Integrating Large Language Models into a Business-Critical Application in a Regulated Environment: An Experience Report

Elton F. de S. Soares
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
elton.soares@bndes.gov.br

Marcio F. Cornelio
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
marcio.cornelio@bndes.gov.br

Pedro H. de A. Sena
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
pedro.sena@bndes.gov.br

Caio H. D. Milão do N.
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
caio.milao@bndes.gov.br

Otavio A. Gomes de O.
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
otavio.augusto@bndes.gov.br

Gladstone M. A. Jr.
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
glads@bndes.gov.br

João A. dos S. Lopes
GETEM/DESIS3/ATI/BNDES
Rio de Janeiro, RJ, Brazil
joao.lopes@bndes.gov.br

Abstract

Organizations are increasingly exploring Large Language Models (LLMs) to augment business-critical applications with document analysis, information extraction, and decision support. This paper reports an ongoing one-year pilot project in a government-owned financial institution to integrate LLM-based capabilities into a client application portal. The solution supports analysts in checking whether information manually provided by clients is consistent with corporate legal documents submitted during the application process. We summarize the project context, the two-phase integration strategy, the prototype evaluation approach, preliminary outcomes, threats to validity, and reusable design guidelines. The report also distinguishes the project-management aspect of the integration strategy from the technical prompt-construction mechanism based on traceable page retrieval.

Keywords

Intelligent Software Engineering, Large Language Models, Generative AI, Business-Critical Systems, Regulated Environments

1 Introduction

LLMs enable document analysis, information extraction, summarization, and decision support in processes that involve large volumes of complex text, such as corporate legal documents. In this setting, expert analysis depends on tacit knowledge: analysts interpret terminology, identify relevant omissions, relate provisions to institutional rules, and apply experience from previous cases. Integrating this knowledge into a workflow that uses Artificial Intelligence (AI) requires expert interaction, iterative validation, and mechanisms for translating expert judgment into prompts, schemas, metrics, and review procedures.

Integrating LLM capabilities into business-critical applications also raises engineering and governance concerns: probabilistic outputs, evolving prompts, legacy integration, cybersecurity risks, auditability, access control, data handling, and human oversight. These concerns are amplified when the AI component supports a regulated business process and must operate as decision support rather than as an autonomous decision-maker. In such settings, the engineering problem is not limited to choosing a model or writing prompts: teams must decide how to expose the capability to users, retrieve and inject document context, evaluate intermediate outputs, and keep expert review and governance controls visible from experimental prototype validation to production-ready integration.

The report therefore treats integration as two connected problems: managing project-level risk through staged delivery and constructing prompts through traceable retrieval of document pages. This distinction frames the experience as an engineering contribution, expressed through three research questions: **RQ1** - How can integration risk be reduced? **RQ2** - How can document-analysis workflows be evaluated during prototyping? **RQ3** - What technical, organizational, security, and compliance challenges emerge in a regulated business process?

2 Project Context

The project is an ongoing one-year pilot aimed at introducing AI capabilities into a client application portal. Its goal is to support analysts in checking whether information provided by clients is consistent with submitted corporate legal documents. These documents include corporate instruments, such as Articles of Association, that define governance structures, shareholder rights, decision-making processes, share capital, and related provisions. Their interpretation is critical for due diligence, regulatory compliance, and corporate risk assessment, but it is labor-intensive and difficult to scale [2].

The AI component extracts structured information from submitted documents and compares it with information manually provided by clients, while analysts remain responsible for reviewing outputs and making application-related decisions. Stakeholders included the analysis, compliance, cybersecurity, infrastructure and development (*dev*) teams. The analysis team validated prototype outputs, whereas compliance and cybersecurity specialists reviewed data handling, traceability, access control, and prompt-injection concerns. The core *dev* team had two data scientists, three *dev* interns, one senior *dev*, and one manager, while a third-party contractor *dev* team supports the ongoing integration phase. The project also informed infrastructure decisions, reusable components, architectural guidelines, and governance practices for subsequent AI initiatives.

LLMs also introduce security risks such as prompt injection, in which malicious instructions are embedded in inputs to manipulate model behavior [8]. The project therefore considered guardrails [1], traceability, access control, and human oversight from the beginning; outputs had to remain reviewable and the AI component could not become an autonomous decision-maker.

3 Development and Evaluation Approach

3.1 Two-Phase Strategy and Architecture

The team adopted a two-phase strategy. In Phase 1, it built a standalone prototype with its own UI and backend services, allowing experimentation with OCR, prompts, model configurations, UI alternatives, and evaluation procedures before changing the client application portal. In Phase 2, which is ongoing, the validated capability is being integrated into the portal by reusing backend components and adding authentication, access management, logging, monitoring, and operational controls. Within each phase, deliverables were decomposed into increments refined, implemented, and reviewed in two-week sprints, with partial adoption of Scrum-related practices such as sprint planning, periodic reviews, and backlog refinement [10]. Separating the phases was important because prompt formats, text extraction strategies, output schemas, and evaluation criteria were still unstable during prototyping. Keeping these experiments outside the portal reduced iteration cost and limited the risk of introducing unfinished behavior into the portal.

From a DevOps perspective, this strategy relates to continuous delivery and progressive release practices, which reduce release risk through controlled exposure, gradual rollout, and validation before full user-facing release [5]. We do not present it as a new deployment mechanism. Its contribution is a contextual adaptation to an LLM workflow in a regulated environment, where the standalone prototype served as a learning and governance boundary before portal integration.

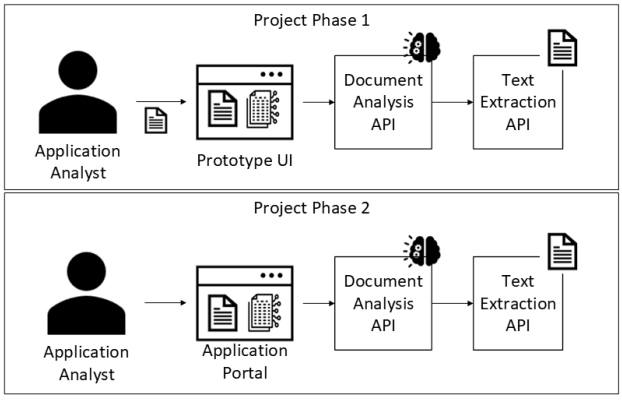

Figure 1: Software architecture across project phases.

The main backend services were the Text Extraction API and the Document Analysis API. The Text Extraction API extracted page-level text from documents using OCR components. The Document Analysis API specialized the extraction workflow for corporate legal documents and interacted with the LLM. After the Text Extraction API processed a document, the Document Analysis API performed regex-based page retrieval to select pages relevant to each group of JSON output fields and injected those pages into prompts. This lightweight retrieval-augmented prompting differs from standard RAG, which often retrieves from external non-parametric memory such as dense vector indexes [7]. The regex-based design was chosen because legal documents contain recognizable textual anchors and because retrieval decisions needed to remain inspectable. This design is less flexible than embedding-based retrieval for open-ended search, but it was adequate for the prototype because the information groups were known in advance and were often associated with recurring legal expressions or document sections. It also simplified debugging: when an extraction failed, the team could inspect the regex pattern, the retrieved page, and the prompt content separately. The same architectural separation also supported reuse: authentication, standardized logging, guardrails, health checks, and error-handling utilities were extracted into shared libraries so that later AI-enabled initiatives could reuse infrastructure and governance mechanisms rather than reimplementing them for each prototype. This reuse-oriented decision was consistent with prior experience in maintaining microservice-based scientific systems and DevOps-oriented pipelines, where modular services and automated delivery mechanisms supported long-term evolution [4, 11].

3.2 Evaluation Approach

The project evaluated two levels: document text extraction and LLM-based information extraction. For text extraction, the team created a benchmark with 9 corporate legal documents, ranging from 2 to approximately 40 pages and including digitally native and scanned content, tables, handwritten signatures, and watermarks. Inspired by public document parsing benchmarks such as OmniDocBench [9], the evaluation used ground-truth references produced through automated extraction followed by human review. A total of 17 tools were evaluated using Normalized Damerau-Levenshtein distance, also known as Normalized Edit Distance (NED) [3, 12]. Unstructured and MinerU were selected for closer analysis; Unstructured was selected as the main text extraction component because, although MinerU had strong public benchmark results, Unstructured performed better on the project document set. This is highlighted in Figure 2, which shows the NED values observed for these tools across the evaluated documents.

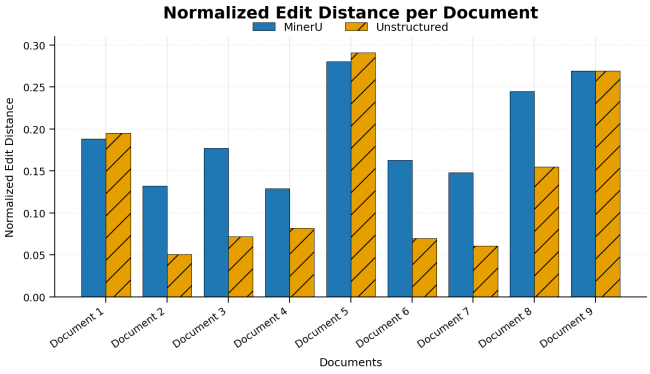

Figure 2: NED per document for each OCR tool.

For LLM-based information extraction, the team created a second benchmark with up to 41 documents and curated JSON ground truth covering fields related to governance, shareholder rights, and share capital provisions. The benchmark evolved during the prototype phase as documents, prompts, and ground truth were revised based on analyst feedback. The output schema was organized into groups

of related fields, and each group was associated with a specific extraction prompt and a set of retrieval patterns. This organization helped the team evaluate not only whether the final JSON was correct, but also whether each prompt received enough document context to support the expected extraction. The team monitored four metrics, summarized in Table 1.

Table 1: Metrics for LLM-based information extraction.

Metric	Definition
Field Presence	Rate of fields in which the system correctly determines whether a value should be retrieved.
Edit Similarity	Complement of NED, 1 − NED, that quantifies character-level similarity between each retrieved field value and its ground truth.
Cosine Similarity	Cosine similarity over text embeddings for quantifying semantic similarity [6].
Exact Match	Proportion of fields in which the retrieved values were identical to their ground truth.

Field Presence was important because an empty field may represent either a system failure or the legitimate absence of information in the document. For each expected field, the metric checks whether the system correctly decided if a value should be returned before evaluating the value itself. Edit Similarity, Cosine Similarity, and Exact Match were then used to assess the quality of fields considered present. This separation helped the team distinguish failures in locating relevant information from failures in normalizing or reproducing the expected value. In practice, this was important because analysts frequently care about both kinds of error: missing a provision that should have been extracted and extracting a value when the document does not actually support that information. The metrics were used to compare prompt combinations, model choices, and regex-based strategies for injecting relevant document pages. Figure 3 shows the metric evolution; values should not be interpreted as directly comparable across all sprints because the documents and ground truth changed.

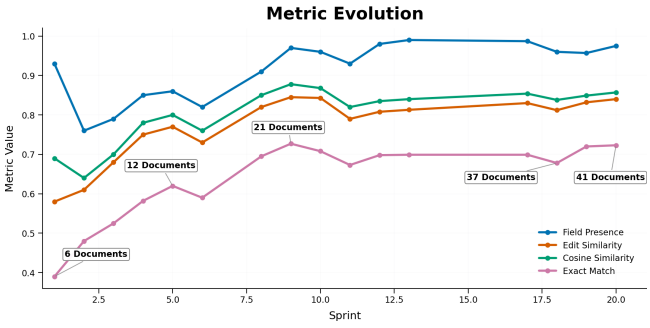

Figure 3: Evolution of metrics across project sprints.

3.3 Validation with Analysts

The evaluation combined metrics with domain-expert review. Early in the prototype phase, analysts reviewed outputs through the prototype UI, and the team manually captured their feedback to refine prompts and formats. This review considered whether the output was syntactically valid, complete, useful for the analyst's task, and aligned with the interpretation expected for the business process. In several cases, analyst feedback identified outputs that appeared structurally correct but were incomplete, ambiguous, or not actionable. Later, the prototype collected usage data semi-automatically, including number of executions, analyst feedback, and document-system similarity based on Edit Similarity. Analyst feedback was collected through a free-text field in the prototype UI, allowing comments immediately after each analysis. The mechanism intentionally avoided interrupting the analysts' workflow, but it also means that the feedback should be interpreted as an operational signal rather than as a controlled user study. At the end of the prototype phase, logs recorded 58 AI executions, a negative feedback rate of 18.96%, and average document-system similarity of 75.08%. Negative feedback reflects perceived usefulness in the analyst's task, not factual accuracy against an independent ground truth.

4 Outcomes, Guidelines, and Limitations

4.1 Preliminary Outcomes

The prototype enabled analysts to inspect AI-generated structured information before portal integration, and analysts reported cases in which AI outputs drew attention to documentation issues requiring review. These reports matter because the system is intended to help identify inconsistencies between submitted documents and client-provided data, not to replace analysts. The project also produced reusable backend components, evaluation datasets, prompt versions, architectural knowledge, and governance insights. These artifacts supported integration planning, helped align the core and contractor teams, and made LLM-specific concerns, such as prompt evolution and retrieval traceability, visible as software engineering concerns. At this stage, however, the project still does not provide robust measurements of business-process impact, such as analysis-time reduction, productivity gains, or rework reduction. Even so, prototype metrics helped the team decide whether the capability was mature enough for integration by comparing configurations, analyzing feedback, and identifying recurrent failure modes before involving the portal development team.

4.2 Challenges and Lessons Learned

The following design guidelines summarize the main lessons and make explicit the evidence and applicability conditions observed in the project: **(1)** Validate LLM capabilities in a standalone prototype before integration into a business-critical system. *Evidence:* the prototype allowed the team to experiment with OCR, prompts, UI alternatives, evaluation procedures, and security constraints before modifying the portal. *Applicability:* systems in which operational stability or compliance requirements make direct experimentation risky. *Confidence:* medium-high. **(2)** Evaluate document-analysis workflows from the beginning using curated datasets and tailored metrics. *Evidence:* benchmark and sprint-level metrics directly supported configuration comparisons and regression detection. *Applicability:* workflows that produce structured outputs from heterogeneous documents. *Confidence:* high in the project context. **(3)** Prefer traceable retrieval mechanisms when retrieval decisions are part of validation. *Evidence:* regex-based page retrieval made it possible to

inspect why specific text was injected into each prompt. *Applicability:* document-analysis tasks with recognizable textual anchors or legal expressions. *Confidence:* medium. **(4)** Use domain-expert validation to complement automated metrics. *Evidence:* analysts identified outputs that were syntactically valid but not useful for the business process. *Applicability:* domains that depend on tacit knowledge or ambiguous terminology. *Confidence:* high. **(5)** Select OCR and model components using project-specific documents and deployment constraints, not only public benchmarks. *Evidence:* the OCR comparison over project documents led to selecting Unstructured despite MinerU's strong public benchmark results. *Applicability:* workflows with domain-specific document formats, noisy scans, or deployment constraints. *Confidence:* medium. **(6)** Treat LLM security and compliance as design constraints rather than deployment checks. *Evidence:* prompt-injection risks, guardrails, traceability, access control, and human oversight were addressed from the beginning. *Applicability:* regulated workflows handling sensitive documents or analyst-facing AI support. *Confidence:* high. Because these guidelines come from a single industrial case, their value is primarily prescriptive rather than predictive: they identify engineering decisions that helped the project move from experimentation toward integration while preserving analyst oversight and governance constraints.

4.3 Threats to Validity

Threats to the validity of the conclusions drawn from this project include the single-organization setting, the specific document class, and relatively small benchmarks. OCR and prompt results may not generalize to other languages, document types, or regulatory contexts. The ground truth was produced through semi-automated drafts curated by analysts, which may introduce bias. Reusing benchmark documents across sprints may also encourage prompt adjustments that fit those documents. The 58 prototype executions are not independent observations, since they may include repeated analyses, related documents, and changing prompts. The feedback mechanism was lightweight and embedded in regular work, so it captures perceived usefulness but not a controlled measure of user satisfaction. The authors also participated in the project, which may introduce interpretation bias despite the use of quantitative metrics and analyst feedback.

5 Conclusion and Future Work

This paper reported an ongoing pilot project integrating LLM capabilities into a business-critical client application portal in a regulated financial context. The experience provides preliminary answers to the research questions by showing how the two-phase strategy reduced integration risk, how benchmark-based and analyst-based evaluation supported prototype validation, and which technical, organizational, security, and compliance challenges emerged.

Future work will complete the integration phase and expand operational evaluation using a broader set of real applications and analyst interactions. The planned evaluation should include operational indicators such as analysis time, rework, analyst workload, frequency of analyst corrections, and cases in which AI-generated inconsistencies change the prioritization or depth of manual review. It should also distinguish failures caused by OCR, retrieval, prompt formulation, and inconsistencies in client-provided data. Future iterations may also expand support to additional legal documents and explore triage or prioritization of client applications while preserving human oversight, cybersecurity controls, and traceability of AI-supported decisions. The operational evaluation will also compare the integrated workflow with the current manual process using logs, analyst feedback, and reviewed cases, focusing on whether AI-generated inconsistencies change inspected sections, correction requests, or prioritization decisions.

Disclaimer. This report summarizes an ongoing internal project and represents research in progress. It reflects the opinions of the authors and does not represent the official position or opinions of their affiliated organizations, their members, or any staff members.

Artifact Availability

Due to confidentiality, security, and regulatory constraints, the corporate documents, datasets, prompts, source code, and internal artifacts cannot be publicly released. To support transferability, the paper describes methodological decisions, metrics, architectural choices, usage indicators, and lessons learned.

Acknowledgements

Microsoft 365 Copilot was used to assist with language revision and to draft selected text fragments based on the authors' prompts, project descriptions, and subsequent manual revisions.

References

[1] Syed Arham Akheel. 2025. Guardrails for large language models: A review of techniques and challenges. *Journal of Artificial Intelligence, Machine Learning and Data Science* 3, 1 (2025), 2504–2512.

[2] Farid Ariai, Joel Mackenzie, and Gianluca Demartini. 2025. Natural language processing for the legal domain: A survey of tasks, datasets, models, and challenges. *Comput. Surveys* 58, 6 (2025), 1–37.

[3] Fred J Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM* 7, 3 (1964), 171–176.

[4] Maximillien De Bayser, Vinícius Segura, Leonardo G Azevedo, Leonardo P Tizzei, Raphael Thiago, Elton Soares, and Renato Cerqueira. 2022. DevOps and microservices in scientific system development: Experience on a multi-year industry research project. In *Proceedings of the 37th ACM/SIGAPP symposium on applied computing*. 1452–1455.

[5] Jez Humble and David Farley. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation.* Addison-Wesley Boston.

[6] Alfirna Rizqi Lahitani, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. 2016. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International conference on cyber and IT service management*. IEEE, 1–6.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[8] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. 2024. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Security 24)*. 1831–1847.

[9] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, et al. 2025. Omnidocbench: Benchmarking diverse pdf document parsing with comprehensive annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24838–24848.

[10] Ken Schwaber and Jeff Sutherland. 2020. The scrum guide. *Scrum Alliance* 21, 19 (2020), 1.

[11] Leonardo P Tizzei, Leonardo Azevedo, Elton Soares, Raphael Thiago, and Rodrigo Costa. 2020. On the maintenance of a scientific application based on microservices: an experience report. In *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 102–109.

[12] Li Yujian and Liu Bo. 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence* 29, 6 (2007), 1091–1095.