TECHNICAL ARCHITECTURE DOCUMENT

Details of the architecture and the main technical choices made

Your contacts SCUB :

Stéphane Traumat
Managing partner
stephane.traumat@scub.net
+33 6 77 68 96 85

Pier-Jean Malandrino
Chief Technology Officer
pierjean.malandrino@scub.net
+33 6 58 27 88 49

Version	Modification	Fait par	Date
1.0	Initialisation	Pier-Jean MALANDRINO	10/02/2025
2.0	Finalisation	Pier-Jean MALANDRINO	28/02/2025

1. Introduction and Context

1.1. Objectives

The objective of the software is to provide an AI-driven assistant tailored for pool technicians in France, helping them efficiently resolve technical issues during onsite maintenance. By leveraging a Retrieval-Augmented Generation (RAG) system, the application integrates technical documentation from various manufacturers into a centralized repository. Pool technicians can submit specific technical questions through the user interface, and the system retrieves relevant documentation and generates precise, context-aware answers using a Large Language Model (LLM). Additionally, administrators manage the document repository by uploading and maintaining manufacturer-provided materials, ensuring that technicians always have access to accurate and up-to-date information. This solution aims to streamline maintenance workflows, reduce time spent searching for manuals, and improve overall service quality.

1.2. Requirements

1.2.1. Functional

1.2.1.1. User Roles

- **Application Users:** Pool technicians accessing the AI assistant for maintenance queries.
- **Manufacturers:** They have access to the document importation interface
- **Application Admins:** Administrators system configurations.

1.2.1.2. Core Features:

Document Import

- Admins upload documents (e.g., PDFs, technical guides) provided by manufacturers.
- Support for future extensions to other formats (e.g., API ingestion).

AI Query Resolution

- UserAs submit technical questions through the app interface.
- The system retrieves relevant documentation and generates responses using a Retrieval-Augmented Generation (RAG) system.

Search and Visualization

- Provide technicians with a user-friendly interface to search for and visualize documents.
- Include advanced filtering options to refine search results (e.g., by manufacturer, model, or keyword).

1.2.2. Technical

Language Management

R00 : Ensure that users can obtain answers in French (Impact on LLM choice)

Error Handling

R01 : Ensure robust error messages for failed queries or document ingestion issues.

Role-Based Access Control (RBAC)
R02 : Implement RBAC to manage access permissions for users and admins.

Mobile Platform Support
R03 : Deploy the application on mobile platforms (iOS and Android)

System Monitoring
- **R04 :** Provide admins with tools to monitor system performance, document ingestion status, and query accuracy metrics.

Validation and Testing
- **R04 :** Ability to validate the quality of results with a large battery of tests (e.g., query accuracy, recall/precision metrics).
- **R05 :** Support for A/B testing of different models to evaluate performance improvements (not in a production mode only in a test area).

Model Flexibility
- **R06 :** Rapidly switch between models to test performance, cost-effectiveness, or compatibility.
- **R07 :** Ensure modular architecture to support easy integration of new models.

Scalability
- **R08 :** Design the architecture and infrastructure to scale with increasing document volumes and user queries. Boundaries will be provided in the Infrastructure detail chapter.

Deployment
- **R09 :** Containerize all components using Docker for portability across environments.
- **R10 :** Ensure compatibility with cloud providers for deployment flexibility. We are not targeting a full cloud agnostic approach, but have a strategy to avoid a full vendor locking.

Data Consistency
- **R11 :** Guarantee transactional integrity during document ingestion (e.g., rollback on failure).
- **R12 :** Ensure vectorized data aligns with parsed chunks for accurate similarity search.

1.2.3. Regulatory

GDPR Compliance

- **R13 :** Ensure all personal data processing complies with GDPR principles, including transparency, accountability, data minimization, and purpose limitation.
- **R14 :** Provide users with clear information about how their data is processed within the application. Implement mechanisms to allow users to exercise their rights under GDPR (e.g., access, rectification, deletion).

Data Security

- **R15 :** Data must be stored in a French cloud provider and in france
- **R16 :** Maintain audit logs for document uploads and user interactions to ensure traceability.

2. Target architecture

2.0.1. Functional view

The functional view of the application is organized into two main roles:

2.0.1.1. Application Users (Pool Technicians)

Role:
Pool technicians interact with the system to resolve technical issues during onsite maintenance.

Workflow:

- They submit a technical question through the application interface.
- The system processes the query using its AI-powered Retrieval-Augmented Generation (RAG) system.
- The user receives an answer along with relevant documentation to guide their intervention.
- Users can have threads of questions that would be historised. In a first implementation we will not include thread questions in the new context, which could be done later.

2.0.1.2. Application Admins

Role:
Admins manage the backend by uploading and maintaining documentation provided by manufacturers.

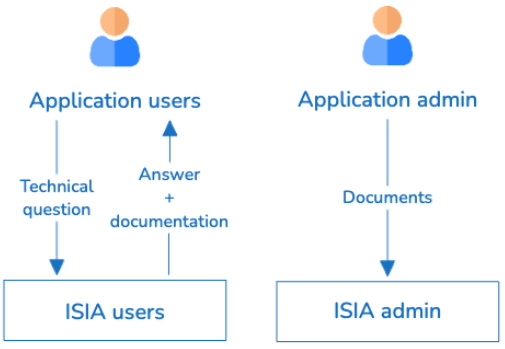
Workflow:

- Admins upload documents (e.g., manuals, technical guides) into the system.
- These documents are processed and stored in a structured format for efficient retrieval.

2.0.1.3. System Components

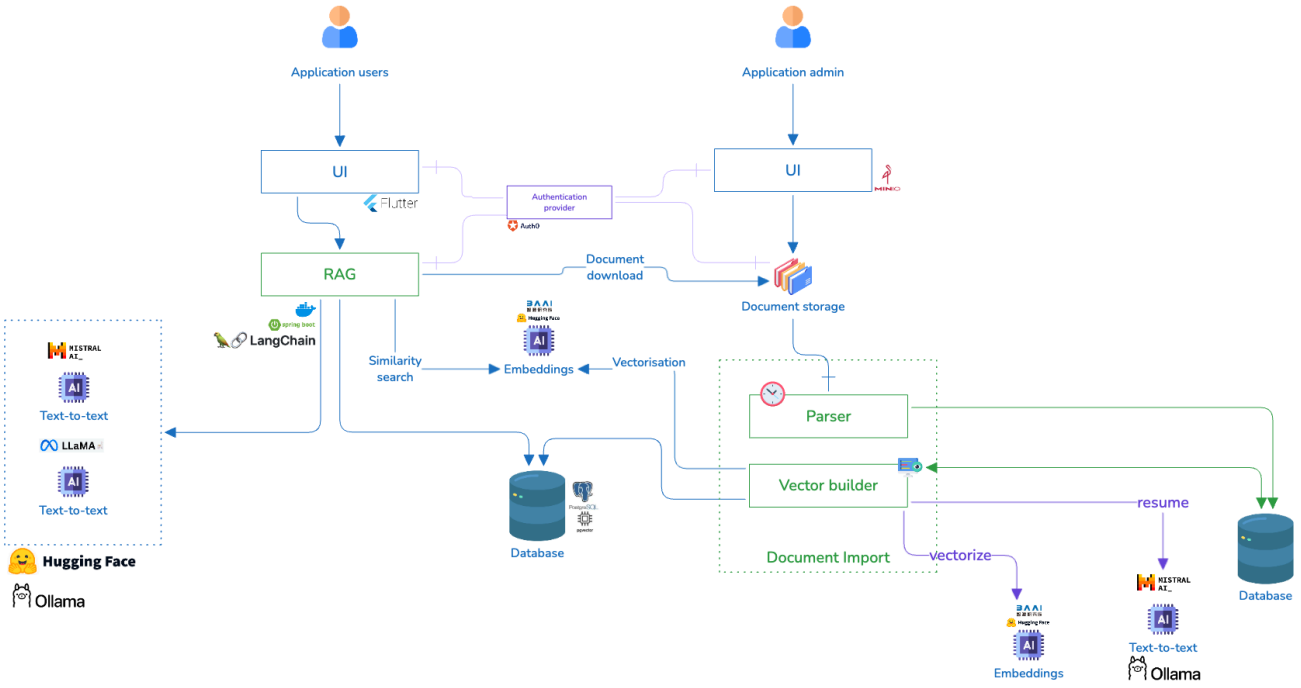
ISIA Users Module: Handles user queries, processes them via AI, and delivers relevant answers and documentation.

ISIA Admin Module: Manages document ingestion and ensures data integrity for accurate query resolution.



2.0.2. Application Layer

2.0.2.1. General application

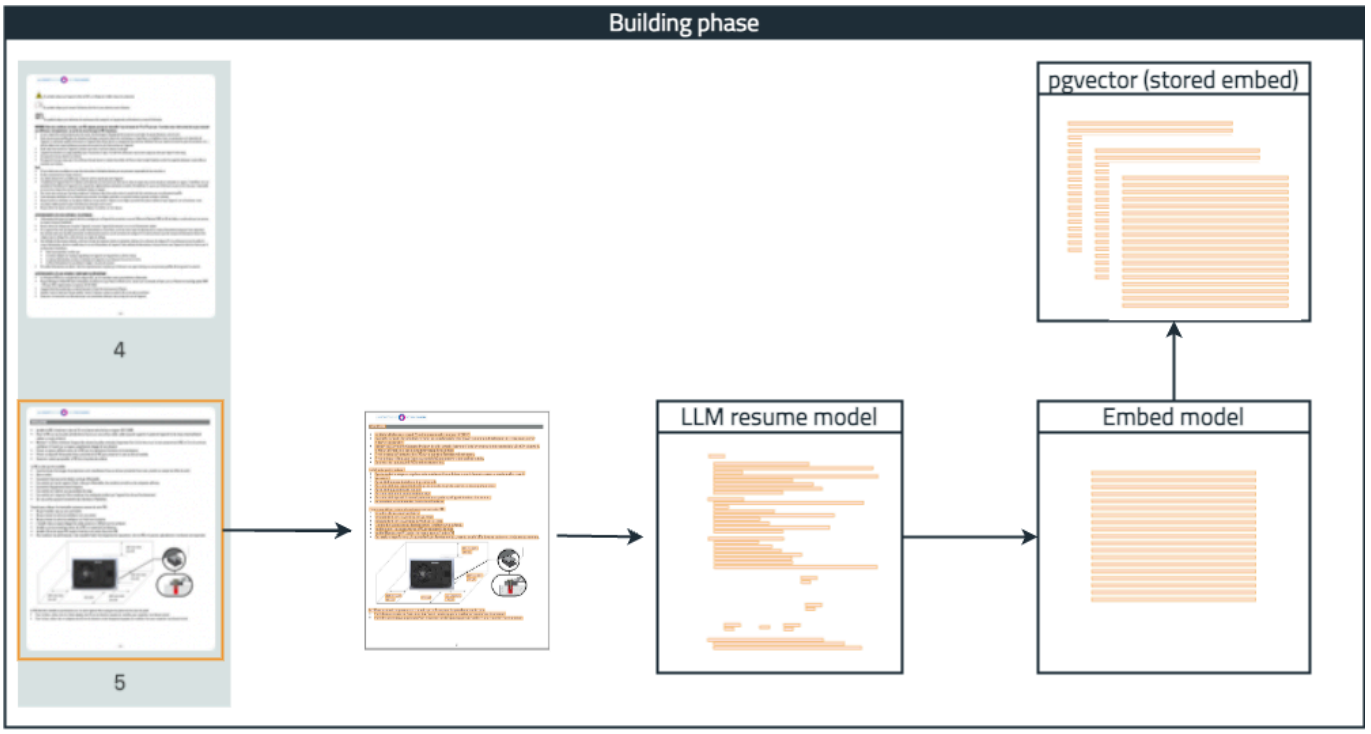


The application is divided into the following key parts:

B2B Administration (Admin Interface) R04

This component focuses on document integration and management, primarily targeting administrators or manufacturers. Initially, it will **handle PDF documents**, leveraging the **Apache PDFBox** library for processing. PDFBox is an open-source Java tool that enables operations such as content extraction, form data handling, and validation of PDF files [1]. This robust library ensures efficient and reliable parsing of PDF documents during the ingestion process.

Future iterations plan to extend support to other formats (e.g., API-based ingestion) to accommodate a wider range of document sources. Detailed functionality for document handling and management is described in the next chapter.

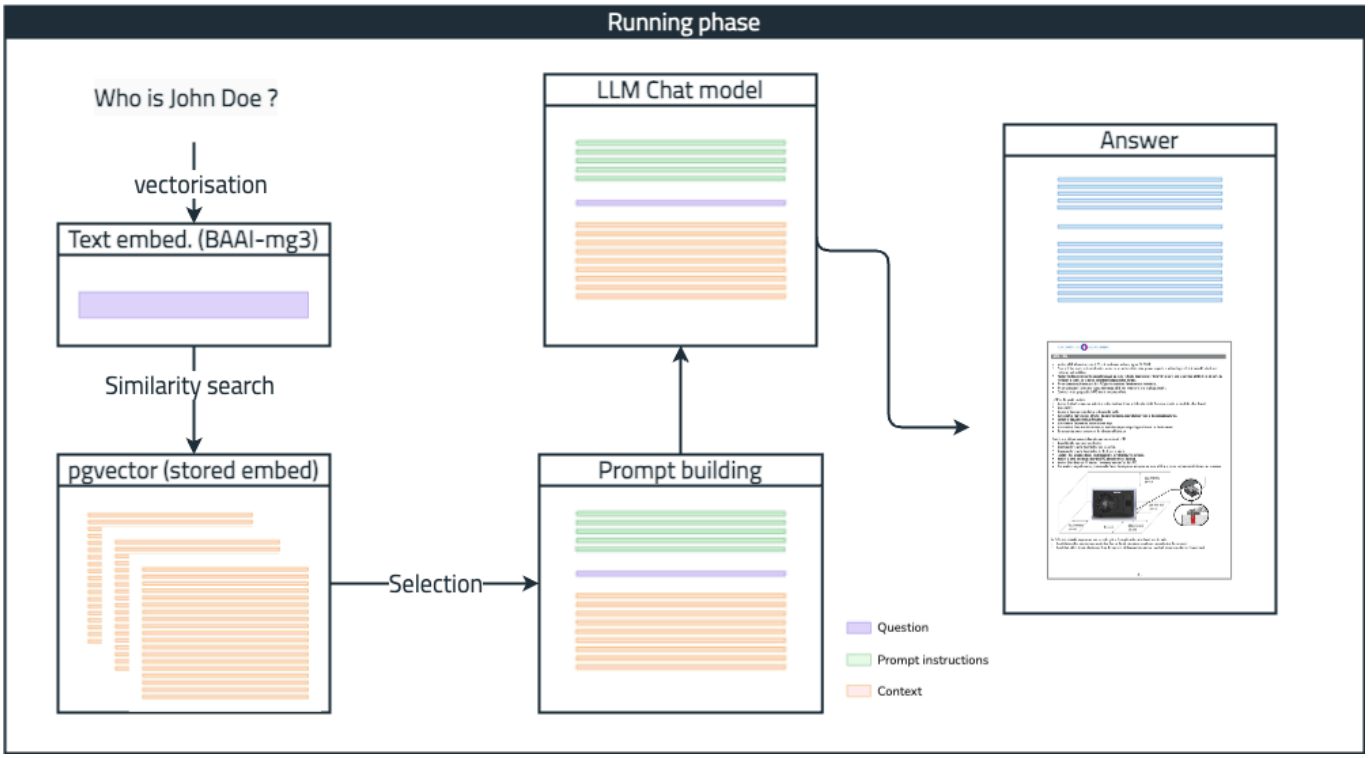


B2C Visualization (End-User Interface)

UI component is a **Flutter-based mobile application** designed for pool technicians (end-users).

It integrates a **Retrieval-Augmented Generation (RAG)** system to allow technicians to search for and visualize technical documents efficiently [11].

The RAG system leverages embeddings and LLMs to provide context-aware responses, ensuring quick access to relevant information for onsite interventions.



LLM Interactions

To support the application's AI-driven features, multiple Large Language Models (LLMs) are integrated into the architecture. LLMs are used for various tasks, including text-to-text generation (chat interactions) and embedding generation (similarity search).

The architecture is designed to allow easy replacement or upgrading of models as needed. All models are sourced from *Hugging Face*, leveraging its repository of open-source AI models for flexibility and scalability [4].

2.0.2.2. Batch focus

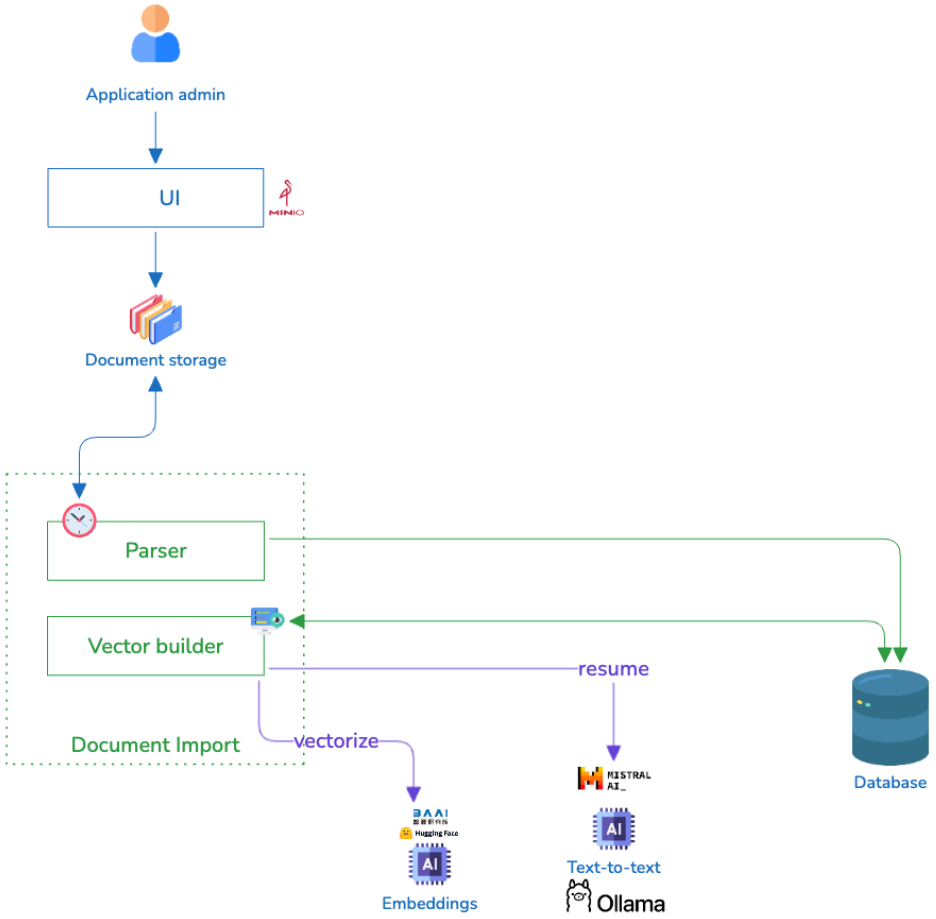
We have deliberately chosen to **strongly decouple the parsing and vectorization phases** in our architecture [R08]. This decision ensures flexibility and adaptability for future developments:

Parsing Phase: Currently, the parser is designed to handle **PDF documents**, but this modular approach allows us to easily implement new parsing methods (e.g., API consumption, XML parsing) without impacting the rest of the pipeline [R11].

This ensures that as new data sources or formats emerge, they can be integrated seamlessly.

Vectorization Phase: The vectorization process is also decoupled to allow for future improvements. For instance, ongoing R&D (e.g., Colpali, as discussed later in this document) suggests that vectorization strategies may evolve significantly in the near term [R12]

By isolating this phase, we can adopt new vectorization techniques or models without requiring changes to the parsing logic or storage mechanisms.



Component Overview

UI (Document Storage Interface)
For now, we are using MinIO's UI for document management. This interface allows administrators to upload and manage documents efficiently [7].

Document Storage
The final solution for document storage will depend on the cloud provider we select (discussion pending with Laurent Delage). MinIO is currently used as a placeholder for local object storage [7].

Parser
Implemented using *Spring Batch*, the parser prepares documents for vectorization by splitting them into manageable chunks.

Vector Builder
Also implemented with *Spring Batch*, the vector builder processes parsed chunks into embeddings.

It checks the Spring Batch log tables to ensure it only triggers after a full document has been successfully parsed. This transactional granularity guarantees data consistency—if any page fails during processing, the entire document is rolled back, meeting our quality requirements.

Future Scalability: While we currently follow a KISS (Keep It Simple, Stupid) principle, we may transition this component to a messaging system (e.g., Kafka or RabbitMQ) if higher throughput is required.

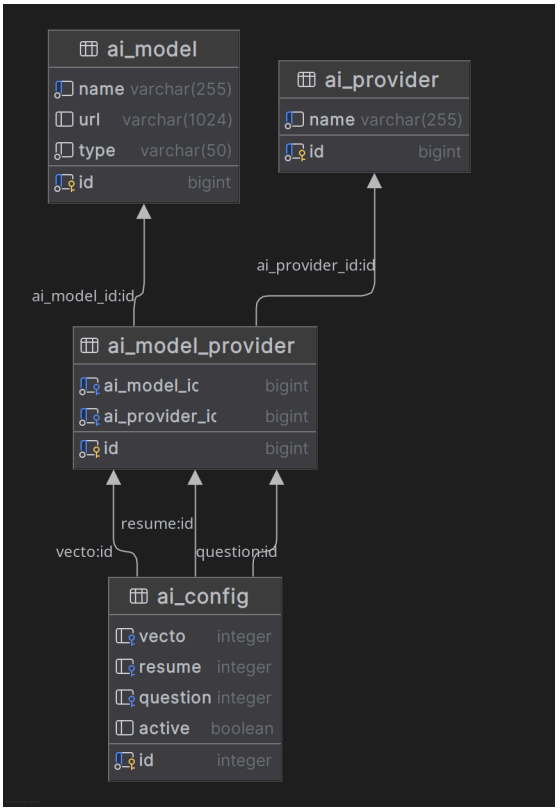
Database
We use `pgvector` within PostgreSQL to store vectorized documents. This choice aligns with our goal of combining relational and vector-based queries in a single database system while maintaining cost efficiency [6].

2.0.2.3. Architecture applicative focus

To ensure a clear separation between business logic and technical implementation, we have chosen a Hexagonal Architecture (Ports and Adapters) [3]. This approach allows us to isolate the core functional behavior from external dependencies, making the system more modular, testable, and adaptable. By abstracting infrastructure concerns, we gain flexibility in evolving our technology stack while maintaining a strong, domain-driven foundation.

This architecture enables us to meet R06 and E07 by ensuring a modular design that allows for the rapid switching of models to test performance, cost-effectiveness, and compatibility while facilitating the seamless integration of new models.

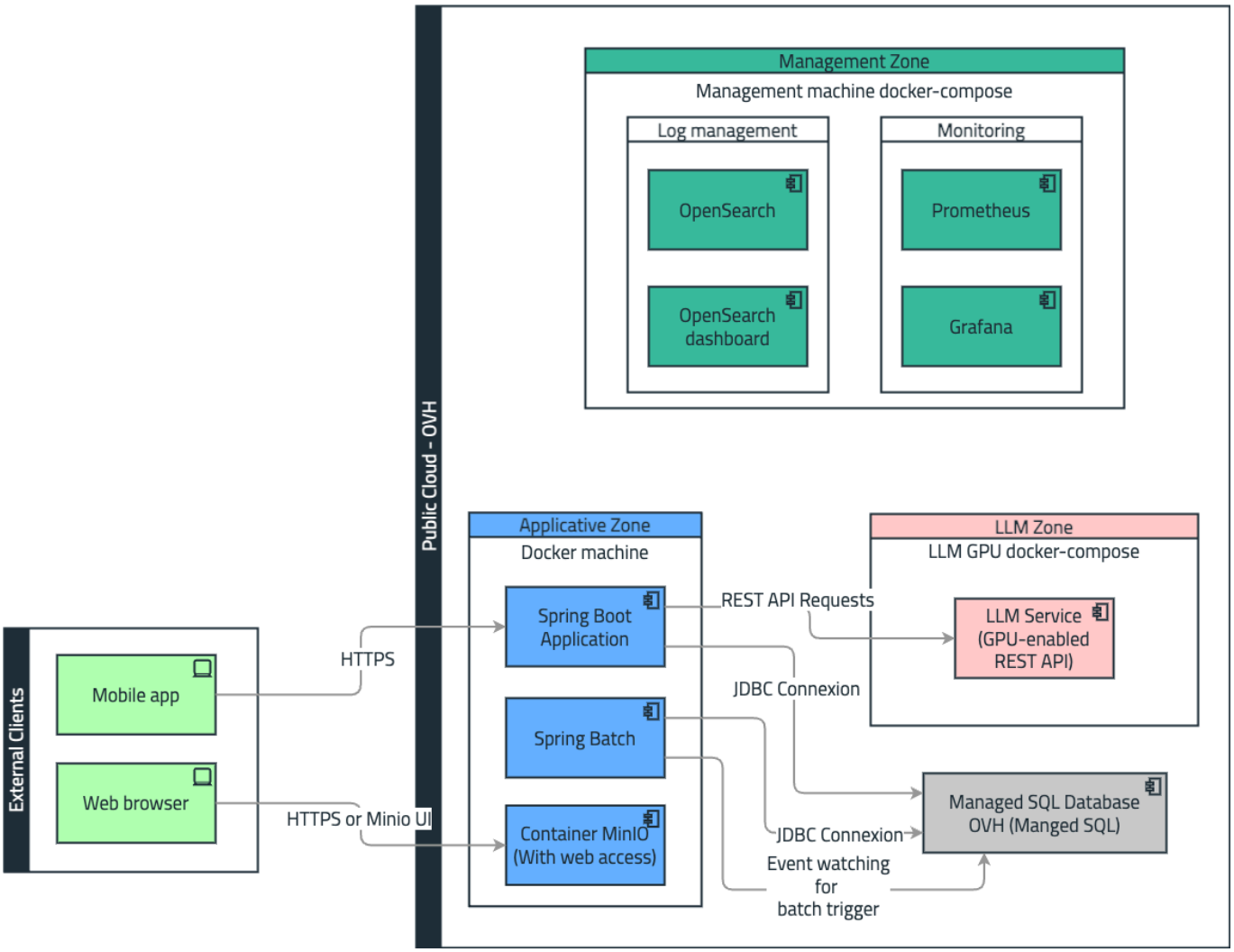
To better illustrate the concept of **model switching**, the diagram below shows how Hexagonal Architecture supports this process [R06][R07]. All models will be stored in the database, and each request can select a specific configuration. Multiple configurations can be active simultaneously.



2.0.3. Infrastructure Layer

The infrastructure is designed to balance scalability, security, and maintainability while leveraging cloud capabilities [R06][R07].

Here is an overview of all components :



2.0.3.1. Cloud Provider Strategy

We have selected OVH as the cloud provider for this project [R15]. This choice ensures:

Scalability: Public cloud infrastructure allows dynamic scaling of resources (compute, storage, and network) to adapt to project demands during both initial setup and ongoing operations [R08].

Cost Control: OVH's pricing model avoids unexpected expenses for storage and compute scaling.

2.0.3.2. Access Management

Infrastructure Team: Direct access to machines (SSH, admin consoles) is restricted to this team to enforce security.
Development Team: Access is limited to:

- Log Aggregation: Monitoring via OpenSearch Dashboards (https://logs.isia.tech).
- Deployment Workflows: Interaction through CI/CD pipelines (GitHub Actions) and container registries.

2.0.3.3. DevOps and Deployment Strategy

Container Management:

All application components are containerized and stored in OVH's Container Registry, enabling seamless scaling without cost surprises.

Production deployments require:

- Merge/pull requests with detailed change descriptions.
- Manual review and approval by the Infrastructure Team before deployment.

Infrastructure as Code (IaC) [R10]:

Terraform provisions all infrastructure components, ensuring:

- Disaster recovery capabilities.
- Version-controlled infrastructure changes.
- Rapid environment replication (e.g., staging, production).

Low-level stacks (e.g., monitoring) are automatically updated to the latest stable versions.

CI/CD Pipeline:

GitHub Actions automates testing, image builds, and deployment triggers.

2.0.3.4. Development Stacks

ISIA v2 continues development under the existing GitHub organization isia-tech (https://github.com/isia-tech), with two new subprojects:

- isia-platform-v2 (https://github.com/isia-tech/isia-platform-v2) : It contains all batches and backend RAG APIs code and image structure with associated Github actions for CI/CD.
- isia-platform-v2-front (https://github.com/isia-tech/isia-platform-v2-front) : all mobile app code with associated Github actions for CI/CD.
- isia-infra-compose (https://github.com/isia-tech/infra-infra-compose) : This repository will host every docker-compose needed by application. One branch per environment.

As we adopt a DevOps approach, it is imperative that we streamline the responsibility of the Dev and Infra teams. For all changes to containers contexts for all stacks, and for version changes on production stacks, the DevTeam will create merge/pull requests in the respective repositories, along with a detailed description of the changes. Each change will be thoroughly reviewed and will be deployed by an InfraTeam member.

2.0.3.5. Frontend Delivery

The Flutter-based frontend is compiled into native binaries for Android and iOS using CodeMagic CI/CD. Deployment artifacts are published directly to app stores, ensuring compliance with platform-specific guidelines.

2.0.4. Operational Layer

2.0.4.1. Monitoring

The monitoring system is designed to provide comprehensive visibility into infrastructure health, application performance, and log management. It combines metrics collection, log aggregation, and alerting in a scalable and centralized manner [R04].

Metrics Monitoring

Components:

- ***Prometheus:*** A time-series database that collects and stores metrics from all infrastructure components [8].
- ***Grafana:*** A visualization and alerting platform that displays dashboards and triggers alerts based on predefined thresholds [9].

- ***Node-exporter***: Exposes low-level host metrics (CPU, memory, disk/GPU usage, network I/O).
- ***cAdvisor:*** Monitors container-level metrics (e.g., resource usage per Docker container).

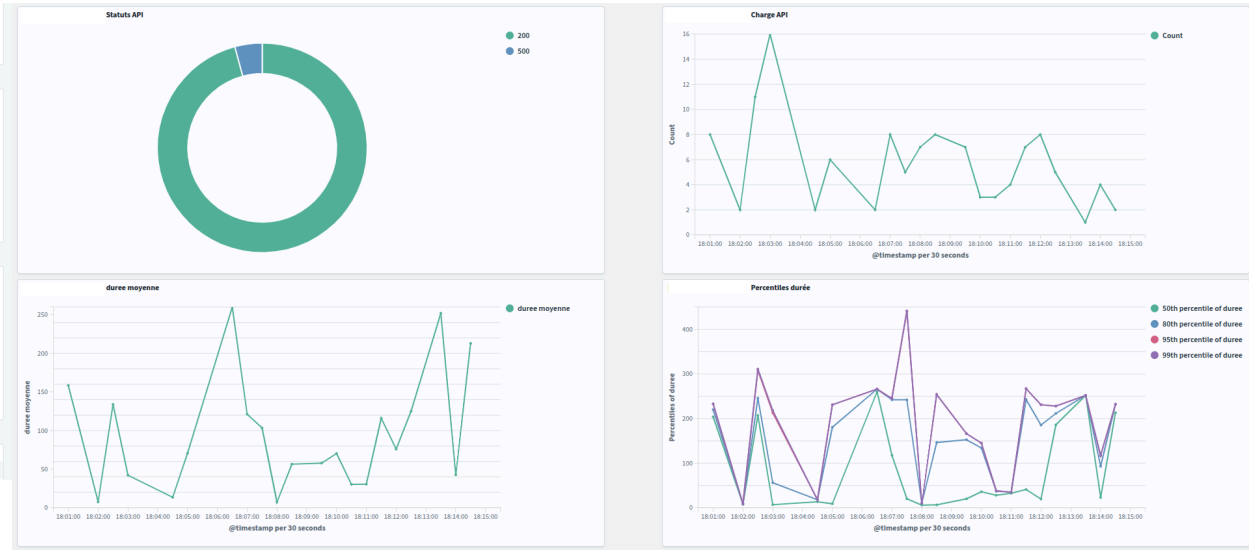
Implementation:

All machines in the management zone run the node-exporter and cAdvisor to expose metrics. Prometheus then scrapes these metrics at regular intervals by default (15 seconds). Grafana connects to Prometheus to visualise data via dashboards (e.g., https://sup.isia.tech) and configure alerts (e.g., disk saturation, GPU overload).

Log Management

Components:

- ***OpenSearch:*** A distributed search and analytics engine for storing and querying logs [10].
- ***Filebeat:*** A lightweight log shipper that forwards logs to OpenSearch.
- ***OpenSearch Dashboards:*** A visualization tool for log analysis (e.g., https://logs.isia.tech).



Implementation:

Application and infrastructure components are responsible for writing logs to local JSON files. Each Docker host runs a Filebeat instance, the purpose of which is to collect logs and forward them to OpenSearch. OpenSearch then indexes logs for fast search and analysis. OpenSearch Dashboards provides prebuilt dashboards for log trends, error analysis, and audit trails.

Log Types Collected:

- ***Application*** logs (e.g., Spring Batch errors, RAG query failures).
- ***Infrastructure*** logs (e.g., Docker daemon events, cron job outputs).
- ***Security*** logs (e.g., Auth0 authentication attempts).

Alerting and Incident Management

To ensure proactive system monitoring and quick incident response, the following alerting rules have been defined. These rules cover both system-level and container-level metrics to maintain the stability and performance of the infrastructure.

System-Level Alerts

High CPU Usage:

- Triggered when CPU usage exceeds 90% for more than 20 minutes.

Disk Usage Threshold:

- Triggered when disk usage exceeds 90% or is projected to reach 90% within 24 hours, based on current fill rates.

Clock Desynchronization:

- Triggered when the system clock becomes desynchronized, which could affect logs and metrics accuracy.

Out-of-Memory (OOM) Kill Events:

- Triggered when an OOM kill event is detected, indicating memory allocation issues.

Blackbox Probes:

- Triggered when a blackbox probe (e.g., HTTP endpoint check) responds in more than 10 seconds or fails entirely.

Container-Level Alerts

High I/O Utilization:

- Triggered when I/O utilization exceeds 80% for a container over a period of 20 minutes.

High CPU Usage in Containers:

- Triggered when CPU usage exceeds 80% for a container over a period of 2 minutes.

Container Termination:

- Triggered when a container unexpectedly stops or is killed, ensuring immediate attention to potential service disruptions.

3. Key technical decisions

3.1. Technology choices

3.1.1. Flutter

Flutter is a cross-platform mobile development framework that also supports web applications. Given the simplicity of the mobile features we are implementing, there is no need for platform-specific code or highly specialized implementations. Flutter is perfectly suited for this use case.

Additionally, the team already has solid experience with Flutter, not only in development but also in the entire app publication process, which will streamline delivery and maintenance.

3.1.2. Spring

Spring boot
We have chosen Spring Boot as the backend framework to package and deploy the project efficiently. Spring Boot provides a robust foundation for building scalable and maintainable microservices.

Spring Batch
For document ingestion, Spring Batch has been selected as it offers fine-grained control over batch processing workflows, enabling us to design and implement an optimized document insertion strategy (detailed separately).

Additionally, the team already has solid experience with Spring (Boot, Core, batch Security etc..) and we can make our client benefit from it.

3.1.3. Postgres

PostgreSQL has been selected as the primary database due to its reliability and ability to handle both structured data and vector data through the pgvector extension. This choice aligns with our current needs:

- The application **will initially have moderate usage**, so PostgreSQL offers a cost-effective solution without requiring immediate investment in advanced ANN (Approximate Nearest Neighbor) methods or specialized vector databases like Milvus or Pinecone.
- Following the KISS principle (Keep It Simple, Stupid), we aim to deploy a fully functional version first and **adapt based on user feedback** to optimize costs and performance incrementally.

The combination of relational structures and vector search capabilities makes PostgreSQL + pgvector a pragmatic choice for our initial deployment while allowing scalability in the future.

3.1.4. Docker

To ensure scalability and deployment flexibility, we are containerizing the application using Docker. Docker is a widely adopted technology that simplifies application deployment by creating portable containers that are agnostic to specific environments or infrastructure constraints.

This approach prepares us for future scaling needs while maintaining compatibility with various deployment strategies (e.g., cloud, on-premise). We also have a solid experience with that technology.

3.1.5. Hugging Face

Hugging Face serves as the central hub for model discovery, evaluation, and integration in our application. We will actively analyze and benchmark models based on:

Selection Criteria:

- ➢ Performance: Accuracy in technical Q&A tasks (e.g., Mistral-7B vs. Llama-3).
- ➢ Cost Efficiency: VRAM requirements and inference speed (quantization-ready models preferred).

GPU Pre-Dimensioning:

- ➢ Identify requirements for target models based on SageMaker dimensions or Inference endpoints suggestion.
- ➢ Provision GPU instances on OVH based that fulfill requirements

3.1.6. Ollama

Ollama acts as the unified layer for GPU resource management and model orchestration, enabling:

Cloud Deployment:

The process of wrapping Hugging Face models in Ollama's standardised runtime will eliminate the need to rework on containerisation every time a model is changed. This will also eliminate the need to manage high-volume Docker images on our registry.

Note: While we rely on Ollama initially, critical models will be replicated locally to ensure resilience and reduce dependency on external platforms during automated deployments.

3.1.7. Minio

MinIO has been selected as the object storage solution for managing documents uploaded by administrators or manufacturers. It provides S3-compatible storage capabilities, making it easy to integrate with existing tools while ensuring scalability for large volumes of technical documentation.

3.1.8. Auth0

Auth0 has been chosen as the authentication provider based on client requirements. While we typically recommend Keycloak due to its open-source nature and cost-efficiency (no licensing fees), the client explicitly requested Auth0 for this project.

Note: As part of our "duty of advice", we informed the client about Keycloak's advantages, including its free open-source model, but ultimately respected their decision.

3.1.9. LangChain4j

LangChain4j is a Java framework designed to simplify the integration of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) workflows into Spring Boot applications.

Key Advantages :

- ➢ Seamless Spring Boot Integration: Provides pre-built components for LLMs, embedding models, and vector stores, reducing boilerplate code.
- ➢ RAG Workflow Support: Includes tools for document loaders, text splitters, and vector search, enabling efficient document retrieval.

- Flexibility: Works with multiple AI providers (e.g., Hugging Face, Mistral AI), allowing us to adapt to evolving needs.
- Performance: Optimized for enterprise applications with faster response times and scalable architecture.
- Its compatibility with pgvector aligns perfectly with our database choice.

3.2. Models choice and validation strategy

3.2.1. Validation Strategy

The validation strategy for LLM models and infrastructure follows a two-phase approach:

3.2.2. Quality testing

Objective: Evaluate the accuracy and relevance of model responses to technical queries.

Methodology:

Dataset: 42 questions derived from 5 technical documents (heat pump manuals, maintenance guides).

Tested Models:

- meta-llama/Llama-3.1-8B
- mistralai/Mistral-Small-24B-Instruct-2501

Both models will be tested for chunking and chatting.

Results:

	Chunk resume model
	Chat model

meta-llama/Llama-3.1-8B			
mistralai/Mistral-Small-24B-Instruct-2501		**meta-llama/Llama-3.1-8B**	
OK	37	**OK**	35
Partial OK	33	**Partial OK**	35
KO	28	**KO**	28
Vecto OK	60	**Vecto OK**	60%
% OK	37,76%	% OK	35,71%
% OK + POK	71,43%	% OK + POK	71,43%
Temps Chat	2m23	*Temps Chat*	2m07

mistralai/Mistral-Small-24B-Instruct-2501			
mistralai/Mistral-Small-24B-Instruct-2501		**meta-llama/Llama-3.1-8B**	
OK	26	**OK**	22

Partial OK	31	**Partial OK**	40
KO	42	**KO**	37
Vecto OK	57	**Vecto OK**	57
% OK	26,26%	% OK	22,22%
% OK + POK	57,58%	% OK + POK	62,63%
Temps Chat	2m19	*Temps Chat*	1m34

Key Findings:

- ➢ Mistral-Small-24B underperforms for chunk summarization, reducing overall answer quality.
- ➢ Llama-3.1-8B balances accuracy (71.43% OK + Partial OK) and speed, making it preferable for chat interactions.

Performance testing :

Objective: Validate cost-efficiency and throughput of selected models on target infrastructure.

Tested configuration :

Name	Memory	vCore	GPU	Storage	Public Network	Private Network	Prive hourly
a10-45	45 Go	30	A10 24 Go	400 Go SSD	8 Gbit/s	8 Gbit/s max.	0,76 €
t1-45	45 Go	8	Tesla V100 16 Go	400 Go NVMe	2 Gbit/s garantis	4 Gbit/s max.	1,65 €

Cost projections:

A10-45		
Price hourly	0,76 €	0,76 €
H per day	10	24
Total monthly	**228,00 €**	**547,20 €**

T1-45		
Price hourly	1,65 €	1,65 €
H per day	10	24
Total monthly	**495,00 €**	**6379**

3.2.3. Performance results

All performance results were obtained through monitoring execution processes that generated log files and Gatling reports. These log files were formatted and compiled using Python scripts available in the repository: https://github.com/isia-tech/isia-performance-results.

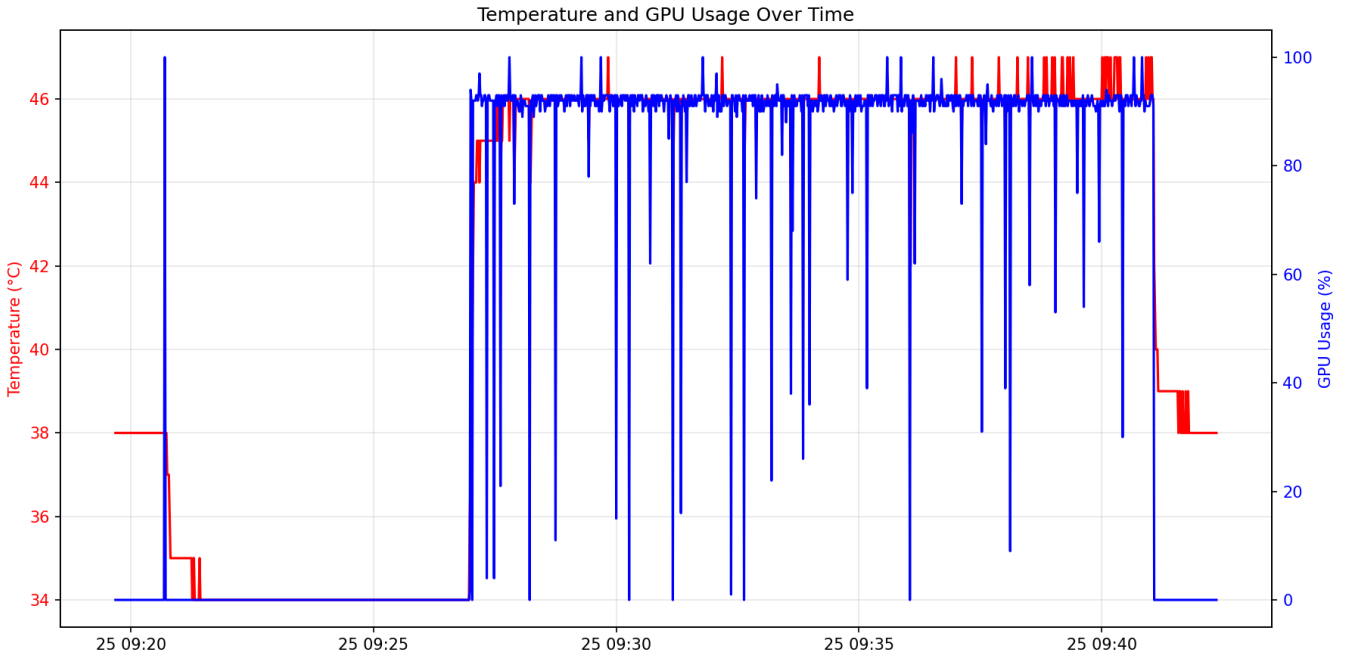
All processed results have been archived in the same repository for future reference and analysis.

3.2.3.1. Configuration 1 : A10-45

GPU Usage and Temperature

The GPU usage graph shows consistent 100% utilization during the majority of the test, indicating that the A10-45's NVIDIA A10 GPU is operating at maximum capacity throughout the workload. This suggests that the GPU is fully leveraged for inference tasks, which is expected for a model like Llama-3.1-8B. However, there are brief dips in GPU usage, which could indicate minor pauses in workload processing or potential bottlenecks elsewhere in the pipeline (e.g., data input/output).

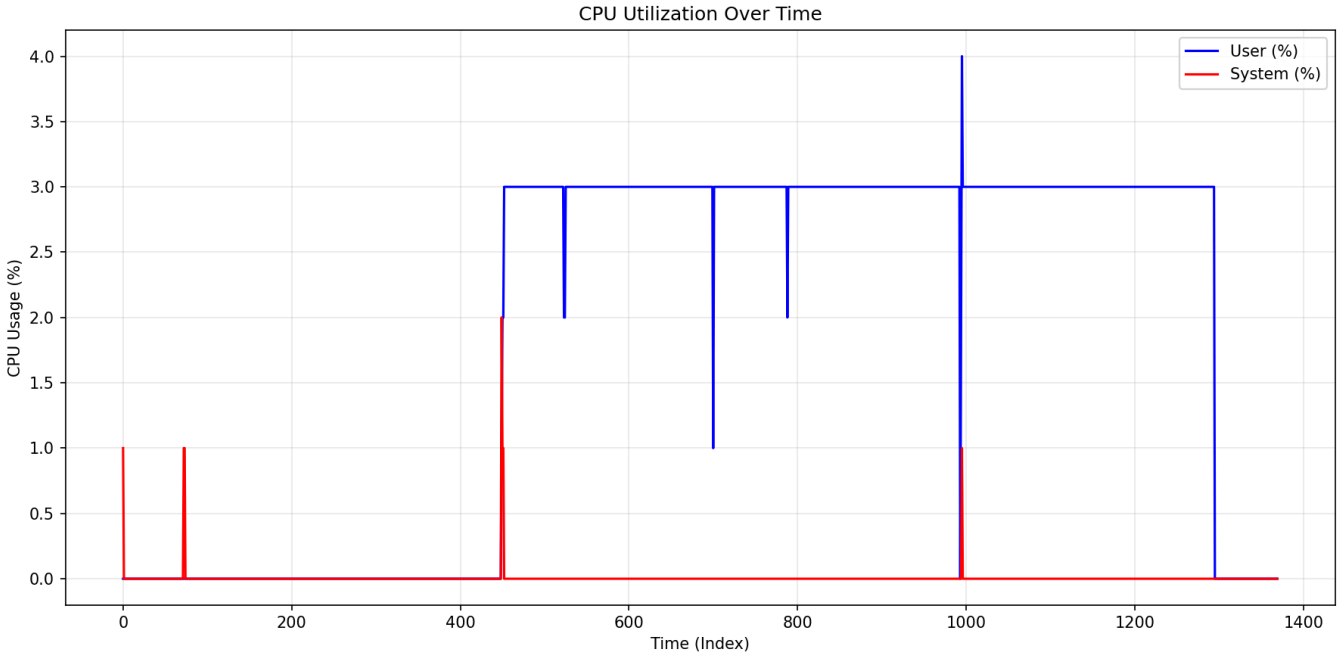
The temperature remains stable around 46°C, which is within acceptable operational limits for a GPU under load. The stability suggests that the cooling system is effective and that thermal throttling is not a concern.



CPU Utilization

The CPU utilization graph indicates very low usage, with peaks at around 3–4% during the test. This aligns with expectations since the A10-45 configuration offloads most of the workload to the GPU. The low CPU usage leaves significant headroom for other tasks, such as embedding generation or preprocessing, which are typically CPU-bound.

The separation of responsibilities between GPU and CPU appears well-balanced, with no signs of contention or overload on the CPU side.



Response Time

From the Gatling report:

- The mean response time is approximately 10.9 seconds, with a standard deviation of 7.2 seconds.
- The 95th percentile response time reaches 30.2 seconds, while the 99th percentile hits 35 seconds. These high percentiles suggest occasional outliers in processing time, possibly due to larger or more complex prompts requiring additional computation.

Overall, response times are consistent with expectations for Llama-3.1-8B running on this hardware configuration.

Observations on Resource Utilization

- The A10-45 configuration demonstrates efficient GPU utilization but leaves significant CPU capacity unused. This opens opportunities for optimizing resource allocation by running additional CPU-bound tasks (e.g., embedding generation) concurrently on the same machine.
- The consistent GPU usage and stable temperature suggest that this configuration can handle sustained workloads without degradation in performance.
- Response time variability at higher percentiles may warrant further investigation into specific query patterns or input complexities causing delays.

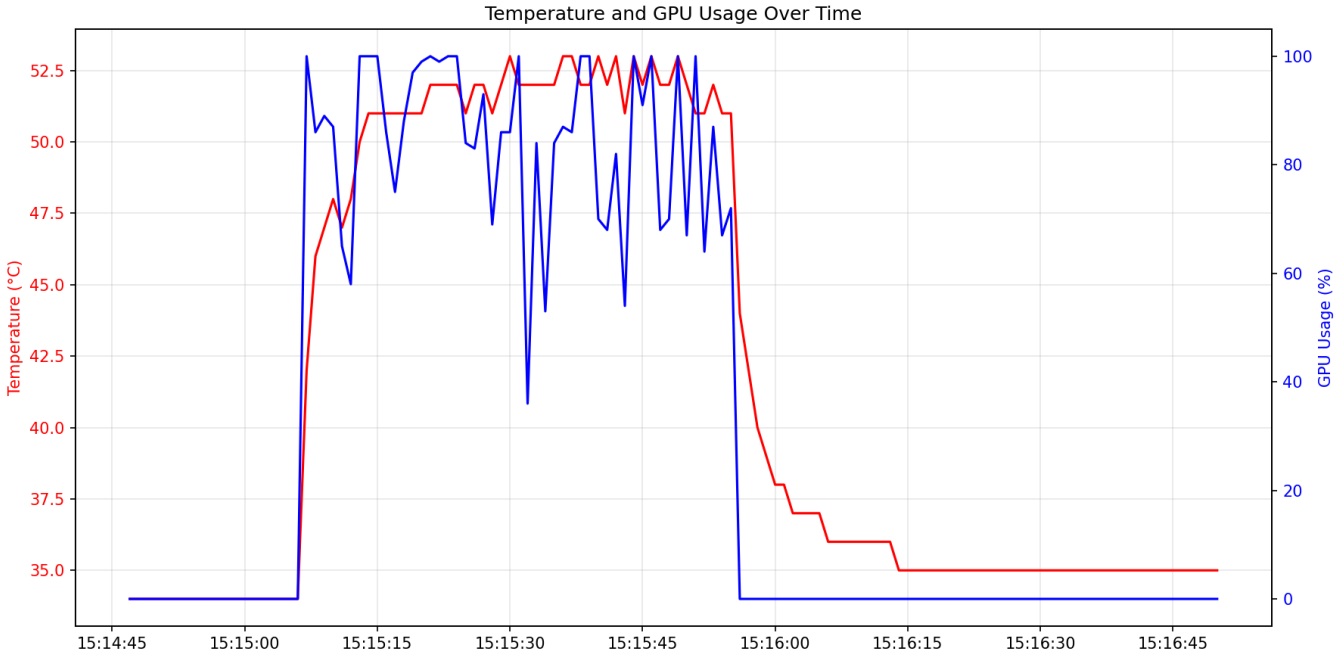
This analysis highlights that the A10-45 configuration is well-suited for inference workloads but may benefit from further resource optimization to fully utilize available CPU capacity.

3.2.3.2. Configuration 2 : T1-45

GPU Usage and Temperature

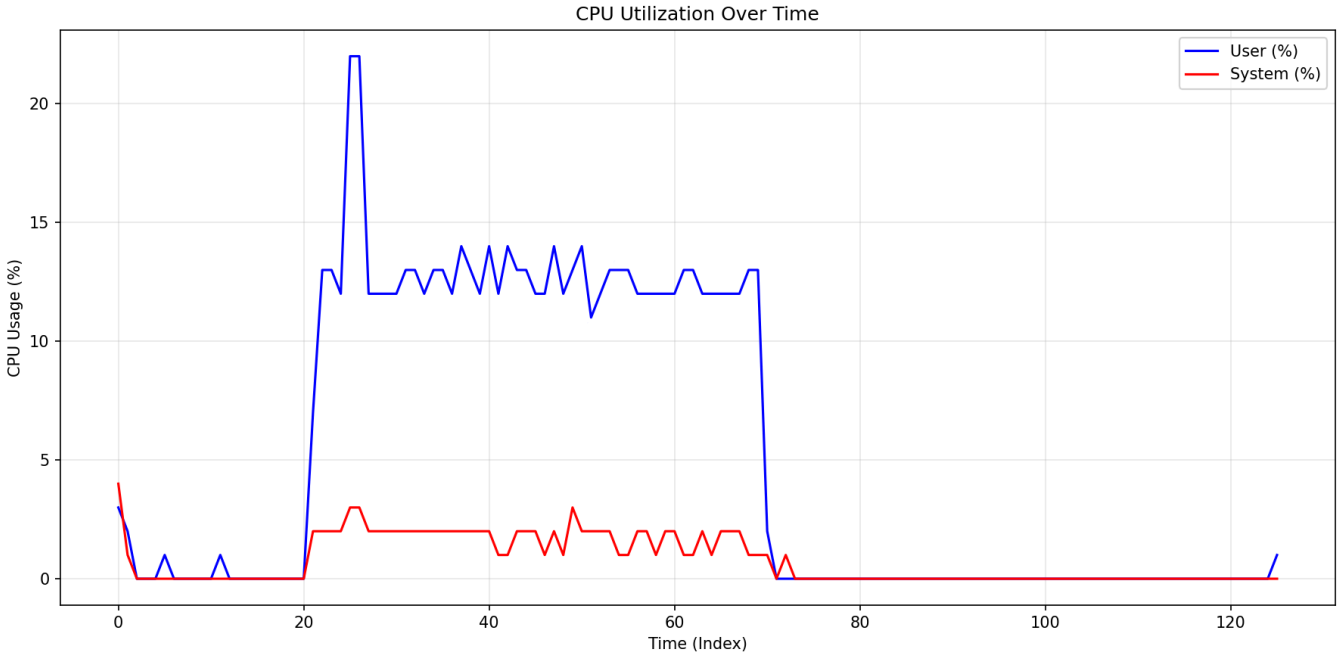
The GPU usage for the T1-45 configuration shows variability between 60% and 100% utilization during the test. This fluctuation indicates that the workload does not consistently saturate the Tesla V100 GPU, suggesting potential inefficiencies in workload distribution or resource allocation.

The GPU temperature stabilizes around 52°C, slightly higher than the A10-45 configuration but still within safe operational limits. The temperature increase could be attributed to higher power consumption spikes during periods of peak GPU usage.



CPU Utilization

The CPU usage graph reveals a significant difference compared to the A10-45 configuration. The T1-45's CPU peaks at approximately 20% utilization, with sustained levels around 15–18% during the test. This higher CPU load suggests that the Tesla V100 relies more on CPU resources for preprocessing or auxiliary tasks, potentially limiting multitasking efficiency for embedding generation or other CPU-bound operations.



Response Time

From the Gatling report:

- The mean response time is approximately 9.98 seconds, slightly better than the A10-45 configuration (~10.9 seconds).
- The 95th percentile response time is 26.2 seconds, showing less variability compared to A10-45 (30.2 seconds).
- The lower standard deviation in response times suggests more consistent performance, possibly due to better handling of input/output operations or smaller prompt sizes during testing.

Observations on Resource Utilization

- The fluctuating GPU utilization indicates that the workload is not fully optimized for the Tesla V100, leading to underutilization during certain periods.
- The higher CPU usage compared to A10-45 could become a bottleneck if additional CPU-bound tasks (e.g., embeddings) are introduced, as it reduces available headroom for multitasking.
- Response times are slightly faster and more consistent than A10-45, which may reflect differences in GPU architecture or driver optimizations.

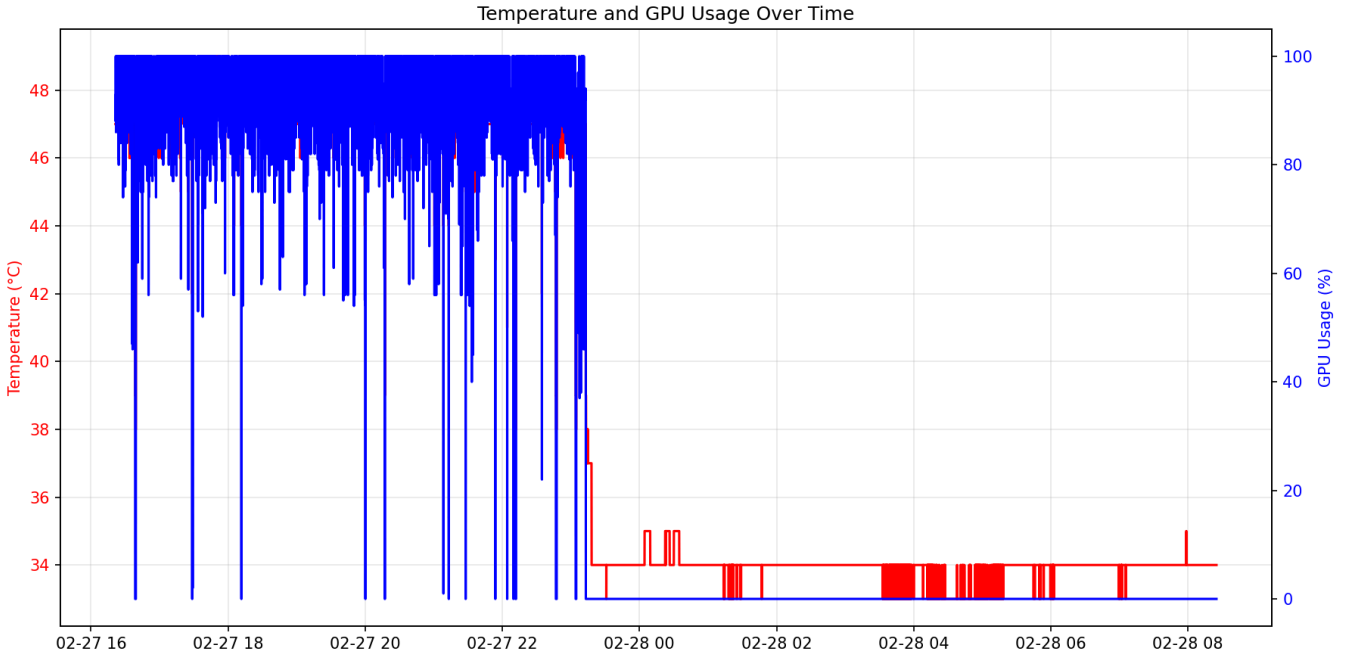
This analysis highlights that while T1-45 delivers comparable performance, its resource utilization patterns suggest room for optimization in workload distribution and CPU/GPU task balancing.

3.2.3.3. Massive doc import

The test involved importing 104 documents totaling 6,379 pages. The process was monitored for performance metrics, including GPU utilization, CPU utilization, and system stability. Below is the analysis based on the available data.

GPU Utilization and Temperature

- The GPU usage graph shows consistent high utilization during the import process, with usage peaking at 100% for extended periods. This indicates that the GPU was heavily leveraged for tasks such as document vectorization or preprocessing.
- Temperature remained stable around 46–48°C, suggesting that the system's cooling mechanisms were effective under sustained load. The absence of significant temperature fluctuations implies that there were no thermal throttling events during the test.

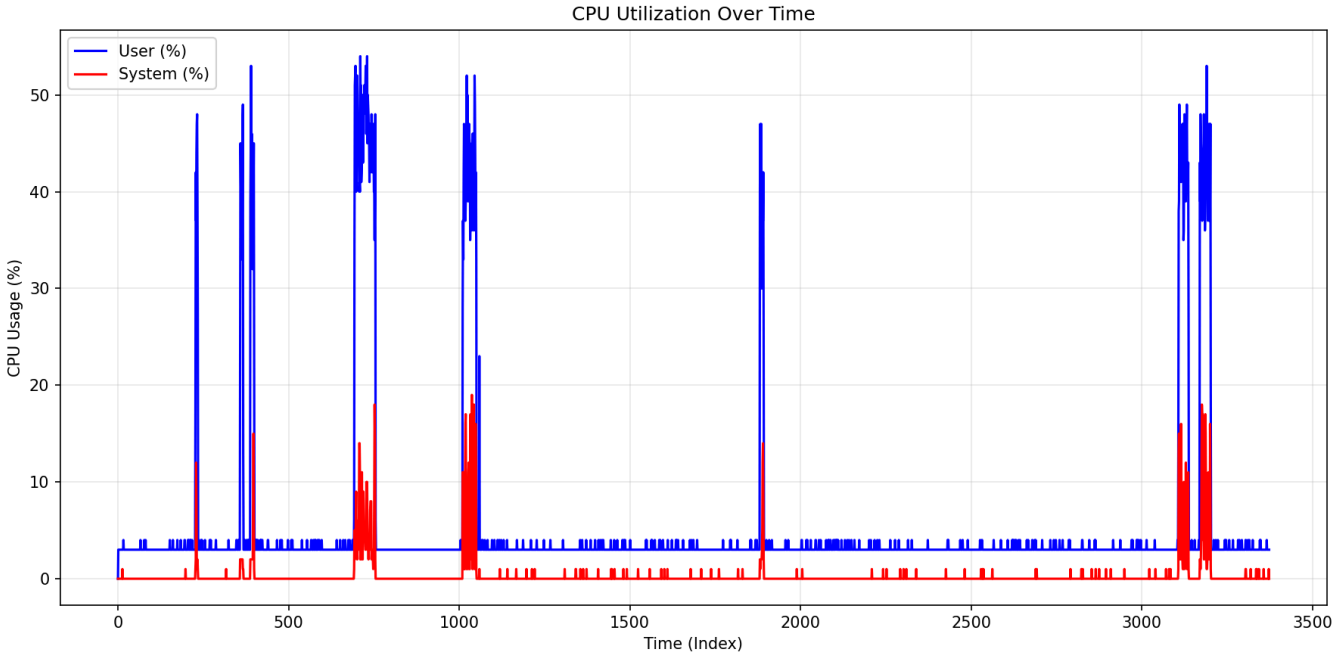


CPU Utilization

The CPU utilization graph reveals a distinct pattern of spikes during the import process.

- User-level CPU usage (blue line) frequently reached peaks of up to 50%, indicating active computation tasks such as chunking or embedding generation.
- System-level CPU usage (red line) remained relatively low, generally under 10%, suggesting minimal kernel overhead or I/O bottlenecks.

The intermittent nature of CPU spikes suggests that the workload alternated between GPU-intensive and CPU-intensive tasks, possibly during transitions between parsing and vectorization.



Performance Bottlenecks

While GPU utilization was consistently high, the variability in CPU usage suggests potential inefficiencies in workload distribution between the CPU and GPU.

The periods of lower GPU utilization could indicate waiting times for preprocessing tasks or I/O operations (e.g., reading from storage or writing to a database).

Throughput Observations

The import process handled an average of approximately 61 pages per document, which is a moderate workload for this test. However, the lack of detailed timing data makes it difficult to assess whether throughput was consistent across all documents.

If certain documents or pages caused longer processing times, it could point to variability in document complexity (e.g., OCR-heavy PDFs or larger-than-average pages).

System Stability

No signs of instability (e.g., OOM errors or excessive system-level CPU usage) were observed during this test. This indicates that the system handled the import workload without encountering critical failures.

Resource utilization patterns suggest that there is headroom for optimizing task parallelism or scaling up throughput by leveraging additional hardware resources.

3.2.3.4. Conclusion

Both configurations (A10-45 and T1-45) deliver comparable response times (~10 seconds per query), which includes input analysis and full prompt creation. However, their resource utilization profiles differ significantly:

GPU Usage:

- **A10-45:** Sustains 100% GPU utilization throughout testing, indicating full utilization of its NVIDIA A10 GPU.
- **T1-45:** Exhibits variable GPU utilization (60–100%), suggesting less consistent load handling under stress.

CPU Utilization:

- **A10-45:** Maintains minimal CPU usage (3%), freeing resources for parallel tasks (e.g., embedding model execution).
- **T1-45:** Peaks at 20% CPU usage, potentially limiting capacity for CPU-bound processes like embeddings.

Implications

***A10-45's Efficiency*:** Its low CPU usage and high GPU dedication make it ideal for hybrid workloads (e.g., running embeddings alongside LLM inference on the same machine).

T1-45's Limitations: Higher CPU demands reduce flexibility for multitasking, which could bottleneck systems during peak embedding or preprocessing tasks.

This analysis highlights **A10-45 as the preferred configuration** for balancing GPU performance with CPU availability, critical for optimizing resource mutualization in production.

3.2.3.5. Model choices

Chat and Resume

Based on the validation results, Llama-3.1-8B has been selected for production deployment. Below is the detailed rationale structured for clarity [13].

Purpose:

Chat Interactions: Generate context-aware responses to technical queries from pool technicians.

Chunk Summarization: Preprocess document chunks during vectorization to improve embedding quality.

Chosen Model: *meta-llama/Meta-Llama-3.1-8B-Instruct* (https://ollama.com/library/llama3.1)

Key Features:

- 128K Token Context Window: Processes long technical documents efficiently [13].
- 4-bit Quantization Support: Reduces VRAM usage by ~50% while retaining performance [13].
- Multilingual: Supports French, German, and Spanish for localized technician queries [R00].

Rationale:

- Accuracy vs. Cost: Achieved 71.43% OK + Partial OK in technical Q&A at €0.76/hour (A10-45 GPU), vs. Mistral-Small-24B's 62.63% at €1.65/hour.
- Resource Efficiency: 3% CPU utilization allows parallel embedding tasks (e.g., BAAI/bge-M3) on the same machine.

- ➢ Scalability: 128K token context handles entire technical manuals in one pass, reducing chunking complexity.
- ➢ Regulatory Alignment: GDPR-compliant deployment via Ollama's on-premise inference [R13].

Embeddings

Purpose: Generate vector representations of document chunks for similarity search in the vector database (`pgvector`) [12].

Chosen Model: *BAAI/bge-M3*

Recognized as one of the best-performing embedding models in recent benchmarks [14].

Rationale:

- ➢ Recognized as one of the best-performing embedding models in recent benchmarks.
- ➢ Proven efficiency from prior phases makes it a reliable choice for this project.

Alternative Testing:

Qdrant/bm25: A promising model with strong benchmark results that may be tested as an alternative for embeddings.

R&D Exploration:

Colpali (vidore/colpali-v1.3): This model proposes a novel approach that bypasses traditional RAG processes. However, early tests have shown insufficient results to justify further investment at this stage (Colpali Tests). The approach remains under observation for future potential.

3.2.4. Development Guidelines

4. Implementation roadmap

The implementation of the project will proceed in a structured and iterative manner to ensure stability, scalability, and performance. Each phase focuses on critical aspects of the system's development and validation.

4.0.1. Phasing strategy

The implementation of the project will follow a structured and iterative approach to ensure stability, scalability, and performance. Below is the phased roadmap:

4.0.1.1. LLM Benchmarking

The first phase involves **LLM Benchmarking**, where candidate models such as Llama-3.1-8B and Mistral-Small-24B are analyzed for their quality, response accuracy, and suitability for technical Q&A tasks. The benchmarking results will guide the selection of the most appropriate model for production deployment.

4.0.1.2. Architecture Design for RAG

Next is the **Architecture Design for RAG**, which aims to finalize the Retrieval-Augmented Generation (RAG) system's architecture. The focus will be on modularity and scalability, with an emphasis on decoupling parsing, vectorization, and retrieval phases to facilitate seamless integration of new components or technologies.

4.0.1.3. Batch Import Design

The third phase addresses **Batch Import Design** by developing a robust ingestion pipeline capable of handling high document volumes efficiently. This design will include mechanisms for error handling, transactional integrity, and retry logic to ensure data consistency throughout the process.

4.0.1.4. LLM Quality Testing

In the **LLM Quality Testing** phase, selected models will be validated using domain-specific datasets, such as 42 questions derived from 5 technical documents. Key metrics like OK rate, Partial OK rate, and response time will be analyzed to confirm that the models meet predefined quality thresholds.

4.0.1.5. LLM Performance Testing for GPU Dimensioning

LLM Performance Testing for GPU Dimensioning follows next. Stress tests will be conducted on selected models to evaluate GPU resource requirements using configurations like A10-45 and T1-45. The results will help identify cost-efficient GPU setups suitable for production workloads.

4.0.1.6. Ingestion of High-Volume Documents

The system's resilience will be tested during the **Ingestion of High-Volume Documents** phase by importing large datasets, such as 104 documents totaling 6,379 pages. Resource utilization (GPU/CPU usage) will be monitored to identify potential bottlenecks in the ingestion pipeline.

4.0.1.7. Introduction of Reranking

Finally, the Introduction of Reranking phase will implement a mechanism to enhance the relevance of retrieved documents before they are passed to the LLM for response generation. This step includes evaluating its impact on response accuracy and latency to ensure improved user experience.

5. References

1. Apache PDFBox : The Apache Software Foundation. (2025). Apache PDFBox. Retrieved February 28, 2025, from https://pdfbox.apache.org/

2. LangChain4j : LangChain4j. (2025). LangChain4j: A Java Framework for LLM Integration. Retrieved February 28, 2025, from https://github.com/langchain4j/langchain4j

3. Hexagonal Architecture : Cockburn, A. (2005). Hexagonal Architecture: Three Principles and an Implementation Example. Retrieved February 28, 2025, from https://blog.octo.com/hexagonal-architecture-three-principles-and-an-implementation-example

4. Hugging Face : Hugging Face. (2025). Hugging Face Model Hub. Retrieved February 28, 2025, from https://huggingface.co/

5. Ollama : Ollama. (2025). Llama-3.1 Model Library. Retrieved February 28, 2025, from https://ollama.com/library/llama3.1

6. pgvector : pgvector Developers. (2025). pgvector: PostgreSQL Extension for Vector Similarity Search. Retrieved February 28, 2025, from https://github.com/pgvector/pgvector

7. MinIO : MinIO Inc. (2025). MinIO: High Performance Object Storage. Retrieved February 28, 2025, from https://min.io/

8. Prometheus & Grafana : Prometheus Developers. (2025). Prometheus Monitoring System. Retrieved February 28, 2025, from https://prometheus.io/

9. Prometheus & Grafana : Grafana Labs. (2025).w Grafana: Open Observability Platform. Retrieved February 28, 2025, from https://grafana.com/

10. OpenSearch : OpenSearch Project Contributors. (2025). OpenSearch: Distributed Search and Analytics Engine. Retrieved February 28, 2025, from https://opensearch.org/

11. Retrieval-Augmented Generation (RAG) : Lewis, P., Oguz, B., Rinott, R., Riedel, S., & Schwenk, H. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems (NeurIPS), 33, pp. 9459–9474. Retrieved February 28, 2025, from https://arxiv.org/pdf/2005.11401.pdf

12. Vector Databases and Similarity Search : Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data. Retrieved February 28, 2025, from https://arxiv.org/pdf/1702.08734.pdf

13. Large Language Models (LLMs) : Touvron, H., Lavril, T., Izacard, G., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. Meta AI Research. Retrieved February 28, 2025, from https://arxiv.org/pdf/2302.13971.pdf

14. Embedding Models and BAAI/bge-M3 : Chen J., Xiao S., Zhang P., Luo K., Lian D., Liu Z., et al. (2024). Text Embeddings Through Self-Knowledge Distillation. arXiv preprint arXiv:2402.03216v4 [cs.CL]. Retrieved February 28, 2025, from https://arxiv.org/pdf/2402.03216.pdf

6. Annexes

- Glossaire technique
- Référentiels (versions logicielles, configurations)

6.1. ColaPli POC results

6.1.1. Test base :

Type,Question,Pages,Expected Answer

```
Installation Manual,Quelle elle la capacité de dehumidification d'un DF408 ?,[11-8],0-90%

Installation Manual,"Pour un déshumidificateur DF 405 MONO, quelle est l'amperage de la protection électrique à envisager ?",[8-5],Connect to a PVC pipe with a siphon and drain cap.

Installation Manual,"Pour un déshumidificateur DF 405 MONO, comment se passe le raccordement de l'Hygro Control ?",[9],Only qualified brazers should perform brazing.

Installation Manual,"Pour un déshumidificateur DF 405 MONO, que le le code produit de la sonde Hygro control ?",[14],10°C to 40°C

Installation Manual,Ou installer un déshumidificateur DF 405 MONO,[11],20 meters

Installation Manual,"Pour un déshumidificateur DF 405 MONO, quel est le taux d'hygrométrie admissible pour Hygro Control",[9],2 to 6 people

Installation Manual,"Pour un déshumidificateur DF 405 MONO, quel est la Variation de tension acceptable",[8],10A fuse and 30mA differential circuit breaker

Installation Manual,"Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, quelle est l'alimentation nécessaire pour un modèle 120m3 ?",[3],"Clip male and female connectors together, forming a 'V' shape"

Installation Manual,"Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, concernant le sel, quel est le taux recommandé ?",[4],350 kg of cement per m³ of concrete

User Manual,"Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, quelle est le taux d'alcalinité recommandé ?",[5],"0.6%, or 6g/litre, or 6kg/m³"

User Manual,"Pour un volet de sécurité IMM'ax, combien de personne il faut prévoir pour la livraison ?",[6],"Connect to a single phase, protected by a differential circuit breaker"
```

```
User Manual,"Pour un volet de sécurité IMM'ax, lors de la fixation du système Coverlock, sur la fixation des gachettes en paroi, quel type de vis faut-il utiliser ?",[15],6.8 to 7.2
```

6.1.2. Script

```python
import os
import csv
import pickle
import torch
from datetime import datetime
from colpali_engine.models import ColPali, ColPaliProcessor

# Initialize ColPALI model and processor (for query embedding)
model_name = "vidore/colpali-v1.3"
model = ColPali.from_pretrained(model_name, torch_dtype=torch.bfloat16,
device_map="mps").eval()
processor = ColPaliProcessor.from_pretrained(model_name)

# Function to load precomputed embeddings from a file
def load_embeddings(embedding_file: str):
    with open(embedding_file, "rb") as f:
        return pickle.load(f)

# Function to compute query embedding
def compute_query_embedding(query: str) -> torch.Tensor:
    batch_query = processor.process_queries([query]).to(model.device)

    with torch.no_grad():
        query_embedding = model(**batch_query)

        # Combine mean and max pooling for richer query embedding
        reduced_query_embedding = torch.cat([
            query_embedding.mean(dim=1),
            query_embedding.max(dim=1).values
        ], dim=-1)

    # Move to CPU and normalize for consistent similarity scoring
    reduced_query_embedding = reduced_query_embedding.cpu()  # Move to CPU
    return reduced_query_embedding.view(-1).div(reduced_query_embedding.norm())

# Function to perform similarity search on stored embeddings
def similarity_search(query_embedding: torch.Tensor, embedding_files: list):
    results = []

    for embedding_file in embedding_files:
        embeddings = load_embeddings(embedding_file)

        scores = []
        for e in embeddings:
```

```
            # Normalize page embedding before computing similarity
            normalized_e = e.view(-1).div(e.norm())
            score = torch.dot(query_embedding, normalized_e).item()
            scores.append(score)

        # Append results with page indices starting at 1
        for idx, score in enumerate(scores):
            results.append((embedding_file, idx + 1, score))

    results.sort(key=lambda x: x[2], reverse=True)
    return results

# Function to read test cases from CSV
def read_test_cases(csv_file: str):
    test_cases = []
    with open(csv_file, newline='', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        for row in reader:
            # Parse "Pages" field safely
            try:
                pages = eval(row["Pages"]) if row["Pages"].startswith("[") else []
            except Exception as e:
                print(f"Error parsing Pages field: {row['Pages']}. Error: {e}")
                pages = []

            # Parse "Expected Answer" field safely (corrected column name)
            try:
                expected_answer = eval(row["Expected Answer"]) if row["Expected
Answer"].startswith("[") else []
            except Exception as e:
                print(f"Error parsing Expected Answer field: {row['Expected Answer']}.
Error: {e}")
                expected_answer = []

            test_cases.append({
                "type": row["Type"],
                "question": row["Question"],
                "expected_pages": pages,
                "expected_answer": expected_answer,
            })
    return test_cases

# Main test execution
if __name__ == "__main__":
    # Paths
    csv_file = "old-data-test.csv"  # Replace with your CSV file path
    embedding_folder = "embeddings"  # Folder containing stored embeddings

    # Load all embedding files from storage
    embedding_files = [os.path.join(embedding_folder, f) for f in
```

```
os.listdir(embedding_folder) if f.endswith(".pkl")]

    # Read test cases from CSV
    test_cases = read_test_cases(csv_file)

    # Print current date and time
    now = datetime.now().strftime("%A, %B %d, %Y, %I:%M %p %Z")
    print(f"\nCurrent date: {now}\n")

    print("\nRunning Tests...\n")

    top1_ok = 0
    top2_ok = 0
    top3_ok = 0

    for i, case in enumerate(test_cases):
        print(f"Test {i + 1}: {case['question']}")

        # Compute query embedding
        query_embedding = compute_query_embedding(case["question"])

        # Perform similarity search
        search_results = similarity_search(query_embedding, embedding_files)

        # Extract top results (Top 1, Top 2, Top 3)
        top_results_pages = [result[1] for result in search_results[:3]]  # Get page
numbers

        expected_pages = case["expected_pages"]

        print(f"Expected page(s): {expected_pages}")

        # Check Top 1 result
        if len(top_results_pages) >= 1:
            print(f"Top 1 result page: {top_results_pages[0]}")
            if top_results_pages[0] in expected_pages:
                print("Test ok")
                top1_ok += 1
            else:
                print("Test ko")

        # Check Top 2 results
        if len(top_results_pages) >= 2:
            print(f"Top 2 result pages: {top_results_pages[:2]}")
            if any(page in expected_pages for page in top_results_pages[:2]):
                print("Test ok")
                top2_ok += 1
            else:
                print("Test ko")
```

```
        # Check Top 3 results
        if len(top_results_pages) >= 3:
            print(f"Top 3 result pages: {top_results_pages[:3]}")
            if any(page in expected_pages for page in top_results_pages[:3]):
                print("Test ok")
                top3_ok += 1
            else:
                print("Test ko")

        print("-" * 50)

    # Print global summary of tests
    total_tests = len(test_cases)
    print("\nGlobal Test Results:")
    print(f"Total Tests: {total_tests}")
    print(f"Top 1 Tests Passed: {top1_ok}/{total_tests}")
    print(f"Top 2 Tests Passed: {top2_ok}/{total_tests}")
    print(f"Top 3 Tests Passed: {top3_ok}/{total_tests}")
```

Results

```
`config.hidden_act` is ignored, you should use `config.hidden_activation` instead.
Gemma's activation function will be set to `gelu_pytorch_tanh`. Please, use
`config.hidden_activation` if you want to override this behaviour.
See https://github.com/huggingface/transformers/pull/29402 for more details.
Loading checkpoint shards:
100%|████████████████████████████████████████████████████████████████| 2/2 [00:01<00:00,
1.54it/s]

Current date: Tuesday, February 11, 2025, 06:25 PM


Running Tests...

Test 1: Quelle elle la capacité de dehumidification d'un DF408 ?
Expected page(s): [3]
Top 1 result page: 1
Test ko
Top 2 result pages: [1, 4]
Test ko
Top 3 result pages: [1, 4, 15]
Test ko
--------------------------------------------------
Test 2: Pour un déshumidificateur DF 405 MONO, quelle est l'amperage de la protection
électrique à envisager ?
```

```
Expected page(s): [3]
Top 1 result page: 13
Test ko
Top 2 result pages: [13, 8]
Test ko
Top 3 result pages: [13, 8, 17]
Test ko
--------------------------------------------------
Test 3: Pour un déshumidificateur DF 405 MONO, comment se passe le raccordement de l'Hygro Control ?
Expected page(s): [9]
Top 1 result page: 9
Test ok
Top 2 result pages: [9, 4]
Test ok
Top 3 result pages: [9, 4, 12]
Test ok
--------------------------------------------------
Test 4: Pour un déshumidificateur DF 405 MONO, que le le code produit de la sonde Hygro control ?
Expected page(s): [14]
Top 1 result page: 9
Test ko
Top 2 result pages: [9, 17]
Test ko
Top 3 result pages: [9, 17, 4]
Test ko
--------------------------------------------------
Test 5: Ou installer un déshumidificateur DF 405 MONO
Expected page(s): [11]
Top 1 result page: 4
Test ko
Top 2 result pages: [4, 17]
Test ko
Top 3 result pages: [4, 17, 3]
Test ko
--------------------------------------------------
Test 6: Pour un déshumidificateur DF 405 MONO, quel est le taux d'hygrométrie admissible pour Hygro Control
Expected page(s): [9]
Top 1 result page: 9
Test ok
Top 2 result pages: [9, 4]
Test ok
Top 3 result pages: [9, 4, 17]
Test ok
--------------------------------------------------
Test 7: Pour un déshumidificateur DF 405 MONO, quel est la Variation de tension acceptable
Expected page(s): [8]
Top 1 result page: 17
```

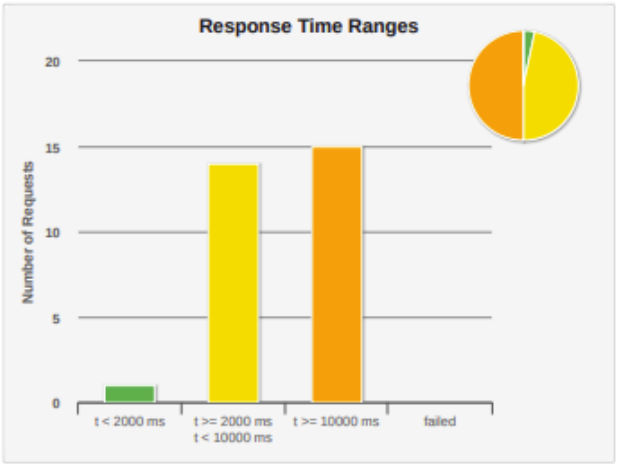
```
Test ko
Top 2 result pages: [17, 3]
Test ko
Top 3 result pages: [17, 3, 4]
Test ko
--------------------------------------------------
Test 8: Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, quelle est l'alimentation nécessaire pour un modèle 120m3 ?
Expected page(s): [3]
Top 1 result page: 3
Test ok
Top 2 result pages: [3, 2]
Test ok
Top 3 result pages: [3, 2, 6]
Test ok
--------------------------------------------------
Test 9: Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, concernant le sel, quel est le taux recommandé ?
Expected page(s): [4]
Top 1 result page: 6
Test ko
Top 2 result pages: [6, 4]
Test ok
Top 3 result pages: [6, 4, 11]
Test ok
--------------------------------------------------
Test 10: Pour un appereil automatique de traitement par électrolyse au sel BIO-POOL, quelle est le taux d'alcalinité recommandé ?
Expected page(s): [5]
Top 1 result page: 4
Test ko
Top 2 result pages: [4, 6]
Test ko
Top 3 result pages: [4, 6, 7]
Test ko
--------------------------------------------------
Test 11: Pour un volet de sécurité IMM'ax, combien de personne il faut prévoir pour la livraison ?
Expected page(s): [6]
Top 1 result page: 24
Test ko
Top 2 result pages: [24, 6]
Test ok
Top 3 result pages: [24, 6, 6]
Test ok
--------------------------------------------------
Test 12: Pour un volet de sécurité IMM'ax, lors de la fixation du système Coverlock, sur la fixation des gachettes en paroi, quel type de vis faut-il utiliser ?
Expected page(s): [15]
Top 1 result page: 35
```

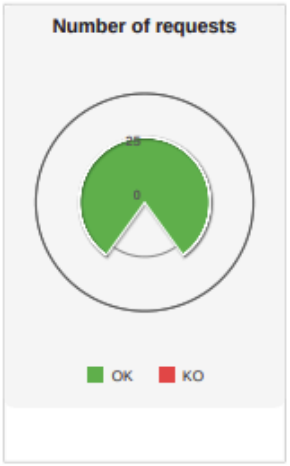
```
Test ko
Top 2 result pages: [35, 9]
Test ko
Top 3 result pages: [35, 9, 10]
Test ko
-------------------------------------------------

Global Test Results:
Total Tests: 12
Top 1 Tests Passed: 3/12
Top 2 Tests Passed: 5/12
Top 3 Tests Passed: 5/12
```

6.2. Gatling report for A10-45

Global Details





Gatling Version

Version: 3.13.4

Released: 2025-02-12

Run Information

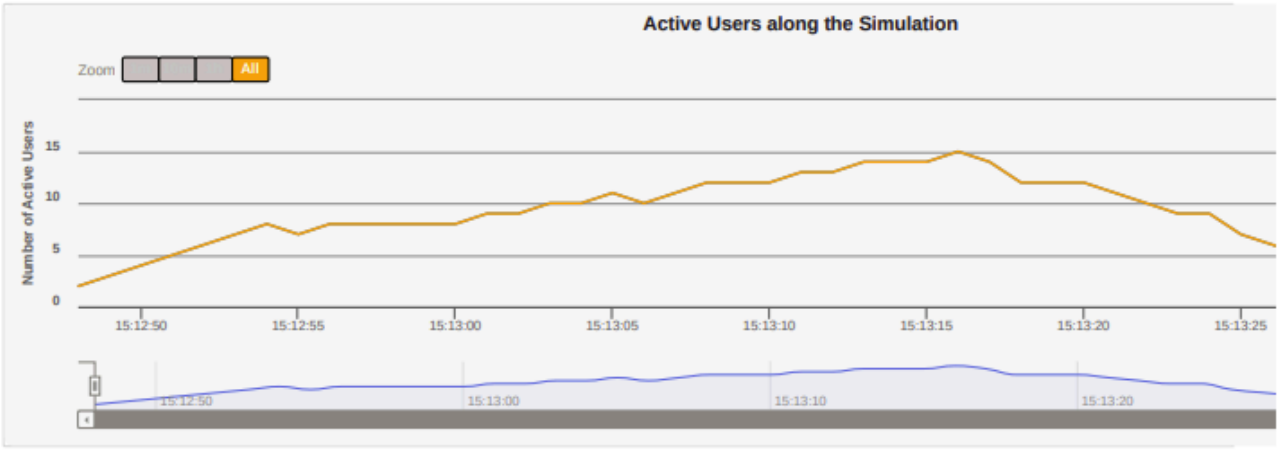
Date: 2025-02-25 14:12:48 UTC

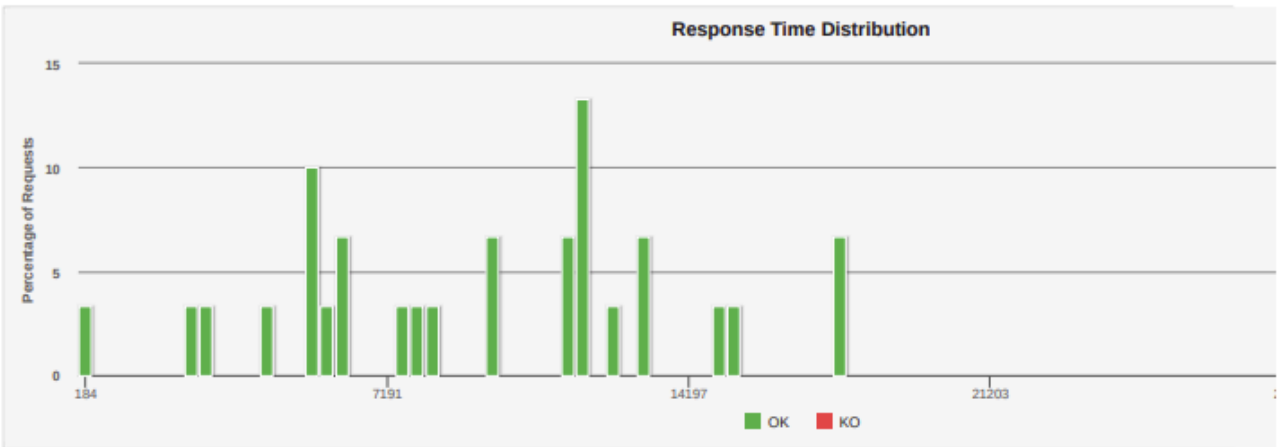
Duration: 48s

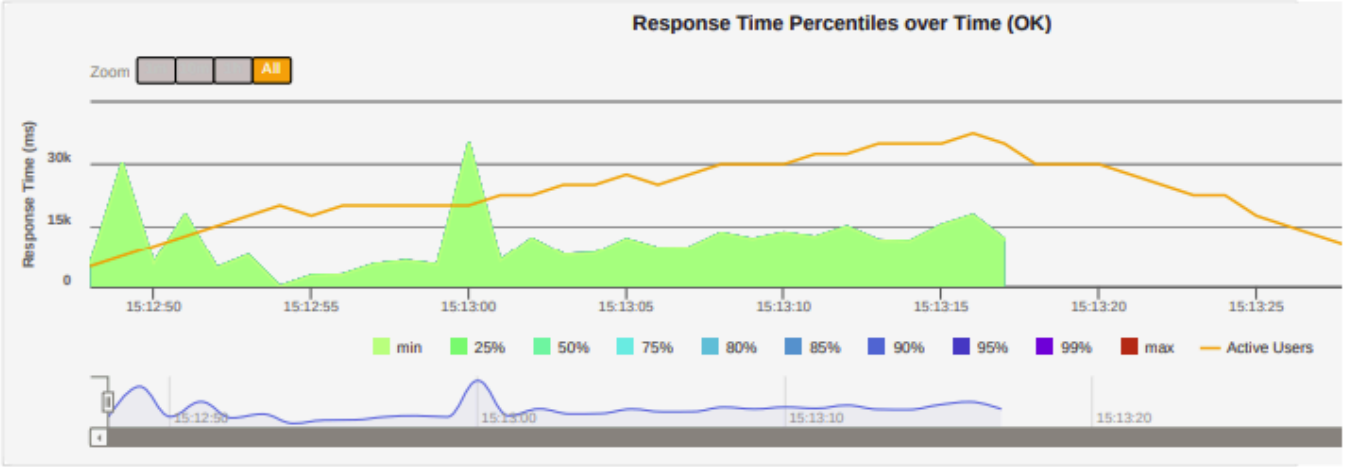
Description: —

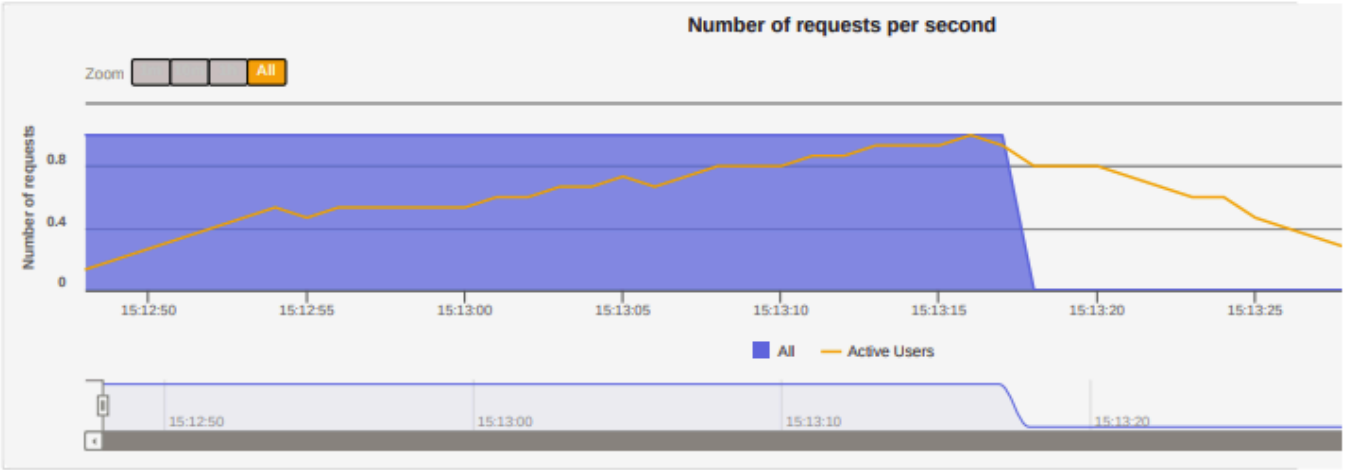
Expand all groups | Collapse all groups

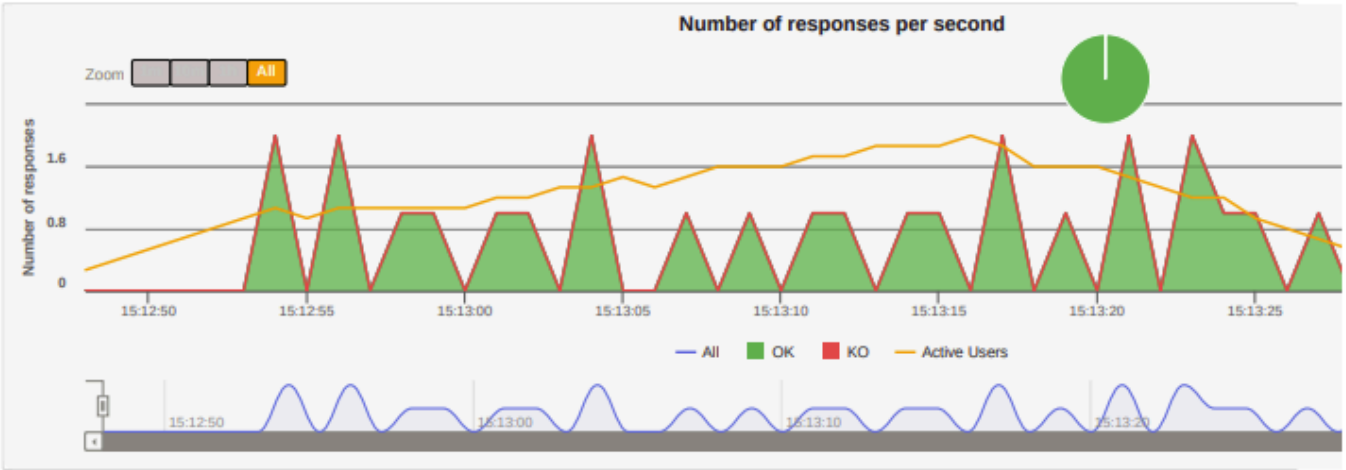
Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	30	30	0	0.00	0.61	9	11,271	13,262	30,205	35,041	35,041	10,886	7,265
Send Question	30	30	0	0.00	0.61	9	11,271	13,262	30,205	35,041	35,041	10,886	7,265





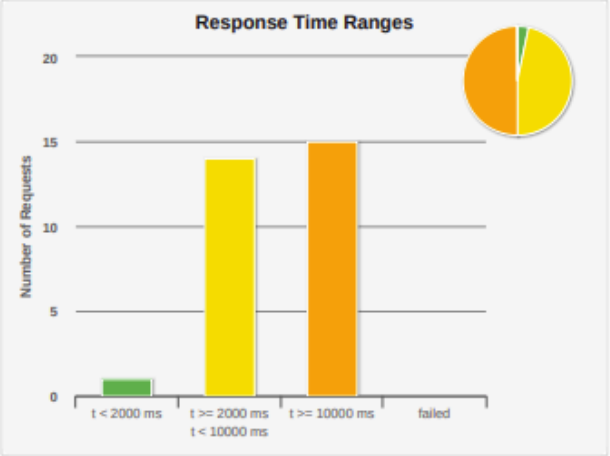

Response Time Percentiles over Time (OK)
Zoom
All
Response Time (ms)
30k
15k
0
15:12:50
15:12:55
15:13:00
15:13:05
15:13:10
15:13:15
15:13:20
15:13:25
min
25%
50%
75%
80%
85%
90%
95%
99%
max
Active Users

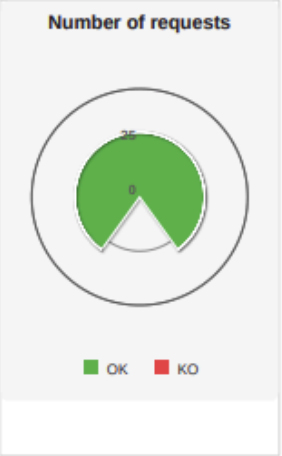

Number of requests per second
Zoom
All
Number of requests
0.8
0.4
0
15:12:50
15:12:55
15:13:00
15:13:05
15:13:10
15:13:15
15:13:20
15:13:25
All
Active Users


Number of responses per second
Zoom
All
Number of responses
1.6
0.8
0
15:12:50
15:12:55
15:13:00
15:13:05
15:13:10
15:13:15
15:13:20
15:13:25
All
OK
KO
Active Users

6.3. Gatling report for T1-45

Global Details





Gatling Version

Version: 3.13.4

Released: 2025-02-12

Run Information

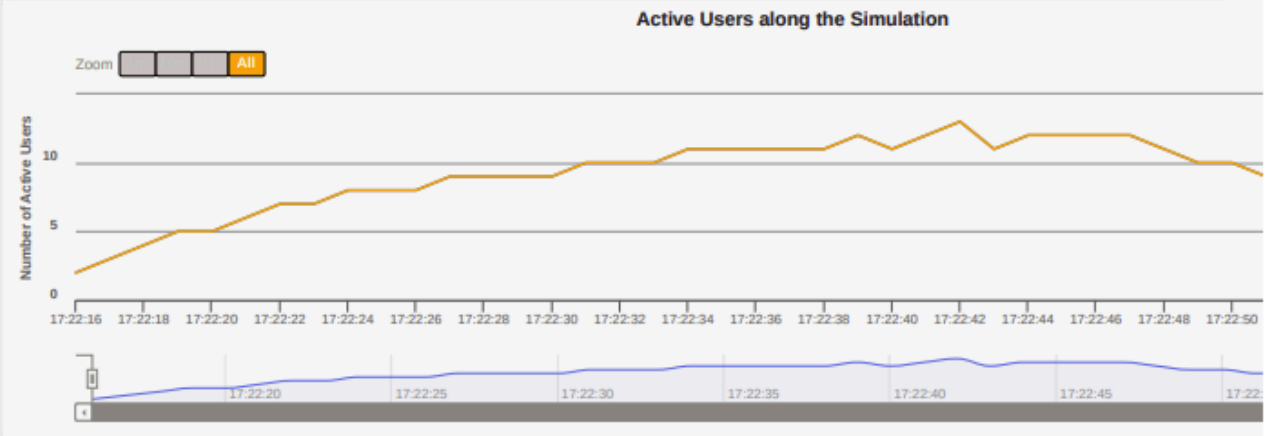
Date: 2025-02-25 16:22:16 UTC

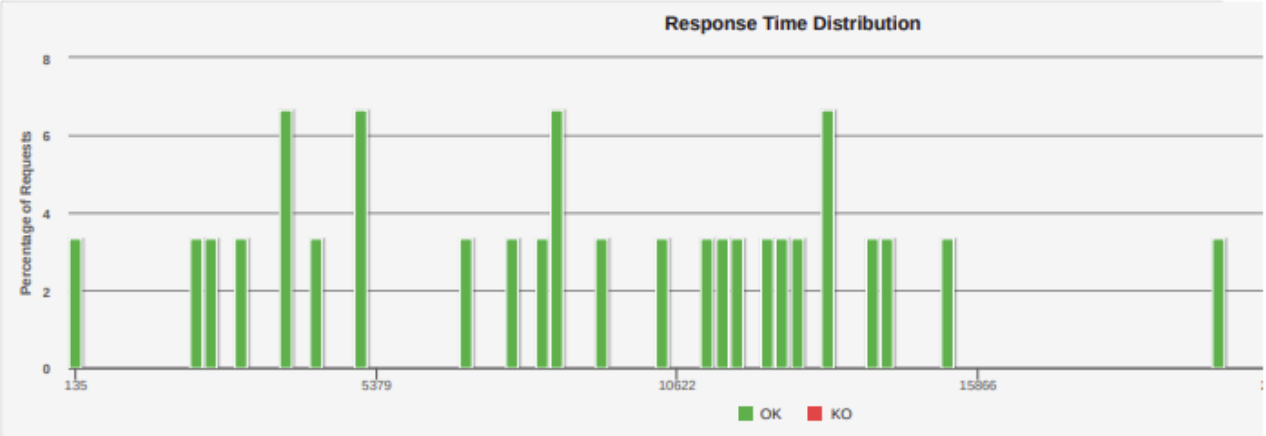
Duration: 44s

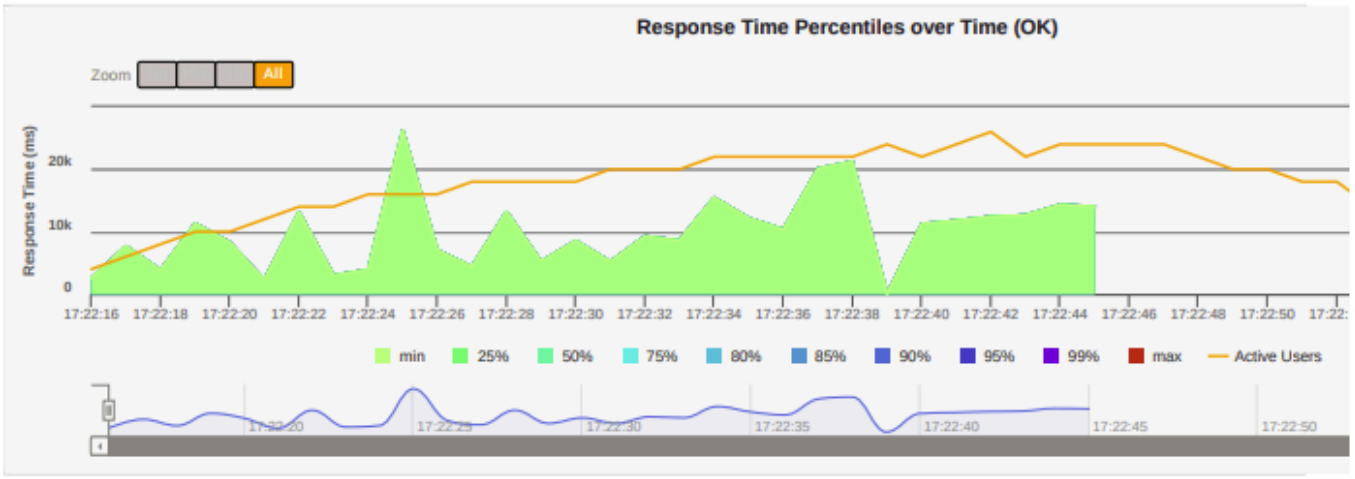
Description: —

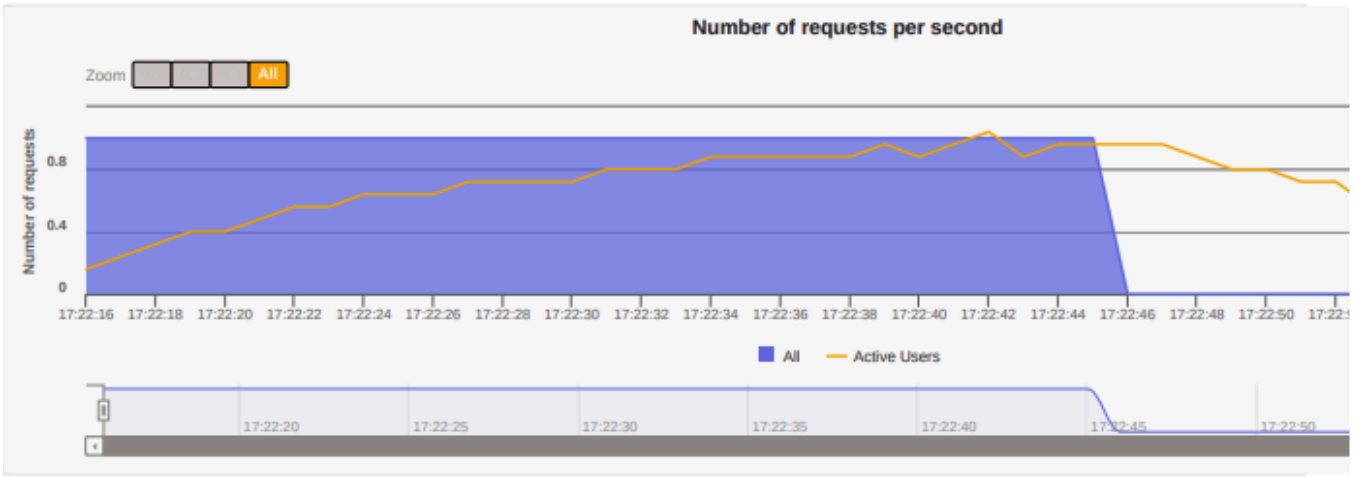
Expand all groups | Collapse all groups

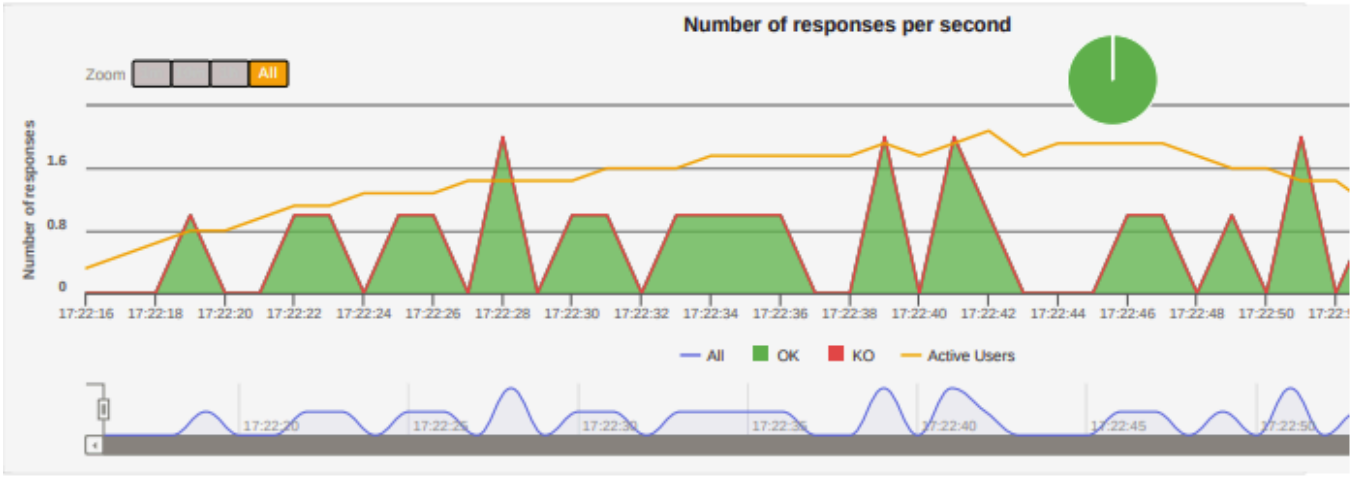
Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	30	30	0	0.00	0.67	4	10,358	13,175	21,239	26,222	26,222	9,985	5,885
Send Question	30	30	0	0.00	0.67	4	10,358	13,175	21,239	26,222	26,222	9,985	5,885






Response Time Percentiles over Time (OK)
Zoom
All
Response Time (ms)
20k
10k
0
17:22:16 17:22:18 17:22:20 17:22:22 17:22:24 17:22:26 17:22:28 17:22:30 17:22:32 17:22:34 17:22:36 17:22:38 17:22:40 17:22:42 17:22:44 17:22:46 17:22:48 17:22:50
min 25% 50% 75% 80% 85% 90% 95% 99% max Active Users
17:22:20 17:22:25 17:22:30 17:22:35 17:22:40 17:22:45 17:22:50


Number of requests per second
Zoom
All
Number of requests
0.8
0.4
0
17:22:16 17:22:18 17:22:20 17:22:22 17:22:24 17:22:26 17:22:28 17:22:30 17:22:32 17:22:34 17:22:36 17:22:38 17:22:40 17:22:42 17:22:44 17:22:46 17:22:48 17:22:50
All Active Users
17:22:20 17:22:25 17:22:30 17:22:35 17:22:40 17:22:45 17:22:50


Number of responses per second
Zoom
All
Number of responses
1.6
0.8
0
17:22:16 17:22:18 17:22:20 17:22:22 17:22:24 17:22:26 17:22:28 17:22:30 17:22:32 17:22:34 17:22:36 17:22:38 17:22:40 17:22:42 17:22:44 17:22:46 17:22:48 17:22:50
All OK KO Active Users
17:22:20 17:22:25 17:22:30 17:22:35 17:22:40 17:22:45 17:22:50