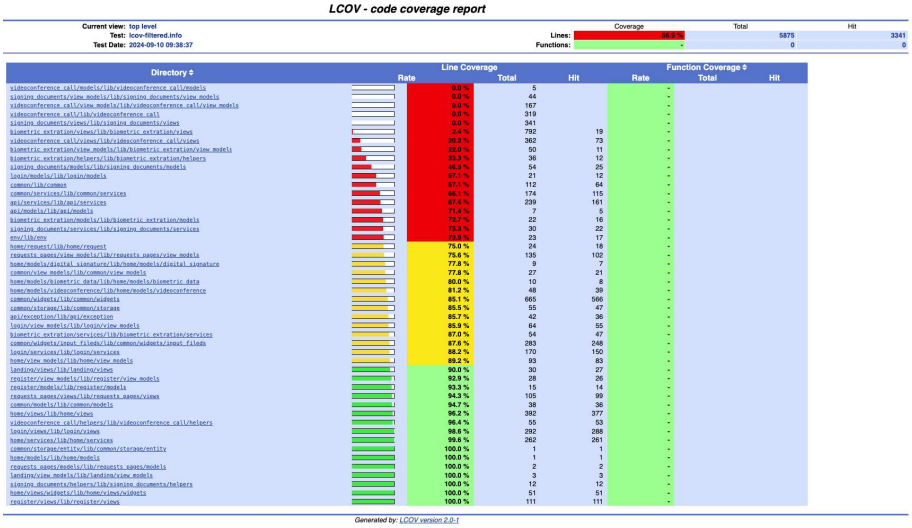
LCOV - code coverage report

		Coverage	Total	Hit
Current view:	top level			
Test:	lcov-filtered.info	Lines: 86.4 %	3532	3051
Test Date:	2024-09-10 09:45:36	Functions: -	0	0

Directory	Line Coverage Rate	Line Coverage Total	Line Coverage Hit	Function Coverage Rate	Function Coverage Total	Function Coverage Hit
login/models/lib/login/models	57.1 %	21	12	-		
common/lib/common	57.1 %	112	64	-		
common/services/lib/common/services	66.1 %	174	115	-		
api/services/lib/api/services	67.4 %	239	161	-		
api/models/lib/api/models	71.4 %	7	5	-		
env/lib/env	73.9 %	23	17	-		
home/request/lib/home/request	75.0 %	24	18	-		
requests_pages/view_models/lib/requests_pages/view_models	75.6 %	135	102	-		
home/models/digital_signature/lib/home/models/digital_signature	77.8 %	9	7	-		
common/view_models/lib/common/view_models	77.8 %	27	21	-		
home/models/biometric_data/lib/home/models/biometric_data	80.0 %	10	8	-		
home/models/videoconference/lib/home/models/videoconference	81.2 %	48	39	-		
common/widgets/lib/common/widgets	85.1 %	665	566	-		
common/storage/lib/common/storage	85.5 %	55	47	-		
api/exception/lib/api/exception	85.7 %	42	36	-		
login/view_models/lib/login/view_models	85.9 %	64	55	-		
common/widgets/input_fileds/lib/common/widgets/input_fileds	87.6 %	283	248	-		
login/services/lib/login/services	88.2 %	170	150	-		
home/view_models/lib/home/view_models	89.2 %	93	83	-		
landing/views/lib/landing/views	90.0 %	30	27	-		
register/view_models/lib/register/view_models	92.9 %	28	26	-		
register/models/lib/register/models	93.3 %	15	14	-		
requests_pages/views/lib/requests_pages/views	94.3 %	105	99	-		
common/models/lib/common/models	94.7 %	38	36	-		
home/views/lib/home/views	96.2 %	392	377	-		
login/views/lib/login/views	98.6 %	292	288	-		
home/services/lib/home/services	99.6 %	262	261	-		
common/storage/entity/lib/common/storage/entity	100.0 %	1	1	-		
home/models/lib/home/models	100.0 %	1	1	-		
requests_pages/models/lib/requests_pages/models	100.0 %	2	2	-		
landing/view_models/lib/landing/view_models	100.0 %	3	3	-		
home/views/widgets/lib/home/views/widgets	100.0 %	51	51	-		
register/views/lib/register/views	100.0 %	111	111	-		

Generated by: LCOV version 2.0-1

Figura 6.2: Gráfico de cobertura de testes do projeto da solução sem as classes específicas



LCOV - code coverage report

		Coverage	Total	Hit
Current view:	top level			
Test:	lcov-filtered.info	Lines: 58.9 %	5875	3341
Test Date:	2024-09-10 09:38:37	Functions: -	0	0

Directory	Line Coverage Rate	Line Coverage Total	Line Coverage Hit	Function Coverage Rate	Function Coverage Total	Function Coverage Hit
videoconference_call/models/lib/videoconference_call/models	0.0 %	5		-		
signing_documents/view_models/lib/signing_documents/view_models	0.0 %	44		-		
videoconference_call/view_models/lib/videoconference_call/view_models	0.0 %	167		-		
videoconference_call/lib/videoconference_call	0.0 %	319		-		
signing_documents/views/lib/signing_documents/views	0.0 %	341		-		
biometric_extration/views/lib/biometric_extration/views	2.4 %	792	19	-		
videoconference_call/views/lib/videoconference_call/views	20.2 %	362	73	-		
biometric_extration/view_models/lib/biometric_extration/view_models	22.0 %	50	11	-		
biometric_extration/helpers/lib/biometric_extration/helpers	33.3 %	36	12	-		
signing_documents/models/lib/signing_documents/models	46.3 %	54	25	-		
login/models/lib/login/models	57.1 %	21	12	-		
common/lib/common	57.1 %	112	64	-		
common/services/lib/common/services	66.1 %	174	115	-		
api/services/lib/api/services	67.4 %	239	161	-		
api/models/lib/api/models	71.4 %	7	5	-		
biometric_extration/models/lib/biometric_extration/models	72.7 %	22	16	-		
signing_documents/services/lib/signing_documents/services	73.3 %	30	22	-		
env/lib/env	73.9 %	23	17	-		
home/request/lib/home/request	75.0 %	24	18	-		
requests_pages/view_models/lib/requests_pages/view_models	75.6 %	135	102	-		
home/models/digital_signature/lib/home/models/digital_signature	77.8 %	9	7	-		
common/view_models/lib/common/view_models	77.8 %	27	21	-		
home/models/biometric_data/lib/home/models/biometric_data	80.0 %	10	8	-		
home/models/videoconference/lib/home/models/videoconference	81.2 %	48	39	-		
common/widgets/lib/common/widgets	85.1 %	665	566	-		
common/storage/lib/common/storage	85.5 %	55	47	-		
api/exception/lib/api/exception	85.7 %	42	36	-		
login/view_models/lib/login/view_models	85.9 %	64	55	-		
biometric_extration/services/lib/biometric_extration/services	87.0 %	54	47	-		
common/widgets/input_fileds/lib/common/widgets/input_fileds	87.6 %	283	248	-		
login/services/lib/login/services	88.2 %	170	150	-		
home/view_models/lib/home/view_models	89.2 %	93	83	-		
landing/views/lib/landing/views	90.0 %	30	27	-		
register/view_models/lib/register/view_models	92.9 %	28	26	-		
register/models/lib/register/models	93.3 %	15	14	-		
requests_pages/views/lib/requests_pages/views	94.3 %	105	99	-		
common/models/lib/common/models	94.7 %	38	36	-		
home/views/lib/home/views	96.2 %	392	377	-		
videoconference_call/helpers/lib/videoconference_call/helpers	98.4 %	55	53	-		
login/views/lib/login/views	98.6 %	292	288	-		
home/services/lib/home/services	99.6 %	262	261	-		
common/storage/entity/lib/common/storage/entity	100.0 %	1	1	-		
home/models/lib/home/models	100.0 %	1	1	-		
requests_pages/models/lib/requests_pages/models	100.0 %	2	2	-		
landing/view_models/lib/landing/view_models	100.0 %	3	3	-		
signing_documents/helpers/lib/signing_documents/helpers	100.0 %	12	12	-		
home/views/widgets/lib/home/views/widgets	100.0 %	51	51	-		
register/views/lib/register/views	100.0 %	111	111	-		

Generated by: LCOV version 2.0-1

Figura 6.3: Gráfico de cobertura de testes de todo o projeto da solução

como a deteção precoce de erros, a garantia de uma integração harmoniosa, a validação de funcionalidades e a conformidade com os requisitos do cliente [58].

6.4 Fastlane

Fastlane[1] é uma ferramenta de automação que facilita o processo de desenvolvimento e implementação de aplicações móveis, simplificando tarefas repetitivas e propensas a erros. Ela permite, a partir da configuração do ficheiro `fastfile` do Fastlane, automatizar fluxos de trabalho como testes, limpeza de código, criação de *builds* e distribuição de aplicações para lojas de aplicações como o Google Play e a App Store.

Ao ser usado em conjunto com o Flutter, o Fastlane ofereceu uma maneira eficiente de gerir o ciclo de vida da solução, garantindo que a aplicação fosse distribuída e testada de

[1] https://fastlane.tools/

forma consistente, rápida e automática, reduzindo significativamente o tempo e o esforço necessários para gerir estas operações.

6.4.1 Builds Automáticas

Como dito anteriormente o Fastlane é capaz de gerar *builds* e fazer a sua distribuição para lojas de aplicações apropriadas de forma automática. Na solução, o ficheiro `fastfile` configurado foi capaz de limpar o projeto, fazer a sua *build* e gerar as versões para iOS e Android respetivamente, fazendo o *upload* para o TestFlight no caso do iOS, e para a Play Store no caso do Android.

O TestFlight[2] é um serviço da Apple que permite distribuir versões beta das aplicações iOS para teste. Este serviço foi usado para testar a solução no decorrer do seu desenvolvimento com outros membros da Opensoft de modo a obter *feedbacks* na versão iOS da aplicação.

6.4.2 Análise de Código

O ficheiro `fastfile` implementado incorporou ainda uma configuração para executar análises de código, exibindo avisos e erros encontrados no projeto, como ilustrado na figura 6.4. Isto possibilitou a deteção precoce de problemas, automatizou a verificação da qualidade do código e garantiu consistência em cada alteração efetuada.

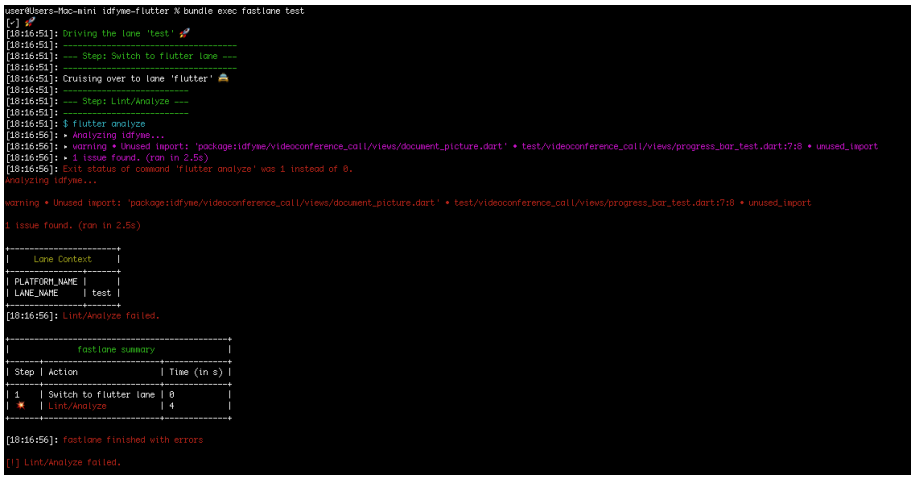


Figura 6.4: Consola a exibir a execução do Fastlane, incluindo erros e avisos encontrados no código

6.4.3 Testes Automáticos

Além da limpeza de código, o `fastfile` também facilitou a execução de testes fazendo-os de uma forma automática. Isto não só economizou tempo, eliminando a necessidade de

[2] https://developer.apple.com/testflight/

executar manualmente cada teste, como também melhorou a confiabilidade da solução ao garantir que todas as partes do código fossem testadas de maneira sistemática e repetitiva.

O `fastfile` também proporcionou a vantagem adicional de gerar relatórios abrangentes de cobertura dos testes realizados, já mostrados na secção anterior e ilustrados nas figuras 6.2 e 6.3. Estes relatórios podiam ser filtrados e convertidos em formato HTML para facilitar a sua visualização, proporcionando uma compreensão intuitiva do estado e da qualidade dos testes implementados. Isto foi crucial para identificar áreas do código que ainda não tivessem sido cobertas por testes, além de ter sido útil para demonstrar a qualidade da solução aos clientes e gerentes do projeto.

As vantagens do Fastlane foram então evidentes na sua capacidade de automatizar tarefas repetitivas e propensas a erros, proporcionando eficiência, consistência e clareza no processo de desenvolvimento. Ao integrar o Fastlane na solução foi possível garantir uma entrega contínua e confiável da mesma, aumentando a produtividade e mantendo a qualidade do código ao longo do tempo de desenvolvimento.

7

Conclusão

A presente tese centrou-se no desenvolvimento de uma solução para o problema identificado no produto **idfyme** da Opensoft. Este produto tem como objetivo transformar a prestação de serviços públicos legais e burocráticos, anteriormente dependentes de uma autenticação presencial obrigatória, num formato digital remoto mais prático, eficiente e conveniente para os cidadãos. No entanto, o produto apresentava uma reduzida inclusão digital, contando apenas com uma versão *web* limitada e uma aplicação Android, o que excluía uma parte significativa do mercado, nomeadamente os utilizadores de dispositivos iOS. Deste modo, a solução proposta visou o desenvolvimento de uma aplicação móvel para iOS, integrada com todas as funcionalidades já presentes na versão Android, proporcionando uma experiência de utilizador mais inclusiva, consistente e eficiente. Esta solução permitiu expandir o alcance do produto, possibilitando que um número maior de utilizadores, independentemente do sistema operativo, pudesse aceder aos seus serviços.

Após uma análise crítica, optou-se por uma abordagem multiplataforma, unificando o desenvolvimento das versões Android e iOS num único projeto com código partilhado. Embora já existisse uma versão Android no produto, decidiu-se criar uma nova versão desenvolvida de forma multiplataforma, devido às suas inúmeras vantagens. Esta abordagem permitiu uniformizar o desenvolvimento e a experiência do utilizador, garantindo consistência entre ambos os sistemas operativos, além de reduzir significativamente o tempo e os custos de desenvolvimento. Adicionalmente, simplificou a manutenção, aumentando a flexibilidade e escalabilidade futura da aplicação.

Optou-se também pela utilização da *framework* Flutter para o desenvolvimento da aplicação, o que permitiu criar uma solução móvel multiplataforma mais eficiente e responsiva. O Flutter destaca-se pela sua capacidade de gerar e personalizar interfaces de utilizador de alta performance para ambos o sistemas operativos Android e iOS, utilizando um único código base. Esta abordagem facilitou a criação de uma interface intuitiva, moderna e consistente, proporcionando uma experiência de utilizador uniforme e de elevada qualidade tanto em Android como em iOS. Além disso, a rapidez na compilação e a vasta biblioteca de *widgets* personalizáveis do Flutter contribuíram para acelerar o desenvolvimento e adaptar a aplicação às necessidades específicas do projeto.

Deste modo, adicionou-se o projeto *idfyme-flutter* ao produto, que constituiu a solução desenvolvida nesta tese: a aplicação móvel multiplataforma.

O projeto *idfyme-flutter* implementou diversas estratégias gerais para melhorar a aplicação, incluindo a arquitetura MVVM com `Services` dedicados, bem como um *design* responsivo e a gestão de temas e internacionalização, estados da aplicação e permissões necessárias ao dispositivo móvel. Adicionalmente, foram implementados mecanismos para que a UI se adaptasse e se reestruturasse de acordo com os diferentes tamanhos e orientações dos ecrãs dos dispositivos móveis.

Destacaram-se também a implementação de uma gestão modular de classes, a navegação dinâmica entre páginas, a construção reutilizável dos pedidos HTTP para comunicação com o servidor, a gestão das exceções do projeto e a implementação de *logs* e mensagens detalhadas de erros e de sucesso. A utilização de uma base de dados local assegurou ainda o armazenamento seguro e rápido dos dados do utilizador.

Foi também realizada uma análise preliminar do *design* da aplicação, tendo em mente a sua integração em ambos os sistemas operativos Android e iOS, garantindo que a aplicação fosse visualmente consistente e intuitiva.

Além da criação desta nova versão móvel multiplataforma, uma das melhorias mais significativas introduzidas na solução para o produto **idfyme** foi no seu projeto *idfyme-plugins*, responsável pelo *backend* do produto. Neste âmbito, foram implementadas melhorias em três *endpoints* do projeto, bem como um novo sistema de mensagens de erro em resposta aos pedidos HTTP que envolvem formulários. Esta atualização melhorou a experiência do utilizador, tornando-a muito mais personalizada, responsiva e eficiente.

O desenvolvimento da solução, ao inserir-se no produto **idfyme** da Opensoft, ofereceu os seus três processos distintos de serviços: agendamento e realização de videoconferências com e sem extração de dados, extração biométrica e assinatura manuscrita de documentos digitais em locais específicos do documento.

Assim, as funcionalidades implementadas na solução incluiram a autenticação do utilizador na aplicação, o ecrã inicial com o histórico dos processos, o agendamento e realização de videoconferências, a extração de dados biométricos e a assinatura manuscrita de documentos digitais. A autenticação na aplicação (*login*), essencial para a segurança da plataforma, foi implementada de forma robusta, integrando tecnologias adicionais como o uso de PIN e autenticação biométrica, proporcionando uma camada extra de proteção contra acessos não autorizados. Foi também desenvolvido o processo de registo do utilizador na aplicação, com especial atenção às mensagens de erro nos respetivos campos do formulário, de modo a facilitar a identificação e correção de problemas pelo utilizador.

O ecrã inicial foi projetado de forma simples e intuitiva, permitindo aos utilizadores aceder facilmente aos processos em curso, agendados ou concluídos, com a possibilidade de aplicar filtros para uma navegação mais eficiente.

A videoconferência, uma funcionalidade central da solução, permitiu uma comunicação remota eficiente, através de vídeo e áudio, usando uma conexão P2P com base na tecnologia WebRTC. Para o estabelecimento da conexão, foi utilizado o servidor de sinalização do produto, com *sockets* para comunicar com ele de forma bidirecional e assíncrona. Adicionalmente foram ainda usados os servidores STUN e TURN de forma a garantir a conectividade da ligação em redes mais complexas.

Foram então implementadas videoconferências normais, para o esclarecimento de dúvidas entre clientes e funcionários, e videoconferências de autenticação, que envolvem a extração de diversos dados, como fotografias de rosto e documentos de identificação e assinaturas manuscritas digitais, diretamente a partir do dispositivo móvel. Estes dados eram solicitados pelo funcionário através de um *script* que ele seguia na sua aplicação *web*, sendo enviados de forma imediata e validados pelo funcionário em tempo real durante a chamada. A solicitação e o envio dos dados na solução foram facilitados pelo uso dos *sockets*, que, ao estarem conectados ao servidor de sinalização e funcionarem de forma assíncrona, conseguiam responder aos comandos do funcionário sem degradar a experiência do utilizador na aplicação. Uma dificuldade significativa enfrentada foi a transição entre a interface da videoconferência e as páginas de extração destes dados, que foi resolvida com a utilização do `ChangeNotifier` do Flutter. Este permitiu alterar o estado do objeto que continha o estado da página a ser mostrada ao utilizador e notificar as restantes páginas subscritas a ele, desencadeando assim a sua renderização e a transição para a nova página necessária.

A segurança destes dados trocados foi ainda garantida por uma prévia autenticação do utilizador na videoconferência, na qual o funcionário valida o ambiente do cidadão e utiliza um código de verificação (*token*) e perguntas pessoais para garantir a sua identidade. Esta segurança foi também reforçada com a codificação dos dados em `base64` antes de serem enviados ao funcionário, garantindo a sua integridade e eficiência e facilitando o seu processo de transmissão na rede.

A funcionalidade de extração biométrica permitiu a recolha de dados sensíveis, como fotografias de rosto e assinaturas manuais digitais, diretamente pelo dispositivo móvel. Aqui implementou-se uma validação adicional da fotografia de rosto, com recurso ao *plugin* `google-mlkit-face-detection`[1], que se baseia no ML Kit[2] da Google. Este *plugin* usa uma API de inteligência artificial de forma a garantir que a imagem cumpra determinados critérios como a posição correta da face do utilizador, a ausência de sorriso ou olhos fechados, e a orientação frontal da cabeça. Adicionalmente, foi verificada a correta posição do dispositivo móvel utilizando o acelerómetro integrado dos dispositivos, garantindo que o telemóvel estava adequadamente posicionado na vertical ou horizontal, evitando que a imagem ficasse inclinada. Esta validação preliminar teve como objetivo reduzir o processamento no servidor, bem como simplificar o processo de extração da fotografia do utilizador, tornando-o mais eficiente e direto.

[1] https://pub.dev/packages/google_mlkit_face_detection
[2] https://developers.google.com/ml-kit?hl=pt-br

A funcionalidade de assinatura de documentos digitais foi implementada para permitir aos utilizadores visualizar e assinar documentos oficiais de forma simples, diretamente no ecrã do dispositivo móvel. Aqui o utilizador pode visualizar documentos por assinar e já assinados, e escolher o local no documento onde deseja aplicar a sua assinatura. A assinatura é então desenhada diretamente no ecrã do dispositivo móvel, e a partir do uso do *plugin* `signature`[3] a sua imagem é capturada, e depois editada no local escolhido no documento.

A implementação desta funcionalidade enfrentou dificuldades devido à necessidade de usar diferentes bibliotecas para renderizar e editar os documentos PDF, pois o Flutter não possuía uma única biblioteca ou *plugin* que cobrisse ambos os processos. Assim foi usado o *plugin* `alh-pdf-view`[4] para renderizar o documento e a biblioteca `syncfusion-flutter-pdf`[5] para o editar com a assinatura.

Outra dificuldade encontrada foi o cálculo das coordenadas corretas para posicionar a assinatura no documento, uma vez que as coordenadas extraídas pelo retângulo posicionado pelo utilizador na escolha do seu local eram relativas ao ecrã e não ao tamanho do documento PDF a ser editado. Isto exigiu o cancelamento do *zoom* do documento e a correção da orientação horizontal do dispositivo, assim como o desenvolvimento de funções específicas para ajustar corretamente a escala e calcular as coordenadas adequadas.

Estas últimas duas funcionalidades da solução, extração biométrica e assinatura de documentos, foram ainda implementadas para passarem por uma camada de segurança extra antes de os dados serem submetidos, solicitando que o utilizador introduza o seu PIN ou utilize a autenticação biométrica, garantindo assim que apenas o utilizador autorizado os possa submeter.

Para assegurar a maior qualidade e segurança da solução desenvolvida, foram também implementadas diversas técnicas e práticas de desenvolvimento. Isto incluiu a elaboração preliminar de User Stories, o uso de vários ambientes de desenvolvimento para testes e produção, a execução de 268 testes unitários, de integração, funcionais e de *widget* na aplicação, e o uso do Fastlane para a automatização de tarefas, como a execução dos testes e a geração dos respetivos relatórios, bem como para a gestão eficiente das *builds* da aplicação para a sua respetiva versão Android e iOS.

7.1 Contribuições

A solução desenvolvida gerou várias contribuições importantes. Primeiramente, o desenvolvimento de uma plataforma multiplataforma eficiente, utilizando Flutter, reduziu significativamente a complexidade do desenvolvimento da aplicação, eliminando a necessidade de manter dois projetos separados para Android e iOS no produto **idfyme**. Isto

[3] https://pub.dev/packages/signature
[4] https://pub.dev/packages/alh_pdf_view
[5] https://pub.dev/packages/syncfusion_flutter_pdf

tornou a solução, e consequentemente o produto, mais consistente, escalável e fácil de manter, além de reduzir os custos e esforços para futuras evoluções da aplicação.

Outro aspeto relevante desta tese foi o impacto positivo na sua inclusão digital. Ao desenvolver uma solução acessível tanto para Android como para iOS, foi possível abranger uma vasta gama de utilizadores, promovendo uma maior democratização no acesso aos serviços oferecidos pelo produto **idfyme**. Esta abordagem permitiu que um número mais alargado de utilizadores, independentemente do sistema operativo dos seus dispositivos móveis, pudesse utilizar a aplicação, tornando-a mais inclusiva, acessível e alinhada com as necessidades de um público diversificado.

Além disso, a solução introduziu um sistema de autenticação e segurança robusto, utilizando tecnologias biométricas, incluindo verificação de fotografias e assinatura manuscrita digital. Este sistema foi complementado por uma autenticação reforçada nas videoconferências de autenticação, com verificações de *tokens*, validação do ambiente do utilizador e perguntas pessoais. Foi também implementada a introdução obrigatória do PIN ou autenticação biométrica na submissão de todos os dados da aplicação. Assim, estes mecanismos garantiram que a identidade dos utilizadores fosse sempre validada de forma segura e confiável, o que protegeu a integridade dos dados e o acesso não autorizado do produto.

Além disso, a solução representa um avanço técnico ao integrar, de forma eficiente, funcionalidades de videoconferência e autenticação e extração biométrica, proporcionando uma alternativa segura, rápida e prática para serviços que anteriormente exigiam presença física obrigatória. Esta inovação demonstra que é possível realizar, com segurança e eficiência, serviços de alta confiança em ambientes digitais móveis, eliminando a necessidade de deslocações e tornando estes procedimentos significativamente mais ágeis e eficazes.

7.2 Trabalho Futuro

Embora a solução tenha atingido os seus objetivos principais ao implementar todas as funcionalidades previamente existentes no produto **idfyme**, existem diversas áreas com potencial para melhorias e expansões futuras. Uma possível evolução seria a implementação de um sistema de *chat* associado aos diferentes processos, permitindo uma comunicação direta entre utilizadores e funcionários, facilitando o esclarecimento de dúvidas e tornando os processos mais interativos e eficientes. Isto poderia facilitar a resolução de problemas e dúvidas, sem a necessidade de interrupções ou adiamentos das atividades. O mesmo *chat* poderia ser ainda integrado dentro das videoconferências, possibilitando comunicação direta e em tempo real em situações em que a rede não permita comunicação por voz ou vídeo. Isto garantiria que a comunicação entre as partes nunca tivesse de ser interrompida, além de servir como auxílio para o envio de documentos ou informações adicionais durante as videoconferências.

Outra área de expansão seria a introdução de notificações em tempo real na aplicação móvel para alertar os utilizadores sobre prazos, alterações nos processos ou lembretes de

videoconferências agendadas. Estas notificações melhorariam a experiência do utilizador, mantendo-o informado sobre o progresso e o estado dos seus processos, e ajudariam a evitar atrasos ou esquecimentos na realização dos atendimentos por videoconferência.

Uma última funcionalidade futura que também agregaria valor seria a integração da aplicação com calendários pessoais, como o Google Calendar[6]. Esta integração permitiria sincronizar automaticamente os compromissos pessoais dos utilizadores com os processos da aplicação, como prazos para extração biométrica e assinatura de documentos, bem como os atendimentos de videoconferências marcados, facilitando assim a sua organização e evitando sobreposições ou esquecimentos.

[6] https://developers.google.com/calendar?hl=pt-br

Bibliografia

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. i).

[2] M. L. Suzie Forell e E. Digiusto. «Legal assistance by video conferencing: What is known?» Em: *Justice Issues* 15 (2011), pp. 01–23. ISSN: 1327-3248. URL: https://www.researchgate.net/publication/277558506_Legal_assistance_by_video_conferencing_what_is_known (ver p. 1).

[3] J. Wajcman et al. *The Impact of the Mobile Phone on Work/Life Balance*. Rel. téc. Australian Research Council, 2007 (ver p. 2).

[4] statcounter GlobalStats. *Mobile Operating System Market Share Worldwide - August 2024*. 2024. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (ver pp. 2–4, 34, 35).

[5] C. L. Caldiroli et al. «Comparing online cognitive load on mobile versus PC-based devices». Em: *Personal and Ubiquitous Computing* 27 (2023), pp. 495–505. ISSN: 1617-4917. DOI: 10.1007/s00779-022-01707-8 (ver p. 2).

[6] N. Clarke e S. Furnell. «Advanced user authentication for mobile devices». Em: *Computers and Security* 26 (2007), pp. 109–119. ISSN: 0167-4048. DOI: 10.1016/j.cose.2006.08.008. (ver p. 2).

[7] E. Belianinik. *Cross-platform vs. native development at a glance*. 2024. URL: https://www.instinctools.com/blog/native-vs-cross-platform-mobile-development/ (ver pp. 8–10).

[8] P. Nawrocki et al. «A Comparison of Native and Cross-Platform Frameworks for Mobile Applications». Em: *Computer* 54.3 (2021), pp. 18–27. ISSN: 1558-0814. DOI: 10.1109/MC.2020.2983893 (ver p. 10).

[9] A. Tashildar et al. «Application Development Using Flutter». Em: *International Research Journal of Modernization in Engineering Technology and Science* 02.08 (2020), pp. 1263–1266. ISSN: 2582-5208 (ver p. 10).

[10] J. Faizullaev. *Native-like Cross-Platform Mobile Development : Multi-OS Engine and Kotlin Native vs Flutter, Master Thesis, Southeast Finland University of Applied Sciences*. 2018 (ver p. 11).

[11] L. Dagne. *Flutter for cross-platform App and SDK development, Master Thesis, Metropolia University of Applied Sciences*. 2019 (ver p. 11).

[12] D. Flanagan. *JavaScript: o guia definitivo*. Bookman Editora, 2012. ISBN: 978-0-596-80552-4 (ver p. 12).

[13] C. Gackenheimer. *Introduction to React*. Apress, 2015. ISBN: 978-1-4842-1246-2 (ver p. 12).

[14] N. Hansson e T. Vidhall. *Effects on performance and usability for cross-platform application development using React Native, Master Thesis, Institutionen f¨or datavetenskap Department of Computer and Information Science*. 2016 (ver p. 12).

[15] S. Dekkati, K. Lal e H. Desamsetti. «React Native for Android: Cross-Platform Mobile Application Development». Em: *Global Disclosure of Economics and Business* 08.02 (2019), pp. 153–164. ISSN: 2307-9592. DOI: https://doi.org/10.18034/gdeb.v8i2.696 (ver pp. 12, 13).

[16] A. Skantz. *Performance Evaluation of Kotlin Multiplatform Mobile and Native iOS Development in Swift, Master Thesis, Kth Royal Institute of Technology*. 2023 (ver pp. 14, 15).

[17] A.-K. Evert. *Cross-Platform Smartphone Application Development with Kotlin Multiplatform, Master Thesis, Kth Royal Institute of Technology*. 2019 (ver p. 14).

[18] *The Six Most Popular Cross-Platform App Development Frameworks*. URL: https://kotlinlang.org/docs/cross-platform-frameworks.html#kotlin-multiplatform-mobile (acedido em 2023-11-24) (ver p. 15).

[19] M. Isitan e M. Koklu. «Comparison and Evaluation of Cross Platform Mobile Application Development Tools». Em: *International Journal of Applied Mathematics, Electronics and Computers* 08.04 (2020), pp. 273–281. ISSN: 2147-8228. DOI: https://doi.org/10.18100/ijamec.832673 (ver p. 17).

[20] G. Suciu et al. «WebRTC role in real-time communication and video conferencing». Em: *2020 Global Internet of Things Summit (GIoTS)*. Dublin, Ireland, 2020, pp. 1–6. ISBN: 978-1-7281-6728-2. DOI: 10.1109/GIOTS49054.2020.9119656 (ver pp. 17, 18).

[21] A. L. Alexander, A. L. Wijesinha e R. Karne. «An Evaluation of Secure Real-Time Transport Protocol (SRTP) Performance for VoIP». Em: *2009 Third International Conference on Network and System Security*. Gold Coast, QLD, Australia, 2009, pp. 95–101. DOI: 10.1109/NSS.2009.90 (ver p. 17).

[22] A. Shiranzaei e R. Z. Khan. «Internet protocol versions — A review». Em: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India, 2015, pp. 397–401. ISBN: 978-9-3805-4416-8 (ver p. 18).

[23] D. Wing. «Network Address Translation: Extending the Internet Address Space». Em: *IEEE Internet Computing* 14.4 (2010), pp. 66–70. ISSN: 1941-0131. DOI: 10.1109/MIC.2010.96 (ver p. 18).

[24] S. Dutton. *WebRTC in the real world: STUN, TURN and signaling*. 2013-11. URL: https://web.dev/articles/webrtc-infrastructure?hl=pt-br (ver p. 18).

[25] V. Pimentel e B. G. Nickerson. «Communicating and Displaying Real-Time Data with WebSocket». Em: *IEEE Internet Computing* 16.4 (2012), pp. 45–53. ISSN: 1941-0131. DOI: 10.1109/MIC.2012.64 (ver p. 18).

[26] A. Mardan. «Socket.IO and Express.js». Em: *Pro Express.js*. Berkeley, CA: Apress, 2014, pp. 193–198. ISBN: 978-1-4842-0037-7. DOI: 10.1007/978-1-4842-0037-7_16. URL: https://doi.org/10.1007/978-1-4842-0037-7_16 (ver pp. 19, 64).

[27] Google. *ML Kit*. URL: https://developers.google.com/ml-kit?hl=pt-br (ver p. 20).

[28] J.-L. G. Tomas Brezmes e J. Cotrina. «Activity recognition from accelerometer data on a mobile phone». Em: *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living: 10th International Work-Conference on Artificial Neural Networks*. Vol. 5518. Salamanca, Spain: Springer, Berlin, Heidelberg, 2009, pp. 796–799. ISBN: 978-3-642-02481-8. DOI: https://doi.org/10.1007/978-3-642-02481-8_120 (ver pp. 21, 83).

[29] F. Gems. *Top Flutter PDF packages*. 2024. URL: https://fluttergems.dev/pdf/ (ver pp. 22, 23).

[30] D. Zelenchuk. «AndroidX Test Library». Em: *Android Espresso Revealed: Writing Automated UI Tests*. Berkeley, CA: Apress, 2019, pp. 271–280. ISBN: 978-1-4842-4315-2. DOI: 10.1007/978-1-4842-4315-2_14. URL: https://doi.org/10.1007/978-1-4842-4315-2_14 (ver p. 23).

[31] X. Wang et al. *Fingerprint-jacking: Practical fingerprint authorization hijacking in Android apps*. Rel. téc. The Chinese University of Hong Kong e Sangfor Technologies Inc., 2020 (ver p. 24).

[32] M. Owens e G. Allen. *SQLite*. Apress LP New York, 2010. ISBN: 978-1-4302-3226-1 (ver p. 24).

[33] P. Sarang. *Practical liferay: Java-based portal applications development*. Apress, 2009. ISBN: 978-1-4302-1848-7 (ver p. 32).

[34] O. Arponen. *Software architectural patterns and principles in Android development, Master Thesis, Metropolia University of Applied Sciences*. 2023 (ver p. 33).

[35] S. Thakur. *How to Choose the Minimum iOS Support Version for Your App*. https://medium.com/mobile-app-development-publication/how-to-choose-the-minimum-ios-support-version-for-your-app-4ff0494a5ee7. 2023 (ver p. 35).

[36] E. S. Jenny Ruiz e M. Snoeck. «Unifying Functional User Interface Design Principles». Em: *International Journal of Human–Computer Interaction* 37.1 (2021), pp. 47–67. DOI: 10.1080/10447318.2020.1805876. eprint: https://doi.org/10.1080/10447318.2020.1805876. URL: https://doi.org/10.1080/10447318.2020.1805876 (ver pp. 37, 38, 42–45, 48).

[37] T. Wetchakorn e N. Prompoon. «Method for mobile user interface design patterns creation for iOS platform». Em: *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 2015, pp. 150–155. DOI: 10.1109/JCSSE.2015.7219787 (ver p. 38).

[38] W. H. Robbins. *Design Practices In Mobile User Interface Design*. Cal Poly, 2014. URL: https://digitalcommons.calpoly.edu/grcsp/130/ (ver p. 38).

[39] C. Anderson. «The Model-View-ViewModel (MVVM) Design Pattern». Em: *Pro Business Applications with Silverlight 5*. Berkeley, CA: Apress, 2012, pp. 461–499. ISBN: 978-1-4302-3501-9. DOI: 10.1007/978-1-4302-3501-9_13. URL: https://doi.org/10.1007/978-1-4302-3501-9_13 (ver pp. 40, 42).

[40] B. Esselink. *A practical guide to localization*. Vol. 4. John Benjamins Publishing, 2000 (ver p. 42).

[41] R. E. Johnson e B. Foote. «Designing reusable classes». Em: *Journal of object-oriented programming* 1.2 (1988), pp. 22–35 (ver pp. 42, 49).

[42] Flutter. *Simple app state management*. https://docs.flutter.dev/data-and-backend/state-mgmt/simple (ver pp. 44, 45).

[43] GeeksforGeeks. *Service Locator Pattern*. https://www.geeksforgeeks.org/service-locator-pattern/. 2023 (ver p. 46).

[44] Justinmind. *Mobile navigation: patterns and examples*. https://www.justinmind.com/blog/mobile-navigation/. 2024 (ver p. 47).

[45] R. Stevens et al. «Asking for (and about) permissions used by Android apps». Em: *2013 10th Working Conference on Mining Software Repositories (MSR)*. 2013, pp. 31–40. DOI: 10.1109/MSR.2013.6624000 (ver p. 49).

[46] A. F. Garcia et al. «A comparative study of exception handling mechanisms for building dependable object-oriented software». Em: *Journal of Systems and Software* 59.2 (2001), pp. 197–222. ISSN: 0164-1212. DOI: https://doi.org/10.1016/S0164-1212(01)00062-0. URL: https://www.sciencedirect.com/science/article/pii/S0164121201000620 (ver p. 50).

[47] S. Gupta. *Application Logging and its importance*. https://medium.com/ula-engineering/application-logging-and-its-importance-c9e788f898c0. 2020 (ver p. 50).

[48] L. .-. E. at Work. *Cloud vs Local Storage: Which is more secure?* https://www.linkedin.com/pulse/cloud-vs-local-storage-which-more-secure-lunik-explorers/. 2023 (ver pp. 50, 51).

[49] D. Gourley e B. Totty. *HTTP: the definitive guide*. O'Reilly Media, Inc., 2002 (ver p. 51).

[50] G. Barbaglia, S. Murzilli e S. Cudini. «Definition of REST web services with JSON schema». Em: *Software: Practice and Experience* 47.6 (2017), pp. 907–920. DOI: 10.1002/spe.2466 (ver p. 52).

[51] SurveyJS. *Client-Side vs Server-Side Form Input Validation*. https://surveyjs.io/stay-updated/blog/client-server-data-validation (ver p. 53).

[52] J. Albano. *Best Practices for REST API Error Handling*. https://www.baeldung.com/rest-api-error-handling-best-practices. 2024 (ver p. 55).

[53] D. Gupta. *The Developer's Guide to Mobile Authentication*. https://readwrite.com/the-developers-guide-to-mobile-authentication/. 2022 (ver pp. 56, 58).

[54] G. Roelofs e R. Koman. *PNG: The Definitive Guide*. USA: O'Reilly & Associates, Inc., 1999. ISBN: 1565925424 (ver p. 77).

[55] S. Moyal. *A Dive into Base64 and Its Significance in Web Development*. https://medium.com/@shimonmoy dive-into-base64-and-its-significance-in-web-development-b6bd4427f61d. 2023 (ver p. 77).

[56] G. Lucassen et al. «The Use and Effectiveness of User Stories in Practice». Em: *Requirements Engineering: Foundation for Software Quality*. Ed. por M. Daneva e O. Pastor. Springer International Publishing, 2016, pp. 205–222. ISBN: 978-3-319-30282-9 (ver pp. 102, 103).

[57] G. Duta. *Best Practices for Dev, QA, and Production Environments*. https://www.bunnyshell.com/blog/be practices-for-dev-qa-and-production-environments/. 2023 (ver p. 103).

[58] S. K. Singh e A. Singh. *Software testing*. Vandana Publications, 2012 (ver pp. 104–106).

[59] Jshosseini. *Exploring the Depths of Flutter Widget Testing: A Step-by-Step Approach*. https://medium.com/@jshosseini643/exploring-the-depths-of-flutter-widget-testing-a-step-by-step-approach-64e5f50450d4. 2023 (ver p. 104).