UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINICIUS FERREIRA NOVACOSKI

REFATORAÇÃO DO SISTEMA WEB DO E-MUSEU

GUARAPUAVA

2025

VINICIUS FERREIRA NOVACOSKI

REFATORAÇÃO DO SISTEMA WEB DO E-MUSEU

Refactoring of the E-museu Web System

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Sistemas para Internet do Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Dra. Sediane Carmem Lunardi Hernandes

Coorientador: Dr. Diego Marczal

GUARAPUAVA

2025

RESUMO

Este Trabalho de Conclusão de Curso apresentará o processo de refatoração, modernização e expansão do sistema web do E-Museu, uma plataforma digital destinada à catalogação e divulgação de peças da história da informática. A justificativa para a realização deste trabalho se fundamentará nas limitações técnicas identificadas na versão atual da aplicação, que apresenta dificuldades de manutenção, ausência de padronização no código, insuficiência de testes automatizados e carência de funcionalidades relevantes para a continuidade do projeto. Diante desse cenário, o trabalho terá como objetivo aprimorar a estrutura interna do sistema, aumentar sua estabilidade e sustentabilidade a longo prazo e, simultaneamente, implementar novas funcionalidades que ampliarão a utilidade prática da plataforma. Como metodologia, será adotada uma abordagem baseada em boas práticas de engenharia de software, incluindo a aplicação de padrões de projeto, a reorganização de módulos, a padronização de versionamento e a configuração de ambientes de desenvolvimento, teste e produção. Além disso, serão utilizados princípios de refatoração para melhorar legibilidade, modularidade e desempenho do código, enquanto testes automatizados serão incorporados para garantir a confiabilidade das funcionalidades e prevenir problemas futuros. Entre os principais resultados esperados estarão a reestruturação completa do código-fonte, a melhoria significativa na organização e rastreabilidade do projeto, a implementação de mecanismos de segurança, o aumento da capacidade de upload de mídias, a internacionalização do sistema e a geração de etiquetas para catalogação física das peças. Espera-se que a conclusão evidencie que o conjunto de melhorias aplicadas tornará o E-Museu mais robusto e coerente com práticas modernas de desenvolvimento, estabelecendo uma base sólida para evoluções futuras e contribuindo para a preservação da memória tecnológica.

Palavras-chave: museu; refatoração; funcionalidades; engenharia de software; sistema web.

ABSTRACT

This Final Project will present the process of refactoring, modernizing, and expanding the E-Museu web system, a digital platform designed for cataloging and disseminating items from the history of computing. The justification for carrying out this work will be based on the technical limitations identified in the current version of the application, which presents maintenance difficulties, lack of code standardization, insufficient automated tests, and the absence of essential functionalities for the continuity of the project. In this context, the work will aim to improve the system's internal structure, increase its long-term stability and sustainability, and simultaneously implement new features that will expand the platform's practical usefulness. As methodology, an approach based on software engineering best practices will be adopted, including the application of design patterns, module reorganization, versioning standardization, and the configuration of development, testing, and production environments. Additionally, refactoring principles will be applied to improve code readability, modularity, and performance, while automated tests will be incorporated to ensure the reliability of features and prevent future issues. Among the expected results are the complete restructuring of the codebase, significant improvements in project organization and traceability, the implementation of security mechanisms, increased media upload capacity, system internationalization, and the generation of labels for the physical cataloging of items. It is expected that the conclusion will demonstrate that the set of improvements applied will make the E-Museu more robust and aligned with modern development practices, establishing a solid foundation for future evolutions and contributing to the preservation of technological memory.

Keywords: museum; refactoring; functionalities; software engineering; web development.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

MoSCoW	*Must-have*, *Should-have*, *Could-have* e *Won't-have*
PHP	*Hypertext Preprocessor*
CI/CD	*Continuous Integration/Continuous Delivery*
DDD	*Domain-Driven Design*
MVC	*Model, View, Controller*
UNICENTRO	Universidade Estadual do Centro-Oeste
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

No campo da computação, a inclinação humana para aperfeiçoar continuamente as tecnologias manifesta-se de forma particularmente significativa. Tecnologias primitivas, baseadas em válvulas a vácuo que controlavam a passagem de energia conforme princípios matematicamente determinados, evoluíram para estruturas quase atômicas, capazes de sustentar computadores modernos e viabilizar o avanço de diversas áreas do conhecimento e da sociedade (MCCARTNEY, 1999). Essa trajetória tecnológica possibilitou o surgimento da Internet e de múltiplos campos de estudo e inovação, todos orientados pelo mesmo princípio: a constante busca por maior eficiência, desempenho, coerência, confiabilidade e inovação, garantindo que os sistemas sejam cada vez mais robustos, adaptáveis e capazes de atender às demandas de um mundo em constante transformação.

No desenvolvimento de software, a lógica do aprimoramento contínuo mantém-se vigente como um princípio essencial. Em um contexto marcado por transformações rápidas e constantes, parte significativa dos sistemas vigentes precisam ser continuamente atualizados, envolvendo refatorações, que visam otimizar e tornar o código mais claro, eficiente e sustentável, assim como a implementação de novas funcionalidades, responsáveis por ampliar a utilidade e a relevância das aplicações frente às novas demandas sociais e tecnológicas, sob pena de obsolescência. Dessa forma, o desenvolvimento de software ocupa posição estratégica, situando-se na vanguarda das transformações tecnológicas, sociais e econômicas, impulsionando a criação de novas formas de interação, produção e conhecimento, além de contribuir para a consolidação de práticas inovadoras e para a manutenção da competitividade no cenário digital contemporâneo.

Nesse contexto de constante evolução tecnológica e de aprimoramento de sistemas, iniciativas educacionais e culturais que envolvem tecnologia digital demandam soluções adaptáveis e capazes de integrar novas funcionalidades de forma contínua. Reconhecendo essa necessidade de adaptação e aprimoramento, as universidades passaram a buscar maneiras de organizar e preservar informações de forma mais estruturada. A Universidade Tecnológica Federal do Paraná (UTFPR) e a Universidade Estadual do Centro-Oeste (UNICENTRO), a partir dos projetos de extensão "Tecno-Lixo: Oficina do Aprender"(RIBEIRO *et al.*, 2024) e "E-Lixo"(Ré; THOMEN, 2021), respectivamente, que trabalham com o reaproveitamento de peças de computadores descartadas, acabaram armazenando diversos itens de informática em suas dependências para criação de um museu, preservando assim um pouco da história da computação. Gonzaga (2024) implementou um museu virtual de informática para cadastro e gerência desses itens, chamado E-Museu.

O E-museu trata-se de um sistema web desenvolvido como espaço de catalogação, gerenciamento e exposição de itens tecnológicos no formato de museu digital. Esse sistema está em uso e encontra-se disponível na Web em e-museu.gp.utfpr.edu.br. Contudo, precisa de refatorações e de aprimoramentos além da adição de novas funcionalidades. Dessa forma,

o E-Museu não apenas reflete a aplicação prática da lógica do desenvolvimento contínuo de software, mas também exemplifica como a inovação e o aperfeiçoamento constante podem potencializar a experiência do usuário e a disseminação do conhecimento cultural e acadêmico.

Esse aperfeiçoamento de software pode apresentar diversos obstáculos ao longo do desenvolvimento. Um dos desafios refere-se à atualização das tecnologias utilizadas no sistema, processo que exige atenção e cautela para preservar sua estabilidade. No âmbito da refatoração, é igualmente essencial garantir que as melhorias estruturais não comprometam o funcionamento atual do software. Por fim, um dos principais desafios consiste em tornar o sistema simples de implementar em ambientes de produção, assegurando confiabilidade e robustez.

Com isso, o presente trabalho tem como objetivo propor soluções e melhorias para tornar o E-Museu um software mais robusto, resiliente a falhas, seguro e de fácil manutenção. Para atingir esses objetivos, serão aplicados padrões de desenvolvimento de software, práticas de teste sistemáticas e padronização de ambientes, visando garantir a qualidade, a confiabilidade e a manutenção eficiente do sistema.

1.1 Objetivos

1.1.1 Objetivo geral

Refatorar e expandir o sistema web do E-Museu por meio da implementação de novas funcionalidades, aplicando boas práticas de programação, padrões de projeto de software e testes automatizados, com o intuito de melhorar a manutenibilidade e a qualidade do código.

1.1.2 Objetivos Específicos

- Estabelecer uma estrutura de versionamento e desenvolvimento de código com padrões bem definidos, utilizando GitHub, de modo a garantir organização, rastreabilidade e colaboração eficiente.
- Configurar ambientes estáveis de desenvolvimento, teste e produção, assegurando consistência, reprodutibilidade e confiabilidade do sistema.
- Refatorar o código existente, aprimorando legibilidade, modularidade, manutenibilidade e desempenho.
- Implementar testes automatizados a fim de assegurar a qualidade, confiabilidade e robustez do código, contemplando testes de fluxo, integração e usabilidade.
- Desenvolver e aprimorar funcionalidades do E-Museu como aumentar a capacidade de *upload* de fotos e vídeos (aumento da capacidade de *upload* para mais de uma foto e

um campo para inserção de link para um vídeo de demostração da peça), internacionalização do sistema, incorporação de mecanismos de segurança como reCAPTCHA e possibilidade de geração de etiquetas para colagem nas peças físicas com o seu código e nome.

- Aplicar padrões de projeto e boas práticas de desenvolvimento, orientando tanto a refatoração quanto a implementação de novas funcionalidades de forma estruturada e eficiente.

1.2 Justificativa

O E-Museu adota uma infraestrutura simplificada, na qual não há distinção entre os ambientes de desenvolvimento e produção em seu ambiente operacional atual. Cada execução da aplicação demanda configurações manuais complexas, como a instalação de tecnologias necessárias ou a preparação do banco de dados. Foi usado como padrão de projeto uma versão simplificada do conhecido padrão de software *Model, View, Controller* (MVC), com a necessidade de implementação da divisão de responsabilidades entre *services*, *actions* e *controllers*, elementos essenciais para a organização e manutenção de sistemas robustos.

Outro ponto crítico é a ausência de mecanismos automatizados que garantam a qualidade e o funcionamento do sistema, como *linters* para análise estática e testes automatizados para validação do comportamento da aplicação. Somado a isso, as funcionalidades existentes podem ser melhoradas, reforçando a necessidade de refatorações e da elaboração de novos recursos, de modo a tornar a plataforma mais completa (IEEE Computer Society, 2025).

Nesse contexto, a refatoração assume papel central, pois permite aprimorar a estrutura interna do código sem alterar seu comportamento externo, garantindo maior clareza, modularidade e facilidade de manutenção. Ao reduzir a complexidade e eliminar duplicações ou práticas inadequadas, a refatoração contribui para a evolução sustentável do software, assegurando que futuras adaptações, correções e expansões sejam implementadas de forma mais ágil, confiável e com menor risco de introdução de falhas (FOWLER, 2004).

2 REFERENCIAL TEÓRICO

2.1 O E-Museu

O E-Museu é uma iniciativa digital dedicada à preservação da história da computação, oferecendo ao público uma plataforma aberta para consulta, e exploração tecnológica. Seu principal propósito é registrar e disponibilizar informações sobre a evolução da informática, abrangendo desde equipamentos clássicos até marcos relevantes do desenvolvimento de hardware. Dessa forma, busca-se garantir que o conhecimento histórico permaneça acessível às futuras gerações, mesmo diante da deterioração física dos materiais originais.

O projeto proporciona uma experiência virtual imersiva que simula a visita a um museu físico, permitindo que os usuários explorem ambientes digitais, observem máquinas antigas, acessem painéis informativos e naveguem por conteúdos históricos de maneira interativa.

Entre os conteúdos disponibilizados, o E-Museu reúne informações técnicas detalhadas sobre componentes eletrônicos que marcaram a evolução da computação, como processadores, placas-mãe, memórias, unidades de armazenamento e periféricos, acompanhados de descrições, imagens e histórico de uso. Além dos componentes individuais.

O projeto adota um modelo colaborativo, permitindo que a comunidade contribua com peças, documentos, fotografias e relatos históricos. Todas as contribuições passam por verificação e catalogação antes de integrarem a coleção, garantindo autenticidade e organização. Essa participação coletiva fortalece o engajamento do público e amplia a diversidade das narrativas preservadas.

Voltado para fins educacionais e culturais, o E-Museu opera como um projeto independente e sem fins lucrativos, comprometido com a divulgação histórica e o fortalecimento da memória tecnológica. Por meio de seu conteúdo acessível e colaborativo, busca promover uma compreensão crítica sobre a evolução da informática e evidenciar a importância de preservar o passado para compreender as inovações do presente.

2.2 Refatoração de software

A refatoração é um processo fundamental no ciclo de desenvolvimento de sistemas, pois busca melhorar a estrutura interna do código sem alterar seu comportamento externo. No E-Museu, essa prática se torna importante para garantir qualidade, legibilidade, extensibilidade e facilidade de manutenção, especialmente diante da evolução constante de requisitos técnicos e funcionais.

De acordo com Fowler (2004), refatorar significa modificar a organização do software de forma sistemática, visando eliminar complexidades desnecessárias, reduzir duplicações, aprimorar o design e facilitar a compreensão do código. Refatoração não é uma etapa isolada, mas

um processo contínuo que acompanha o desenvolvimento, permitindo que melhorias incrementais sejam aplicadas conforme o sistema cresce e amadurece.

Além disso, práticas de refatoração auxiliam na prevenção da chamada “dívida técnica”, evitando que o código se torne difícil de compreender, testar ou modificar. Essa preocupação é especialmente relevante em sistemas acadêmicos e colaborativos, nos quais novas funcionalidades podem surgir a partir de pesquisas, contribuições externas ou mudanças institucionais.

Fowler (2004) também enfatiza o papel dos testes automatizados no processo de refatoração, garantindo que o comportamento do software permaneça correto após cada melhoria estrutural. Dessa forma, refatorar não apenas aprimora o design, mas fortalece a confiabilidade do sistema.

Portanto, no desenvolvimento do E-Museu, a refatoração não é vista como esforço corretivo, mas como estratégia de sustentabilidade do projeto, assegurando que o sistema permaneça acessível, evolutivo, organizado e preparado para futuras demandas tecnológicas e museológicas.

2.3 Padrões de projeto

O objetivo dessa seção é apresentar os principais padrões de projeto considerados para o desenvolvimento do E-Museu, evidenciando como cada abordagem pode contribuir para a organização, qualidade, manutenção e escalabilidade do sistema.

2.3.1 *Model, View, Controller* (MVC)

O padrão *Model, View, Controller* (MVC) estabelece a separação do software em três camadas principais: *Model*, *View* e *Controller* (DEACON, 2009). Essa divisão organiza a aplicação de forma que a lógica de negócios, a interface e o controle de requisições não se misturem, reduzindo acoplamento e facilitando a manutenção. No contexto do E-Museu, o MVC pode ser aplicado para estruturar o fluxo de exibição do acervo digital, manipulação de dados históricos e controle das interações do usuário, tornando o sistema mais claro, previsível e modular. Além disso, o padrão favorece o trabalho colaborativo entre equipes de desenvolvimento, design e curadoria técnica.

2.3.2 *Clean Code*

O princípio de *Clean Code* orienta a escrita de códigos simples, legíveis, coesos e de fácil entendimento, priorizando qualidade sobre complexidade desnecessária (MARTIN, 2009). Sua aplicação incentiva boas práticas como nomes descritivos, funções curtas, ausência de códigos duplicados, clareza semântica e utilização consistente de padrões arquiteturais. Para

o E-Museu, adotar *Clean Code* contribui diretamente para a evolução contínua do sistema, reduzindo custos de manutenção, facilitando correções, incorporando novas funcionalidades com segurança e possibilitando que futuros desenvolvedores compreendam o projeto rapidamente, mesmo sem conhecimento prévio.

2.3.3 *Domain-Driven Design* (DDD)

O *Domain-Driven Design* (DDD) propõe que o desenvolvimento de software seja guiado pelo domínio do negócio, priorizando o entendimento profundo do problema antes da solução tecnológica (EVANS, 2004). Sua abordagem envolve criação de modelos conceituais, definição de linguagens compartilhadas entre equipe e especialistas, além de estruturas arquiteturais orientadas ao domínio. No caso do E-Museu, o DDD pode auxiliar na representação fiel de entidades como itens do acervo, categorias, sugestões da comunidade e processos de curadoria, garantindo consistência lógica e operacional.

3 MATERIAIS E MÉTODOS

Neste capítulo são descritos os materiais e os métodos que serão utilizados no planejamento e no desenvolvimento do projeto. O conteúdo está estruturado em duas partes: a primeira aborda os materiais empregados, e a segunda apresenta os procedimentos adotados nas etapas de elaboração e execução do trabalho.

3.1 Materiais

Na presente seção são apresentadas as tecnologias utilizadas na refatoração e na implementação de novas funcionalidades do E-Museu, sendo elas:

3.1.1 *PHP*

Hypertext Preprocessor (*PHP*) é uma linguagem de programação desenvolvida e mantida por uma ampla comunidade de desenvolvedores *open source*, amplamente utilizada no desenvolvimento de aplicações, especialmente no lado do servidor, sendo capaz de gerar conteúdo dinâmico para a *World Wide Web*. Neste projeto, será utilizada a versão 8.4 (The PHP Group, 2025).

3.1.2 *Laravel*

O *Laravel* é um conjunto de ferramentas desenvolvidas em *PHP* que tem como objetivo facilitar o desenvolvimento de aplicações web, tornando tarefas complexas mais simples de realizar, manter e atualizar, além de oferecer mais segurança. No projeto, anteriormente era utilizada a versão 10, que será atualizada para a versão mais recente disponível, o Laravel 12 (Laravel Team, 2025).

3.1.3 *MySQL*

O *MySQL* é um sistema de gerenciamento de banco de dados relacional, amplamente utilizado no desenvolvimento de aplicações web, que permite o armazenamento, consulta e manipulação de dados de forma eficiente e segura (Oracle Corporation, 2025). No projeto, o *MySQL* é utilizado para gerenciar as informações do E-Museu, garantindo integridade e performance.

3.1.4 *Docker e Docker Compose*

O *Docker* é uma plataforma que permite criar e executar aplicações em ambientes independentes, garantindo isolamento, portabilidade e consistência entre diferentes ambientes de desenvolvimento e produção (Docker, Inc., 2025b). O *Docker Compose* complementa o *Docker*, permitindo coordenar múltiplos desses ambientes de forma simples, utilizando arquivos de configuração. No projeto, essas ferramentas são utilizadas para organizar e gerenciar os ambientes do *PHP*, *Laravel* e *MySQL*, garantindo que todas as dependências sejam inicializadas corretamente (Docker, Inc., 2025b; Docker, Inc., 2025a).

3.1.5 *Git*

O *Git* é a principal ferramenta de controle de versões, utilizada para registrar alterações, organizar, compartilhar e gerenciar o código-fonte de projetos de forma eficiente entre desenvolvedores (Git SCM, 2025).

3.1.6 *GitHub*

O *GitHub* é uma plataforma online integrada ao *Git*, que permite o armazenamento e o compartilhamento de códigos-fonte. Oferece recursos como *pull requests*, *issues*, *projects* e ferramentas de automação, como *Continuous Integration/Continuous Delivery* (CI/CD), que auxiliam na organização, no controle de versão e na documentação dos projetos (GITHUB, 2025).

3.1.7 *Coolify* e *CapRover*

O *Coolify* é uma plataforma de *deploy* e gerenciamento de aplicações que simplifica a hospedagem de sistemas em servidores próprios ou em nuvem (CoolLabs, 2025; CapRover Team, 2025; Railway, 2025). Ela oferece uma interface intuitiva e suporte nativo a containers Docker, possibilitando configurar e monitorar aplicações, bancos de dados e variáveis de ambiente de forma automatizada e segura. No projeto, o *Coolify* será utilizado para implantar e gerenciar o sistema do E-Museu no servidor da UTFPR, tornando o processo de atualização e manutenção mais ágil e confiável.

Como alternativa, o *CapRover* também foi avaliado por oferecer funcionalidades semelhantes de automação de *deploy* e administração de serviços web. No entanto, optou-se pelo uso do *Coolify* por apresentar vantagens importantes, como suporte nativo ao Docker Compose, uso de Nixpacks - uma ferramenta que identifica automaticamente o tipo de projeto e prepara o ambiente necessário para sua execução, além de contar com gestão integrada de projetos,

usuários e permissões, uma comunidade mais ativa, documentação mais completa e melhor integração com o GitHub, que permite automatizar *deploys* de forma mais simples e direta. Já o *CapRover* depende de Dockerfiles manuais e de webhooks configurados para automação.

3.2 Métodos

Nesta seção serão apresentados, em ordem cronológica, os procedimentos que serão adotados nas etapas de elaboração e execução deste Trabalho de Conclusão de Curso.

3.2.1 Levantamento de requisitos

O levantamento de requisitos é a etapa responsável por definir o que será desenvolvido e o que se espera de qualquer projeto. Trata-se de uma fase que exige muita atenção, pois um levantamento impreciso pode comprometer diretamente a qualidade e a precisão do produto final. No caso do E-Museu, essa etapa mostrou-se essencial, sendo amplamente discutida com os professores para garantir o alinhamento entre as necessidades reais e as funcionalidades e refatorações a serem implementadas no sistema.

3.2.2 Priorização *MoSCoW*

Must-have, *Should-have*, *Could-have* e *Won't-have* (*MoSCoW*) é uma técnica de priorização de requisitos em projetos, amplamente utilizada no contexto do desenvolvimento de software, que consiste na divisão dos requisitos em quatro ou cinco categorias distintas, dependendo da preferência (Agile Business Consortium, 2025):

- *Must-have* (deve ter): requisitos essenciais e obrigatórios para o funcionamento mínimo do sistema. Sem eles, o projeto é considerado inviável;
- *Should-have* (deveria ter): requisitos importantes, mas não críticos. Sua ausência pode ser contornada temporariamente, sem comprometer o funcionamento básico;
- *Could-have* (poderia ter): requisitos desejáveis que agregam valor; porém, a implementação pode ser adiada ou descartada se houver limitação de tempo ou recursos;
- *Won't-have* (não terá): requisitos que foram deliberadamente deixados de fora do escopo atual, mas que podem ser considerados em versões futuras do projeto;
- *Wish* (desejar): categoria adicional usada em algumas variações da técnica para representar ideias ou funcionalidades que não fazem parte do planejamento atual, mas refletem desejos ou possibilidades futuras.

Essa técnica auxilia as equipes a gerenciar o escopo e as expectativas de entrega, garantindo que os esforços sejam concentrados nos aspectos mais críticos e de maior valor para o projeto. A necessidade da utilização desse método surgiu após o levantamento de requisitos do projeto, ou seja, quais requisitos seriam prioridade ou não.

3.2.3 Criar padrão de *Gitflow*

O *Gitflow* é um modelo de ramificação (*branching model*) que define uma estrutura organizada para o uso do *Git* durante o desenvolvimento de um projeto (Git Documentation Team, 2025; GitHub, Inc., 2025). Ele propõe um fluxo de trabalho padronizado, com regras claras sobre como e quando criar, mesclar e remover *branches*, facilitando o controle de versões, a colaboração entre desenvolvedores e a entrega contínua de novas funcionalidades. Nesse modelo, o desenvolvimento principal é dividido entre dois ramos estáveis:

- O `main`, que representa o código em produção;
- E o `develop`, que concentra o código em desenvolvimento.

A partir deles, são criados ramos auxiliares para diferentes propósitos:

- `work/`: originalmente chamada de *feature*, é o ambiente onde são criadas *branches* específicas para cada tipo de tarefa no desenvolvimento;
- `release/`: para preparar versões estáveis antes do lançamento;
- `hotfix/`: para correções urgentes diretamente na versão em produção.

A Figura 1 ilustra os ramos do *GitFlow* mencionados acima.

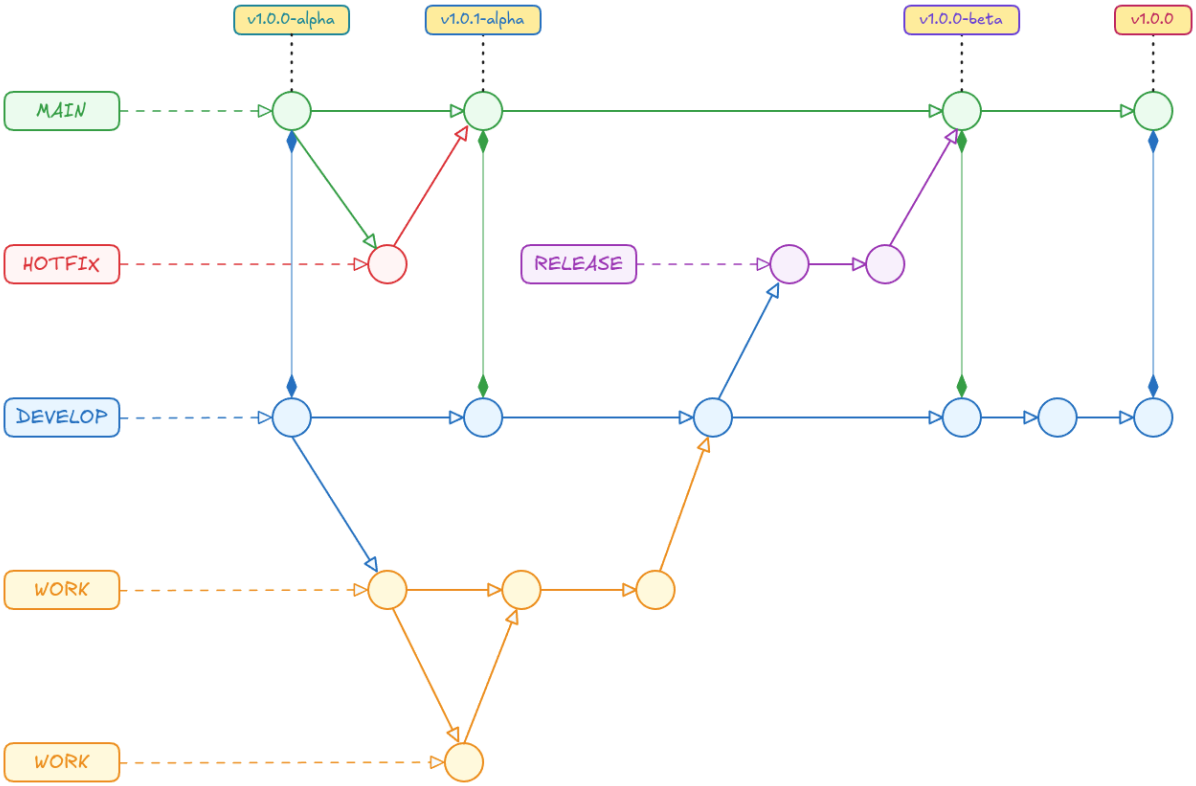

Figura 1 – Fluxo de branches no GitFlow.
Fonte: Autoria própria.

Dessa forma, o *Gitflow* estabelece um ciclo de vida bem definido para cada etapa do projeto, reduzindo conflitos de código e aumentando a rastreabilidade das mudanças.

No contexto do projeto, a padronização inclui a definição de convenções para nomes de *branches* (como `feature/atividade`), mensagens de *commit* claras e descritivas, regras para abertura de *pull requests* e um fluxo de revisão colaborativo. Essa organização contribui para um desenvolvimento mais coeso, rastreável e escalável.

3.2.4 Preparação de ambiente na máquina física na UTFPR para *deploy* automatizado

Antes do início da refatoração e do desenvolvimento de novas funcionalidades do sistema E-Museu, identificou-se a necessidade de automatizar o processo de *deploy*, relacionando diferentes *branches* do repositório com domínios específicos. Assim, a *branch develop* estará associada ao ambiente de testes, acessível em staging.e-museu.gp.utfpr.edu.br, enquanto a *branch main* corresponderá ao ambiente de produção, disponível em e-museu.gp.utfpr.edu.br. Para implementar essa automatização será utilizada a ferramenta *CapRover*, já definida anteriormente neste trabalho. Essa solução permitirá maior controle sobre as versões do sistema e facilitará o processo de atualização contínua das aplicações.

Além disso, a configuração do *Coolify* será realizada diretamente na máquina física localizada na UTFPR, onde o servidor do E-Museu está hospedado. Por questões de segurança

institucional, não há acesso remoto a esse ambiente, o que torna necessário que toda a preparação e manutenção dessa etapa sejam feitas localmente.

3.2.5 Refatoração e desenvolvimento de novas funcionalidades

Com todas as etapas de preparação, infraestrutura e planejamento concluídas, inicia-se a fase principal do projeto, a qual é voltada ao desenvolvimento efetivo do sistema. Essa etapa seguirá o fluxo do *GitFlow* definido anteriormente, com documentação e controle realizados por meio das ferramentas do *GitHub*, como *Pull Requests* e *Issues*. Junto ao desenvolvimento serão realizados testes contínuos para garantir a qualidade, a estabilidade e o correto funcionamento tanto das novas funcionalidades implementadas quanto das já existentes no sistema.

3.2.6 Experimentações e testes

Por fim, após o desenvolvimento, serão realizadas etapas de experimentação e testes para validar o funcionamento do sistema. Inicialmente, os testes ocorrerão no ambiente de *staging*, permitindo a verificação das novas implementações em um ambiente controlado e semelhante ao de produção.

Somente após a validação completa, o sistema será implantado em produção no domínio principal. Em seguida, serão realizadas verificações adicionais para assegurar o funcionamento correto de todas as funcionalidades. Caso sejam identificados erros ou inconsistências, o sistema retornará à etapa de desenvolvimento, na qual as correções necessárias serão aplicadas antes de uma nova implantação.

4 RESULTADOS PARCIAS

Serão aqui apresentados os resultados alcançados até o momento da produção da proposta.

4.1 Levantamento de requisitos

Essa foi a primeira etapa a ser realizada. Os requisitos foram definidos a partir das observações da orientadora, que identificou no sistema E-Museu uma estrutura de código que poderia ser aprimorada, além da ausência de testes automatizados, evidenciando a necessidade de refatoração e do desenvolvimento de novas funcionalidades.

Após reuniões com a orientadora, o coorientador e demais professores envolvidos, os requisitos foram refinados e delimitados, garantindo um foco mais claro e direcionado aos objetivos centrais do trabalho.

4.2 Uso da priorização *MoSCoW*

Com os requisitos levantados, iniciou-se a etapa de priorização utilizando o método MoSCoW. O objetivo é direcionar o desenvolvimento para funcionalidades que geram maior impacto no sistema, otimizando tempo, esforço e recursos disponíveis. A classificação resultante dessa análise pode ser visualizada na Figura 2.

Priorização *MoSCoW* - Refatoração do Sistema E-Museu		
MoSCoW	Requisito	Descrição
MUST	Refatorar os *Controllers*, adicionar *Services* e *Actions*.	A refatoração dos *Controllers* deve remover toda a lógica de negócio e realocá-la em *Services* e *Actions*, garantindo que os *Controllers* atuem apenas como ponto de entrada das requisições. Essa reorganização deveria melhorar a manutenção, clareza e testabilidade do código, enquanto partes adicionais da lógica poderiam ser fragmentadas em *Actions* específicas. Nenhuma nova funcionalidade será adicionada neste processo, focando exclusivamente na reestruturação interna.
	Implementar *deploy* automatizado.	Implementar um processo de *deploy* automatizado na máquina física da UTFPR usando o *Coolify*, garantindo a publicação contínua do site principal (a partir da *branch main*) e de um ambiente de *staging* (a partir da *branch develop*), permitindo atualizações rápidas e sem intervenção manual.
SHOULD	Implementar testes e qualidade de código.	Implementar testes automatizados e mecanismos de verificação para manter a qualidade e a consistência do código durante o desenvolvimento.
	Aumentar a capacidade de upload de fotos e vídeos.	Ampliar a funcionalidade atual para permitir o envio de múltiplas fotos por peça, além de adicionar um campo dedicado para inserir o *link* de um vídeo de demonstração, garantindo uma apresentação mais completa e detalhada do conteúdo.
	Internacionalização do sistema.	Implementar a internacionalização no código utilizando i18n, permitindo que a aplicação ofereça suporte a múltiplos idiomas e facilitando a adaptação de textos e interfaces para diferentes públicos.
	Geração de etiquetas para peças físicas.	Criar um mecanismo para gerar etiquetas contendo o código e o nome de cada peça, permitindo sua impressão e colagem em itens físicos para facilitar identificação e organização.
COULD	Mecanismos de segurança.	Implementar ferramentas de proteção, como o *reCAPTCHA*, para fortalecer a segurança do sistema e evitar acessos ou ações automatizadas maliciosas.
WON´T	Reestilizar o *design* e implementar no E-Museu.	A reestilização do *design* do E-Museu foi cogitada como uma possível melhoria visual e de experiência do usuário; porém, após avaliar o escopo e as prioridades atuais, decidiu-se não realizar a alteração neste momento. Além disso, qualquer mudança exigiria uma pesquisa prévia para validar a necessidade de alterar o *design* atual, seguida de etapas de prototipação e posterior implementação no código, aumentando significativamente o esforço envolvido.

Figura 2 – Priorização *MoSCoW*.
Fonte: Autoria própria.

4.2.1 Elaboração da Documentação do Gitflow

Para garantir um fluxo de desenvolvimento consistente e devidamente documentado, foi realizada, previamente ao início da implementação, uma elaboração completa da documentação referente ao Gitflow, convenções de *commits*, gestão de *issues* e demais práticas relacionadas ao versionamento. Toda essa documentação está disponível no repositório oficial do projeto (Novacoski, Vinicius Ferreira, 2025).

4.2.2 Criação do repositório do E-Museu e configuração de ambiente

O repositório do E-Museu foi criado a partir de um *fork*, que se trata de uma cópia independente de um projeto existente, permitindo sua evolução de forma separada da primeira versão desenvolvida por (GONZAGA, 2024). A partir dele, estruturou-se uma configuração inicial de ambiente utilizando Docker Compose, servindo como base para o desenvolvimento posterior (Novaocoski, Vinicius Ferreira, 2025).

5 CONSIDERAÇÕES FINAIS

As atividades realizadas neste trabalho concentraram-se na refatoração e modernização do sistema web do E-Museu (GONZAGA, 2024), com o objetivo de corrigir limitações técnicas da versão anterior e estabelecer uma base mais sólida para sua evolução. A reorganização do código, a padronização do repositório e a configuração de ambientes com Docker Compose contribuem para criar um ecossistema de desenvolvimento mais estável, previsível e aderente às boas práticas da engenharia de software.

Essas melhorias estruturais resultarão em um sistema mais robusto, seguro e preparado para receber novas funcionalidades. A adoção de padrões arquiteturais e a integração de testes ampliaram a confiabilidade e facilitaram a manutenção, permitindo que o projeto avance com maior controle e qualidade. Além disso, tais ajustes fortalecem a continuidade da proposta original do E-Museu enquanto plataforma de preservação digital.

Com isso, o trabalho estabelece uma base técnica consistente para futuras expansões, contribuindo diretamente para a evolução do E-Museu como ferramenta educacional, cultural e tecnológica. A partir desse novo alicerce, torna-se possível aprimorar a experiência dos usuários, ampliar recursos e assegurar que o sistema esteja alinhado às demandas contemporâneas de desenvolvimento web e preservação histórica.

REFERÊNCIAS

Agile Business Consortium. **MoSCoW Prioritisation**. 2025. Acesso em: 28 out. 2025. Disponível em: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html.

CapRover Team. **CapRover: Build your own PaaS in a few minutes!** 2025. Acesso em: 14 out. 2025. Disponível em: https://caprover.com.

CoolLabs. **Coolify: An Open-Source Self-Hostable Heroku Alternative**. 2025. Acesso em: 14 out. 2025. Disponível em: https://coolify.io.

DEACON, J. **Model View Controller (MVC) Architecture**. 2009. PDF. Acesso em 26 de novembro de 2025. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/50526307/MVC-libre.pdf.

Docker, Inc. **Docker Compose: Define and Run Multi-Container Applications**. 2025. Acesso em: 14 out. 2025. Disponível em: https://docs.docker.com/compose/.

Docker, Inc. **Docker: Empowering App Development for Developers**. 2025. Acesso em: 14 out. 2025. Disponível em: https://www.docker.com.

EVANS, E. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. Addison-Wesley Professional, 2004. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=xCoIAAPGubgC&oi=fnd&pg=PR9&dq=domain-driven+design+book&ots=qdYD8gPK3q&sig=besaDRaOWUTqRwniCWgyUGXM1jg&redir_esc=y#v=onepage&q=domain-driven%20design%20book&f=false.

FOWLER, M. **Refatoração**. Bookman, 2004. ISBN 9788577804153. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=p97eDwAAQBAJ&oi=fnd&pg=PT4&dq=refatora%C3%A7%C3%A3o&ots=l4ruH06Xda&sig=BlsHSJsKhhMPPhvTbGs91JnUeRM&redir_esc=y#v=onepage&q=refatora%C3%A7%C3%A3o&f=false.

Git Documentation Team. **Git - Documentation**. 2025. Acesso em: 14 out. 2025. Disponível em: https://git-scm.com/doc.

Git SCM. **Git: Distributed Version Control System**. 2025. Acesso em: 14 out. 2025. Disponível em: https://git-scm.com.

GITHUB. **GitHub**. 2025. Acesso em: 3 set. 2025. Disponível em: https://github.com.

GitHub, Inc. **About Pull Requests - GitHub Docs**. 2025. Acesso em: 14 out. 2025. Disponível em: https://docs.github.com/en/pull-requests.

GONZAGA, A. T. O. **E-Museu: explorando a história da informática em um ambiente virtual**. 2024. Monografia (Graduação) — Universidade Tecnológica Federal do Paraná, Guarapuava, 2024. Acesso em 17 de março de 2025. Disponível em: https://e-museu.gp.utfpr.edu.br/.

IEEE Computer Society. **The Importance of Software Testing**. 2025. Acesso em: 31 ago. 2025. Disponível em: https://www.computer.org/resources/importance-of-software-testing.

Laravel Team. **Laravel: The PHP Framework for Web Artisans**. 2025. Acesso em: 14 out. 2025. Disponível em: https://laravel.com.

MARTIN, R. C. **Código Limpo: Habilidades Práticas do Agile Software**. Alta Books, 2009. Tradução da obra *Clean Code*. ISBN 9788576082675. Disponível em: https://books.google.com.br/books?id=bmpeDwAAQBAJ.

MCCARTNEY, S. **ENIAC: The Triumphs and Tragedies of the World's First Computer**. Walker Company, 1999. Disponível em: https://dl.acm.org/doi/abs/10.5555/520152.

Novacoski, Vinicius Ferreira. **Gitflow Documentation for Project Development**. 2025. Acesso em: 22 nov. 2025. Disponível em: https://github.com/vinifen/gitflow-documentation.

Novaocoski, Vinicius Ferreira. **E-Museu Repository**. 2025. Acesso em: 22 nov. 2025. Disponível em: https://github.com/vinifen/e-museu.

Oracle Corporation. **MySQL: The World's Most Popular Open Source Database**. 2025. Acesso em: 14 out. 2025. Disponível em: https://www.mysql.com.

Railway. **Nixpacks: Build Applications Without Dockerfiles**. 2025. Acesso em: 14 out. 2025. Disponível em: https://nixpacks.com.

RIBEIRO, J. V. *et al.* Tecno-lixo: Oficina do aprender - relatos de experiência de um projeto de extensão. *In*: **Anais do XIV Seminário de Extensão e Inovação & XXIX Seminário de Iniciação Científica e Tecnológica da UTFPR - SEI/SICITE 2024**. Francisco Beltrão: [*s.n.*], 2024. p. 1–6.

Ré, A. M. D.; THOMEN, M. A. F. **Cartilha de sugestões para atividades em oficinas pedagógicas utilizando de informática LIXOELETRÔ**. 2021. https://www3.unicentro.br/decomp/wp-content/uploads/sites/77/2021/09/CartilhaLixoEletronico.pdf. Acesso em: 25 set. 2022.

The PHP Group. **PHP: Hypertext Preprocessor**. 2025. Acesso em: 14 out. 2025. Disponível em: https://www.php.net.