UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Autenticação Federada com Keycloak e Active Directory

João Lucas Teta

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutor Alan Oliveira de Sá
Prof. Doutor Mário João Barata Calha

2025

Resumo

Esta dissertação insere-se no contexto do Departamento de Informática (DI) da Faculdade de Ciências da Universidade de Lisboa, cuja infraestrutura atual, baseada em Active Directory (AD) on-premises, apresenta limitações em termos de resiliência, interoperabilidade e suporte a protocolos modernos de autenticação. Num cenário de crescente adoção de serviços em nuvem e necessidade de acesso federado a aplicações externas, torna-se essencial modernizar o sistema de gestão de identidades e acessos (IAM) e garantir a continuidade dos serviços críticos.

Para responder a este desafio, foi concebida uma solução de autenticação federada que integra o AD existente com o Keycloak, uma plataforma open-source de IAM, em ambiente híbrido. Foram avaliadas duas alternativas arquiteturais: (i) Keycloaks a partilhar uma base de dados gerida por Patroni, assegurando consistência e failover automático; e (ii) Keycloaks independentes, cada um com a sua base de dados, federados ao mesmo AD.

Os resultados obtidos em testes laboratoriais sob carga indicam latências médias entre 0.48 s e 1.1s, taxas de erro inferiores a 0.5% e taxas de processamento até 90 requisições por segundo, valores compatíveis com os requisitos de uma experiência de utilizador fluida e responsiva.

Conclui-se que ambas as arquiteturas são viáveis: a primeira destaca-se pela robustez e consistência dos dados, enquanto a segunda oferece simplicidade operacional e menor complexidade de implementação. Esta análise comparativa fornece contributos relevantes para a adoção de soluções híbridas de IAM em contextos institucionais, conciliando segurança, disponibilidade e interoperabilidade entre ambientes locais e na nuvem.

Palavras chave: Autenticação Federada, Keycloak, Active Directory, IAM, Infraestrutura Híbrida

Abstract

This dissertation is set in the context of the Informatics Department (DI) of the Faculty of Sciences, University of Lisbon, whose current infrastructure — based on an on-premises Active Directory (AD) — presents limitations in terms of resilience, interoperability, and support for modern authentication protocols. In a scenario of growing cloud adoption and the need for federated access to external applications, it becomes essential to modernize the Identity and Access Management (IAM) system and ensure the continuity of critical services.

To address these challenges, a federated authentication solution was designed, integrating the existing AD with Keycloak, an open-source IAM platform, within a hybrid architecture. Two architectural hypotheses were implemented and evaluated: (i) Keycloak instances sharing a PostgreSQL database managed by Patroni, ensuring consistency and automatic failover; and (ii) independent Keycloak instances, each with its own database, federating the same AD.

Experimental load tests revealed average response times between 0.48 s and 1.1s, error rates below 0.5%, and throughput up to 90 requests per second — values consistent with a smooth and responsive user experience.

Results show that both approaches are feasible: the first excels in robustness and data consistency, while the second stands out for its operational simplicity and lower complexity. This comparative analysis provides valuable insights for adopting hybrid IAM solutions in institutional contexts, combining security, scalability, and interoperability across on-premises and cloud environments.

Keywords: Federated Authentication, Keycloak, Active Directory, IAM, Hybrid Infrastructure

Agradecimentos

Chegado ao final desta jornada académica, não posso deixar de expressar a minha sincera gratidão a todos os que, de diferentes formas, contribuíram para a concretização deste trabalho.

Em primeiro lugar, aos meus orientadores, pela dedicação, acompanhamento e disponibilidade constante ao longo de todo o processo. Cada sugestão, cada comentário e cada reunião foram fundamentais para moldar este projeto e para me guiar com clareza através dos desafios que foram surgindo. Trabalhar sob a vossa orientação foi não apenas um privilégio, mas também uma experiência profundamente gratificante e enriquecedora.

Aos meus pais, pelo exemplo de perseverança e valores que sempre me transmitiram. Um agradecimento especial ao meu pai, cuja vocação como professor foi uma fonte constante de inspiração e motivação para seguir em frente, mesmo nos momentos mais exigentes.

À minha mulher, pelo amor, paciência e apoio incondicional. Foi um verdadeiro pilar ao longo desta caminhada, aceitando comigo as mudanças e desafios que este mestrado implicou — incluindo a mudança de país — e oferecendo sempre compreensão e encorajamento.

Aos meus filhos, pela alegria que dão ao meu quotidiano e por serem, em tantos momentos, a razão para continuar. Este percurso é também vosso.

Por fim, uma nota pessoal: a clareza com que procuramos expressar as ideias reflete a clareza com que as compreendemos. Inspirado na Lei de Kidlin — que afirma que, ao conseguirmos escrever um problema com clareza, já temos metade da sua solução alcançada — procurei fazer da escrita deste trabalho um exercício de entendimento. Se escrever é compreender, então cada frase deste trabalho foi também um passo rumo à solução.

A todos, o meu mais profundo agradecimento.

Índice

ÍNDICE

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A autenticação federada tem-se destacado como uma solução essencial para simplificar e centralizar a gestão de autenticação e autorização em ambientes híbridos, combinando infraestruturas locais e em nuvem. Essa abordagem garante uma experiência consistente de autenticação para os utilizadores, independentemente de onde os serviços estejam alocados, ao mesmo tempo que assegura maior segurança e resiliência. Ferramentas como o Active Directory (AD) e as plataformas de gestão de identidade e acesso (IAM) – por exemplo, o Keycloak – têm sido amplamente utilizadas para implementar sistemas de autenticação federada, aproveitando protocolos como OAuth 2.0, OpenID Connect e SAML para habilitar Single Sign-On (SSO) e outras funcionalidades avançadas . Com SSO, um utilizador autentica-se apenas uma vez e obtém acesso a múltiplos sistemas e aplicações, eliminando a necessidade de múltiplos logins e reduzindo a carga de gestão de credenciais.

1.1 Motivação

No contexto do Departamento de Informática (DI) da Faculdade de Ciências da Universidade de Lisboa (Ciencias - ULisboa), a principal motivação para esta proposta reside na garantia de resiliência dos serviços críticos, nomeadamente aqueles que suportam atividades letivas e administrativas, como o serviço de registo de contentores, bases de dados e outros sistemas essenciais. O AD existente on-premises funciona como fonte primária de autenticação para os recursos internos; contudo, para modernizar a arquitetura e aumentar a disponibilidade dos serviços, considerou-se integrar um serviço IAM na nuvem que funcionasse de forma federada com o AD .

Um dos principais desafios desta transição é assegurar que os utilizadores mantenham uma experiência de autenticação consistente e ininterrupta. A solução proposta envolve a integração do Keycloak – uma solução IAM open-source – na nuvem, operando como Identity Provider (IdP) federado em conjunto com o AD. Desta forma, o Keycloak na cloud poderá, em caso de falhas no ambiente local, assumir temporariamente o serviço de autenticação, garantindo que os serviços permaneçam acessíveis mesmo diante de problemas na infraestrutura on-premises . Em situações normais, o Keycloak atuará como ponte entre aplicações modernas (na nuvem ou locais) e o AD on-premises, fornecendo SSO unificado e suporte a protocolos modernos que o AD, por si só, não disponibiliza nativamente (como OAuth 2.0 e OpenID Connect).

1.2 Infraestrutura do DI

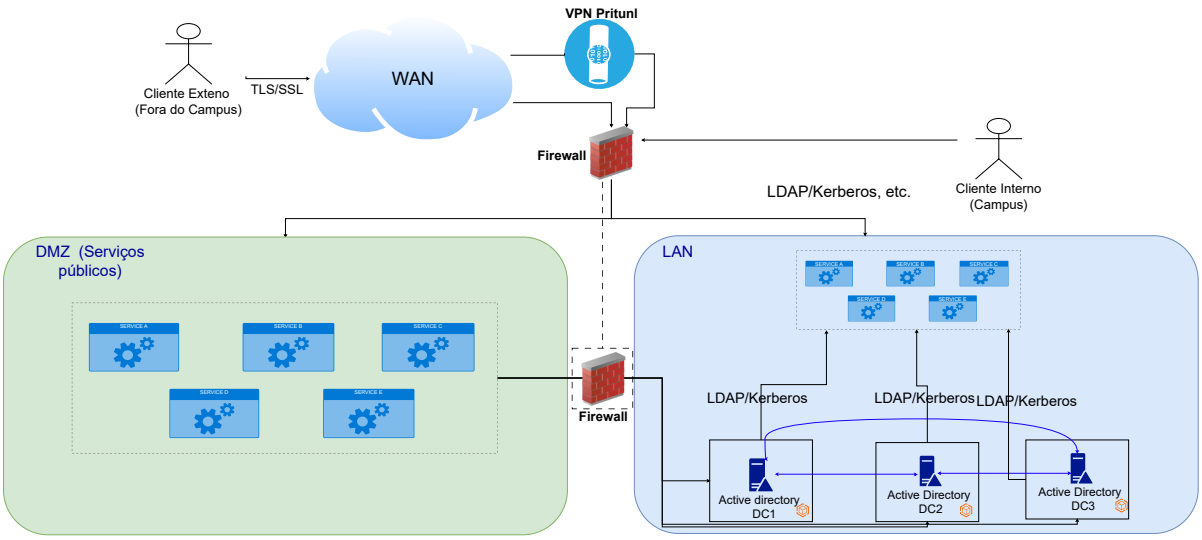


Figura 1.1: Arquitetura atual do DI.

Atualmente, o DI/Ciências – ULisboa opera com três controladores de domínio Active Directory (AD) on-premises, distribuídos por três máquinas configuradas em replicação multi-master, assegurando redundância local (Figura 1.1). Estes três *Domain Controllers* (DC) garantem alta disponibilidade do serviço de diretório — caso um deles falhe, os restantes continuam a autenticar utilizadores e a manter a consistência das informações de identidade na rede interna.

A autenticação é realizada através dos protocolos LDAP/Kerberos, sendo que os clientes internos comunicam através da LAN, enquanto o acesso remoto é efetuado por VPN, garantindo uma ligação segura à infraestrutura. Os serviços públicos, por sua vez, encontram-se segmentados na DMZ, protegidos por firewalls que filtram e controlam o tráfego entre as diferentes zonas de rede.

Apesar de funcional, esta arquitetura exclusivamente local apresenta limitações significativas para os desafios atuais do Departamento:

(i) Ausência de Integração Cloud – não existe suporte nativo para autenticação federada de aplicações em nuvem, dificultando a adoção de soluções *SaaS* e serviços externos;

(ii) Risco de Falha – uma interrupção geral na infraestrutura local (por exemplo, falha de energia prolongada ou problemas na rede do campus) pode isolar o AD de serviços externos e de utilizadores remotos, impossibilitando autenticações;

(iii) Protocolos suportados limitados ao legado (*LDAP/Kerberos*) – sem suporte direto a *OAuth2/OIDC*, *MFA*, ou outros protocolos modernos, exigindo adaptações customizadas para integração com aplicações web e APIs contemporâneas.

Estas limitações motivam a transição para uma arquitetura híbrida, mais resiliente, flexível e alinhada com os padrões modernos de autenticação e gestão de identidades.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é projetar e implementar uma solução de autenticação federada que assegure a continuidade e a resiliência dos serviços críticos do Departamento de Informática (DI) da Faculdade de Ciências da Universidade de Lisboa, explorando uma arquitetura híbrida que combina a infraestrutura on-premises com recursos em nuvem.

1.3.2 Objetivos Específicos

- Estudar diferentes abordagens e tecnologias para sincronização de identidades entre ambientes local e nuvem;
- Implementar uma solução baseada em IAM (Identity and Access Management) que permita a replicação segura da autenticação na nuvem;
- Avaliar a solução proposta através de testes que comprovem sua eficácia em termos de disponibilidade, segurança e desempenho;
- Garantir que a transição para a nuvem seja transparente para os utilizadores e compatível com os sistemas existentes.

1.4 Metodologia

A metodologia adotada neste trabalho segue uma abordagem combinada, envolvendo pesquisa exploratória, implementação prática e validação experimental. O estudo está estruturado em três fases principais:

1.4.1 Pesquisa Bibliográfica

Inicialmente, foi realizada uma pesquisa bibliográfica para compreender as melhores práticas e tecnologias utilizadas na replicação de infraestruturas de autenticação para a nuvem. Serão analisados estudos académicos, artigos técnicos e documentação de ferramentas relevantes, tais como Keycloak, Opta, EntraID, OpenLDAP e outras soluções de gestão de identidades.

1.4.2 Desenvolvimento da Solução

Com base no conhecimento adquirido na revisão bibliográfica, será definida uma arquitetura que permita integrar a infraestrutura local existente do Departamento de Informática da Ciências ULisboa com um ambiente de backup na nuvem. O desenvolvimento será dividido em:

- Levantamento de requisitos e definição da arquitetura do sistema;
- Implementação de uma solução de sincronização de credenciais;
- Configuração da infraestrutura de autenticação na nuvem.

1.4.3 Avaliação e Validação

A solução proposta será avaliada por meio de testes que verificam sua viabilidade e desempenho. Os critérios de avaliação incluem:

- **Disponibilidade:** Capacidade do sistema em manter a autenticação funcional em caso de falha nos servidores locais;
- **Segurança:** Garantia de que a replicação da autenticação não compromete a privacidade dos utilizadores;
- **Experiência do Utilizador:** Transparência da transição entre autenticação local e em nuvem.

Os resultados obtidos serão analisados e discutidos, permitindo validar a eficácia da solução proposta e sugerir melhorias para implementações futuras.

1.4.4 Contribuições

As principais contribuições deste trabalho podem ser resumidas em três pontos:

- **Proposta e implementação de uma arquitetura híbrida de autenticação federada**, integrando Active Directory on-premises com o Keycloak em ambiente de nuvem, garantindo continuidade do serviço e resiliência perante falhas de componentes críticos (IdP, AD ou base de dados).
- **Avaliação experimental através de testes de carga e falha**, que permitiram medir a disponibilidade, latência e taxa de erros em diferentes cenários (incluindo falha total da infraestrutura local), demonstrando a eficácia da solução em termos de alta disponibilidade.
- **Análise comparativa de alternativas arquiteturais**, evidenciando os trade-offs entre simplicidade, robustez e complexidade operacional, oferecendo um guia prático para futuras implementações de soluções de IAM híbridas em instituições académicas ou empresariais.

1.5 Organização do documento

Este documento está estruturado da seguinte forma:

- **Capítulo 2 - Trabalhos Relacionados**: Apresenta a revisão da literatura e trabalhos académicos relevantes que fundamentam a pesquisa, destacando abordagens já existentes para autenticação federada.
- **Capítulo 3 - Arquitetura e Implementação**: Descreve a infraestrutura desenvolvida, os requisitos do sistema, os mecanismos de segurança adotados e a integração entre o Active Directory e o Keycloak.
- **Capítulo 4 - Resultados e Discussão**: Expõe os testes realizados para validar a solução proposta, analisando métricas de desempenho, segurança e resiliência do sistema.
- **Capítulo 5 - Conclusão e Trabalhos Futuros**: Apresenta as considerações finais sobre o estudo, destacando as principais contribuições e sugerindo melhorias e direções para pesquisas futuras.

Capítulo 2

Trabalhos relacionados

2.1 Revisão da Literatura

A gestão de identidade e acesso (IAM - Identity and Access Management) é um componente essencial na administração de sistemas e infraestruturas computacionais, especialmente em ambientes académicos e corporativos. A necessidade de um controle eficiente de autenticação e autorização tem impulsionado o desenvolvimento de diversas soluções baseadas em protocolos como OAuth 2.0, SAML e OpenID Connect.

A presente dissertação busca explorar soluções de IAM para o Departamento de Informática da Faculdade de Ciências da ULisboa, com foco na integração entre Active Directory (AD) e Keycloak. Para embasar essa análise, é essencial compreender os avanços e desafios apresentados em estudos anteriores e soluções já implementadas em diferentes instituições.

2.1.1 Autenticação Federada e Single Sign-On (SSO)

A autenticação federada é um modelo no qual múltiplos sistemas ou domínios estabelecem confiança mútua para compartilhar informações de identidade dos utilizadores. Em vez de cada aplicação manter o seu próprio mecanismo de login, um provedor de identidade (IdP) central (por exemplo, o Keycloak, ou um serviço como Google, Okta, etc.) autentica o utilizador e depois comunica às aplicações ou provedores de serviço (SP) que aquele utilizador já se encontra autenticado e possui determinadas permissões. Um dos principais benefícios dessa abordagem é a implementação de Single Sign-On (SSO): o utilizador realiza login uma única vez e, a partir daí, obtém acesso a vários serviços distintos sem necessidade de novas autenticações. Isso melhora significativamente a experiência do utilizador e a segurança, já que reduz o número de vezes em que as credenciais são inseridas e transmitidas .

Os mecanismos modernos de SSO e federação de identidade baseiam-se em protocolos padronizados. OAuth 2.0 é amplamente utilizado para delegação de autorização, permitindo que um utilizador conceda a uma aplicação (cliente) acesso limitado aos seus recursos noutro serviço, sem expor as suas credenciais. Sobre OAuth 2.0 surgiu o OpenID Connect (OIDC), uma camada de autenticação que define como obter informações de identidade do utilizador (perfil, email, etc.) de forma padronizada, tornando possível implementar autenticação federada e SSO de forma simples em aplicações modernas. SAML (Security Assertion Markup Language) é outro protocolo de federação, anterior ao OAuth, baseado em XML, muito utilizado para SSO corporativo – nele, o IdP emite assertions (asseverações) de autenticação que são confiadas pelos SPs. Por fim, protocolos legados como **LDAP** e Kerberos têm relevância em ambi-

entes corporativos clássicos: o AD, por exemplo, usa Kerberos internamente para autenticar utilizadores num domínio Windows e expõe serviços LDAP para consulta e gestão do diretório de utilizadores. Uma solução IAM contemporânea como o Keycloak consegue interoperar com esses protocolos, servindo de ponte entre sistemas legados e aplicações modernas .

No ecossistema atual do DI/Ciências - ULisboa, o Active Directory (AD) da Microsoft já existe como repositório central de identidades e autenticações para serviços on-premises. O AD é uma solução madura e largamente adotada em organizações e universidades para gestão centralizada de utilizadores, grupos e políticas de acesso. Entretanto, integrar o AD com aplicações em nuvem ou oferecer SSO multidomínio requer componentes adicionais, já que o AD por si só foi concebido para ambientes principalmente Windows/empresariais. A Microsoft oferece o Entra ID (Azure AD) – evolução cloud do AD – que permite federar identidades locais com a nuvem Microsoft, mas nessa transição pode haver custos e complexidades consideráveis se a organização não estiver disposta a migrar totalmente para a stack Azure.

Como alternativa open-source, destaca-se o Keycloak, um IdP desenvolvido pela Red Hat (hoje parte da família IBM) que fornece SSO e IDM (Identity Management) com suporte nativo a OAuth2, OIDC e SAML, além de integração com LDAP/AD. O Keycloak permite criar realms (domínios de segurança) para isolar conjuntos de utilizadores, configurar clientes (as aplicações que usam o IdP), definir fluxos de autenticação personalizados (incluindo suporte a autenticação multifator – MFA) e aplicar políticas de controlo de acesso centralizadas. Por ser open-source e extensível, Keycloak suporta customizações via SPI (Service Provider Interfaces), o que permite integrar lógica adicional – por exemplo, sincronização customizada de utilizadores ou validações específicas – conforme necessidades da organização. Em contraste com soluções comerciais (Okta, Auth0, ForgeRock, etc.), o Keycloak não tem custos de licenciamento e pode ser executado on-premises ou na cloud à escolha, sendo por isso atrativo para instituições académicas com restrições orçamentais, apesar de exigir esforço de implantação e manutenção próprios.

A Tabela 2.1 evidencia que as principais soluções IAM do mercado (Keycloak, Okta, Auth0, Azure AD/Entra ID e Amazon Cognito) convergem no suporte aos protocolos modernos de federação e SSO, nomeadamente OAuth 2.0, OpenID Connect e SAML, o que cobre a maioria dos cenários de autenticação em aplicações web e móveis. As diferenças mais relevantes surgem no modelo de entrega e na integração com sistemas legados: o Keycloak, por ser open-source e self-hosted, oferece maior controle e capacidade de customização, destacando-se também por suportar, além de LDAP, protocolos frequentemente presentes em ambientes institucionais, como Kerberos, RADIUS e CAS, bem como SCIM para provisionamento. Em contrapartida, soluções SaaS como Okta e Auth0 tendem a facilitar a adoção por disponibilizarem conectores, aplicações pré-integradas e SDKs, bem como escalabilidade automática, mas tipicamente com custos recorrentes por subscrição ou por utilizador e, em alguns casos, menor flexibilidade fora do ecossistema suportado. O Azure AD/Entra ID e o Amazon Cognito tornam-se particularmente atrativos quando a organização já se encontra ancorada, respetivamente, no ecossistema Microsoft ou AWS, beneficiando de integrações nativas, embora isso possa introduzir dependência do fornecedor e limitações de customização fora do respetivo ambiente. Assim, a tabela suporta que a escolha entre IAMs não se resume ao conjunto de protocolos suportados, mas sim ao compromisso entre autonomia e esforço operacional, integração com a infraestrutura existente (por exemplo, AD/LDAP), capacidade de personalização e custos.

IAM		Keycloak	Okta	Auth0	Azure AD	Amazon Cognito
Tipo		Open-source / Self-hosted	SaaS	SaaS	SaaS	SaaS
Suporte a Protocolos	OAuth2	✓	✓	✓	✓	✓
	OpenID Connect (OIDC)	✓	✓	✓	✓	✓
	SAML	✓	✓	✓	✓	✓
	LDAP	✓	✓(via integração com AD/LDAP)	✓(via integração com AD/LDAP)	✓(via integração com AD/LDAP)	✓(via integração com AWS Directory Service)
	Kerberos	✓	X	X	✓	X
	RADIUS	✓	X	X	✓	X
	CAS	✓	X	X	X	X
	SCIM	✓	✓	✓	✓	✓
Facilidade de Integração		Média (requer configuração manual)	Alta (muitos apps pré-integrados)	Alta (muitos apps e SDKs)	Alta (integra com MS ecosystem)	Média (integra com AWS, apps móveis)
Escalabilidade		Alta (com configuração e infraestrutura próprias)	Alta (escala automaticamente)	Alta (escala automaticamente)	Alta (especialmente em grandes empresas MS)	Alta (especialmente para apps móveis e web)
Custos		Gratuito (com custos de infraestrutura)	Assinatura paga, custo por utilizador	Assinatura paga, custo por utilizador	Assinatura paga, custo por utilizador	Custo por uso, ideal para integrações AWS
Customização		Alta (código aberto)	Média (configurações limitadas)	Alta (extensível via APIs)	Média (principalmente para MS ecossistema)	Média (limitado a serviços AWS)

Tabela 2.1: Comparação entre diferentes soluções IAM

2.1.2 Trabalhos Académicos e Soluções Existentes

Vários estudos na literatura e implementações reportadas demonstram a viabilidade e benefícios da utilização de um IdP como o Keycloak integrado a serviços legados como o Active Directory, bem como apontam desafios a considerar numa arquitetura federada híbrida.

Voicu et al., 2024 descrevem a adoção do Keycloak como IdP central na Universidade Técnica de Cluj-Napoca (Roménia), suportado por HAProxy em configuração de alta disponibilidade, complementado com integrações a Microsoft EntraID e eduGAIN. O trabalho valida, em contexto académico, a robustez do Keycloak como solução open-source de IAM, em linha com os objetivos desta dissertação.

Indu et al., 2018 apresentam uma visão abrangente sobre os mecanismos e desafios da gestão de identidades em ambientes de nuvem. A interoperabilidade entre diferentes sistemas de IAM – como integrar uma solução baseada em Keycloak com um AD existente – é destacada como essencial para mitigar ameaças como acessos não autorizados e gestão inadequada de permissões. Este trabalho sublinha a importância de padrões abertos e interoperabilidade para garantir um ambiente de SSO seguro, especialmente em organizações que gerem recursos tanto on-premises quanto em nuvem. De modo semelhante, Aldo-

sary et al., 2021 conduziram um inquérito sobre limitações comuns em sistemas de Federated Identity Management (FIM), identificando desafios práticos na implementação de IAM federado – por exemplo, a complexidade de configuração, dificuldades de integrar múltiplos domínios de segurança, e a necessidade de manter níveis elevados de segurança durante todo o processo de autenticação federada. Os autores também discutem soluções para mitigar essas limitações, como adoção de ferramentas robustas (p. ex. Keycloak) e o seguimento de boas práticas de configuração e gestão de credenciais. Essas análises reforçam a necessidade de planeamento cuidadoso ao estender a infraestrutura de autenticação para a cloud, garantindo que a resiliência buscada não introduza novas vulnerabilidades.

Divyabharathi et al., 2020 focaram especificamente numa revisão do Keycloak enquanto servidor de IAM, evidenciando a sua capacidade de centralizar autenticação e autorização em ambientes corporativos. O estudo destaca a integração do Keycloak com tecnologias como LDAP e AD, permitindo que organizações unifiquem credenciais de sistemas locais com serviços baseados na nuvem . Uma vantagem salientada é que, ao usar o Keycloak, elimina-se a necessidade de cada aplicação gerir utilizadores e permissões individualmente, passando essa responsabilidade para o IdP central – isso não só oferece uma experiência de utilizador mais fluida (login único) como facilita a aplicação de políticas de segurança consistentes em todos os serviços . Os autores comparam ainda o Keycloak com outras ferramentas IAM (p.ex., Shibboleth, WSO2 IS), concluindo que o Keycloak apresenta maior flexibilidade, facilidade de configuração e compatibilidade com padrões modernos . Essas características tornam-no uma solução robusta e escalável, especialmente adequada para organizações que procuram gerir identidades em infraestruturas híbridas com segurança e eficiência.

No que toca à integração de Active Directory e SSO, Karasawa et al., 2020 exploraram a gestão de contas num ambiente federado, enfatizando a utilização de SSO integrado a sistemas corporativos. Um ponto central do artigo é a utilização de protocolos como LDAP e Kerberos em conjunto com um IdP moderno. Ao integrar um IdP que suporta esses protocolos (como o Keycloak) com o AD, torna-se possível centralizar a autenticação – os utilizadores autenticam-se uma única vez via IdP e obtêm acesso a diversos serviços, inclusive aqueles que dependem de mecanismos legados do AD . O estudo de Karasawa et al. demonstra que essa abordagem é particularmente relevante em ambientes corporativos com múltiplos sistemas heterogéneos, pois a federação atua como cola entre plataformas diferentes, aumentando a eficiência e segurança sem obrigar a migração imediata de todos os sistemas para uma única tecnologia.

Thorgersen et al., 2021 , autores do livro “Keycloak – Identity and Access Management for Modern Applications”, compilam as principais funcionalidades e práticas de utilização do Keycloak em cenários reais. Eles destacam a facilidade de integrar protocolos modernos (OAuth2, OIDC, SAML) e legados (LDAP, Kerberos via AD) numa única plataforma . Essa capacidade torna o Keycloak uma opção ideal como intermediário entre sistemas legados (e.g., AD on-premises) e aplicações modernas, possibilitando que empresas modernizem gradualmente a sua infraestrutura de autenticação sem descontinuar subitamente os sistemas existentes. Kallela, 2008 já havia realizado uma comparação de diferentes soluções e protocolos de identidade federada (SAML, Liberty Alliance, Shibboleth, WSFederation, etc.), mostrando como esses padrões permitem criar ambientes SSO abrangentes. Embora algumas tecnologias tenham evoluído desde então, a análise de Kallela fornece o contexto histórico-tecnológico que explica a existência de vários padrões: a necessidade de interoperabilidade entre domínios e organizações, mantendo a segurança, levou ao surgimento de múltiplas abordagens. A solução proposta neste trabalho beneficia

desses avanços ao combinar o suporte do Keycloak a protocolos modernos com a compatibilidade retroativa a sistemas como o AD (via LDAP/SAML), criando assim uma ponte entre diferentes gerações de soluções IAM.

Alsadeh et al., 2022 propõem um modelo dinâmico de gestão de identidades federadas usando OpenID Connect. A sua abordagem visa integrar sistemas legados, como o AD, com IdPs modernos como o Keycloak, enfatizando a interoperabilidade e a segurança. Eles demonstram uma solução onde credenciais e tokens são negociados de forma dinâmica entre domínios, permitindo autenticação centralizada e simplificada sem comprometer a segurança dos dados dos utilizadores . Esta proposta alinha-se diretamente com os objetivos do presente trabalho: utilizar o OpenID Connect (via Keycloak) para fazer a ponte entre utilizadores/credenciais do AD e as novas aplicações cloud, garantindo uma transição suave e segura.

No contexto de aplicações emergentes, Gonçalves et al., 2023 apresentam uma solução de autenticação federada para o domínio da Internet das Coisas (IoT), usando o Keycloak para assegurar acesso seguro e gestão de dados em ambientes de agricultura inteligente. Embora o foco seja distinto (IoT agrícola), o estudo é pertinente pois comprova as vantagens de usar o Keycloak como IAM em cenários distribuídos: os autores mostram como o Keycloak pode integrar sistemas locais e remotos através de uma abordagem federada . Os princípios e benefícios relatados – por exemplo, segurança centralizada, escalabilidade na gestão de utilizadores, facilidade de integrar diversos dispositivos/serviços – são igualmente aplicáveis a ambientes corporativos tradicionais onde se combine AD e Keycloak. A mensagem é clara: uma arquitetura federada bem desenhada aumenta a segurança e escalabilidade mesmo em cenários heterogéneos ou geograficamente distribuídos.

Relativamente a arquiteturas híbridas visando alta disponibilidade, Rajba et al., 2024 discutem a importância de garantir elevada disponibilidade em sistemas de autenticação federada que integram instâncias locais e remotas de servidores de identidade. No âmbito do projeto SILVANUS, eles propõem uma arquitetura que combina o Keycloak com o AD para centralizar a gestão de identidades numa organização geograficamente distribuída . Nessa solução, o AD continua a ser utilizado para autenticação local (on-premises), enquanto o Keycloak serve os serviços baseados em nuvem – muito similar ao enfoque deste trabalho. Os resultados de Rajba et al. evidenciam a flexibilidade e robustez do Keycloak, permitindo gestão segura de utilizadores, políticas de acesso e inclusão de autenticação multifator (MFA), ao passo que o AD fornece um back-end confiável para autenticação local . Além disso, destacam que a combinação de instâncias distribuídas do Keycloak, devidamente sincronizadas, pode assegurar redundância e failover transparente em caso de falha de um dos ambientes, alinhando-se ao objetivo de alta disponibilidade que também perseguimos na nossa implementação. Resultados recentes de Mihai et al., 2024 reforçam esta visão, demonstrando que a utilização de Keycloak em cenários de alta disponibilidade melhora significativamente a postura de segurança global da infraestrutura de autenticação.

Complementarmente, estudos que não se focam diretamente em IAM, mas em mecanismos de disponibilidade de serviços, reforçam a pertinência das opções adotadas. Prasetijo et al., 2016, analisaram comparativamente algoritmos de balanceamento de carga em HAProxy, aliado ao Heartbeat, demonstrando que a solução é capaz de atingir tempos de failover na ordem dos 10 ms e melhorar métricas de throughput e latência. Estes resultados confirmam a robustez do HAProxy como mecanismo de encami-

nhamento e redundância, aplicável não apenas a servidores web, mas também — como nesta dissertação — à gestão de tráfego LDAP, IdP e Base de Dados.

Por fim, no que respeita à seleção de soluções IAM, a dissertação de Freire, 2021 oferece uma comparação prática entre plataformas como Okta, ForgeRock e Keycloak. Na análise de Freire, soluções comerciais como Okta e Auth0 sobressaem em funcionalidades avançadas – por exemplo, em User and Entity Behavior Analytics (UEBA), onde utilizam técnicas de análise de comportamento e deteção de anomalias para reforçar a segurança proactivamente. No entanto, embora reconheça esses pontos fortes, o estudo também salienta o custo e natureza proprietária dessas soluções. Por outro lado, o Keycloak, por ser open-source, oferece flexibilidade e um custo significativamente inferior, fatores cruciais para muitas organizações académicas ou empresas de menor porte . A tabela comparativa apresentada por Freire evidencia que o Keycloak atende a um amplo conjunto de requisitos de IAM com elevado nível de conformidade com padrões, carecendo apenas de alguns recursos adicionais que muitas vezes podem ser integrados via extensões ou configurações adicionais. No contexto deste trabalho, a escolha pelo Keycloak justifica-se tanto por essas vantagens de custo-benefício e flexibilidade, quanto pela possibilidade de instalar e controlar internamente a solução (evitando dependências de terceiros e permitindo personalizações específicas ao ambiente do DI/Ciencias - ULisboa).

2.2 Discussão das Soluções Relacionadas

A análise da literatura revela diferentes abordagens para a gestão de identidades e mecanismos de alta disponibilidade, cada uma com vantagens e limitações que importam discutir criticamente.

Indu et al. (2018) salientam a relevância da interoperabilidade e da adoção de padrões abertos no contexto de Single Sign-On (SSO), evidenciando benefícios claros em termos de mitigação de riscos. Contudo, reforçam que a complexidade de integração em ambientes híbridos e o risco associado a más configurações podem comprometer a segurança, o que se relaciona diretamente com os desafios do DI/FCUL no cenário atual.

De forma complementar, Aldosary et al. (2021) identificam boas práticas no âmbito do Identity and Access Management (IAM) federado, recomendando soluções robustas como o Keycloak. Ainda assim, sublinham as dificuldades inerentes à gestão de múltiplos domínios de segurança e a elevada complexidade inicial de implementação, aspectos que também surgem como barreiras em infraestruturas acadêmicas heterogêneas.

Vários estudos (Divyabharathi et al., 2020; Karasawa et al., 2020) enfatizam a centralidade do Keycloak como solução de autenticação e autorização, compatível com diretórios legados como AD/LDAP. Estas abordagens demonstram a flexibilidade do Keycloak, mas igualmente a necessidade de manutenção contínua e o risco de perpetuar dependências de protocolos antigos (LDAP/Kerberos), quando não acompanhadas de uma estratégia de modernização.

A dimensão da alta disponibilidade surge de forma mais explícita em trabalhos recentes. Rajba et al. (2024), no âmbito do projeto europeu SILVANUS, exploram arquiteturas resilientes que conjugam Keycloak e AD distribuídos, com suporte a MFA e failover transparente. Mihai et al. (2024) corroboram essa linha, evidenciando ganhos de segurança ao operar Keycloak em alta disponibilidade, embora reconheçam que a generalização depende fortemente do contexto infraestrutural de cada instituição.

Estudos de caráter prático, como os de Voicu et al. (2024), documentam implementações académicas reais (Roménia) com recurso a HAProxy, Keycloak e integração no eduGAIN, partilhando lições

relevantes para cenários institucionais. Contudo, salientam que a integração depende de recursos específicos, dificultando a transposição para contextos empresariais. De modo análogo, Prasetijo et al. (2016) avaliam soluções de failover com HAProxy e Heartbeat, demonstrando tempos de recuperação da ordem dos 10ms. Apesar de o estudo se centrar em servidores web e não diretamente em IAM, a evidência sugere aplicabilidade indireta em arquiteturas de autenticação.

Por fim, Freire (2021) contribui com uma análise comparativa entre Keycloak e soluções comerciais (Okta, Auth0, ForgeRock), destacando a competitividade do software livre, ainda que com limitações ao nível de funcionalidades avançadas (e.g., UEBA nativo). Este contraste reforça o dilema entre custo, flexibilidade e funcionalidades, recorrente em instituições acadêmicas com restrições orçamentais.

Em síntese, a literatura indica que o Keycloak se posiciona como uma solução flexível e alinhada com padrões abertos, mas que requer cuidados adicionais para mitigar riscos de configuração, garantir manutenção contínua e evoluir para suportar protocolos modernos e alta disponibilidade. Estes aspectos revelam-se particularmente relevantes para a modernização da infraestrutura de autenticação do DI/F-CUL, que enfrenta desafios semelhantes aos descritos nos estudos analisados.

Para complementar esta discussão qualitativa, a Tabela 2.2 apresenta uma síntese das tecnologias e metodologias identificadas nos trabalhos relacionados. O objetivo é oferecer ao leitor uma visão comparativa imediata sobre a cobertura de cada estudo, destacando quais abordaram explicitamente soluções como o *Keycloak*, integração com AD/LDAP, protocolos modernos (*OAuth2/OIDC, SAML*), ou ainda mecanismos de alta disponibilidade (e.g., HAProxy, failover, MFA). Esta representação permite identificar de forma clara as áreas de maior incidência da literatura — como a interoperabilidade entre sistemas legados e modernos — bem como lacunas relevantes, por exemplo a escassa exploração empírica de MFA, apesar do seu suporte nativo em plataformas como o Keycloak.

Tabela 2.2: Síntese das tecnologias e metodologias abordadas nos trabalhos relacionados.

Referência	*Keycloak*	*AD/LDAP*	*OAuth2/OIDC*	*SAML*	*HA/Failover*	*HAProxy*	*MFA*
Voicu et al., 2024	x	x	x	x	x	x	x
Indu et al., 2018		x		x			
Aldosary et al., 2021	x	x	x				
Divyabharathi et al., 2020	x	x	x				
Karasawa et al., 2020	x	x					
Thorgersen et al., 2021	x	x	x	x			
Kallela, 2008				x			
Alsadeh et al., 2022	x	x	x				
Gonçalves et al., 2023	x		x				
Rajba et al., 2024 (SILVANUS)	x	x	x		x		x
Mihai et al., 2024	x				x		
Prasetijo et al., 2016					x	x	
Freire, 2021	x		x	x			
Singh, 2023			x	x			
Kavuluri et al., 2025					x		

Capítulo 3

Arquitetura e implementação

Nesta secção, apresenta-se a arquitetura do sistema de autenticação federada desenvolvido, bem como detalhes da sua implementação. Primeiramente, são definidos os Requisitos do Sistema que orientaram o desenho. Em seguida, descreve-se a Infraestrutura montada (incluindo componentes locais, em nuvem, e tecnologias de suporte como HAProxy e VPN WireGuard). Apresentam-se então as Alternativas Arquiteturais consideradas durante a fase de design – três cenários distintos – e discute-se a seleção da arquitetura final. Por fim, exploram-se aspectos específicos de Segurança e Disponibilidade incorporados na solução.

3.1 Requisitos do Sistema

Com base nos objetivos delineados na secção 1.3 e no estudo do ambiente atual, foram estabelecidos requisitos técnicos para a nova arquitetura de autenticação:

- **Continuidade de Serviço:** A solução deve garantir que a autenticação permaneça funcional mesmo em caso de falha de componentes on-premises (por exemplo, indisponibilidade do AD local) ou da ligação entre ambientes. Ou seja, deve existir redundância suficiente para evitar um ponto único de falha no processo de autenticação.

- **Arquitetura Híbrida (Cloud e On-Premises):** A solução deve funcionar de forma integrada tanto com serviços localizados na infraestrutura on-premises (rede interna da faculdade) quanto com serviços e aplicações na nuvem. Deve permitir autenticar utilizadores para recursos em ambos ambientes, preferencialmente com SSO unificado.

- **Federacão AD–Cloud:** Deve ser possível federar o AD on-premises com a nova plataforma IAM na cloud. Isto implica sincronização de identidades (utilizadores, senhas ou hashes, grupos) ou ao menos um trust para que credenciais do AD sejam reconhecidas via IdP na nuvem. **LDAP** foi identificado como o método principal de integração (o Keycloak deverá atuar consultando o AD via LDAP/LDAPS para validar credenciais).

- **Sincronização de Dados e Persistência:** Os dados de utilizador (credenciais, atributos) devem permanecer consistentes entre on-premises e cloud. Idealmente, qualquer alteração (como criação/atualização de contas) num ambiente deve ser replicada no outro. Este requisito aponta para a necessidade de replicação do diretório AD para a nuvem (e.g., através de um controlador de domínio adicional na cloud) e/ou utilização de mecanismos de sincronização periódica.

- **Segurança:** A solução deve assegurar confidencialidade e integridade das comunicações de autenticação. Isso inclui o uso de criptografia na comunicação (por exemplo, uso de LDAPS – LDAP over SSL/TLS – entre Keycloak e AD; utilização de VPN cifrada para tráfego entre cloud e campus) e proteção das credenciais em repouso. Além disso, deve suportar MFA de forma centralizada via Keycloak, para potencial aumento da segurança no futuro.
- **Disponibilidade e Desempenho:** A arquitetura deve assegurar elevada disponibilidade do serviço de autenticação e tempos de resposta consistentes, mesmo perante falhas parciais ou picos de carga. Para tal, componentes críticos como o IdP (Keycloak) e os serviços de diretório devem evitar pontos únicos de falha, suportando redundância e comutação automática para instâncias saudáveis (e.g., múltiplas instâncias com mecanismos de balanceamento/failover). Em paralelo, o desempenho deve manter-se previsível, minimizando a latência introduzida pela federação (consultas LDAP remotas, validação de tokens, sincronização/replicação, etc.). O sistema deve ser validado com testes de carga e testes de falha, garantindo níveis aceitáveis de throughput, latência e taxa de erro tanto em condições normais como em cenários de degradação.
- **Compatibilidade e Transparência:** A introdução da nova camada de autenticação federada não deve obrigar à reformulação de todos os serviços existentes de imediato. Deve ser compatível com os métodos de autenticação já em uso (por exemplo, serviços web internos que hoje autenticam via AD LDAP ou Kerberos devem continuar a funcionar, possivelmente integrando-se gradualmente com o Keycloak via SAML/OIDC). Para os utilizadores finais, a transição deve ser transparente – idealmente, eles continuarão a usar as mesmas credenciais e perceberão apenas melhorias (como menos logins separados, interface de login modernizada, etc.).

Com estes requisitos em mente, passou-se à concepção de possíveis arquiteturas que os satisfizessem, descritas a seguir.

3.2 Infraestrutura e Alternativas Arquiteturais

A infraestrutura considerada consiste de dois ambientes principais: (1) o ambiente *on-premises*, correspondente ao datacenter/local da instituição, onde atualmente residem o Active Directory (AD) e diversos serviços legados; (2) o ambiente em *nuvem* pública (neste caso, uma máquina virtual na Google Cloud Platform, embora pudesse ser Azure, AWS ou outro), destinado a hospedar o Keycloak e um conjunto de serviços em regime de redundância, assegurando a sua disponibilidade mesmo em caso de interrupções de rede ou falhas na infraestrutura *on-premises*. A conectividade segura entre os dois ambientes foi estabelecida através de uma VPN privada WireGuard, em substituição da solução encontrada na infraestrutura local inicial, baseada em Pritunl. Este último assenta num modelo centralizado de rede privada virtual, em que todo o tráfego depende de um servidor central para encaminhamento, criando assim um ponto único de falha para a conectividade. Em contrapartida, o WireGuard adota uma arquitetura distribuída, suportando ligações *peer-to-peer* diretas entre os nós, eliminando a dependência de um hub central e aumentando a resiliência. Para além desta característica fundamental, o WireGuard distingue-se pela leveza do seu código, pela eficiência criptográfica moderna e pela facilidade de configuração, resultando em menores tempos de latência e menor sobrecarga em comparação com soluções tradicionais como Pritunl, OpenVPN ou IPSec conforme detalhado na tabela 3.2. Deste modo, a extensão da rede local para a *cloud* passa a ser realizada de forma mais robusta e eficiente, garantindo comunicação cifrada entre o Keycloak em *cloud* e o AD *on-premises*, bem como a replicação segura dos controladores

de domínio e a partilha/backup da base de dados do Keycloak, em conformidade com os requisitos de resiliência da nova arquitetura.

3.2.1 Alternativas propostas

Arquitetura 1

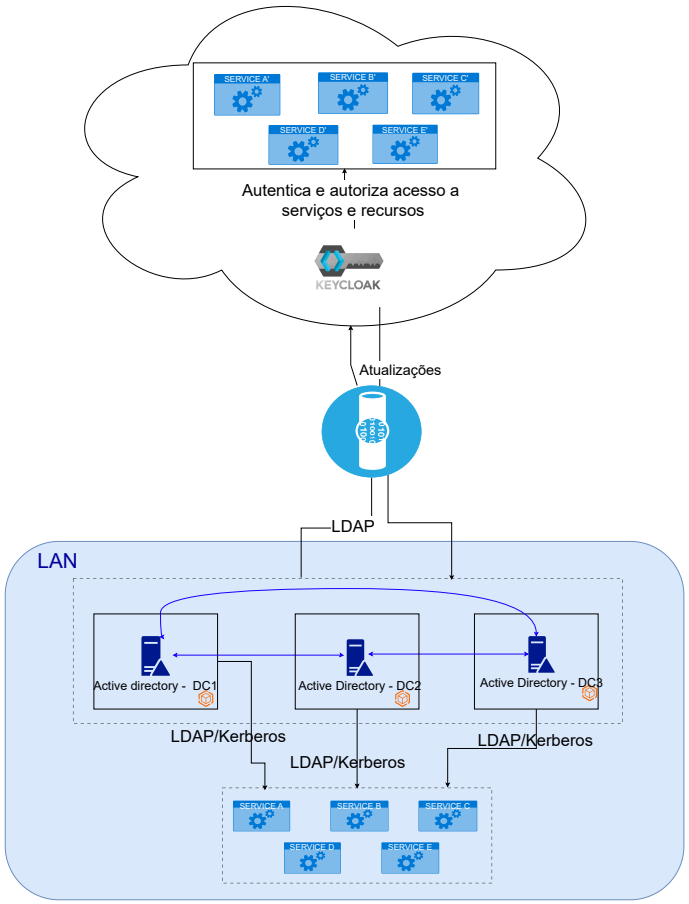


Figura 3.1: Keycloak na Nuvem com LDAP ao AD

Numa primeira abordagem arquitetural, o Keycloak seria implantado na infraestrutura cloud como IdP principal para novos serviços, enquanto o AD on-premises permaneceria intocado como fonte de dados de utilizador. O Keycloak comunicaria com o AD via LDAP para autenticar utilizadores, funcionando essencialmente como um frontend moderno para o velho AD. A Figura 3.1 ilustra este cenário. Os serviços em nuvem (*marcados como Service A', B', ...*) passariam a confiar no Keycloak para SSO; o Keycloak, por sua vez, delegaria a verificação de credenciais ao AD local (consultando os DCs via LDAP). Esta arquitetura tem a vantagem de uma transição gradual – não altera o processo de login dos serviços já existentes on-premises e introduz o Keycloak inicialmente apenas para novos serviços. Garante também continuidade durante a migração, pois o sistema legado (AD) continua operacional em

paralelo . No entanto, apresenta limitações em tolerância a falhas: se a conectividade com o AD local se perder (ou o AD ficar indisponível), o Keycloak na cloud não consegue autenticar utilizadores, pois depende exclusivamente daquele backend. Em termos de "Arquitetura Híbrida", o suporte é limitado – os serviços cloud tiram proveito do Keycloak, mas os serviços on-premises tradicionais não são integrados no SSO federado (a não ser por configurações adicionais para usarem o Keycloak). Resumidamente, a Arquitetura 1 melhora a escalabilidade e modernização (Keycloak aproveita recursos cloud e protocolos modernos) porém tem baixa tolerância a falhas fora do ambiente local e mantém certa compartimentação entre on-prem e nuvem.

Arquitetura 2

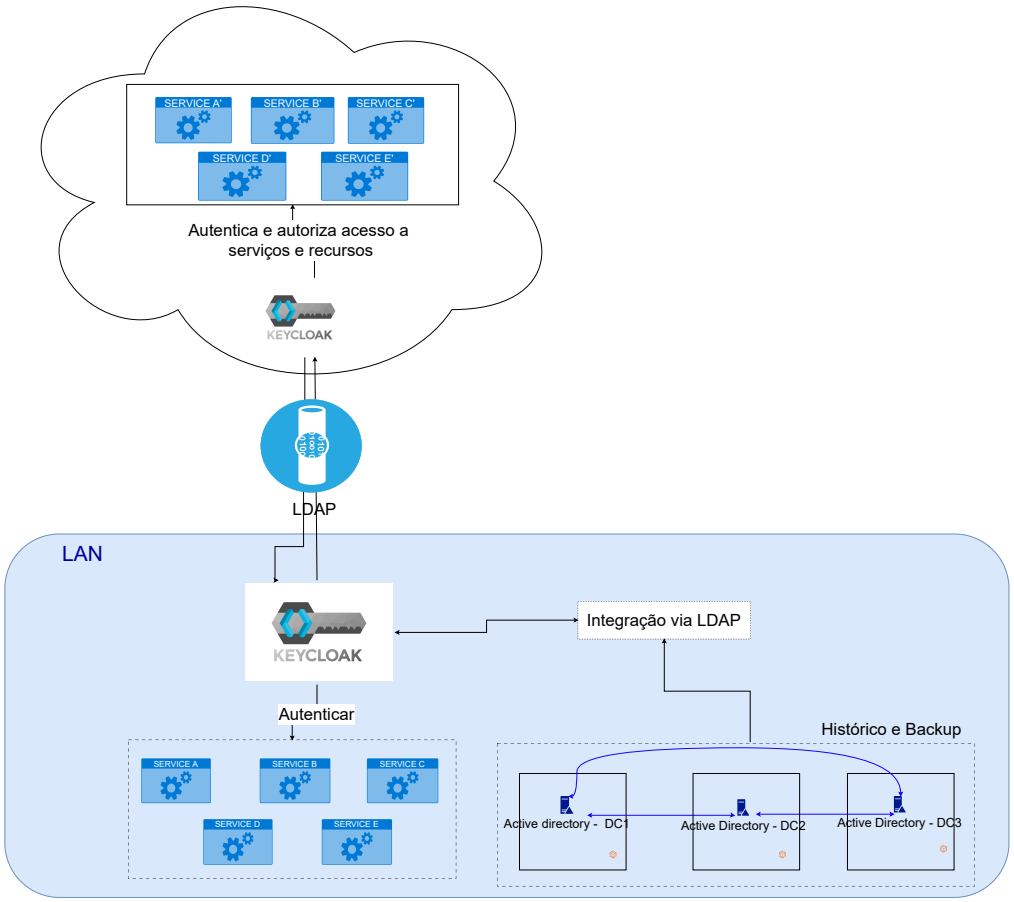


Figura 3.2: Keycloak local e na Nuvem (Autenticação totalmente via Keycloak)

Nesta abordagem, adota-se o Keycloak tanto localmente quanto na nuvem, replicando o serviço de IAM em ambos os ambientes. O Keycloak passaria a assumir completamente a responsabilidade pela autenticação e autorização de todos os serviços (novos e existentes), enquanto o AD gradualmente se tornaria apenas um histórico e backup dos dados de credencial. Na prática, os utilizadores seriam migrados para bases de dados geridas pelo Keycloak (que poderiam sincronizar periodicamente do AD ou até substituir o AD como fonte primária). Os AD locais permaneceriam apenas para consulta ou como repositório secundário. A Figura 3.2 ilustra esse cenário: os serviços locais e na nuvem autenticam contra o Keycloak (local ou nuvem, respectivamente), e o AD é mantido sincronizado (por exemplo, via importação/exportação de utilizadores) mas não é contactado diretamente pelas aplicações. Esta arquitetura

maximiza a continuidade de serviço em caso de falha de um dos ambientes, pois cada localização tem um IdP Keycloak funcional – se o campus ficar isolado, os utilizadores localmente poderiam eventualmente usar o Keycloak local; se o campus falhar, o Keycloak na nuvem continua a servir utilizadores externos. Também melhora a tolerância a falhas pelo facto de não depender de uma única base de dados (o AD torna-se dispensável em tempo real). Por outro lado, implica maior complexidade de implementação: é necessário manter dois Keycloaks sincronizados (o que pode ser feito com mecanismos de cluster federado ou sincronização de utilizadores via scripts/API) e garantir que as alterações (criação de novos utilizadores, mudança de senhas) se propagam. Além disso, o bootstrap inicial desta arquitetura envolve replicar credenciais do AD para o Keycloak (possivelmente via LDAP import), e os serviços existentes teriam de ser reconfigurados para delegar autenticação ao Keycloak – um esforço não trivial caso sejam muitos serviços heterogéneos. Em termos de custos, é uma solução mais onerosa comparada com a anterior, pois introduz mais componentes a gerir (duas instâncias Keycloak completas e suas bases de dados) e duplicação de funcionalidades já providas pelo AD.

Arquitetura 3

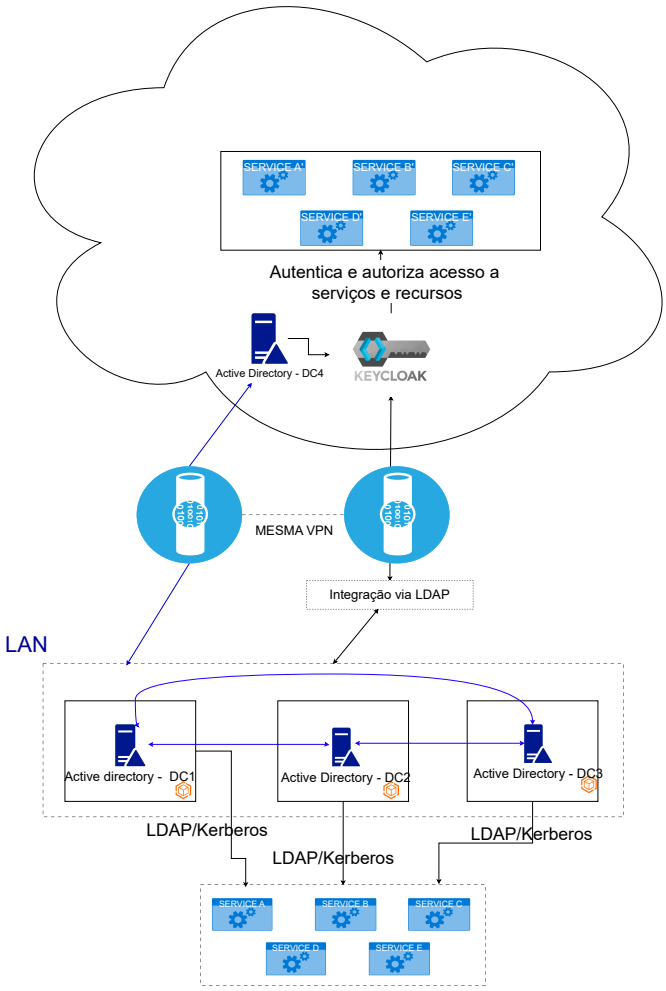


Figura 3.3: Keycloak na nuvem + Domain Controller AD na nuvem (Federação Completa)

Esta abordagem, ilustrada na figura 3.3, é aquela que busca fornecer redundância completa ao AD local sem substituí-lo inteiramente. Mantém-se um único IdP Keycloak (na nuvem), federado com o AD; contudo, adiciona-se um Controlador de Domínio adicional na Cloud que replica o AD local. Ou seja, estende-se o domínio Active Directory existente para um servidor na nuvem (via VPN segura), de modo que esse DC cloud seja membro do mesmo domínio e receba todas as sincronizações de objetos do AD (utilizadores, grupos, senhas). O Keycloak é configurado para integrar via LDAP não apenas com um endereço único de AD local, mas sim de forma redundante: a inclusão do DC cloud significa que, mesmo que os DCs locais fiquem inacessíveis, o DC na nuvem pode responder às consultas LDAP do Keycloak. Na prática, identificou-se a necessidade de introduzir um componente intermédio de alta disponibilidade entre o Keycloak e o Active Directory, tanto na arquitetura da figura 3.3, como nas arquiteturas das figuras 3.2 e 3.1, capaz de abstrair a seleção do DC a contactar e de garantir continuidade do serviço em caso de falha de um ou mais controladores. Em vez de o Keycloak depender diretamente de um único IP de AD, as consultas LDAP passam a ser encaminhadas para um ponto de entrada único, que distribui as requisições entre os vários DCs disponíveis (três locais + um na cloud). Em condições normais, as consultas LDAP podem ser atendidas por qualquer DC (espalhando carga, ou preferindo o local pelo menor tempo de resposta); se o site local falhar, este componente detecta a indisponibilidade dos DCs on-premises e direciona as consultas para o DC na nuvem restante. Esta Arquitetura 3.3 aumenta significativamente a tolerância a falhas – há redundância tanto do IdP (Keycloak em nuvem com uptime elevado) quanto do backend de diretório (AD replicado em locais distintos). A redunância em 4 DCs perminte nos ainda garantir uma maior disponibilidade geral do sistema de autenticação, tendo isto como desvantagem a inserção de um custo ligeiramente superior (manter uma VM extra como DC na cloud implica custos de infraestrutura e licenciamento Microsoft) e maior complexidade administrativa (é necessário configurar e manter a replicação do AD via site-to-site VPN, bem como o mecanismo de encaminhamento/failover LDAP). Ainda assim, o custo adicional foi considerado pouco significativo face aos ganhos de robustez, e a complexidade de implementação é contornável dado que adicionar um DC a um domínio AD é um procedimento bem documentado.

3.2.2 Discussão das alternativas

As três arquiteturas foram avaliadas de acordo com os requisitos definidos, e um resumo comparativo é apresentado na Tabela 3.1. A Arquitetura 3 destacou-se como a opção mais vantajosa no equilíbrio entre continuidade do serviço, resiliência e facilidade de migração. Apesar de envolver custos ligeiramente superiores (um servidor AD adicional na cloud), esse impacto orçamental é modesto e justifica-se pelos benefícios substanciais em redundância e disponibilidade . Ao adotar o Keycloak como solução central e estender o AD para a nuvem, modernizam-se os protocolos de autenticação disponíveis (por exemplo, passa a ser possível usar MFA centralizado via Keycloak) , ao mesmo tempo que se garante uma transição coerente e com menos pontos de falha. Importa notar que, considerando um cenário de migração completa no futuro, a Arquitetura 3 pode evoluir para incluir também um Keycloak on-premises (combinando-se com aspectos da Arquitetura 2) para redundância total do IdP – de fato, conforme sugerido no relatório preliminar, uma junção das arquiteturas 2 e 3 seria a estratégia ótima caso se pretendesse migrar integralmente o serviço de autenticação para o Keycloak, mantendo o AD apenas como contingência. Inicialmente, porém, optou-se por implementar o cenário 3, usando o AD on-premises e na cloud em federação completa com o Keycloak cloud, e preparando terreno para eventualmente acrescentar um Keycloak on-premises de reserva.

Requisito	Arquitetura 1	Arquitetura 2	Arquitetura 3
Continuidade do Serviço	Média (transição gradual, mas depende do AD local)	Alta (infraestrutura com redundancia de IdP)	Alta (resiliência total do AD e IdP)
Arquitetura Híbrida	Baixa (integração limitada - cloud apenas)	Média (suporte redundante)	Alta (federação completa)
Tolerância à Falha	Baixa (AD local é single point of failure)	Média (redundância híbrida de IdP, AD serve de histórico)	Alta (redundância do AD em 4 DCs, IdP em cloud)
Escalabilidade	Média (Keycloak na cloud escalável; AD local pode limitar)	Alta (Keycloaks redundantes e escaláveis)	Alta (Keycloak cloud escalável; AD escalável na cloud)
Segurança	Média (SSO básico; depende AD para MFA via métodos legados)	Alta (SSO redundante; Keycloak pode centralizar MFA)	Alta (MFA centralizado; redundância minimiza ataques DoS a um nó)
Custo	Baixo (infraestrutura simples, poucos componentes)	Médio (mais componentes: 2× Keycloak)	Alto (DC adicional na cloud + infra HAProxy)
Complexidade de Implementação	Baixa (configuração direta Keycloak->AD)	Alta (migração de utilizadores para Keycloak, sync complexa)	Média (adicionar DC + HAProxy; relativamente simples na prática)

Tabela 3.1: Comparação das Alternativas Arquiteturais Propostas (avaliação qualitativa de como cada arquitetura atende aos requisitos):

Como se depreende, a Arquitetura 3 atinge níveis altos em quase todos os requisitos (exceto no custo, que ainda assim não é proibitivo). Decidiu-se então implementar a Arquitetura 3, com alguns aprimoramentos adicionais de alta disponibilidade inspirados pela possibilidade de combinar com a Arquitetura 2. Concretamente, além do DC na nuvem, optou-se por instanciar também um Keycloak local de backup, e configurar um HAProxy para balancear as instâncias do Keycloak – dessa forma, obtém-se redundância total também do lado do IdP. A próxima seção detalha a arquitetura final implementada e como cada componente interage.

3.2.3 Topologia Simplificada

Com base na arquitetura atual 1.1 e nas alternativas arquiteturais analisadas (3.1, 3.2 e 3.3), definiu-se um modelo simplificado para efeitos de validação experimental. A Figura 3.4 representa esta topologia reduzida, concebida para os testes laboratoriais que vão ser apresentados mais à frente. Trata-se de um mapeamento lógico de alto nível, centrado apenas nos componentes e ligações essenciais, suprimindo elementos secundários que poderiam aumentar a complexidade da implementação.

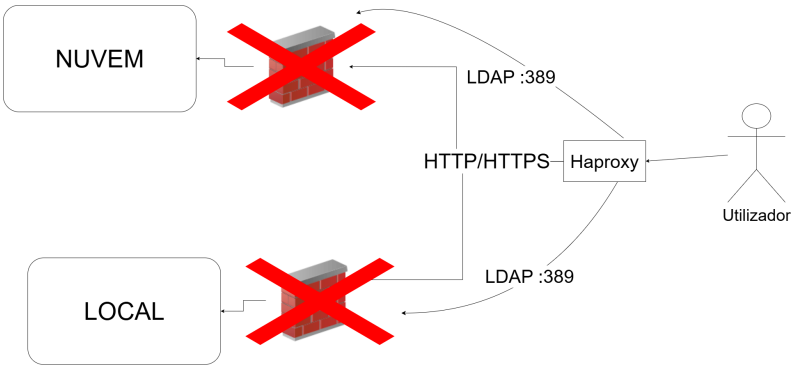


Figura 3.4: Topologia simplificada utilizada nos testes laboratoriais.

Entre as simplificações adotadas destaca-se a remoção da camada de firewalls, decisão tomada com o objetivo de reduzir a complexidade operacional e facilitar os testes de comunicação direta entre as instâncias locais e em nuvem, garantindo a replicabilidade do ambiente de forma controlada.

Ainda assim, a função das firewalls — controlo de tráfego e segmentação de zonas de segurança — permanece implícita no desenho lógico, sendo considerada essencial numa implementação em produção.

A topologia simplificada serviu, assim, como base para a implementação das hipóteses arquiteturais avaliadas na secção seguinte.

3.3 Arquitetura Final Implementada

3.3.1 Componentes Principais

Keycloak (IdP) – Instâncias em nuvem e local

Implementaram-se duas instâncias do *Keycloak* em contentores Docker (versão 20.x) – uma hospedada numa *VM* na nuvem (GCP) e outra numa máquina local dentro da rede do DI. Estas instâncias não operam em *cluster* sincronizado automaticamente; em vez disso, foram testadas duas abordagens diferentes quanto à sua sincronização e utilização da base de dados:

- **Hipótese 1 – Arquitetura com base de dados partilhada:** A figura 3.5 ilustra esta hipótese. Nesta configuração, ambas as instâncias do *Keycloak* utilizam uma base de dados PostgreSQL partilhada, acessível por ambas as infraestruturas, através de um *HAProxy* adicional em cada infraestrutura que encaminha o tráfego para a instância de base de dados ativa. A instância em *cloud*, funcionando como secundária, dispõe de uma base de dados de contingência configurada como réplica da principal, de forma a assumir automaticamente em caso de falha da infraestrutura *on-premises*.

  Para possibilitar esta comutação automática em caso de falha, foi implementado o *Patroni* (ver *Anexo C* para a configuração completa dos nós - documentação *Patroni Documentation*), em conjunto com o mecanismo de eleição de líder baseado em *Raft*. A inclusão de um terceiro nó (*wit-*

ness) permitiu assegurar o quórum necessário para decisões de failover, garantindo assim uma gestão automática da eleição da base de dados e uma replicação consistente entre instâncias.

Esta abordagem assegurava maior consistência e integridade imediata dos dados, com as sessões, *tokens* e configurações sempre sincronizados entre as instâncias de *Keycloak*. No entanto, apesar das suas vantagens, a sincronização contínua entre a base de dados principal e a de backup revelou-se uma das maiores dificuldades da solução, exigindo configurações delicadas, gestão cuidadosa de falhas, e mecanismos seguros de promoção e failback.

A escolha do PostgreSQL como base de dados partilhada nesta (e na hipótese seguinte) não foi apenas motivada pela sua natureza *open-source* e compatibilidade com o Keycloak, mas também pelo facto de apresentar um desempenho competitivo quando sujeito a mecanismos adequados de otimização e replicação. Estudos recentes de Kavuluri et al., 2025 demonstram que estratégias de *tuning* e gestão de conexões podem colocar o PostgreSQL em condições de desempenho equivalentes ou superiores a soluções comerciais como o Oracle em cenários de carga elevada. A adoção do *Patroni* nesta hipotese, com orquestração de failover e reciclagem de ligações, insere-se nessa linha de boas práticas, fornecendo não apenas alta disponibilidade, mas também maior previsibilidade de latências.

Desvantagens operacionais e estruturais.

Apesar das suas vantagens em termos de continuidade de serviço e experiência do utilizador, esta abordagem apresentou diversas limitações do ponto de vista técnico e de manutenção:

- *Elevada complexidade de configuração*: o correto funcionamento da solução depende da coordenação entre vários componentes (Keycloak, PostgreSQL, HAProxy, Patroni), exigindo conhecimento técnico aprofundado;
- *Acoplamento forte entre serviços*: qualquer alteração numa componente crítica (como o Patroni ou o HAProxy) pode comprometer todo o sistema;
- *Manutenção contínua*: falhas em qualquer um dos elementos requerem intervenção rápida e especializada;
- *Dificuldade na sincronização da base de dados de backup*: garantir a consistência entre a base de dados principal e a sua réplica, bem como gerir o *failover* de forma automática ou manual, constitui uma das maiores complexidades desta solução. No cenário real, não existem três nós para assegurar um quórum que permita uma gestão mais robusta e consistente, o que, apesar de a solução implementada se ter revelado funcional e eficiente, não está totalmente alinhado com os requisitos ideais para ambientes de produção.
- *Maior superfície de falha*: a base de dados partilhada torna-se um ponto crítico cuja disponibilidade depende de múltiplas camadas — rede, balanceadores e mecanismos de replicação;
- *Dificuldade de teste e simulação de falhas*: testar o comportamento em cenários reais de falha exige controlo total sobre a infraestrutura e ferramentas para monitorização em tempo real.

Estas limitações levaram à reavaliação da estratégia, favorecendo uma arquitetura mais simples e resiliente, mesmo que à custa de alguma conveniência na experiência do utilizador.

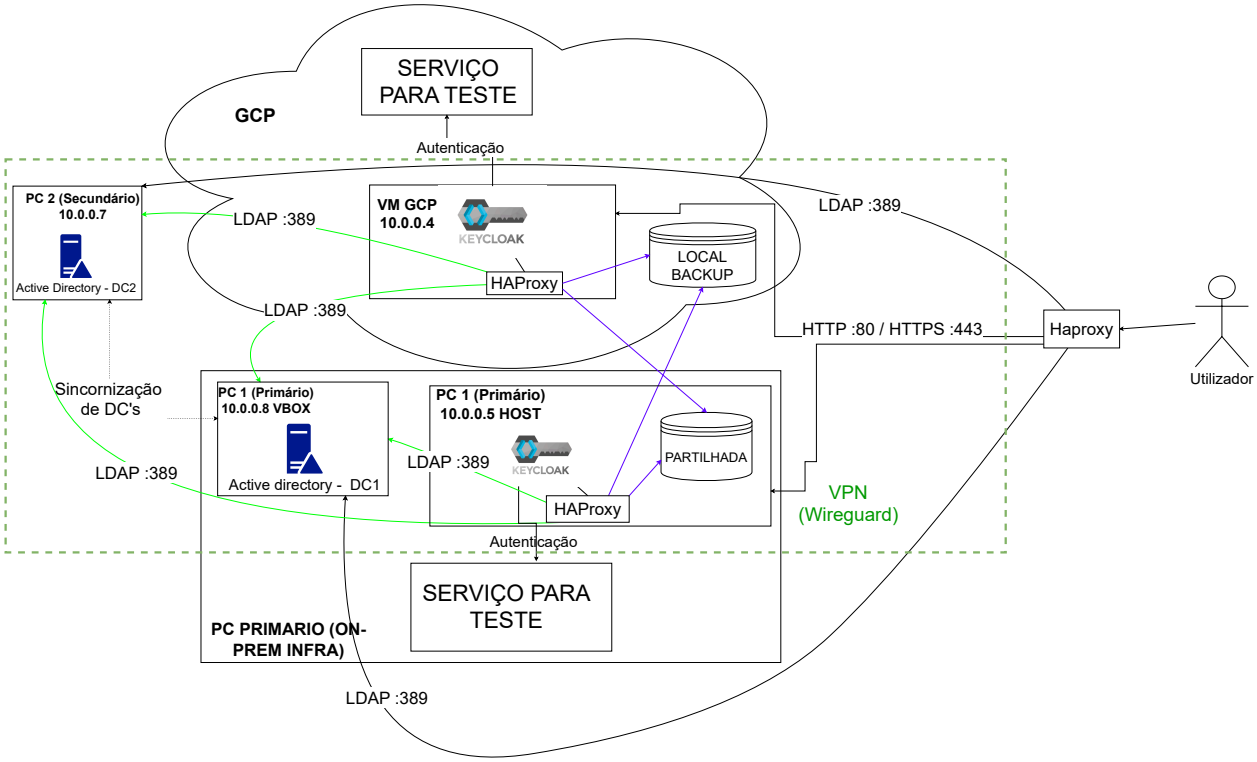


Figura 3.5: Arquitetura implementada na Hipótese 1, concebida para garantir maior robustez no processo de autenticação federada.

- **Hipótese 2 – Arquitetura simplificada com bases de dados independentes:**
  A figura 3.6 mostra a abordagem final adotada, resultante de uma análise de *trade-offs* entre robustez e simplicidade que pode ser analisada de forma resumida na tabela 3.4. Nesta arquitetura, cada instância do *Keycloak* (local e em nuvem) possui a sua própria instância de base de dados PostgreSQL, completamente isolada da outra. As bases de dados não se sincronizam diretamente; em vez disso, considera-se o *Active Directory* (federado via LDAP) como fonte de verdade dos dados dos utilizadores. Assim, a sincronização de contas e atributos é feita dinamicamente, a cada autenticação, diretamente a partir dos controladores de domínio, e periodicamente através de um *script* que corre entre as duas bases de dados. Esta arquitetura reduz significativamente a complexidade operacional, elimina pontos únicos de falha associados ao *HAProxy* da base de dados e ao *Keepalived* e requer menor intervenção humana para manutenção. A adoção desta estratégia como primeiro passo permite consolidar a autenticação federada baseada em DCs, adiando para uma fase posterior a complexidade adicional de sincronização de estado entre as instâncias do *Keycloak*.

  Consequências ao nível de sessões e tokens.

  Como cada instância mantém a sua própria base de dados, os *JWTs* emitidos por uma não são reconhecidos pela outra. Isto deve-se principalmente a:

  – *Chaves de assinatura distintas*: cada *realm* gera o seu par RSA por defeito;

  – *Issuer diferente (`iss`)*: o valor inclui o *hostname* de cada Keycloak;

  – *Estado de sessão não partilhado*: *refresh tokens*, revogações e metadados de sessão permanecem locais.

Consequentemente, em caso de falha da instância local é necessário que o utilizador volte a autenticar-se na instância de contingência — um custo considerado aceitável nesta fase em troca da redução drástica de complexidade.

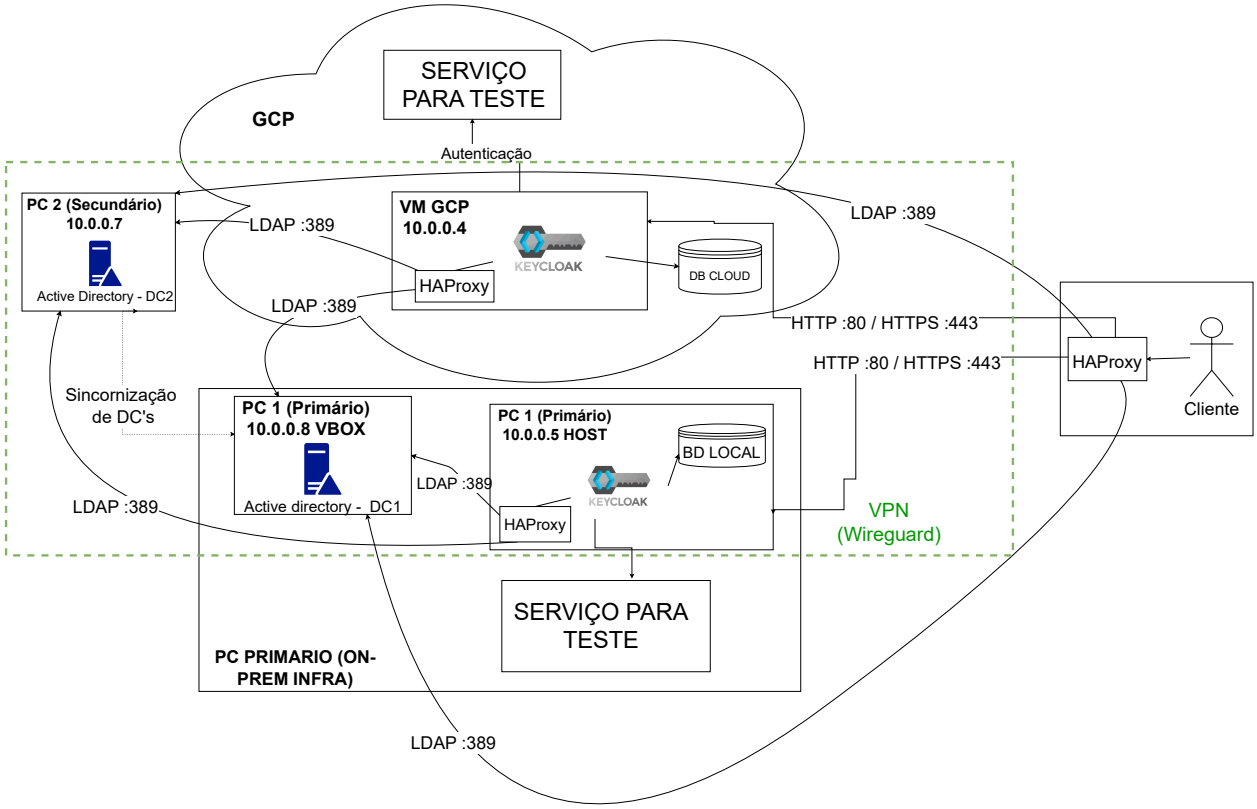


Figura 3.6: Arquitetura implementada na Hipótese 2, concebida com maior simplicidade e alinhada com as necessidades atuais da infraestrutura.

Foi ainda configurado um *HAProxy* (IdP LB) principal na rede local, responsável por expor um *endpoint* unificado (URL) para os clientes consumirem o IdP. Este *HAProxy* distribui o tráfego HTTP/HTTPS entre os dois *Keycloaks*, garantindo balanceamento de carga e *failover*: se a instância local ficar inacessível, a em nuvem responde, e vice-versa.

Tal como evidenciado em Voicu et al., 2024, a utilização de HAProxy como camada de balanceamento e failover em frente ao Keycloak é uma prática comum em ambientes universitários, assegurando redundância e gestão transparente de sessões.

Active Directory – DCs Locais e em nuvem

Foi estendido o *Domain Controller* existente localmente para uma *VM* na nuvem. Agora garantem-se duas instâncias replicadas do *Domain Controller* e, consequentemente, do *Active Directory*. A replicação nativa do AD assegura que todos os objectos de diretório (contas de utilizador, grupos, atributos, *hashes* de senhas) sejam continuamente sincronizados entre os DCs. Com isso, obtém-se redundância geográfica: se os DCs locais falharem, o DC na nuvem tem uma cópia atualizada do diretório. Inversamente, se a ligação de rede cair e algum utilizador alterar a senha localmente, quando a conexão for restabelecida, a mudança é propagada para a nuvem.

HAProxy para *failover* LDAP (DC's)

Implementou-se um *HAProxy* em cada ambiente (nuvem e local) para gerir o acesso LDAP aos controladores de domínio agora existentes. Cada *Keycloak* comunica com o *HAProxy* da sua própria infraestrutura, que por sua vez conhece todos os DCs – tanto locais como remotos. Este encaminha as requisições LDAP para um DC disponível, com preferência pelo local, por questões de latência. Caso este não esteja acessível, o *HAProxy* redireciona automaticamente para um DC remoto.

VPN WireGuard

Toda a comunicação sensível entre os componentes da solução — nomeadamente entre os *Keycloaks*, os controladores de domínio (DCs) e os *HAProxys* — é realizada através de uma VPN privada construída com recurso ao *WireGuard*. Esta VPN, de natureza *peer-to-peer*, permite estabelecer canais cifrados ponto a ponto entre os nós envolvidos, utilizando endereços IP estáticos definidos manualmente e algoritmos criptográficos modernos, como Curve25519 para troca de chaves e ChaCha20 para cifragem dos dados.

Inicialmente, foi considerada a utilização do *Pritunl* como solução VPN, uma vez que esta já se encontrava instalada na infraestrutura local. O *Pritunl* baseia-se numa arquitetura centralizada, em que todos os nós se ligam a um servidor VPN central. No entanto, esta abordagem revelou uma limitação crítica: a indisponibilidade do servidor central compromete automaticamente a conectividade de todos os restantes nós. Esta vulnerabilidade constitui um ponto único de falha, o que contraria os princípios de resiliência e alta disponibilidade exigidos pela solução.

Face a esta limitação, optou-se pela adoção do *WireGuard*, cuja arquitetura descentralizada oferece comunicações diretas entre os nós, sem dependência de um ponto de controle único. Esta característica garante maior robustez e flexibilidade operacional, bem como escalabilidade simplificada — qualquer novo nó pode ser adicionado à rede sem impactar os existentes. A decisão foi sustentada por documentação oficial do projeto *WireGuard*, assim como por diversas análises comparativas presentes na literatura.

Além disso, apesar de o LDAP utilizado não ser LDAPS (por se tratar de um ambiente de prototipagem e de menor complexidade), o tráfego entre os componentes permanece seguro ao circular dentro do túnel VPN cifrado. Para ambientes de produção, recomenda-se adicionalmente a configuração de LDAPS, reforçando assim a segurança na comunicação com os controladores de domínio.

Em termos de desempenho, os testes realizados indicaram que a latência introduzida pelo *WireGuard* é negligenciável, mesmo com a VM na nuvem localizada num centro de dados europeu.

A tabela 3.2 resume as principais diferenças entre as duas soluções analisadas.

Adicionalmente, foram analisadas na tabela 3.3, 4 diferentes soluções de VPN, onde também incluímos o Pritunl (*Pritunl Documentation*) e WireGuard (*Quick Start — WireGuard VPN*), incluindo desta vez o OpenVPN (*OpenVPN Reference Manual*) e o IPSec (*RFC 4301 - Security Architecture for the Internet Protocol*), de modo a identificar a mais adequada para a arquitetura proposta. O WireGuard mostrou-se o mais adequado pela sua natureza distribuída, elevado desempenho, criptografia moderna e configuração simples, resultados que estão em consonância com estudos feitos por Abdulazeez et al., 2020.

Tabela 3.2: Comparação entre *Pritunl* e *WireGuard*

Critério	**Pritunl**	**WireGuard**
Arquitetura	Centralizada (servidor principal)	Descentralizada (*peer-to-peer*)
Resiliência	Sujeita a ponto único de falha	Elevada tolerância a falhas
Facilidade de configuração	Interface gráfica amigável	Requer configuração manual inicial
Desempenho	Latência ligeiramente superior	Alto desempenho com baixa latência
Criptografia	Baseada em OpenVPN / IPsec	Curve25519, ChaCha20, moderno e leve
Escalabilidade	Limitada por servidor central	Elevada, sem impacto em nós existentes
Consumo de recursos	Mais elevado, dependente de camadas adicionais	Reduzido, código compacto e eficiente
Manutenção	Requer servidor dedicado, base de dados e sincronização	Apenas configuração entre pares

Tabela 3.3: Comparação entre diferentes soluções de VPN

Solução	**Arquitetura**	**Desempenho**	**Segurança**	**Complexidade / Gestão**
Pritunl	Centralizada (modelo servidor-cliente). Depende de um servidor intermediário.	Boa, mas com overhead devido ao encaminhamento central.	Suporta encriptação forte (baseada em OpenVPN/IPSec).	Fácil interface gráfica, mas cria ponto único de falha.
OpenVPN	Centralizada (servidor-cliente).	Desempenho moderado (maior overhead).	Muito segura, baseada em TLS/SSL.	Configuração relativamente complexa, mais pesada.
IPSec	Normalmente centralizada, ponto-a-ponto via gateways.	Bom desempenho, mas maior carga administrativa.	Padrão consolidado, muito seguro.	Configuração bastante complexa; difícil de gerir em larga escala.
WireGuard	Distribuída, conexões *peer-to-peer* entre nós.	Elevado desempenho (código leve, menos overhead).	Criptografia moderna (ChaCha20, Poly1305).	Configuração simples, não depende de ponto central; adequado para resiliência.

3.3.2 Fluxo de autenticação federada

Com a arquitetura final, o fluxo de autenticação federada ocorre conforme ilustrado na Figura **??**. Assume-se que o serviço está registado no Keycloak como client OIDC, que a comunicação com o diretório é realizada por LDAPS e que existe conectividade segura entre os ambientes on-premises e cloud.

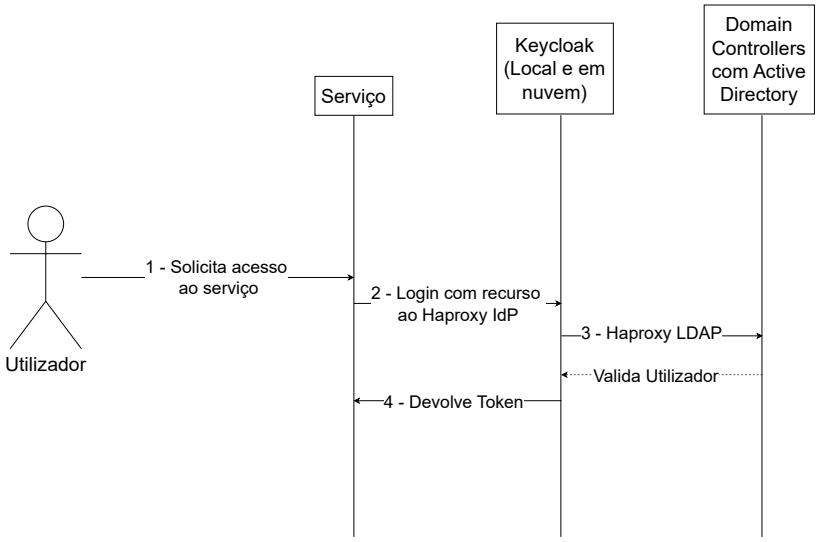


Figura 3.7: Fluxo de autenticação federada entre o utilizador, o(s) serviço(s), o Keycloak e os controladores de domínio com Active Directory.

1. Um utilizador tenta aceder a um serviço (por exemplo, um portal académico) integrado com o Keycloak; o serviço redirecciona o utilizador para o *endpoint* do IdP (fornecido pelo HAProxy IdP, que decide qual instância do Keycloak responde).

2. O utilizador introduz as credenciais na página de login do Keycloak.

3. O Keycloak consulta o seu *User Federation LDAP* para validar as credenciais; esta consulta é encaminhada para o HAProxy LDAP respetivo, o qual reencaminha o pedido para um dos Controladores de Domínio (DC) do Active Directory (por exemplo, um DC on-premises activo). O Active Directory valida o nome de utilizador e a palavra-passe (via *LDAP bind*) e responde ao Keycloak.

4. Em caso de sucesso, o Keycloak gera um *token* de autenticação (JWT) e cria uma sessão SSO para o utilizador, redireccionando-o de volta para o serviço original, com um *token* OIDC ou SAML, conforme configurado.

Vamos supor agora um cenário de falha: se todo o Active Directory local estiver indisponível ou inacessível, no passo (3) o HAProxy LDAP detecta a falha dos DCs locais e encaminha a consulta para o DC na *cloud*, o qual validará as credenciais, uma vez que possui os dados devidamente replicados. O utilizador obtém acesso sem notar qualquer diferença, evidenciando a resiliência da autenticação federada.

Num outro cenário, caso o Keycloak na *cloud* esteja indisponível, o HAProxy IdP utilizará automaticamente o Keycloak on-premises para o processo de login, o qual continuará a validar as credenciais contra o Active Directory local e a gerar os *tokens* necessários. Assim, garante-se a continuidade do serviço, mesmo perante falhas significativas num dos centros de dados.

3.3.3 Segurança integrada

Em termos de segurança, para além das comunicações cifradas através da VPN e da possibilidade de utilização de LDAPS, o Keycloak implementa diversas medidas, como políticas de palavra-passe (sincronizadas com as do Active Directory, quando configurado), configuração de timeouts e rotação de

tokens para mitigar o uso indevido de sessões. Suporta ainda a adição de mecanismos de autenticação multifator (MFA), a explorar mais adiante, como complemento às políticas existentes.

3.4 Considerações de Segurança

A arquitetura proposta teve a segurança como preocupação central em todas as camadas. Destacam-se, a seguir, as medidas e configurações adotadas para garantir um nível de segurança elevado:

- **Comunicação Segura:** Conforme referido, todo o tráfego entre o Keycloak, Active Directory (AD) e outros componentes ocorre dentro de um túnel VPN *WireGuard* cifrado. Esta abordagem previne ataques de *sniffing* ou *Man-in-the-Middle* na comunicação entre o IdP e o AD, sendo especialmente importante considerando que algumas dessas comunicações envolvem credenciais (via bind LDAP). Adicionalmente, na interface pública do Keycloak (acesso de utilizadores via navegador), é obrigatório o uso de HTTPS/TLS. O HAProxy IdP está configurado com certificados digitais válidos e reencaminha a comunicação via TLS até ao Keycloak. Assim, credenciais de autenticação e tokens de SSO estão protegidos durante a transmissão.

- **Controlos de Acesso e Políticas:** No Keycloak foram definidas políticas de acesso alinhadas com as do AD. Realizou-se a importação dos grupos do AD para o Keycloak, permitindo a este tomar decisões de autorização com base em papéis herdados do AD. Desta forma, permissões previamente existentes (como grupos de *Professores*, *Alunos*, etc.) são refletidas diretamente nos tokens emitidos. O Keycloak está também configurado para acesso apenas de leitura ao LDAP (AD), não podendo modificar entradas no diretório. Toda a gestão de contas continua a ser feita com as ferramentas habituais do AD, reduzindo o risco de alterações indesejadas provenientes do Keycloak. Adicionalmente, definiu-se explicitamente quais atributos do utilizador são inseridos nos tokens OIDC, minimizando a exposição de informação desnecessária.

- **Autenticação Multifator (MFA):** Embora não exigida inicialmente, a infraestrutura suporta MFA através do Keycloak. Foi testada a ativação de um segundo fator (via aplicação TOTP) para contas administrativas, adicionando uma camada de proteção contra o uso indevido de credenciais. Em contexto de produção, pode-se aplicar gradualmente a MFA a utilizadores privilegiados ou à totalidade dos utilizadores, de acordo com a política institucional.

- **Segurança no Active Directory:** A adição de um controlador de domínio na nuvem exigiu cuidados adicionais para evitar vulnerabilidades. Este servidor está protegido por *firewall*, permitindo apenas tráfego VPN e tráfego necessário para replicação do AD. A sincronização entre os DCs é segura por conceção (utilizando RPC autenticado por Kerberos sobre IPsec, quando via sites). Ainda assim, estando protegida pela VPN, a comunicação é cifrada. Verificou-se que as políticas de senha e bloqueio de conta definidas no domínio AD se aplicam igualmente no DC da nuvem, não havendo discrepâncias. Testou-se ainda o uso de um *Read-Only Domain Controller* (RODC), que mantém apenas cópias de leitura do diretório, limitando o impacto de um eventual comprometimento. Contudo, optou-se posteriormente por um DC completo, permitindo operações de escrita, já que a infraestrutura na nuvem também deverá permitir alterações, quando necessário.

- **Logs e Auditoria:** Tanto o Keycloak como o AD produzem registos de eventos de segurança, incluindo logins e falhas de autenticação. O Keycloak foi configurado com retenção de logs de administrador e eventos de login, permitindo auditorias de acessos federados (quem acedeu, a que

aplicação e quando). Estes logs podem ser integrados com sistemas de monitorização e análise de eventos (SIEM), proporcionando supervisão contínua. Foram também configurados alertas para eventos críticos, como falhas de conexão LDAP ou a indisponibilidade de um DC, permitindo resposta rápida por parte dos administradores.

Em suma, a solução implementada mantém (ou reforça) todos os controlos de segurança do sistema original e acrescenta novos mecanismos de proteção através do Keycloak. A federação de autenticação, quando devidamente configurada, não introduz vulnerabilidades adicionais. Pelo contrário, permite consolidar a segurança num ponto central (o IdP), com capacidades mais robustas e atualizadas do que cada aplicação isoladamente conseguiria oferecer.

3.5 Disponibilidade e Desempenho

A disponibilidade e o desempenho da solução foram analisados em dois eixos complementares: (i) continuidade do serviço de autenticação perante falhas parciais (serviços, nós e ligações entre sites), e (ii) previsibilidade de latência e capacidade de resposta sob carga. A arquitetura apresentada endereça estes objetivos da seguinte forma:

- **IdP Keycloak:** Para evitar pontos únicos de falha, o serviço de autenticação é disponibilizado através de instâncias redundantes de Keycloak (nuvem e local), permitindo continuidade do serviço caso uma instância se torne indisponível. Em cenários de maior carga ou manutenção planeada, podem ser adicionadas instâncias adicionais sem impacto na integração com o diretório, desde que exista um mecanismo de encaminhamento que selecione instâncias saudáveis.

  Ao nível da persistência, optou-se por reforçar a disponibilidade da base de dados subjacente através de replicação com *failover* automático. Para este efeito, foi adotado o Patroni, que gere a replicação, a promoção de réplicas e a recuperação de falhas no PostgreSQL, assegurando consistência e reduzindo o tempo de indisponibilidade em caso de falha do nó primário. Assim, a indisponibilidade de um nó de base de dados não implica, por si só, interrupção prolongada do serviço de autenticação.

- **Serviço de diretório (AD):** A redundância do backend de diretório é garantida pela existência de múltiplos controladores de domínio (DCs), distribuídos por três localizações on-premises e uma na nuvem. Esta distribuição aumenta a tolerância a falhas e permite que as consultas LDAP sejam atendidas por diferentes DCs, reduzindo o risco de indisponibilidade total do diretório.

  Em condições normais, a seleção do DC pode privilegiar a menor latência (preferência por proximidade), enquanto em caso de falha de um site local as consultas podem ser redirecionadas para DCs alternativos, mantendo o serviço funcional. Em termos de desempenho, a capacidade do diretório é ampliada ao distribuir consultas por múltiplos DCs, mitigando contenção em cenários de concorrência elevada.

- **Cache e sincronização:** Para melhorar o desempenho e reduzir dependência de consultas remotas ao LDAP, foi utilizado o mecanismo de *cache* do Keycloak para dados provenientes do diretório. Este *cache* diminui a latência de autenticação e reduz a carga sobre os DCs, sobretudo em logins repetidos num intervalo curto.

  Contudo, por questões de consistência (e.g., alterações de palavra-passe no diretório), o *cache* foi configurado com TTL moderado e políticas de invalidação, incluindo revalidação em caso de falha

de autenticação (forçando reconsulta ao diretório quando necessário), equilibrando desempenho com atualização atempada de credenciais.

- **Testes de desempenho e cenários de falha:** Antecipando a discussão da Secção 4, foram realizados testes de carga com múltiplos logins concorrentes via Keycloak (OIDC), avaliando latência, throughput e taxa de erro. Os resultados mostraram comportamento estável em condições normais, com erros residuais. O principal ponto de pressão observado ocorreu nas verificações LDAP sob concorrência extrema, sugerindo que, acima de determinado limiar, reforçar a capacidade do diretório (DC adicional) ou otimizar o padrão de consultas (e.g., *cache*/parâmetros LDAP) é prudente para manter latências previsíveis.

Em síntese, a arquitetura foi concebida para manter o serviço de autenticação disponível perante falhas parciais, ao mesmo tempo que preserva desempenho aceitável sob carga. A redundância do Keycloak e do diretório, combinada com replicação automática da base de dados e mitigação de latência via *cache*, reduz a probabilidade de interrupções e melhora a previsibilidade do tempo de resposta – alinhando-se assim com o requisito de disponibilidade e desempenho do sistema.

- **Continuidade de serviço:** assegurada pela redundância dos componentes críticos e pela capacidade de comutação automática em caso de falha, tanto ao nível do IdP como da camada de persistência e do diretório.
- **Arquitetura híbrida:** estabelecida pela interligação segura entre ambientes on-premises e cloud, permitindo autenticação federada para serviços locais e em nuvem.
- **Federação AD–Cloud:** concretizada através da integração LDAP entre o Keycloak e o serviço de diretório, suportada por replicação dos controladores de domínio e manutenção consistente de identidades.
- **Segurança:** garantida pelo uso de comunicações cifradas (por exemplo, LDAPS e túnel VPN), pela centralização de autenticação via Keycloak e pela possibilidade de evolução para MFA.
- **Disponibilidade e desempenho:** reforçados por redundância, replicação e mecanismos de encaminhamento que mantêm o serviço operacional sob falhas e asseguram latências previsíveis, mitigando a carga no diretório através de cache e otimizações de consulta.
- **Compatibilidade e transparência:** asseguradas pelo uso de protocolos standard (LDAP, OIDC, SAML), permitindo continuidade de integração com serviços legados e adoção progressiva do Keycloak por novos serviços.

Deste modo, a arquitetura concebida demonstra cumprir os requisitos estabelecidos inicialmente, fornecendo uma solução resiliente, segura e escalável de autenticação federada. No capítulo seguinte, esta arquitetura será sujeita a testes práticos para avaliar o seu desempenho e a sua capacidade de garantir a continuidade de serviço em diferentes cenários de falha.

Tabela 3.4: Comparação entre as duas arquiteturas implementadas

Critério	**Hipótese 1 – Base de dados partilhada**	**Hipótese 2 – Bases de dados independentes**
Complexidade de configuração	Elevada — exige coordenação entre Keycloak, PostgreSQL, HAProxy e Keepalived	Reduzida — cada Keycloak tem instância autónoma e configuração simplificada
Acoplamento entre componentes	Forte — falhas num único componente afetam todo o sistema	Fraco — cada instância opera de forma independente
Resiliência a falhas	Alta — com failover automático e continuidade de sessão	Moderada — falhas requerem reautenticação manual, mas não afetam a outra instância
Manutenção e operação	Exige monitorização ativa e intervenção especializada em caso de falha	Manutenção simplificada, com menos pontos de falha e menos dependências
Sincronização de dados e sessões	Total — sessões, tokens e configuração centralizados na BD partilhada	Limitada — sincronização parcial via LDAP e scripts periódicos
Validação de tokens entre instâncias	Possível — mesma chave de assinatura e `issuer`	Não possível — chaves distintas, `issuer` diferente e sessões isoladas
Escalabilidade	Limitada — a BD partilhada torna-se gargalo e ponto crítico de falha	Elevada — cada instância pode escalar horizontalmente de forma independente
Facilidade de teste e simulação	Difícil — envolve múltiplos serviços e coordenação de IPs virtuais	Simples — cada instância pode ser testada isoladamente
Adequação ao contexto atual	Boa para robustez e continuidade total de sessão, mas com maior custo técnico	Boa para primeira fase de implementação — reduz complexidade sem comprometer funcionalidade

Capítulo 4

Resultados e Discussão

Após a implementação das arquiteturas descritas, procedeu-se a um conjunto de testes de funcionalidades, resiliência e desempenho para validar a solução em ambiente de laboratório. Nesta secção, apresentamos os resultados desses testes e discutimos em que medida os objetivos foram alcançados, bem como as lições aprendidas e limitações identificadas.

4.1 Testes Realizados

Os testes foram realizados num ambiente controlado que simula um cenário híbrido com componentes em nuvem e em *campus/local*, interligados por VPN *WireGuard* (`wg0`). A topologia inclui: (i) um nó na Google Cloud Platform (GCP), (ii) um nó a *simular o campus*, (iii) uma VPS utilizada como *witness* no Patroni.

4.1.1 Topologia

A topologia é composta por quatro nós principais:

- **Node 1 (GCP)**: instância `e2-medium` com 2 vCPUs e 4 GB de RAM. Hospeda serviços de aplicação e/ou base de dados conforme a hipótese arquitetural em teste.
- **Node 2 (Campus simulado)**: portátil MacBook Pro (2014) convertido em `Ubuntu Server`, com 16 GB de RAM e CPU Intel i5. Este nó *simula o ambiente de campus/local*.
- **Node 3 (VPS Witness)**: plano KVM4 num fornecedor externo. Atua como *witness* no cluster Patroni para assegurar quórum em eleições e *failover*.
- **Node 4 (DC/AD simulado na cloud)**: máquina adicional configurada como *Domain Controller* (Active Directory). Representa o DC na cloud, *simulado localmente* para evitar custos nesta fase.
- **Node 5 (DC/AD local simulado)**: máquina configurada como *Domain Controller* para simular o ambiente local. Está ligado exclusivamente ao Node 2 (Campus), representando o domínio on-premises.

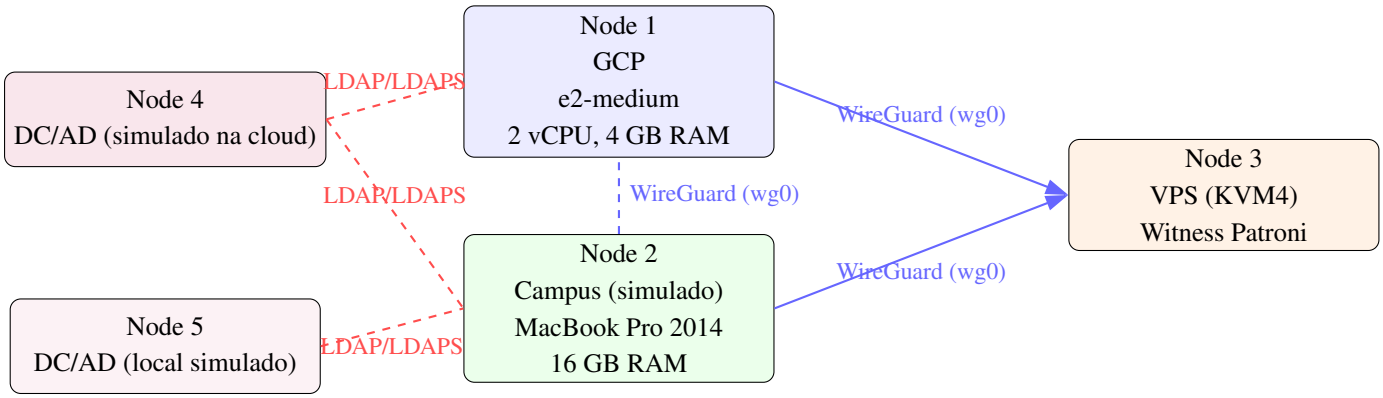


Figura 4.1: Topologia híbrida com cinco nós: GCP, campus (simulado), VPS/witness e dois DC/AD (cloud e local simulados) interligados por WireGuard.

4.1.2 Hipóteses avaliadas

As hipóteses avaliadas dizem respeito à *distribuição lógica* dos componentes na topologia da Fig. 4.1. Os componentes destas mesmas hipóteses foram apresentados em maior detalhe na Secção 3.3.1. Segue a síntese das hipoteses:

- **Hipótese 1 doravante referenciada por H1:** Keycloak local e Keycloak na nuvem a usar uma base de dados partilhada (PostgreSQL 17) gerida por Patroni para replicação e *failover* conforme ilustrado na figura 3.5.
- **Hipótese 2 doravante referenciada por H2: Keycloaks independentes**, cada um com a sua BD, consumindo apenas dados de utilizador via federação LDAP para os mesmos Controladores de Domínio (AD) conforme ilustrado na figura 3.6.

4.1.3 Ambiente de Teste

Redes e VPN Duas redes virtuais uma local e outra em nuvem, interligadas por WireGuard (*Quick Start — WireGuard VPN*).

```
# Exemplo (resumo) de wg0.conf
[Interface]
Address = 10.0.0.4/24
PrivateKey = <...>
ListenPort = 51820

[Peer]
PublicKey = <...>
AllowedIPs = 10.0.0.0/24, 10.0.1.0/24
Endpoint = cloud.example.com:51820
PersistentKeepalive = 25
```

Mais detalhes desta configuração podem ser vistos em anexo A.2

HAProxy (IdP e LDAP) Um HAProxy, versão 2.9 (*HAProxy Configuration Manual — HAProxy 2.9*) em cada lado (local/nuvem) a encaminhar:

```
#Exemplo
# Frontend IdP (Keycloak)
frontend kc_front
  bind *:443 ssl crt /etc/ssl/certs/fullchain.pem
  default_backend kc_back

backend kc_back
  option httpchk GET /realms/master
  server kc_local  10.0.0.10:8443 check
  server kc_cloud  10.0.1.10:8443 check backup

# Frontend LDAP
frontend ldap_front
  bind *:389
  default_backend ldap_back

backend ldap_back
  mode tcp
  option tcp-check
  server dc_local1 10.0.0.20:389 check
  server dc_local2 10.0.0.21:389 check
  server dc_cloud  10.0.1.20:389 check backup
```

Mais detalhes desta configuração podem ser vistas em anexo B

Active Directory Dois DCs locais e um DC na nuvem, com replicação multi-site ativa. O AD contém utilizadores/grupos de teste (Professores, Alunos, ...).

Keycloak Duas instâncias: `kc-local` (on-prem) e `kc-cloud` (nuvem). Realm do Departamento, cliente OIDC da Aplicação de Teste (Quarkus) e federação LDAP via HAProxy LDAP.

Base de Dados

- **H1:** PostgreSQL 17 gerido por Patroni (líder e réplica). Endereço virtual exposto ao Keycloak.
- **H2:** BDs locais a cada Keycloak (sem replicação). Dados de utilizador vêm apenas do LDAP.

Aplicação de Teste (Quarkus) Aplicação OIDC com Keycloak Client configurado para o HAProxy IdP como ponto de entrada único. Usada para exercícios de login/SSO.

4.1.4 Cenários e Procedimentos

4.1.4.1 C1. Autenticação básica e SSO

Objetivo: validar login OIDC e SSO entre aplicações.
Procedimento comum:

1. Abrir a Aplicação de Teste (*client* configurado no realm).

2. Autenticar com utilizador do AD (via Keycloak).
3. Abrir segunda aplicação protegida pelo mesmo IdP e verificar SSO sem novo login.

Expectativa: tokens válidos (claims: nome, email, grupos) e SSO funcional.
Resultado Observado:

- **H1:** Consistência total (atributos e sessões) devido à BD partilhada; experiência uniforme em ambas as instâncias.
- **H2:** Autenticação OK via LDAP; SSO funcional por partilharem o mesmo IdP (HAProxy à frente). Atributos locais específicos criados apenas num KC não aparecem no outro.

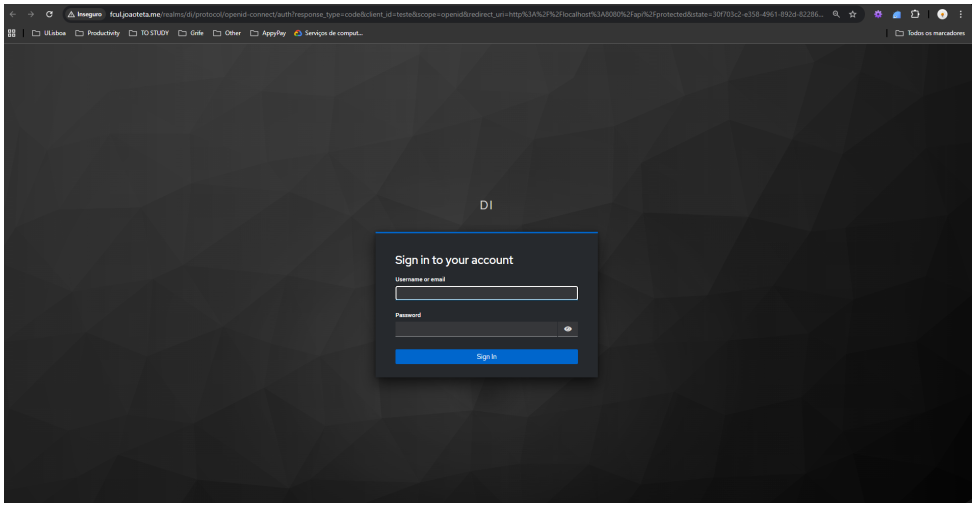


Figura 4.2: Login na aplicação de teste (Porta 8082)

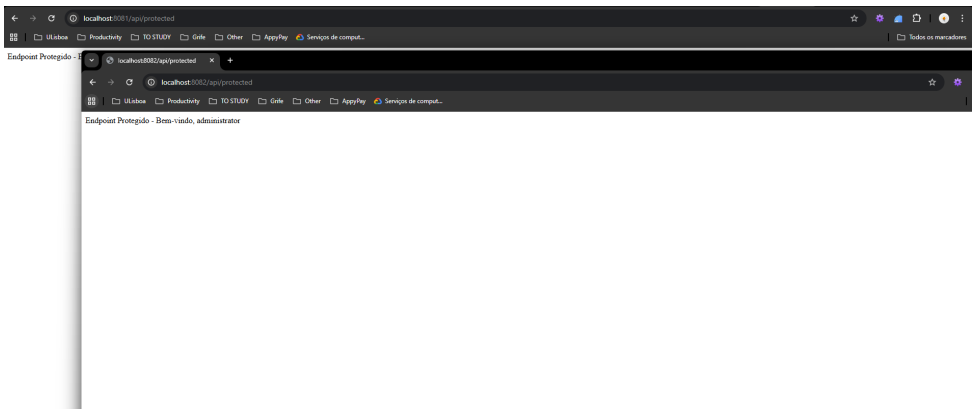


Figura 4.3: Token deu acesso a aplicação na porta 8081 tal como 8082

4.1.4.2 C2. Falha do AD local

Objetivo: Garantir a continuidade quando DCs locais estão indisponíveis.
Procedimento:

1. Parar temporariamente os DCs locais (ex.: desligar serviço AD).
2. Tentar novo login com OIDC.

Expectativa: O HAProxy LDAP encaminha os pedidos LDAP para o DC na nuvem. O login sofre um ligeiro aumento de latência, visto que, estamos a falar de comunicação com um DC fora da infraestrutura local.

Resultado Observado:

- **H1 e H2:** Login ocorre com sucesso em ambos os cenários.

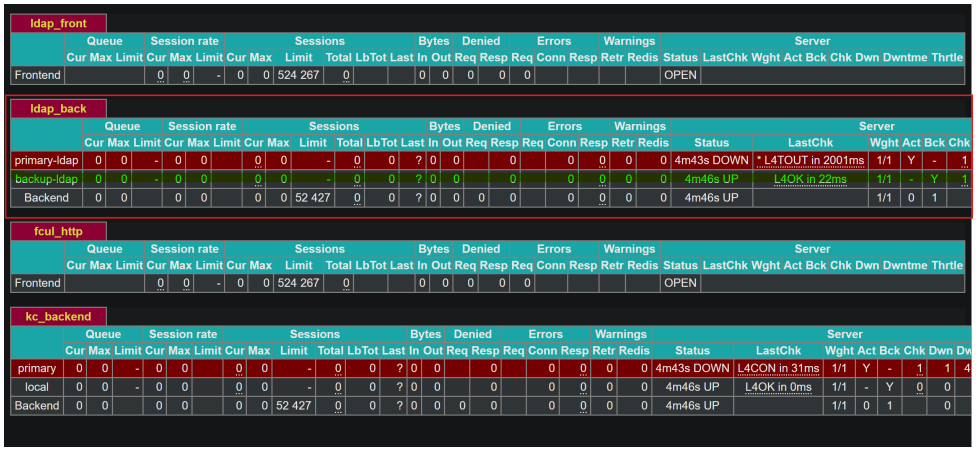


Figura 4.4: Admin panel do haproxy mostra o Ldap primário down e secundário up

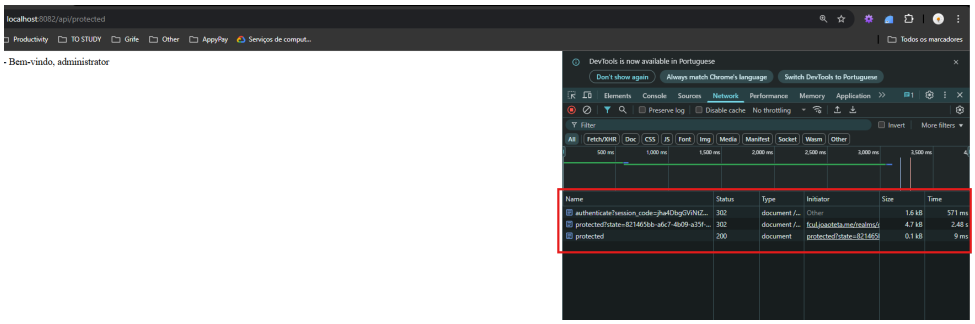


Figura 4.5: Login com sucesso. Tempos acima da média (571ms)

4.1.4.3 C3. Falha de uma instância Keycloak (on-prem)

Objetivo: validar *failover* do IdP.
Procedimento:

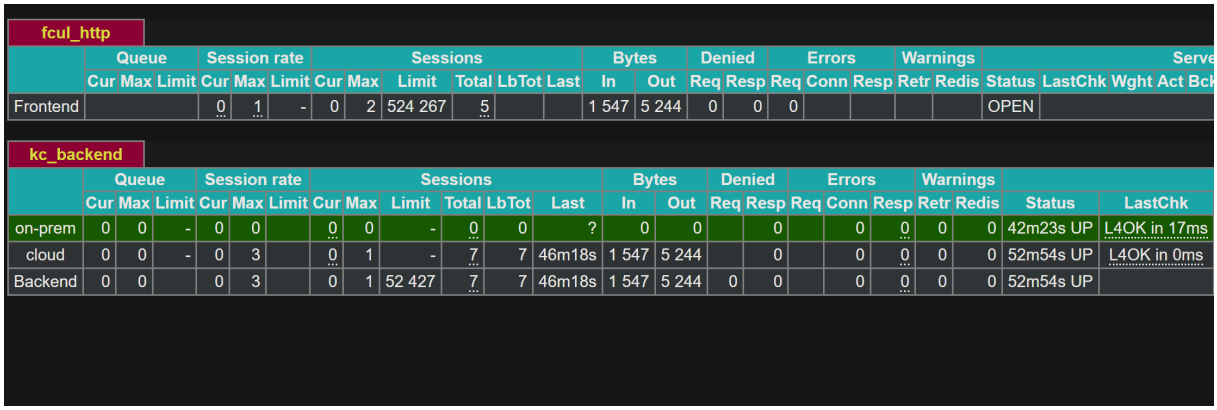
```
# Exemplo (parar o KC local - Docker)
docker stop kc_local
```

1. Executar login via HAProxy IdP.

Expectativa: HAProxy remove nó `kc-local` e redireciona para `kc-cloud`.
Resultado Observado:

- **H1:** O serviço de autenticação manteve-se disponível e funcional. A base de dados partilhada e orquestrada com Patroni garantiu consistência de dados e sessões. Tanto os serviços que utilizam o Keycloak quanto as suas instâncias continuaram com as mesmas sessões, não tendo de voltar a fazer login para voltar a aceder qualquer um dos dois (serviços ou keycloak), isto garantiu uma

melhor experiência de utilizador, tanto para a admin di (acesso ao painel do keycloak), quanto aos utilizadores dos serviços que dependem da autenticação via Keycloak.



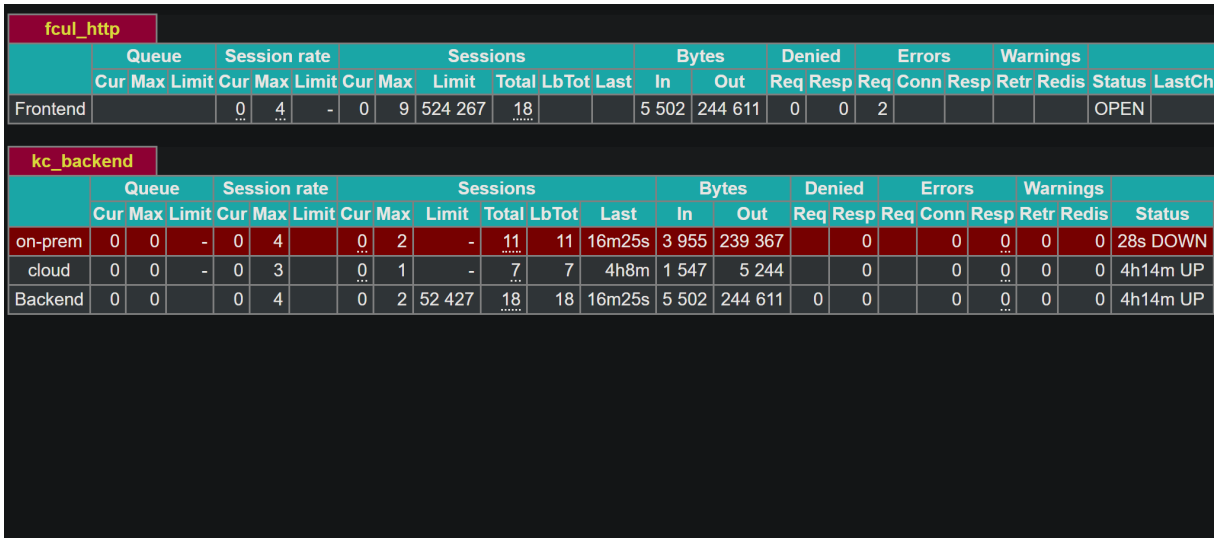
fcul_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Serve			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck
Frontend				0	1	-	0	2	524 267	5			1 547	5 244	0	0	0					OPEN				

kc_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk
on-prem	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	42m23s UP	L4OK in 17ms
cloud	0	0	-	0	3		0	1	-	7	7	46m18s	1 547	5 244		0		0	0	0	0	52m54s UP	L4OK in 0ms
Backend	0	0		0	3		0	1	52 427	7	7	46m18s	1 547	5 244	0	0		0	0	0	0	52m54s UP	

Figura 4.6: O Keycloak primário está disponível.



fcul_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastCh
Frontend				0	4	-	0	9	524 267	18			5 502	244 611	0	0	2					OPEN	

kc_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
on-prem	0	0	-	0	4		0	2	-	11	11	16m25s	3 955	239 367		0		0	0	0	0	28s DOWN
cloud	0	0	-	0	3		0	1	-	7	7	4h8m	1 547	5 244		0		0	0	0	0	4h14m UP
Backend	0	0		0	4		0	2	52 427	18	18	16m25s	5 502	244 611	0	0		0	0	0	0	4h14m UP

Figura 4.7: Keycloak primário foi desligado e secundário assume.

- **H2:**O serviço manteve-se funcional. Como ambas instâncias do Keycloak federam o mesmo AD, o serviço de autenticação não sofreu interrupções, mas as sessões ativas foram perdidas, pois, as bases de dados são diferentes, o que forçou os utilizadores a terem de reiniciar a sessão com novo login.

4.1.4.4 C4. Sincronização de dados de utilizador

Objetivo: verificar propagação de novos utilizadores/alterações.
Procedimento:

1. Criar utilizador de teste no AD local (grupo: Alunos).
2. Aguardar o ciclo de sincronização no(s) Keycloak(s).
3. Autenticar e inspecionar `id_token` / `access_token` (claims/grupos).

Resultado Observado:

- **H1:** Utilizador disponível em ambas as instâncias (LDAP + BD comum). Bloqueios/alterações refletem-se de imediato.
- **H2:**O utilizador ficou disponível em ambas as instâncias (federação LDAP). Customizações feitas apenas em uma instancia do keycloak não propagaram para outra (Ex: Mappers, tempos de sincronização, etc).

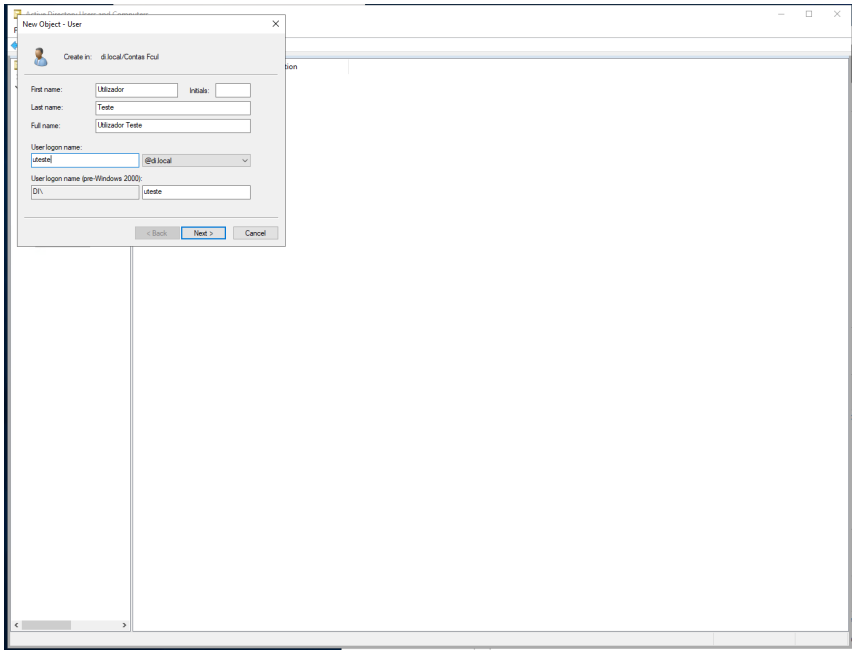


Figura 4.8: Criação de utilizador de teste

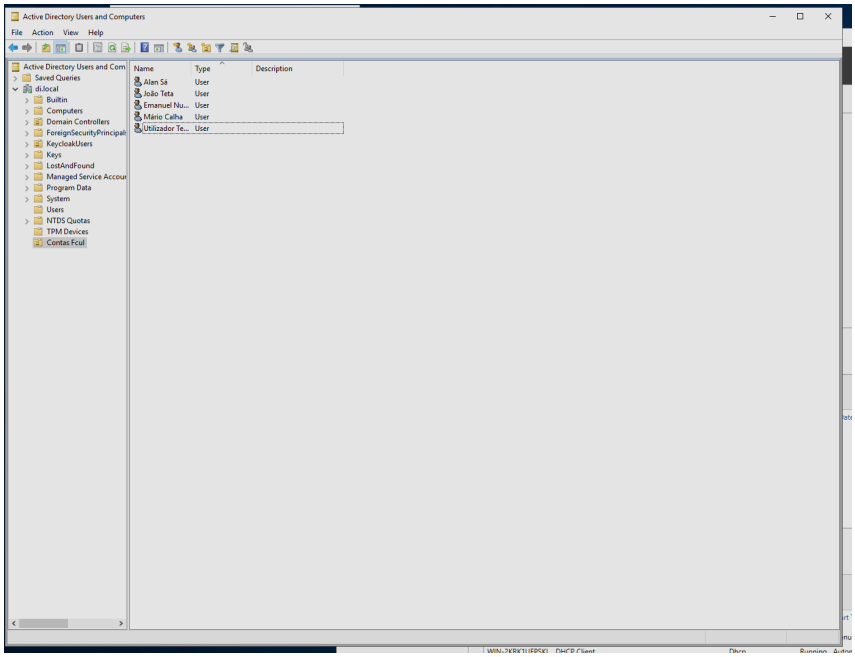


Figura 4.9: Resultado da criação de utilizador de teste

4. RESULTADOS E DISCUSSÃO

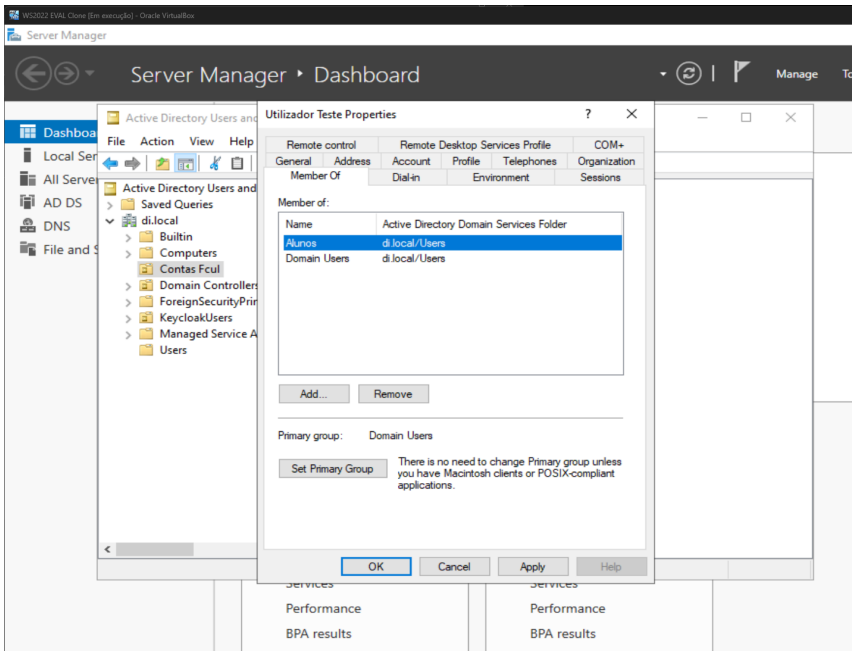


Figura 4.10: Atribuição de role Aluno

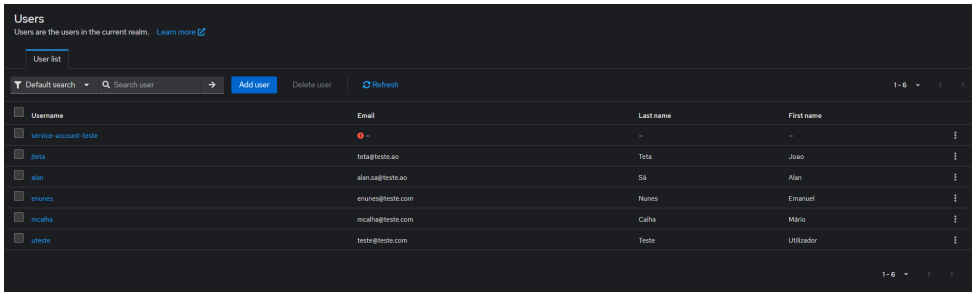


Figura 4.11: Resultado no Keycloak

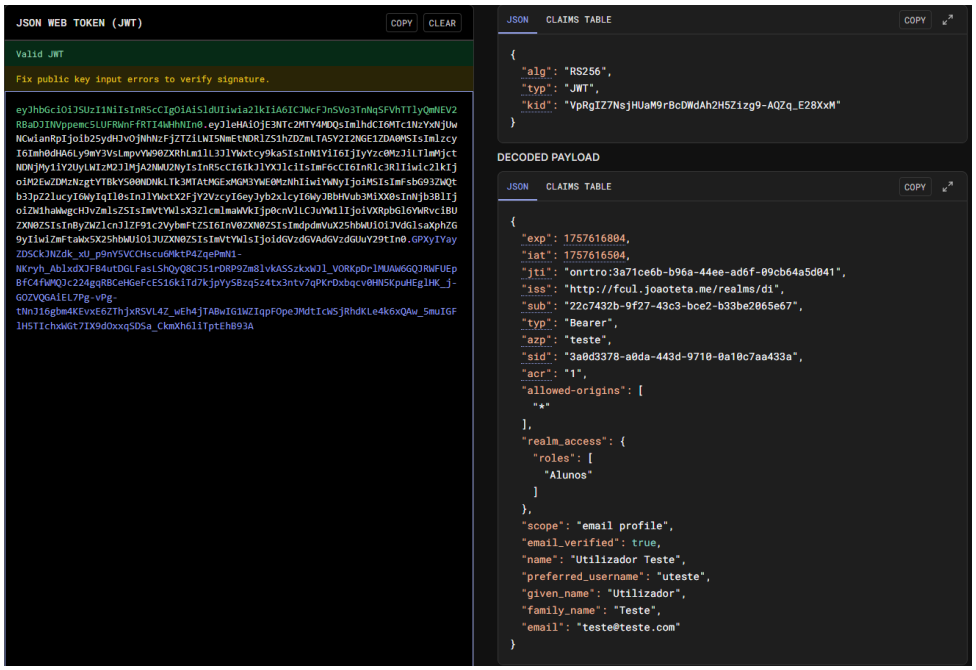


Figura 4.12: Estrutura do token JWT obtido no login

4.1.4.5 C5. Desempenho sob carga concorrente

Objetivo: Avaliar o comportamento do sistema em termos de tempos de resposta e disponibilidade sob carga concorrente de logins, considerando cinco cenários distintos: (i) cenário ótimo, com todas as instâncias disponíveis; (ii) indisponibilidade da instância primária do Keycloak; (iii) falha de um Domain Controller local; (iv) indisponibilidade da base de dados sem alternativa configurada; (v) indisponibilidade total da infraestrutura local;

Nota: Optou-se por apresentar boxplots apenas para a métrica de latência, por esta evidenciar maior variabilidade entre execuções. Nos restantes indicadores — taxa de erro e taxa de processamento — os resultados mostraram-se suficientemente estáveis, não se justificando a representação gráfica da sua distribuição.

Procedimento (K6):

- Execução com 50 *Virtual Users* (VUs) em simultâneo, durante um período contínuo de 5 minutos, sem ramp-up, refletindo um cenário de carga concorrente estável;
- A escolha de 50 VUs justifica-se pelo número atual de utilizadores do sistema, estimado em cerca de 200, representando assim uma fração significativa da carga real esperada, sem deixar de manter a experiência de teste controlada e comparável entre cenários;
- Simulação integral do fluxo de autenticação OIDC (`authorize` → `token` → `userinfo` → `refresh`), abrangendo os principais pontos de interação com o sistema de identidade;
- Ponto de entrada definido no HAProxy IdP, assegurando a distribuição de carga e o encaminhamento para as instâncias subjacentes.

4.1.4.6 C5.1 Cenário ótimo

Resultados observados:

- **H1:** O tempo médio de resposta registado foi de aproximadamente 0.48 s (Fig. 4.14), com uma taxa de erros residual de 0.004 % (Fig. 4.15). A taxa de processamento sustentou-se em cerca de 90 requisições/segundo (Fig. 4.16).
- **H2:** O tempo médio de resposta situou-se em torno de 0.7 s (Fig. 4.14), acompanhado de uma taxa de erros próxima de 0.016 % (Fig. 4.15). A capacidade de processamento foi de aproximadamente 60 requisições/segundo (Fig. 4.16).
- **Dispersão das latências:** O boxplot da Figura 4.13 complementa a análise da latência média, evidenciando a *variabilidade dos tempos de resposta* e a *presença de outliers* em ambas as hipóteses. Observa-se que, embora a mediana de H2 seja superior à de H1, a dispersão dos valores é igualmente mais ampla, refletindo uma menor previsibilidade no desempenho. Em contrapartida, H1 apresenta uma distribuição mais concentrada, o que sugere maior estabilidade temporal nas requisições, em consonância com a sua latência média inferior. Importa referir que os testes representados na Figura 4.13 foram realizados num instante distinto dos restantes cenários, reforçando a consistência do padrão de desempenho observado e demonstrando que as diferenças entre H1 e H2 se mantêm independentemente de variações de latência externas ao sistema.

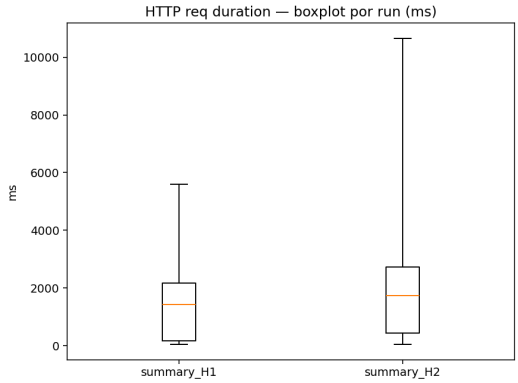


Figura 4.13: Dispersão da latência C5.2

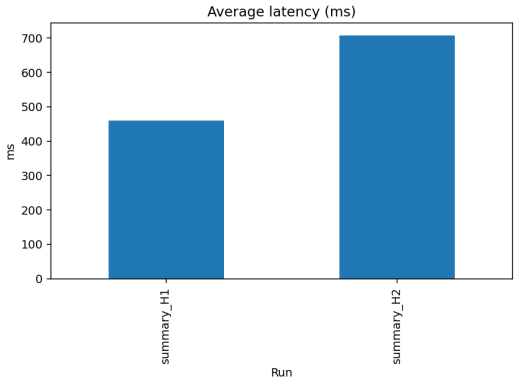


Figura 4.14: Latência média C5.2

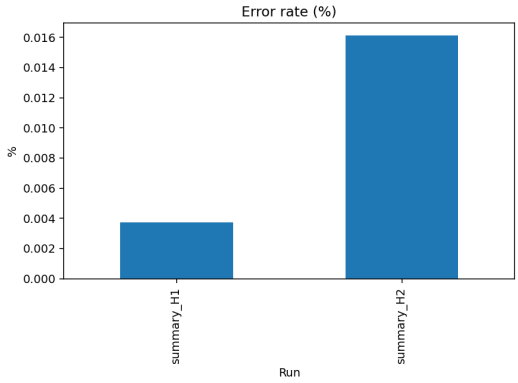


Figura 4.15: Taxa de erros C5.2

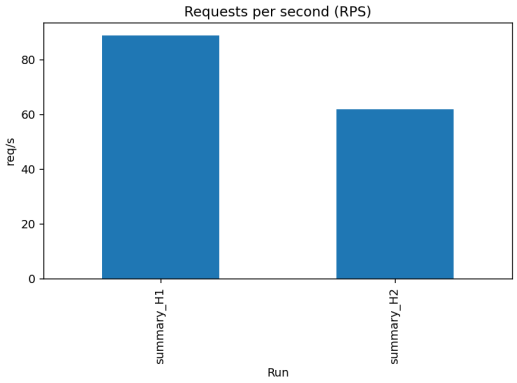


Figura 4.16: Pedidos por segundo C5.2

4.1.4.7 C5.2 Cenário com falha de uma instância do Keycloak (primária)

Resultados observados:

- **H1:** O tempo médio de resposta situou-se em aproximadamente 2.9 s (Fig. 4.20), com uma taxa de erros em torno de 1.5 % (Fig. 4.21). A taxa de processamento registada foi de cerca de 16 requisições/segundo (Fig. 4.22).
- **H2:** O tempo médio de resposta foi de aproximadamente 1.2 s (Fig. 4.20), acompanhado de uma taxa de erros próxima de 0.8 % (Fig. 4.21). A taxa de processamento atingiu cerca de 40 requisições/segundo (Fig. 4.22).
- **Dispersão das latências:** O boxplot da Figura 4.19 reforça a consistência entre as métricas observadas, evidenciando uma *dispersão significativamente superior em H1* face a H2. A maior variabilidade dos tempos de resposta em H1, associada à sua latência média mais elevada, indica menor estabilidade e maior suscetibilidade a picos de demora. Por contraste, H2 apresenta uma *distribuição mais concentrada e previsível*, com menores desvios e outliers, confirmando o seu melhor desempenho global. Estes resultados demonstram um padrão consistente entre as métricas de latência média e dispersão, reforçando a fiabilidade das observações registadas.

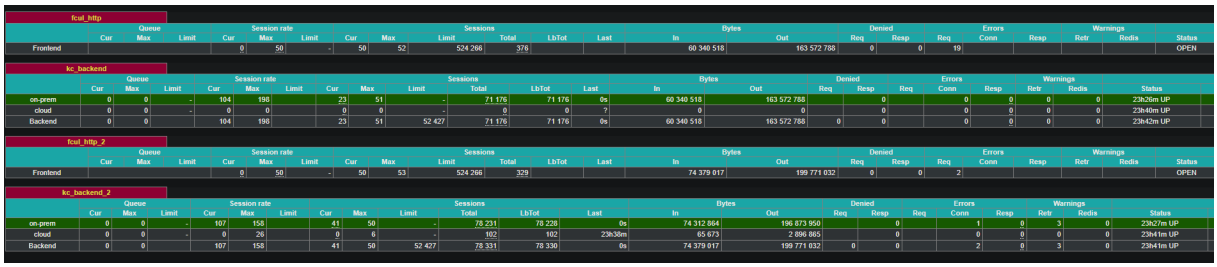


Figura 4.17: Instância Primária Ativa C5.2

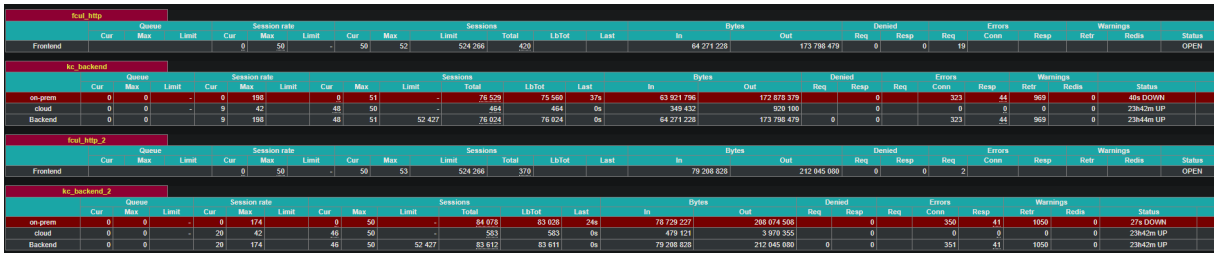


Figura 4.18: Instância Primária Inativa C5.2

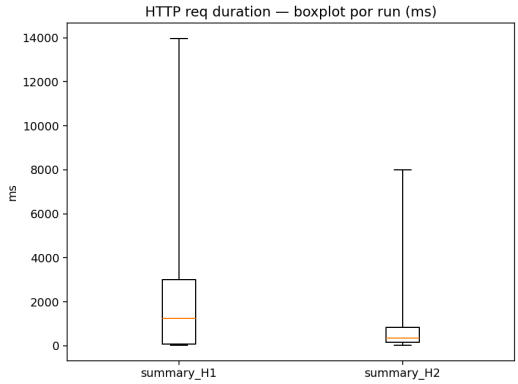


Figura 4.19: Dispersão da latência C5.2

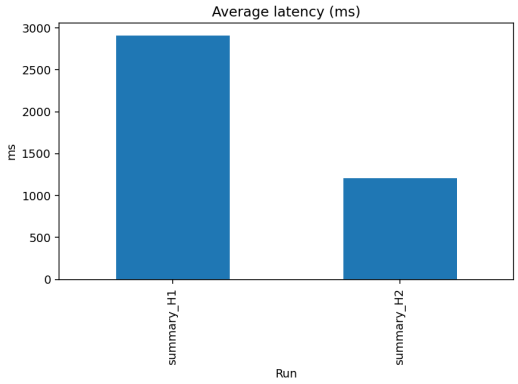


Figura 4.20: Latência média C5.2

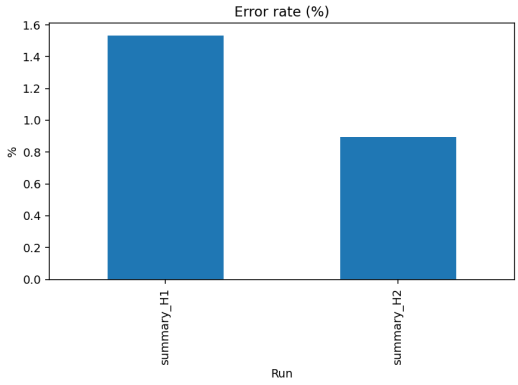


Figura 4.21: Taxa de erros C5.2

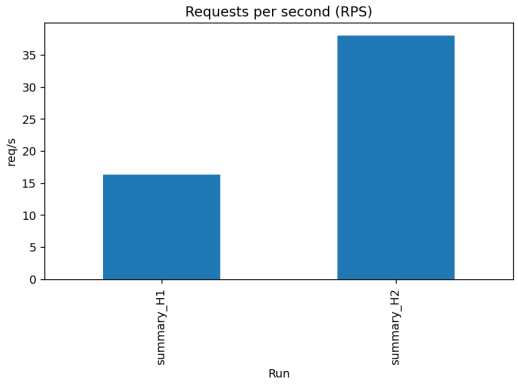


Figura 4.22: Pedidos por segundo C5.2

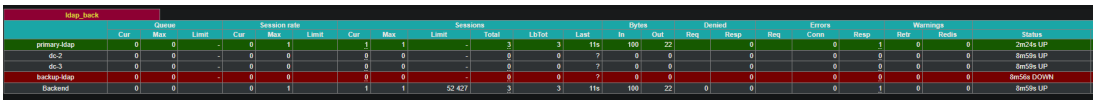
4.1.4.8 C5.3 Cenário com falha de um Domain Controller

Resultados observados:

- **H1:** O tempo médio de resposta foi de aproximadamente 0.58 s (Fig. 4.26), registando-se uma

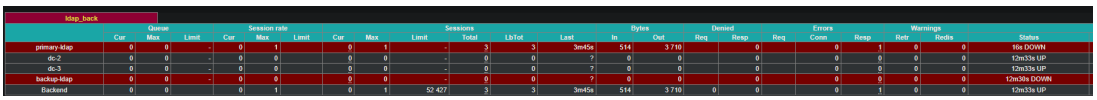
taxa de erros próxima de 0.22 % (Fig. 4.27). O sistema manteve uma taxa de processamento de cerca de 75 requisições/segundo (Fig. 4.28).

- **H2:** O tempo médio de resposta aumentou para aproximadamente 0.82 s (Fig. 4.26), acompanhado de uma taxa de erros em torno de 0.53 % (Fig. 4.27). A capacidade de processamento reduziu-se para cerca de 50 requisições/segundo (Fig. 4.28).
- **Dispersão das latências:** O boxplot da Figura 4.25 demonstra um padrão consistente com as métricas de latência média, evidenciando maior dispersão dos tempos de resposta em H2. Embora ambas as hipóteses apresentem amplitudes elevadas, a mediana superior e a maior variabilidade de H2 sugerem menor previsibilidade e estabilidade do desempenho. Por contraste, H1 revela uma distribuição ligeiramente mais concentrada, condizente com a sua menor latência média e taxa de erros inferior. Estes resultados reforçam a relação direta entre latência média, dispersão e taxa de sucesso das requisições.



ldap_back	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
primary-ldap	0	0	-	0	1		1	1	-	3	3	11s	100	22		0		0	1	0	0	2m24s UP
dc-2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	8m59s UP
dc-3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	8m59s UP
backup-ldap	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	8m56s DOWN
Backend	0	0		0	1		1	1	52 427	3	3	11s	100	22	0	0		0	1	0	0	8m59s UP

Figura 4.23: LDAP Primário Ativo C5.3



ldap_back	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
primary-ldap	0	0	-	0	1		0	1	-	3	3	3m45s	514	3 710		0		0	1	0	0	16s DOWN
dc-2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	12m33s UP
dc-3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	12m33s UP
backup-ldap	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	12m30s DOWN
Backend	0	0		0	1		0	1	52 427	3	3	3m45s	514	3 710	0	0		0	1	0	0	12m33s UP

Figura 4.24: LDAP Primário inativo C5.3

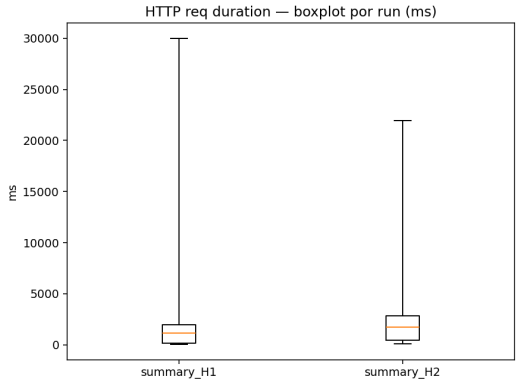


Figura 4.25: Dispersão da Latência C5.3

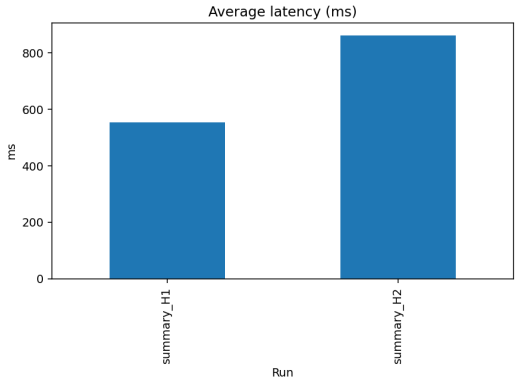


Figura 4.26: Latência média C5.3

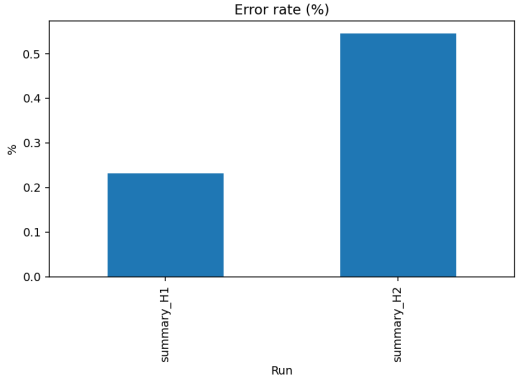


Figura 4.27: Taxa de erros C5.3

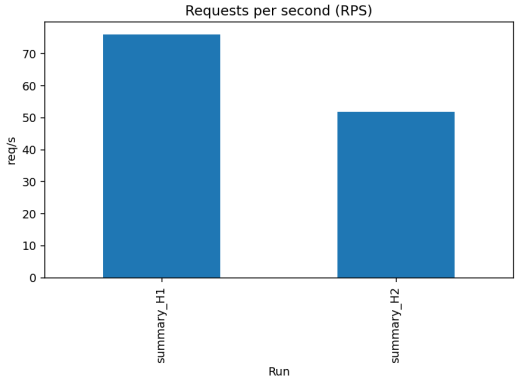


Figura 4.28: Pedidos por segundo C5.3

4.1.4.9 C5.4 Cenário com falha da Base de Dados

Resultados observados:

- **H1:** O serviço respondeu como esperado, sem sobressaltos, mantendo a taxa de erros em 0.42 %. As latências observadas na geração e no refresh de tokens são compreensíveis, uma vez que em cada chamada é necessário contactar o Active Directory para autenticar o utilizador. Contudo, este cenário não corresponde totalmente à realidade: após a emissão, o token mantém-se válido durante um determinado período antes de expirar e, só então, precisa de ser renovado. Na prática, isto reduz de forma significativa o tempo de espera sentido pelos utilizadores.

  Podemos ver em detalhe os resultados nas figuras 4.29 e 4.30.

- **H2:** Não aplicável à hipótese em análise. A indisponibilidade da base de dados, sem a existência de uma alternativa, torna todo o sistema local indisponível, impossibilitando o HAProxy de detetar a falha e efetuar a comutação necessária.

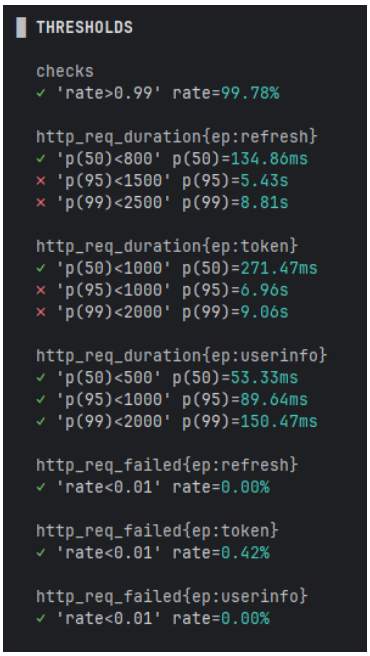


Figura 4.29: Resultados gerais C5.4

Figura 4.30: Observações de comutação no cluster (Patroni) C5.4

4.1.4.10 C5.5 Cenário com falha total da infraestrutura local

Resultados observados:

- **H1:** O serviço manteve-se funcional, sem sobressaltos, mantendo a taxa de erros em $\approx 0.44\,\%$ (Fig. 4.33). A latência foi de $\approx 0.68\,\text{s}$ (Fig. 4.32) e um total $\approx 70\,\text{requisições/segundo}$ (Fig. 4.34).
- **H2:** O serviço manteve-se funcional, sem sobressaltos, mantendo a taxa de erros em $\approx 0.48\,\%$ (Fig. 4.33), não muito diferente de H1. A latência foi de $\approx 1.1\,\text{s}$ (Fig. 4.32) e um total $\approx 40\,\text{requisições/segundo}$ (Fig. 4.34), o que é ligeiramente pior que H1 (mais latência para menos pedidos).
- **Dispersão das latências:** O boxplot da Figura 4.31 reforça esta análise, evidenciando um comportamento semelhante entre as duas hipóteses, com amplitudes elevadas mas consistentes. A distribuição de H1 apresenta-se ligeiramente mais concentrada, refletindo uma maior previsibilidade e estabilidade temporal. Já H2, embora com uma mediana próxima, mostra maior variabilidade nos tempos de resposta, o que está em consonância com a sua latência média superior e menor taxa de processamento.

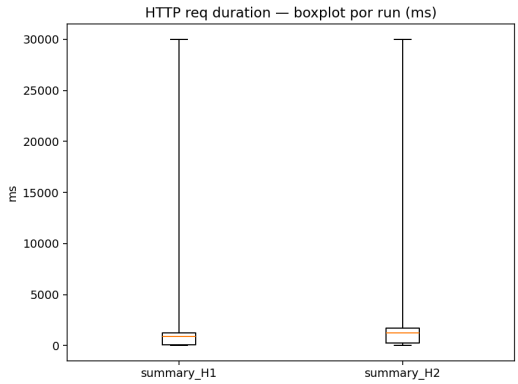


Figura 4.31: Dispersão da latência C5.5

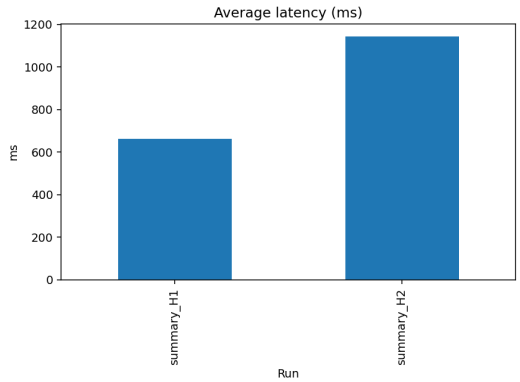


Figura 4.32: Latência média C5.5

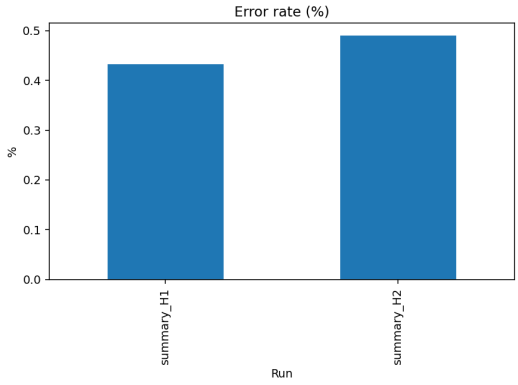


Figura 4.33: Taxa de erros C5.5

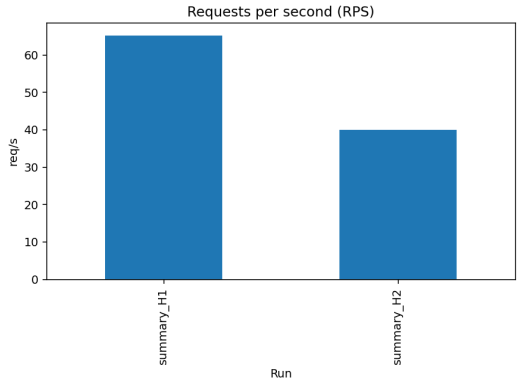


Figura 4.34: Pedidos por segundo C5.5

4.2 Análise dos Resultados

Esta secção discute os resultados obtidos nos testes funcionais (C1–C4) e de desempenho sob carga (C5), comparando as duas hipóteses arquiteturais: H1 (Keycloaks a partilhar uma BD PostgreSQL gerida por *Patroni*) e H2 (Keycloaks independentes, cada um com a sua BD, a federar o mesmo AD).

Importa referir que as análises de dispersão da latência (boxplots) foram realizadas em instantes distintos dos restantes testes de desempenho. Assim, os valores absolutos apresentados nesses gráficos não

devem ser interpretados de forma direta face às métricas médias, mas sim enquanto indicadores complementares de comportamento — evidenciando padrões de variabilidade, consistência e estabilidade temporal observados em cada hipótese. Esta abordagem permite reforçar a validade comparativa dos resultados, ao destacar a coerência entre latência média e dispersão nas diferentes execuções.

4.2.1 Síntese por tipo de teste

C1 – Autenticação básica e SSO. Ambas as hipóteses cumpriram os requisitos de autenticação OIDC e SSO. Em H1, a *consistência* de dados (sessões, atributos, realms e *mappers*) é imediata entre instâncias por via da BD partilhada, proporcionando experiência uniforme de administração e de utilizador. Em H2, a autenticação e o SSO são funcionais (ponto de entrada único via HAProxy IdP), mas *customizações locais* feitas num Keycloak (por exemplo, mapeamentos e tempos de sincronização) não se refletem na outra instância, exigindo disciplina operacional para não divergir configuração.

C2 – Falha do AD local. Nos dois cenários (H1/H2), o HAProxy LDAP encaminhou pedidos para um DC alternativo, preservando a disponibilidade de login. Observou-se apenas aumento moderado da latência, compatível com a distância de rede até ao DC de contingência. A capacidade de isolamento da falha ao nível LDAP confirmou a adequação da *camada de proxy* para reduzir impacto no IdP.

C3 – Falha de uma instância Keycloak (on-prem). Em H1, o *failover* ao KC da nuvem manteve sessões e estado de administração graças à BD partilhada; os utilizadores permaneceram autenticados, e o acesso ao painel de administração não exigiu novo login. Em H2, embora o serviço se tenha mantido funcional (o HAProxy redirecionou para a outra instância), *sessões ativas* foram perdidas por não existir estado partilhado entre BDs, obrigando re-autenticação. Conclui-se que H1 oferece *continuidade de sessão* superior em falhas de nó do IdP.

C4 – Sincronização de dados de utilizador. Ambas as hipóteses apresentaram propagação correta de utilizadores do AD. Em H1, além do LDAP, a BD comum propagou imediatamente alterações administrativas (p. ex., atribuição de *roles*, parametrização de *mappers*). Em H2, a sincronização de utilizadores via federação LDAP foi consistente, mas customizações específicas do Keycloak requerem duplicação manual para evitar *config drift* entre instâncias.

C5 – Desempenho sob carga concorrente (50 VUs, 5 min). No cenário ótimo (C5.1), H1 evidenciou latência média inferior ($\sim 0{,}48\,\text{s}$ vs $\sim 0{,}7\,\text{s}$ em H2) e maior débito (~ 90 RPS vs ~ 60 RPS), com taxa de erros residual. A vantagem é coerente com a *orquestração nativa* do Postgres por *Patroni* (tuning de parâmetros, reciclagem de ligações e gestão de saúde), reduzindo a variabilidade de I/O e *timeouts* no acesso à BD.

Com **falha da instância primária do KC (C5.2)**, H2 teve melhor latência média ($\sim 1{,}2\,\text{s}$ vs $\sim 2{,}9\,\text{s}$ em H1) e maior RPS, sugerindo que a redistribuição de carga da aplicação (sem partilha de estado) foi mais ligeira do lado servidor. Em H1, o aumento de latência é consistente com overhead momentâneo de reconexões à Base de Dados e reequilíbrio de ligações do *pool* após a perda do primário do IdP.

Na **falha de um Domain Controller (C5.3)**, H1 manteve vantagem de desempenho ($\sim 0{,}58\,\text{s}$, ~ 75 RPS) face a H2 ($\sim 0{,}82\,\text{s}$, ~ 50 RPS). Como o fluxo de autenticação consulta o AD em pontos sensíveis (emissão e *refresh* do token), melhorias de latência na camada de BD (Patroni) e estabilidade geral do IdP amortizam o impacto da comutação LDAP.

Na **indisponibilidade da BD (C5.4)**, apenas H1 manteve o serviço, confirmando a resiliência provida por *Patroni* (endereço virtual e *failover* controlado). Em H2, a falta de alternativa de BD torna o serviço local indisponível e inviabiliza a comutação automatizada só com o HAProxy IdP.

Por fim, na **falha total da infraestrutura local (C5.5)**, ambas as hipóteses funcionaram via nuvem, mas H1 preservou melhor equilíbrio entre latência ($\sim 0{,}68\,\text{s}$) e RPS (~ 70), enquanto H2 ficou mais degradada ($\sim 1{,}1\,\text{s}$, ~ 40 RPS), coerente com maior variabilidade no acesso a uma BD isolada e sem orquestração.

4.2.2 Interpretação e implicações

Os resultados corroboram que:

1. H1 oferece *melhor desempenho em regime estável* e *continuidade de sessão* superior em falhas do IdP, ao custo de maior complexidade operacional (cluster de BD, *failover* coordenado).

2. H2 simplifica a camada de dados por instância, mas transfere a responsabilidade de *consistência de configuração* para processos operacionais e é mais vulnerável a falhas de BD, além de perder sessões ativas quando há comutação entre nós.

3. A utilização do HAProxy (IdP e LDAP) provou-se eficaz para mascarar falhas de componentes e reduzir MTTR percebido, mas *não substitui* a orquestração do estado persistente.

Em síntese, se a prioridade é *experiência de utilizador consistente* (SSO sem reautenticação em falhas), *menor latência média* e *resiliência de dados*, H1 é preferível. Se a prioridade for *simplicidade* e *isolamento por instância*, aceitando reautenticação em falhas e maior disciplina de configuração, H2 pode ser considerada.

4.3 Limitações

Apesar do cuidado metodológico, há limitações que condicionam a generalização:

- **Escala e padrão de carga.** Os testes usaram 50 VUs por 5 minutos, sem *ramp-up*. Embora representativos, cargas maiores, perfis *burst*, ou *open-model* (taxa fixa de chegada) podem produzir comportamentos distintos (p. ex., saturação de *pool* de ligações).

- **Horizonte temporal curto.** Não se observaram efeitos de longo prazo (fragmentação de memória, crescimento de *WAL*, *autovacuum*, rotação de logs), nem janelas de manutenção (p. ex., *checkpoint* pesado).

- **Dependência de rede.** A latência entre locais (VPN) e entre IdP/AD/BD é específica do laboratório. Em produção, jitter e perdas podem alterar os *percentis* de latência.

- **Escopo funcional restrito.** O foco foi o fluxo OIDC (authorize/token/userinfo/refresh). Outras operações (administração massiva, sincronizações prolongadas de LDAP, rotação de chaves/realms, *backchannel logout*) não foram medidas.

- **Medições e export do k6.** Algumas execuções não incluíram todas as estatísticas (p. ex., p99, RPS por endpoint). Ainda que não afete as conclusões, limita a granularidade de comparação. Em produção, recomenda-se padronizar o `K6_SUMMARY_TREND_STATS` e métricas por endpoint.

- **Config & tuning específicos.** O desempenho observado em H1 reflete não só a presença do *Patroni*, mas também o *tuning* de Postgres/Keycloak/HAProxy adotado. Diferentes tamanhos de *pool*, parâmetros de WAL/checkpoints e hardware/volumes podem enviesar resultados.

4.3.1 Ameaças à validade e testes futuros

- **Validade interna.** Pequenas variações de rede ou contenda de recursos entre execuções podem ter influenciado percentis de latência. Mitigar com repetições adicionais e *confidence intervals*.
- **Validade externa.** A extrapolação para ambientes com ordens de grandeza superiores de tráfego requer testes com *stages* realistas (picos, *soak tests* de horas/dias) e perfis de utilizador heterogéneos.
- **Testes futuros.** (i) testes com *open-model* (arrivals/s), (ii) instrumentação de RPS por endpoint diretamente no k6 (counters com tags), (iii) avaliação de *sticky sessions* vs *stateless tokens*, (iv) impacto de *token lifetimes* e *cache* de chave pública, (v) comparação de *Patroni* com outros gerentes de HA, e (vi) *cost/performance* das hipóteses.

Tabela 4.1: Resumo qualitativo por cenário (H1 vs H2)

Cenário	Latência	RPS	Continuidade de sessão
C5.1 Ótimo	H1 melhor	H1 melhor	H1 melhor
C5.2 Falha KC primário	H2 melhor	H2 melhor	H1 melhor
C5.3 Falha de DC	H1 melhor	H1 melhor	empate
C5.4 Falha da BD	H1 opera	H1 opera	H1 opera
C5.5 Falha total on-prem	H1 melhor	H1 melhor	H1 melhor

4.3.2 Decisão arquitetural (1.ª fase) e evolução

Apesar de a H1 evidenciar, no geral, melhor desempenho e maior resiliência, a opção para a 1.ª fase do projeto recai sobre a H2. Esta decisão fundamenta-se em três eixos:

1. **Aderência aos requisitos atuais.** A H2 satisfaz plenamente os requisitos funcionais e não funcionais definidos para esta fase (SSO, disponibilidade com HAProxy, integração com AD via federação LDAP), com níveis de latência e RPS considerados adequados para a carga prevista.
2. **Complexidade de implementação e operação.** A H1 implica a introdução e operação de um *cluster* Patroni para a base de dados, com *failover* orquestrado e parâmetros de tuning específicos. Tal acrescenta complexidade operacional (observabilidade, *runbooks*, *backups*/*pitR*, janelas de manutenção) que não é estritamente necessária na fase inicial.
3. **Custos e constrangimentos de infraestrutura.** A adoção de H1 requer, tipicamente, um *terceiro nó* (quórum / árbitro) e recursos adicionais (armazenamento, rede e computação), com potenciais impactos de custo não contemplados nos requisitos iniciais.

Deste modo, a H2 é a escolha pragmática para **início de produção**, minimizando tempo de implementação e risco operacional, mantendo margem para evolução.

Roteiro de evolução para H1. Está previsto que a solução *evolua* no sentido de H1, acompanhando a maturidade operacional e eventuais aumentos de carga/criticidade. Os passos propostos são:

- Introdução faseada do **cluster Patroni** (ambiente de ensaio → *canary* → produção), com observabilidade (métricas, *logs*, *alerts*) e *runbooks* de *failover*.
- **Tuning** controlado de PostgreSQL (WAL, *checkpoints*, *autovacuum*) e *pooling* de ligações (Keycloak) com testes de *soak* e *stages* realistas.
- Revisão de **custo/benefício** e **SLA/SLO**: quando a criticidade da continuidade de sessão e a escala de tráfego justificarem, migrar progressivamente para BD partilhada (H1).

Em síntese, a adoção de H2 nesta fase equilibra simplicidade e cumprimento de requisitos, enquanto se reconhece que a H1 constitui a trajetória natural de evolução para ganhos adicionais de desempenho e resiliência quando os pressupostos de negócio e operação assim o exigirem. A tabela **??** sintetiza esta comparação.

Tabela 4.2: Resumo comparativo dos resultados (H1 vs H2)

Cenário	**H1 – BD Partilhada (Patroni)**	**H2 – KCs Independentes**
C1: Autenticação & SSO	Experiência uniforme: sessões, atributos e configurações consistentes entre instâncias graças à BD comum.	SSO funcional via HAProxy; autenticação garantida, mas atributos e *mappers* locais podem divergir.
C2: Falha de AD local	HAProxy redireciona automaticamente para DC remoto; login mantido com ligeiro aumento de latência.	Mesmo comportamento: login preservado via DC remoto, dependência direta do LDAP.
C3: Queda de uma instância KC	HAProxy redireciona para instância ativa; sessões mantêm-se devido à BD partilhada (sem reautenticação).	HAProxy redireciona para instância ativa; utilizadores têm de reautenticar (sessões não partilhadas).
C4: Sincronização de utilizadores	Propagação imediata de alterações (LDAP + BD comum); roles e mapeamentos replicados.	Novos utilizadores disponíveis via LDAP; customizações locais não se propagam entre instâncias.
C5: Desempenho sob carga (50 VUs)	Melhor desempenho global: menor latência média, maior débito (RPS) e taxa de erros residual.	Desempenho inferior mas dentro dos requisitos; latências e erros compatíveis com a comunicação federada.

Capítulo 5

Conclusão e trabalhos futuros

5.1 Conclusão

Nesta dissertação, foi projetada e implementada uma solução de autenticação federada híbrida integrando o Keycloak e o Active Directory de uma instituição académica, com o objetivo de modernizar a gestão de identidades e garantir a continuidade do serviço de autenticação. Com base na análise de requisitos e no estudo de trabalhos relacionados, foram consideradas três arquiteturas alternativas de integração AD–Keycloak. A opção selecionada combinou o melhor de dois mundos: a inclusão de um controlador de domínio do AD na nuvem (garantindo redundância de diretório) e o uso de um IdP Keycloak central na nuvem para SSO (modernização), acrescido de componentes de alta disponibilidade (HAProxy, instâncias redundantes) para eliminar pontos únicos de falha.

A solução final implementada atinge os objetivos propostos. Em ambiente de teste, demonstrou-se que mesmo em casos de falha completa da infraestrutura local, os utilizadores conseguem autenticar-se através do IdP na cloud, graças à replicação dos dados de autenticação (AD) e à resiliência da configuração federada. Inversamente, falhas na cloud não paralisam a autenticação local, já que uma instância Keycloak de backup e o AD on-premises podem sustentar temporariamente o serviço. Este modelo híbrido garantiu, assim, uma experiência ininterrupta de autenticação para os utilizadores e elevou a disponibilidade do serviço a um patamar superior.

Para além da continuidade, a solução proporcionou benefícios adicionais: introduziu Single Sign-On entre aplicações díspares, reduzindo a necessidade de múltiplos logins; melhorou a segurança com possibilidade de MFA centralizado e políticas consistentes via Keycloak; e preparou o terreno para futuras expansões – tanto a nível de integração de novos serviços cloud, como de migração paulatina de legados para padrões modernos. Tudo isto foi alcançado mantendo a compatibilidade com o ecossistema existente (o AD continua a ser a fonte primária de identidades), o que valida a abordagem de migração gradual defendida.

O contributo central desta dissertação não se esgotou na validação da integração entre Keycloak e AD. Foi conduzida uma comparação experimental rigorosa entre duas hipóteses arquiteturais contrastantes:

H1, com Keycloaks a partilhar uma base de dados PostgreSQL gerida por *Patroni*;

H2, com Keycloaks independentes, cada um com a sua base de dados, federados ao mesmo AD.

Através de cenários realistas (C1–C5) e testes de carga com *k6*, evidenciou-se que H1 garante melhor desempenho médio e continuidade de sessão em falhas do IdP, mas exige maior complexidade operacional e custos acrescidos (ex.: cluster de três nós de BD). Já H2, embora menos eficiente, revelou-se suficiente para os requisitos atuais, oferecendo simplicidade de gestão à custa de maior vulnerabilidade a falhas de

base de dados e perda de sessões em comutação.

No contexto académico do DI/Ciencias - ULisboa, esta implementação serve de prova de conceito de como a modernização da infraestrutura de autenticação pode ocorrer sem ruptura: um IdP open-source como Keycloak mostrou-se capaz de conviver e cooperar com o AD corporativo, trazendo novas funcionalidades e robustez a um custo reduzido. Os resultados obtidos reforçam que a federação de identidades é o caminho certo para instituições que buscam combinar ambientes on-premises e cloud de forma segura e eficiente.

Em suma, as principais contribuições deste trabalho incluem: (i) a definição de uma arquitetura híbrida inovadora para IAM no âmbito universitário, unindo AD e Keycloak com alta disponibilidade; (ii) a implementação concreta dessa arquitetura, ultrapassando desafios técnicos de sincronização e *failover*; (iii) a validação experimental das hipóteses H1 e H2, demonstrando os seus respetivos trade-offs em termos de resiliência, consistência e desempenho; (iv) a documentação pormenorizada dos mecanismos envolvidos (LDAP federation, SSO, VPN, HAProxy, Patroni), que pode servir de guia para outras organizações.

Conclui-se, portanto, que embora H1 se revele mais eficiente e resiliente no geral, a opção por H2 nesta primeira fase é a mais adequada, dado que satisfaz plenamente os requisitos atuais com menor complexidade e custos controlados. Reconhece-se, contudo, que a solução poderá evoluir para H1 à medida que as exigências de escala e resiliência aumentem. Esta dissertação deixa, assim, uma base sólida para futuras decisões arquiteturais e contribui para a literatura aplicada sobre sistemas de identidade híbridos em contexto académico real.

5.1.1 Trabalhos Futuros

Várias direções de evolução foram identificadas durante o desenvolvimento desta dissertação. Estas poderão aprofundar os resultados obtidos e alinhar a solução com as práticas modernas de *Identity and Access Management* (IAM):

Implementação em ambiente real de produção

Migrar do protótipo laboratorial para produção no DI/Ciencias - ULisboa, incluindo *rollout* gradual para todos os utilizadores, integração de aplicações legadas (SAML, adaptadores específicos) e monitorização contínua. Este passo permitirá validar a solução em larga escala com utilizadores reais.

Clustering e distribuição geográfica com Kubernetes

Explorar a execução do Keycloak em clusters Kubernetes com *StatefulSets* e Infinispan distribuído, incluindo cenários multi-região. Tal permitiria alta disponibilidade nativa e *failover* entre datacenters, suportado por balanceadores globais (Azure Traffic Manager, Route53).

Este tipo de abordagem encontra em concordância com Singh, 2023, que apresenta uma arquitetura de referência multi-cloud baseada em Kubernetes, Istio e Keycloak, demonstrando a viabilidade de orquestração de instâncias distribuídas com identidade federada em diferentes regiões. Os resultados reforçam que cenários de alta disponibilidade com failover entre datacenters são exequíveis e alinhados com as exigências de resiliência em infraestruturas acadêmicas e empresariais modernas.

Automatização e Infraestrutura como Código (IaC)

Desenvolver *playbooks* Ansible e módulos Terraform para provisionar AD, Keycloak, HAProxy e WireGuard de forma reproduzível. Isto reduziria erros manuais, facilitaria *disaster recovery* e serviria de documentação formal da topologia.

Monitorização, métricas e testes de escala

Integrar Prometheus/Grafana para recolha contínua de métricas (latência, sessões, throughput da VPN). Testes de carga com milhares de utilizadores simulados permitiriam antecipar pontos de saturação e otimizar parâmetros (pool de ligações, *timeouts*, limites de memória).

Integração com tendências modernas de segurança

- *Zero Trust* e MFA obrigatório: políticas de acesso baseadas em contexto (localização, dispositivo, risco).
- *Passwordless* e WebAuthn: adoção de credenciais sem password (FIDO2) como evolução natural.
- UEBA e deteção de anomalias: integração com SIEM ou módulos de *machine learning* para identificar padrões suspeitos de login e aplicar políticas dinâmicas (ex.: MFA adicional).

Integração com ecossistemas externos de identidade

Avaliar a interoperabilidade com Azure Entra ID, Google Identity ou IdPs académicos (ex.: eduGAIN), permitindo cenários de *multi-federation* em que utilizadores externos também se autenticam.

Autorização centralizada (RBAC/ABAC federado)

Expandir o âmbito para a autorização, explorando o módulo de *Authorization Services* do Keycloak para gerir permissões *fine-grained* centralmente, evitando duplicação de regras nas aplicações.

Um passo natural será integrar o sistema de autenticação com os serviços já existentes no Departamento de Informática (DI). Esta evolução abriria caminho para a expansão gradual a toda a Faculdade de Ciências e, eventualmente, à Universidade de Lisboa, permitindo um ecossistema unificado de identidade e autorização. Tal cenário potenciaria não só a simplificação do acesso dos utilizadores, mas também a implementação de políticas transversais de segurança e de gestão de credenciais.

Em conclusão, este trabalho abre diversas possibilidades de melhoria e investigação adicional. À medida que as organizações continuam a adotar arquiteturas híbridas e multi-*cloud*, a importância de sistemas IAM resilientes e integrados só tenderá a crescer. Acreditamos que os conhecimentos adquiridos e os resultados obtidos nesta dissertação possam servir de alicerce para evoluções futuras, contribuindo para infraestruturas de autenticação mais seguras, unificadas e confiáveis nas instituições.

Bibliografia

Abdulazeez, Adnan Mohsin et al. (2020). "Comparison of VPN Protocols at Network Layer Focusing on Wire Guard Protocol". Em: *International Journal of Interactive Mobile Technologies (iJIM)* 14.18, pp. 157–177. DOI: 10.3991/ijim.v14i18.16507.

Aldosary, M. e N. Alqahtani (2021). "A Survey on Federated Identity Management Systems Limitation and Solutions". Em: *International Journal of Network Security and its Applications*. Resumo: Revisão das limitações e soluções em sistemas de gestão de identidade federada. URL: https://www.semanticscholar.org/paper/A-Survey-on-Federated-Identity-Management-Systems-Aldosary-Alqahtani/486236050cbd5ac534f8d7a8f2a31424c9599962.

Alsadeh, A., N. Yatim e Y. Hassouneh (2022). "A Dynamic Federated Identity Management Using OpenID Connect". Em: *MDPI*. Resumo: Modelo dinâmico de gestão de identidade federada com recurso ao OpenID Connect. URL: https://www.mdpi.com/1999-5903/14/11/339.

Divyabharathi, D.N. e N.G. Cholli (2020). "A Review on Identity and Access Management Server (KeyCloak)". Em: *IGI Global*. Resumo: Revisão das funcionalidades e características do Keycloak. URL: https://www.igi-global.com/gateway/article/259351.

Freire, Vasco Chouzal (2021). "Framework para controlo de acessos". Tese de mestrado. Instituo Politécnico do Porto.

Gonçalves, C. et al. (2023). "A federated authentication and authorization approach for IoT farming". Em: *ScienceDirect*. Resumo: Abordagem federada de autenticação e autorização em agricultura inteligente (IoT). URL: https://www.sciencedirect.com/science/article/pii/S2542660523001087.

Indu, I., P.M. Rubesh Anand e V. Bhaskar (2018). "Identity and access management in cloud environment: Mechanisms and challenges". Em: *ScienceDirect*. Resumo: Gestão de identidade na cloud, desafios e soluções. URL: https://www.sciencedirect.com/science/article/pii/S2215098617316750.

Kallela, Jyri (2008). "Federated Identity Management Solutions". Em: *Computer Science, Engineering*. URL: https://www.semanticscholar.org/paper/Federated-Identity-Management-Solutions-Kallela/3aa04883b50ff339a75e2c9402e1e147ad8690a8.

Karasawa, M., J. Hover e S. Misawa (2020). "Federated User Account Management". Em: *EPJ Web of Conferences*. Resumo: Gestão federada de contas de utilizadores no contexto de computação distribuída. URL: https://www.epj-conferences.org/articles/epjconf/pdf/2020/21/epjconf_chep2020_07058.pdf.

Kavuluri, Harsha Vardhan Reddy, Suresh Babu Avula e Adithya Sirimalla (2025). "Performance Tuning for Cloud-Based Databases Oracle/Postgres: Analyzing Query Optimization Techniques". Em: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)* 16.2. Comparação de técnicas de otimização de queries em Oracle Cloud Database e Amazon RDS

PostgreSQL, pp. 459–475. ISSN: 2093-5374. DOI: 10.58346/JOWUA.2025.I2.028. URL: https://doi.org/10.58346/JOWUA.2025.I2.028.

Mihai, Darius et al. (2024). "Security Posture Improvement with High-Availability Keycloak SSO". Em: *2024 23rd RoEduNet Conference: Networking in Education and Research (RoEduNet)*. Caso académico que demonstra a melhoria da postura de segurança via Keycloak com alta disponibilidade. IEEE, pp. 1–7. DOI: 10.1109/RoEduNet64292.2024.10722696. URL: https://doi.org/10.1109/RoEduNet64292.2024.10722696.

OpenVPN Reference Manual (s.d.). https://openvpn.net/community-resources/reference-manual/. Acedido em Setembro 2025.

Prasetijo, Anang, Wahyudi Wibowo e Purnomo Wicaksono (2016). "Performance Comparisons of Web Server Load Balancing Algorithms on HAProxy and Heartbeat". Em: *2016 International Conference on Electrical Engineering and Computer Science (ICECOS)*. Estudo comparativo de algoritmos de balanceamento em HAProxy e Heartbeat, demonstrando impacto em latência e throughput. IEEE, pp. 147–152. DOI: 10.1109/ICECOS.2016.7872891. URL: https://doi.org/10.1109/ICECOS.2016.7872891.

Pritunl Documentation (s.d.). https://docs.pritunl.com/. Acedido em Setembro 2025.

Project, WireGuard (2025). *Quick Start — WireGuard VPN*. Guia oficial para configuração inicial do WireGuard. URL: https://www.wireguard.com/quickstart/.

Rajba, P. et al. (2024). "Identity and Access Management Architecture in the SILVANUS Project". Em: *ACM*. Resumo: Arquitectura de gestão de identidade e acesso no âmbito do projeto europeu SILVANUS. URL: https://dl.acm.org/doi/10.1145/3664476.3670935.

RFC 4301 - Security Architecture for the Internet Protocol (s.d.). https://datatracker.ietf.org/doc/html/rfc4301. Acedido em Setembro 2025.

Singh, Prashant (mar. de 2023). "End-to-End Encryption and Identity Federation for Multi-Cloud FinTech Deployments". Em: *International Journal of Innovative Research in Management, Pharmacy and Sciences (IJIRMPS)* 11.2, pp. 1–9. ISSN: 2349-7300. URL: https://www.ijirmps.org/.

Technologies, HAProxy (2025). *HAProxy Configuration Manual — HAProxy 2.9*. Manual oficial de configuração do HAProxy versão 2.9. URL: https://docs.haproxy.org/2.9/configuration.html.

Thorgersen, S. e P.I. Silva (2021). "Keycloak - Identity and Access Management for Modern Applications, 2nd Edition". Em: *IEEE Xplore*. Resumo: Guia prático sobre implementação do Keycloak em aplicações modernas. URL: https://ieeexplore.ieee.org/document/10163450.

Voicu, Vladimir et al. (2024). "University Identity and Access Management Infrastructure with Keycloak: Lessons Learned". Em: *2024 23rd RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, pp. 20–24. DOI: 10.1109/RoEduNet64292.2024.10722517.

Zalando SE, Patroni Contributors (2025). *Patroni Documentation*. Manual oficial do Patroni. URL: https://patroni.readthedocs.io/en/latest.

Glossário

AD Active Directory – Serviço da Microsoft para gestão de identidades e diretórios.

IdP Identity Provider – Sistema que autentica utilizadores e emite credenciais/tokens para aplicações.

IAM Identity and Access Management – Conjunto de processos e tecnologias para gerir identidades e acessos.

SSO Single Sign-On – Mecanismo que permite autenticação única para acesso a múltiplos serviços.

MFA Multi-Factor Authentication – Autenticação que combina mais de um fator (ex.: password + OTP).

HAProxy High Availability Proxy – Software de balanceamento de carga e alta disponibilidade.

VPN Virtual Private Network – Rede privada construída sobre infraestrutura pública com encriptação.

OIDC OpenID Connect – Protocolo de autenticação baseado em OAuth 2.0.

LDAP Lightweight Directory Access Protocol – Protocolo utilizado para aceder e gerir serviços de diretório, como o Active Directory, permitindo consultas e autenticação de utilizadores.

JWT JSON Web Token – Formato compacto e seguro para representar *claims* (afirmações) entre duas partes, amplamente usado em protocolos de autenticação (ex.: OIDC) para transportar identidades e permissões de forma assinada e, opcionalmente, criptografada.

Apêndice A

Configurações WireGuard

Este anexo documenta a configuração dos três pontos da VPN WireGuard implementados em ambiente de laboratório.

A.1 Topologia e Convenções

A infraestrutura utiliza uma **VPN WireGuard** em topologia *mesh*, ligando três nós de forma *peer-to-peer*. Esta rede suporta a comunicação segura entre o **Keycloak** e o **Active Directory (AD)**, garantindo baixa latência e resiliência.

A.1.1 Topologia

- **Node 1 (local):** aloja uma instância do Keycloak e integra-se com os controladores de domínio do AD local.
- **Node 2 (nuvem):** aloja uma instância redundante do Keycloak e um controlador de domínio do AD local.
- **Node 3 (apoio para witness patroni):** não executa Keycloak, mas participa no cluster de base de dados (Patroni) para assegurar quórum.

A.1.2 Convenções

- Cada nó é identificado como `wg-node1`, `wg-node2` e `wg-node3`.
- A VPN utiliza o espaço de endereçamento `10.0.0.0/24`, com IPs estáticos atribuídos a cada nó.
- Todos os serviços (LDAP/LDAPS para o AD e HTTPS para o Keycloak) circulam exclusivamente através da VPN.

Assim, a topologia em malha e as convenções adotadas asseguram comunicação segura e redundante entre o Keycloak e o AD nos dois ambientes.

A.2 Configuração da VPN WireGuard

A listagem seguinte apresenta um exemplo da configuração utilizada para os nós da VPN WireGuard. Este modelo aplica-se a todos os nós da infraestrutura, sendo necessário apenas comutar os valores relativos a `PrivateKey`, `PublicKey`, `Address` e `Endpoint` de acordo com cada servidor.

```
# NODE LOCAL
[Interface]
Address = 10.0.0.1/24
PrivateKey = <PrivateKey>
ListenPort = 51820
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; \
         iptables -A FORWARD -o wg0 -j ACCEPT; \
         iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; \
           iptables -D FORWARD -o wg0 -j ACCEPT; \
           iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE


# NODE EM NUVEM
[Peer]
PublicKey = <PublicKey>
Endpoint = 95.93.141.118:51820
AllowedIPs = 10.0.0.2/32, 10.233.0.0/16
PersistentKeepalive = 25


# NODE WITNESS (VPS)
[Peer]
PublicKey = <PublicKey>
Endpoint = 34.175.187.140:51820
AllowedIPs = 10.0.0.4/32
PersistentKeepalive = 25
```

A secção `[Interface]` define a configuração local da interface WireGuard (`wg0`), incluindo o endereço interno, a chave privada e as regras de encaminhamento (ativadas e removidas com `PostUp`/`PostDown`). As secções `[Peer]` especificam os pares da rede: a respetiva chave pública, o *endpoint* (IP:porto) e os endereços permitidos (`AllowedIPs`) a encaminhar através do túnel. O parâmetro `PersistentKeepalive` garante a manutenção da ligação ativa mesmo em cenários com NAT.

Desta forma, cada nó pode ser configurado de forma idêntica, alterando apenas os parâmetros específicos de chave e endereçamento.

Apêndice B

Configuração do HAProxy

Abaixo segue a documentação referente às configurações feitas no HAProxy.

Listing B.1: Configuração Patroni do nó pg-1

```
global
    log stdout format raw local0

defaults
    log global
    mode tcp
    timeout connect 5s
    timeout client 30s
    timeout server 30s

listen stats
  bind *:8404
  mode http
  stats enable
  stats uri /
  stats refresh 10s
  stats auth admin:admin # opcional, para autenticao

frontend pgsql_front
    bind *:5432
    default_backend pgsql_back

backend pgsql_back
  mode tcp
  option tcp-check
  default-server inter 1s fall 2 rise 1 on-marked-down shutdown-sessions
  tcp-check connect port 8008
  tcp-check send GET\ /primary\ HTTP/1.1\r\nHost:\ localhost\r\n\r\n
  tcp-check expect rstring 200

  server pg2 10.0.0.2:5432 check port 8008
  server pg1 10.0.0.4:5432 check port 8008
  server pg3 10.0.0.1:5432 check port 8008

```

```
35
36 frontend ldap_front
37     bind *:389
38     mode tcp
39     default_backend ldap_back
40
41 backend ldap_back
42     mode tcp
43     balance first
44     option tcp-check
45
46     default-server inter 1000ms fastinter 200ms downinter 200ms fall 2 rise 1 on-marked
           -down shutdown-sessions
47
48     tcp-check connect port 389
49     timeout check 1s
50
51
52
53     server primary-ldap 10.0.0.7:389 check
54     server dc-2 10.0.0.11:389 check backup
55     server dc-3 10.0.0.12:389 check backup
56     server backup-ldap 10.0.0.10:389 check backup
57
58 frontend fcul_http
59     bind *:88
60     mode http
61     acl host_fcul hdr(host) -i fcul.joaoteta.me
62     use_backend kc_backend if host_fcul
63     default_backend kc_backend
64
65 backend kc_backend
66     mode http
67     http-check send meth GET uri /admin/ ver HTTP/1.1 hdr Host fcul.joaoteta.me
68     http-check expect status 302
69
70     timeout check 1s
71     default-server inter 800ms fastinter 200ms downinter 2s fall 2 rise 1 on-marked-
           down shutdown-sessions
72
73     server on-prem 10.0.0.2:8888 check
74     server cloud keycloak:8080 check backup
75
76 frontend fcul_http_2
77     bind *:89
78     mode http
79     acl host_fcul hdr(host) -i fcul.joaoteta.me
80     use_backend kc_backend_2 if host_fcul
81     default_backend kc_backend_2
82
83 backend kc_backend_2
```

```
84      mode http
85      http-check send meth GET uri /admin/ ver HTTP/1.1 hdr Host fcul.joaoteta.me
86      http-check expect status 302
87
88      timeout check 1s
89      default-server inter 700ms fastinter 200ms downinter 2s fall 2 rise 1 on-marked-
            down shutdown-sessions
90
91      server on-prem 10.0.0.2:8889 check
92      server cloud keycloak_2:8080 check backup
```

Apêndice C

Configurações Patroni

Este anexo documenta a configuração dos três nós do cluster PostgreSQL gerido por *Patroni* (utilizado na hipótese com BD partilhada). Cada nó expõe: (i) `restapi` para controlo do Patroni; (ii) `raft` para eleição de líder; (iii) parâmetros de `bootstrap` e `postgresql`; e (iv) regras `pg_hba`. **Nota de segurança:** em produção, recomenda-se armazenar credenciais via variáveis de ambiente/secret manager e *não* em ficheiros versionados.

C.1 Topologia & Convenções

- **Scope**: `fcul-cluster` (todos os nós partilham o mesmo `scope`).
- **Raft**: comunicação entre pares em `:7000`; `data_dir` separado do `data_dir` do PostgreSQL.
- **Preferred leader**: atribuído a um nó (ver `preferred_leader: true`) para arranque mais previsível.
- **Rewind/Slots**: `use_pg_rewind` e `use_slots` ativados para *failover* limpo e replicação estável.

C.2 Nó `pg-1`

Listing C.1: Configuração Patroni do nó `pg-1`

```
scope: fcul-cluster
name: pg-1

restapi:
  listen: 0.0.0.0:8008
  connect_address: 10.0.0.4:8008
  metrics: true


bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    postgresql:
```

```
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        wal_log_hints: on
        hot_standby: on
        max_wal_senders: 10
        max_replication_slots: 10
        archive_mode: on
        archive_command: '/bin/true'
  initdb:
    - auth-local: md5
    - auth-host: md5
    - encoding: UTF8
    - locale: C.UTF-8
    - data-checksums
  users:
    kc_user:
      password: kc_pass
      options: [superuser, createrole, createdb]
    kc_replicator:
      password: kc_replicator_pass
      options: [replication]


# ---- DCS: RAFT embutido (sem etcd!) ----
raft:
  self_addr: 10.0.0.4:7000
  partner_addrs:
    - 10.0.0.2:7000
    - 10.0.0.1:7000
  data_dir: /var/lib/patroni/raft

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 10.0.0.4:5432
  data_dir: /var/lib/postgresql/17/main
  bin_dir: /usr/lib/postgresql/17/bin
  parameters:
    wal_level: replica
    max_wal_senders: 20
    max_replication_slots: 20
    hot_standby: "on"
  authentication:
    superuser:

      username: kc_user
      password: kc_pass
    replication:
      username: kc_replicator
      password: kc_replicator_pass
```

```
  pg_hba:
    - local all postgres peer
    - local replication postgres peer
    - host all all 172.24.0.0/16 md5
    #- host all all 172.26.0.32/32
    #- host all all 172.18.0.0/16 md5
    - host all all 127.0.0.1/32 md5
    - host replication kc_replicator 127.0.0.1/32 md5
    - host replication kc_replicator ::1/128 md5
    - host replication kc_replicator 10.0.0.0/24 md5
    - host all all 10.0.0.0/24 md5

tags:
  nosync: false
```

Notas rápidas sobre a configuração (pg-1).

- **DCS via Raft:** este nó usa RAFT embutido (sem etcd/consul), guardando estado em `/var/lib/patroni/raft`.
- **pg_hba:** além dos acessos locais, abre a rede `172.24.0.0/16` — adequado ao teu ambiente Docker/K8s. As linhas comentadas (`172.26.0.32/32`, `172.18.0.0/16`) sugerem ranges alternativos.
- **Autenticação local:** mantêm-se entradas `peer` para o utilizador `postgres`, mas os utilizadores reais da solução são `kc_user` e `kc_replicator`.
- **Tags:** `nosync: false` explicita que este nó participa normalmente na replicação (poderia ser usado para desativar sync num nó apenas de leitura).
- **Parâmetros PostgreSQL:** semelhantes ao nó `pg-2`, mas este nó não tem definido `preferred_leader`, pelo que pode ser eleito dinamicamente.

C.3 Nó pg-2

Ficheiro: `/etc/patroni/patroni.yml`

Listing C.2: Configuração Patroni do nó pg-2

```
scope: fcul-cluster
name: pg-2

restapi:
  listen: 0.0.0.0:8008
  connect_address: 10.0.0.2:8008
  metrics: true

raft:
  self_addr: 10.0.0.2:7000
  partner_addrs: [10.0.0.4:7000,10.0.0.1:7000]
  data_dir: /var/lib/postgresql/raft

```

```
bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    postgresql:
      use_pg_rewind: true
      use_slots: true
      parameters:
        wal_level: replica
        wal_log_hints: on
        hot_standby: on
        max_wal_senders: 10
        max_replication_slots: 10
        archive_mode: on
        archive_command: '/bin/true'
  initdb:
    - auth-local: md5
    - auth-host: md5
    - encoding: UTF8
    - locale: C.UTF-8
    - data-checksums
  users:
    kc_user:
      password: kc_pass
      options: [superuser, createrole, createdb]
    kc_replicator:
      password: kc_replicator_pass
      options: [replication]

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 10.0.0.2:5432
  data_dir: /var/lib/postgresql/17/main
  bin_dir: /usr/lib/postgresql/17/bin
  parameters:
    wal_level: replica
    max_wal_senders: 20
    max_replication_slots: 20
    hot_standby: "on"
  authentication:
    superuser:
      username: kc_user
      password: kc_pass
    replication:
      username: kc_replicator
      password: kc_replicator_pass
  pg_hba:
    - "local all all trust"
    - "host all all 127.0.0.1/32 md5"
    - "host all all ::1/128 md5"
```

```
    - "host replication kc_replicator 10.0.0.0/24 md5"
    - "host replication kc_replicator 127.0.0.1/32 md5"
    - "host replication kc_replicator ::1/128 md5"
    - "host all kc_user 10.0.0.0/24 md5"
    - "host all all 10.0.0.0/24 md5"
    - "host all all 172.18.0.0/16 md5"
    - "host all kc_user 195.200.15.13/32 md5"
    - "host all manager 0.0.0.0/0 md5"
    - "host all manager ::/0 md5" tags:
  preferred_leader: true
```

Notas rápidas sobre a configuração (pg-2).

- `restapi.metrics: true` ativa *endpoint* de métricas (útil para Prometheus/Grafana).
- `raft.partner_addrs`: pares `pg-1` e `pg-3`.
- `bootstrap.dcs.postgresql.parameters`: define requisitos para replicação (WAL, slots, *hot standby*).
- `users`: cria `kc_user` (admin) e `kc_replicator` (replicação).
- `pg_hba`: abre acessos necessários a replicação e clientes; em produção, restringe redes e evita `0.0.0.0/0`.
- `preferred_leader: true`: preferência deste nó como líder inicial (não impede reeleição automática).

C.4 Nó `pg-3 (witness)`

Listing C.3: Configuração Patroni do nó `pg-3`

```
scope: fcul-cluster
name: pg-3

restapi:
  listen: 0.0.0.0:8008
  connect_address: 10.0.0.1:8008

raft:
  self_addr: 10.0.0.1:7000
  partner_addrs:
    - 10.0.0.2:7000 # pg-2
    - 10.0.0.4:7000 # pg-1
  data_dir: /var/lib/patroni/raft

postgresql:
  listen: 0.0.0.0:5432
  connect_address: 10.0.0.1:5432
  data_dir: /var/lib/postgresql/17/witness
```

```
  bin_dir: /usr/lib/postgresql/17/bin
  parameters:
    wal_level: replica
    hot_standby: "on"
    max_wal_senders: 5
    max_replication_slots: 5
  authentication:
    superuser:
      username: kc_user
      password: kc_pass
    replication:
      username: kc_replicator
      password: kc_replicator_pass
  pg_hba:
    - "local all all trust"
    - "host all all 127.0.0.1/32 md5"
    - "host all all ::1/128 md5"
    - "host replication kc_replicator 10.0.0.0/24 md5"
    - "host replication kc_replicator 127.0.0.1/32 md5"
    - "host replication kc_replicator ::1/128 md5"
    - "host all kc_user 10.0.0.0/24 md5"
    - "host all all 10.0.0.0/24 md5"
    - "host all all 172.18.0.0/16 md5"
    - "host all kc_user 195.200.15.13/32 md5"
```

Notas rápidas sobre a configuração (pg-3).

- **Papel de witness/terceiro voto:** participa no quórum *Raft* para eleição de líder (`partner_addrs` aponta para `pg-1` e `pg-2`). Isto evita *split-brain*.
- **PostgreSQL minimalista:** mantém `hot_standby=on` e valores reduzidos de `max_wal_senders`/`max_replication_slots`, adequados a um nó testemunha/standby leve.
- **Diretório dedicado:** `data_dir` distinto (`/var/lib/postgresql/17/witness`) para separar claramente o papel deste nó.
- **Segurança (atenção):** `pg_hba` contém `"local all all trust"` (conveniente em laboratório, *não recomendado* em produção). Em produção, trocar por `peer`/`md5`/`scram-sha-256` conforme a política.
- **Sem `bootstrap` neste ficheiro:** este nó junta-se a um cluster já inicializado (o `bootstrap` foi feito noutro nó). Mantém utilizadores `kc_user` e `kc_replicator` alinhados com os restantes.
- **Eleições:** não define `preferred_leader`; pode ser eleito se os restantes caírem, mas o objetivo principal é fornecer voto/quórum e standby.