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

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

Posted Date: 11 December 2025

doi: 10.20944/preprints202512.1056.v1

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

 doi:10.20944/preprints202512.1056.v1

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, Perú

* 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 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 minutes (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 minutes, the accumulation of incidents 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].

 doi:10.20944/preprints202512.1056.v1

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].

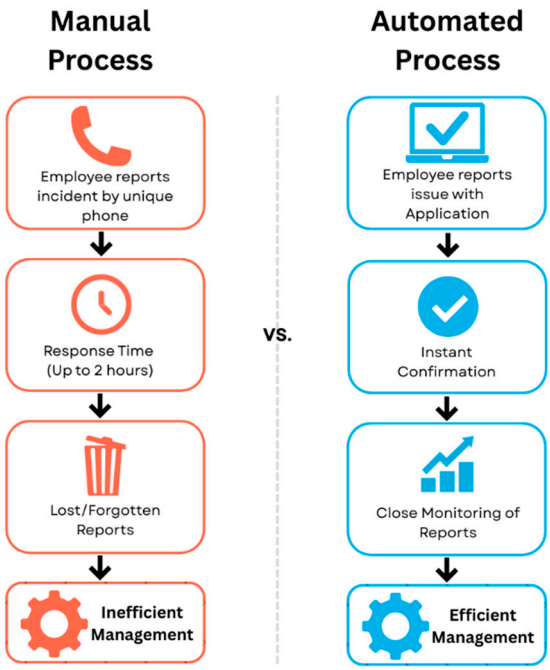


Figure 1. Flowchart of Current vs. Proposed Problem/Solution.

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

 doi:10.20944/preprints202512.1056.v1

operational risks and allows faster adaptation to regulatory changes [15]. Their incorporation into technical incident management also contributes to increased process efficiency [4, 2, 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 seconds 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%. Merino Morillo (2018), who 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.

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.

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) [25][13] with a Likert scale consisting of 21 items that evaluated navigability and efficiency, and automatic system logs used to collect operational metrics. Table 1 summarizes the main instruments and their applications.

Table 1. 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]

 doi:10.20944/preprints202512.1056.v1

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 [26, 27]. 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.

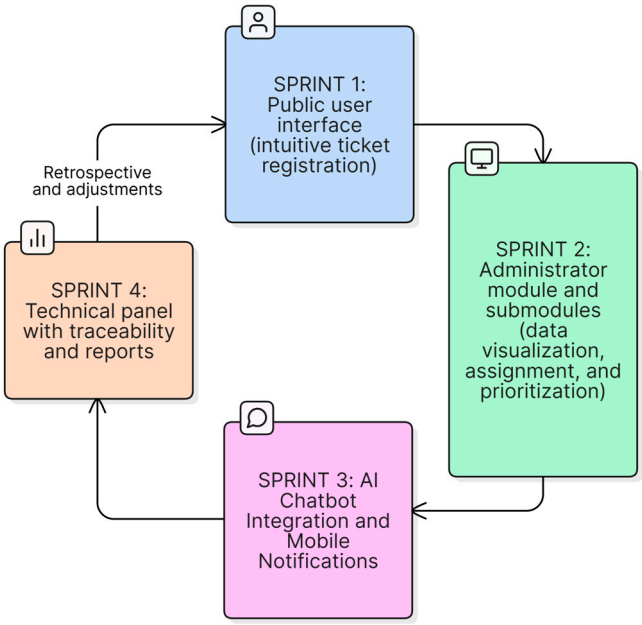


Figure 2. Scrum methodology workflow, with emphasis on the incremental development of interfaces, AI functionalities, and mobile notifications.

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

- **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.

 doi:10.20944/preprints202512.1056.v1

- **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.

4. System 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 [26, 7]. 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, 28]. During the evaluation period, more than 1,050 tickets were processed without performance issues. This architecture ensures relational traceability through MySQL, minimal API latency, and low operational cost.

4.1. Developed Architecture

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 devices, incorporating components such as chatbot modals and tables for tracking ticket statuses (pending/in progress/resolved/cancelled) [29, 11].
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 [30, 11]. External integrations, including OpenRouter (stateless, via Guzzle HTTP) [31] 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].

 doi:10.20944/preprints202512.1056.v1

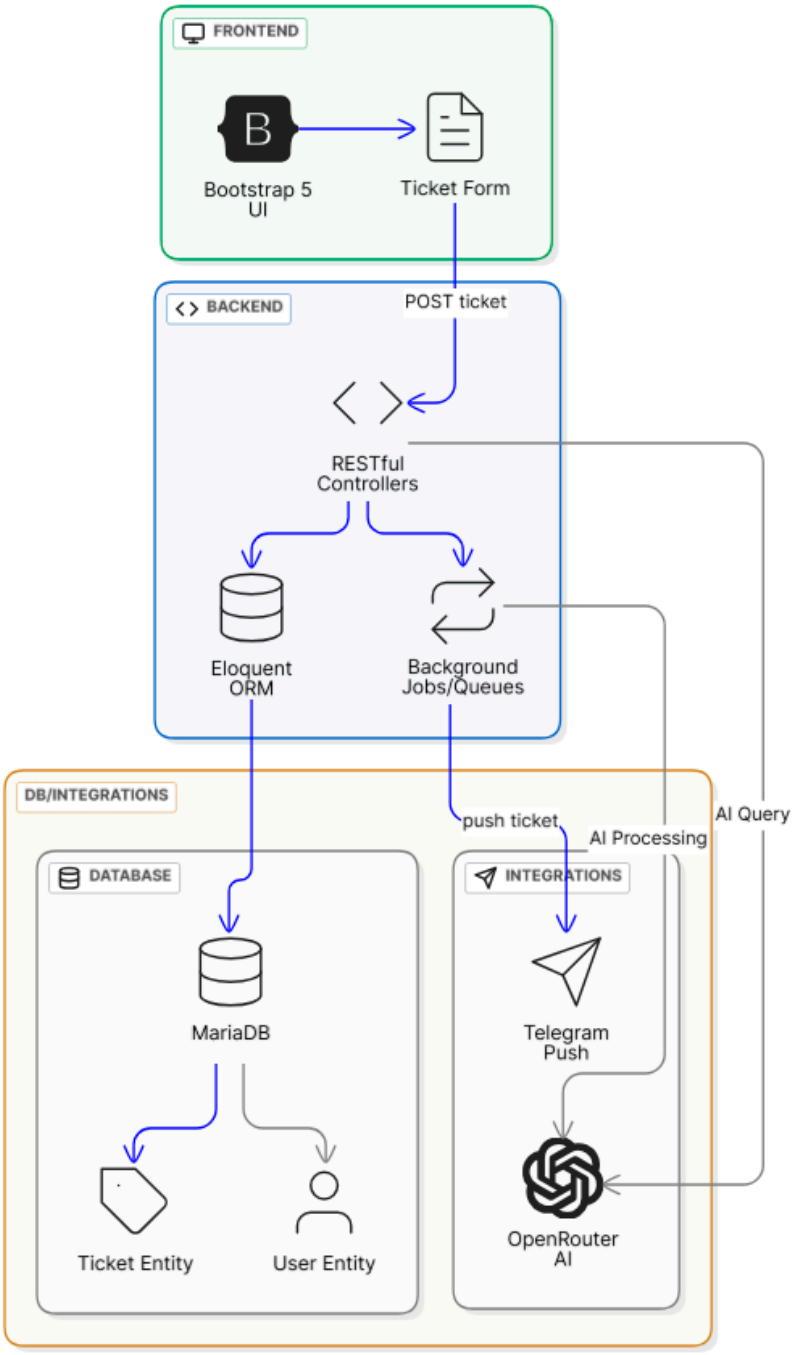


Figure 3. Architecture diagram (Bootstrap frontend → Laravel backend → MariaDB database → integrations), highlighting data flows.

4.2. Data Model

The data model was implemented in MariaDB 10.4.32 (integrated within XAMPP 8.2.12) [30], using a normalized relational schema to ensure data integrity and efficiency in CRUD operations [32]. 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.

 doi:10.20944/preprints202512.1056.v1

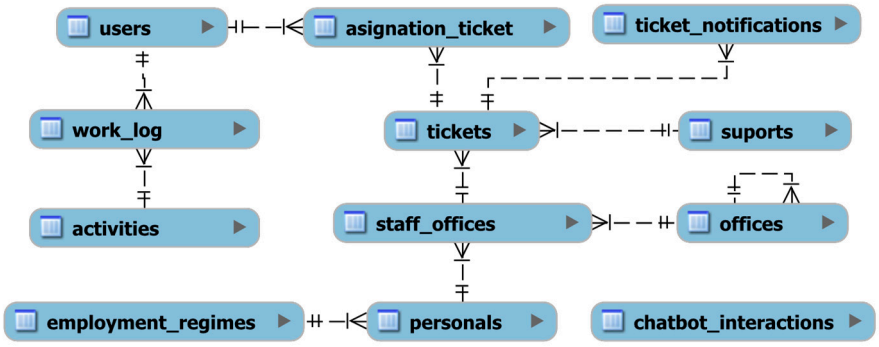


Figure 4. ER diagram of the data model.

4.3. 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.

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.

 doi:10.20944/preprints202512.1056.v1

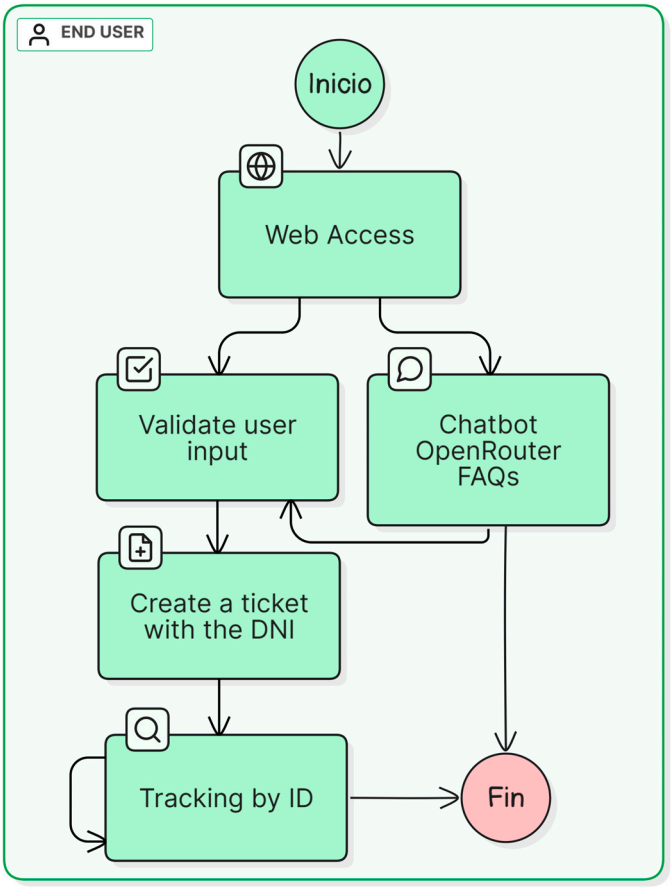


Figure 5. End-user workflow.

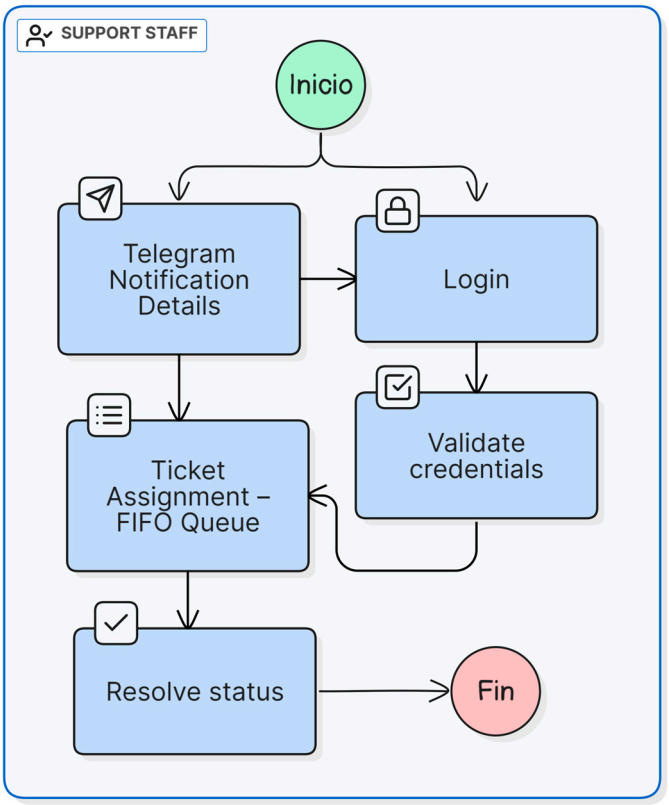


Figure 6. Technical staff workflow.

 doi:10.20944/preprints202512.1056.v1

4.4. 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. **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 [31][9]. 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."

2. **Telegram Bot API (Notifications):**

POST-based webhooks are triggered according to ticket creation timestamps and are sent asynchronously [33]. 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.

4.5. 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].
- **Scalability**: The system uses Laravel job queues to handle asynchronous notifications, ensuring stability and continuous responsiveness. OpenRouter model rotation distributes chatbot load efficiently [31]. The MariaDB database supports concurrent queries without locking, and the XAMPP environment [11] maintains stable performance even with more than 50 tickets per day.
- **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 [34].

4.6. 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,

 doi:10.20944/preprints202512.1056.v1

thereby validating the iterative applicability of the Scrum framework in resource-constrained contexts.

5. Results

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 2). The following sections detail the findings, emphasizing the system components that enabled these outcomes.

Table 2. 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% (4,979/5,290)
Resolution time	Average time from registration to resolution (minutes)	31 minutes
Number of incidents processed and resolved	Cases resolved vs. reported	99.34% (1,051/1,058)
Technical staff response time	Minutes from notification to initial action	11 minutes

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 1,364 interactions, automatically answering 1,166 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 Fig. 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.

 doi:10.20944/preprints202512.1056.v1

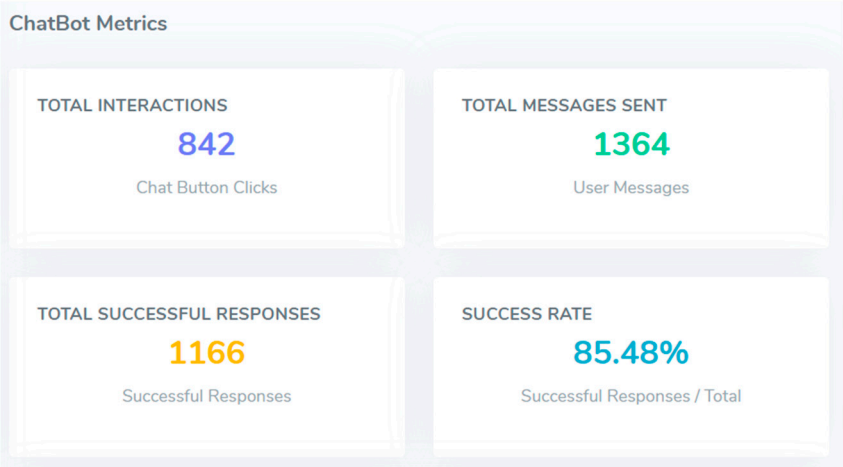


Figure 7. Dashboard view displaying AI chatbot metrics for the January–April 2025 period.

5.3. Effectiveness of Mobile Notifications

Through the Telegram Bot API, a total of 5,290 notifications were sent to the technical team, of which 4,979 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.

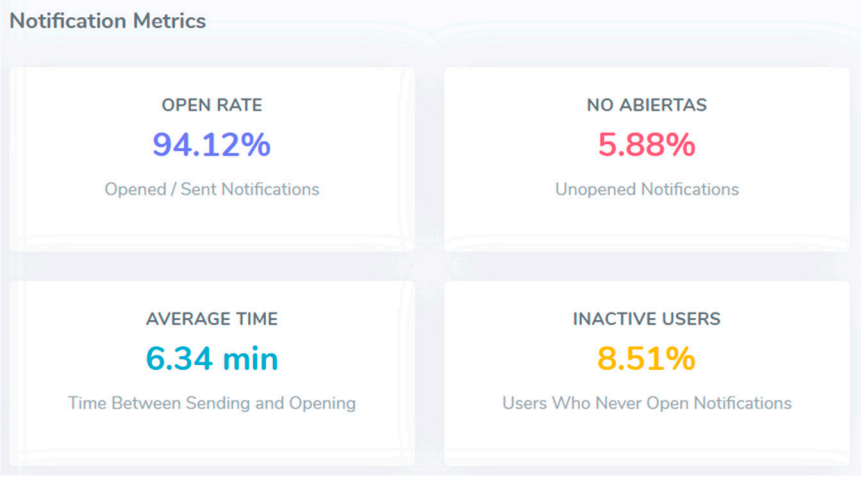
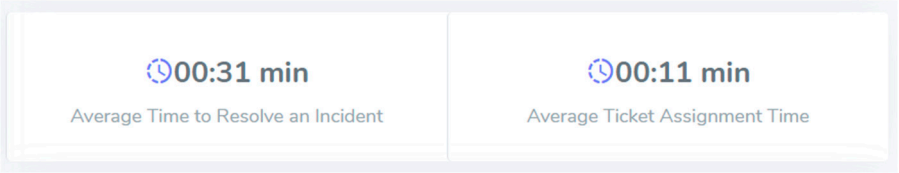


Figure 8. Dashboard view displaying mobile notification metrics for the January–April 2025 period.

5.4. Incident Handling Time

The average incident handling time decreased from 120 minutes (pre-implementation, based on 2024 historical reports) to 31 minutes post-implementation, as shown in Figure 9, derived from 1,058 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.



 doi:10.20944/preprints202512.1056.v1

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 minutes (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.

5.6. Incident Resolution

Of the 1,058 incidents recorded, 1,051 were resolved, representing a resolution rate of 99.34% (see Fig. 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.

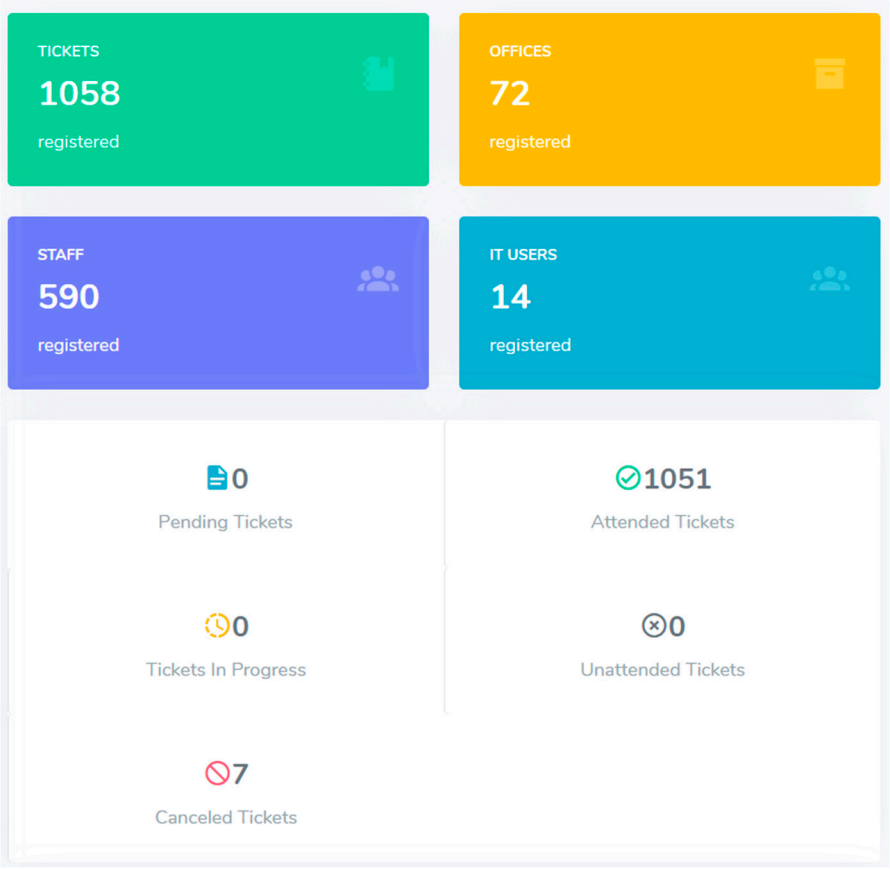


Figure 10. Dashboard view displaying incident resolution metrics for the January–April 2025 period.

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

 doi:10.20944/preprints202512.1056.v1

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 minutes (a 74.17% improvement), driven by the OpenRouter-based chatbot (85.48% automation, with 1,166 responses out of 1,364 queries) and Telegram notifications (94.12% open rate, 4,979 of 5,290 sent). Together, these accelerated technical responses to an average of 11 minutes, reducing bottlenecks associated with manual channels [3]. This surpasses national studies such as Ramos Herrera (2022), which achieved only a 30-second 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.

The resolution rate of 99.34% (1,051 of 1,058 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), where 100% of users supported digital systems [24], and Scotti and Carman (2024), who reported ~85% efficiency in LLM-supported environments [35]

These findings extend previous work by validating AI within the context of Peru's digital divides [36]. 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].

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 minutes), a 99.34% resolution rate, an 88% usability score (QWU), an 85.48% chatbot interaction rate, and 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.

Supplementary Materials: The following supporting information can be downloaded at: https://github.com/AlbertPF/tickets.

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.; 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

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

References

1. COMEX, P. Transformación digital del Gobierno: retos y compromisos no atendidos Disponible en: https://www.comexperu.org.pe/articulo/transformacion-digital-del-gobierno-retos-y-compromisos-no-atendidos.
2. Regional, G.G. Reporte de Avance Plan de Gobierno Digital 2023-2025 - Informes y publicaciones - Gobierno Regional Tacna - Plataforma del Estado Peruano.
3. Sangama-Reyna, E.J. Transformación digital y gobernanza regional en Perú: Revisión sistemática. *Gest. Prod. Rev. Electrónica Ciencias Gerenciales* **2025**, *7*, 91-106, doi:10.35381/GEP.V7I13.329.
4. Sánchez Casanova, F.S.; Valles Coral, M.Á. Influencia de ITIL V3 en la gestión de incidencias de una municipalidad peruana. *Univ. las Ciencias Informaticas* **2021**.
5. Sub Gerencia de Desarrollo Institucional, E. e I. *Reporte de Dependencias y Personal del GORE-Apurimac Sede Central 2024.*; Abancay, 2024;
6. Unidad de Informatica *Informe Situacional de la Oficina de Soporte Tecnico 2024*; Abancay, 2024;
7. Subecz, Z. Web-development with Laravel framework. *Gradus* **2021**, *8*, 211-218, doi:10.47833/2021.1.csc.006.
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, doi:10.12731/2227-930X-2023-13-3-163-179.
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, doi:10.1016/J.ESWA.2024.123224.
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 Amikon Yogyakarta. *METHOMIKA J. Manaj. Inform. dan Komputerisasi Akunt.* **2021**, *5*, 183-188, doi:10.46880/jmika.vol5no2.pp183-188.
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; Institute of Electrical and Electronics Engineers Inc., noviembre 2015; pp. 107-111.
14. GitHub GitHub - laravel-notification-channels/telegram: Telegram Notifications Channel for Laravel.
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; 2025.
16. Soel Juarez, M.A.; Rojas, H. 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, 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, doi:10.26555/jiteki.v8i1.23632.
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. *PervasiveHealth Pervasive Comput. Technol. Healthc.* **2019**, 347-351, doi:10.1145/3306500.3306572.
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, doi:https://doi.org/10.53294/ijfetr.2024.7.2.0050.
20. Setälä, M.; Abrahamsson, P.; Mikkonen, T. Elements of Sustainability for Public Sector Software Mosaic Enterprise Architecture, Macroservices, and Low-Code. *12th Int. Conf. ICSOB 2021* **2021**, doi:10.1007/978-3-030-91983-2_1.
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, 2022. *Univ. Priv. del Norte* **2022**.
22. Mali, D.; Salhotra, J.; Waikar, D.; Jadhav, N.; Tamboli, S. Online Chatbot Based Ticketing System. *Int. J. Multidiscip. Res.* **2025**, *7*, doi:10.36948/ijfmr.2025.v07i02.40692.
23. Karimi, Z.; Gallipoli, G.; Cagliero, L.; Chicco, F.; Rosa, C.; Sechi, A. Empowering University-level Help Desk for International Applicants with AI Chatbots. *Proc. - 2024 IEEE Int. Conf. Big Data, BigData 2024* **2024**, 5263-5270, doi:10.1109/BIGDATA62323.2024.10825040.
24. Merino Morillo, V.M. Implementación de un sistema de gestión de incidencias para la empresa BEMAST E.I.R.L – Chimbote; 2018. *Univ. Católica Los Ángeles Chimbote* **2019**.
25. Google Forms Formularios de Google. *Encuesta sobre Café (Coffea Arab.* 2021, 588-588.
26. 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. y Sist.* **2020**, *3*, 3-10, doi:10.15381/rpcs.v3i2.19256.
27. Ameta, U.; Patel, M.; Sharma, A.K. Scrum Framework Based on Agile Methodology in Software Development and Management. **2021**, 321-332, doi:10.1007/978-981-16-3915-9_28.
28. 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, doi:10.1016/J.INFFUS.2018.01.011.
29. GetBoostrap.com *Bootstrap 5 introduction*; 2023;
30. Torres, M.Á. Desarrollo de aplicaciones web con PHP y MySQL. *Editor. MACRO* **2020**, 344.
31. OpenRouter Quickstart Guide | Developer Documentation | OpenRouter | Documentation.
32. 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: República Dominicana, 2018.
33. Telegram Messenger Inc. Bots: An introduction for developers.
34. 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*, doi:10.1155/2023/9990933.
35. Scotti, V.; Carman, M.J. LLM Support for Real-Time Technical Assistance. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* **2024**, *14948 LNAI*, 388-393, doi:10.1007/978-3-031-70371-3_26.
36. El panorama de la IA en Perú: Transformación, adopción y su impacto futuro | COEM.