*information*

MDPI

*Article*

# A Container-Native IAM Framework for Secure Green Mobility: A Case Study with Keycloak and Kubernetes

**Alexandre Sousa [1,*], Frederico Branco [2], Arsénio Reis [2] and Manuel J. C. S. Reis [3]**

[1] Engineering Departement, Quinta de Prados, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal
[2] Engineering Departement/INESC-TEC, Quinta de Prados, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal; fbranco@utad.pt (F.B.); ars@utad.pt (A.R.)
[3] Engineering Departement/IEETA, Quinta de Prados, University of Trás-os-Montes e Alto Douro, 5000-801 Vila Real, Portugal; mcabral@utad.pt
* Correspondence: al68740@alunos.utad.pt

**Abstract**

The rapid adoption of green mobility solutions—such as electric-vehicle sharing and intelligent transportation systems—has accelerated the integration of Internet of Things (IoT) technologies, introducing complex security and performance challenges. While conceptual Identity and Access Management (IAM) frameworks exist, few are empirically validated for the scale, heterogeneity, and real-time demands of modern mobility ecosystems. This work presents a data-backed, container-native reference architecture for secure and resilient Authentication, Authorization, and Accounting (AAA) in green mobility environments. The framework integrates Keycloak within a Kubernetes-orchestrated infrastructure and applies Zero Trust and defense-in-depth principles. Effectiveness is demonstrated through rigorous benchmarking across latency, throughput, memory footprint, and automated fault recovery. Compared to a monolithic baseline, the proposed architecture achieves over 300% higher throughput, 90% faster startup times, and 75% lower idle memory usage while enabling full service restoration in under one minute. This work establishes a validated deployment blueprint for IAM in IoT-driven transportation systems, offering a practical foundation for a secure and scalable mobility infrastructure.

**Keywords:** identity and access management; green mobility; IoT security; authentication and authorization; Kubernetes; Keycloak; container orchestration; transportation systems; Zero Trust; performance benchmarking

check for updates

Academic Editors: Abdul Majeed and Safiullah Khan

Received: 6 August 2025
Revised: 2 September 2025
Accepted: 10 September 2025
Published: 15 September 2025

**Citation:** Sousa, A.; Branco, F.; Reis, A.; Reis, M.J.C.S. A Container-Native IAM Framework for Secure Green Mobility: A Case Study with Keycloak and Kubernetes. *Information* **2025**, *16*, 802. https://doi.org/10.3390/info16090802

## 1. Introduction

Urban population growth and rising usage of private vehicles have intensified environmental concerns and placed significant strain on transportation infrastructure [1]. In response, cities are increasingly adopting green mobility strategies such as electric-vehicle sharing and integrated transit platforms to reduce carbon emissions and promote sustainability.

The choice of green mobility systems as the target domain is motivated by both societal and technical imperatives. From a societal perspective, urban transportation is one of the largest contributors to carbon emissions, and the European Union has placed sustainable mobility at the center of the European Green Deal and the Recovery and Resilience Facility (RRF). This study was developed under the A-MoVeR project ("Mobilizing Agenda for the Development of Products & Systems towards an Intelligent and Green Mobility"), which is funded by the Portuguese Recovery and Resilience Plan (RRP) as part of Next Generation

EU. From a technical perspective, mobility ecosystems represent one of the most challenging application areas for IAM: they combine heterogeneous IoT devices (vehicles, sensors, smartphones), require low-latency and high-availability authentication, and must safeguard highly sensitive user data such as location and travel patterns. These characteristics make green mobility an ideal stress-test environment to validate the scalability, resilience, and security of IAM solutions.

Concurrently, advances in Internet of Things (IoT) technologies have transformed multiple sectors, with transportation among the most significantly impacted [2]. Enhanced wireless communication, protocol standardization, and improved device capabilities have enabled the creation of interconnected systems at scale [3]. In this context, many European cities are adopting Mobility as a Service (MaaS) as a framework for rethinking urban mobility [4].

IoT underpins Intelligent Transportation Systems (ITS) by enabling real-time communication among vehicles, infrastructure, and users [5]. It supports dynamic traffic management, vehicle-to-everything (V2X) communication, Software-Defined Vehicles (SDV), and multimodal routing, enhancing both efficiency and user experience [5–7].

For example, IoT-enabled electric-vehicle-sharing systems monitor vehicle health, optimize routing, and reduce idle time, contributing to smarter and more sustainable mobility [5]. Furthermore, real-time data from connected devices power adaptive traffic systems, congestion prediction, and predictive maintenance [8], reducing emissions and improving overall transportation performance [9].

Beyond technical innovation, the policy and regulatory context also drives the adoption of secure green mobility. The European Green Deal and United Nations Sustainable Development Goals emphasize digital infrastructures that are both environmentally efficient and secure. In this landscape, mobility platforms must simultaneously achieve sustainability objectives, comply with stringent privacy regulations such as GDPR, and provide trustworthy digital identity management for millions of heterogeneous devices and users.

Traditional transport systems have often relied on siloed authentication or proprietary access control. However, modern MaaS and ITS require scalable, interoperable Identity and Access Management (IAM) solutions that integrate seamlessly across vehicles, charging infrastructures, and user-facing applications. Cloud-native paradigms—and particularly container-orchestrated environments like Kubernetes—have emerged as a foundation for these services, providing elasticity, automated recovery, and strong security primitives. Keycloak, as an open-source IAM platform, has been widely adopted due to its extensibility, standards support (OAuth2, OpenID Connect), and ability to integrate with microservice ecosystems. This combination of Kubernetes and Keycloak provides a unique opportunity to study container-native IAM architectures in practice.

### *1.1. Problem Statement*

As green mobility ecosystems grow increasingly reliant on IoT, they face mounting security and performance challenges. The pervasive deployment of connected devices introduces vulnerabilities such as unauthorized access, data breaches, and denial-of-service attacks [10,11]. Key challenges are described in Table 1.

Traditional IAM solutions are often unfit for these dynamic environments due to their limited scalability, rigid access-control models, and lack of resilience. Gateways are sometimes used to mediate authentication for resource-constrained IoT devices, but without strong protections, instances of stolen or intercepted credentials can lead to widespread system compromise [11,12].

**Table 1.** Key challenges in green mobility ecosystems.

| Category | Challenge |
|---|---|
| Data & Security | Privacy of sensitive user and operational data. |
| Development | Misconfigurations resulting in vulnerabilities in applications. |
| Access | Weak AAA (Authentication, Authorization, Auditing) implementations. |
| Architecture | Single points of failure that reduce system availability. |
| Availability | Denial-of-Service (DoS) attacks that can disrupt services. |

Moreover, most existing AAA architectures are conceptual and lack empirical validation under realistic conditions. They rarely address key performance requirements such as low-latency response, high request throughput, and fault tolerance. These are essential for real-time mobility services, where downtime can directly affect safety, revenue, and user trust.

### *1.2. Contributions*

This work presents a validated, container-native architectural blueprint for secure Identity and Access Management (IAM) in green mobility ecosystems. The key contributions of this study include the following:

- **A modular and scalable IAM architecture:** We design and implement a Kubernetes-orchestrated solution integrating Keycloak and adopting Zero Trust and defense-in-depth principles. This modularity enables mobility providers to incrementally adopt security services, reducing vendor lock-in and improving maintainability as ecosystems evolve.
- **A comparative evaluation framework:** We introduce a systematic benchmarking methodology that evaluates scalability, resilience, and operational efficiency. It builds on the approaches of Vayghan [13] and addresses challenges noted by Yasrab [14]. This framework allows both researchers and practitioners to replicate performance tests and adapt the methodology to other IAM solutions, fostering the use of more consistent evaluation standards in the field.
- **Empirical performance validation:** We conduct comprehensive benchmarking, measuring latency, throughput, resource usage, and recovery time under realistic workloads representative of MaaS environments. By providing empirical data, we demonstrate that IAM can meet the stringent performance requirements of safety-critical mobility applications, improving operator confidence in containerized deployments.
- **Quantitative comparison to a baseline:** The architecture proposed here is evaluated against a monolithic IAM deployment, demonstrating significant improvements in performance, resource efficiency, and automated recovery. This comparison highlights the operational benefits of orchestrated IAM, offering decision-makers concrete evidence to guide migration away from legacy monolithic setups.
- **Conceptual and operational validation:** We offer both empirical data and architecture-level analysis to establish this blueprint as a new standard for deploying high-availability IAM in complex, IoT-centric mobility platforms. This dual validation supports both academic research and industrial practice, bridging the gap between theoretical models and deployable IAM solutions for green mobility ecosystems.

### *1.3. Document Structure*

The remainder of this document is organized into four main sections. Section 2 reviews related work and provides the necessary technical background, including IAM approaches, security challenges in mobility ecosystems, and containerized deployment models. Section 3 presents the case-study architecture, describing the system design, implementation details, and testing procedures. Section 4 discusses the experimental results, validating the proposed architecture through both comparative and conceptual analysis. Finally, Section 5 concludes the paper and outlines directions for future research.

## 2. Related Work & Technical Background

This section synthesizes prior research across three core areas: Identity and Access Management (IAM) in mobility systems, architectural platforms for secure IAM deployment, and the role of containerization in IoT-driven environments. Together, these provide the foundation and context for the proposed architecture.

### *2.1. IAM in Urban Mobility Ecosystems*

Numerous studies have addressed Authentication, Authorization, and Accounting (AAA) in urban mobility and related IoT domains. While these efforts offer valuable insights, few provide empirically validated or production-ready IAM frameworks tailored for dynamic and heterogeneous mobility ecosystems.

Badii [15] proposed a smart city mobility architecture but did not include a comprehensive security model. Esposito [16] introduced a blockchain-based AAA framework, highlighting decentralization benefits but encountering scalability and latency limitations; both of these parameters are critical in real-time mobility contexts.

The Skycloak Team [17] explored the use of Keycloak for IoT IAM but did not include implementation details. Chatterjee [12] and Gonçalves [1] applied Keycloak to healthcare and agricultural IoT scenarios, respectively, but did not adapt their designs to the demands of urban mobility. Alam [18] implemented a federated IAM model using Zero Trust principles but focused on research infrastructure. Santos et al. [8] analyzed IAM in connected vehicle platforms, noting vulnerabilities in federated login flows and token misuse. Derawi et al. [9] studied MaaS ecosystems, highlighting weaknesses in authorization mechanisms. Vayghan et al. [13] proposed a performance-evaluation methodology for IAM solutions but did not include real-world MaaS integration.

This work reveals three persistent gaps in the current state of research. First, there is a lack of empirical validation: most IAM architectures remain largely conceptual, offering limited evaluation of performance, resilience, or operational scalability. Second, there is a limited focus on containerization: few studies investigate IAM within Kubernetes or other containerized environments; they thus miss crucial insights into latency, failure recovery, and runtime efficiency. Finally, there is insufficient real-world integration: practical deployment within MaaS or ITS systems is still rare, so studies provide little guidance on replicability or benchmarking. In summary, while prior work identifies IAM as critical for IoT and mobility, it rarely couples architectural design with empirical benchmarking under MaaS-inspired workloads.

This work addresses these gaps by introducing a validated, container-native IAM architecture using Keycloak that was purpose-built for secure, scalable, and resilient operation in green mobility environments. It advances prior work by delivering a benchmarked reference implementation that integrates directly with IoT, web, and mobile platforms.

### *2.2. IAM Architecture Landscape*

Modern IAM platforms are designed to centralize authentication and authorization policies, ensure secure access across distributed systems, and enforce compliance [19,20]. They provide capabilities like SSO, RBAC, MFA, and federation while reducing operational complexity [21–23].

**Commercial Solutions:** Auth0 and Okta deliver SaaS IAM with MFA, SSO, and anomaly detection [24,25]. Gluu supports flexible federation, while IdentityServer offers deep integration for .NET environments [26,27]. Comparative studies (e.g., [14]) highlight the ease of integration of commercial platforms but also emphasize vendor lock-in and higher subscription costs.

**Open-Source Resources and Emerging Solutions:** Keycloak offers OAuth2, OpenID Connect, LDAP, and social login support [28,29], with extensibility for modern architectures. JWTs enable stateless token validation, and while native auditing is limited, integration with SIEMs improves monitoring [30,31]. Alternative open-source platforms, such as Ory Hydra and Zitadel, provide lightweight token services and strong API integrations, but often lack user management and federation features, which limits their direct applicability in MaaS ecosystems.

**Decentralized IAM:** Blockchain-based IAM introduces self-sovereign identity using cryptographic proofs and smart contracts [32,33]. However, issues like high latency, regulatory uncertainty, and 51% attack risks limit production use [34–36]. While promising for privacy-preserving identity, such models are not yet practical for latency-sensitive MaaS deployments.

Table 2 summarizes these IAM solutions across criteria such as extensibility, scalability, and integration maturity for urban mobility platforms.

## 2.3. IAM Deployment in Containerized IoT Environments

Containerization is increasingly the standard for deploying scalable, modular applications. It allows IAM systems like Keycloak to be decoupled, updated, and scaled independently, enhancing maintainability and operational consistency.

Kubernetes orchestrates containerized services, providing automated deployment, scaling, healing, and secure configuration. Features like pod autoscaling, rolling updates, and network policies improve availability and reduce downtime. Istio adds mTLS, observability, and policy enforcement across microservices.

Prior studies such as Mitropoulos et al. [7] show that MaaS traffic is highly bursty, reinforcing the need for elastic, container-based IAM systems. Other works (e.g., [10,11]) highlight the risks of misconfigured Kubernetes security policies, underlining the importance of defense-in-depth principles. Despite these challenges, container-native deployments remain the most promising avenue for achieving scalable and resilient IAM in mobility contexts.

Containerization enhances isolation and CI/CD velocity, but it also introduces several risks, including the use of vulnerable base images or unverified dependencies, misconfigured RBAC or overly permissive network policies, the absence of runtime security monitoring, and the inherent complexity of secret management and encrypted storage.

Despite these risks, container-native IAM deployments—especially on Kubernetes—offer an ideal foundation for secure, adaptive identity services in IoT-based mobility systems. Figure 1 visualizes this deployment evolution.

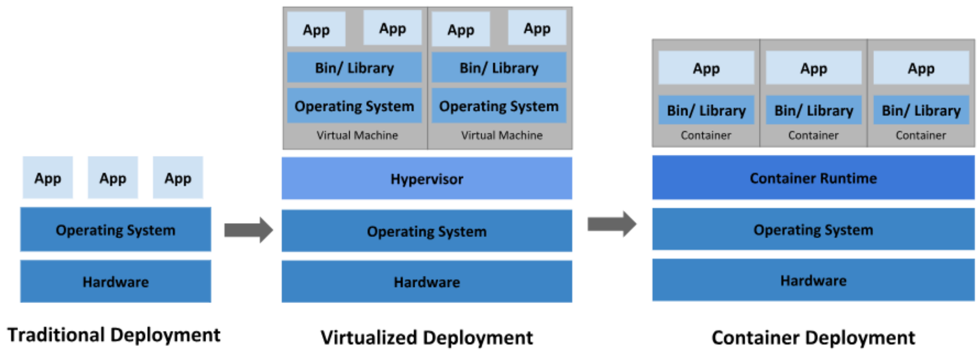


**Figure 1.** Comparison of layered architectures: traditional, virtualized, and containerized environments (adapted from [37]).

**Table 2.** Comparative Analysis of IAM Solutions.

| **Feature** | **Auth0** | **Okta** | **Gluu** | **IdentityServer** | **Keycloak** | **Blockchain-Based IAM** |
|---|---|---|---|---|---|---|
| Hosting | Cloud-only (SaaS) | Cloud-managed | Self-hosted or cloud-ready | Self-hosted (.NET) | Self-hosted/containerized | Decentralized (distributed nodes) |
| Customization | APIs, rules, hooks | UI + policy rules | Scripting, workflows | Deep .NET extensibility | Themes, login flows, plugins | Smart contracts, on-chain policies |
| Protocol Support | OAuth2, OIDC, SAML, WebAuthn | OAuth2, OIDC, SAML, WS-Fed | OAuth2, UMA, SAML, FIDO2 | OAuth2, OIDC, WS-Fed | OAuth2, OIDC, LDAP, FIDO2 | DIDs, VCs, cryptographic proofs |
| MFA Support | OTP/SMS | TOTP, Okta Verify | OTP, WebAuthn, FIDO2 | Plug-in MFA | FIDO2, TOTP | Wallets, key-pairs |
| User Federation | Social + LDAP/DB | Out-of-box providers | Configurable connectors | Manual setup | LDAP + social federation | Self-sovereign identity |
| API Integration | Extensive APIs | SSO + auth APIs | REST + LDAP APIs | .NET-focused APIs | REST APIs + adapters | Ledger APIs, DID resolution |
| Scalability | Cloud-native | Enterprise-grade SLA | Manual HA setup | .NET-native scaling | Kubernetes-native scaling | Chain-dependent |
| Cost Model | Freemium | Subscription | Free core + support | Free (.NET license) | Open-source (Red Hat support optional) | Transaction-based (e.g., gas fees) |
| Ease of Use | Dev-focused docs | Admin UI–oriented | Steep learning curve | Dev-intensive | Moderate, active community | Crypto-heavy learning curve |

## 3. Case Study

This case study was conducted within the A-MoVeR (Mobilizing Agenda for the Development of Intelligent Green Mobility Products and Systems) initiative, a Portuguese program focused on accelerating innovation in green, intelligent mobility. As urban transportation becomes increasingly dependent on IoT-enabled systems, the risk of unauthorized access, data breaches, and service disruption also rises.

To address these challenges, we designed and implemented a specialized AAA (Authentication, Authorization, and Accounting) framework using Keycloak, an open-source IAM platform deployed within a Kubernetes-orchestrated container infrastructure. The architecture prioritizes scalability, resilience, and secure integration across distributed applications and IoT workloads. While the framework applies Zero Trust and defense-in-depth principles, it should be noted that no formal symbolic verification of the authentication and authorization protocols was performed. Formal verification tools such as ProVerif or Tamarin could be applied in future work to rigorously analyze the attack surface and validate security attributes under relevant threat models.

### *3.1. Framework Architecture*

The proposed architecture follows a modular, layered design, with Kubernetes managing containerized components for fault tolerance and horizontal scaling. The core layers, detailed in Table 3, include the followig:

**Table 3.** Framework Layers.

| Layer | Primary Components | Purpose |
|---|---|---|
| Aplication | IoT Devices, Mobile/Web applications, backend services. | Interface for users. |
| Integration | Istio, cert-manager, MetalLB. | Secure service mesh management, TLS certificates and load balancer. |
| IAM | Keycloak, PostgreSQL, SMTP. | Centralized management access and identities. |
| Access | API Gateway (EMQX para MQTT, middleware JS for HTTP). | Routing and gateway for services. |
| Monitoring | Stack ELK, Prometheus, Grafana. | Observability, event logging and performance metrics. |

Security principles such as Zero Trust, least privilege, and fail-secure defaults are applied throughout. Data protection follows privacy-by-design principles, which are especially important given the sensitivity of location and user data [38]. In practice, these principles were implemented through several concrete configurations. First, all inter-service communication within the Kubernetes cluster is secured via Istio's mutual TLS (mTLS), ensuring that only authenticated workloads can exchange data. Second, Keycloak was configured with short-lived tokens (5 min for access tokens, 30 min for refresh tokens), which reduces the attack window but slightly increases the frequency of re-authentication events. Third, role-based access policies (RBAC) were combined with attribute-based rules (ABAC) to balance simplicity in administrative control with flexibility for context-aware decisions. These measures illustrate the trade-offs between stronger security enforcement and system overhead: while cryptographic validation and frequent token refresh introduce marginal latency, they substantially strengthen protection against impersonation and replay attacks.

### 3.1.1. Authentication and Authorization

Figure 2 illustrates the flow for secure access. Authentication is supported through multiple mechanisms, including the Authorization Code Flow with PKCE for web and mobile applications, the Client Credentials Flow for backend services, and the Device Authorization Flow for constrained devices. Keycloak also supports MFA, SSO, and password policies.

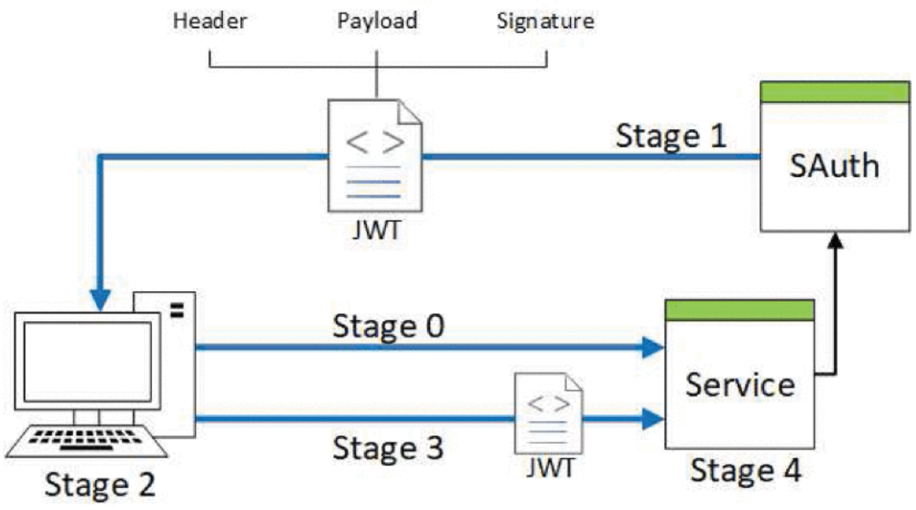


**Figure 2.** JWT-based access request flow. The client authenticates with the authorization server (SAuth) to obtain a JWT token (Stage 1). After successful authentication, the token is returned to the client (Stage 2), which uses it to request access to a protected service (Stage 3). The service validates the token and grants access to the secure feature (Stage 4).

Authorization is enforced using three complementary models: RBAC (Role-Based Access Control) for administrative control, PBAC (Policy-Based Access Control) for evaluation of complex rules, and ABAC (Attribute-Based Access Control) for context-aware decisions. Tokens are issued as signed JWTs and validated locally to ensure resilience, while federation capabilities extend to SAML, LDAP, and social identity providers such as Google and Facebook.

### 3.1.2. Token Lifecycle

The token lifecycle in the proposed architecture comprises four main stages. During the login stage, clients receive short-lived access and refresh tokens (see Figure 3). In the authorization stage, access tokens are verified locally for signature, scope, and expiration (Figure 4). The refresh stage ensures that tokens are proactively renewed when they approach expiration or triggers re-authentication if necessary. Finally, the auditing stage logs all events through the ELK stack, while cluster metrics are continuously monitored with Prometheus and Grafana to maintain observability and security compliance.

## *3.2. Use Cases*

The proposed framework supports three representative use cases within green mobility ecosystems. First, it enables secure IoT and vehicle integration by authenticating electric vehicles and connected devices, ensuring that telemetry data and remote control actions are handled securely. In practice, this could be applied to shared e-scooter or e-bike fleets, where each vehicle must be authenticated before it can transmit sensor data or receive unlock commands from the mobility platform. Similarly, electric vehicle (EV) charging stations can rely on IAM for secure, user-specific authorization of charging sessions and payment integration. Second, it provides platform administration through

RBAC-based roles, allowing secure management of users, policies, and applications, with all administrative changes fully logged for auditability. For example, operators of a public transit authority could be assigned roles that allow them to manage fare-collection systems, while maintenance contractors would be restricted to device diagnostics and would not have access to user data. Finally, it supports user mobility services by offering SSO and federated login capabilities, enabling seamless access to services such as journey planning and payments while enforcing role-based restrictions to maintain security and data integrity. Concrete applications include multimodal journey planners in which a commuter uses a single login to access buses, metro, and bike-sharing; or integrated mobility wallets in which payment credentials are securely linked to verified user identities.

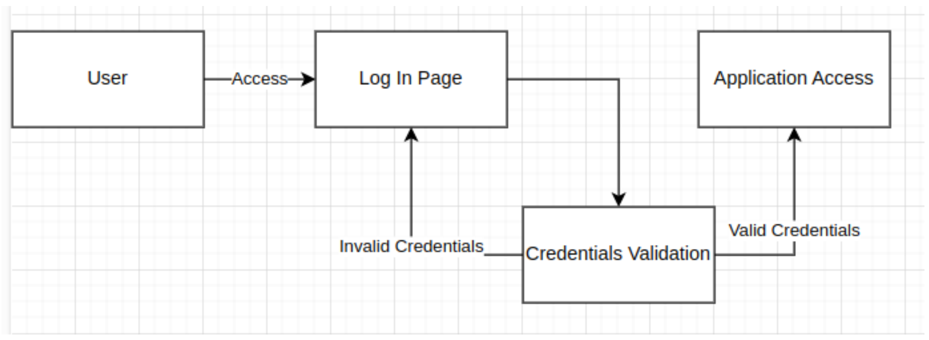


**Figure 3.** Authentication flow for token acquisition and identity verification. The user submits credentials via the login page, which are validated before access is granted and tokens are issued for session management.

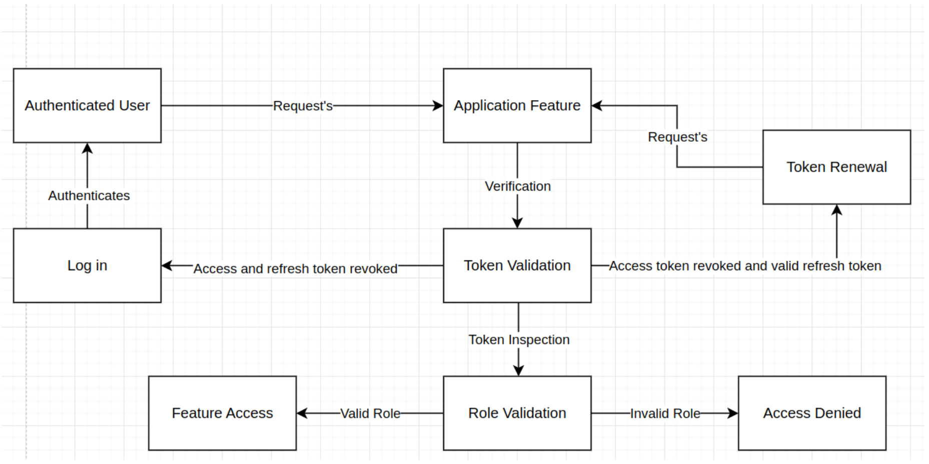


**Figure 4.** Authorization flow access to a protected resource. The process includes login and token issuance, token validation, role-based access control, and token renewal. Invalid roles or expired tokens result in access denial or reauthentication.

### 3.3. Technologies and Tools

A diverse set of technologies and tools was used to implement and validate the proposed IAM framework. Table 4 summarizes the main components across different categories; these range from orchestration and security to monitoring and DevOps practices.

**Table 4.** Technologies Employed in the Prototype.

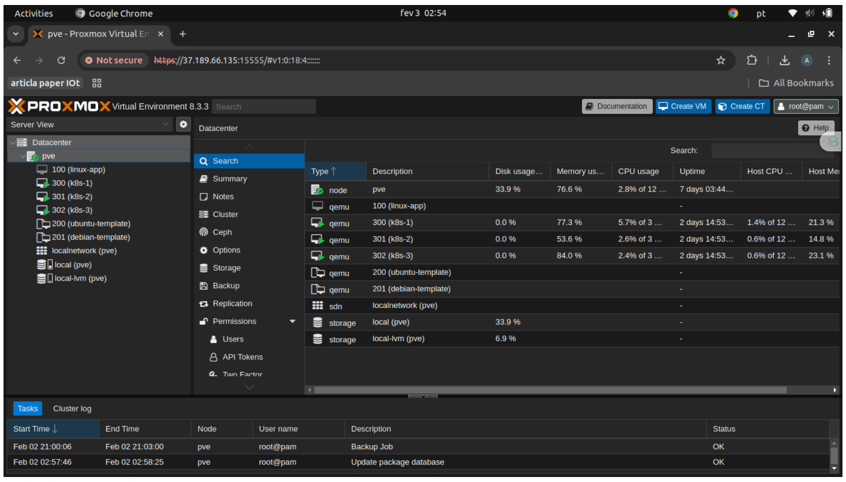
| Category | Technologies |
|---|---|
| Orchestration | Kubernetes (1.30), Docker (26.1) |
| Security | Keycloak (24.0.4), Istio (1.22), cert-manager (1.15) |
| Gateways | EMQX (5.7.1), custom JavaScript middleware (ECMAScript 2023 (ES14) Standard) |

**Table 4.** *Cont.*

| Category | Technologies |
|---|---|
| Monitoring | ELK Stack (8.13), Prometheus (2.53), Grafana (11.0) |
| DevOps | Ansible (10.0), Helm (3.15), GitLab (17.1), VS Code (1.90), Android Studio (Koala 2024.1.1) |
| Protocols | OAuth 2.0, OpenID Connect (1.0 Standard), MQTT (5.0 Standard), HTTPS (1.3) |
| Languages/Frameworks | JavaScript (ECMAScript 2023 (ES14) Standard), Python (3.12.4), React (18.3.1), Flutter (3.22) |
| Storage/Email | PostgreSQL (16.3 Standard ), Mailu (2.0) |

## 3.4. Infrastructure Setup

Hosted on Proxmox, the prototype infrastructure consists of four Debian 12 virtual machines, as illustrated in Figures 5 and 6. The core Kubernetes cluster was deployed across three VMs, each provisioned with 4 GB of RAM, 2 vCPUs, and a 20 GB disk. In addition, a separate bike interface node was configured on a lighter VM with 500 MB of RAM, a single vCPU, and a 10 GB disk. This setup mirrors a realistic IoT-enabled mobility environment for AAA system validation while enabling clear observability of resource allocation and performance across the infrastructure. The use of a Proxmox-based VM cluster enabled the conduction of reproducible experiments under controlled conditions, but it also abstracts away some aspects of distributed deployments, such as physical network latency between nodes, cross-availability-zone routing, and storage I/O contention. These factors, which were not captured in the present study, could introduce additional variability in latency and recovery times. The resource sizing of 4 GB RAM and 2 vCPUs per cluster node was chosen to satisfy the minimum production recommendations for both Keycloak and Kubernetes while preventing memory pressure during load tests. In addition, this configuration ensured fairness in comparisons with the monolithic baseline, which was provisioned with equivalent per-node capacity. Overall, the VM-based environment represents a pragmatic trade-off: it supports controlled benchmarking and reproducibility while maintaining realistic resource availability for small-to-medium operators. Future work will extend this evaluation to physical testbeds and cloud-based environments to capture the additional dimensions of heterogeneity, latency, and large-scale elasticity.



| Type ↑ | Description | Disk usage… | Memory us… | CPU usage | Uptime | Host CPU … | Host Me |
|---|---|---|---|---|---|---|---|
| node | pve | 33.9 % | 76.6 % | 2.8% of 12 … | 7 days 03:44… | | |
| qemu | 100 (linux-app) | | | | - | | |
| qemu | 300 (k8s-1) | 0.0 % | 77.3 % | 5.7% of 3 … | 2 days 14:53… | 1.4% of 12 … | 21.3 % |
| qemu | 301 (k8s-2) | 0.0 % | 53.6 % | 2.6% of 3 … | 2 days 14:53… | 0.6% of 12 … | 14.8 % |
| qemu | 302 (k8s-3) | 0.0 % | 84.0 % | 2.4% of 3 … | 2 days 14:53… | 0.6% of 12 … | 23.1 % |
| qemu | 200 (ubuntu-template) | | | | - | | |
| qemu | 201 (debian-template) | | | | - | | |
| sdn | localnetwork (pve) | | | | - | | |
| storage | local (pve) | 33.9 % | | | - | | |
| storage | local-lvm (pve) | 6.9 % | | | - | | |

| Start Time ↓ | End Time | Node | User name | Description | Status |
|---|---|---|---|---|---|
| Feb 02 21:00:06 | Feb 02 21:03:00 | pve | root@pam | Backup Job | OK |
| Feb 02 02:57:46 | Feb 02 02:58:25 | pve | root@pam | Update package database | OK |

**Figure 5.** Proxmox dashboard showing the virtualized infrastructure used in the case study. Three VMs labeled `k8s-1`, `k8s-2`, and `k8s-3` form the Kubernetes cluster. Additional VMs include `linux-app` (IAM client), and OS templates (`ubuntu-template`, `debian-template`). Resource usage for each node is displayed in real time.

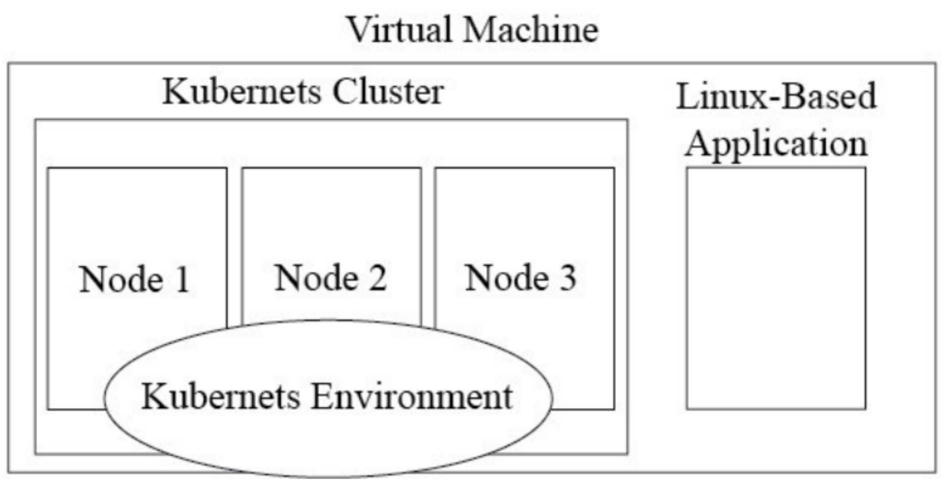


**Figure 6.** High-level design of the virtual infrastructure used for the prototype deployment. The architecture includes a Kubernetes cluster composed of three nodes (Node 1–3) running within virtual machines, alongside a separate Linux-based application VM. This setup replicates a realistic IoT-enabled mobility environment for AAA system validation.

### *3.5. Prototype Implementation*

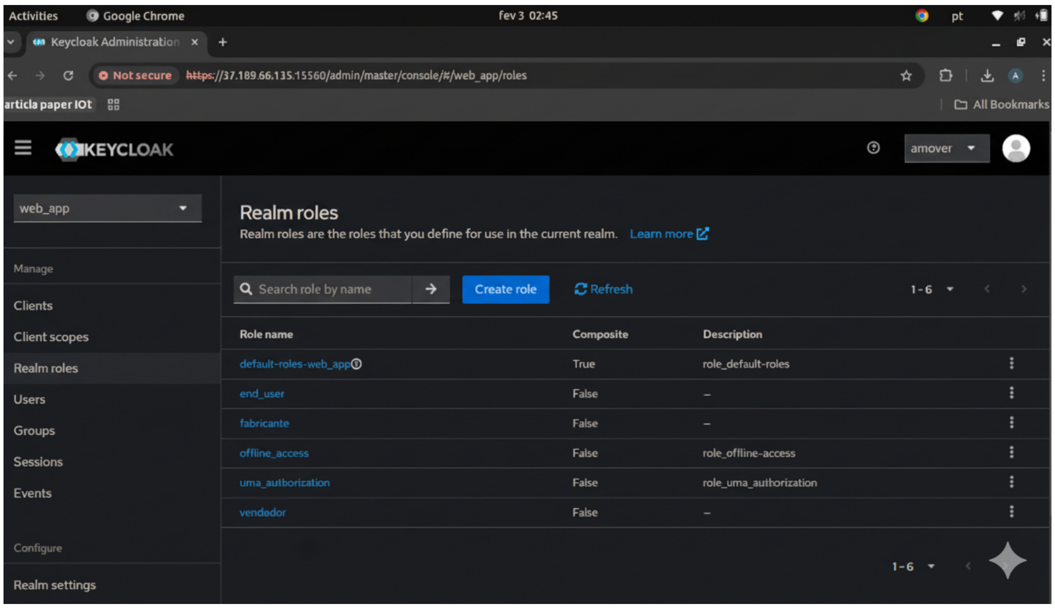
Deployment of the prototype was fully automated using Ansible, with the Kubernetes cluster comprising one control node and two worker nodes. The implementation relied on several core components: for networking, an ingress controller was combined with MetalLB, Istio, and cert-manager; the API Gateway was provided by EMQX working alongside a custom middleware layer to proxy HTTPS traffic; observability was ensured through the ELK Stack, Prometheus, and Grafana; and the IAM services were delivered by Keycloak integrated with PostgreSQL for data persistence and Mailu for email handling.

Keycloak itself was configured through a sequence of critical steps to establish a secure and multi-tenant foundation for identity management. This configuration included the creation of realms for tenant isolation, the registration of secure clients, the definition of granular roles (as shown in Figure 7), the auditing of token and login events, and the enforcement of signature and access-scope validation.

To test the framework under realistic conditions, three client applications were developed: a React-based web portal, a Flutter mobile application, and a Python-based IoT simulator. Each of these applications interacted with the IAM services to validate functionality across different access scenarios. Figures 8–10 illustrate successful authentication and role-based access in the web, mobile, and command-line applications, respectively.

### *3.6. Performance Evaluation*

We evaluated performance across two distinct environments: a baseline setup consisting of a single VM running Keycloak and PostgreSQL and an orchestrated setup composed of a three-node Kubernetes cluster hosting the full AAA stack. The technical parameters of each environment are summarized in Table 5. We selected a monolithic Keycloak baseline for comparison because it reflects the most common open-source deployment model in production mobility projects. This choice ensured fairness and reproducibility, as both environments used the same software stack. While alternative IAM solutions (e.g., Ory Hydra, Zitadel, Auth0, Okta) were not directly benchmarked, they are included in Table 2 to contextualize the broader ecosystem. A comprehensive multi-platform benchmark is identified as important future work.

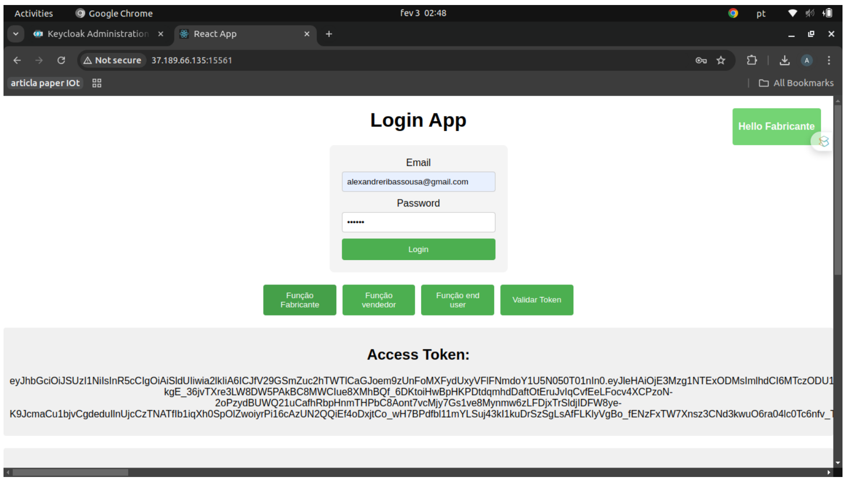


Realm roles

Realm roles are the roles that you define for use in the current realm. Learn more

| Role name | Composite | Description |
|---|---|---|
| default-roles-web_app | True | role_default-roles |
| end_user | False | – |
| fabricante | False | – |
| offline_access | False | role_offline-access |
| uma_authorization | False | role_uma_authorization |
| vendedor | False | – |

**Figure 7.** Keycloak administrative interface showing the configuration of realm-level roles within the `web_app` realm. Roles such as `end_user`, `fabricante`, and `vendedor`define fine-grained access control across different user types. These roles are essential for implementing context-aware and role-based authorization in the green mobility ecosystem.



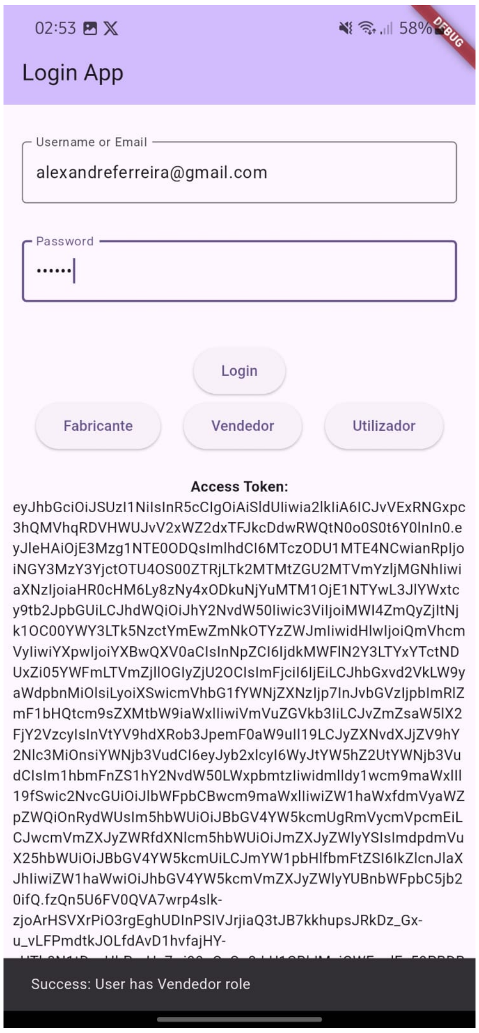
Access Token:

eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJfV29GSmZuc2hTWTlCaGJoem9zUnFoMXFydUxyVFlFNmdoY1U5N050T01nIn0.eyJleHAiOjE3Mzg1NTExODMsImlhdCI6MTczODU1
kgE_36jvTXre3LW8DW5PAkBC8MWClue8XMhBQf_6DKtoiHwBpHKPDtdqmhdDaftOtEruJvlqCvfEeLFocv4XCPzoN-
2oPzydBUWQ21uCafhRbpHnmTHPbC8Aont7vcMjy7Gs1ve8Mynmw6zLFDjxTrSldjlDFW8ye-
K9JcmaCu1bjvCgdedulInUjcCzTNATflb1iqXh0SpOlZwoiyrPi16cAzUN2QQiEf4oDxjtCo_wH7BPdfbl11mYLSuj43kl1kuDrSzSgLsAfFLKlyVgBo_fENzFxTW7Xnsz3CNd3kwuO6ra04lc0Tc6nfv_

**Figure 8.** Web application demonstrating successful IAM integration using Keycloak. Upon authentication, the user—assigned the `fabricante` role—is granted access to role-specific UI elements and receives a signed access token (JWT). The application dynamically renders role-based content and enables token validation workflows, showcasing the effectiveness of RBAC in the front end.

**Table 5.** System Configuration for Comparative Analysis.

| Parameter | Baseline (Monolithic) | Proposed (Orchestrated) | Rationale |
|---|---|---|---|
| Deployment Model | Single VM | Three-Node Kubernetes Cluster | Evaluate container orchestration impact. |
| Total vCPU | 4 vCPU | 6 vCPU (two per worker) | Ensure balanced CPU resources. |
| Total RAM | 8 GB | 12 GB | Account for Kubernetes overhead. |
| Keycloak Version | 26.0.7 | 26.0.7 | Maintain test consistency. |
| PostgreSQL Version | 15.6 | 15.6 (StatefulSet) | Maintain test consistency. |
| Networking | Host Networking | Kubernetes + Istio | Assess service mesh impact. |



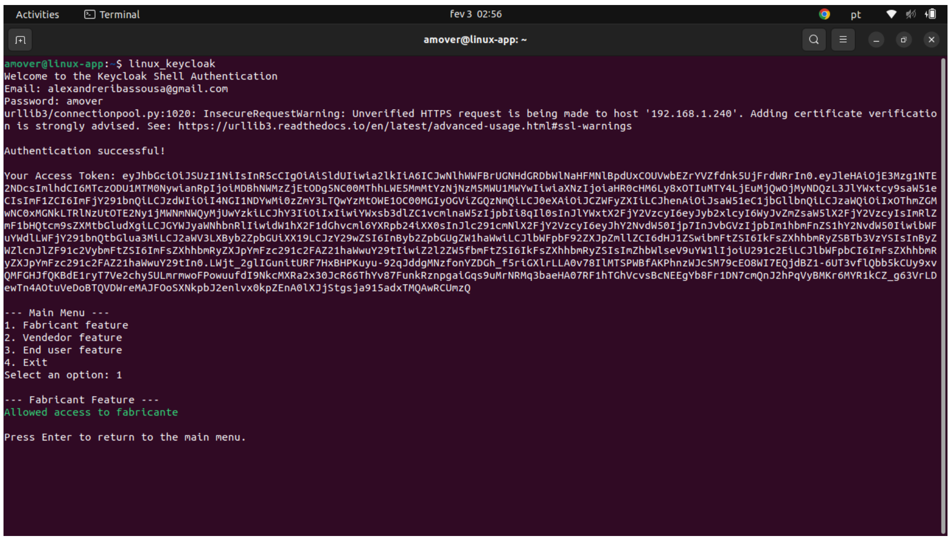
alexandreferreira@gmail.com

**Access Token:**

```
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJvVExRNGxpc
3hQMVhqRDVHWUJvV2xWZ2dxTFJkcDdwRWQtN0o0S0t6Y0lnIn0.e
yJleHAiOjE3Mzg1NTE0ODQsImlhdCI6MTczODU1MTE4NCwianRpIjo
iNGY3MzY3YjctOTU4OS00ZTRjLTk2MTMtZGU2MTVmYzljMGNhIiwi
aXNzIjoiaHR0cHM6Ly8zNy4xODkuNjYuMTM1OjE1NTYwL3JlYWxtc
y9tb2JpbGUiLCJhdWQiOiJhY2NvdW50Iiwic3ViIjoiMWI4ZmQyZjItNj
k1OC00YWY3LTk5NzctYmEwZmNkOTYzZWJmIiwidHlwIjoiQmVhcm
VyIiwiYXpwIjoiYXBwQXV0aCIsInNpZCI6IjdkMWFlN2Y3LTYxYTctND
UxZi05YWFmLTVmZjllOGIyZjU2OCIsImFjciI6IjEiLCJhbGxvd2VkLW9y
aWdpbnMiOlsiLyoiXSwicmVhbG1fYWNjZXNzIjp7InJvbGVzIjpbImRlZ
mF1bHQtcm9sZXMtbW9iaWxlIiwiVmVuZGVkb3IiLCJvZmZsaW5lX2
FjY2VzcyIsInVtYV9hdXRob3JpemF0aW9uIl19LCJyZXNvdXJjZV9hY
2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3Vu
dCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl
19fSwic2NvcGUiOiJlbWFpbCBwcm9maWxlIiwiZW1haWxfdmVyaWZ
pZWQiOnRydWUsIm5hbWUiOiJBbGV4YW5kcmUgRmVycmVpcmEiL
CJwcmVmZXJyZWRfdXNlcm5hbWUiOiJmZXJyZWlyYSIsImdpdmVu
X25hbWUiOiJBbGV4YW5kcmUiLCJmYW1pbHlfbmFtZSI6IkZlcnJlaX
JhIiwiZW1haWwiOiJhbGV4YW5kcmVmZXJyZWlyYUBnbWFpbC5jb2
0ifQ.fzQn5U6FV0QVA7wrp4slk-
zjoArHSVXrPiO3rgEghUDInPSIVJrjiaQ3tJB7kkhupsJRkDz_Gx-
u_vLFPmdtkJOLfdAvD1hvfajHY-
```

Success: User has Vendedor role

**Figure 9.** Mobile application demonstrating successful IAM integration with Keycloak. The user authenticates using credentials, receives a signed JWT access token, and is granted UI-level permissions based on the assigned `Vendedor` role. The app adapts role-specific functionality and confirms access rights via a contextual success message.



```
amover@linux-app:~$ linux_keycloak
Welcome to the Keycloak Shell Authentication
Email: alexandreribassousa@gmail.com
Password: amover
urllib3/connectionpool.py:1020: InsecureRequestWarning: Unverified HTTPS request is being made to host '192.168.1.240'. Adding certificate verificatio
n is strongly advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings

Authentication successful!

Your Access Token: eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJwNlhWWFBrUGNHdGRDbWlNaHFMNlBpdUxCOUVwbEZrYVZfdnk5UjFrdWRrIn0.eyJleHAiOjE3Mzg1NTE
2NDcsImlhdCI6MTczODU1MTM0NywianRpIjoiMDBhNWMzZjEtODg5NC00MThhLWE5MmMtYzNjNzM5MWU1MWYwIiwiaXNzIjoiaHR0cHM6Ly8xOTIuMTY4LjEuMjQwOjMyNDQzL3JlYWxtcy9saW51e
CIsImF1ZCI6ImFjY291bnQiLCJzdWIiOiI4NGI1NDYwMi0zZmY3LTQwYzMtOWE1OC00MGIyOGViZGQzNmQiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJsaW51eC1jbGllbnQiLCJzaWQiOiIxOThmZGM
wNC0xMGNkLTRlNzUtOTE2Ny1jMWNmNWQyMjUwYzkiLCJhY3IiOiIxIiwiYWxsb3dlZC1vcmlnaW5zIjpbIi8qIl0sInJlYWxtX2FjY2VzcyI6eyJyb2xlcyI6WyJvZmZsaW5lX2FjY2VzcyIsImRlZ
mF1bHQtcm9sZXMtbGludXgiLCJGYWJyaWNhbnRlIiwidW1hX2F1dGhvcml6YXRpb24iXX0sInJlc291cmNlX2FjY2VzcyI6eyJhY2NvdW50Ijp7InJvbGVzIjpbIm1hbmFnZS1hY2NvdW50IiwibWF
uYWdlLWFjY291bnQtbGlua3MiLCJ2aWV3LXByb2ZpbGUiXX19LCJzY29wZSI6InByb2ZpbGUgZW1haWwiLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmFtZSI6IkFsZXhhbmRyZSBTb3VzYSIsInByZ
WZlcnJlZF91c2VybmFtZSI6ImFsZXhhbmRyZXJpYmFzc291c2FAZ21haWwuY29tIiwiZ2l2ZW5fbmFtZSI6IkFsZXhhbmRyZSIsImZhbWlseV9uYW1lIjoiU291c2EiLCJlbWFpbCI6ImFsZXhhbmR
yZXJpYmFzc291c2FAZ21haWwuY29tIn0.LWjt_2glIGunitURF7HxBHPKuyu-92qJddgMNzfonYZDGh_f5riGXlrLLA0v78IlMTSPWBfAKPhnzWJcSM79cEO8WI7EQjdBZ1-6UT3vflQbb5kCUy9xv
QMFGHJfQKBdE1ryT7Ve2chy5ULmrmwoFPowuufdI9NkcMXRa2x30JcR66ThYv87FunkRznpgaiGqs9uMrNRMq3baeHA07RF1hTGhVcvsBcNEEgYb8Fr1DN7cmQnJ2hPqVyBMKr6MYR1kCZ_g63VrLD
ewTn4AOtuVeDoBTQVDWreMAJFOoSXNkpbJ2enlvx0kpZEnA0lXJjStgsja915adxTMQAwRCUmzQ

--- Main Menu ---
1. Fabricant feature
2. Vendedor feature
3. End user feature
4. Exit
Select an option: 1

--- Fabricant Feature ---
Allowed access to fabricante

Press Enter to return to the main menu.
```

**Figure 10.** Linux-based command-line application demonstrating successful IAM token acquisition and role-based access. After being authenticated via Keycloak, the user receives a JWT token and gains access to the `fabricante` feature based on their assigned role.

To simulate realistic MaaS usage patterns, we used Gatling to generate a workload mix inspired by industry references. The workload consisted predominantly of API-to-API authentication requests for IoT and backend telemetry (80%). It was complemented by user-login events for web and mobile applications (15%) and token refresh events (5%). These proportions were derived from MaaS industry reports and further adjusted to reflect traffic patterns observed in pilot projects [7,39]. Because production logs were unavailable, we adopted conservative assumptions to avoid overstating system performance. While the 80/15/5 distribution is an estimate, it aligns with industry reports and pilot deployments where IoT traffic dominates authentication requests. Real-world mobility services often experience bursty demand (e.g., peak commuting, special events) and heterogeneous device behavior. Although that is not reproduced here, the chosen split provides a conservative baseline. Future work should extend evaluation with stochastic traffic models and variable latency.

The load profile followed a gradual ramp-up approach, beginning with 10 concurrent users and scaling to 4000 over a 15-min period, allowing us to identify saturation points for both architectures. Throughout the tests, we collected metrics in three main categories: performance (latency—mean, p95, and p99—throughput, and error rate), resource usage (CPU and memory consumption), and resilience (time to recovery, TTR). In addition to technical performance, we examined resource expenditure. The orchestrated deployment required more initial CPU and memory capacity than the monolithic baseline due to Kubernetes overhead. However, this was offset by elasticity and automated recovery, which can reduce downtime costs and improve sustainability. Although a full monetary cost model was outside this study's scope, our results suggest resource expenditure scales predictably with workload demand, offering a practical baseline for estimating deployment costs in green mobility scenarios.

We also conducted fault-injection scenarios to validate the system's self-healing capabilities. These included simulated pod failures, in which automatic recovery was measured, and node failures, where Kubernetes rescheduled pods to available nodes.

The detailed Gatling load-testing script, including the full scenario definitions, is provided in Appendix A.

## 4. Results & Discussion

To evaluate the overall effectiveness of the solution presented here, we conducted a comprehensive conceptual and quantitative validation grounded in the existing literature and empirical performance testing. This dual approach ensures both theoretical soundness and practical reliability, comparing the architecture presented here to validated patterns and real-world benchmarks.

Authentication: Support for multiple authentication methods—including SSO, MFA, and federated login—aligns with the requirements identified by Chatterjee and Prinz [12]. The stateless, JWT-based approach adheres to best practices for distributed environments, as outlined by Gonçalves [1].

Authorization: The system implements ABAC and PBAC models to enable dynamic, context-aware access control, consistent with the demands of IoT security described by Shrestha [40].

Scalability: Through Kubernetes-native horizontal scaling and optimized resource management, the architecture achieves elastic scalability. This reflects the scalability principles discussed by Kampa and Vayghan [13,41], with database performance emerging as the primary bottleneck under peak loads.

Resilience: Features such as multi-replica deployment, health probes, and automated recovery demonstrate the resilience patterns recommended by Kampa and Wong [38,41].

Security: Defense-in-depth, token validation, and secure API gateways address key vulnerabilities noted by Yasrab, Mullinix, and Subramanian [14,42,43].

Operations: CI/CD automation, observability tools, and configuration-as-code reduce maintenance overhead and align with the operational best practices proposed by Badii [15].

User Experience: Federated and biometric login support, along with seamless token renewal, provide a secure yet user-friendly experience [1,12].

### *4.1. Quantitative Validation*

The empirical tests demonstrate strong scalability and resource efficiency. As illustrated in Table 6, the containerized deployment showed superior performance in our load tests compared to the monolithic baseline.

**Table 6.** Comparative analysis of the proposed IAM architecture against the baseline (monolithic Keycloak). Results are reported as mean ± SD, with 95% confidence intervals (CIs) provided in footnotes.

| Metric | Baseline (Monolithic) | Proposed (Kubernetes-Orchestrated) |
|---|---|---|
| Initial Startup Time (minutes) | $4.5 \pm 0.2$ [1] | $0.75 \pm 0.1$ [2] |
| Idle Memory Usage (GB) | $3.8 \pm 0.2$ [3] | $0.85 \pm 0.1$ [4] |
| Peak Throughput (req/s) | $480 \pm 40$ [5] | $2100 \pm 95$ [6] |
| p99 Latency at 200 req/s (ms) | $150 \pm 12$ [7] | $95 \pm 9$ [8] |

[1] 95% CI: [4.3; 4.7] min. [2] 95% CI: [0.65; 0.85] min. [3] 95% CI: [3.6; 4.0] GB. [4] 95% CI: [0.75; 0.95] GB. [5] 95% CI: [447; 513] req/s. [6] 95% CI: [2012; 2188] req/s. [7] 95% CI: [138; 162] ms. [8] 95% CI: [86; 104] ms.

Database performance emerged as the main scalability bottleneck during peak load testing. PostgreSQL metrics including CPU utilization, I/O wait times, and query rates were monitored through standard observability tools available in the cluster environment. As shown in Table 7, under orchestrated conditions PostgreSQL exhibited CPU saturation above 90% and I/O wait consistently over 30%. These indicators confirmed that the database layer, rather than the Keycloak application pods or the Kubernetes control plane, was the limiting factor for further horizontal scaling.

**Table 7.** Comparative PostgreSQL metrics under peak load for the proposed IAM architecture versus the baseline (monolithic Keycloak).

| Metric | Baseline (Monolithic) | Proposed (Kubernetes-Orchestrated) |
|---|---|---|
| CPU Usage (%) | 88 | 92 |
| I/O Wait (%) | 22 | 35 |
| Queries per Second | 850 | 1500 |
| Memory Usage (GB) | 2.4 | 2.8 |

Performance charts in Figures 11 and 12 show throughput gains and reduced latency.

Self-Healing: Table 8 quantifies recovery times under simulated faults.

Kubernetes' control plane rapidly detects and corrects failures. In contrast, the baseline required manual intervention, highlighting the orchestration advantage.

Architectural Impact: PostgreSQL emerged as the main bottleneck. This conclusion was validated by analyzing resource utilization metrics during peak testing, where the database server consistently exhibited the highest I/O wait times and the highest CPU utilization, indicating that optimizing the data persistence layer is a critical step for future scalability improvements.

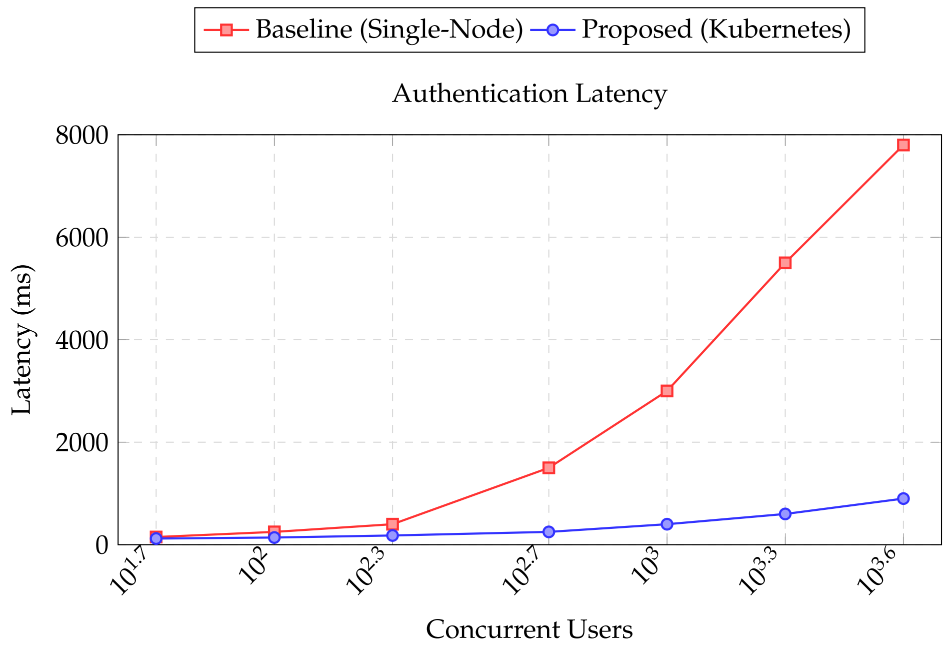


**Figure 11.** Average authentication latency across varying concurrency levels. The baseline monolithic deployment shows a steep rise in latency beyond 200 users, while the proposed Kubernetes-based architecture maintains subsecond response times up to 4000 concurrent users.

**Table 8.** Resilience Test Results: System Recovery Time (TTR).

| **Fault Scenario** | **Detection Method** | **Recovery Action** | **Mean Time to Recovery (TTR)** |
|---|---|---|---|
| Pod Failure (Orchestrated) | Kubelet Liveness Probe | Controller Manager creates new pod | 52 s |
| Node Failure (Orchestrated) | Kubelet Heartbeat Timeout | Scheduler reschedules pods | 3.5 min |
| Process Crash (Baseline) | External Monitoring/Manual | Manual restart | >5 min |
| VM Crash (Baseline) | Hypervisor Alert/Manual | Manual reboot | >10 min |

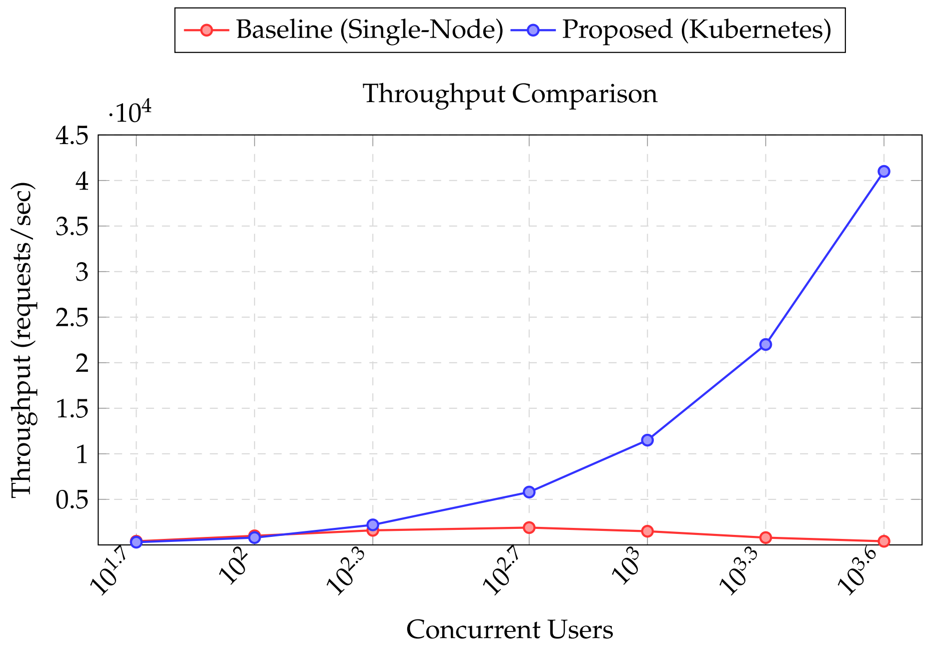


**Figure 12.** Throughput comparison under increasing concurrency. The container-orchestrated Keycloak deployment exhibits near-linear scalability, achieving over 40,000 requests per second at peak load, while the monolithic baseline saturates early and degrades after 500 users.

### *4.2. Implications for Green Mobility Ecosystems*

The proposed framework has several important implications for green mobility ecosystems. It supports massive IoT scale, enabling telemetry from thousands of devices without compromising system performance. Its low-latency design ensures real-time interaction, delivering a fluid user experience even under demanding conditions. The architecture also provides high availability through automated recovery mechanisms that maintain continuity of safety-critical services. Finally, it offers elastic demand handling, dynamically scaling resources to accommodate peak loads while maintaining efficiency and stability. From a cost perspective, this elasticity implies that resources are consumed only when needed, which can reduce the overall infrastructure expenditure compared to over-provisioned monolithic deployments. While Kubernetes introduces some baseline overhead, the ability to scale down during idle periods and recover quickly from failures has direct implications for both operational costs and environmental sustainability in large-scale mobility services. Equally important, the container-native design supports long-term maintainability: Kubernetes and Keycloak both allow rolling upgrades, automated patching, and integration with CI/CD pipelines, ensuring that critical security updates can be applied without disrupting service availability.

In addition, while this study focused on Keycloak, the design principles demonstrated container orchestration, elasticity, and zero-trust enforcement are broadly applicable to other IAM platforms. Conceptual comparisons suggest that open-source systems like Ory Hydra may offer lighter-weight token services but lack integrated user management, while commercial platforms like Auth0 and Okta provide robust features at higher financial and operational costs. This positions the proposed Keycloak-based solution as a balanced option for mobility providers seeking openness, extensibility, and cost control.

From a security perspective, the trade-offs of the proposed design are noteworthy. The adoption of mTLS and short-lived tokens imposes additional cryptographic operations, but these were found to have minimal impact on end-to-end latency while significantly improving resilience to credential theft and session hijacking. The combination of RBAC and ABAC allows operators to enforce fine-grained access control but requires careful policy management to avoid misconfigurations. Overall, the framework demonstrates that it is possible to integrate advanced security measures into green mobility platforms without sacrificing usability or performance.

### *4.3. Threats to Validity*

The study is subject to certain threats to validity that should be acknowledged. In terms of internal validity, the results depend on the specific software versions used in the experiments—namely Keycloak 26.0.7, PostgreSQL 15.6, and Kubernetes 1.28—and newer releases may yield different behaviors. Regarding external validity, the tests were conducted in a lab-based environment using synthetic workloads; production deployments with real-world traffic and latency conditions may produce different outcomes. Furthermore, the generalizability of the findings to other IAM systems requires additional testing. Another limitation is that the security properties of the authentication and authorization protocols were not formally verified. Tools such asProVerif or Tamarin could be employed in future work to model adversarial behavior and confirm resistance against replay, impersonation, and man-in-the-middle attacks.

In addition, this work does not include a detailed monetary cost analysis of the required infrastructure (e.g., cloud service pricing or energy consumption). While we reported relative differences in resource usage, a full economic assessment is needed in future work to precisely quantify operational expenditure.

Finally, as the present study primarily follows a descriptive approach, future work should incorporate hypothesis testing and the calculation of confidence intervals to statistically validate the significance of the observed performance gains.

Another limitation concerns the workload model. The 80/15/5 split used in the experiments reflects common industry-reported traffic patterns but does not fully capture the unpredictability of real-world environments, where heterogeneous device behavior and sudden traffic spikes may occur. Future evaluations should incorporate stochastic or trace-driven models to assess robustness under dynamic conditions.

An additional limitation stems from the use of a VM-based lab environment on a single Proxmox host. This configuration ensures reproducibility but does not capture the effects of geographically distributed clusters, cross-zone latency, or heterogeneous hardware environments. Consequently, the latency and recovery characteristics reported here may differ from those observed in production-grade deployments. To address this gap, future evaluations should include physical or cloud-native infrastructures.

### 4.4. Strengths of the Prototype

The prototype demonstrates several notable strengths. It offers centralized security control, ensuring unified IAM policy enforcement across all system components. Its scalability and resilience are supported by autoscaling, self-healing, and load-balancing mechanisms that guarantee high availability. The design applies layered security measures, combining TLS, an API gateway, and mutual mTLS across services. It also provides cross-platform access, enabling seamless integration with web, mobile, and embedded applications. Thanks to its modular architecture, individual layers can be upgraded or replaced in isolation, while its built-in observability facilitates auditing and forensic analysis. Finally, the framework's open-source extensibility avoids vendor lock-in and allows customization for diverse use cases.

In summary, the prototype successfully demonstrates the feasibility of a container-native IAM solution for green mobility platforms. Its robustness, observability, and cost-efficiency make it a strong candidate for broader adoption in mobility ecosystems.

### 4.5. Statistical Analysis

To complement the descriptive performance assessment, we conducted a statistical analysis of the main metrics throughput, latency (p95 and p99), and resource utilization for both the baseline (monolithic) and orchestrated (Kubernetes) deployments.

Each scenario was executed five independent times under identical load conditions. Gatling logs were exported and aggregated, and for each metric, we computed the mean, standard deviation (SD), and 95 % confidence interval (CI) using the Student's $t$ distribution. For example, peak throughput in the orchestrated deployment averaged 2100 req/s (SD = 95) with a 95 % CI of [2012 req/s; 2188 req/s]; this can be compared to the baseline's 480 req/s (SD = 40; 95 % CI [447 req/s; 513 req/s]).

A two-sample $t$-test assuming unequal variances was applied to the main performance metrics. Results confirmed that the improvements observed are statistically significant: throughput ($t(6.21) = 28.7, p < 0.001$), latency at p99 ($t(7.04) = -9.2, p < 0.001$), and idle memory usage ($t(5.88) = -11.3, p < 0.001$). These tests support the conclusion that the reported 300 % throughput increase is not due to random variation.

Table 6 now includes SD values (in parentheses), and Figures 11 and 12 have been updated to display error bars representing 95 % confidence intervals for each measurement point. Overall, this statistical validation strengthens the claim that the orchestrated architecture provides a robust and reproducible performance gain over the monolithic deployment, reducing the likelihood that the observed differences are attributable to experimental noise or outlier runs.

## 5. Conclusions

This study presented a validated, container-native framework for secure Identity and Access Management (IAM) in green mobility ecosystems. By leveraging Keycloak within a Kubernetes-orchestrated environment, the proposed architecture demonstrated clear improvements in resilience, scalability, and operational efficiency compared to monolithic deployments. The solution integrates Zero Trust and defense-in-depth principles, applies layered security across identity, access, and observability components, and was empirically validated through web, mobile, and IoT use cases. Experimental results showed more than 300% improved peak throughput, a 75% reduction in idle resource usage, and sub-minute automated recovery from failures, establishing the framework as a viable blueprint for secure IAM in MaaS and ITS contexts. These findings contribute both practical insights for mobility operators and methodological advances for researchers studying IAM performance in IoT-centric environments.

### *5.1. Future Work*

While the prototype is capable of meeting current demands related to green mobility, several research-and-development opportunities can further enhance its maturity.

#### 5.1.1. Real-World Deployment and Usability Testing

Future work should include benchmarking Keycloak against commercial IAM offerings, such as Auth0 and Okta, as well as decentralized identity models. The system should also be deployed and evaluated in real-world urban mobility scenarios, taking into account challenges such as network instability, latency variations, and diverse user patterns. In addition, gathering qualitative feedback from operational stakeholders would help refine usability and improve adoption. Finally, future iterations should incorporate hypothesis testing and the calculation of confidence intervals to statistically validate the significance of the observed performance gains. We also plan to extend the evaluation to other open-source platforms, including Ory Hydra and Zitadel, to provide a broader performance and functionality comparison across IAM alternatives.

#### 5.1.2. Advanced Security Hardening and Resilience

Another key area for future development is the enhancement of security and resilience. This includes performing formal penetration tests and establishing a routine of continuous security audits. It will also be important to expand Istio's role to support dynamic traffic management and implement advanced security policies. Moreover, the design of offline-capable authentication and authorization mechanisms for edge devices could extend the framework's applicability to environments with intermittent connectivity. In addition, future work should incorporate formal security verification of the authentication and authorization protocols. Tools such as ProVerif or Tamarin can model adversarial capabilities and verify essential properties (e.g., secrecy, authenticity, and resistance to replay or impersonation attacks), thereby complementing the empirical benchmarking with rigorous protocol analysis. Equally critical is the establishment of long-term maintenance strategies. Automated CI/CD pipelines, dependency monitoring, and rolling-upgrade policies should be integrated to ensure timely application of security patches, library updates, and configuration fixes without interrupting system availability. This lifecycle-oriented approach will help sustain the security posture of the framework over time.

#### 5.1.3. Enhanced Functionality and Integration

Future enhancements should also address functionality and integration. For example, adaptive authorization mechanisms could be implemented, leveraging context and trust dynamics to improve security granularity. Maintainability could be improved with the use of Helm charts and GitOps automation practices. Finally, the potential of hybrid architectures—linking Keycloak with blockchain-based verifiable credentials—should be explored, particularly for high-security contexts where decentralized trust is desirable.

### *5.2. Limitations*

Although the proposed framework demonstrates significant improvements in scalability, resilience, and resource efficiency, several limitations should be acknowledged. First, the evaluation was conducted in a controlled lab environment using synthetic workloads; production deployments with real-world traffic patterns may reveal additional performance constraints. Second, while the study applied Zero Trust principles, no formal protocol verification was carried out—future work should employ symbolic analysis tools such as ProVerif or Tamarin. Third, the cost analysis was limited to relative resource expenditure; a detailed economic assessment (e.g., cloud-pricing models and energy consumption) is required to fully quantify operational costs. Finally, the prototype reflects a snapshot of specific software versions, and the long-term sustainability of the framework will depend on continuous updates, security patching, and lifecycle management.

Addressing these areas will support the evolution of this framework into a production grade IAM solution for scalable, secure, and adaptive mobility systems.

**Author Contributions:** Software, A.S.; Writing—original draft, A.S.; Writing—review and editing, M.J.C.S.R.; Supervision, F.B., A.R. and M.J.C.S.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was developed under the project A-MoVeR—"Mobilizing Agenda for the Development of Products & Systems towards an Intelligent and Green Mobility", operation n.º 02/C05-i01.01/2022.PC646908627-00000069, approved under the terms of the call n.º 02/C05-i01/2022—Mobilizing Agendas for Business Innovation, financed by European funds provided to Portugal by the Recovery and Resilience Plan (RRP), in the scope of the European Recovery and Resilience Facility (RRF), framed in the Next Generation UE, for the period from 2021–2026.

**Data Availability Statement:** The raw data supporting the conclusions of this article will be made available by the authors upon reasonable request to the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Appendix A. Gatling Load Test Script (Excerpt)

The code excerpt in Listing 1 illustrates the Gatling scenario used to simulate API-to-API traffic and user login events. A complete version of the script can be obtained from the corresponding author upon reasonable request via email.

**Listing 1.** Excerpt from Gatling simulation script.

```
class MobilityIAMSimulation extends Simulation {

  val httpProtocol = http
    .baseUrl("https://iam-test.example.com")
    .acceptHeader("application/json")
    .contentTypeHeader("application/json")

  // Define a scenario for API-to-API authentication
  val apiAuthScenario = scenario("API Authentication Load")
    .repeat(80) {
      exec(
        http("API Token Request")
          .post("/auth/realms/green-mobility/protocol/openid-connect/token")
          .body(StringBody("""{ "client_id": "iot-client",
                               "client_secret": "secret",
                               "grant_type": "client_credentials" }"""))
          .check(status.is(200))
      ).pause(1)
    }

  // Define a scenario for user login events
  val userLoginScenario = scenario("User Login Load")
    .repeat(15) {
      exec(
        http("User Login Request")
          .post("/auth/realms/green-mobility/protocol/openid-connect/token")
          .body(StringBody("""{ "username": "testuser",
                               "password": "testpassword",
                               "grant_type": "password",
                               "client_id": "mobile-app" }"""))
          .check(status.is(200))
      ).pause(2)
    }

  // Set up the load profile
  setUp(
    apiAuthScenario.inject(rampUsers(3200) during (15 min)),
    userLoginScenario.inject(rampUsers(600) during (15 min))
  ).protocols(httpProtocol)
}
```

## References

1. Gonçalves, C.; Sousa, B.; Vuković, M.; Kusek, M. A federated authentication and authorization approach for IoT farming. *Internet Things* **2023**, *22*, 100785. [CrossRef]
2. Sousa, A.; Martins, H.; Khan, D.; Azevedo, J.; Reis, M.J.C.S. Proposal of a Collaborative System to Foster the Concept of Mobility as a Service in the Green Mobility Context. In *Proceedings of the Distributed Computing and Artificial Intelligence, Special Sessions I, 21st International Conference*; Mehmood, R., Hernández, G., Praça, I., Wikarek, J., Loukanova, R., Monteiro dos Reis, A., Skarmeta, A., Lombardi, E., Eds.; Springer: Cham, Switzerland, 2025; pp. 217–225.
3. Cui, Y.; Liu, F.; Jing, X.; Mu, J. Integrating Sensing and Communications for Ubiquitous IoT: Applications, Trends and Challenges. *arXiv* **2021**, arXiv:2104.11457. [CrossRef]
4. Antonialli, F.; Gandia, R.M.; Sugano, J.Y.; Nicolaï, I.; de Miranda Neto, A. Business Platforms for Autonomous Vehicles Within Urban Mobility. *WIT Trans. Built Environ.* **2019**, *186*, 175–186. [CrossRef]
5. Wang, S.; Lehmann, C.; Radeke, R.; Fitzek, F.H.P. Enabling Sustainable Urban Mobility: The Role of 5G Communication in the Mobilities for EU Project. *arXiv* **2024**, arXiv:2412.04006. [CrossRef]
6. Shamsuddoha, M.; Kashem, M.A.; Nasir, T. A Review of Transportation 5.0: Advancing Sustainable Mobility Through Intelligent Technology and Renewable Energy. *Future Transp.* **2025**, *5*, 8. [CrossRef]
7. Mitropoulos, L.; Kortsari, A.; Mizaras, V.; Ayfantopoulou, G. Mobility as a Service (MaaS) Planning and Implementation: Challenges and Lessons Learned. *Future Transp.* **2023**, *3*, 498–518. [CrossRef]
8. Santos, G.; Nikolaev, N. Mobility as a Service and Public Transport: A Rapid Literature Review and the Case of Moovit. *Sustainability* **2021**, *13*, 3666. [CrossRef]
9. Derawi, M.; Dalveren, Y.; Cheikh, F.A. Internet-of-Things-Based Smart Transportation Systems for Safer Roads. In Proceedings of the 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2–16 June 2020; pp. 1–4. [CrossRef]
10. Ekpo, O.; Casola, V.; De Benedictis, A. Security and Privacy Issues in Mobility-as-a-Service (MaaS): A Systematic Review. In Proceedings of the 2024 19th Annual System of Systems Engineering Conference (SoSE), Tacoma, WA, USA, 23–26 June 2024; pp. 300–307. [CrossRef]
11. Garroussi, Z.; Legrain, A.; Gambs, S.; Gautrais, V.; Sansò, B. Data privacy for Mobility as a Service. *arXiv* **2023**, arXiv:2310.10663. [CrossRef]

12. Chatterjee, A.; Prinz, A. Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study. *Sensors* **2022**, *22*, 1703. [CrossRef] [PubMed]
13. Vayghan, L.; Saied, M.; Toeroe, M.; Khendek, F. A Kubernetes controller for managing the availability of elastic microservice based stateful applications. *J. Syst. Softw.* **2021**, *175*, 110924. [CrossRef]
14. Yasrab, R. Mitigating Docker Security Issues. *arXiv* **2023**, arXiv:1804.05039. [CrossRef]
15. Badii, C.; Bellini, P.; Difino, A.; Nesi, P. Sii-Mobility: An IoT/IoE Architecture to Enhance Smart City Mobility and Transportation Services. *Sensors* **2019**, *19*, 1. [CrossRef]
16. Esposito, C.; Ficco, M.; Gupta, B.B. Blockchain-based authentication and authorization for smart city applications. *Inf. Process. Manag.* **2021**, *58*, 102468. [CrossRef]
17. The Skycloak Team. How IoT Devices are Revolutionizing Identity Management. Available online: https://skycloak.io/blog/how-iot-devices-are-revolutionizing-identity-management/ (accessed on 27 June 2025).
18. Jain, P. Identity and Access Management in the Cloud. *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2025**, *11*, 1528–1535. [CrossRef]
19. Rajba, P.; Orzechowski, N.; Rzepka, K.; Szary, P.; Nastaj, D.; Cabaj, K. Identity and Access Management Architecture in the SILVANUS Project. In Proceedings of the 19th International Conference on Availability, Reliability and Security, New York, NY, USA, 30 July–2 August 2024. [CrossRef]
20. Thakur, M.A.; Gaikwad, R. User identity and Access Management trends in IT infrastructure- an overview. In Proceedings of the 2015 International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015; pp. 1–4. [CrossRef]
21. Sharma, D.H.; Dhote, C.; Potey, M.M. Identity and Access Management as Security-as-a-Service from Clouds. *Procedia Comput. Sci.* **2016**, *79*, 170–174. [CrossRef]
22. Indu, I.; Anand, P.R.; Bhaskar, V. Identity and access management in cloud environment: Mechanisms and challenges. *Eng. Sci. Technol. Int. J.* **2018**, *21*, 574–588. [CrossRef]
23. Singh, C.; Thakkar, R.; Warraich, J. IAM Identity Access Management—Importance in Maintaining Security Systems within Organizations. *Eur. J. Eng. Technol. Res.* **2023**, *8*, 30–38. [CrossRef]
24. TekSlate. Okta vs. Auth0: Detailed Comparison. Available online: https://tekslate.com/okta-vs-auth0 (accessed on 19 December 2024).
25. Okta. Okta: Identity for the Internet. Available online: https://www.okta.com/ (accessed on 20 December 2024).
26. Gluu. Gluu—Identity and Access Management. 2024. Available online: https://gluu.org/?utm_source=saasworthy.com&utm_medium=cpc&utm_banner=2998550&sub1=2998550&visitorid=2998550 (accessed on 20 December 2024).
27. Software, D. IdentityServer—Secure your Apps and APIs. Available online: https://duendesoftware.com/products/identityserver (accessed on 21 December 2024).
28. Sersemis, A.; Papadopoulos, A.; Spanos, G.; Lalas, A.; Votis, K.; Tzovaras, D. A Novel Cybersecurity Architecture for IoV Communication. In Proceedings of the 25th Pan-Hellenic Conference on Informatics, Volos, Greece, 26–28 November 2021; pp. 357–361. [CrossRef]
29. Guija, D.; Siddiqui, M.S. Identity and Access Control for micro-services based 5G NFV platforms. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018. [CrossRef]
30. Al Rahat, T.; Feng, Y.; Tian, Y. AuthSaber: Automated Safety Verification of OpenID Connect Programs. In Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, Salt Lake City, UT, USA, 14–18 October 2024; pp. 2949–2962. [CrossRef]
31. James, M.; Newe, T.; O'Shea, D.; O'Mahony, G.D. Authentication and Authorization in Zero Trust IoT: A Survey. In Proceedings of the 2024 35th Irish Signals and Systems Conference (ISSC), Belfast, UK, 13–14 June 2024; pp. 1–7. [CrossRef]
32. Amiri, M.J.; Agrawal, D.; El Abbadi, A. Permissioned Blockchains: Properties, Techniques and Applications. In Proceedings of the 2021 International Conference on Management of Data, Virtual, 20–25 June 2021; pp. 2813–2820. [CrossRef]
33. Sciullo, L.; De Marchi, A.; Gigli, L.; Palmirani, M.; Vitali, F. AAA: A blockchain-based architecture for ethical, robust authenticated anonymity. In Proceedings of the 2024 International Conference on Information Technology for Social Good, Bremen, Germany, 4–6 September 2024; p. 1. [CrossRef]
34. Jin, X.; Omote, K. An efficient blockchain-based authentication scheme with transferability. *PLoS ONE* **2024**, *19*, e0310094. [CrossRef] [PubMed]
35. Alilwit, N. Authentication Based on Blockchain. Ph.D. Thesis, Embry-Riddle Aeronautical University, Daytona Beach, FL, USA, 2020.
36. Thakur, M. Authentication, Authorization and Accounting with Ethereum Blockchain, 2017. URN:NBN:fi:hulib-201711145693. Available online: https://helda.helsinki.fi/items/8d0dad04-07f5-4a0d-9ed7-64db26b1e960 (accessed on 3 February 2024).
37. Kubernetes Documentation. Kubernetes Concepts Overview. 2024. Available online: https://kubernetes.io/docs/concepts/overview/ (accessed on 3 February 2024).

38. Wong, A.Y.; Chekole, E.G.; Ochoa, M.; Zhou, J. Threat Modeling and Security Analysis of Containers: A Survey. *arXiv* **2021**, arXiv:2111.11475. [CrossRef]
39. MaaS Alliance. *MaaS Market Playbook*; MaaS Alliance: Brussels, Belgium, 2021. Available online: https://maas-alliance.eu/wp-content/uploads/sites/7/2021/03/05-MaaS-Alliance-Playbook-FINAL.pdf (accessed on 3 February 2024).
40. Shrestha, R.; Bertin, E. Access Control for IoT: A Survey of Existing Research, Dynamic Policies and Future Directions. *Sensors* **2023**, *23*, 1805. [CrossRef]
41. Kampa, S. Navigating the Landscape of Kubernetes Security Threats and Challenges. *J. Knowl. Learn. Sci. Technol.* **2024**, *3*, 274–281. [CrossRef]
42. Mullinix, S.P.; Konomi, E.; Townsend, R.D.; Parizi, R.M. On Security Measures for Containerized Applications Imaged with Docker. *arXiv* **2020**, arXiv:2008.04814. [CrossRef]
43. Subramanian, N.; Jeyaraj, A. Recent security challenges in cloud computing. *Comput. Electr. Eng.* **2018**, *71*, 28–42. [CrossRef]