

Design and Validation of an Agentless Web Application for IT Infrastructure Operations Management

Carlos Otárola-Mermau*, Alex Pacheco-Pumaleque

School of Systems Engineering, Universidad César Vallejo, Trujillo, Perú

Abstract—The manual management of information technology (IT) infrastructure operations at an IT services company in Lima, Peru, was characterized by fragmented procedures, delays, and limited traceability, which hindered service continuity and compromised information security. This study presents the design, development, and validation of an on-premises web application for the centralized management of IT infrastructure operations, evaluated across three dimensions: operational efficiency, execution effectiveness, and security with auditing. Built on a layered architecture, the solution employs Ansible as an automation engine that executes specific tasks on Linux, Windows, and VMware servers for diverse operations, without installing agents on the managed hosts. Access control is performed against Active Directory using roles that separate operations from administration. A pre-test/post-test quasi-experimental design compared the consolidated version of the application with its pilot version in a controlled test environment replicating the corporate infrastructure. The time reduction index increased by 41 percentage points, the operation success rate rose by 24 points, and the traceability level improved by 39 points, with Wilcoxon signed-rank tests confirming significance and large Cohen's effect sizes. The results confirm the viability of the solution and provide replicable evidence for the modernization of enterprise IT operations.

Keywords—Web application; infrastructure orchestration; agentless automation; digital transformation

I. INTRODUCTION

The growing adoption of web solutions by organizations has become a key factor in the modernization of their services. Over the past decade, web applications have proven to be platforms capable of centralizing data and executing complex processes remotely and securely [1]. The interaction between software and technological infrastructure is now recognized as an essential component for ensuring business continuity; accordingly, the adoption of specific digital platforms has transformed operations management, moving from purely reactive approaches to strategic and proactive models [2].

Globally, information technology-related outages have been linked to human error and to ignored or inadequate procedures; nearly 40 percent of organizations have experienced outages caused by human error in the past three years [3]. Furthermore, the lack of centralized management tools increases the vulnerability of critical systems to operational failures [4]. In this scenario, digital transformation faces obstacles arising from the fragmentation of IT infrastructure and the slow adoption of

integrated platforms, leading to recurring difficulties related to the lack of interoperability and the overloading of technical support channels.

From a theoretical perspective, process systematization is a key element for understanding how organizations succeed in transforming manual operations into structured, reproducible, and measurable workflows [5]. Along the same lines, sociotechnical systems theory holds that successful technology implementation requires simultaneous consideration of the technical and social subsystems, since their interaction determines the sustainability of organizational change [6]. Innovation theory, for its part, emphasizes that cloud computing, artificial intelligence, integrated platforms, and web applications act as transformation vectors with a significant impact on operational efficiency [7].

However, although technological solutions designed for IT service management exist, many lack the flexibility required to adapt to the administrative and regulatory workflows typical of corporate environments [8]. At a technology services company in Lima, Peru, the infrastructure department faced limitations stemming from the shortage of specialized personnel, the absence of clear policies, and deficiencies in process automation, which resulted in delays in task execution and in insufficient or nonexistent auditing mechanisms. Based on the analysis of the background and the theoretical foundations reviewed, the hypothesis was formulated that the implementation of a web application for IT infrastructure operations management would positively influence the organization's operational performance. Specifically, it was proposed that the application would improve time reduction rates, increase the task execution success rate, and optimize the level of event traceability.

In this context, the scientific contribution of this study is summarized in three contributions. First, an agentless orchestration architecture is proposed that centralizes the administration of heterogeneous IT infrastructure operations (Windows, Linux, and VMware servers) without requiring the installation of agents on the managed nodes, thereby reducing the attack surface and the administrative burden compared with agent-based approaches. Second, a traceability model—with audit logging per user, task, and host—is integrated into a single platform together with LDAP/Active Directory corporate authentication and role-based access control, capabilities that individual automation tools do not natively incorporate. Third, although some related works report quantitative improvements

*Corresponding author.

(Table I), these are limited to descriptive comparisons of indicators; this study additionally provides a quasi-experimental validation with inferential testing (Wilcoxon signed-rank test) and large effect sizes across the three indicators (r between 0.785 and 0.873, above Cohen's 0.5 threshold), which makes it possible to assess both the statistical significance and the practical relevance of the improvements in reducing response time, increasing the success rate, and enhancing traceability. Taken together, the contribution lies not in the individual tools—most of them open source—but in demonstrating that their orchestration under a coherent web architecture produces measurable and statistically significant operational improvements.

II. RELATED WORK

Various studies show that process automation through web solutions significantly increases operational efficiency in business environments. For IT infrastructure monitoring, centralized platforms based on Prometheus and Grafana have been shown to provide real-time visibility of operational status, thereby reducing incident detection times and improving system availability [9]. Likewise, the adoption of service management frameworks such as FitSM, ITIL, COBIT, and ISO/IEC 20000 has been associated with measurable improvements in service quality and organizational operational efficiency [10]. Similarly, studies on intelligent automation platforms report incident resolutions up to 50% faster than traditional methods, attributable to the orchestration of repetitive tasks through automated workflows [11]. In the Peruvian mining industry, for example, web-based support systems have raised the incident resolution rate from 52% to 88% through ticket tracking and service level agreement monitoring [12]. Similarly, the digitalization of ITIL-based support services has improved management indicators, raising them from 55% to 82% following the adoption of digital processes [13].

Recent literature also highlights the advantages of agentless automation tools. In particular, research on Ansible demonstrates its ability to manage infrastructure through a declarative approach, which simplifies the management of IT configurations and operations, reducing costs and operational impacts [14], [15], [16]. However, these studies also show that, when operated independently from the command line, its use remains limited to specialized personnel, without integrated governance, traceability, or access control mechanisms for the rest of the organization.

Regarding access control, various studies have demonstrated the effectiveness of role-based models implemented with LDAP, as they enable scalable security and guarantee the functional separation between operational and administrative profiles [17], [18]. Table I presents a comparative analysis of the existing models and the proposed model.

Despite these advances, significant limitations persist. Consolidated orchestration platforms such as Ansible AWX/Tower [19] and Rundeck [20] offer centralized task execution but are centered on playbook management and scheduled jobs; they offer neither an integrated template catalog with categories and role-based separation of duties, nor a traceability model linking user, task, host, and execution output in a single auditable record, nor native hardware inventory

modules with change detection. For their part, agent-based configuration management tools such as Puppet, Chef, and SaltStack [21] require deploying and maintaining persistent software on each managed node, which increases the operational burden and demands securing an additional component on every host. Moreover, many of these solutions are designed for cloud-native architectures, which conflicts with data sovereignty requirements in on-premises environments. In light of these gaps, this study proposes an on-premises web application that integrates, into a single platform, agentless orchestration governed by a template catalog, auditable traceability, and centralized corporate authentication.

TABLE I. ANALYSIS OF THE PROPOSED MODEL AND RELATED WORK

Reference	Technology / Framework	Domain	Key benefits
Trikusuma et al., 2023 [9]	Grafana + Prometheus	IT Monitoring	Real-time visibility; reduction in Mean Time to Detection
Sarwar et al., 2023 [10]	FitSM, ITIL, COBIT, ISO/IEC 20000	Enterprise IT	Improves service quality and governance
Nugroho, 2022 [11]	Intelligent Automation Platforms	Enterprise IT	Results up to ~50% faster than manual methods
S. Perez and F. Mendoza, 2024 [12]	Web-Based Help Desk System	Mining company	Resolution rate: 52% → 88% through ticket tracking
Sánchez & Valles, 2021 [13]	ITIL v3 Implementation	Enterprise IT	Improved management indicators: 55% → 82%
Proposed Model, 2026	Web-based IT operations management platform (Flask, Ansible, LDAP, RBAC)	Enterprise IT	+41% reduction in time; +23.99% success rate; +39.17% traceability

III. METHODOLOGY

A. Research Type and Design

This is applied research, since its purpose is to solve a specific and verifiable problem: the fragmented, slow, and poorly traceable management of IT infrastructure operations at an IT services company, through the design and implementation of a software solution. The study was framed within the Design Science Research methodology, which establishes the construction of a technological artifact relevant to an organizational problem and its evaluation to demonstrate its usefulness [22]. The analysis is developed under a quantitative approach, using measurable and comparable indicators that allow an objective comparison between the initial condition and the one obtained after the intervention.

The design is quasi-experimental, with pre-test and post-test measurements applied to the same operational processes, without random assignment of units—as befits a business context where procedures and assets are predetermined [23]. The unit of analysis is IT infrastructure operations (for example, a password change on a group of servers, an NTP configuration, or security updates on a group of virtual machines); each operation was executed first with the pilot version of the application and then with the consolidated version. The sample

was census-based: the 30 operations comprising the infrastructure department's catalog of representative tasks were included, which eliminates selection bias. This within-subject comparison design reduces the variability attributable to differences between processes and isolates the effect of the technological intervention on the study's three indicators. The design is consistent with previous studies that evaluated web systems through before-after comparisons and reported significant improvements in resolution rates and service quality [12], [13]. To mitigate threats to internal validity, records were obtained directly from the system database, thereby avoiding manual transcription and observer bias.

The choice of a single-group design is grounded in operational feasibility criteria: in a real business environment, it is not possible to maintain two identical infrastructures operating in parallel, nor to subject the same assets to differentiated procedures simultaneously. Likewise, given the sample size and the non-normality of the data, the Wilcoxon signed-rank test was used, which is appropriate for small samples and non-normal distributions. Statistical analysis was performed with SPSS Statistics v25, applying the Shapiro-Wilk test to verify normality and the Wilcoxon signed-rank test with a significance level of 0.05; effect sizes were calculated using $r = Z/\sqrt{N}$ and interpreted according to Cohen's criteria.

B. Variables and Indicators

The independent variable is the web application, defined as the technological tool that centralizes and automates operations. The dependent variable is the management of IT infrastructure operations, operationalized in three dimensions, each with a quantitative indicator and a calculation formula, according to the study's consistency matrix. The operational efficiency dimension is measured by the response time reduction index (Ir), which expresses, as a percentage, the additional reduction in execution time achieved when moving from the pilot version of the application to its consolidated version. A higher value indicates a greater time reduction relative to the reference manual procedure. The execution effectiveness dimension is measured by the operation success rate (TasaE), which compares the number of completed operations with the total number of operations initiated, capturing the system's reliability in the face of partial failures or unreachable hosts. The security and auditing dimension is measured by the level of operational event traceability (NivelT), which quantifies the proportion of operations that generated a corresponding auditable record, reflecting the system's capacity to support subsequent control and accountability.

The three indicators are expressed on a ratio scale (0–100%) and are calculated using the formulas in Table II. Their selection is based on classifications of IT infrastructure performance indicators, which group the relevant metrics into availability, performance, security, and capacity, and on the concept of operational maturity as a key element for assessing the progress of digital transformation in the public and private sectors [24], [25]. It should be noted that the baseline measurements, recorded with the pilot version of the application, reflect the operating conditions prior to the intervention, which already included ad hoc scripts and partial command-line automation routines.

In this way, the model integrates the three dimensions into a single quantitative framework that can be replicated by other companies.

TABLE II. DIMENSIONS, INDICATORS, AND FORMULAS FOR THE DEPENDENT VARIABLE

Dimension	Indicator	Formula
Operational Efficiency	Time Reduction Index (Ir)	$Ir = ((T_{\text{manual}} - T_{\text{auto}}) / T_{\text{manual}}) \times 100^a$
Execution Effectiveness	Operation Success Rate (TasaE)	$TasaE = (\text{CompletedOperations} / \text{TotalOperations}) \times 100$
Security and Auditing	Traceability Level (NivelT)	$NivelT = (\text{Log_Entries} / \text{Operations}) \times 100$

T_{manual}: time for the reference manual procedure; T_{auto}: time recorded by the application version corresponding to each phase (pilot in the pre-test, consolidated in the post-test).

C. Technology Selection

The selection of the technology stack was based on maturity and interoperability criteria, prioritizing components reliable for enterprise environments. The decision to implement a layered architecture was made on the principle of separation of concerns, which improved testability, facilitated the independent horizontal scalability of each layer, and reduced coupling between components [26]. Python was chosen as the main development language because it is a multi-paradigm language, with a moderate learning curve and an extensive standard library that facilitates integration with automation engines, network protocols, and database management systems [27]. Flask was adopted as the web framework for its lightweight core and modular extensions, which allow building scalable applications without imposing a rigid architecture [28]. Its support for the Model-View-Controller (MVC) pattern facilitates the separation between presentation, business logic, and data access [29], which is key for applications that require long-term maintainability. Flask also seamlessly integrates template rendering, database abstraction, and efficient HTTP request handling, accelerating development without sacrificing control over application behavior.

Ansible Core was chosen as the agentless automation engine for its declarative architecture, its ability to orchestrate tasks across multiple nodes simultaneously, and its native integration with Python through the ansible-runner library [30]. Unlike tools that require the installation of persistent agents on each managed node, it leverages protocols native to the operating systems themselves: Secure Shell (SSH), available by default on Linux distributions, and Windows Remote Management (WinRM), built into Microsoft Windows. By not requiring additional software on the managed nodes, this approach greatly simplifies initial deployment. The ecosystem was complemented with specialized libraries for data access, programmatic playbook execution, corporate authentication, vSphere integration, and credential protection, detailed in Table III.

Finally, PostgreSQL was selected as the relational database management system for its compliance with SQL standards, its native support for the JSONB data type—which allows storing semistructured data without sacrificing referential integrity—as well as its logical replication engine for high availability [31].

TABLE III. COMPLEMENTARY LIBRARIES OF THE TECHNOLOGY STACK

Component	Library	Purpose
Data access	SQLAlchemy 2 (ORM)	Database access layer
Automation engine	ansible-runner 2.4	Programmatic playbook execution
Corporate authentication	ldap3 2.9	Active Directory bind over LDAP
Virtualization	pyvmomi 8	VMware vSphere API integration
Credential encryption	cryptography 42 (Fernet AES-128-CBC)	Encryption of stored credentials
Password hashing	passlib (bcrypt)	Hashing of local passwords

D. Development Methodology

The web application was developed with an iterative and incremental approach, with progressive cycles adapted to the changing needs of the infrastructure. A hybrid methodology was adopted that combined the agile practices of Scrum—iterative sprints, a prioritized backlog, and retrospectives—with the risk management of the Rational Unified Process (RUP) [32].

The choice of this hybrid approach responded to three factors: the need for frequent testing cycles arising from the multiple integrations with heterogeneous systems (Active Directory, VMware vSphere, Linux, and Windows); the requirement for detailed technical documentation characteristic of an on-premises environment subject to audits, which purely agile methods tend to underestimate; and operation with limited resources, which made the full process overhead of RUP impracticable.

The backlog consisted of ten user stories estimated at 81 story points, distributed across six sprints: the first dedicated to the authentication core, and the remaining five to playbook execution, server management, the execution wizard, auditing, VMware integration, the playbook editor, internationalization, and exportable reports. Each increment was validated through integration testing against the laboratory environment servers, executing real operations through Ansible as the automation engine before acceptance.

The development cycle was organized into six phases; Fig. 1 shows the development flowchart.

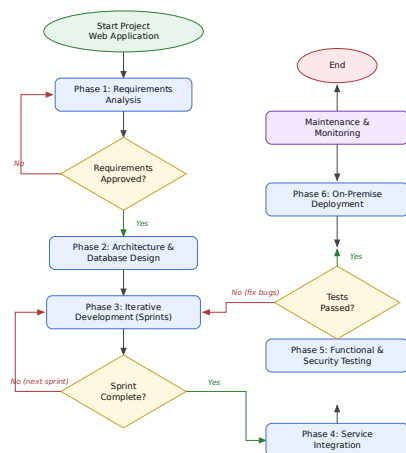


Fig. 1. Development flowchart.

1) *Requirements analysis*: Interviews Were Conducted with System Administrators and IT Operators to Identify Critical Manual Processes Amenable to Automation. The information was consolidated into a Catalog of Ten Functional Requirements and the Associated Non-functional Requirements—security, Performance, Auditability, and Usability—prioritized according to Their Operational Value and Technical Complexity, Which Constituted the Baseline Backlog for Development.

2) *Architecture and database design*: A Decoupled Layered Architecture—presentation, Application, Services, and Data—was defined, Which Allowed a Clear Distribution of Responsibilities and Improved System Maintainability; the Application Layer Was Organized into Independent Blueprints per Functional Domain (authentication, Operations, Administration, and REST API). the Database Model Was Designed Following Normalization Principles up to the Third Normal Form, Guaranteeing Referential Integrity and Minimizing Redundancy, with a Cross-cutting Auditing Component that Records Every Executed Operation.

3) *Iterative development*: The Project Was Structured into Six Two-week Sprints, according to the Thematic Distribution of the Backlog Described Above. Each Sprint Included Planning, Development, Unit Testing with Pytest, and Increment Review with Stakeholders, Whose Findings Fed Back into the Backlog of the Following Sprint.

4) *Service integration*: The Interoperability of the Application with the Environment's External Systems Was Consolidated: Centralized Authentication against Active Directory via LDAP, Infrastructure Inventory Management through the VMware vSphere API, and Programmatic Execution of Ansible Playbooks on the Managed Nodes. Each Integration Was Verified with Tests against the Laboratory Servers before Acceptance of the Corresponding Sprint.

5) *Functional and security testing*: The Acceptance Criteria of Each Module Were Verified, and a Security Testing Program Was Executed that Included: Brute-force Resistance on Authentication, with Account Lockout after Five Attempts; Privilege Escalation between Roles, Rejected with HTTP 403; Vulnerability Scanning with OWASP ZAP and SQL Injection Detection with SQLmap, with No Exploitable Findings; and Concurrent Load Testing with Locust. Additionally, Session Management (timeout, Secure Cookies) and At-rest Credential Encryption Were Validated.

6) *On-premises deployment*: The Application Was Deployed on Ubuntu Server 22.04 LTS as a Systemd Service, with Gunicorn 22 as the WSGI Server—with Workers Adjusted According to the Available vCPUs ($2 \times \text{vCPU} + 1$)—and Nginx as a Reverse Proxy for TLS Termination. Deployment Was Automated through Bash Scripts that Install Dependencies, Synchronize the Catalog Playbooks, and Restart the Service, with Database Backup and Log Rotation, Guaranteeing Reproducible Deployments.

E. System Architecture

The application was designed as an on-premises system structured in decoupled layers, deployed entirely within the company's internal network. The following description outlines the architecture from the user interface to persistent storage, reflecting the organization of the code in its current version; this architecture is illustrated in Fig. 2.

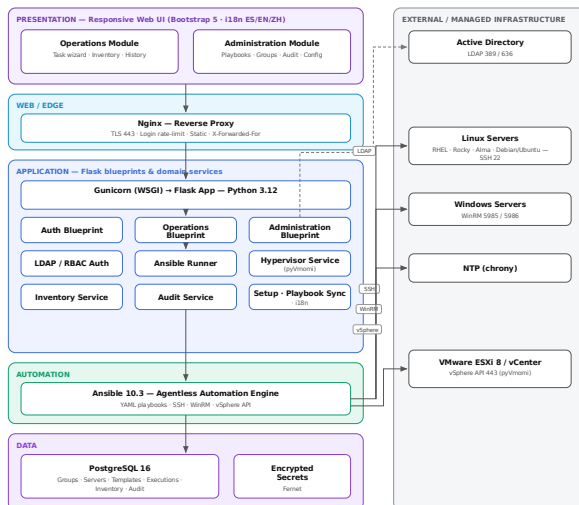


Fig. 2. System architecture.

1) *Presentation layer*: The Responsive Web Interface, Built with Bootstrap 5, Supports Multiple Languages (Spanish, English, and Chinese) and Strictly Separates the Operations and Administration Modules.

2) *Web or edge layer*: Nginx Acts as a Reverse Proxy and Single Entry Point. It handles TLS Encryption, Serves Static Files, Limits Login Attempts to Mitigate Brute-force Attacks, and forwards the Real Client Headers to Record IP Addresses for Auditing Purposes.

3) *Application layer*: Implemented in Python 3.12 with Flask and served by Gunicorn Behind Nginx. The logic is organized into Four Blueprints (authentication, Operations, Administration, and a REST API for External Integrations) and Several Specialized Services: LDAP/RBAC Authentication, Ansible Execution, VMware Hypervisor Management, Hardware Inventory, Event Auditing, Server Provisioning, Playbook Synchronization, and Internationalization. This Separation Facilitates System Extension and Maintenance.

4) *Automation layer*: Ansible, in Agentless Mode, Executes Declarative Playbooks on the Managed Hosts through the Native Protocols of Each Platform. The dynamic inventory is built from the Selected Servers and the Operator's Parameters, with Secure Credential Injection When Necessary.

5) *Data layer*: PostgreSQL Persistently Stores the System's Catalog, Executions, Inventories, and Audit Records; Sensitive Credentials Are Encrypted Using Fernet. External Integrations (Active Directory, Linux/Windows Servers, NTP, and VMware ESXi/vCenter) Remain Outside the Application and Are Accessible Only from the Internal Network, Preserving the On-premises Nature of the System.

F. Data Model

The relational model presented in Fig. 3 organizes the information around the entities that directly support the study's three indicators. Hypervisors constitute the root of the hierarchy: server groups (server_groups) and individual servers (servers) depend on them; the groups are composed of hosts homogeneous by platform and act as the target of operations.

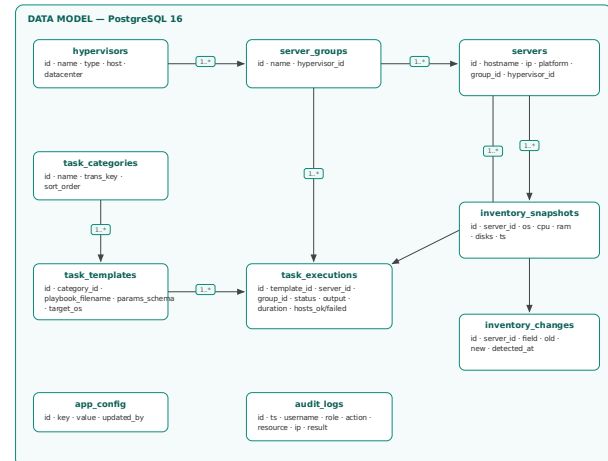


Fig. 3. Entity-relationship model of the PostgreSQL database.

For its part, the task catalog is structured into categories (task_categories) that contain templates (task_templates). Each template references a playbook, its parameter schema, and the target operating system, and must belong to a category to appear in the operations catalog. As for the transactional core, it consists of executions (task_executions): each record links the executed template, the target server or group, and the final status, storing the complete Ansible output, the duration, and the host metrics (ok, failed, unreachable) together with the parameters used. The indicators are naturally derived from this entity: time reduction index, success rate, and—together with the audit log (audit_logs)—the traceability level.

Finally, the inventory module is embodied in the snapshots (inventory_snapshots) and the detected changes (inventory_changes), both associated with the server, while the system configuration (app_config) consolidates the general parameters. This organization guarantees that every executed action leaves a structured, searchable trail—an essential requirement for subsequent auditing.

G. Authentication and Access Control

Access is authenticated against Active Directory using LDAP, avoiding credential duplication and leveraging the existing corporate identity. After authentication, the directory service retrieves the user's groups and applies role-based access control (RBAC): membership in the operators group enables the operations module, while membership in a specific administrators group (for example, group_admin_web) enables the administration module. Users without an authorized group are denied access, and the event is recorded as denied access (LOGIN_DENIED). As an additional safeguard against automated access attempts, the form incorporates the ALTCHA captcha—locally operated, based on proof of work—whose challenge is generated and verified with an HMAC key owned

by the application, without calls to external services, thus preserving the on-premises nature of the solution.

H. Operations Module and Task Catalogue

The operations module presents a three-step wizard—category, task template, and parameters/target group—that guides the operator without requiring command-line knowledge (Fig. 4). Form parameters are automatically transferred to Ansible, including the secure injection of hypervisor credentials when the operation involves VMware. The catalog is organized into five categories—password management, service configuration, virtualization, inventory, and maintenance—and includes, among others, validated tasks for password changes on Linux and Windows, time protocol configuration, on-demand restart of virtual machines on ESXi 8 and vCenter, hardware inventory collection, and operating system and SSL certificate updates. Operations requiring an irreversible step request explicit operator confirmation. The catalog is extensible: the administrator can incorporate new playbooks from the administration module, which become available to operators after validation. A read-only view of the hardware inventory was also incorporated into this module, allowing the operator to see the most recently collected status for each server without accessing the administration module.

I. Orchestration with Ansible and Traceability

When a task is confirmed, the system creates an execution record, logs the event in the audit log, and starts, on an asynchronous thread, the dynamic construction of the Ansible inventory from the hosts of the selected group; the client IP address is captured via ProxyFix and attached to the audited event. The engine is then invoked; upon completion, a PLAY RECAP parser extracts the host metrics (ok, failed, unreachable), determines the final status (success or failure), and persists the complete output, duration, and parameters. The interface allows viewing the complete Ansible output of each execution, which facilitates troubleshooting without having to access the server console. The inventory service analyzes the output of the collection playbooks and generates hardware snapshots—with change detection against the previous snapshot—that feed both the operator view and the administration dashboard.

J. Administration Module

The administration module—which is independent of the operations module and accessible only to the administrators group—organizes its functions in a collapsible accordion-style side menu; from there, YAML playbooks are edited with syntax validation, and—as a catalog governance requirement—both uploading a new playbook and editing an existing one require selecting one of the five available categories (password management, service configuration, virtualization, inventory, and maintenance), so that the resulting template appears classified under “Task Templates.” In addition, the module allows managing server groups and hosts by platform—including provisioning and automatic configuration of initial access via SSH/sudo—as well as managing templates and hypervisors and viewing inventory and audit reports; the latter incorporates a per-user filter and, like the inventory and execution history views, is displayed in paginated format. Finally, every relevant administrative action is recorded in the

audit log, guaranteeing traceability and control over the operations performed.

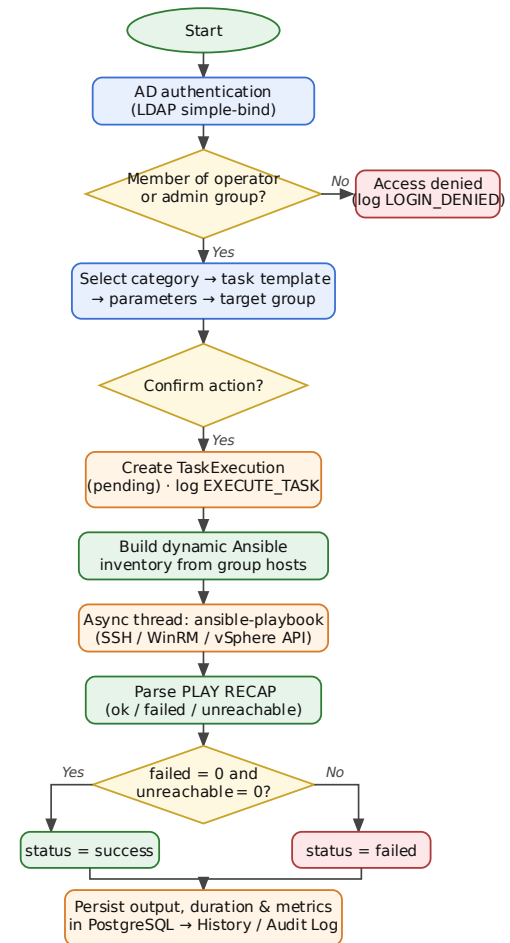


Fig. 4. Operator task execution flow.

K. Functional Validation Procedure

Since the project focuses on enterprise IT modernization, validation was performed in a controlled environment that replicated the target infrastructure using test virtual machines and development environments. The laboratory environment was deployed on VMware Workstation on a workstation with 24 CPU cores, 64 GB of RAM, and 2 TB of storage. The topology included ten virtual machines interconnected through an isolated internal virtual network: Linux servers (Oracle Linux, Ubuntu Server, and Debian), Windows Server machines, a VMware ESXi 8 host nested in standalone mode, and the virtual machine that hosted the web application. For each operation, two execution times were recorded: on the one hand, the execution time of the reference manual procedure—documented with IT personnel—and, on the other hand, the execution time automatically recorded by the web application in its version corresponding to each phase, stored directly in its database, which guaranteed consistent measurements free of transcription errors. In addition, the number of completed operations relative to the total initiated was counted, and the proportion of events in the audit log was verified. Data collection was supported by structured observation guides per indicator—one per phase, pre-test and post-test. The metrics

were calculated using the formulas in Table II, guaranteeing data traceability from the source to the indicator.

It should be noted that this was a laboratory environment and not the production infrastructure. This decision reflects security and operational ethics criteria—running experimental procedures on production systems would have put service continuity at risk—and is common practice in the evaluation of infrastructure automation systems. The environment faithfully replicated production conditions (same operating system distributions, same network topology, and same procedures documented with IT personnel), so the results are transferable to the real context, although this transferability is analyzed as a threat to external validity in the limitations section.

L. Security Considerations

Security was addressed comprehensively. Hypervisor credentials are stored encrypted using Fernet, and sensitive passwords are omitted from logs (no_log) to prevent them from leaking into log outputs. Communication with hosts uses SSH with a private key for Linux and WinRM for Windows, while the vSphere API is invoked over port 443. The number of concurrent executions is limited by a configurable semaphore that protects server resources. At the perimeter, Nginx handles TLS termination and limits the rate of login attempts; the form validates an ALTCHA captcha based on a proof-of-work mechanism—verified locally, without external services—and RBAC strictly separates operational and administrative privileges. Finally, for external connections, user access is routed through a VPN, guaranteeing that the application always operates within the internal network and is never directly exposed to the Internet.

IV. RESULTS

A. Developed System

A fully functional on-premises web application was delivered that centralizes the management of IT infrastructure operations. The system integrates the two planned modules—operations and administration—as well as authentication against Active Directory with role-based access control, a catalog of automated tasks (Fig. 5), a hardware inventory module, and a comprehensive audit log. The administration dashboard consolidates in a single view the number of managed groups and servers, the available templates, the overall success rate, and the latest executions and audit events, eliminating the need to switch between disparate tools (Fig. 6).

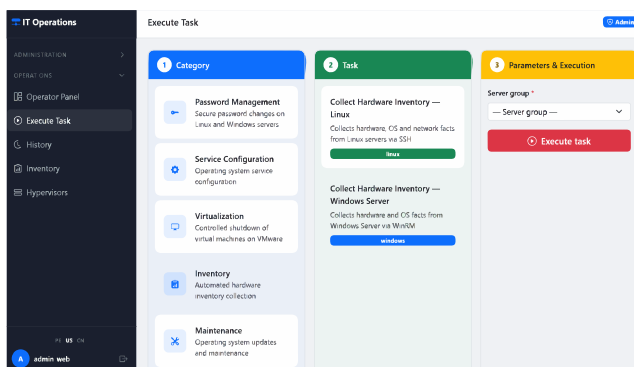


Fig. 5. Task execution wizard.

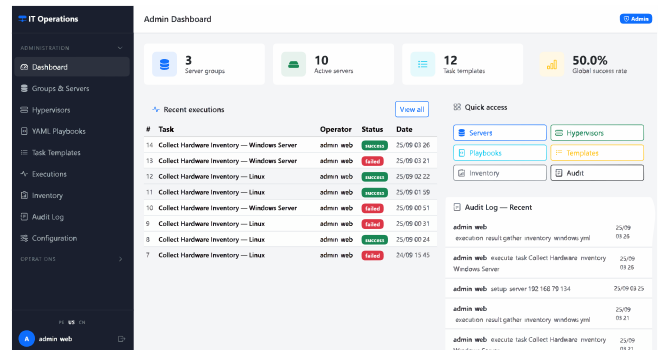


Fig. 6. Dashboard displaying consolidated operational metrics.

Group and server management organizes hosts by platform, shows their status, and allows additions, removals, and modifications, as well as automatic configuration of initial access. This organization directly feeds the dynamic construction of the Ansible inventory during each execution. Meanwhile, the hardware inventory module is now fully operational: once the collection playbook is executed, the system interprets its output and generates snapshots for each server, which are displayed both in the administration dashboard and in the operator's read-only view, thereby addressing a requirement that lacked systematic support in manual operations.

The audit log embodies the security and traceability dimensions: every significant action—logins, task executions, playbook edits, server provisioning, and denied access requests—is recorded with timestamp, user, role, action, affected resource, source IP address, and result, and can be filtered by user and paginated for review. This ability to reconstruct activity after the fact serves as direct input for the organization's internal control processes.

B. Operational Efficiency

TABLE IV. ABSOLUTE EXECUTION TIMES BY PHASE AND CONDITION

Phase	Condition	Mean (min)	SD	Min	Max
Pre-test	Reference manual procedure	53.90	6.76	45	65
Pre-test	Web application—pilot version	34.13	3.97	28	40
Post-test	Reference manual procedure	55.47	5.59	47	65
Post-test	Web application—consolidated version	12.97	3.35	8	18

Note: n = 30 operations per condition.

The descriptive analysis revealed substantial improvements across all dimensions. Table IV presents the absolute execution times recorded in each phase. In the pre-test, the pilot version of the application required an average of 34.13 minutes per operation (SD = 3.97), compared with 53.90 minutes for the reference manual procedure documented with IT personnel, which corresponds to the baseline time reduction index. In the post-test, the consolidated version reduced the mean time to 12.97 minutes per operation (SD = 3.35, maximum 18 minutes), while the times of the reference manual procedure remained stable between phases (53.90 versus 55.47 minutes), which rules out the improvement being due to a lower complexity of the

operations measured in the second phase. Moreover, the dispersion of times was notably reduced with the consolidated version ($SD = 3.35$), evidencing more predictable and homogeneous executions among operators.

From these measurements, the study's three indicators were calculated, whose descriptive statistics are presented in Table V. The time reduction index showed a positive difference of 41.00 percentage points, rising from a pre-test mean of 35.42% to a post-test mean of 76.42% with the consolidated version of the application. The operation success rate increased by 23.99 percentage points, from 70.74% to 94.73%. The event traceability level improved by 39.17 percentage points, from 55.02% to 94.19%.

TABLE V. DESCRIPTIVE STATISTICS FOR THE THREE INDICATORS

Indicator	Phase	N	Min	Max	Mean	SD
Ir (%)	Pre-test	30	11.11	56.25	35.42	12.65
Ir (%)	Post-test	30	66.04	86.15	76.42	6.47
TasaE (%)	Pre-test	30	62.86	80.77	70.74	5.05
TasaE (%)	Post-test	30	88.89	97.37	94.73	2.25
NivelT (%)	Pre-test	30	42.42	66.67	55.02	7.00
NivelT (%)	Post-test	30	89.66	97.50	94.19	2.60

The normality analysis using the Shapiro-Wilk test indicated that at least one measurement phase of each indicator did not meet the normality assumption ($p < 0.05$). Consequently, for methodological consistency, the nonparametric Wilcoxon signed-rank test for paired samples was applied in all three cases.

The statistical analysis confirmed significant improvements in the three indicators, with p-values below 0.001 and effect sizes classified as large according to Cohen's criteria. The magnitude of the standardized effect was greatest for the time reduction index, while the success rate and the traceability level showed comparable, albeit slightly smaller, effects. The detailed inferential statistics are summarized in Table VI.

TABLE VI. WILCOXON SIGNED-RANK TEST RESULTS

Indicator	Z	p-value	Effect Size (r)	Interpretation
Ir	-4.782	<0.001	0.873	Large
TasaE	-4.653	<0.001	0.850	Large
NivelT	-4.302	<0.001	0.785	Large

V. DISCUSSION

The results confirm that a web application that orchestrates tasks through Ansible significantly improves the three dimensions of enterprise IT infrastructure operations management. The improvement in time reduction of approximately 41 percentage points is consistent with the evidence that web platforms that automate technical tasks achieve notably faster resolutions than traditional methods [9], [11]. The improvement in the success rate, of approximately 24 percentage points, agrees with business studies reporting significant increases in incident resolution rates after the

implementation of web-based systems [12], [13]. The improvement in traceability of approximately 39 percentage points directly addresses the security and auditing dimension that infrastructure performance indicator classifications identify as essential [24]. The large effect sizes (r between 0.785 and 0.873) of the three indicators demonstrate that the improvements are not only statistically significant but also relevant to business operations. It is worth noting that the success rate showed the smallest absolute gain (23.99 percentage points), which is partly explained by a ceiling effect: the pilot version already achieved a baseline performance of 70.74% and, in the post-test, values concentrated near 95%, where residual failures correspond to unreachable hosts or environment unavailability rather than to tool deficiencies; even so, its effect size was large ($r = 0.850$).

The hybrid development methodology that combines the iterative practices of Scrum with the documentary rigor of the Rational Unified Process proved effective for managing the complexity of enterprise infrastructure integrations [32]. The security validation program, which covered authentication, authorization, encryption, input validation, and session management testing, confirmed that the platform meets enterprise-level security requirements for on-premises deployment.

The immutable record of every action—including user, role, resource, and IP address—provides the audit trail that manual procedures could not guarantee, and offers the empirical evidence that studies on digital transformation in the public sector call for to assess progress toward modernization [10], [25].

Several limitations must be acknowledged. The single-group quasi-experimental design limited the ability to draw robust causal inferences, and the generalization of the findings to other corporate contexts is limited; both threats are examined in detail below according to the validity categories. The study was conducted in a development environment using virtual machines; deployment in production under sustained real business load is still pending. Additionally, because the baseline corresponded to the pilot version of the application itself, the measured effects represent the improvement attributable to the consolidation of the system and not the total impact relative to a purely manual process.

To systematize their analysis, the limitations are examined according to the four validity categories customary in empirical software engineering research. Regarding internal validity, the absence of a control group makes it impossible to completely rule out learning or history effects; to mitigate this threat, each operation was first measured with the pilot version as the baseline, and all measurements were automatically recorded in the system database, eliminating manual transcription and observer bias.

Regarding external validity, the results come from a single organization and a controlled development environment that replicated its operating conditions, which limits direct generalization. Nevertheless, the faithful reproduction of the environment—same operating system distributions, network topology, and procedures with IT personnel—favors the transferability of the findings to organizations with similar technological profiles.

Regarding construct validity, the three indicators (Ir, TasaE, and NivelT) operationalize IT operations management through verifiable formulas and ratio scales; however, they do not capture qualitative dimensions such as operator satisfaction or cognitive load, whose incorporation could be addressed in future work.

Finally, regarding conclusion validity, the use of the Wilcoxon test—appropriate for the sample size and the non-normality verified through Shapiro-Wilk—together with the reporting of effect sizes, reduces the risk of erroneous statistical conclusions.

VI. CONCLUSION AND FUTURE WORK

This study designed, developed, and functionally validated an on-premises web application that improves the management of enterprise IT infrastructure operations. The application, built on a layered architecture using Ubuntu Server, PostgreSQL 16, Python 3 (Flask), and Ansible as the agentless automation engine, features authentication through the Lightweight Directory Access Protocol (LDAP) of Active Directory, role-based access control, and comprehensive audit logging.

Regarding the specific objectives, the solution increased the time reduction index by approximately 41 percentage points, raised the operation success rate by approximately 24 percentage points, and improved the event traceability level by approximately 39 percentage points. Wilcoxon signed-rank tests confirmed statistically significant improvements in the three indicators (p-values below 0.001, large effect sizes), with Z statistics ranging from -4.782 to -4.302. The detailed inferential results are presented in Table VI. The research went beyond the development of a software tool and provided empirical evidence that the systematic digitalization of enterprise infrastructure operations, adapted to the local corporate context, is a possible path to modernizing the technology industry.

The agentless architecture, validated by recent studies [15], [16], eliminates the overhead of agent management while maintaining scalability. The implementation of role-based access control, grounded in established access control models, ensures that administrative and operational functions remain properly segregated [33]. These findings provide a replicable framework for organizations seeking to modernize their information technology operations using agentless automation platforms.

A promising direction for future development is the implementation of a tiered change-approval workflow for significant infrastructure modifications. When an operator requests a high-impact change, the system could require a mandatory approval step by a user with an administrator profile before the automation playbook is sent to the execution engine. This governance layer aligns with recent frameworks that incorporate risk mitigation and internal controls into information technology operations, guaranteeing that critical changes receive adequate oversight and that accountability is maintained throughout the entire change life cycle [34].

Finally, as a broader extension, the incorporation of artificial intelligence-assisted validation of the Ansible playbooks uploaded to the platform is proposed. This component would automatically analyze files before execution to detect syntax

errors, deviations from best practices, and potentially risky configurations, alerting the operator before the task reaches the execution engine. This line of work would connect the platform with recent AIOps trends, where language models and static analysis are combined to increase the reliability of infrastructure automation.

Another improvement involves strengthening the security layer through the integration of two-factor authentication based on time-based one-time passwords, which would further protect the platform against credential compromise. A systematic review of multifactor authentication methods confirms that time-based one-time password schemes remain one of the most practical and effective mechanisms for enterprise systems, and their inclusion complements the existing dual authentication model without disrupting the user experience [35].

REFERENCES

- [1] E. Veloz, V. Veloz, and D. Zamora, "Aplicaciones digitales como apoyo en la producción, agilidad y administración dentro de la empresa," *Revista Científica Ciencia y Tecnología*, vol. 22, no. 36, Oct. 2022, doi: 10.47189/RCCT.V22I36.524.
- [2] S. Nambisan, M. Wright, and M. Feldman, "The digital transformation of innovation and entrepreneurship: Progress, challenges and key themes," *Res. Policy*, vol. 48, no. 8, p. 103773, Oct. 2019, doi: 10.1016/J.RESPOL.2019.03.018.
- [3] Uptime Institute, "Uptime Institute's 2022 Outage Analysis Finds Downtime Costs and Consequences Worsening as Industry Efforts to Curb Outage Frequency Fall Short." [Online]. Available: <https://uptimeinstitute.com/about-ui/press-releases/2022-outage-analysis-finds-downtime-costs-and-consequences-worsening>
- [4] D. Hajializadeh and M. Imani, "RV-DSS: Towards a resilience and vulnerability-informed decision support system framework for interdependent infrastructure systems," *Comput. Ind. Eng.*, vol. 156, p. 107276, Jun. 2021, doi: 10.1016/J.CIE.2021.107276.
- [5] S. Angelopoulos, E. Bendoly, J. Fransoo, K. Hoberg, C. Ou, and A. Tenhiälä, "Digital transformation in operations management: Fundamental change through agency reversal," *Journal of Operations Management*, vol. 69, no. 6, pp. 876–889, Sep. 2023, doi: 10.1002/joom.1271.
- [6] M. Govers and P. Van Amelsvoort, "A theoretical essay on socio-technical systems design thinking in the era of digital transformation," *Gruppe. Interaktion. Organisation. Zeitschrift für Angewandte Organisationspsychologie (GIO)*, vol. 54, no. 1, pp. 27–40, Apr. 2023, doi: 10.1007/S11612-023-00675-8.
- [7] A. A. Vărzaru and C. G. Bocean, "Digital Transformation and Innovation: The Influence of Digital Technologies on Turnover from Innovation Activities and Types of Innovation," *Systems*, vol. 12, no. 9, 2024, doi: 10.3390/systems12090359.
- [8] P. Kumar, N. Duraimurugan, and K. Ananthajothi, "Centralized Service Management Across Hybrid Environments: Harnessing Zookeeper with Cloud Computing," 2024 International Conference on Smart Technologies for Sustainable Development Goals, ICSTSDG 2024, 2024, doi: 10.1109/ICSTSDG61998.2024.11026216.
- [9] K. Trikusuma, V. Yasin, T. Budiman, A. Zulkarnain Sianipar, and A. Budi Yulianto, "IT Infrastructure Dashboard Monitoring Application Development Using Grafana and Prometheus, a Case Study at Astra Polytechnic School," 2023 IEEE International Conference of Computer Science and Information Technology: The Role of Artificial Intelligence Technology in Human and Computer Interactions in the Industrial Era 5.0, ICOSNIKOM 2023, 2023, doi: 10.1109/ICOSNIKOM60230.2023.10364485.
- [10] M. I. Sarwar, Q. Abbas, T. Alyas, A. Alzahrani, T. Alghamdi, and Y. Alsaawy, "Digital Transformation of Public Sector Governance With IT Service Management—A Pilot Study," *IEEE Access*, vol. 11, pp. 6490–6512, 2023, doi: 10.1109/ACCESS.2023.3237550.

- [11] H. Nugroho, Aradea, and K. Surendro, "IT Service Management Intelligence Model to Support the Implementation of Electronic Government System (EGS) in Indonesia," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 12, no. 2, pp. 494–500, Apr. 2022, doi: 10.18517/ijaseit.12.2.13789.
- [12] S. Pérez y F. Mendoza, "Sistema Help Desk: una solución para la gestión de incidencias," *Gaceta Científica*, vol. 10, no. 2, pp. 76–81, mayo 2024, doi: 10.46794/gacien.10.2.2188.
- [13] F. Sánchez and M. Valles, "Implementation of ITIL version 3 in organizations: Reasons for success and failure," *Revista Científica de Sistemas e Informática*, vol. 1, no. 2, pp. 54–67, Jul. 2021, doi: 10.51252/RCSI.V1I2.191.
- [14] P. Patil, S. Phatkare, P. Rane, and S. Janathe, "Smart Infrastructure Management," *Int. J. Innov. Sci. Res. Technol.*, vol. 11, no. 4, p. 18, May 2026, doi: 10.38124/IJISRT/26APR1306.
- [15] R. Miñón et al., "ArtifactOps and ArtifactDL: a methodology and a language for conceptualizing and operationalising different types of pipelines," *Journal of Cloud Computing*, vol. 14, no. 42, Aug. 2025, doi: 10.1186/S13677-025-00761-W.
- [16] R. Shrestha and A. A. N. Ali, "Configuration Management in Kubernetes Environments: A GitOps Approach," *Proceedings - 2024 IEEE/ACM 17th International Conference on Utility and Cloud Computing, UCC 2024*, pp. 497–502, 2024, doi: 10.1109/UCC63386.2024.00077.
- [17] A. K. Akuthota, "Role-Based Access Control (RBAC) in Modern Cloud Security Governance: An In-depth Analysis," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 2, pp. 3297–3311, Apr. 2025, doi: 10.32628/CSEIT25112793.
- [18] G. Fragkos, J. Johnson, and E. E. Tsiropoulou, "Centralized and Decentralized Distributed Energy Resource Access Control Implementation Considerations," *Energies (Basel)*, vol. 15, no. 17, p. 6375, Sep. 2022, doi: 10.3390/EN15176375.
- [19] Red Hat, "Ansible AWX Documentation," 2026. [Online]. Available: <https://docs.ansible.com/projects/awx/>
- [20] PagerDuty, "Rundeck Documentation," 2026. [Online]. Available: <https://docs.rundeck.com/>
- [21] J. O. Benson, J. J. Prevost, and P. Rad, "Survey of Automated Software Deployment for Computational and Engineering Research," in 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, USA: IEEE, Apr. 2016, pp. 1–6. doi: 10.1109/SYSCON.2016.7490666.
- [22] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *Management Information Systems Quarterly*, vol. 28, no. 1, pp. 75–106, Mar. 2004, doi: 10.2307/25148625.
- [23] D. I. K. Sjøberg et al., "Conducting realistic experiments in software engineering," *ISESE 2002 - Proceedings, 2002 International Symposium on Empirical Software Engineering*, pp. 17–26, 2002, doi: 10.1109/ISESE.2002.1166921.
- [24] R. R. Fatkueva and A. S. Sudakov, "IT Infrastructure Performance Indicators," *Scientific and Technical Information Processing*, vol. 52, no. 1, pp. 38–44, Jun. 2025, doi: 10.3103/S0147688225700029.
- [25] N. Fahmi, S. Murdha, and Suhardi, "Developing Digital Service Transformation Maturity Model in Public Sector," *IEEE Access*, vol. 12, pp. 174491–174506, 2024, doi: 10.1109/ACCESS.2024.3468341.
- [26] Z. Tu, "Research on the Application of Layered Architecture in Computer Software Development," *Journal of Computing and Electronic Information Management*, vol. 11, no. 3, pp. 34–38, Nov. 2023, doi: 10.54097/JCEIM.V11I3.08.
- [27] R. Dyer and J. Chauhan, "An exploratory study on the predominant programming paradigms in Python code," *ESEC/FSE 2022 - Proceedings of the 30th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 684–695, Nov. 2022, doi: 10.1145/3540250.3549158.
- [28] M. L. Hetland and F. Nelli, "Activity 3: Building a Web App with Flask," in *Beginning Python: From Novice to Professional*, M. L. Hetland and F. Nelli, Eds., Berkeley, CA: Apress, 2024, pp. 517–535. doi: 10.1007/979-8-8688-0196-9_27.
- [29] M. Jannah, Dahlan, and C. Akbar, "Development of Management Information System Based on MVC Architecture to Improve Business Process Efficiency," *Journix: Journal of Informatics and Computing*, vol. 1, no. 2, pp. 63–76, Aug. 2025, doi: 10.63866/JOURNIX.V1I2.11.
- [30] H. Sanugommula, "Exploration of the AWS Step Functions, Ansible, and AWS Lambda: Integrating Cloud Automation and Orchestration for Enhanced DevOps Efficiency," *ESP Journal of Engineering & Technology Advancements*, vol. 4, no. 3, pp. 108–110, 2024, doi: 10.56472/25832646/JETA-V4I3P109.
- [31] M. Sheng, Y. Wang, C. Yang, S. V. Salunke, and A. Ouda, "A Performance Benchmark for the PostgreSQL and MySQL Databases," *Future Internet*, vol. 16, no. 10, p. 382, Oct. 2024, doi: 10.3390/FI16100382.
- [32] J. Reiff and D. Schlegel, "Hybrid project management – a systematic literature review," *International Journal of Information Systems and Project Management*, vol. 10, no. 2, pp. 45–63, Jul. 2022, doi: 10.12821/IJISPM100203.
- [33] J. S. Park and R. Sandhu, "RBAC on the Web by smart certificates," in *Proceedings of the Fourth ACM Workshop on Role-Based Access Control*, in RBAC '99. New York, NY, USA: Association for Computing Machinery, 1999, pp. 1–9. doi: 10.1145/319171.319172.
- [34] O. H. Plant, J. van Hillegerberg, and A. Aldea, "Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment," *International Journal of Accounting Information Systems*, vol. 45, p. 100560, Jun. 2022, doi: 10.1016/J.ACCINF.2022.100560.
- [35] F. Sunny, A. Das, and J. Jobin, "A Review on Two-Factor Authentication Systems," *International Journal of Engineering Research & Technology*, vol. 11, no. 4, Jul. 2023, doi: 10.17577/IJERTCONV11IS04039.