Cliente Mobile e Desktop de Comunicações Unificadas para IPBrick OS

Diogo Paiva[1], Miguel Ramalhão[3], Carlos José Campos[1,4], Veríssimo Lima[1], Pedro Reis[1], Vítor Cardoso[1], António Sousa[1,2], Sandra Aires[1], Carla Pinto[1] e Fernando Carvalho[1]

[1] Instituto Superior de Engenharia do Porto (ISEP) – P.Porto, Porto, Portugal
[2] LEMA, ISEP-IPP, Porto, Portugal
[3] IPBRICK - Expandindustria - Estudos, Projectos e Gestão de Empresas, S.A
[4] Instituto de Engenharia de Sistemas e Computadores em Tecnologia e Ciência (INESC TEC), Porto, Portugal
{1221340, crc, vms, mps, vcc, ats, sma, cap, fjc}@isep.ipp.pt

Resumo. O IPBRICK OS é um sistema operativo baseado em Linux Debian, concebido para suportar as soluções empresariais da IPBRICK no domínio das Comunicações Unificadas sobre IP (UCoIP) e Ferramentas Colaborativas para Gestão de Documentos e Processos. Entre os serviços UCoIP integrados pelo módulo IPBRICK.PBX (baseado em Asterisk), destaca-se o *Voice Over Internet Protocol* (VoIP), por norma, suportado por um conjunto distribuído de servidores que executam, entre outros, os protocolos *Session Initiation Protocol* (SIP), responsável pela sinalização e gestão de chamadas, e o *Real-Time Transport Protocol* (RTP), que assegura a transmissão de áudio em tempo real.
A versão 8.0 exigiu a implementação e integração de uma aplicação cliente Mobile e Desktop de Comunicações Unificadas (*softphone* VoIP) que privilegia a simplicidade de configuração, a eficiência de utilização e a estética do IPBRICK OS. Nesse sentido, este trabalho apresenta uma aplicação *softphone* VoIP responsiva, implementada em Dart através da framework Flutter. É baseada numa arquitetura cliente-servidor, em que a comunicação é realizada via HTTP, recorrendo a APIs PHP que estabelecem a interface com o servidor SIP. Deste modo, descrevem-se os principais aspetos técnicos e arquiteturais da solução proposta, incluindo a modelação dos fluxos de comunicação, a estrutura das APIs e os layouts funcionais da interface gráfica. Apresentam-se ainda diagramas de fluxo e os protótipos de ecrã das principais componentes da aplicação – autenticação *Lightweight Directory Access Protocol* (LDAP), interface principal, teclado de marcação, lista de contactos, registo de histórico e de chamadas.

Palavras-chave: Flutter, IPBRICK OS, PHP, SIP, *Softphone*, RTP, VoIP.

1 Introdução

A evolução tecnológica tem promovido o uso intensivo de telemóveis, estimando-se que, em abril de 2025, cerca de 5.81 mil milhões de pessoas, correspondendo a 70.7%

da população mundial, utilizem telemóveis [1]. No centro desta tendência encontram-se as aplicações móveis, que simplificam tarefas do quotidiano [2]. As aplicações são geralmente classificadas como móveis, desktop e Web [3]. No domínio das aplicações móveis, a App Store disponibiliza 1.8 milhões de aplicações [4], enquanto a Google Play Store oferece cerca de 1.55 milhões [5]. Neste vasto universo de aplicações, destacam-se pela sua popularidade, as dedicadas à comunicação baseada em texto, áudio e vídeo – facto que motivou o desenvolvimento de uma aplicação multiplataforma de comunicações com capacidade de realizar chamadas de voz através da Internet – um dos serviços de *softphone VoIP* (doravante designado por *softphone*). Tratando-se de uma aplicação multiplataforma, tem a vantagem de funcionar em vários dispositivos – *write once, run everywhere* – permite diminuir custos e tempo de desenvolvimento. No entanto, as aplicações multiplataforma comparadas com as aplicações nativas, apresentam desempenho inferior e limitações impostas por atualizações realizadas nos sistemas operativos que podem comprometer total ou parcialmente as suas funcionalidades [6-8].

Tecnicamente, os *softphones* são aplicações que recorrem a tecnologias de Voz sobre IP (VoIP), oferecendo protocolos como o *Session Initiation Protocol* (SIP), o *Real-time Transport Protocol* (RTP) e o *Extensible Messaging and Presence Protocol* (XMPP) para mensagens instantâneas. Os mais modernos já integram *Web Real-Time Communication* (WebRTC), uma tecnologia que combina múltiplos protocolos, entre outros, o RTP, o *Secure* RTP (SRTP) e o *Datagram Transport Layer Security* (DTLS); permitindo transmitir áudio, vídeo e dados entre navegadores e aplicações em tempo real [9]. Apesar da sua flexibilidade e da compatibilidade com diferentes sistemas operativos, a maioria dos *softphones* exige configurações manuais complexas, tais como: a introdução de endereços de servidores SIP, credenciais do utilizador e definições específicas dos serviços. Muitas destas configurações envolvem conhecimento que os utilizadores nem sempre possuem, o que dificulta a sua utilização, especialmente em ambientes empresariais. Por esse facto, a solução proposta foi desenvolvida com o objetivo de simplificar a experiência de utilização dos clientes dos serviços IPBRICK, através de uma aplicação funcional e prática, que requer o mínimo de configuração possível. Para o efeito, uma análise ao estado da arte relativamente à evolução das comunicações VoIP, mostra que esta tecnologia tem impulsionado o desenvolvimento de diversos *softphones* que oferecem funcionalidades avançadas, compatibilidade multiplataforma e suporte a codecs de alta qualidade. Entre os mais utilizados, destacam-se: o Linphone, o MicroSIP, o Zoiper, o Jitsi, o Bria e o PhonerLite, com características específicas à medida da experiência do utilizador [10]. O Linphone é open-source e permite elevada flexibilidade e integração com serviços VoIP, mas a sua interface e a configuração inicial são complexas para utilizadores inexperientes. O MicroSIP é muito eficiente para o Windows, mas apresenta limitações de compatibilidade noutros ambientes. O Zoiper permite configurações automáticas nas versões empresariais, mas já na sua versão gratuita exige conhecimentos de configurações SIP. O Jitsi disponibiliza suporte nativo a videoconferência e é muito usado em ambientes colaborativos. O Bria é uma solução comercial para integração

empresarial e oferece suporte técnico dedicado [11]. Finalmente, o PhonerLite é uma aplicação bastante simples para uso pessoal em Windows, mas a interface que disponibiliza é considerada pouco apelativa. [10]

Em [12] e [13], comparam arquiteturas baseadas em WebRTC e SIP, concluindo que o WebRTC apresenta melhor desempenho em métricas como latência, PSNR e estabilidade de rede.

2 Tecnologias e Protocolos

A framework Flutter utiliza linguagem Dart, semelhante a Java e JavaScript, permitindo o desenvolvimento de interfaces nativas multiplataforma baseadas em *widgets*. A integração da biblioteca *dart_sip_ua* na framework possibilita a comunicação via protocolo SIP, utilizado em sistemas VoIP para o estabelecimento, controlo e terminação de sessões de comunicação sobre WebSocket. Embora a sinalização da chamada seja gerida pelo *dart_sip_ua*, a transmissão de áudio e vídeo é realizada através do protocolo RTP que, integrado no protocolo WebRTC, garante que a mesma transmissão se realize em tempo real [9].

O processo de autenticação de utilizadores recorre a *Lightweight Directory Access Protocol* (LDAP), tecnologia que fornece acesso a informações de diretório, ou seja, armazena dados estruturados, entre outros, sobre utilizadores e dispositivos [14].

3 Implementação

De acordo com a arquitetura geral do projeto apresentada na Fig. 1, a solução desenvolvida considera três entidades distintas: a aplicação, os serviços de comunicação VoIP da IPBRICK e as APIs que permitem estabelecer a comunicação entre ambas. A comunicação é realizada através de pedidos HTTP POST, com o objetivo de definir os diferentes campos de configuração necessários para que as bibliotecas se liguem aos respetivos servidores. Quando as bibliotecas não são necessárias, as APIs estabelecem a ligação direta entre a aplicação e os servidores.

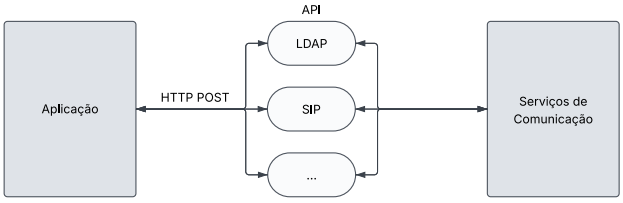

Fig. 1. Arquitetura geral do projeto.

A aplicação integra funcionalidades de autenticação e de gestão de chamadas, cujo respetivo fluxo de utilização se encontra representado na arquitetura geral da aplicação

da Fig. 2. Começa por apresentar ao utilizador um ecrã inicial, onde devem ser inseridas as credenciais de autenticação. Caso a autenticação seja válida, ocorre o registo no servidor SIP; caso contrário, o utilizador é redirecionado para o ecrã de autenticação. Após a autenticação e o respetivo registo no servidor SIP, o utilizador tem acesso ao ecrã principal, podendo navegar entre os ecrãs disponíveis e dedicados ao teclado, aos contactos e ao histórico, a partir dos quais é possível efetuar ou receber chamadas e regressar ao ecrã principal após a chamada terminar.

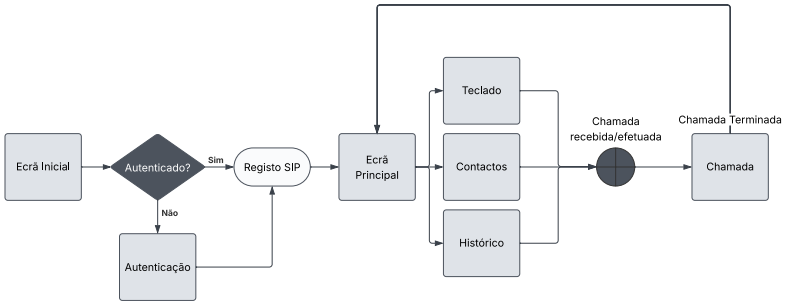

Fig. 2. Arquitetura geral da aplicação - *softphone*.

Considerando os numerosos detalhes que envolvem a lógica de programação dos vários ecrãs que constituem a arquitetura geral da aplicação, apresentam-se apenas os principais e de fluxo mais complexo, utilizando fluxogramas, assumindo-se que os restantes possam ser facilmente concebidos a partir destes exemplos.

3.1 Ecrã de Autenticação

O ecrã de autenticação assegura que só utilizadores autorizados acedem às funcionalidades da aplicação, apresentando um formulário simples, que solicita o preenchimento do email e da palavra-passe submetidos pela ação de um botão, Fig. 3.

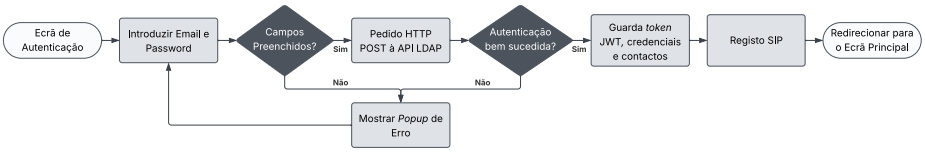

Fig. 3. Fluxograma do ecrã de autenticação.

Após a submissão do formulário, verifica-se se os campos foram preenchidos. Havendo campos vazios, é apresentado um *popup* de erro solicitando o preenchimento. Já se foram preenchidos, é efetuado um pedido HTTP POST à API de autenticação, que envia as credenciais em formato JSON, conforme é ilustrado no fluxograma da Fig.4.

A API procede à receção e extração dos dados do corpo da mensagem para autenticação no servidor LDAP. Se a autenticação não for bem-sucedida, é enviada uma mensagem de erro para a aplicação a informar o utilizador. Em caso de sucesso, é gerado um token JWT, contendo informações relevantes do utilizador e o tempo de expiração do token (atualmente definido para uma hora), que será utilizado nas iterações subsequentes com a API. De seguida, é efetuada uma consulta aos restantes utilizadores existentes no servidor LDAP, sendo estes filtrados e ordenados alfabeticamente para integração na lista de contactos da aplicação. Por fim, é enviada uma mensagem de sucesso, incluindo o token e a lista de contactos, os quais são armazenados localmente no dispositivo do utilizador através das bibliotecas *shared_preferences* e *flutter_secure_storage*, Fig. 4.

Fig. 4. Fluxograma relativo à API de autenticação dos utilizadores.

Após o processo de autenticação, a aplicação regista o utilizador no servidor SIP, redireciona-o para o ecrã principal e habilita-o à realização e receção de chamadas.

```
<? php
     header ('Content - Type : application / json');
     header ('Access - Control - Allow - Origin : * ');
     header ('Access - Control - Allow - Methods : POST');
     header ('Access - Control - Allow - Headers : Content - Type');
     $data = json_decode (file_get_contents ("php :// input"),true );
     $email = $data ['email'];
     $pass = $data ['password'];
     $displayName = $data ['displayName'];
     list ( $uid , $domain ) = explode ( '@' , $email );
     list ( $emailp , $orgp ) = explode ( '.' , $domain );
     $uri = " ";
     $webSocketUrl = " ";
     $authorizationUser = $uid;
     $password = $pass ;
     $displayName = $displayName;
     $userAgent = "CAFE Phone";
     $port = "5060";
     $register_expires = 3600;
     $sip_config = [
     " uri " => $uri ,
     " webSocketUrl " => $webSocketUrl,
     " authorizationUser " => $authorizationUser,
     " password " => $password,
     " displayName " => $displayName,
     " userAgent " => $userAgent,
     " register_expires " => $register_expires]
     echo json_encode ( $sip_config ) ;
 ? >
```

List. 1. API de configuração SIP.

Para que tal suceda, os dados de autenticação armazenados na aplicação – como o nome, o email e a palavra-passe – são enviados através de um pedido HTTP POST para uma API (List. 1), responsável por processar essa informação e gerar a configuração completa dos parâmetros necessários à ligação com o servidor SIP. Entretanto, esses parâmetros são devolvidos à aplicação em formato JSON, permitindo-lhe prosseguir com o registo (REGISTER SIP) do utilizador no servidor SIP através da biblioteca *dart_sip_ua*. Conforme o desejado, este procedimento garante características de configuração mínima e dinâmica da aplicação, uma vez que as definições específicas do servidor SIP são geridas no servidor HTTP, evitando a necessidade de modificar a aplicação sempre que ocorrem alterações ao nível do processo de comunicação e alojar de forma permanente os dados sensíveis sobre o utilizador na aplicação. Adicionalmente, no registo do servidor SIP são configurados notificadores de eventos, para o estado de registo e para o estado das chamadas, que permitem que os restantes ecrãs recebam, em tempo real, o estado da ligação e das chamadas, bem como alterações registadas na aplicação.

3.2 Ecrã Inicial (*Splash Screen*)

O ecrã inicial exibe-se momentaneamente no ecrã e atua como um *splash screen*, sendo exibido durante um curto período enquanto são verificadas as credenciais e carregados os recursos essenciais, Fig. 5.

Fig. 5. Fluxograma relativo ao ecrã inicial.

3.3 Ecrã Principal

O ecrã principal permite ao utilizador aceder às suas funcionalidades essenciais, como o teclado, contactos e histórico de chamadas, Fig. 6.

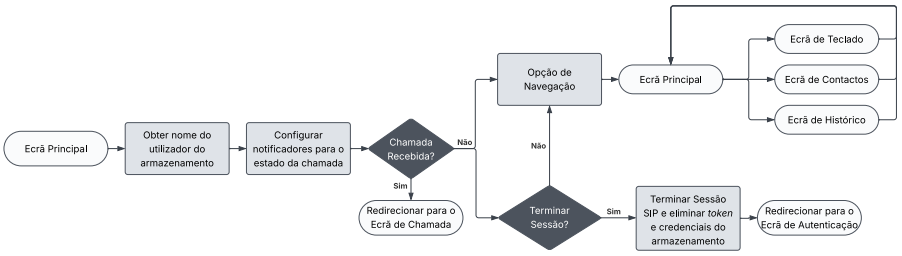

Fig. 6. Fluxograma do ecrã principal.

3.4 Ecrã do Teclado

O ecrã de teclado permite ao utilizador a introdução de números para efetuar chamadas ou pesquisar contactos, Fig. 7.

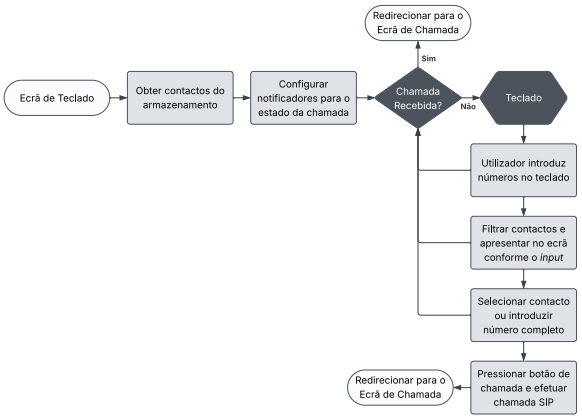

Fig. 7. Fluxograma do ecrã do teclado.

3.5 Ecrã da Chamada

O ecrã da chamada é responsável por gerir toda a experiência de comunicação em tempo real, tanto para chamadas efetuadas como recebidas. O seu funcionamento encontra-se representado no fluxograma da Fig. 8.

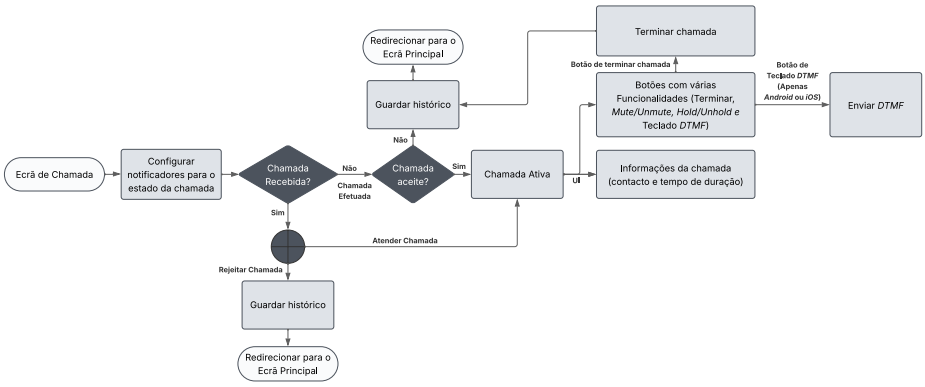

Fig. 8. Fluxograma do ecrã da chamada.

Neste ecrã, a aplicação configura os notificadores do estado de chamada, que verificam se a chamada foi recebida ou efetuada. Nesta fase, são apresentadas informações sobre a chamada, como: o nome, o email e a fotografia do contacto.

4 Resultados

A Fig. 9 apresenta os *layouts* da maioria dos ecrãs da aplicação, alguns dos quais implementados de acordo com as descrições realizadas na seção anterior.

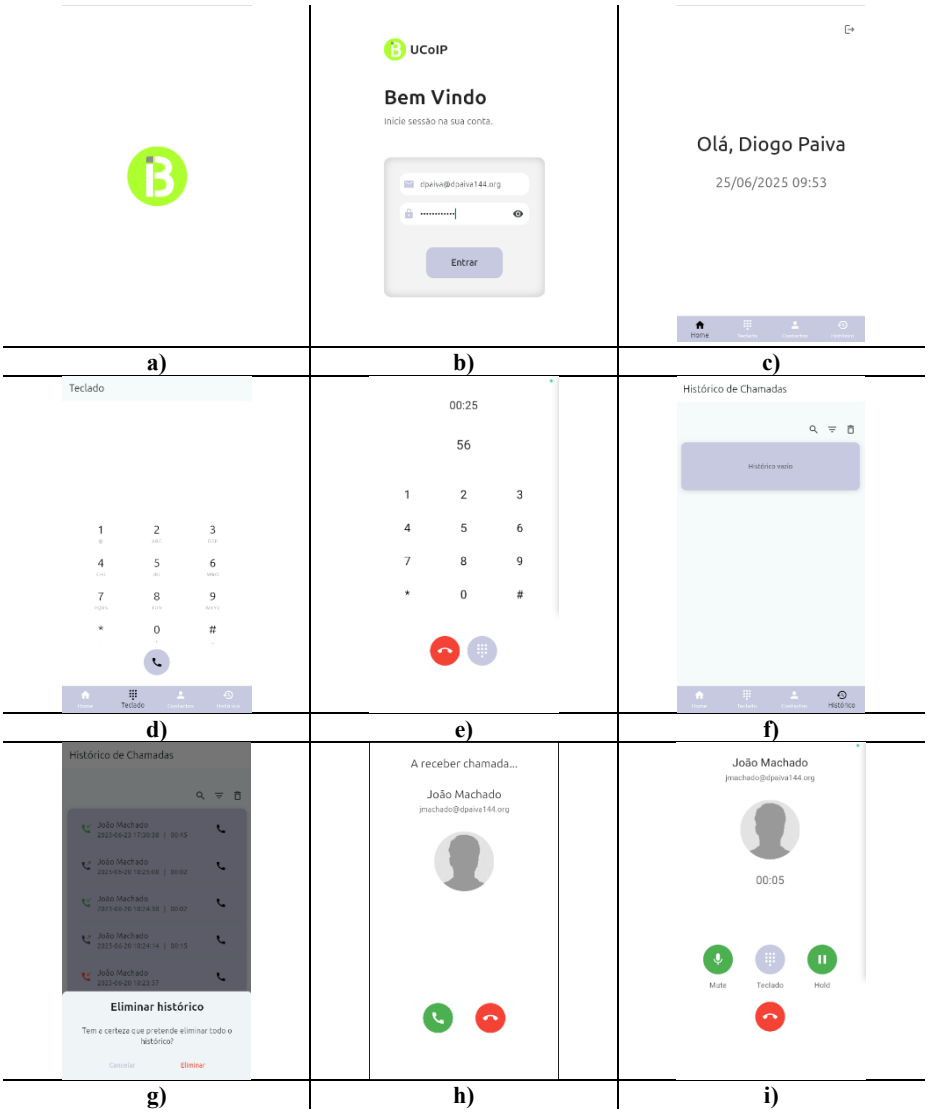

Fig. 9. Ecrãs da aplicação para equipamentos *mobile*: ecrã inicial em a), ecrã de autenticação em b), ecrã principal em c), ecrã do teclado com números inseridos em d), ecrã do teclado em marcação em e), ecrã de histórico das chamadas em f), ecrã de eliminação do histórico das chamadas em g), ecrã de chamada recebida em h) e ecrã de chamada em curso em i).

De referir que, apesar da Fig. 9 apresentar ecrãs destinados a dispositivos *mobile*, alguns deles são responsivos, ajustando a disposição dos seus elementos para ecrãs de desktop. Como exemplo, o ecrã ilustrado na Fig. 10 corresponde ao ecrã da Fig. 9 c).

Fig. 10. Ecrã principal da aplicação destinado a equipamentos desktop.

Aspetos como configuração minimalista, design, funcionalidade e robustez do *softphone* foram avaliados qualitativamente pelo responsável do projeto, colaboradores e alguns clientes do IPBRICK OS. A avaliação revelou: uma perceção positiva relativamente à simplicidade de configuração da aplicação; o design do *softphone* destaca-se pela interface intuitiva e agradável; em termos de funcionalidade, a aplicação disponibiliza recursos de utilização completos para chamadas de voz e sem opções redundantes; a robustez do sistema foi confirmada em testes práticos, mostrando estabilidade durante chamadas e fácil integração com o IPBRICK.PBX. Holisticamente, o *softphone* apresenta-se como uma solução eficiente, confiável e responde às necessidades dos utilizadores empresariais do ecossistema IPBRICK OS.

5 Conclusões e perspetivas de trabalho futuro.

Este projeto permitiu desenvolver uma aplicação multiplataforma dedicada a serviços de telefonia – um *softphone* – que comunica com os serviços de VoIP do IPBRICK OS através de APIs desenvolvidas em PHP.

Apesar dos resultados positivos alcançados, a aplicação ainda carece da implementação de funcionalidades adicionais. Assim, considera-se necessário implementar funcionalidades de *chat* e chamadas de vídeo, que poderão ser desenvolvidas recorrendo a bibliotecas do Flutter como *xmpp_stone* ou *xmpp_plugin* para mensagens e *flutter_web_rtc* ou *jitsi_meet_wrapper* para chamadas de vídeo. A interface da aplicação pode ser modificada para integrar estas funcionalidades adicionais, incluindo a criação de um ecrã de definições que permita ao utilizador personalizar a aplicação tanto a nível estético como funcional.

A implementação de notificações *push* revela-se essencial no futuro. Para dispositivos móveis, soluções como o *Firebase Cloud Messaging* (Android) e o *Apple*

Push Notification Service (iOS) permitirão alertar os utilizadores sobre chamadas pendentes, mensagens instantâneas e chamadas de vídeo. Atualmente, no caso das chamadas de voz, se um utilizador ligar para outro que não se encontre registado no servidor SIP, a chamada é direcionada para o voice-mail. Para ultrapassar esta limitação, a chamada deverá permanecer em espera no servidor enquanto este envia um pedido aos serviços de *push notifications*, que despertam a aplicação no dispositivo do destinatário, permitindo o registo no servidor SIP e a receção da chamada.

Referências

1. Kemp, S.: *Digital 2025 April Global Statshot Report.* In: DataReportal. Disponível em: https://datareportal.com/reports/digital-2025-april-global-statshot (Último acesso em 03/06/2025) (2025)
2. Shirey, R. W.: *Internet Security Glossary, Version 2.* In: RFC 4949, Aug. (2007)
3. Kissflow: *What are the Different Types of Applications?* In: Kissflow. Disponível em: https://kissflow.com/faq/what-are-the-different-types-of-applications (Último acesso em 03/06/2025) (2025)
4. Apple: *App Store.* In: Apple. Disponível em: https://www.apple.com/pt/app-store/ (Último acesso em 01/06/2025) (2025)
5. AppBrain: *Number of Android Apps on Google Play (Jun 2025).* In: AppBrain. Disponível em: https://www.appbrain.com/stats/number-of-android-apps (Último acesso em 01/06/2025) (2025)
6. El-Kassas, W. S., Abdullah, B. A., Yousef, A. H., Wahba, A. M.: *Taxonomy of Cross-Platform Mobile Applications Development Approaches.* In: Ain Shams Engineering Journal, vol. 8, no. 2, pp. 163–190 (2017)
7. Fentaw, A. E.: *Cross-Platform Mobile Application Development: A Comparison Study of React Native vs Flutter.* In: Master's Thesis, University of Jyväskylä Faculty of Information Technology, Jyväskylä (2020)
8. Farooq, S., Riaz, S., Alvi, A., Ali, A., Rehman, I.: *Cross-Platform Mobile Development Approaches and Frameworks.* In: VFAST Transactions on Software Engineering, vol. 10, pp. 79–93 (May 2022)
9. Voizcall: *WebRTC SIP Client & Softphone App: Explained.* In: Voizcall. Disponível em: https://www.voizcall.com/blog/webrtc-sip-client-softphone-app-explained (2025)
10. Cotocus: *Top 10 SIP/VoIP Clients Tools in 2025 – Features, Pros & Cons Comparison.* In: Cotocus. Disponível em: https://www.cotocus.com/blog/top-10-sip-voip-clients-tools-in-2025-features-pros-cons-comparison/ (2025)
11. CounterPath: *Bria Softphone Product Overview.* In: CounterPath. Disponível em: https://www.counterpath.com/bria/ (2024)
12. Hasby, M. A., Putrada, A. G., Dawani, F.: *The Quality Comparison of WebRTC and SIP Audio and Video Communications with PSNR.* In: Indonesian Journal on Computing, vol. 6, no. 1, pp. 73–84. https://doi.org/10.34818/INDOJC.2021.6.1.549 (2021)
13. Fowdur, T. P., Ramkorun, N., Chiniah, P. K.: *Performance Analysis of WebRTC and SIP-Based Audio and Video Communication Systems.* In: SN Computer Science, vol. 1, p. 362. https://doi.org/10.1007/s42979-020-00380-z (2020)
14. Sermersheim J.: *Lightweight Directory Access Protocol (LDAP): The Protocol*. In: RFC 4511 (2006)