Desenvolvimento de um Repositório Institucional Digital com Suporte a IA para a Pró-Reitoria de Pesquisa e Pós-Graduação

Silas Henrique M. Cerqueira[1]

[1]Engenharia da Computação – Universidade Estadual do Maranhão (UEMA)
Cidade Universitária Paulo VI – Caixa Postal 09 – São Luís/MA – Brazil

silas.20210003273@aluno.uema.br

Abstract. *This paper describes the second development phase of the PPG/UEMA Digital Institutional Repository. The work consolidates the initial planning through the implementation of an MVP (Minimum Viable Product). It presents the microservices architecture based on Spring Boot and React, the updated data modeling, interfaces validated via prototyping, and the software quality strategy, including unit and integration tests. The results demonstrate a functional workflow for document submission and approval.*

Resumo. *Este artigo descreve a segunda fase de desenvolvimento do Repositório Institucional Digital da PPG/UEMA. O trabalho consolida o planejamento inicial através da implementação de um MVP (Minimum Viable Product). São apresentados a arquitetura baseada em microsserviços com Spring Boot e React, a modelagem de dados atualizada, as interfaces validadas via prototipagem e a estratégia de qualidade de software, incluindo testes de unidade e integração. Os resultados demonstram um fluxo funcional de submissão e aprovação de documentos.*

1. Introdução

A gestão eficiente da produção académica e administrativa é um desafio crítico para a Pró-Reitoria de Pesquisa e Pós-Graduação (PPG) da UEMA. Dando continuidade à etapa de planeamento e levantamento de requisitos realizada anteriormente, este trabalho foca na construção prática e validação do Repositório Institucional Digital.

O objetivo desta fase foi desenvolver um Produto Mínimo Viável (MVP) que permitisse a submissão, fluxo de aprovação e consulta de documentos, integrando tecnologias modernas de Inteligência Artificial para automação de tarefas.

2. Fundamentação e Metodologia

O desenvolvimento seguiu a metodologia ágil **Scrum**, organizada em *sprints* para entrega incremental de valor, conforme exigido pelo roteiro da disciplina. Paralelamente, utilizou-se a prototipagem de alta fidelidade no Figma para validação antecipada das interfaces de utilizador (UX/UI).

3. Arquitetura e Tecnologias

O sistema opera sobre uma arquitetura cliente-servidor distribuída, garantindo escalabilidade e facilidade de manutenção.

3.1. Backend e Inteligência Artificial

O núcleo do sistema foi desenvolvido em **Java 21** com o framework **Spring Boot 3.5.8**. Para a camada de segurança, utilizou-se Spring Security com autenticação JWT.

A inovação do projeto reside na utilização do módulo **Spring AI** [Spring AI 2025]. Este módulo orquestra a comunicação com Modelos de Linguagem Grande (LLMs) locais. O modelo escolhido foi o **Qwen-2.5** [Qwen 2025], reconhecido pela sua eficiência, executado localmente através da ferramenta **Ollama** [Ollama 2025], garantindo a privacidade dos dados institucionais ao evitar o envio de documentos para APIs externas.

3.2. Processamento de Arquivos

Para a extração avançada de conteúdo em documentos complexos (PDFs com tabelas e formatações variadas), foi implementado um microsserviço em Python utilizando a biblioteca **Docling**. Esta ferramenta permite converter documentos não estruturados em formatos processáveis pela IA [Docling 2025].

3.3. Frontend

A interface foi construída com **React 19** e **TypeScript**, utilizando **Vite** para build otimizado. A estilização recorreu ao **Tailwind CSS** e componentes acessíveis da biblioteca Radix UI, assegurando uma experiência de utilizador responsiva e moderna.

3.4. Infraestrutura

Todo o ambiente de desenvolvimento e produção foi contentorizado utilizando **Docker** [Docker 2025], garantindo a consistência entre os ambientes de teste e deploy. A configuração dos servidores e contentores seguiu as boas práticas de administração de sistemas Linux [Negus 2020].

4. Arquitetura Tecnológica

A persistência dos dados é gerida pelo PostgreSQL, com migrações versionadas via Flyway.

A Figura 1 apresenta o Diagrama Entidade-Relacionamento (DER) que ilustra as interações entre as entidades principais: **Documento, Usuário, Programa** e **FluxoAprovacao**

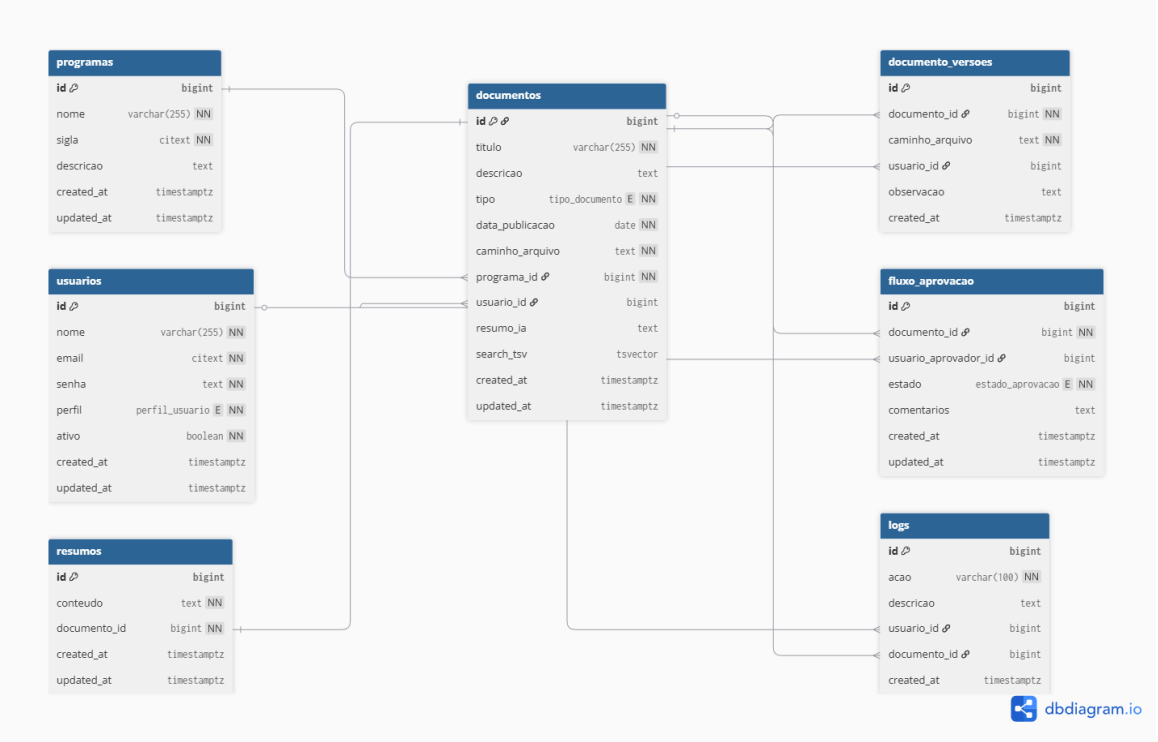

Figura 1 Diagrama Entidade-Relacionamento (DER).

A estrutura de tabelas reflete as regras de negócio onde um documento não pode existir sem um programa vinculado e passa por estados de aprovação geridos pela entidade **FluxoAprovacao.**

5. Interface e Experiência do Usuário

As interfaces foram desenvolvidas fielmente aos protótipos validados, garantindo consistência visual e usabilidade.

5.1. Visão Geral e Dashboard

A tela inicial fornece ao gestor uma visão rápida do sistema.

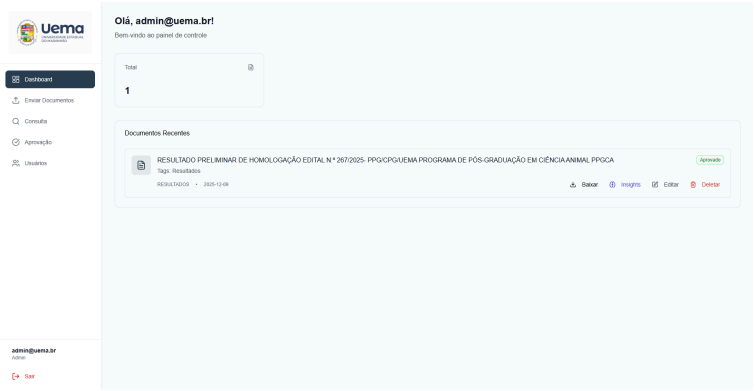

Figura 2. Tela de Dashboard do Sistema

5.2. Fluxo de Submissão e Aprovação

A funcionalidade central permite o *upload* de arquivos e o acompanhamento do seu *status*.

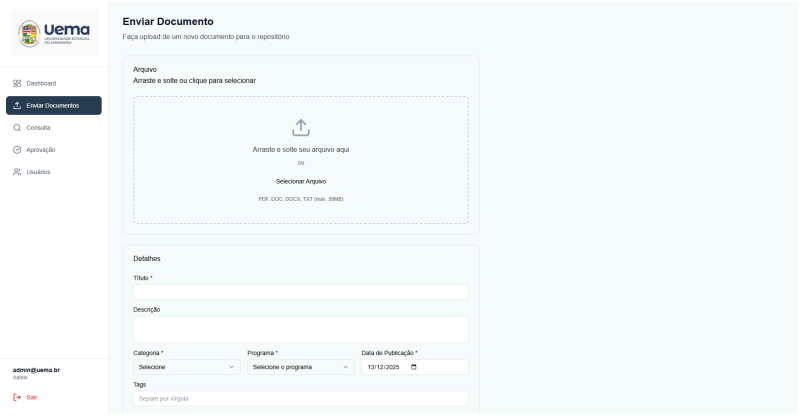

Figura 2. Tela de Submissão de Documentos

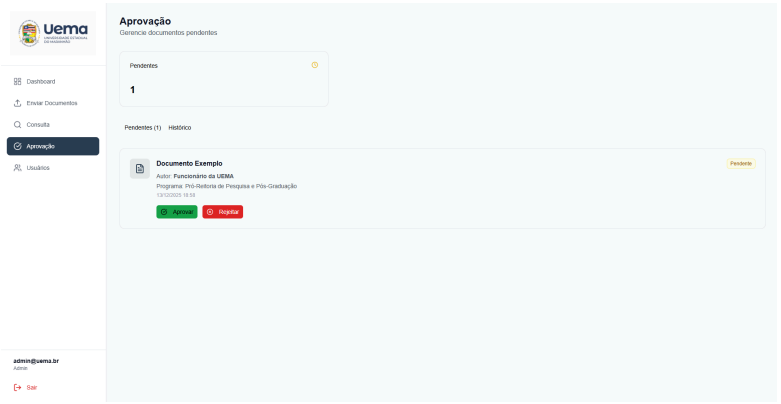

Figura 3. Tela de Aprovação de Documentos

6. Testes e Validação de Software

Atendendo aos requisitos de qualidade, foram implementados testes automatizados utilizando JUnit 5 e Mockito.

6.1. Testes de Unidade

Os testes de unidade focaram no isolamento das regras de negócio na camada de serviço. O Quadro 1 detalha o teste de submissão de documentos por um administrador.

Tabela 1.Caso de Teste de Unidade: Submissão de Documento

Critério	Descrição
Identificador	`uploadQuandoAdminEnviaDocumentoEntaoSucesso`
Classe de Teste	`DocumentoServiceTest.java`
Objetivo	**Verificar se um administrador consegue submeter um documento e se o fluxo de aprovação é iniciado automaticamente.**
Procedimento	**Mock do repositório; Simulação de envio de `MockMultipartFile`; Chamada ao serviço `upload`.**

Resultado Esperado	**Persistência do documento e criação de registo na tabela `fluxo_aprovacao`.**
Resultado Obtido	**Sucesso. O serviço retornou o DTO correto e invocou o `save` do repositório.**

6.2. Testes de Integração

Os testes de integração validaram a resposta dos *endpoints* REST e a interação com o banco de dados H2 em memória.

Tabela 2.Caso de Teste de Integração: Listagem de Documentos

Critério	Descrição
Identificador	**`listarDocumentos_DeveRetornarJSONCorreto`**
Classe de Teste	**`DocumentoControllerIT.java`**
Objetivo	**Garantir que a requisição GET `/documentos` retorna a estrutura JSON correta.**
Configuração	**Banco H2 limpo (`deleteAll`), inserção prévia de 1 Programa e 1 Documento.**
Procedimento	**Requisição HTTP simulada via `MockMvc`.**
Resultado Esperado	**Status HTTP 200 (OK) e JSON contendo o título "Edital Teste Integração".**
Resultado Obtido	**Sucesso. Validação de campos JSON via `jsonPath` confirmada.**

7. Considerações Finais

A implementação do Projeto Final resultou num MVP funcional que atende aos requisitos fundamentais de submissão e controlo de documentos. A integração do **Spring AI** com **Ollama** e **Qwen** demonstrou ser uma solução viável para trazer inteligência ao sistema sem comprometer a privacidade dos dados, enquanto o uso de **Docker** simplificou o *deploy* da infraestrutura. O sistema encontra-se validado por testes e pronto para as próximas fases de expansão na UEMA.

8. Referências

[Docling 2025] DS4SD. "Docling: Document Parsing & Conversion". Disponível em: https://ds4sd.github.io/docling/. Acesso em: 13 dez. 2025.

[Docker 2025] Docker Inc. "Docker Documentation: Containerization Platform". Disponível em: https://docs.docker.com/. Acesso em: 13 dez. 2025.

[Negus 2020] Negus, C. (2020) *Linux Bible*. 10th Edition. Indianapolis: Wiley.

[Ollama 2025] Ollama. "Get up and running with Llama 2, Mistral, and other large language models". Disponível em: https://ollama.com/. Acesso em: 13 dez. 2025.

[Qwen 2025] Qwen Team. "Qwen2.5: A Comprehensive Series of Large Language Models". Hugging Face. Disponível em: https://huggingface.co/Qwen. Acesso em: 13 dez. 2025.

[React 2025] React. "React: The library for web and native user interfaces". Disponível em: https://react.dev/.

[Spring AI 2025] VMware. "Spring AI Reference Documentation". Disponível em: https://docs.spring.io/spring-ai/reference/. Acesso em: 13 dez. 2025.

[Spring Boot 2025] VMware. "Spring Boot Reference Documentation". Disponível em: https://spring.io/projects/spring-boot.

[UEMA 2025] UEMA. "Universidade Estadual do Maranhão". Disponível em: https://www.uema.br/.