UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

Tesis en formato de artículo científico

Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector

Presentado por:

Luis Alberto Pfuño Alccahuamani

Anthony Meza Bautista

Para optar el título de Ingeniero Informático y Sistemas

Abancay, Perú

2026

UNIVERSIDAD NACIONAL MICAELA BASTIDAS DE APURÍMAC

FACULTAD DE INGENIERÍA

ESCUELA ACADÉMICO PROFESIONAL DE INGENIERÍA INFORMÁTICA Y SISTEMAS

TESIS EN FORMATO DE ARTÍCULO CIENTÍFICO

Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector

Presentado por **Luis Alberto Pfuño Alccahuamani** y **Anthony Meza Bautista**, para optar el título de Ingeniero Informático y Sistemas

Sustentado y aprobado el 16 de julio del 2026, ante el jurado evaluador:

Presidente: *Mtro. Marleny Peralta Ascue*

Primer miembro: *Mg. Mario Aquino Cruz*

Segundo miembro: *Mtro. Virgilio Martinez Duran*

Asesor: *Dra. Hesmeralda Rojas Enriquez*

Año del Fortalecimiento de la Soberanía Nacional

CONSTANCIA DE ORIGINALIDAD
Nº 050-2026

La Universidad Nacional Micaela Bastidas de Apurímac, a través de la Unidad de Investigación de la Facultad de Ingeniería declara que, la tesis en formato de articulo cientifico titulada: **Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector,** presentado por los Bachrs. **LUIS ALBERTO PFUÑO ALCCAHUAMANI y ANTHONY MEZA BAUTISTA,** para optar el título de **Ingeniero Informática y Sistemas;** ha sido sometido a un mecanismo de evaluación y verificación de similitud, a través del Software Turnitin, siendo el índice de similitud ACEPTABLE de **(5%)** por lo que, cumple con los criterios de originalidad establecidos por la Universidad.

Abancay, 18 de febrero del 2026

Atentamente,

UNIVERSIDAD NACIONAL MICAELA BASTIDAS
FACULTAD DE INGENIERÍA
Ing. Hesmeralda Rojas Enriquez
DIRECTORA DE LA UNIDAD DE INVESTIGACIÓN
FACULTAD DE INGENIERÍA

C. c.
Archivo
REG. Nº 183

Agradecimiento

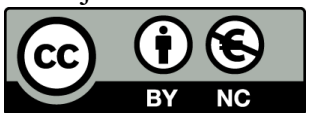
Expresamos nuestra más profunda gratitud a nuestra asesora, la Dra. Hesmeralda Rojas Enriquez, por su invaluable guía, su paciencia y por compartir su conocimiento con nosotros en cada etapa de esta investigación. Asimismo, extendemos nuestro agradecimiento a todas aquellas personas e instituciones que, de manera directa o indirecta, brindaron su apoyo, tiempo y consejos para hacer posible la culminación de este trabajo.

Dedicatoria

Dedicamos este trabajo a nuestras familias, por ser nuestro apoyo incondicional, nuestro refugio en los momentos difíciles y el motor que nos impulsa a cumplir cada una de nuestras metas. Todo nuestro esfuerzo es para ustedes.

Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector

Línea de investigación: Ingeniería de software e innovación tecnológica



ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

computers

MDPI

Article

Hybrid Web Architecture with AI and Mobile Notifications to Optimize Incident Management in the Public Sector

Luis Alberto Pfuño Alccahuamani, Anthony Meza Bautista and Hesmeralda Rojas *

Departamento Académico de Informática y Sistemas, Universidad Nacional Micaela Bastidas de Apurímac, Abancay 03001, Peru; 191229@unamba.edu.pe (L.A.P.A.); 181221@unamba.edu.pe (A.M.B.)
* Correspondence: hrojas@unamba.edu.pe

Abstract

This study addresses the persistent inefficiencies in incident management within regional public institutions, where dispersed offices and limited digital infrastructure constrain timely technical support. The research aims to evaluate whether a hybrid web architecture integrating AI-assisted interaction and mobile notifications can significantly improve efficiency in this context. The ITIMS (Intelligent Technical Incident Management System) was designed using a Laravel 10 MVC backend, a responsive Bootstrap 5 interface, and a relational MariaDB/MySQL model optimized with migrations and composite indexes, and incorporated two low-cost integrations: a stateless AI chatbot through the OpenRouter API and asynchronous mobile notifications using the Telegram Bot API managed via Laravel Queues and webhooks. Developed through four Scrum sprints and deployed on an institutional XAMPP environment, the solution was evaluated from January to April 2025 with 100 participants using operational metrics and the QWU usability instrument. Results show a reduction in incident resolution time from 120 to 31 min (74.17%), an 85.48% chatbot interaction success rate, a 94.12% notification open rate, and a 99.34% incident resolution rate, alongside an 88% usability score. These findings indicate that a modular, low-cost, and scalable architecture can effectively strengthen digital transformation efforts in the public sector, especially in regions with resource and connectivity constraints.

Keywords: low-cost architecture; decoupled architecture; OpenRouter AI chatbot; Telegram notifications; ticket management

1. Introduction

In the context of the digital transformation of the Peruvian public sector, optimizing processes for managing technical IT incidents—failures or malfunctions in computer equipment and information systems—requires robust technical architectures to enhance operational efficiency and scalability [1]. Peru achieved a 20% increase in interoperability and deployed 24 Computer Security Incident Response Teams (CSIRTs) in 2024 [2], in alignment with the National Digital Government Plan, which aims to accelerate interoperability and strengthen cybersecurity capabilities [2,3].

However, in Andean regions, technical and territorial gaps persist, limiting the integration of digital solutions and the adoption of online platforms [4]. This is the case of the Regional Government of Apurímac (GORE) in Abancay, the institution that administers the region and employs more than 450 staff across approximately 66 offices [5]. One of its core processes is incident management, where employees report technical issues through a single telephone channel, resulting in delays of up to 120 min, the accumulation of incidents

Academic Editor: Michail Kalogiannakis

Received: 8 December 2025
Revised: 1 January 2026
Accepted: 3 January 2026
Published: 12 January 2026

lacking clear descriptions or database indicators, and ultimately reduced productivity and constrained technical planning [6]. This situation reflects broader national challenges that are exacerbated by the digital divide affecting regions outside the capital [1].

In response to this situation, the design, development and implementation of a web-based ticketing system integrated with an AI chatbot and mobile notifications was proposed, using a hybrid MVC architecture based on Laravel 10 for the backend, responsive Bootstrap 5 for the frontend, and MySQL for relational persistence [7,8]. The system incorporates low-cost integrations through external APIs such as OpenRouter (for generative AI restricted to FAQs) and the Telegram Bot API (for push alerts) [9,10]. Accordingly, the following research question was formulated:

- RQ: How does a web architecture, integrating AI and mobile notifications, optimize incident management efficiency in the GORE, by reducing response times and increasing resolution rates?

The implementation of the system aligns with the general objective of improving incident management through an automated and scalable architecture (see Figure 1), with the following specific objectives:

- RO1: Reduce the average incident resolution time through optimized backend processes in Laravel [11].
- RO2: Increase the number of incidents resolved versus reported through MySQL-based traceability and FIFO assignment [12].
- RO3: Optimize the technical response time from notification to initial action through push alerts via Telegram [9].
- RO4: Improve overall system usability by measuring satisfaction with navigability and efficiency using the QWU instrument and the Bootstrap 5 frontend [13].
- RO5: Increase the interaction rate of the AI chatbot using structured prompts in OpenRouter [10].
- RO6: Raise the open rate of mobile notifications to accelerate workflow processing through asynchronous Telegram webhooks [14].

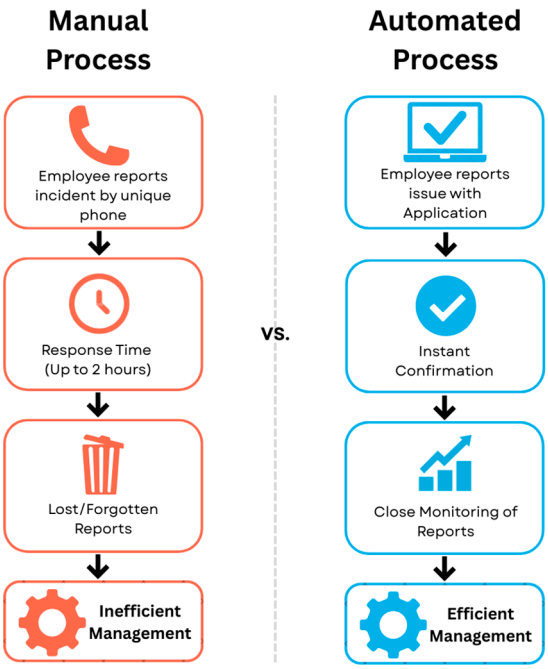


Figure 1. Flowchart of Current vs. Proposed Problem/Solution.

https://doi.org/10.3390/computers15010047

2. Related Work

The review of related work situates this study within the development of decoupled architectures aimed at achieving greater flexibility, scalability, and resilience. When applied in public institutions, these architectures demonstrate how modularity enables system integration, reduces operational risks and allows faster adaptation to regulatory changes [15]. Their incorporation into technical incident management also contributes to increased process efficiency [2,4,16]. In local implementations, such as municipalities or public entities with limited resources, these architectures have reduced incident response times by up to 40% by optimizing workflow processes [4,17,18].

Such is the case of the work by [19], who implemented a microservices architecture in which components communicated through lightweight protocols such as REST APIs or gRPC (Google Remote Procedure Call), achieving more efficient scalability by enabling only the components requiring additional capacity to scale. Similarly, the study by [20] adopted a microservices-like approach by decoupling macroservices and enabling their interaction through APIs. Ramos Herrera (2022) also contributed to this field by implementing the osTicket management system in the Superior Court of Justice of Lima, processing more than 50 requests per day and achieving an average savings of approximately 30 s per ticket during registration and closure stages [21], thereby demonstrating its effectiveness for infrastructure management. Likewise, Mali et al. (2025) designed an "Online Chatbot Based Ticketing System" in India, integrating NLP for automatic categorization to reduce operational costs [22]; although scalable, the system remains commercial and lacks technical notification features. Additionally, Karimi et al. (2024) implemented hybrid intentional/generative chatbots for the Help Desk at the Polytechnic University of Turin, supporting more than 20,000 requests and reducing waiting times [23].

Regarding user satisfaction with the implemented technology, the study by [24], reported that, according to survey results, satisfaction increased from 84.38% to 100%. The same study [24], which developed a system for the company BEMAST E.I.R.L. for sales management, also found through a survey of 32 participants that 84.38% were dissatisfied with previous processes and that 100% supported the adoption of digital technologies, highlighting the need for digital solutions, although the study was limited to a private-sector context.

Recent research in distributed intelligent systems highlights the importance of efficient and coordinated communication to reduce response times under resource constraints. Although this work does not formally model multi-agent systems, it shares these principles by prioritizing asynchrony, information flow optimization, and the reduction of information congestion. Within this framework, the integration of mobile notifications and AI-based assistance is interpreted as a practical application of effective coordination among human actors in institutional environments.

To further clarify the contribution of this study, Table 1 summarizes the main differences between representative prior systems and the proposed ITIMS solution, considering architectural approach, role of artificial intelligence, notification mechanisms, and evaluation scope.

https://doi.org/10.3390/computers15010047

Table 1. Comparison of related work and the proposed ITIMS system.

Study	Architecture	AI Role	Notification Mechanism	Evaluation Scope
Ramos Herrera [21]	Monolithic (osTicket)	None	Email	Single institution, descriptive
Mali et al. [22]	Centralized web system	NLP-based categorization	None	Commercial deployment
Karimi et al. [23]	Hybrid academic system	Generative chatbot	Web interface	University help desk
This work (ITIMS)	Modular MVC (Laravel)	FAQ-restricted AI assistant	Mobile (Telegram)	Public sector case study

3. Methodology

3.1. Population and Sample

The target population comprised more than 450 employees of the Regional Government of Apurímac (GORE), Central Headquarters in Abancay. It included end users and operational staff, as well as the technical team consisting of five IT support specialists from the Informatics Unit [5]. The sample consisted of 100 participants selected through non-probabilistic convenience sampling, of whom 95 were end users and 5 belonged to the IT support team. The sampling strategy corresponds to a non-probabilistic convenience sample, which is appropriate for exploratory and applied case studies in institutional settings but limits statistical generalizability beyond the studied context [25].

3.2. Techniques and Instruments

The data collection techniques combined qualitative and quantitative methods, integrating non-participant observation and analysis of historical records [6] to identify patterns of recurrent failures, such as hardware and connectivity issues. These were complemented by semi-structured interviews with key stakeholders (two IT managers, ten representative employees, and five technicians), guided by a script focused on manual inefficiencies and automation needs. The instruments included digital surveys to evaluate post-implementation satisfaction, the Questionnaire for Website Usability (QWU) [13,26] with a Likert scale consisting of 21 items that evaluated navigability and efficiency, and automatic system logs used to collect operational metrics.

During the pre-implementation phase, a baseline response time of approximately 120 min was established as a referential estimate. This value was derived from triangulation of historical reports, manual records, and interviews with technical staff, rather than systematic automated measurement, due to the high variability of incidents and the absence of digital logging mechanisms at that stage. This estimated reference average was derived from the triangulation of historical reports, manual records, and interviews with technical staff, as automated measurement was not feasible due to the high variability of incidents. In the post-implementation phase, response time was automatically measured using precise timestamps from ticket registration to resolution, allowing for more accurate and consistent measurements.

Table 2 summarizes the main instruments and their applications.

https://doi.org/10.3390/computers15010047

Table 2. Evaluation dimensions, instruments, and performance indicators of the AI-enhanced web-based ticketing system with mobile notifications.

Dimension	Indicator	Instrument
System usability	Satisfaction	Questionnaire for Website Usability (QWU) [13]
AI chatbot	Interaction rate (percentage of queries answered)	Automatic system log
Mobile notifications	Open rate (percentage of notifications opened)	Notification platform log
Resolution time	Average time from registration to resolution (minutes)	System report
Number of incidents processed and resolved	Cases resolved vs. cases reported	System log
Technical staff response time	Minutes from notification to initial action	Automatic system log

3.3. Development and Implementation Methodology

The procedure was carried out using an iterative Scrum methodology, as shown in Figure 2, and adapted to a two-researcher team working in collaborative roles [27,28]. Development and implementation took place from October to December 2024, culminating in deployment following institutional approval. Subsequently, data collection for evaluating the indicators was conducted from January to April 2025. The problem was addressed through objectives defined as tasks or user stories, organized and prioritized in the backlog and executed across four sprints.

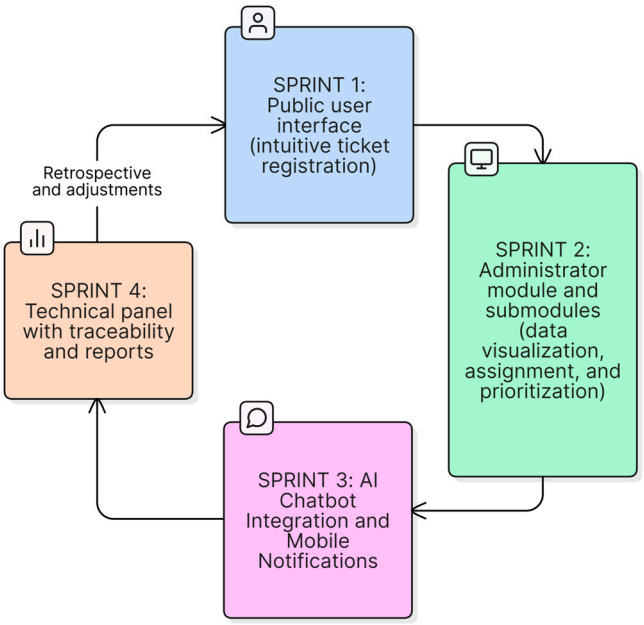


Figure 2. Scrum methodology workflow, with emphasis on the incremental development of interfaces, AI functionalities, and mobile notifications.

The procedural phases were structured within an incremental cycle, with the following key activities carried out sequentially:

https://doi.org/10.3390/computers15010047

- Requirements elicitation and analysis: Observation of historical processes (2023–2024) and interviews to identify failure patterns and develop user stories.
- Product Backlog creation: Prioritized list of functionalities, including ticket registration, FAQ-oriented chatbot, and mobile notifications.
- Sprint planning: Sprint 1 focused on the public interface; Sprint 2 on the admin module and submodules; Sprint 3 on AI and notification integration; and Sprint 4 on the technical panel and refinements.
- Incremental development: The system was coded in local environments (XAMPP), using Laravel 10 for the backend and MySQL as the database, Bootstrap 5 for the responsive frontend, OpenRouter for the chatbot (Meta/Google models), and the Telegram API for notifications.
- Meetings and review: At the end of each sprint, review sessions were conducted with the technical team for feedback.
- Testing and validation: Unit and integration tests (for AI and notifications) were performed, and bugs were corrected in short iterations.
- Implementation and monitoring: After institutional approval, the system was deployed on the GORE Apurímac server. Initial metrics were monitored during the first month (January 2025) through automated dashboards.
- Documentation and evaluation: A user manual and final reports were prepared, and results were assessed against the objectives. All documentation was stored in GitHub repository for version control and access.

4. System Architecture

4.1. Rationale for the Hybrid Custom Architecture

Although consolidated ticketing platforms such as osTicket represent widely adopted commercial off-the-shelf (COTS) solutions, their suitability for public sector organizations is often constrained by structural, economic, and regulatory factors. Public institutions typically operate under strict budgetary limitations, require full control over institutional data, and depend on highly specific administrative workflows that are difficult to accommodate within preconfigured software solutions. As noted by Sommerville (2016) [29], COTS systems tend to impose predefined process models that may conflict with organizational practices, particularly in contexts where adaptability and long-term sustainability are critical. These limitations are summarized and contrasted with the proposed solution in Table 3, which highlights the main architectural and functional differences between existing ticketing platforms and the custom-built approach.

In addition, the growing incorporation of generative artificial intelligence into service management systems poses further challenges when relying on third-party platforms. The integration of AI functionalities in off-the-shelf solutions usually depends on external services and opaque conversational logic, reducing institutional control over data processing, prompt design, and validation mechanisms [30]. For public organizations, this lack of control raises concerns related to data sovereignty, accountability, and compliance with regulatory frameworks governing information management.

From an architectural standpoint, modular and decoupled systems offer greater flexibility to adapt to evolving administrative requirements and heterogeneous technological environments commonly found in the public sector. According to Bass [31], such architectures enhance maintainability, scalability, and functional evolution, allowing systems to grow incrementally without incurring excessive costs or vendor lock-in. In this context, the proposed hybrid architecture, developed using open-source technologies and deployed primarily on-premise, enables the customization of business logic, the controlled integration

https://doi.org/10.3390/computers15010047

of generative AI, and the use of Telegram as an active operational channel rather than a simple notification mechanism.

Table 3. Comparative analysis between commercial solutions and the proposed modular architecture.

Criterion	Existing Solutions (e.g., osTicket)	Proposed Architecture (Laravel + AI + Telegram)
Deployment model	Monolithic architecture with extensions and plugins	Decoupled and modular service-based architecture
Institutional workflow adaptation	Limited; requires core modifications or additional plugins	Fully customizable according to the GORE (Regional Government) structure
Generative AI integration	Non-native; depends on external services without flow control	Controlled AI integration (FAQs, structured prompts, and ticket routing)
Conversational flow control	Limited or non-existent	Total control over prompts, scope, validation, and traceability
Telegram integration	Generally restricted to notifications	Operational channel: notification, assignment, and status changes
Metrics instrumentation	Predefined generic metrics	Custom-designed metrics according to institutional indicators
Licensing costs	Variable; dependent on plugins	Low cost, based on open-source tools
Data sovereignty control	Dependent on extensions or external services	Primarily on-premise processing and institutional control
Functional scalability	Conditioned by platform capabilities	Scalability by design (via queues, APIs, and modules)

Therefore, the decision to implement a custom-built architecture responds not to a technological preference but to the specific constraints and needs of a public institution, prioritizing budget efficiency, data sovereignty, adaptability to administrative processes, and long-term sustainability. This approach aligns with established software engineering principles and represents a viable alternative for governmental environments with limited resources and complex operational requirements.

4.2. Developed Architecture

The web-based ticketing system, designed for scalability without resource limitations (with a server capable of handling operational peaks) [6], supports differentiated workflows: end users, who access the system without authentication and focus on ticket submission and stateless chatbot interaction, and technical staff, who authenticate through Laravel Sanctum [7,27]. Key integrations include the OpenRouter API for the chatbot, featuring rotation of free models with prompts restricted to FAQs [10], and the Telegram Bot API for chronological push notifications based on creation timestamps [9,32]. During the evaluation period, more than 1050 tickets were processed without performance issues. This architecture ensures relational traceability through MySQL, minimal API latency, and low operational cost.

Figure 3 presents the layers of the developed architecture:

1. Presentation Layer (Frontend): Developed using Bootstrap 5 to provide responsive and accessible interfaces, with CSS/JS components for form validation and timeline visualization. Views are rendered through Blade templates, capturing end-user inputs (e.g., national ID for automatic data completion) without requiring initial authentication. This layer is responsible for delivering an intuitive user experience on mobile

https://doi.org/10.3390/computers15010047

devices, incorporating components such as chatbot modals and tables for tracking ticket statuses (pending/in progress/resolved/cancelled) [11,33].

2. Business Logic Layer (Backend): Built on Laravel 10, this layer defines RESTful routes (in web.php and api.php) and controllers for handling requests. It uses the Eloquent ORM for model abstraction, processing logic such as description validation, ID generation, and notification triggers. Sanctum middleware manages authentication exclusively for technical staff, while queued jobs ensure asynchronous execution for external integrations [7,11].
3. Database and Integrations Layer: MariaDB (MySQL-compatible) serves as the relational database for persistent storage. Eloquent models map key entities with optimized queries [11,34]. External integrations, including OpenRouter (stateless, via Guzzle HTTP) [35] and the Telegram Bot API (POST-based webhooks), are invoked from controllers. Fallback events are logged in Laravel, and quantitative metrics—such as messages sent, success/error rates, model used, and timestamps—are stored in the Chatbot table [10,14].

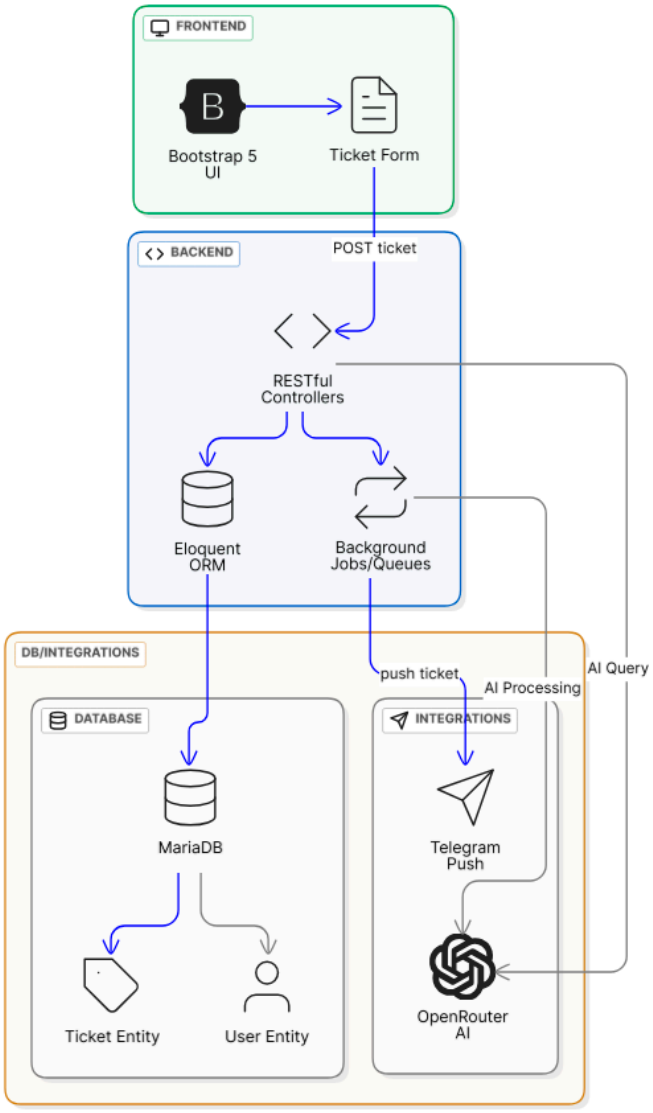


Figure 3. Architecture diagram (Bootstrap frontend → Laravel backend → MariaDB database → integrations), highlighting data flows.

https://doi.org/10.3390/computers15010047

4.3. Data Model

The data model was implemented in MariaDB 10.4.32 (integrated within XAMPP 8.2.12) [34], using a normalized relational schema to ensure data integrity and efficiency in CRUD operations [36]. It was defined through Laravel migrations, incorporating composite indexes on fields such as creation_date and status to optimize FIFO ordering and paginated queries [12]. The Eloquent ORM abstracts models with dynamic relationships. This design handles tickets without performance overhead and stores quantitative logs of chatbot interactions for aggregated metrics (success/failure, model used, timestamps) [22]. The Entity–Relationship diagram is shown in Figure 4.

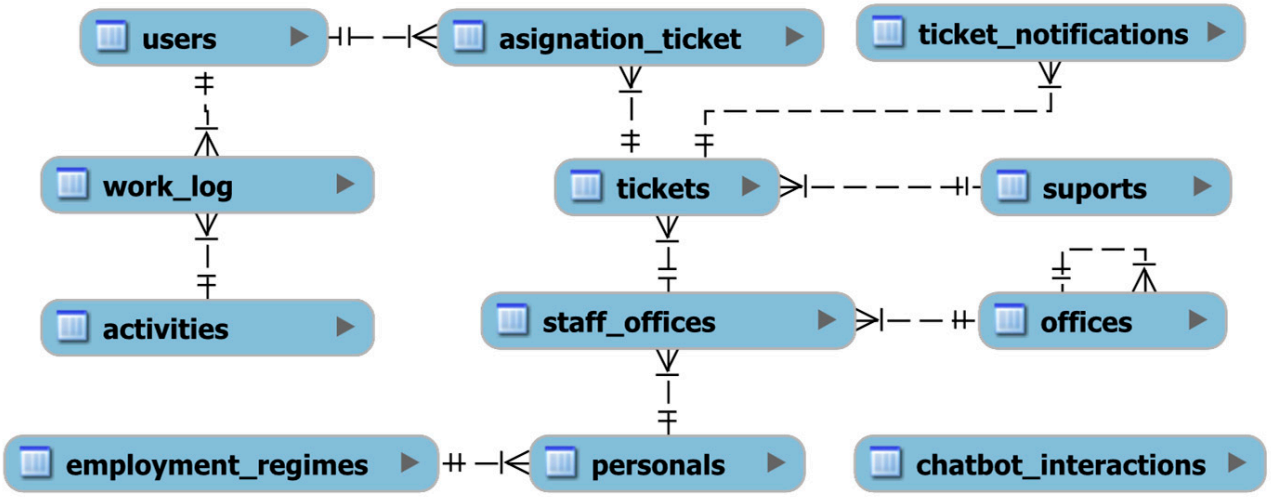


Figure 4. ER diagram of the data model.

4.4. Main Workflows

The system workflows were designed for simplicity and efficiency, differentiated by user role and leveraging the MVC model. They process a high volume of tickets according to operational demand, with automatic assignment to balance workload.

1. End-User Workflow (Unauthenticated)

The process begins when the user accesses the system through the main interface, as shown in Figure 5. Optionally, the user may interact with the integrated chatbot, which provides responses to frequently encountered technical issues. The user then proceeds to register an incident by creating a ticket, entering their national ID number to enable automatic completion of personal data and providing a detailed description of the problem. Once the form is submitted, the system automatically generates the ticket with the initial status "Registered," ensuring its insertion into the database. Finally, the user can check the progress of their request by entering the ticket ID, which displays its current status—Registered, In Progress, Resolved, or Cancelled.

https://doi.org/10.3390/computers15010047

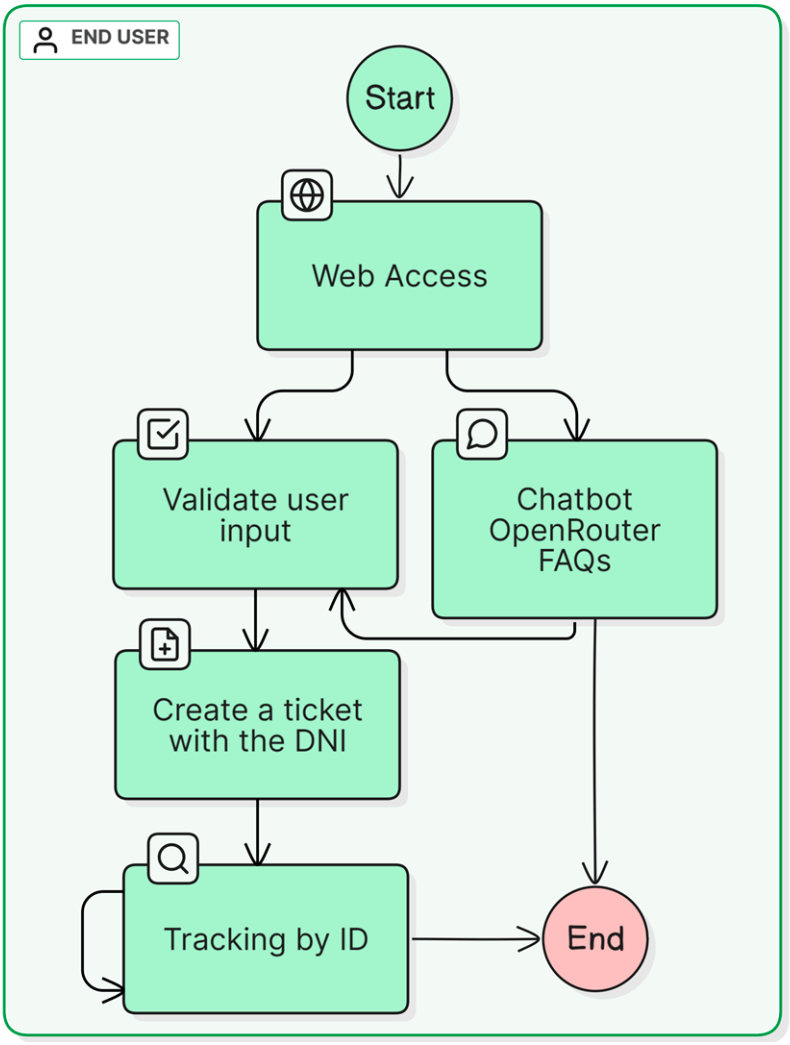


Figure 5. End-user workflow.

2. Technical Staff Workflow (Sanctum Authentication)

The process begins when the technician logs into the system panel using authorized credentials. When a new ticket is generated, the system automatically sends a Telegram notification to the technician's mobile device, including essential details such as the ticket ID, the user's national ID number, a summarized description of the issue, and a direct link to the corresponding record in the web system. The technician may assign the ticket to themselves directly from Telegram using an action button that updates its status to "In Progress," or they may perform the assignment from the web panel, where tickets are organized in a FIFO (First In, First Out) queue [12] based on creation time. Once assigned, the technician accesses the system to review the full details of the incident, update its status to "Resolved" or "Cancelled," and log the actions performed, ensuring traceability and proper case resolution. The detailed workflow is shown in Figure 6.

https://doi.org/10.3390/computers15010047

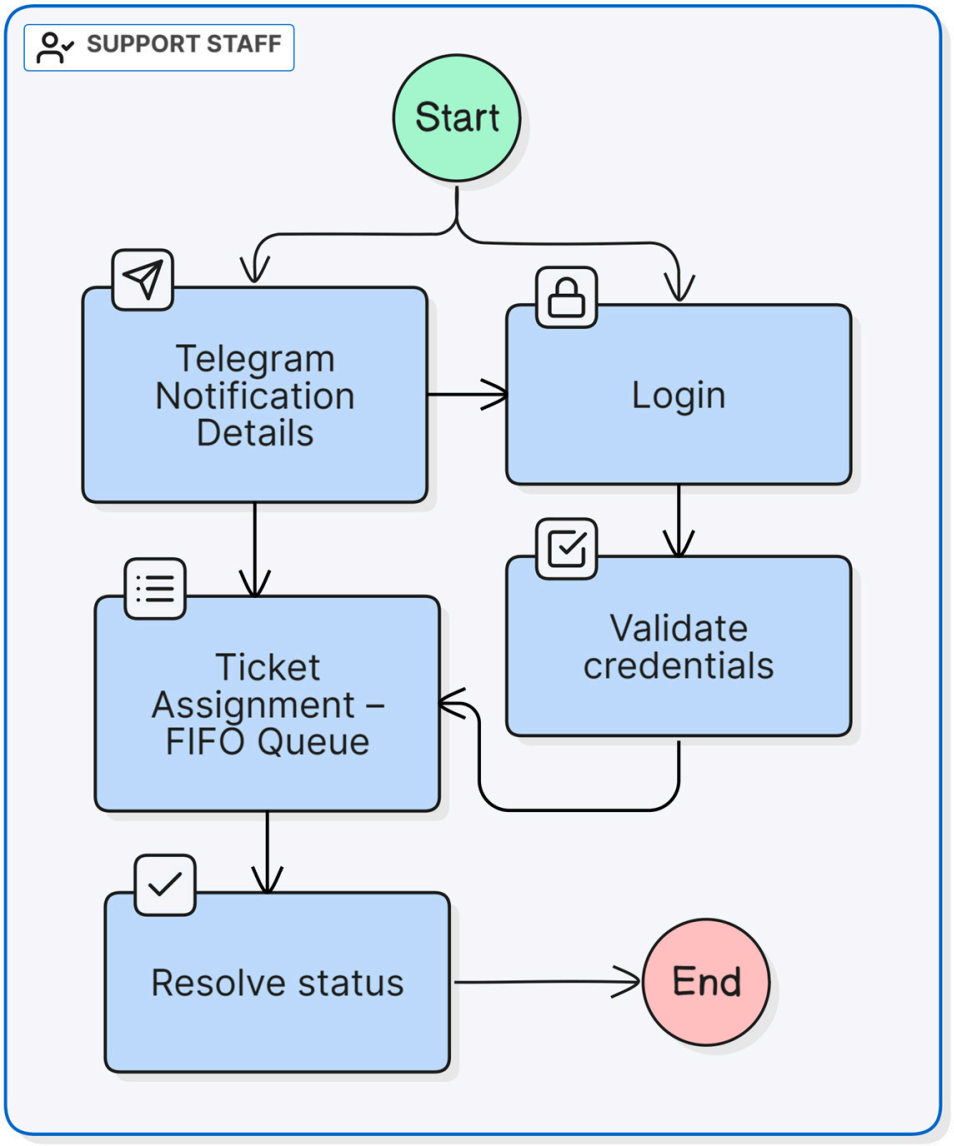


Figure 6. Technical staff workflow.

4.5. Specific Integrations

The external integrations enhance automation through low-cost APIs invoked from Laravel controllers via Guzzle HTTP, using stateless handling for privacy and asynchronous execution through queues. Both integrations record metrics in the Chatbot table (quantitative data such as messages sent, success/failure rates, model used, and timestamps), achieving an 85.48% interaction success rate during the evaluation period (January–April 2025). Errors and fallback events are logged in storage/logs/laravel.log for diagnostic purposes without exposing sensitive data [11,31].

1. Data protection and pre-submission sanitization:

The chatbot does not require authentication, nor does it request personal data, as it is oriented toward the resolution of basic technical incidents. Since users can enter free text, the system applies a sanitization layer prior to sending requests to OpenRouter, using masking rules that prevent the transmission of personal or sensitive information. In this way, only generic and anonymized technical data are sent to the language models. Additionally, prompts are not stored; only aggregated usage metrics are recorded, which reinforces user privacy and minimizes exposure risks.

https://doi.org/10.3390/computers15010047

2. OpenRouter API (Chatbot)

Operates in a stateless manner and is limited to static knowledge (FAQs about common issues such as hardware and connectivity) [10]. It is configured with automatic rotation of free models to ensure reliability and support for high token limits (~4k). The primary model is meta-llama/llama-3.3-8b-instruct:free, with fallbacks including google/gemini-2.0-flash-exp:free, deepseek/deepseek-r1-0528:free, mistralai/mistral-small-3.1:free, microsoft/phi-3.5-mini:free, minimax/minimax-m2:free, and google/gemma-3-27b-it:free [9,31]. Structured prompts guide the responses, for example:

> *"You are a friendly technical assistant for a support ticket system. Provide help only for simple and frequent issues based on this documentation. If the problem is complex or requires special credentials, suggest creating a support ticket. Always respond in Spanish, clearly, in an organized manner, and step by step."*

To reduce inconsistencies arising from the use of multiple language models, the system employs a unique and strict system prompt shared across all models accessed via OpenRouter. This prompt delimits a closed domain of authorized technical issues and establishes clear behavioral rules. This mechanism standardizes the scope and format of responses, minimizes semantic variations, and reduces the risk of incorrect answers, maintaining the artificial intelligence in an exclusively advisory role and preserving operational control of the ticketing system in the hands of authorized technical staff.

3. Telegram Bot API (Notifications):

POST-based webhooks are triggered according to ticket creation timestamps and are sent asynchronously [37]. When a ticket is generated, the system notifies the next available technician in FIFO order [12] with a message such as: "New ticket registered ID: {id}—Support: {support}—Registered by: {user}—Office: {office}—Date: {date}—View Ticket: [link button]—Assign: [action button]." From Telegram, the technician can self-assign the ticket (updating the status to "In Progress"); however, closing actions ("Resolved" or "Cancelled") must be performed through the web interface to ensure complete logging.

4.6. Cross-Cutting Aspects

The cross-cutting aspects ensured system robustness, security, and maintainability across all architectural layers.

- Security: Laravel Sanctum provides token-based authentication exclusively for technical staff. API tokens (OpenRouter/Telegram) are stored in the env file, Eloquent uses parameterized queries to prevent SQL injection, and HTTPS is enforced on the public subdomain to secure data in transit [7,14,35].
  The system applies the data minimization principle, limiting the processing of personal information to the institutional infrastructure. Integrations with external artificial intelligence services operate under a stateless scheme and process only previously sanitized technical content. This design reduces the risks associated with the use of third-party APIs and reinforces the institution's data sovereignty.
- Scalability: The system utilizes Laravel job queues to handle asynchronous notifications, ensuring stability and operational continuity by decoupling critical processes from the main service flow. Furthermore, model rotation via OpenRouter is applied to distribute the chatbot's load and mitigate specific failures in external services nuous responsiveness. OpenRouter model rotation distributes chatbot load efficiently [35]. The MariaDB database enables efficient handling of concurrent queries and transactional operations, while the institutional XAMPP environment [11] maintained stable performance during the evaluation period, with an operational load exceeding 50 daily tickets.

https://doi.org/10.3390/computers15010047

However, this evaluation did not include formal stress tests or high concurrency scenarios; therefore, the system's scalability is understood from an architectural and functional perspective. Under significantly higher loads, bottlenecks associated with database concurrency or rate limits imposed by the external APIs employed could arise. These considerations are acknowledged as a limitation of the study and are proposed as lines of future work.

- Testing and Monitoring: Unit tests executed with PHPUnit achieved 85% coverage, encompassing CRUD operations and integrations. The RESTful endpoints were validated using Postman, consistently returning successful 200 responses. Centralized logs in storage/logs/laravel.log record errors and fallback events, while ApexCharts dashboards enable real-time monitoring of system metrics [38].
- System licensing: The developed incident management system is distributed as open-source software under the MIT license, which allows its use, modification, and redistribution without licensing costs. This decision facilitates the adoption of the system by other public institutions facing budgetary constraints and promotes code reuse and adaptation in similar contexts.

4.7. Data Protection and Privacy Considerations

The chatbot does not require authentication or request personal data, as it is oriented toward resolving basic technical incidents. Since users can input free text, the system applies a sanitization layer before sending requests to OpenRouter, utilizing masking rules that prevent the transmission of personal or sensitive information. Consequently, only generic and anonymized technical data are sent to the language models. Additionally, prompts are not stored; only aggregated usage metrics are recorded, which reinforces user privacy and minimizes exposure risks.

4.8. Final Evaluation of the Proposed Architecture

In summary, the system's MVC architecture—structured into well-defined layers and supported by a relational model in MariaDB—integrates role-specific workflows and hybrid interfaces (OpenRouter and Telegram), providing an efficient and replicable technical solution for incident management in regional public-sector environments. Its low-cost design, based on free and open-source tools, together with its scalable structure leveraging asynchronous queues and FIFO processing, enables the system to handle demand fluctuations without performance degradation, thereby validating the iterative applicability of the Scrum framework in resource-constrained contexts.

5. Results

The results are presented using descriptive statistics, focusing on operational metrics derived from system logs and user surveys. Inferential statistical analysis was not conducted, as the study was designed as an applied case study aimed at evaluating system performance in a real institutional context rather than testing population-level hypotheses. The evaluation results of the web-based ticketing system validate the Laravel 10 MVC architecture, highlighting how integrations such as OpenRouter and the Telegram Bot API contributed to overall efficiency (see Table 4). The following sections detail the findings, emphasizing the system components that enabled these outcomes.

https://doi.org/10.3390/computers15010047

Table 4. Result obtained from the metrics.

Dimension	Main Metric	Result
System usability	Satisfaction	88%
AI chatbot	Interaction rate (% of queries answered)	85.48%
Mobile notifications	Open rate (% of notifications opened)	94.12% (4979/5290)
Resolution time	Average time from registration to resolution (minutes)	31 min
Number of incidents processed and resolved	Cases resolved vs. reported	99.34% (1051/1058)
Technical staff response time	Minutes from notification to initial action	11 min

5.1. System Usability

Usability was measured using the Questionnaire for Website Usability (QWU), administered to 100 participants, yielding an average satisfaction score of 88%, with mean ratings of 4.4/5 for navigability and 4.3/5 for interface efficiency. This high performance is attributed to the responsive Bootstrap 5 frontend, which enabled intuitive, authentication-free access for end users, and to Laravel Blade views that rendered streamlined workflows (ticket submission and tracking), reducing friction across mobile devices and geographically dispersed offices.

5.2. Performance of the AI Chatbot

The chatbot, integrated via the OpenRouter API, processed 1364 interactions, automatically answering 1166 queries (an interaction rate of 85.48%), while the remaining cases were forwarded to the Laravel backend for complex ticket handling. Quantitative metrics were recorded in the Chatbot table, confirming an overall success rate of 85.48% (see Figure 7). Performance improvements resulted from the rotation of free models (primary meta-llama/llama-3.3-8b-instruct:free and fallbacks such as google/gemini-2.0-flash-exp:free), combined with structured prompts restricted to FAQs, which minimized errors and enabled fast stateless responses without imposing database load.

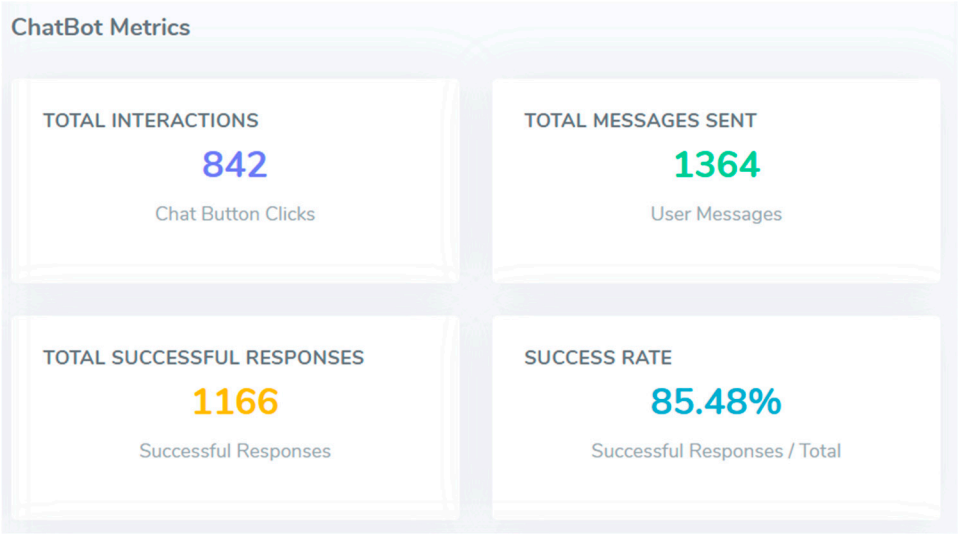


Figure 7. Dashboard view displaying AI chatbot metrics for the January–April 2025 period. The numerical values are color-coded as part of the interface design to improve readability and visual differentiation among chatbot performance metrics. Blue is used for general interaction indicators, green highlights activity-related metrics such as total messages sent, and orange emphasizes outcome-related metrics, including successful responses and success rate.

https://doi.org/10.3390/computers15010047

5.3. Effectiveness of Mobile Notifications

Through the Telegram Bot API, a total of 5290 notifications were sent to the technical team, of which 4979 were opened, representing an open rate of 94.12% (see Figure 8). This high effectiveness facilitated FIFO assignment directly from the app or the web interface through asynchronous webhooks. The performance is attributed to the implementation of Laravel Queue jobs for message dispatching and chronological ordering by creation timestamp, which enabled immediate interventions and reduced delays in the ticket queue, integrating seamlessly with the administrator module.

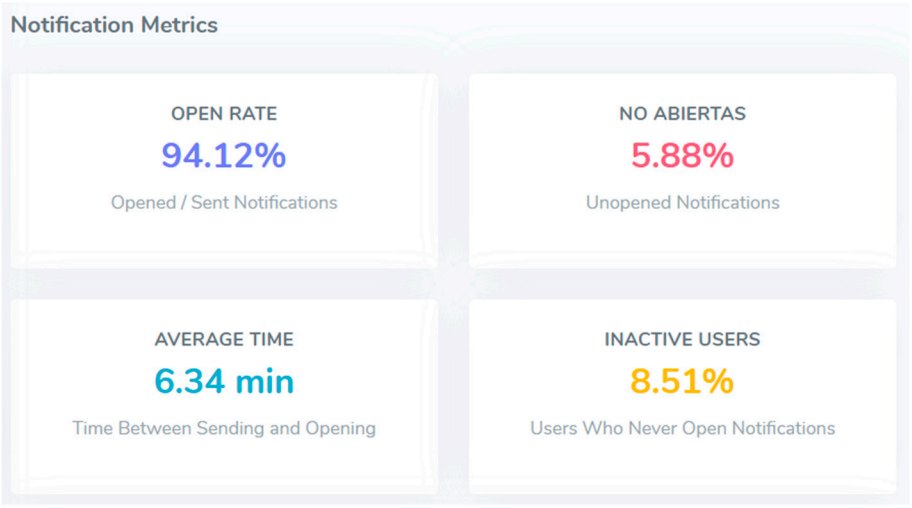


Figure 8. Dashboard view displaying mobile notification metrics for the January–April 2025 period. The colors used to display the numerical values are part of the visual design and are intended to facilitate rapid interpretation of the metrics. Blue is used for positive performance indicators (e.g., open rate and average response time), while orange highlights metrics that require attention, such as inactive users. Red is used to emphasize negative outcomes, such as unopened notifications.

5.4. Incident Handling Time

The average incident handling time decreased from 120 min (pre-implementation, based on 2024 historical reports) to 31 min post-implementation, as shown in Figure 9, derived from 1058 tickets processed in MariaDB 10.4.32. This 74.17% improvement was driven by the Laravel 10 backend layer with Eloquent ORM, which optimized queries for state traceability from ticket creation to resolution.

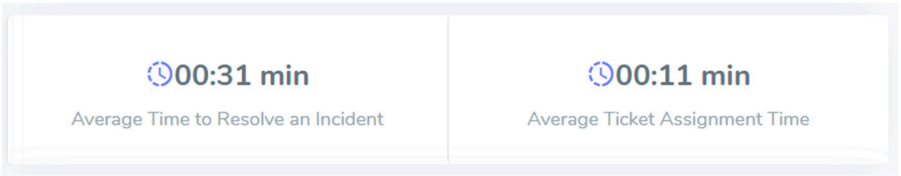


Figure 9. Dashboard view displaying incident handling and technical staff response time metrics for the January–April 2025 period.

5.5. Technical Staff Response Time

The initial response time of the technical staff—measured from the moment the notification was received to the first recorded action—averaged 11 min (see Figure 9). This improvement is attributed to mobile notifications delivered through the Telegram Bot API, which trigger chronological push alerts based on creation timestamps, accelerating FIFO assignment in the technical panel and minimizing delays in interventions.

https://doi.org/10.3390/computers15010047

5.6. Incident Resolution

Of the 1058 incidents recorded, 1051 were resolved, representing a resolution rate of 99.34% (see Figure 10), while the remaining 0.66% remained pending or was cancelled due to external constraints, managed through Laravel controllers. The high resolution rate is attributed to the relational model (Ticket/Assignment entities), which enabled full traceability and efficient updates within the technical panel using ApexCharts, allowing for prioritization and rapid closure without duplications.

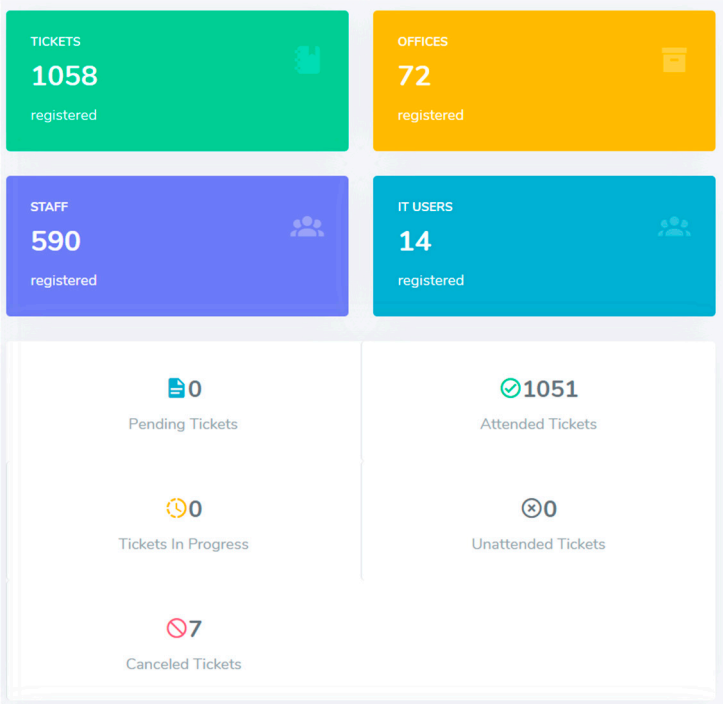


Figure 10. Dashboard view displaying incident resolution metrics for the January–April 2025 period.

In summary, the system addresses inefficiencies in the GORE Apurímac (74.17% reduction in handling time, 99.34% resolution rate, 88% usability), catalyzing inclusive digital transformation through a hybrid architecture that reinforces Peru's agile governance objectives.

6. Discussion

The implementation of the web-based ticketing system with an AI chatbot and mobile notifications at GORE Apurímac marks a clear technical transformation. It validates the MVC architecture as an effective framework for improving incident management, directly addressing the research question regarding optimization of handling times and resolution rates.

The most significant improvement is the reduction in incident handling time from 120 to 31 min (a 74.17% improvement), driven by the OpenRouter-based chatbot (85.48% automation, with 1166 responses out of 1364 queries) and Telegram notifications (94.12% open rate, 4979 of 5290 sent). Together, these accelerated technical responses to an average of 11 min, reducing bottlenecks associated with manual channels [3]. This surpasses national studies such as Ramos Herrera (2022), which achieved only a 30 s reduction per ticket using osTicket [21] and Mali et al. (2025), which reported ~80% automation using commercial NLP tools [22], owing to the use of free rotating models and asynchronous webhooks [9,10]. It also aligns with findings from Karimi et al. (2024) on hybrid educational chatbots [23], but with greater impact due to regional geographic dispersion.

https://doi.org/10.3390/computers15010047

The resolution rate of 99.34% (1051 of 1058 incidents) and usability score of 88% (QWU, with mean scores of 4.4/5 for navigability) demonstrate strong prioritization and tracking capabilities enabled by MariaDB traceability and dashboard monitoring. These results are consistent with Merino Morillo (2018) [24], where 100% of users supported digital systems, and Scotti and Carman (2024), who reported ~85% efficiency in LLM-supported environments [39].

These findings extend previous work by validating AI within the context of Peru's digital divides [40]. In computing, they expand upon Ramos Herrera's contributions [21,35] by integrating Scrum with ethical LLMs. In public administration, the results support the interoperability goals of the 2023–2025 Digital Government Plan, aimed at reducing bureaucratic burden [2].

The use of external artificial intelligence services improves operational efficiency but introduces relevant considerations regarding privacy and data sovereignty in public environments. In the proposed architecture, models are accessed through OpenRouter as an intermediary via stateless APIs, without session-level information persistence. The system applies data minimization and sanitization principles to prevent the transmission of personal information; however, due to free-text input and dependence on third-party policies, a residual risk remains. In this context, the use of external APIs is acknowledged as a limitation, and evaluating on-premise solutions or institutional data loss prevention mechanisms is proposed as future work to strengthen information sovereignty.

Complementarily, although the rotation of multiple language models may introduce variations in response style or phrasing, in the proposed architecture, this risk is mitigated through the use of a restrictive system prompt and a closed knowledge domain. Nevertheless, it is recognized that natural language generation involves a residual risk of incomplete or imprecise responses inherent to LLM-based systems. For this reason, the solution adopts a conservative approach, where the AI does not make operational decisions, and any case outside defined patterns is routed to human technical staff.

The comparison between phases presents an asymmetry in measurement instruments, as pre-implementation time is based on non-automated estimations, while post-implementation utilizes precise system logs. Although this may introduce bias, the observed reduction allows for the assertion of a significant improvement in operational efficiency.

It should be noted that the comparison between pre- and post-implementation phases relies on different measurement instruments. While post-implementation times were captured automatically through system logs, the baseline was estimated through qualitative and documentary triangulation. This asymmetry may introduce measurement bias; therefore, the reported improvements should be interpreted as indicative of substantial efficiency gains rather than as precise causal effects.

Given the use of non-probabilistic convenience sampling, the findings of this study should be interpreted as context-specific to the Regional Government of Apurímac. While the results provide strong evidence of local effectiveness, they do not support broad generalization to all public sector institutions without further replication.

7. Conclusions and Recommendations

7.1. Conclusions

The development and implementation of the web-based ticketing system with an AI chatbot and mobile notifications at the GORE validate its architecture as an innovative solution for managing technical IT incidents. The results demonstrate significant technical impacts: a 74.17% reduction in incident handling time (from 120 to 31 min), a 99.34% resolution rate, an 88% usability score (QWU), an 85.48% chatbot interaction rate, and

https://doi.org/10.3390/computers15010047

a 94.12% notification open rate. These outcomes directly address the research question, showing how integrations such as OpenRouter (with rotation of free models for stateless FAQ responses) and the Telegram Bot API (FIFO webhooks based on creation timestamps) alleviate manual saturation while enabling relational traceability in MariaDB 10.4.32.

The project achieved its overall objective of improving incident management, as well as specific goals related to reducing response times, increasing resolution rates, and optimizing technical response efficiency. Implemented under the Scrum methodology, using Laravel 10, Bootstrap 5, and visualization tools such as ApexCharts, the system establishes itself as a scalable, low-cost model for public sector digital transformation, aligned with the 2023–2025 Digital Government Plan [3]. Ultimately, it promotes agile, measurable, and inclusive governance across regional contexts.

7.2. Recommendations

To maximize impact: Train technical personnel (five staff members) on AI updates and Laravel queue management to ensure long-term sustainability and integrate predictive machine learning into MariaDB to identify recurrent failures based on chatbot logs.

For future research: Conduct longitudinal studies (12 months) with randomized samples to enhance causal inference and perform comparative analyses across Peruvian regions to assess replicability. These efforts would further strengthen contributions to Peru's digital public administration.

Author Contributions: Conceptualization, H.R.; methodology, H.R.; software, L.A.P.A. and A.M.B.; validation, L.A.P.A. and A.M.B.; investigation, H.R., L.A.P.A. and A.M.B.; resources, H.R., L.A.P.A. and A.M.B.; writing—original draft preparation, H.R.; writing—review and editing, H.R., L.A.P.A. and A.M.B.; project administration, L.A.P.A. and A.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The project's source code can be downloaded from the following address: https://github.com/AlbertPF/tickets (accessed on 28 December 2025).

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

References

1. Comex, P. Transformación Digital Del Gobierno: Retos y Compromisos No Atendidos. Available online: https://www.comexperu.org.pe/articulo/transformacion-digital-del-gobierno-retos-y-compromisos-no-atendidos (accessed on 26 October 2025).
2. Regional, G.G. Reporte de Avance Plan de Gobierno Digital 2023–2025-Informes y Publicaciones-Gobierno Regional Tacna-Plataforma Del Estado Peruano. Available online: https://www.gob.pe/institucion/regiontacna/informes-publicaciones/7256244-reporte-de-avance-plan-de-gobierno-digital-2023-2025 (accessed on 26 October 2025).
3. Sangama-Reyna, E.J. Transformación Digital y Gobernanza Regional En Perú: Revisión Sistemática. *Gest. Prod. Rev. Electrónica Cienc. Gerenciales* **2025**, *7*, 91–106. [CrossRef]
4. Sánchez Casanova, F.S.; Valles Coral, M.Á. Influencia de ITIL V3 En La Gestión de Incidencias de Una Municipalidad Peruana. *Rev. Cuba. Cienc. Informáticas* **2021**, *15*, 1–19.
5. Sub Gerencia de Desarrollo Institucional Estadistica e Informatica. *Reporte de Dependencias y Personal Del GORE-Apurimac Sede Central 2024. Internal Report (Not Publicly Available)*; Gobierno Regional de Apurímac: Abancay, Peru, 2024.
6. Unidad de Informática, Gobierno Regional Apurímac. *Informe Situacional de La Oficina de Soporte Tecnico 2024. Internal Report (Not Publicly Available)*; Gobierno Regional de Apurímac: Abancay, Peru, 2024.
7. Subecz, Z. Web-Development with Laravel Framework. *Gradus* **2021**, *8*, 211–218. [CrossRef]
8. Rajabovich, M.B. Analysis and Advantages of Laragon Software in Data Processing. *J. Innov. Creat. Art* **2023**, *2*, 2023.
9. Petrosyan, A.S.; Surmachevskaya, A.A.; Novichkov, A.A.; Lukin, D.G.; Gareeva, G.A. Automatic Notification of Academic Debts Based on Telegram. *Int. J. Adv. Stud.* **2023**, *13*, 163–179. [CrossRef]

10. Khennouche, F.; Elmir, Y.; Himeur, Y.; Djebari, N.; Amira, A. Revolutionizing Generative Pre-Traineds: Insights and Challenges in Deploying ChatGPT and Generative Chatbots for FAQs. *Expert Syst. Appl.* **2024**, *246*, 123224. [CrossRef]
11. Sinaga, T.M.; Zuhdi, A.; Santoso, G.B. MVC Implementation in Laravel Framework for Development Web-Based E-Commerce Applications. *Intelmatics* **2021**, *6*, 37–42.
12. Fauzi, M.F.; Rahmi, A.N. Penerapan Metode First in First out (FIFO) Dalam Sistem Antrian Pelayanan Administrasi Mahasiswa Studi Kasus DAAK Universitas AMIKOM Yogyakarta. *METHOMIKA J. Manaj. Inform. dan Komputerisasi Akunt.* **2021**, *5*, 183–188. [CrossRef]
13. Aziz, N.S.; Kamaludin, A. Using Pre-Test to Validate the Questionnaire for Website Usability (QWU). In Proceedings of the 2015 4th International Conference on Software Engineering and Computer Systems, ICSECS 2015: Virtuous Software Solutions for Big Data, Kuantan, Malaysia, 19–21 August 2015; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2015; pp. 107–111.
14. GitHub. GitHub—Laravel-Notification-Channels/Telegram: Telegram Notifications Channel for Laravel. Available online: https://github.com/laravel-notification-channels/telegram (accessed on 12 November 2025).
15. Castro, P.; Cambarieri, M. Estrategias Para Desacoplar Funcionalidades y Facilitar El Desarrollo De Software En Instituciones Públicas: El Caso de La Universidad Nacional de Río Negro. In Proceedings of the Workshop de Investigadores en Ciencias de la Computación 2025, Mendoza, Argentina, 10–11 April 2025; pp. 10–11.
16. Soel Juarez, M.A. *Aplicación de Un Sistema Web Help Desk Basado En ITIL Para El Registro y Control de Requerimientos e Incidencias Al Personal de La Corte Superior de Justicia de Apurímac, Año 2020—2023*; Universidad Nacional Micaela Bastidas de Apurímac: Abancay, Peru, 2024.
17. Firdausi, S.; Setiawan, M.A. ITIL v3 Framework Application to Design Information Technology Incident Management Governance. *J. Ilm. Tek. Elektro Komput. Dan Inform.* **2022**, *8*, 128. [CrossRef]
18. Ayquipa, R.A.R.; Enriquez, H.R.; Huayllani, W.J.A.; Mezarina, Z.H.A.; Cabrera, M.J.I. Challenges in the Implementation of E-Government for Public Institutions in Peru. In Proceedings of the 10th International Conference on E-Education, E-Business, E-Management and E-Learning, Tokyo, Japan, 10–13 January 2019; pp. 347–351. [CrossRef]
19. Babatunde Johnson, O.; Olamijuwon, J.; Cadet, E.; Olajide Soji, O.; Oke Ekpobimi, H. Building a Microservices Architecture Model for Enhanced Software Delivery, Business Continuity and Operational Efficiency. *Int. J. Front. Eng. Technol. Res.* **2024**, *7*, 70–81. [CrossRef]
20. Setälä, M.; Abrahamsson, P.; Mikkonen, T. Elements of Sustainability for Public Sector Software—Mosaic Enterprise Architecture, Macroservices, and Low-Code. In Proceedings of the 12th International Conference, ICSOB 2021, Drammen, Norway, 2–3 December 2021; Volume 434 LNBIP, pp. 3–9.
21. Agustin Ramos Herrera Asesor, J.; Leonardo José Torres Argomedo, M. *Implementación de OsTicket Para Optimizar La Gestión de Incidencias Del Área de Infraestructura de La Corte Superior de Justicia de Lima*; Universidad Privada del Norte: Trujillo, Peru, 2022.
22. Mali, D.; Salhotra, J.; Waikar, D.; Jadhav, N.; Tamboli, S. Online Chatbot Based Ticketing System. *Int. J. Multidiscip. Res.* **2025**, *7*, 2–9. [CrossRef]
23. Karimi, Z.; Gallipoli, G.; Cagliero, L.; Chicco, F.; Rosa, C.; Sechi, A. Empowering University-Level Help Desk for International Applicants with AI Chatbots. In Proceedings of the 2024 IEEE International Conference on Big Data, BigData, Washington, DC, USA, 15–18 December 2024; Institute of Electrical and Electronics Engineers Inc.: Washington, DC, USA, 2024; pp. 5263–5270.
24. Merino Morillo, V.M. *Implementación de Un Sistema de Gestión de Incidencias Para La Empresa BEMAST EIRL*; Universidad Católica Los Ángeles de Chimbote: Chimbote, Peru, 2019.
25. Kanaki, K.; Kalogiannakis, M. Sample Design Challenges: An Educational Research Paradigm. *Int. J. Technol. Enhanc. Learn.* **2023**, *15*, 266–285. [CrossRef]
26. Formularios de Google 1. Encuesta de Usabilidad Del Sistema Web de Tickets. Available online: https://docs.google.com/forms/d/e/1FAIpQLSccIZwiStmwEG863nl9Eblv1yLmMOwI6JdVjNxOq44Pqjwzdw/viewform(accessed on 12 September 2025).
27. Avilés Matute, S.; Avila-Pesantez, D.; Avila, M. Desarrollo de Sistema Web Basado En Los Frameworks de Laravel y VueJs, Para La Gestión Por Procesos: Un Estudio de Caso. *Rev. Peru. Comput. Sist.* **2020**, *3*, 3–10. [CrossRef]
28. Ameta, U.; Patel, M.; Sharma, A.K. Scrum Framework Based on Agile Methodology in Software Development and Management. In *Emerging Trends in Data Driven Computing and Communications*; Springer: Singapore, 2021; pp. 321–332. [CrossRef]
29. Sommerville, I. *Software Engineering*, 10th ed.; Pearson Educación: London, UK, 2016; ISBN 978-1-292-09613-1.
30. Pressman, R.S.; Maxim, B.R. *Software Engineering: A Practitioner's Approach*; Hill, M., Ed.; McGraw-Hill Education: Columbus, OH, USA, 2019; ISBN 1259872971.
31. Bass, L.; Clements, P.; Kazman, R. *Software Architecture in Practice*; Addison-Wesley Professional: Boston, MA, USA, 2012; ISBN 978-0-321-81573-6.
32. Salah, S.; Maciá-Fernández, G.; Díaz-Verdejo, J.E. Fusing Information from Tickets and Alerts to Improve the Incident Resolution Process. *Inf. Fusion* **2019**, *45*, 38–52. [CrossRef]

33. Bootstrap Bootstrap 5 Introduction. Available online: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (accessed on 15 August 2025).
34. Torres, M.Á. Desarrollo de Aplicaciones Web Con PHP y MySQL. In *Editorial MACRO*; Marcombo, S.A.: Barcelona, Spain, 2020; p. 344.
35. OpenRouter OpenRouter Quickstart Guide | Developer Documentation | OpenRouter | Documentation. Available online: https://openrouter.ai/docs/quickstart (accessed on 9 November 2025).
36. Vargas De la Maza, K.E. *Propuesta de Sistema de Control de Ticket Para La Gestión de Negocios y Seguridad: Caso Estadio Quisqueya, 2017*; Universidad APEC: Santo Domingo, Dominican Republic, 2018.
37. Telegram Messenger Inc. Bots: An Introduction for Developers. Available online: https://core.telegram.org/bots (accessed on 4 August 2025).
38. Almasi, S.; Bahaadinbeigy, K.; Ahmadi, H.; Sohrabei, S.; Rabiei, R. Usability Evaluation of Dashboards: A Systematic Literature Review of Tools. *Biomed Res. Int.* **2023**, *2023*, 11. [CrossRef] [PubMed]
39. Scotti, V.; Carman, M.J. LLM Support for Real-Time Technical Assistance. In *Machine Learning and Knowledge Discovery in Databases. Research Track and Demo Track*; Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer: Cham, Switzerland, 2024; Volume 14948 LNAI, pp. 388–393. [CrossRef]
40. El Panorama de La IA En Perú: Transformación, Adopción y Su Impacto Futuro | COEM. Available online: https://blog.controlesempresariales.com/el-panorama-de-la-ia-en-peru-transformacion-adopcion-y-su-impacto-futuro (accessed on 26 October 2025).