UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tainá Silva Santos

Plataformas Digitais para a Educação *Online*: Projeto e Desenvolvimento do Observatório de Ensino de História e Geografia

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Tainá Silva Santos

Plataformas Digitais para a Educação *Online*: Projeto e Desenvolvimento do Observatório de Ensino de História e Geografia

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Iara Vieira Guimarães

Coorientador: Maria Adriana Vidigal de Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2025

Tainá Silva Santos

Plataformas Digitais para a Educação *Online*: Projeto e Desenvolvimento do Observatório de Ensino de História e Geografia

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 22 de Setembro de 2025:

Iara Vieira Guimarães
Orientador

Maria Adriana Vidigal de Lima
Coorientador

Uberlândia, Brasil
2025

Agradecimentos

Agradeço a meus pais, Adriane Maria e Fábio Ramos, que sempre prezaram pela minha educação sem qualquer sombra de hesitação. Reconheço profundamente todos os esforços dedicados à minha formação, que me permitiram trilhar um caminho repleto de conhecimentos e experiências essenciais para o meu crescimento profissional e pessoal. Agradeço à minha irmã, Júlia Silva, presença singular na minha jornada, que tanto me ensina e me acolhe nos momentos mais intensos da vida.

Agradeço à mulher que desperta, continuamente, o melhor de mim. Camila Barra, obrigada por ser a companhia que me ensina a enxergar a vida com mais calma, confiança e beleza. Obrigada pela paciência e por ser tão presente.

Agradeço aos amigos que a faculdade me presenteou. Em especial, ao Bruno Ferreira, Huryel Souto e Victor Hugo, que ajudaram a tornar a caminhada dos últimos cinco anos (aproximadamente) significativamente mais leve e prazerosa. Obrigada pelo suporte, pelo estresse compartilhado em trabalhos "infindáveis", pelas risadas e pela paz nos momentos de descontração, e pelo prazer das vitórias conjuntas.

Por fim, e não menos importante, registro minha profunda gratidão aos professores que compartilharam seus conhecimentos e me ajudaram a tornar possível a conclusão desta etapa tão especial da minha trajetória acadêmica. Agradeço, em especial, à professora Iara Vieira, pela orientação dedicada e por me apresentar uma oportunidade de atuação incrível na área da educação; à professora Maria Adriana, pela atenção, simpatia e disposição constantes, que foram essenciais para que este projeto alcançasse seus objetivos; e ao doutorando em Educação Rodrigo Menezes, um profissional e ser humano admirável, cujo apoio e saberes foram fundamentais para o desenvolvimento desta pesquisa. Estendo ainda meus agradecimentos aos professores Selva Guimarães e Fabiano Dorça, que, com tanta generosidade e atenção, aceitaram participar da avaliação e contribuir de forma tão valiosa para este trabalho.

Resumo

Este trabalho apresenta o processo de desenvolvimento e migração do Observatório de Ensino de História e Geografia, plataforma digital vinculada à Faculdade de Educação da Universidade Federal de Uberlândia. Originalmente estruturado em *WordPress* e *Tainacan*, o sistema foi reconstruído com o *framework Flutter* e o ecossistema *Firebase*, visando maior desempenho, escalabilidade, responsividade e autonomia da equipe gestora no gerenciamento dos conteúdos. A pesquisa aborda desde a análise das limitações da versão anterior até a definição de requisitos e a adoção de metodologias ágeis em modelo incremental, contemplando decisões arquiteturais, escolha de tecnologias e estratégias de preservação digital.

Entre os principais avanços alcançados, destacam-se uma interface responsiva, com acesso multiplataforma, a integração de conteúdos associados como o GeoEnsine e a concepção de novos módulos, como o ExpoGEO, que reforçam o caráter agregador e colaborativo do Observatório. Além de disponibilizar uma infraestrutura tecnológica robusta, o projeto contribui para práticas pedagógicas inovadoras, promovendo a difusão crítica do conhecimento em História e Geografia por meio da curadoria digital.

Por fim, esta pesquisa demonstra que a transição tecnológica ampliou a autonomia da equipe e consolidou a plataforma como referência em ambientes digitais de educação, mantendo o compromisso com acessibilidade, preservação e disseminação do conhecimento.

Palavras-chave: Educação *online*; Curadoria digital; Tecnologias educacionais.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABED	Associação Brasileira de Educação a Distância
API	Application Programming Interface (Interface de Programação de Aplicações)
ASD	Adaptive Software Development (Desenvolvimento Adaptativo de Software)
CMS	Content Management System (Sistema de Gestão de Conteúdo)
CI/CD	Continuous Integration and Continuous Deployment (Integração/Entrega Contínua)
CRUD	Create, Read, Update, Delete (Criar, Ler, Atualizar, Deletar)
CSV	Comma-separated Values (Valores Separados por Vírgulas)
DNS	Domain Name System (Sistema de Nomes de Domínio)
DSDM	Dynamic Systems Development Method (Método de Desenvolvimento de Sistemas Dinâmicos)
DRY	Don't Repeat Yourself (Não se repita)
EAD	Educação a Distância
FACED	Faculdade de Educação
FDD	Feature Driven Development (Desenvolvimento Orientado a Recursos)
FIA	Fundação Instituto de Administração
FPS	Frames per Second (Quadros por Segundo)
FTP	File Transfer Protocol (Protocolo de Transferência de Arquivos)
GEPEGH	Grupo de Estudos e Pesquisas em Ensino de Geografia e História
HTML	Hypertext Markup Language (Linguagem de Marcação de Hipertexto)
IBGE	Instituto Brasileiro de Geografia e Estatística

INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
IP	Internet Protocol (Protocolo de *Internet*)
JS	JavaScript
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
MVVM	Model-View-ViewModel (Design Orientado a Domínio)
NoSQL	Not Only SQL (Não Apenas SQL)
OA	Objetos de Aprendizagem
PDF	Portable Document Format (Formato de Documento Portátil)
PHP	Hypertext Preprocessor (Pré-processador de Hipertexto)
PWA	Progressive Web App (Aplicativo Web Progressivo)
RBAC	Role-Based Access Control (Controle de Acesso Baseado em Funções)
REST	Representational State Transfer (Transferência de Estado Representacional)
SAFe	Scaled Agile Framework (Desenvolvimento Orientado a Recursos)
SDK	Software Development Kit (Kit de Desenvolvimento de Software)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
TICs	Tecnologias da Informação e Comunicação
TTFB	Time To First Byte (Tempo Até o Primeiro Byte)
UI	User Interface (Interface do usuário)
UFU	Universidade Federal de Uberlândia
UNESCO	Organização das Nações Unidas para a Educação, a Ciência e a Cultura
URL	Uniform Resource Locator (Localizador Uniforme de Recursos)
WIP	Work in Progress (Trabalho em Andamento)
XP	Extreme Programming (Programação Extrema)

Sumário

1 Introdução

Nos últimos anos, a conexão entre educação e tecnologia tem se tornado cada vez mais significativa, motivada pela necessidade de adaptação às novas exigências educacionais. A crescente digitalização de processos e a disponibilidade de recursos educacionais *online* transformaram a maneira como o conhecimento é acessado, compartilhado e produzido. Esse fenômeno foi acelerado pela pandemia do COVID-19, que obrigou instituições de ensino de diferentes níveis a adotarem modelos remotos em resposta ao isolamento social. Essa pandemia, caracterizada por uma infecção respiratória aguda causada pelo coronavírus SARS-CoV-2, de elevada transmissibilidade e distribuição global, foi decretada no dia 11 de março de 2020.

O impacto gerado pelo vírus evidenciou a importância do uso de plataformas digitais como ferramentas essenciais para a continuidade da educação, ressaltando tanto os benefícios quanto os desafios da transição para o ensino a distância. De acordo com dados da UNESCO (2020), mais de 1,5 bilhão de estudantes foram afetados pelo fechamento de escolas, o que levou à adoção em massa de tecnologias educacionais e ambientes virtuais de aprendizagem.

Nesse contexto, iniciativas que integram plataformas digitais ao ensino de disciplinas específicas, como História e Geografia, ganham relevância por promoverem novas dinâmicas de aprendizagem. Um exemplo concreto dessa integração é o Observatório de Ensino de História e Geografia[1], plataforma digital desenvolvida por docentes e pesquisadores da Faculdade de Educação (FACED) da Universidade Federal de Uberlândia (UFU), criado em 2020, pelo GEPEGH, com o objetivo de ampliar a disseminação do conhecimento nessas áreas e fomentar práticas pedagógicas inovadoras dentro e fora do ambiente universitário.[2]

O Observatório se constitui como um espaço digital dedicado à curadoria de conteúdos e à promoção de ações formativas voltadas ao ensino crítico de História e Geografia. Seu propósito é viabilizar a construção de redes colaborativas entre professores da educação básica e do ensino superior, valorizando experiências didáticas, compartilhando materiais pedagógicos

[1] A plataforma digital Observatório do Ensino de História e Geografia possui certificação concedida pelo Instituto Nacional da Propriedade Industrial (INPI), obtida em 2020, sob o Processo nº BR5120205207-2, com titularidade atribuída à Universidade Federal de Uberlândia. Além disso, o Observatório do Ensino de História e Geografia está registrado como projeto de extensão inovadora no Sistema de Informações de Extensão (SIEX/UFU), sob o nº 20479.

[2] Este trabalho resulta de um projeto de pesquisa iniciado no âmbito da Iniciação Científica, posteriormente desenvolvido com apoio do CNPq e da FAPEMIG. Agradecemos, em especial, à FAPEMIG pelo financiamento da bolsa que viabilizou esta pesquisa.

e incentivando o debate científico sobre os desafios contemporâneos dessas disciplinas. Nesse ambiente, também são abrigados repositórios acessíveis com publicações, documentos, textos acadêmicos e materiais didáticos voltados à formação docente e ao enriquecimento das práticas de sala de aula.

Além disso, o "Observatório"funciona como projeto guarda-chuva para outras iniciativas digitais que emergem no seu âmbito, como as plataformas GeoEnsine, voltada à formação de professores de Geografia, e ExpoGEO, em desenvolvimento, que visa a curadoria de conteúdos voltados para o público escolar. Nessa rede de projetos, a curadoria digital[3] é concebida não apenas como uma ferramenta de organização de conteúdos, mas como uma estratégia pedagógica capaz de aprimorar os processos de ensino e aprendizagem.

A importância de plataformas digitais voltadas para o compartilhamento de conhecimento é, portanto, inegável, tanto no meio acadêmico quanto na sociedade em geral. Segundo o Censo da Educação Superior BRASIL.INEP (2022), o uso de recursos tecnológicos no ensino tem experimentado um crescimento notável, com mais de 70% das instituições de ensino superior adotando plataformas computacionais para a divulgação de conteúdos.

No caso específico de História e Geografia, os recursos digitais permitem abordagens mais dinâmicas, como o uso de geotecnologias, análise de documentos históricos digitalizados e atividades interativas. Essas ferramentas favorecem o desenvolvimento do pensamento crítico, promovem maior interatividade e facilitam a contextualização dos temas escolares. Estudiosos como Tamanini e Souza (2019), defendem que as tecnologias digitais possibilitam um ensino mais multidirecional, rompendo com o modelo pedagógico tradicional centrado na memorização de fatos e datas.

Por sua vez, Quadros e Warpechowski (2019) destacam que Objetos de Aprendizagem (OA), quando integrados de forma adequada ao cotidiano escolar, despertam o interesse dos alunos e potencializam a interdisciplinaridade, desde que acompanhados de formação docente contínua.

Nesse contexto, este trabalho tem como objetivo apresentar o processo de desenvolvimento e migração do *website* do Observatório de Ensino de História e Geografia, primeiramente hospedado na plataforma *WordPress*, que foi reconstruído utilizando o *framework Flutter*. A transição foi realizada com o propósito de modernizar a infraestrutura do Observatório, pro-

[3] Curadoria digital é o processo de seleção, organização, preservação e disponibilização de conteúdos em ambiente digital, de modo a garantir sua relevância, acessibilidade e permanência ao longo do tempo.

porcionando maior flexibilidade de customização, melhor desempenho, responsividade e uma experiência de navegação mais eficiente para os usuários.

O foco principal do projeto foi a reconstrução completa da plataforma digital do Observatório, com o desenvolvimento de um Painel Administrativo restrito, destinado à equipe gestora da Faculdade de Educação (FACED/UFU). Esse painel viabiliza o gerenciamento autônomo e seguro dos conteúdos publicados, incluindo textos, documentos, materiais didáticos e recursos audiovisuais, reforçando a proposta de curadoria digital voltada ao ensino de História e Geografia.

A nova estrutura do *site* foi preparada para contemplar a integração de iniciativas associadas, como a plataforma ExpoGEO, que se encontra em fase inicial de concepção. Embora ainda em desenvolvimento, o projeto ExpoGEO foi considerado, durante a construção do projeto, com o levantamento de requisitos técnicos e pedagógicos realizados em parceria com o doutorando em Educação da UFU, Rodrigo da Silva Menezes. Isso garante que sua futura implementação esteja alinhada ao currículo escolar e à proposta pedagógica do Observatório.

Outra iniciativa vinculada é o GeoEnsine, uma plataforma digital voltada à formação de professores de Geografia, já existente e associada ao Observatório. Embora não tenha sido objeto de reformulação neste trabalho, o novo *site* incorpora *link* direto e integração funcional com o GeoEnsine, reforçando o papel do Observatório como um ambiente agregador de projetos educacionais voltados à área.

As mídias que compõem o acervo, como imagens, vídeos e documentos, estão armazenadas na nuvem[4] por meio do *Firebase Storage*, enquanto os recursos educacionais foram organizados com base em metadados padronizados, assegurando a acessibilidade, a preservação e a fácil navegação no acervo digital. A interface responsiva, desenvolvida com *Flutter*, permite o acesso multiplataforma, tanto em computadores quanto em dispositivos móveis.

Todo o código-fonte do projeto está versionado no *GitHub*[5], promovendo transparência e possibilidade de colaboração. A hospedagem do *site* permanece sendo realizada pela *Locaweb*, mantendo a continuidade da infraestrutura anteriormente utilizada pelo Observatório.

Dessa forma, este trabalho apresenta a materialização de um ambiente digital robusto, colaborativo e inovador, em funcionamento, voltado à valorização do ensino de História e Ge-

[4] No contexto de armazenamento, nuvem é um modelo de serviço que permite guardar, acessar e gerenciar dados em servidores remotos pela internet, sem depender de armazenamento físico local.

[5] Disponível em: <https://github.com/observatoriogeohistoria/observatorio_geo_hist>.

ografia por meio de práticas pedagógicas mediadas por tecnologias acessíveis, escaláveis e alinhadas aos desafios da educação contemporânea.

2 Educação *Online* e o Desenvolvimento de Plataformas de Curadoria Digital

A EAD é uma modalidade educacional que utiliza Tecnologias da Informação e Comunicação (TICs) para promover a mediação didático-pedagógica, permitindo que docentes e discentes participem de atividades de aprendizagem em espaços ou tempos distintos. Essa definição foi prevista pela primeira vez no Decreto nº 5.622, em 19 de dezembro de 2005, Brasil (2005), regulamentando o Art. 80 da Lei nº 9.394/1996, de forma a estabelecer diretrizes e critérios específicos para a oferta de cursos a distância, incluindo a organização segundo metodologia, gestão e avaliação peculiares, com a obrigatoriedade de momentos presenciais para avaliações, estágios, defesa de trabalhos de conclusão de curso e atividades laboratoriais. Além disso, a Associação Brasileira de Educação a Distância (ABED) (2025) caracteriza a EAD como uma oferta educacional organizada em que os processos de ensino e aprendizagem, síncronos ou assíncronos, ocorrem com a utilização de tecnologias, possibilitando que estudantes e professores desenvolvam atividades em lugares e tempos diversos.

Embora esse modelo de ensino não seja um conceito novo, sua popularização ocorreu nas últimas décadas, especialmente com o advento da *Internet*. Inicialmente, as modalidades de EAD eram limitadas a materiais impressos e correspondências. Com a evolução tecnológica, plataformas *online* começaram a emergir, permitindo uma interação mais dinâmica e acessível.

A pandemia de COVID-19 acelerou a transição para o ensino remoto, forçando instituições de ensino em todo o mundo a adotarem modelos online para garantir a continuidade da educação. Como consequência, muitos estudantes enfrentaram desafios significativos, como a falta de acesso a dispositivos e à internet, o que intensificou desigualdades educacionais existentes, conforme apontam Mota e Santos (2024) e Sousa e Lage (2025). Além disso, as instituições precisaram se adaptar, rapidamente, a novas demandas, muitas vezes sem infraestrutura ou recursos pedagógicos adequados para implementar um ensino remoto eficaz.

Apesar da proximidade conceitual, é importante diferenciar a EAD da educação *online*. Enquanto a primeira é uma modalidade formal de ensino, com exigências legais, currículos estruturados e certificação, a segunda representa um campo mais amplo e flexível, que abrange práticas educativas mediadas, digitalmente, muitas vezes de maneira informal, colaborativa e

contínua.

De acordo com Santos (2024), a educação *online* permite experiências de aprendizagem com diferentes graus de institucionalização, abrindo espaço para iniciativas formativas não necessariamente vinculadas a sistemas formais de ensino, como plataformas de recursos educacionais abertos, observatórios, cursos livres e espaços colaborativos entre professores.

Nessa perspectiva, o Observatório de Ensino de História e Geografia se insere como uma iniciativa de formação *online*, ao oferecer um ambiente digital para o compartilhamento e organização de conteúdos educativos, promovendo o engajamento docente, o acesso ampliado a materiais didáticos e o fortalecimento das redes de colaboração na área.

No contexto da educação *online*, as tecnologias digitais de apoio desempenham um papel fundamental ao potencializar a qualidade pedagógica, a acessibilidade, a formação contínua dos profissionais e pesquisadores e a personalização dos processos educativos. Entre essas tecnologias, destaca-se a curadoria digital de conteúdo, que envolve a seleção, organização e compartilhamento de recursos educacionais de forma estratégica, alinhada aos objetivos de aprendizagem e ao perfil dos estudantes.

De acordo com Rocha e Gouveia (2023), a curadoria de conteúdo realizada por professores, bibliotecários e gestores é considerada uma prática essencial para assegurar a qualidade do ensino, contribuindo para a organização coerente dos materiais e a diversificação de formatos que favorecem a aprendizagem ativa. Essa prática também permite otimizar o tempo dos educadores ao disponibilizar recursos pedagógicos relevantes e acessíveis, promovendo uma experiência de aprendizagem mais rica e personalizada. Apesar dos desafios, como a carência de formação específica e a necessidade de maior integração entre os profissionais envolvidos, a curadoria digital mantém-se como uma ferramenta transformadora na construção de ambientes educacionais mais inclusivos e eficazes.

Complementarmente, Moran (2021) destaca o surgimento da curadoria assistida por tecnologias, como algoritmos de recomendação e modelos de inteligência artificial. Esses recursos têm sido utilizados para automatizar parte do processo de seleção de conteúdos, sugerindo materiais adequados ao perfil do estudante ou ao tema em estudo. Segundo o autor, "tal prática contribui para a montagem de trilhas de aprendizagem mais interativas e personalizadas, ao mesmo tempo que favorece a autonomia discente e otimiza o tempo docente". Embora essa abordagem tecnológica não seja aplicada atualmente ao *site* de curadoria digital que é foco deste

trabalho, seu uso futuro pode representar um avanço importante para aumentar a personalização e eficiência na gestão dos conteúdos educativos.

No caso do desenvolvimento do Observatório de Ensino de História e Geografia, a integração de recursos de curadoria digital fortalece sua proposta enquanto plataforma de formação *online*, ao oferecer:

- Exibição de coleções temáticas com metadados bem estruturados, favorecendo a contextualização e a busca por conteúdos específicos;
- Possibilidade de curadoria colaborativa entre docentes e estudantes em ambiente digital;
- Integração com repositórios de objetos de aprendizagem, como imagens, mapas, documentos e infográficos;
- Destaque de recursos por tipo ou relevância pedagógica, contribuindo para uma navegação mais eficiente e uma experiência educacional aprimorada.

Esse enfoque evidencia que a curadoria digital é uma tecnologia de apoio essencial à educação *online*, enriquecendo o valor pedagógico das plataformas digitais e qualificando a experiência formativa por meio de organização, interatividade e alinhamento didático.

2.1 Trabalhos Correlatos

A fim de guiar o desenvolvimento do projeto educacional, foram selecionados três estudos que abordam aspectos-chave da integração entre tecnologia e educação. O objetivo é aproveitar esses trabalhos para criar plataformas educacionais que sejam eficientes, acessíveis e bem estruturadas.

O estudo de Oliveira e Santos (2020) investiga como a pandemia de COVID-19 afetou o ensino no Brasil, com foco na educação básica. A transição repentina para o ensino remoto trouxe muitos desafios, como a falta de acesso a dispositivos tecnológicos e à *internet* para grande parte dos estudantes e professores. Para superar esses obstáculos, os autores analisam o uso de diferentes plataformas digitais e práticas pedagógicas emergenciais adotadas durante esse período. Esse estudo é essencial para o projeto, pois apresenta estratégias eficazes para garantir uma educação mais acessível e equitativa, objetivo central da plataforma do observatório, voltada para o ensino de História e Geografia.

Além disso, Oliveira e Santos (2020) ressaltam a importância de uma curadoria adequada dos conteúdos digitais. A organização eficiente dos materiais é vista como fundamental para garantir o acesso equitativo à informação em um ambiente educacional. Esse ponto se alinha diretamente com os objetivos do projeto, que busca criar plataformas digitais robustas, onde o armazenamento e a distribuição de recursos educacionais sejam sustentáveis e inclusivos. A ideia é não apenas facilitar o acesso ao conhecimento, mas também garantir que ele seja bem preservado e estruturado para uso futuro.

O ebook de Grácio e Madio (2021) se aprofunda na curadoria digital e na preservação de conteúdos educacionais. Os autores discutem a necessidade de técnicas eficazes para organizar e manter informações educacionais disponíveis ao longo do tempo. Essa perspectiva é essencial para o projeto, já que envolve a criação de uma plataforma que não só armazena, mas também preserva os conteúdos de forma que eles permaneçam relevantes para diferentes públicos, incluindo estudantes e educadores.

Essa abordagem dialoga com a preocupação do projeto em garantir que o Observatório de Ensino de História e Geografia funcione como um repositório digital de qualidade. O foco é assegurar que esses conteúdos educacionais estejam sempre atualizados, sejam de fácil acesso e tenham um impacto duradouro. A curadoria digital é, portanto, um pilar central para a construção de uma plataforma confiável e segura.

Por fim, a monografia de Burd (1999) apresenta uma análise sobre o desenvolvimento de plataformas educacionais, em que explora-se como escolher as tecnologias e linguagens de programação certas para criar *softwares* educacionais interativos e eficientes. É enfatizada a importância de projetar interfaces que sejam intuitivas e que melhorem a experiência do usuário, elementos críticos para o sucesso de qualquer plataforma de ensino digital. Esse estudo fornece uma base sólida de práticas e tecnologias para a criação de plataformas educacionais, auxiliando diretamente no desenvolvimento técnico do observatório.

A Monografia também destaca a importância de criar ambientes que incentivem o aprendizado ativo e a participação dos alunos. Isso está alinhado com os objetivos do projeto, que inclui o ExpoGEO, um espaço interativo para o ensino de Geografia. Ao examinar como os *softwares* podem transformar o processo de ensino-aprendizagem, o estudo oferece insights para criar interfaces colaborativas e dinâmicas, ajustadas às necessidades contemporâneas da educação.

Dessa forma, esses três estudos se complementam ao fornecer uma base diversificada para o projeto. Oliveira e Santos (2020) oferecem contexto sobre a urgência de plataformas digitais adaptadas ao ensino remoto. Grácio e Madio (2021) fornecem diretrizes sobre organização e preservação de conteúdos digitais, essenciais para a curadoria de História e Geografia. E ainda, Burd (1999) apresenta as melhores práticas no desenvolvimento de *softwares* educacionais, direcionando a construção de plataformas que sejam interativas e eficientes. Esses trabalhos, juntos, trazem uma visão abrangente sobre acessibilidade, organização e aspectos técnicos necessários para o sucesso de ambientes educacionais digitais.

Este trabalho, por sua vez, utiliza conceitos desses estudos como base, mas também os aprimora em diferentes dimensões. Da discussão de Oliveira e Santos (2020), retoma-se a preocupação com acessibilidade e equidade, materializadas aqui em uma plataforma responsiva, escalável e de acesso multiplataforma. De Grácio e Madio (2021), incorpora-se a ênfase na curadoria e preservação, reforçada no projeto pelo uso de ferramentas modernas como *Firebase Firestore* e *Storage*, que asseguram organização padronizada e durabilidade dos acervos digitais. Já as contribuições técnicas de Burd (1999) são expandidas por meio da adoção de metodologias ágeis, arquitetura em camadas e tecnologias contemporâneas como *Flutter* e *GitHub Actions*. Assim, o Observatório avança em relação aos trabalhos correlatos ao propor uma solução concreta que alia acessibilidade, preservação digital e inovação tecnológica no ensino de História e Geografia.

3 Tecnologias de Desenvolvimento de Software

Howey (2002) apresenta a definição do Dr. Rias van Wyk, da *University of Minnesota* em que as tecnologias de desenvolvimento dizem respeito a "competências manifestadas em dispositivos, procedimentos e habilidades humanas que criam, operam e mantêm sistemas intensivos em *software*". O Dr. van Wyk é pesquisador na área de gestão de tecnologia e seu trabalho fornece uma abordagem conceitual que engloba tanto dispositivos físicos quanto virtuais, procedimentos e habilidades humanas adquiridas, enfatizando que essas tecnologias encapsulam competências criadas e aplicadas. Essa definição reforça a importância das tecnologias de desenvolvimento como base para a criação de plataformas digitais que promovem o ensino, a aprendizagem e a inovação tecnológica, evidenciando que não apenas os artefatos técnicos, mas também o conhecimento embutido neles, é essencial para a construção e manutenção de sistemas complexos.

O desenvolvimento de *software* teve início na década de 1940, acompanhando a criação dos primeiros computadores. Desde então, a evolução das linguagens de programação e dos paradigmas de desenvolvimento tem se acelerado, refletindo as necessidades crescentes da sociedade. Segundo Bergin (2007), a história das linguagens de programação revela tanto os avanços tecnológicos quanto as mudanças na forma como os programadores pensam e estruturam o código. Nos anos 1970 e 1980, surgiram linguagens como *C* e *Pascal*, que introduziram conceitos fundamentais para a programação moderna. A linguagem *C*, por exemplo, tornou-se a base para muitos sistemas operacionais e aplicações devido à sua eficiência e flexibilidade.

Com a comercialização e a popularização da *internet* nos anos 1990, novas tecnologias, como HTML, CSS e *JavaScript* (JS), revolucionaram a criação de aplicações *web*, tornando-as interativas e dinâmicas. Nas últimas décadas, a crescente demanda por aplicações responsivas e multiplataforma impulsionou o surgimento de novos *frameworks*[1] e serviços que simplificam e aceleram o desenvolvimento.

[1] Conjunto de ferramentas, bibliotecas e regras que fornece uma estrutura pronta para desenvolver aplicações, facilitando e padronizando o trabalho do programador.

Entre essas soluções, destacam-se ferramentas como o *Flutter*[2] e o *Firebase*[3], que juntos permitem o desenvolvimento de aplicações modernas, responsivas, escaláveis e com experiência de usuário aprimorada. O *Flutter*, desenvolvido pelo *Google*, possibilita a criação de aplicações multiplataforma a partir de uma única base de código. O *Firebase*, desenvolvido pela mesma empresa, por sua vez, funciona como plataforma de *back-end* em nuvem, oferecendo banco de dados, autenticação, hospedagem, armazenamento, funções em nuvem e outras funcionalidades essenciais para *apps* robustos.

A combinação dessas tecnologias foi essencial para a construção da nova versão do *site* do Observatório de Ensino de História e Geografia, permitindo maior controle sobre a interface, melhor desempenho e integração com recursos modernos de curadoria digital.

A seguir, são apresentadas e detalhadas as principais ferramentas, plataformas e metodologias utilizadas ao longo do desenvolvimento do projeto, destacando suas características técnicas e justificando suas escolhas com base nos objetivos educacionais e tecnológicos da proposta.

3.1 Metodologias de Desenvolvimento de Software

3.1.1 Abordagens Ágeis

Para garantir a entrega contínua de valor e a satisfação do cliente em qualquer projeto de *software*, é essencial adotar uma metodologia que oriente a implementação das funcionalidades demandadas. Essa escolha proporciona um roteiro claro para priorização, planejamento e avaliação de cada etapa, promovendo maior alinhamento entre equipe e cliente, transparência nos processos e previsibilidade nos resultados.

A prática tradicional de desenvolvimento em cascata muitas vezes impede ajustes ao longo do processo e retarda a entrega de valor, uma vez que segue fases rígidas de requisitos, projeto, implementação, testes e implantação. Em contraste, o modelo incremental divide o sistema em partes funcionais, chamadas incrementos, planejadas, implementadas e entregues ciclicamente, enquanto o modelo iterativo repete fases de análise, projeto, implementação e testes, refinando o produto em cada iteração. Essa combinação de ciclos curtos com entregas

[2] Disponível em: <https://flutter.dev/>
[3] Disponível em: <https://firebase.google.com/docs>

parciais permite *feedback* frequente, identificação precoce de falhas e adaptações rápidas, reduzindo riscos e garantindo que cada incremento seja validado pelo cliente.

Inspirados por esses princípios, em meados dos anos 1990 começaram a surgir *frameworks* que combinavam planejamento estruturado com ciclos de entrega ágeis. Em 2001, doze especialistas formalizaram o Manifesto Ágil (2001), definindo quatro valores fundamentais:

- **Indivíduos e interações** acima de processos e ferramentas;
- ***Software* em funcionamento** acima de documentação extensa;
- **Colaboração com o cliente** acima de negociação contratual;
- **Resposta a mudanças** acima de seguir um plano;

Além desses valores, foram estabelecidos doze princípios que enfatizam entregas frequentes, satisfação do cliente e melhoria contínua. Como comentado por Matharu et al. (2015), a partir dos fundamentos ágeis, surgiram diversos métodos utilizados no desenvolvimento de *software*. Os mais difundidos e empiricamente respaldados são o *Scrum*, *Kanban* e o *Extreme Programming* (XP). Por exemplo, o *Scrum* organiza o trabalho em *sprints*[4] de duas a quatro semanas, com papéis bem definidos (*Product Owner*, *Scrum Master* e equipe) e cerimônias regulares (planejamento, acompanhamento diário, revisão e retrospectiva), o que proporciona maior visibilidade e previsibilidade do projeto. O *Kanban*, por sua vez, adota um quadro visual com colunas *To Do*, *Doing* e *Done* e restrição ao *work in progress* (WIP), favorecendo um fluxo contínuo e a identificação rápida de gargalos. Já o XP enfatiza boas práticas de engenharia, como testes automatizados, integração contínua e programação em par, promovendo feedback imediato e alta qualidade técnica.

Outros métodos, como *Feature Driven Development* (FDD), *Lean*, *Scaled Agile Framework* (SAFe), além de abordagens como *Crystal*, DSDM e ASD, também fazem parte do universo ágil, embora sejam menos comuns no dia a dia das equipes. Esses métodos têm perfis específicos e podem complementar ou ser aplicados conforme o contexto do projeto, mas com menor prevalência em estudos de adoção empírica.

[4] Ciclo de trabalho curto e fixo, usado em metodologias ágeis, no qual a equipe desenvolve e entrega um conjunto específico de tarefas ou funcionalidades.

A escolha do método mais adequado deve considerar a maturidade da equipe, a estabilidade dos requisitos, a complexidade do projeto e as demandas por flexibilidade ou previsibilidade. A Tabela 1 sintetiza os critérios essenciais para comparar *Scrum*, *Kanban* e XP, três das metodologias mais difundidas.

Tabela 1 – Comparações entre Metodologias Ágeis

Critério	**Scrum**	**Kanban**	**XP**
Fluxo de trabalho	Iterações fixas	Contínuo	Iterações curtas
Mudanças em curso	Limitadas por sprint	A qualquer momento	Contínuas
Papéis	Definidos	Flexíveis	Mínimos
Foco	Gestão de ciclos	Otimização de fluxo	Qualidade de código

Fonte: Autoria própria (2025).

3.2 Tecnologias da Versão Anterior

Para planejar a migração do *site* do Observatório de *WordPress* para *Flutter*, foi fundamental realizar um estudo minucioso da infraestrutura tecnológica existente. Esse levantamento teve o objetivo de compreender a arquitetura, os fluxos de dados, os pontos fortes e as limitações do sistema atual, a fim de que fosse possível preservar as funcionalidades essenciais e otimizar a experiência no novo ambiente. A seguir, são detalhadas duas das principais tecnologias que sustentam o *site* na versão anterior.

3.2.1 WordPress

O *WordPress*[5] foi a plataforma-base utilizada na construção do *site* original. Trata-se de um *Content Management System* (CMS)[6] de código aberto amplamente utilizado em diversos contextos por oferecer um ambiente flexível e extensível por meio de *plugins*[7] e temas. No entanto, ao longo do uso no contexto do Observatório, foram identificadas algumas limitações importantes, especialmente relacionadas à complexidade de sua interface e à dificuldade de manutenção e personalização.

Durante a análise, foram considerados os seguintes aspectos:

[5] Disponível em: <https://wordpress.org>

[6] Aplicação de *software* que facilita a criação, edição e gerenciamento de conteúdo digital em *websites*, sem a necessidade de conhecimentos técnicos avançados.

[7] Extensões de *software* que adicionam ou ampliam funcionalidades de um sistema ou aplicativo sem alterar seu núcleo.

- **Ecossistema de *plugins***: o *WordPress* possui uma vasta biblioteca de *plugins*, que permite adicionar funcionalidades específicas sem a necessidade de desenvolvimento do zero. No entanto, a sobreposição de funcionalidades, diferenças de padrão entre *plugins* e possíveis incompatibilidades entre versões contribuíram para tornar a gestão do sistema mais complexa e propensa a erros.

- **Flexibilidade de temas**: a plataforma permite personalizações visuais por meio de temas, facilitando ajustes no *layout*. Entretanto, alterações mais profundas exigem conhecimento técnico em PHP, HTML, CSS e JS, dificultando a autonomia de usuários não especializados.

- **Modelo de dados**: o uso de tipos de publicações personalizadas, taxonomias e campos avançados (muitas vezes gerenciados por *plugins*) oferece possibilidades de organização ricas e adaptáveis. Ainda assim, a configuração desses elementos demanda familiaridade com conceitos específicos do *WordPress* e pode se tornar pouco intuitiva para novos usuários.

- **Interface administrativa sobrecarregada**: à medida que novos *plugins* são instalados, o painel administrativo torna-se progressivamente mais poluído e difícil de navegar, o que dificulta a gestão do conteúdo e a localização de funcionalidades específicas.

- **Arquitetura cliente-servidor tradicional**: o uso de PHP e *MySQL* garante estabilidade e compatibilidade com a maioria dos serviços de hospedagem, mas impõe limitações de desempenho e escalabilidade, sobretudo em sistemas com alta demanda de acesso ou de manipulação de dados.

Embora o *WordPress* forneça recursos robustos e versáteis, os fatores acima evidenciaram obstáculos significativos no contexto do Observatório, principalmente no que diz respeito à usabilidade e à curva de aprendizado da equipe editorial. A experiência demonstrou que a flexibilidade do sistema, embora vantajosa em alguns cenários, acabou exigindo conhecimentos técnicos específicos e rotinas de manutenção que dificultavam a autonomia e a escalabilidade do projeto.

3.2.2 Tainacan

O *Tainacan*[8] é um *plugin* desenvolvido para o *WordPress* com foco na gestão de acervos digitais e organização de coleções. No *site* do Observatório, foi utilizado para estruturar e exibir conteúdos classificados por temas, autores, datas e outras categorias, o que permitiu a organização semântica dos dados. Dentre os principais recursos do *Tainacan*, destacam-se:

- **Metadados customizáveis**: permite a criação e aplicação de campos personalizados e vocabulários controlados, essenciais para a catalogação acadêmica e preservação da coerência dos registros.
- **Interface de busca e filtragem**: fornece automaticamente filtros facetados que facilitam a navegação e a recuperação de itens por múltiplos critérios.
- **Controle de acesso e visibilidade**: possibilita configurar diferentes níveis de permissão para usuários editores e visitantes, promovendo uma gestão mais segura do conteúdo.
- **Exportação de dados**: oferece suporte à exportação em diversos formatos, como CSV e JSON, assegurando a interoperabilidade com outras plataformas e a preservação dos dados.

Não obstante os avanços proporcionados pelo *Tainacan* em termos de organização e padronização dos conteúdos, observou-se que sua performance tende a ser comprometida em coleções com um grande volume de itens. Além disso, por estar fortemente integrado ao ecossistema *WordPress*, herda os mesmos desafios relacionados à manutenção técnica, à usabilidade da interface administrativa e à dependência de uma arquitetura tradicional.

O estudo das tecnologias utilizadas na versão anterior do *site* foi fundamental para o planejamento da nova proposta. A análise permitiu identificar quais funcionalidades precisavam ser preservadas, quais poderiam ser melhoradas e quais demandavam uma nova abordagem.

A decisão pela migração para *Flutter* fundamentou-se principalmente na busca por:

1. Reduzir a complexidade da interface administrativa, oferecendo uma experiência mais acessível a usuários não técnicos;

[8] Disponível em: <https://tainacan.org/>

2. Modernizar a arquitetura do sistema, adotando uma base mais leve, performática e escalável;

3. Recriar os fluxos de catalogação, exibição e filtragem de conteúdos com maior controle sobre o design e a experiência do usuário.

Nos próximos capítulos, serão apresentadas as tecnologias da nova versão do Observatório, bem como as estratégias adotadas para manter e aprimorar as funcionalidades já existentes, agora sob uma arquitetura moderna baseada em *Flutter* e *Firebase*.

3.3 Tecnologias do Front-end da Nova Versão

3.3.1 Flutter: Um Framework Multiplataforma Emergente

O *Flutter* é um framework de código aberto desenvolvido pelo *Google* e lançado oficialmente em 2018, com o objetivo de unificar o desenvolvimento de interfaces para diferentes dispositivos a partir de uma única base de código na linguagem de programação *Dart*. Sua história começa em 2015, quando foi concebido internamente sob o nome de "*Sky*". Desde então, o projeto buscou oferecer desempenho gráfico excepcional, chegando a suportar até 120 quadros por segundo (FPS), isto é, exibir 120 imagens diferentes a cada segundo, o que gera uma sensação de movimento extremamente suave, muito além dos 30 FPS comuns em vídeos padrão, conforme explicado em Medium (2025).

Com o avanço do projeto, o *Flutter* passou a suportar dispositivos móveis (*Android* e *iOS*), navegadores modernos (*web*), desktops (*Windows*, *macOS* e *Linux*) e até mesmo sistemas embarcados, ampliando seu alcance para múltiplos cenários de uso.

Entre os principais recursos oferecidos pelo *Flutter*, destacam-se:

- **Código único para várias plataformas**: possibilita o desenvolvimento de aplicações para diferentes sistemas operacionais a partir de uma única base de código, reduzindo custos e tempo de desenvolvimento.

- ***Hot Reload***: permite que alterações no código sejam visualizadas instantaneamente na aplicação em execução, sem a necessidade de reiniciá-la, facilitando o processo de desenvolvimento e depuração.

- ***Widgets* componíveis**: toda parte da interface é tratada como um bloco reutilizável, o que torna a construção e a manutenção do *layout* mais intuitivas e modulares.
- **Desempenho próximo ao nativo**: não depende dos componentes nativos de cada sistema, mas desenha cada pixel por si só usando a *engine* gráfica *Skia*. Isso significa que toda a interface, incluindo botões, textos e animações, é "pintada" diretamente na tela, garantindo transições e movimentos muito suaves, mesmo em dispositivos com hardware mais modesto.
- **Ecossistema e comunidade ativos**: conta com uma vasta coleção de pacotes prontos (*plugins*) para funcionalidades comuns e suporte contínuo do *Google* e da comunidade de desenvolvedores.

Essas características fazem do *framework* uma opção especialmente adequada para reconstruir o *front-end* do Observatório. Ao concentrar todo o código em um único projeto, simplifica-se significativamente tanto a manutenção quanto a evolução da aplicação. A renderização direta via *Skia*, combinada com a capacidade de exibir altas taxas de quadros por segundo, garante uma interface fluida e consistente em diferentes dispositivos. O *Hot Reload* acelera o processo de prototipagem ao permitir que alterações no código sejam imediatamente refletidas na aplicação em execução, sem reinicializações. Finalmente, o uso de *widgets* componíveis oferece flexibilidade no *layout* e mantém a consistência visual, aspectos essenciais para assegurar uma navegação agradável em páginas ricas em conteúdo.

Possibilidade de PWA: embora ainda não implementada na versão atual, o *Flutter* permite compilar o mesmo código como uma *Progressive Web App* (PWA), oferecendo ao usuário a opção de "instalar" o Observatório diretamente pelo navegador, operar em modo *offline* em conteúdos previamente carregados e receber notificações. Essa funcionalidade representa um caminho futuro para ampliar o alcance da aplicação, sem perder a experiência de uma aplicação nativa.

3.3.2 Arquitetura em Camadas

No contexto do desenvolvimento de *software*, a utilização de uma arquitetura em camadas é uma prática consolidada, pois organiza o sistema de forma a promover clareza, manutenibilidade e testabilidade. Segundo Richards (2015), "o padrão de arquitetura em camadas

(ou *n-tier*) organiza o sistema em camadas horizontais que separam apresentação, negócio e persistência, promovendo clareza e manutenibilidade". Essa abordagem surgiu como resposta ao aumento da complexidade dos sistemas e à necessidade de estruturar melhor o código, permitindo que mudanças em uma parte não afetem diretamente as demais.

Dentre os modelos e padrões que orientaram a construção do projeto destacam-se:

1. ***Clean Architecture***, padrão arquitetural proposto por Martin (2017) (conhecido como "*Uncle Bob*"), que defende a divisão do código em círculos concêntricos, onde as regras de negócio ficam no centro, totalmente independentes de *frameworks*, bancos de dados ou interfaces externas. Essa técnica segue a chamada "*Dependency Rule*", segundo a qual o código só deve depender de camadas internas, tornando o sistema flexível, testável e resistente a mudanças.

2. ***Model-View-ViewModel* (MVVM)**, modelo bastante aplicado em *frameworks* como *Flutter*, em que os *Widgets* representam a *View*, os *Stores* funcionam como *ViewModel* (mantendo e reagindo ao estado da aplicação), e os *Models/Repositories* representam a camada de dados. Segundo Mohamed e Shaffi (2016), o padrão MVVM permite separar completamente a lógica de apresentação dos dados, facilita a manutenção e os testes unitários, além de garantir que alterações no *ViewModel* sejam refletidas automaticamente na *View*, promovendo uma arquitetura mais simples, modular e reativa.

No projeto Observatório, foi adotada uma combinação entre os princípios da *Clean Architecture* e o padrão MVVM. Enquanto a *Clean Architecture* fornece a base de organização e independência entre as camadas, o MVVM orienta a interação entre interface, gerenciamento de estado e dados, resultando em uma estrutura modular, expansível e alinhada às boas práticas de desenvolvimento em *Flutter*.

3.3.3 Boas Práticas e Clean Code

Boas práticas no desenvolvimento de *software* são um conjunto de diretrizes e métodos que visam produzir um código mais claro, eficiente e fácil de manter. O conceito de *Clean Code* (Código Limpo) foi popularizado por Robert C. Martin (2008), que defende que um código bem escrito deve ser legível e compreensível para outros desenvolvedores, facilitando sua manutenção e evolução.

No desenvolvimento de *software*, a legibilidade e organização do código são fundamentais para garantir que a aplicação seja sustentável a longo prazo. Entre os princípios mais

importantes destacam-se:

- **Nomes significativos**: escolher nomes claros e descritivos para variáveis, funções e classes para que seu propósito seja imediatamente compreendido.
- **Funções pequenas e específicas**: cada função deve realizar uma única tarefa, facilitando a compreensão, testes e manutenção.
- **Organização e padronização do código**: utilizar uma estrutura consistente, com indentação correta e separação lógica em módulos, para facilitar a navegação e colaboração entre a equipe.
- **Evitar duplicação**: aplicar o princípio *DRY (Don't Repeat Yourself)*, evitando repetir códigos similares, o que reduz erros e facilita futuras modificações.
- **Tratamento adequado de erros**: implementar mecanismos claros para lidar com falhas, garantindo que o sistema responda de forma segura e previsível.
- **Comentários relevantes**: comentar o código para explicar as razões por trás de decisões complexas, evitando comentários óbvios ou redundantes.

A adoção dessas práticas traz benefícios importantes, como a facilitação do trabalho em equipe, redução de erros, facilidade para testes e uma base de código que pode crescer e se adaptar às novas demandas com maior segurança. Além disso, facilita a integração de novos desenvolvedores, tornando o projeto mais acessível e colaborativo.

Em resumo, a aplicação consistente de boas práticas e os princípios de *Clean Code* são essenciais para garantir a qualidade, eficiência e sustentabilidade do desenvolvimento de *software*, especialmente em projetos que envolvem migração e reestruturação, como a transformação do *site Wordpress* para *Flutter* apresentada neste trabalho.

3.4 Banco de Dados

3.4.1 Modelagem NoSQL

A evolução dos sistemas de informação trouxe consigo novos paradigmas de armazenamento de dados, complementando e, em muitos casos, substituindo o tradicional modelo relacional. De acordo com Sadalage e Fowler (2012), os bancos de dados NoSQL (*Not Only* SQL)

emergiram como resposta às limitações dos sistemas relacionais tradicionais, especialmente em cenários que demandam alta escalabilidade, flexibilidade de esquema e processamento de grandes volumes de dados não estruturados.

Diferentemente dos bancos relacionais, que organizam dados em tabelas com relacionamentos bem definidos através de chaves estrangeiras, os sistemas NoSQL adotam estruturas mais flexíveis. Esta flexibilidade permite que aplicações modernas, como o Observatório, adaptem suas estruturas de dados conforme a evolução dos requisitos, sem a necessidade de migrações complexas de esquema.

A modelagem NoSQL baseia-se em quatro principais categorias: documentos, chave-valor, colunas e grafos. O *Cloud Firestore*[9], escolhido para este projeto, utiliza o modelo de documentos, onde os dados são organizados em coleções que contêm documentos individuais. Cada documento pode conter campos de diferentes tipos, incluindo arrays, objetos aninhados e referências a outros documentos.

Esta abordagem oferece vantagens significativas para aplicações *web* modernas, especialmente aquelas que necessitam de sincronização em tempo real e escalabilidade horizontal. A estrutura de dados do projeto foi projetada considerando estas características, organizando o conteúdo em coleções lógicas que refletem a estrutura conceitual da aplicação.

3.4.2 Cloud Firestore

O *Cloud Firestore* é uma solução de banco de dados NoSQL orientado a documentos, desenvolvida pelo *Google* como parte de sua plataforma *Firebase*. Lançado em 2017, o *Firestore* surgiu como uma evolução do *Firebase Realtime Database*[10], com o objetivo de superar suas limitações e oferecer melhor desempenho, escalabilidade e recursos de consulta.

A estrutura do *Firestore* é baseada em três elementos principais: documentos, coleções e subcoleções. Documentos são registros flexíveis que armazenam dados no formato *JSON-like*, permitindo diferentes tipos de campos e objetos aninhados. Esses documentos são organizados em coleções, e cada documento pode conter subcoleções, o que permite criar estruturas hierárquicas e bem organizadas. Essa abordagem favorece a modelagem de dados mais próxima da lógica do domínio da aplicação.

[9] Disponível em: <https://firebase.google.com/docs/firestore>
[10] Disponível em: <https://firebase.google.com/docs/database>

Além disso, a ferramenta oferece três características fundamentais que a tornam especialmente adequada para aplicações modernas: flexibilidade, consultas expressivas e projeto voltado para escala. Sua flexibilidade permite armazenar estruturas de dados dinâmicas e evoluir o esquema sem necessidade de migrações complexas. As consultas expressivas possibilitam recuperar documentos com múltiplos filtros, ordenações e indexações automáticas, otimizando o desempenho mesmo em grandes volumes de dados. Por fim, o serviço é projetado para escalar globalmente, contando com replicação multirregional, transações atômicas e alta disponibilidade por meio da infraestrutura do *Google Cloud*[11].

Essas características são particularmente valiosas em projetos acadêmicos como o Observatório, que podem evoluir de forma incremental e exigir adaptações constantes em sua estrutura de dados.

3.4.3 Regras de Segurança

A segurança em aplicações *web* modernas é fundamental, especialmente no tratamento de dados sensíveis e na execução de operações administrativas. O *Firestore* apresenta um sistema robusto de regras de segurança baseado em uma linguagem declarativa, que possibilita controle granular sobre quem pode ler, escrever e modificar os dados armazenados no banco.

As regras de segurança do sistema são definidas em um arquivo específico de configuração, conhecido como "arquivo de regras", onde são declaradas as permissões de acesso para cada nível da estrutura de dados: projeto, coleções e documentos. Cada operação de leitura ou escrita é avaliada contra essas regras antes de ser executada, garantindo que apenas usuários autorizados tenham acesso aos dados adequados. O sistema integra autenticação via *Firebase Authentication*[12], permitindo que as regras considerem o estado de autenticação e as informações presentes no *token*[13] do usuário.

Para o Observatório de Ensino de História e Geografia, as regras de segurança foram configuradas para implementar um modelo de controle de acesso baseado em *roles*[14]. Usuários não autenticados têm acesso restrito a operações de leitura, principalmente aos *posts* publicados,

[11] Disponível em: <https://cloud.google.com/>
[12] Disponível em: <https://firebase.google.com/docs/auth>
[13] Código digital gerado para identificar e autenticar um usuário, permitindo acesso seguro a um sistema ou serviço.
[14] Conjunto de permissões e responsabilidades atribuídas a usuários ou grupos, conforme o modelo RBAC (*Role-Based Access Control*).

enquanto usuários autenticados com *roles* específicos (`ADMIN`, `EDITOR`) possuem permissões diferenciadas para criação, edição e exclusão de conteúdos.

A implementação das regras considera a estrutura hierárquica dos dados, aplicando níveis distintos de acesso para diferentes coleções e documentos. Por exemplo, a coleção `users` possui regras mais restritivas, permitindo acesso somente a administradores, enquanto outras coleções permitem leitura pública, restringindo operações de escrita a usuários autenticados com as permissões adequadas, o que e exemplificado na Figura 1.

Além do controle de acesso, o sistema de segurança implementa validação de dados para garantir que apenas informações válidas sejam armazenadas. Essas regras verificam a estrutura dos documentos, tipos e formatos dos dados, e valores permitidos, prevenindo a inserção de informações malformadas que poderiam comprometer a integridade do banco.

A configuração de segurança do projeto também inclui proteção contra ataques comuns, como injeção de dados maliciosos e acessos não autorizados. As regras são testadas regularmente para assegurar a ausência de vulnerabilidades que possam comprometer a segurança da aplicação ou a privacidade dos usuários.

Figura 1 – *Firebase Firestore* - Regras de Segurança

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    match /posts {
      allow read: if true;
      allow write: if request.auth != null && (isAdmin(request.auth.uid) || isEditor(request.auth.uid));
    }

    match /users {
      allow read, write: if request.auth != null && isAdmin(request.auth.uid);
    }

    function isAdmin(userId) {
      return exists(/databases/$(database)/documents/users/$(userId)) &&
             get(/databases/$(database)/documents/users/$(userId)).data.role == "ADMIN" &&
             get(/databases/$(database)/documents/users/$(userId)).data.isDeleted == false;
    }

    function isEditor(userId) {
      return exists(/databases/$(database)/documents/users/$(userId)) &&
             get(/databases/$(database)/documents/users/$(userId)).data.role == "EDITOR" &&
             get(/databases/$(database)/documents/users/$(userId)).data.isDeleted == false;
    }
}
```

Fonte: Autoria própria (2025).

3.5 Armazenamento em Nuvem

Com o avanço das tecnologias, tornou-se comum utilizar soluções de armazenamento em nuvem para lidar com arquivos como imagens, vídeos e documentos. Diferente de armazenar diretamente no servidor do *site*, esse modelo permite que os arquivos fiquem guardados em uma infraestrutura escalável, acessível pela *internet* e com gerenciamento mais eficiente de segurança e permissões.

3.5.1 Firebase Storage

Neste projeto, optou-se por utilizar o *Cloud Storage*[15], um serviço oferecido pelo *Google* que permite armazenar arquivos na nuvem de forma segura e com integração direta ao restante da plataforma *Firebase*. Ele é altamente escalável, ou seja, permite crescimento conforme a demanda de uso, e oferece ferramentas nativas para *upload*, *download* e controle de acesso aos arquivos.

Todas as mídias da aplicação, como imagens de postagens e documentos PDF, foram armazenadas utilizando esse serviço. Os arquivos são organizados em diretórios virtuais e acessados por meio de URLs geradas automaticamente, o que facilita sua integração com a interface *Flutter Web*.

3.5.2 Regras de Segurança

O controle de acesso aos arquivos do projeto é realizado por meio das regras do *Cloud Storage* do *Firebase*, cuja definição segue a mesma lógica declarativa utilizada no *Firestore*, já descrita em subseção anterior. Essas regras avaliam, a cada solicitação, se o usuário está autenticado e se possui o papel adequado (`ADMIN`, `EDITOR`), além de verificar se sua conta está ativa no banco de dados.

No caso do *Storage*, as permissões são aplicadas a caminhos ou pastas específicas, definidos por blocos `match`. Dentro desses blocos, comandos `allow` especificam quem pode ler (`read`) ou escrever (`write`) arquivos. Por exemplo, no caso do Observatorio, o sistema permite leitura pública dos arquivos disponíveis, mas restringe operações de escrita a usuários autenticados e autorizados, conforme exemplificado na Figura 2. Essa configuração garante que apenas

[15] Disponível em: <https://firebase.google.com/docs/storage>

usuários autorizados realizem *uploads*, enquanto mantém os arquivos públicos acessíveis para visualização.

Figura 2 – *Firebase Storage* - Regras de Segurança

```
rules_version = '2';

service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read: if true;
      allow write: if request.auth != null && (isAdmin(request.auth.uid) || isEditor(request.auth.uid));
    }

    function isAdmin(userId) {
      return firestore.get(/databases/(default)/documents/users/$(userId)).data.role == "ADMIN" &&
             firestore.get(/databases/(default)/documents/users/$(userId)).data.isDeleted == false;
    }

    function isEditor(userId) {
      return firestore.get(/databases/(default)/documents/users/$(userId)).data.role == "EDITOR" &&
             firestore.get(/databases/(default)/documents/users/$(userId)).data.isDeleted == false;
    }
  }
}
```

Fonte: Autoria própria (2025).

3.6 Controle de Versão e Deploy

O *GitHub* foi utilizado como plataforma principal para o versionamento do código-fonte do projeto. Através do sistema de controle de versões *Git*, foi possível manter um histórico completo das alterações, possibilitando rastrear contribuições, reverter mudanças indesejadas e facilitar a colaboração entre desenvolvedores.

Além do versionamento, o projeto também utilizou funcionalidades avançadas oferecidas pela plataforma, como a automação de fluxos de trabalho por meio do *GitHub Actions*, integração com servidores de hospedagem e gerenciamento seguro de variáveis sensíveis.

3.6.1 GitHub Actions

Para automatizar o processo de *build*[16] e *deploy*[17] da versão *web* da aplicação, utilizou-se o recurso *GitHub Actions*. Essa ferramenta permite definir fluxos de trabalho (*workflows*) que são executados automaticamente em resposta a eventos no repositório, como o envio de novas alterações para a *branch* principal.

[16] Processo de compilar e gerar a versão executável de um software a partir do seu código-fonte.
[17] Processo de disponibilizar uma aplicação ou atualização para uso em um ambiente de produção.

No caso deste projeto, o fluxo foi definido no arquivo `deploy.yml`, localizado na raiz do repositório. Esse arquivo descreve as etapas necessárias para compilar e publicar a aplicação de forma automatizada:

1. **Clonagem do código-fonte**: nesta etapa, o *GitHub Actions* faz o *checkout* do repositório. Isso significa que ele acessa os arquivos do projeto armazenados no *GitHub* e os copia para o ambiente temporário da automação. Assim, o sistema pode executar as próximas etapas com base na versão mais atualizada do código, geralmente a branch principal (`main`).

2. **Configuração do *Flutter***: o ambiente digital da automação não possui o *Flutter* instalado por padrão. Por isso, é necessário configurar o SDK[18] *Flutter*, especificando a versão exata utilizada no projeto (no caso, `3.32.4`). Isso garante compatibilidade e evita falhas durante o processo de compilação.

3. **Limpeza e dependências**: o comando `flutter clean` remove arquivos de cache ou construções anteriores, assegurando que a compilação ocorra com base em um estado limpo. Em seguida, o comando `flutter pub get` instala todas as bibliotecas e pacotes utilizados no projeto, definidos no arquivo `pubspec.yaml`.

4. **Compilação**: após preparar o ambiente, a aplicação é compilada com o comando `flutter build web -release`. Essa ação transforma o código-fonte em arquivos otimizados para execução no navegador (HTML, CSS e JS), reduzindo o tamanho final e melhorando a performance em produção.

5. **Deploy via FTP**: por fim, os arquivos gerados na etapa anterior são enviados automaticamente ao servidor de hospedagem da *HostGator*. Para isso, é utilizada a ferramenta de Kirkland (2024), que realiza o envio via protocolo *File Transfer Protocol* (FTP), usando as credenciais armazenadas com segurança nos *secrets* do *GitHub*.

Esse processo automatizado torna a publicação da aplicação mais ágil, confiável e segura. Desse modo, elimina-se etapas manuais suscetíveis a erro, reduz-se o tempo gasto em operações repetitivas e garante-se que o ambiente de produção esteja sempre atualizado com a versão mais recente e validada do projeto.

[18] *Software Development Kit* (SDK) é um conjunto de ferramentas e recursos que facilita o desenvolvimento de aplicações para uma plataforma específica.

3.6.2 Secrets e Variáveis

Durante o processo de *deploy*, é necessário o uso de credenciais FTP para acesso ao servidor da *HostGator*. Para garantir a segurança dessas informações sensíveis, foi utilizado o recurso *GitHub Secrets*, que permite armazenar variáveis de ambiente de forma criptografada.

No projeto, foram utilizadas as seguintes variáveis:

- `FTP_HOST`: endereço do servidor FTP.
- `FTP_USER`: nome de usuário de autenticação.
- `FTP_PASSWORD`: senha de acesso.

Essas variáveis são referenciadas nos *scripts*[19] de *deploy* de maneira segura, sem ficarem visíveis no código-fonte. Isso reforça as boas práticas de segurança no ciclo de integração e entrega contínua (CI/CD), protegendo o projeto contra exposições indesejadas de dados sensíveis.

3.7 Hospedagem e Infraestrutura

A publicação de um *site* na *internet* exige a combinação de dois elementos fundamentais: a hospedagem e o domínio. A hospedagem consiste em um serviço que armazena os arquivos do *site* (como imagens, textos, *scripts* e páginas) e os disponibiliza para acesso público. Já o domínio é o endereço digital que permite que os usuários encontrem o *site*, como por exemplo `observatoriogeohistoria.net.br`.

Além disso, há um componente essencial chamado DNS (*Domain Name System*), responsável por traduzir o nome do domínio em um endereço IP (*Internet Protocol*, conectando o usuário corretamente ao servidor que hospeda o *site*.

3.7.1 Locaweb

A *Locaweb*[20] foi mantida como empresa responsável pelo serviço de hospedagem. Essa escolha visou reaproveitar a infraestrutura da versão anterior do *site*, otimizando recursos já

[19] Pequenos programas ou conjuntos de instruções usados para automatizar tarefas ou controlar a execução de funções em um sistema ou aplicação.

[20] Disponível em: <https://www.locaweb.com.br/>

contratados e utilizados. Por meio do painel da *Locaweb*, é possível gerenciar contas de e-mail, configurações do servidor e os arquivos da aplicação *web*. A nova versão do projeto, construída com *Flutter Web*, foi implantada neste ambiente após a automação do *deploy*.

3.7.2 HostGator

O gerenciamento do domínio e das configurações de DNS foi realizado por meio da *HostGator*[21], onde o domínio principal do *site* está registrado. Esta separação entre a empresa que hospeda os arquivos (*Locaweb*) e aquela que gerencia o domínio (*HostGator*) permite maior controle e flexibilidade sobre os recursos da infraestrutura.

No painel de DNS da *HostGator*, foram configurados os apontamentos necessários para redirecionar o domínio para o servidor da *Locaweb*. Dessa forma, ao digitar o endereço do *site* no navegador, o sistema DNS garante que o usuário seja corretamente direcionado à versão desejada da aplicação.

[21] Disponível em: <https://www.hostgator.com.br/>

4 Desenvolvimento do Sistema

Este capítulo descreve detalhadamente o processo de desenvolvimento do sistema, abordando desde o planejamento inicial e a definição de requisitos até a publicação e lançamento da aplicação. São apresentados o modelo de desenvolvimento adotado, a arquitetura projetada, as tecnologias utilizadas no *front-end* e *back-end*, bem como os procedimentos de migração de dados e estratégias para garantir responsividade e desempenho. O objetivo é demonstrar, de forma clara e estruturada, como cada etapa contribuiu para a construção de uma solução funcional e alinhada às necessidades do projeto.

Considerando que o Observatório de Ensino de História e Geografia já existia, a principal tarefa consistiu na migração do *site* atual, desenvolvido em *WordPress*, para o *Flutter*. O foco foi simplificar a plataforma e melhorar a experiência do usuário, preservando a identidade visual.

A primeira etapa foi a análise do sistema atual. Apesar de funcional, o *WordPress* apresentava limitações de usabilidade e flexibilidade. Além disso, a administração de postagens e conteúdos exigia lidar com um painel repleto de funcionalidades não utilizadas pelo Observatório, o que tornava a gestão mais lenta e confusa. No aspecto técnico, a manutenção também se mostrava complexa, uma vez que a localização e edição de arquivos específicos, responsáveis por estilos, disposição de elementos e *layout* de determinadas páginas, demandava tempo e conhecimento aprofundado da estrutura interna do *WordPress*. Diante desse cenário, a adoção do *Flutter* visou oferecer uma plataforma mais enxuta, personalizada e de fácil manutenção, mantendo apenas os recursos necessários para o funcionamento eficiente do *site*.

Os requisitos levantados foram classificados em duas categorias principais: *funcionais* (relacionados diretamente às funcionalidades que o sistema deve oferecer) e *não funcionais* (relacionados a características de qualidade e restrições técnicas), conforme apresentado na Tabela 2.

Tabela 2 – Requisitos Funcionais e Não Funcionais

Tipo	**Descrição**
Requisitos Funcionais	
Exibir conteúdo estático	Páginas institucionais e informativas fixas.
Listar postagens	Exibição dinâmica com filtros e ordenação.
Disponibilizar biblioteca de documentos	Área para consulta e *download* com categorização.
Autenticar usuários	Acesso restrito ao painel com *login* seguro.
Criar painel administrativo simplificado	Gerenciar postagens, categorias, documentos, usuários e equipe.
CRUD	Criar, ler, atualizar e excluir registros.
Integrar front-end/back-end	Sincronizar dados do painel no *site*.
Responsividade	Ajuste automático a *desktop*, *tablet* e *smartphone*.
Reorganizar conteúdo	Separação clara entre áreas de História e Geografia.
Migrar *posts* gradualmente	Transferência de conteúdo conforme módulos concluídos, com testes contínuos.
Migrar documentos	Importação automatizada da biblioteca via *script*.
Requisitos Não Funcionais	
Preservação da identidade visual	Manter paleta original (branco, cinza e laranja).
Melhor desempenho	Carregamento rápido e navegação fluida.
Facilidade de manutenção	Estrutura simplificada para atualizações.
Migração sem impacto	Desenvolvimento/testes em subdomínio, preservando o *site* original.
Segurança	Proteger acessos e evitar alterações não autorizadas.
Escalabilidade	Permitir expansão sem reestruturação significativa.

Fonte: Autoria própria (2025).

4.1 Modelo Incremental Utilizado

Com base nas características do projeto e nas abordagens discutidas no capítulo anterior, foi possível identificar que a metodologia adotada ao longo do desenvolvimento se aproximou fortemente de um modelo incremental com práticas ágeis inspiradas no *Scrum* e no XP. Essa escolha ocorreu de forma natural, conforme a necessidade de dividir o projeto em entregas parciais, validar continuamente com o cliente e adaptar funcionalidades de acordo com o andamento da implementação.

O fluxo de trabalho adotado priorizou a construção gradual da estrutura inicial do sis-

tema, evoluindo progressivamente para a implementação das funcionalidades dinâmicas e integrações. Inicialmente, desenvolveu-se o *front-end* do *site* público voltado às telas de conteúdo majoritariamente estático, como por exemplo a página inicial, permitindo testar o layout, responsividade e identidade visual, além de disponibilizar as primeiras entregas validadas pelo cliente — neste caso, o doutorando da FACED, que assumiu o papel de "dono do produto".

Na etapa seguinte, o foco passou para o desenvolvimento das telas do *site* público que possuíam elementos dinâmicos, como listagens de postagens e a biblioteca de documentos. Essa fase preparou a estrutura visual e os componentes necessários para futura integração com o *back-end*.

O processo passou a incorporar ciclos iterativos que englobavam desde a implementação do *front-end*, passando pelos testes de responsividade e identidade visual, até a integração com o *back-end* e testes de funcionalidade completos, culminando na entrega e apresentação ao cliente. Esse ciclo se repetiu para cada módulo, assegurando entregas incrementais e funcionais.

Após consolidar a interface do *site* público, iniciou-se o desenvolvimento do *front-end* do painel administrativo, começando pela funcionalidade de autenticação. Em seguida, foi criada uma tela central de navegação, dando acesso às diferentes seções de administração, contemplando formulários para o gerenciamento (CRUD) de postagens, usuários, categorias, representantes da equipe e documentos da biblioteca.

O desenvolvimento de cada CRUD seguiu o ciclo incremental de implementação e validação: criação do formulário no painel administrativo, implementação do *back-end*, integração entre as partes e, por fim, conexão das telas do *site* público ao *back-end* para exibir os dados gerenciados pelo painel. Esse processo repetitivo permitiu entregas constantes e verificações contínuas da qualidade do sistema.

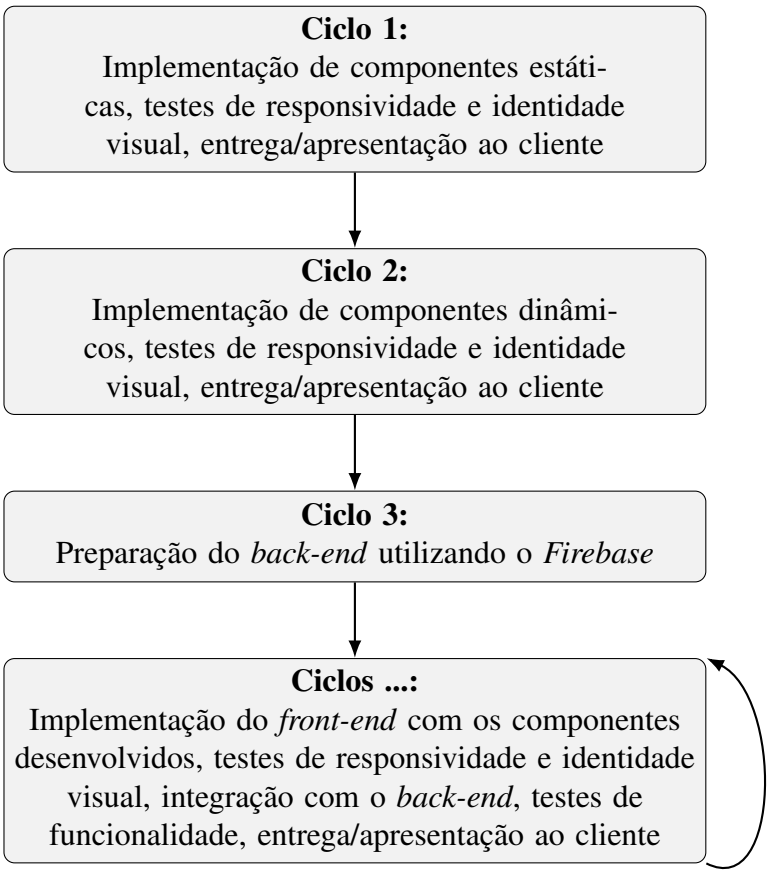
Cada uma dessas fases pode ser considerada uma etapa do modelo incremental, com ciclos curtos de desenvolvimento, entregas frequentes e validações constantes com o doutorando. As reuniões de demonstração funcionaram como revisões de *sprint*, enquanto a priorização das tarefas seguiu a lógica de um *backlog* dinâmico, ajustado conforme surgiam novas necessidades. O foco na qualidade do código, nos testes frequentes e na integração contínua das partes aproxima o processo de práticas típicas do XP.

A Figura 3 ilustra de forma genérica esse fluxo incremental e cíclico de desenvolvimento adotado no projeto, que contempla tanto ciclos mais simples, envolvendo apenas o *front-*

end (implementação, testes de responsividade e identidade visual, entrega), quanto ciclos mais completos, com integração e testes funcionais do *back-end*.

Assim, pode-se afirmar que o desenvolvimento seguiu um modelo incremental estruturado, inspirado nas práticas do *Scrum* e do XP, adaptado à realidade de um projeto de pequeno porte conduzido por uma única desenvolvedora, mas com alto nível de interação e *feedback* contínuo do cliente final.

Figura 3 – Representação genérica dos ciclos incrementais de desenvolvimento adotados no projeto



Fonte: Autoria própria (2025).

4.2 Arquitetura do Sistema

Seguindo a proposta de arquitetura em camadas apresentada na Seção 3, e itens seção 3.3.2, a estrutura do projeto, tanto na implementação do *site* público quanto do painel administrativo, foi organizada com base nos princípios da *Clean Architecture* e nos modelos propostos pelo MVVM.

A adoção da *Clean Architecture* garante a separação clara entre as diferentes responsabilidades do sistema, permitindo que mudanças em uma camada tenham impacto mínimo nas demais. Isso facilita a manutenção, aumenta a testabilidade do código e proporciona uma base flexível para futuras evoluções da aplicação.

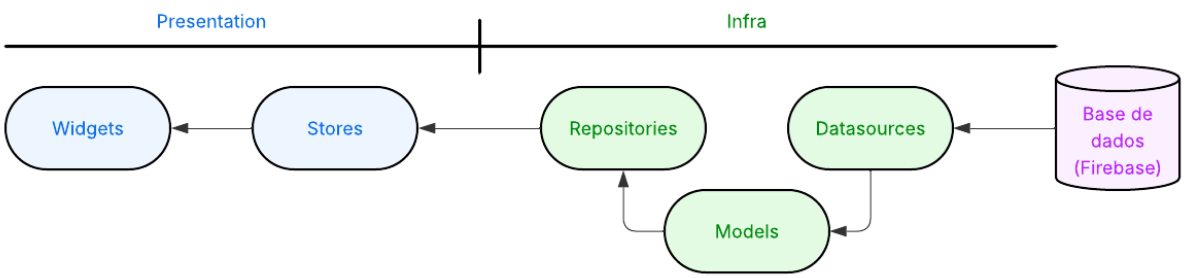
O padrão *MVVM* se mostrou adequado ao projeto por proporcionar um fluxo reativo de dados entre a interface do usuário e a lógica de negócio. Ele permite que os *widgets* se mantenham desacoplados do estado e das regras de negócio, garantindo que a interface reflita automaticamente alterações nos dados e que a lógica possa ser testada de forma independente.

Essa combinação de princípios e padrões foi especialmente útil considerando que o sistema envolve múltiplas funcionalidades e integrações externas, como o *Firebase*, e precisa oferecer uma experiência de usuário consistente e responsiva.

A Figura 4 apresenta o diagrama geral da arquitetura em camadas adotada. O fluxo de dados inicia-se nas fontes externas, como o banco de dados *Firebase*, que é acessado pela camada de infraestrutura. Esta camada realiza operações de consulta, armazenamento e transformação, repassando os dados para a camada de apresentação, onde os *widgets* e telas exibem a informação ao usuário, possibilitando interação e atualização reativa do estado da aplicação.

De forma inversa, as interações do usuário nas telas, como inserção, atualização ou exclusão de informações, são capturadas pela camada de apresentação e encaminhadas para a camada de infraestrutura. Esta, por sua vez, executa as operações de persistência necessárias no banco de dados, garantindo que as mudanças realizadas pelo usuário sejam refletidas de maneira consistente no sistema.

Figura 4 – Fluxo de dados desde a base de dados até as interfaces de usuário, seguindo a arquitetura adotada no projeto



Fonte: Autoria própria (2025).

A organização do código segue essa divisão, refletida na estrutura de diretórios do projeto. A Figura 5 mostra a disposição geral dos principais diretórios e arquivos da aplicação.

Figura 5 – Estrutura geral do projeto

```
lib/
|- main.dart                    # Ponto de entrada
|- firebase_options.dart        # Configurações do Firebase
\- app/
   |- app_setup.dart            # Configuração de injeção de dependência (GetIt)
   |- app_widget.dart           # Widget raiz da aplicação
   |- router/                   # Sistema de roteamento (GoRouter)
   |- theme/                    # Configurações de tema
   |- core/                     # Camada compartilhada
   \- features/                 # Funcionalidades específicas
```

Fonte: Autoria própria (2025).

O diretório `features/` organiza as funcionalidades específicas do sistema em módulos independentes. Cada *feature* agrupa seus elementos necessários, como interface, lógica de apresentação e acesso a dados, conforme ilustrado na Figura 6 e ainda, seguindo o modelo arquitetural apresentado na Figura 4.

O diretório `core/` representa a camada compartilhada da aplicação. Ele reúne componentes genéricos, utilitários e serviços que são utilizados transversalmente em todo o projeto, evitando duplicação e promovendo consistência. Exemplos incluem temas, *widgets* base, constantes, funções auxiliares e serviços de apoio, que podem ser acessados por qualquer *feature* sem criar dependências desnecessárias entre módulos.

Figura 6 – Estrutura típica de uma *feature*

```
lib/app/features/{feature}/
|- presentation/
|  |- pages/         # Telas da feature
|  |- components/    # Widgets reutilizáveis específicos
|  \- stores/        # Stores e estados reativos (MobX)
\- infra/
   |- datasources/   # Acesso direto Firebase
   |- repositories/  # Contratos e implementações de repositório
   |- models/        # Estruturas de dados da feature
   \- errors/        # Tratamento de erros e falhas
```

Fonte: Autoria própria (2025).

Camada de Apresentação Responsável pela interface do usuário, esta camada reúne as pági-

nas da aplicação, componentes visuais reutilizáveis e o gerenciamento reativo de estado por meio de *stores*. Para isso, utilizou-se a biblioteca *MobX*[1], que facilita a sincronização entre dados e interface do usuário (UI) através de *observables*, *actions* e *reactions*, promovendo um desenvolvimento mais reativo e menos verboso.

Camada de Infraestrutura Centraliza a comunicação com serviços externos, como o Firebase, por meio das seguintes responsabilidades:

- ***Datasources*:** implementam o acesso direto a APIs e bancos de dados.
- ***Repositories*:** atuam como intermediários, expondo métodos coerentes para as camadas superiores, realizando transformações de dados e tratamento de erros.
- ***Models*:** definem as estruturas de dados específicas de cada *feature*.
- ***Error Handling*:** encapsulam falhas e exceções em objetos de erro claros para controle adequado.

Injeção de Dependência Para desacoplamento e melhor organização do código, utiliza-se o padrão *Service Locator* via biblioteca *GetIt*[2], que permite registrar e recuperar instâncias de objetos em toda a aplicação, facilitando testes e manutenção.

Essa arquitetura modular e em camadas, ilustrada pelo diagrama da Figura 4, permite que cada parte do sistema seja isolada, facilmente testável e preparada para futuras evoluções, ao mesmo tempo que mantém um ponto central para funcionalidades comuns no diretório `core`.

4.3 Front-end do Sistema

A reconstrução do *front-end* do Observatório teve como prioridade aprimorar a experiência do usuário, preservando a identidade visual consolidada da plataforma. Para isso, foram adotadas soluções que resultaram em uma interface mais leve, moderna e intuitiva, capaz de oferecer uma navegação fluida sem abrir mão dos elementos visuais que caracterizam o projeto. A paleta de cores original — composta por tons de branco, cinza e laranja — foi mantida, garantindo a continuidade da marca e reforçando a identidade do Observatório perante seu público.

Além da preservação dos elementos visuais, a organização estrutural do conteúdo foi aprimorada. As áreas de História e Geografia passaram a ser apresentadas de forma mais clara

[1] Disponível em: <https://pub.dev/packages/mobx>
[2] Disponível em: <https://pub.dev/packages/get_it>

e segmentada, o que possibilitou uma navegação mais objetiva e eficiente. Essa reorganização, somada à simplificação dos menus e à padronização de elementos gráficos, contribuiu para reduzir a complexidade da interface, tornando o acesso às informações mais intuitivo.

A comparação entre as páginas “Sobre” das duas versões do *site* (Figuras 7 e 8) evidencia a modernização do *layout*. Enquanto a versão antiga apresentava um *design* mais carregado e pouco hierarquizado, a nova interface utiliza tipografia mais legível, espaçamento adequado e um visual mais limpo, favorecendo a leitura e a clareza das informações.

Figura 7 – Página "Sobre"do novo *site*, com *layout* limpo e hierarquia visual mais clara.

Fonte: Autoria própria (2025).

Figura 8 – Página "Sobre"do *site* antigo em *WordPress*, com interface mais carregada e menos organizada.

Fonte: Autoria própria (2025).

As melhorias também são visíveis na seção “Experiências Educativas” (Figuras 9 e 10). Antes, todos os conteúdos ficavam reunidos em uma única listagem, o que dificultava a navegação. Agora, a segmentação por área, História ou Geografia, e o mecanismo de busca

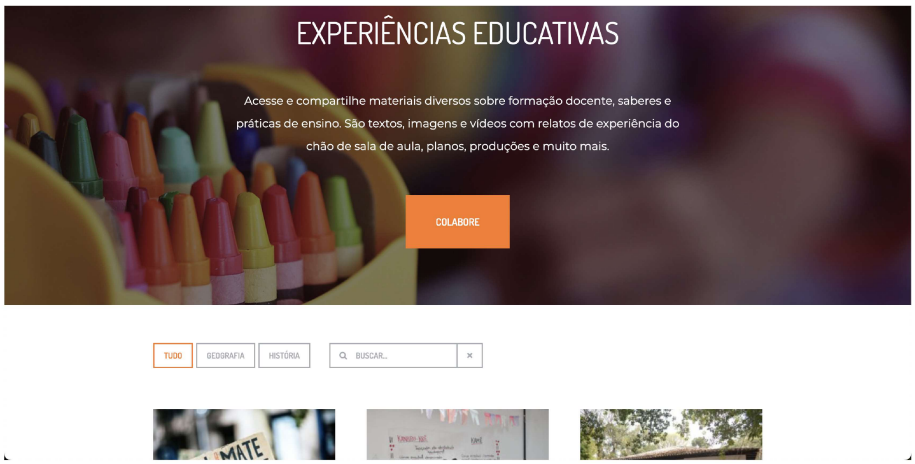
mais visível permitem uma interação mais rápida e objetiva, melhorando significativamente a experiência do usuário.

Outra inovação foi a introdução de um submenu de navegação mais organizado (Figura 11), que agrupa as categorias de conteúdo de forma clara e acessível, permitindo ao usuário chegar rapidamente aos materiais desejados. Além dos avanços visuais e estruturais, a implementação em *Flutter* proporcionou maior fluidez no carregamento de imagens, vídeos e documentos, garantindo uma navegação estável e responsiva em diferentes dispositivos. Essa base técnica contribuiu diretamente para uma experiência mais consistente e agradável ao usuário.

Figura 9 – Nova interface da seção “Experiências Educativas”, com segmentação dos conteúdos.

Fonte: Autoria própria (2025).

Figura 10 – Interface antiga da seção “Experiências Educativas”, sem segmentação dos conteúdos.



Fonte: Autoria própria (2025).

Figura 11 – Novo submenu de navegação, com categorias organizadas por área.

Fonte: Autoria própria (2025).

4.3.1 Responsividade

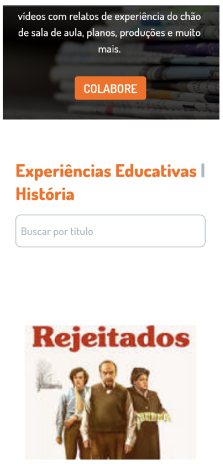
O acesso a *sites* por dispositivos móveis, como *smartphones* e *tablets*, é predominante. Segundo o Instituto Brasileiro de Geografia e Estatística BRASIL.(IBGE) (2025), em 2024, 98,8% dos acessos à internet no Brasil foram realizados por celulares. Nesse cenário, garantir que o *site* seja responsivo, adaptando-se a diferentes tamanhos e formatos de tela, é essencial para uma navegação eficiente e acessível.

No *site* anterior, desenvolvido em *WordPress*, a responsividade não era aplicada de forma consistente em todas as páginas. A manutenção dos estilos dependia de ajustes manuais em arquivos CSS, o que tornava o processo pouco ágil e propenso a falhas, dificultando a adaptação correta da interface a dispositivos móveis e comprometendo atualizações e correções.

Na nova versão, desenvolvida em *Flutter*, a responsividade foi planejada desde a concepção do projeto. Foram implementados mecanismos para calcular fatores de escala com base nas dimensões reais da tela, considerando largura, altura e orientação (retrato ou paisagem), aplicados dinamicamente ao tamanho de fontes e espaçamentos. Esse processo garante que os elementos visuais se ajustem proporcionalmente, preservando legibilidade, harmonia visual e usabilidade em diferentes dispositivos.

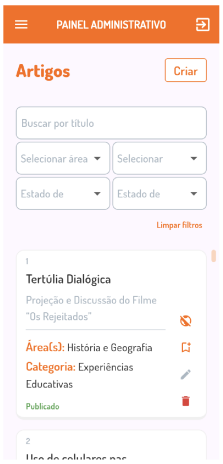
As Figuras 12 e 14 apresentam a seção “Experiências Educativas” em dispositivos móveis (*smartphone* e *tablet*, respectivamente), enquanto a Figura 16 mostra a mesma página em *desktop*. Nota-se que, nas versões móveis, os elementos são reorganizados para otimizar o espaço disponível, enquanto, nas telas maiores, há melhor aproveitamento da largura, exibindo mais informações sem comprometer a clareza ou a usabilidade.

Figura 12 – Visualização da seção “Experiências Educativas” em um *smartphone*, com interface otimizada para telas menores.



Fonte: Autoria própria (2025).

Figura 13 – Interface do painel administrativo em um *smartphone*, com menus e filtros adaptados para interação por toque.

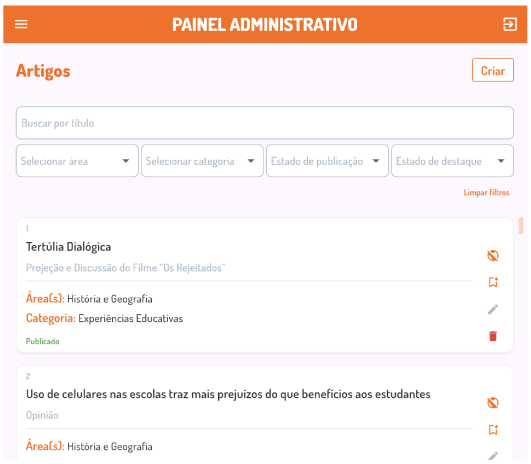


Fonte: Autoria própria (2025).

Figura 14 – Visualização da seção “Experiências Educativas” em um *tablet*, com aproveitamento otimizado do espaço da tela.

Fonte: Autoria própria (2025).

Figura 15 – Interface do painel administrativo em um *tablet*, com organização ampliada dos elementos para maior usabilidade.



Fonte: Autoria própria (2025).

O painel administrativo, mostrado nas Figuras 13, 15 e 17, também é totalmente responsivo. Sua interface reorganiza filtros, botões e *cards* de maneira dinâmica, garantindo a eficiência da gestão de conteúdo tanto em dispositivos móveis quanto em telas maiores.

Além disso, foram definidos limites mínimos e máximos para a escala dos elementos, evitando que textos ou espaçamentos se tornem excessivamente pequenos ou grandes. Essa estratégia favorece a consistência visual, simplifica a manutenção do *design* e garante uma experiência de navegação fluida e acessível para todos os usuários.

Figura 16 – Visualização da seção “Experiências Educativas” em *desktop*, com *layout* expandido para aproveitar melhor a largura da tela.

Fonte: Autoria própria (2025).

Figura 17 – Interface do painel administrativo em *desktop*, exibindo maior volume de informações simultaneamente.

Fonte: Autoria própria (2025).

4.4 Back-end do Sistema

O desenvolvimento do *back-end* do sistema fundamentou-se quase que integralmente na utilização dos serviços oferecidos pelo *Firebase*, que provê a infraestrutura necessária para armazenamento, autenticação e gestão dos dados, eliminando a necessidade de criação de uma API customizada. Para o armazenamento de dados não relacionais, foi utilizado o *Cloud Firestore*, que oferece uma estrutura flexível, escalável e em tempo real. A integração com serviços em nuvem, por meio do *Cloud Storage*, garantiu o armazenamento seguro de mídias e documentos. A autenticação e o controle de acessos foram implementados com o *Firebase Authentication*, assegurando que apenas usuários autorizados possam acessar ou manipular conteúdos específicos da plataforma.

A comunicação entre o aplicativo desenvolvido em *Flutter* e os serviços do *Firebase* ocorre diretamente por meio dos pacotes oficiais disponibilizados pelo *Flutter*, dispensando a necessidade de intermediários, como a definição de APIs REST[3], simplificando o desenvolvimento e manutenção do sistema.

4.4.1 Cloud Firestore

Para recapitular, o banco de dados *Cloud Firestore* adota uma estrutura não relacional. Diferentemente do modelo tradicional baseado em tabelas e linhas, os dados são organizados em coleções e documentos, que podem conter subcoleções e campos aninhados, de modo a atender às diversas funcionalidades da aplicação. No caso do Observatório, a organização dos

[3] Interfaces que permitem a comunicação entre sistemas via protocolo HTTP, seguindo princípios de arquitetura REST para operações simples e padronizadas.

dados segue uma arquitetura bem definida, composta por quatro coleções principais, cada uma correspondendo a uma funcionalidade central da plataforma:

- `library`**:** Comporta documentos provenientes do plugin *Tainacan* do *WordPress*, representando a biblioteca de documentos acadêmicos. Cada documento contém informações como autor, categoria, instituição, ano de publicação, tipo de documento (tese, dissertação, etc.) e URL do arquivo. A estrutura flexível do *Firestore* permite armazenar metadados específicos de cada tipo de documento, facilitando consultas e filtros avançados.

- `posts`**:** Armazena documentos representando categorias pré-definidas criadas via painel administrativo. Cada documento nesta coleção representa uma categoria de conteúdo (como "Eventos", "Experiências Educativas", "Narrativas", etc.), contendo metadados da categoria como título, descrição, imagem de fundo e áreas de conhecimento associadas. Um *post* em uma categoria pode pertencer a múltiplas áreas (como "História"e "Geografia"simultaneamente), permitindo classificação flexível do conteúdo. Dentro de cada categoria, existe uma subcoleção `category_posts` que armazena os *posts* específicos daquela categoria. Esta estrutura hierárquica permite organização lógica do conteúdo e facilita consultas por categoria e área.

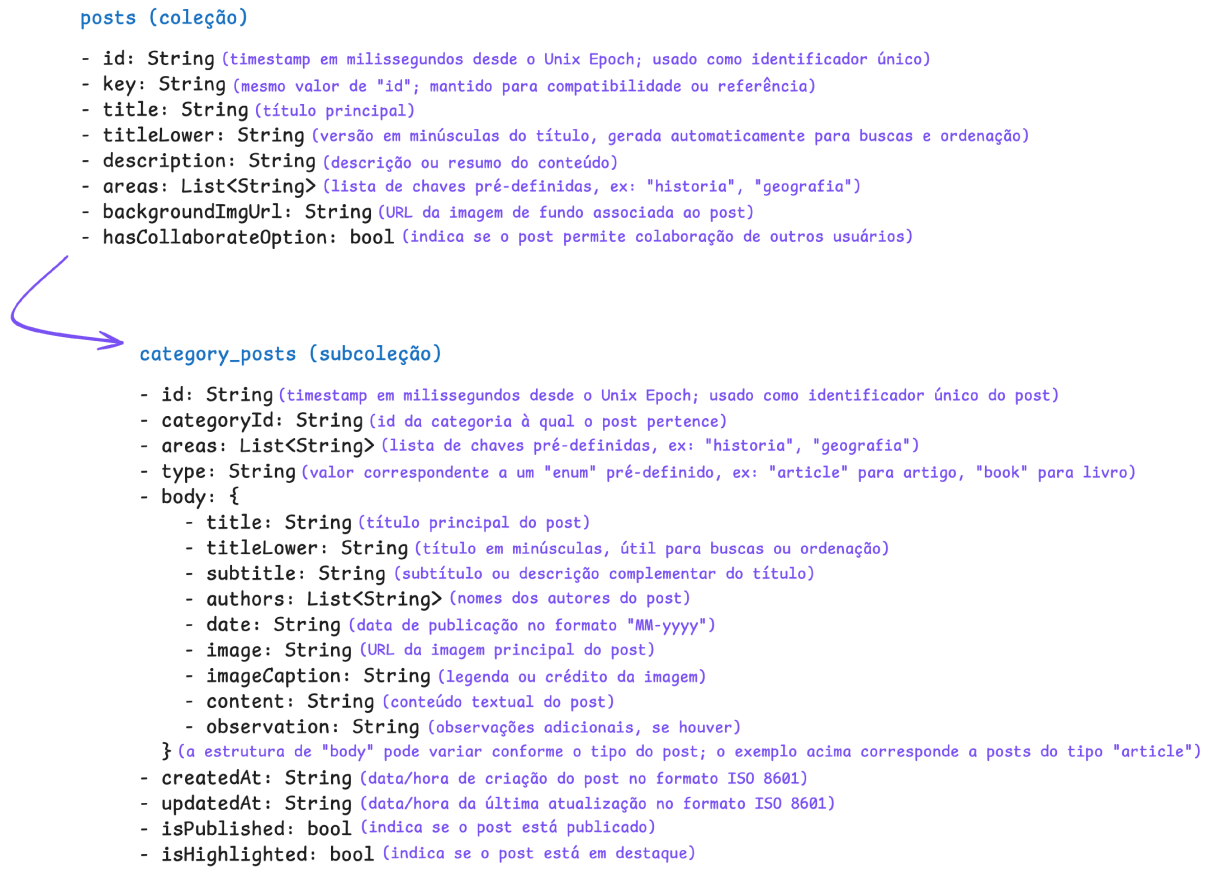
- `team`**:** Dedicada às informações sobre as pessoas da equipe da FACED/UFU que colaboram para o funcionamento do projeto de curadoria. Cada documento representa um membro da equipe, contendo dados como nome, função, instituição, foto e informações de contato. Esta estrutura facilita a exibição da equipe no *site* público e no painel administrativo.

- `users`**:** Gerencia usuários que utilizam o painel administrativo, permitindo operações de CRUD (*Create, Read, Update, Delete*) nos conteúdos. Cada documento representa um usuário autenticado, contendo informações como nome, e-mail, *role*/papel (`ADMIN`, `EDITOR`, `VIEWER`) e status de exclusão. Esta coleção é fundamental para o controle de acesso e permissões, permitindo que usuários autorizados publiquem *posts*, marquem conteúdo como destaque e realizem outras operações que impactam diretamente o *site* público.

A Figura 18 ilustra de forma simplificada a estrutura da coleção `posts`, oferecendo uma visão visual que facilita a compreensão do modelo; as demais coleções adotam um esquema

semelhante, com suas respectivas definições de atributos.

Figura 18 – Representação visual simplificada da estrutura da coleção `posts` no *Cloud Firestore*, mostrando categorias como documentos e suas subcoleções contendo os *posts* específicos.



Fonte: Autoria própria (2025).

A estrutura hierárquica implementada no projeto permite consultas eficientes e escalabilidade. Por exemplo, a consulta de *posts* publicados utiliza o recurso `collectionGroup` para buscar em todas as subcoleções `category_posts` simultaneamente, aplicando filtros por área, categoria e *status* de publicação. Essa combinação de múltiplos critérios é viabilizada pela criação de *indexes* compostos no *Firestore*, que permitem junções eficientes entre filtros e ordenações específicas. Essa abordagem otimiza a performance das consultas, especialmente considerando o volume de dados acadêmicos e publicações que o Observatório gerencia.

A flexibilidade do esquema NoSQL permite que novos campos sejam adicionados aos documentos sem necessidade de migrações complexas, facilitando a evolução da aplicação conforme novos requisitos surgem. Esta característica é particularmente valiosa em um projeto acadêmico que pode expandir suas funcionalidades ao longo do tempo.

4.4.2 Cloud Storage

O *Firebase Storage* foi utilizado para armazenar os arquivos do Observatório de História e Geografia. A organização das pastas foi planejada para manter a separação lógica entre os diferentes tipos de conteúdo. Foram definidas duas pastas principais: `media` e `library`. A pasta `media` armazena as imagens associadas aos *posts* publicados no *site*, como ilustrações e demais elementos visuais. Já a pasta `library` é destinada aos documentos disponibilizados na biblioteca digital do Observatório. Dentro desta, foram criadas duas subpastas: `geography` e `history`, classificando os arquivos conforme a área do conhecimento à qual pertencem. Essa estrutura facilita tanto o gerenciamento interno dos arquivos quanto a organização e recuperação eficiente das mídias e documentos.

Quanto ao controle de acesso, a leitura dos arquivos é pública, permitindo que qualquer visitante do *site* visualize os conteúdos disponibilizados. No entanto, as permissões de escrita (envio, exclusão ou modificação de arquivos) são restritas a usuários autenticados com papéis de `ADMIN` ou `EDITOR`, desde que também estejam marcados como ativos no sistema. Essa política garante a segurança e integridade do repositório, evitando alterações indevidas e assegurando que apenas membros autorizados da equipe possam realizar modificações no armazenamento.

4.4.2.1 Análise de Custo

A fim de verificar a viabilidade financeira do uso do serviço, foi realizada uma estimativa dos gastos com o *Firebase Storage*, considerando o consumo previsto de armazenamento e tráfego de dados. A Tabela 3 apresenta os limites do plano gratuito e os valores cobrados após sua utilização.

Tabela 3 – Limites e custos do serviço *Firebase Storage*

Item	**Limite Gratuito**	**Valor após limite**
Armazenamento	5 GB	USD 0,026 por GB armazenado
Download de dados	1 GB/dia	USD 0,12 por GB baixado
Upload de operações	20 mil/dia	USD 0,05 por 10 mil uploads
Download de operações	50 mil/dia	USD 0,004 por 10 mil downloads

Fonte: Autoria própria (2025).

Considerando o custo por *gigabyte* armazenado por mês como C_g e o orçamento disponível como B, a capacidade máxima de armazenamento em *gigabytes* pode ser estimada por:

$$\text{Capacidade de armazenamento (GB)} = \frac{B}{C_g}$$

Supondo que cada documento tenha tamanho médio S_d (em *gigabytes*), o número máximo de documentos armazenáveis N será:

$$N = \frac{B}{C_g \times S_d}$$

Por exemplo, considerando documentos de 2 MB (0,002 GB) e custo de armazenamento de R$0,13 por GB, é possível armazenar milhares de documentos dentro do orçamento previsto.

Além disso, os custos com *download* e operações são relativamente baixos e não impactam significativamente o orçamento para volumes moderados de acesso.

Nesse cenário, identificou-se que os arquivos mais volumosos pertencem à seção da biblioteca digital, anteriormente gerenciada pelo *Tainacan*, composta majoritariamente por documentos em formato PDF. Como esse acervo não ultrapassa mil arquivos, concluiu-se que a demanda de armazenamento seria plenamente atendida pela solução proposta.

Adicionalmente, observou-se que a maior parte das mídias armazenadas corresponde a imagens vinculadas aos *posts*. Esses arquivos podem ser otimizados por meio de compressão, reduzindo o espaço necessário sem comprometer a qualidade visual.

Outro fator relevante é a frequência de operações no sistema: ações como criação, edição e remoção de *posts* e imagens não ocorrem de forma intensa ou contínua, o que contribui para manter os custos com *upload* e *download* de operações dentro dos limites gratuitos ou com impacto financeiro mínimo.

Dessa forma, a escolha pelo uso do *Firebase Storage* mostrou-se tecnicamente eficiente e economicamente viável, atendendo tanto às limitações orçamentárias quanto às necessidades de escalabilidade e manutenção do Observatório.

4.5 Migração de Dados

A migração de dados do Observatório foi e tem sido uma etapa fundamental na transição da plataforma anterior, baseada no sistema *WordPress*, para o novo sistema desenvolvido em *Flutter* com integração ao *Firebase*. Inicialmente, realizou-se um levantamento detalhado do

conteúdo existente, incluindo artigos, documentos e mídias, com o objetivo de organizá-los e transferi-los de forma estruturada para o banco de dados *Firestore*, além do armazenamento adequado dos arquivos no *Cloud Storage*.

Embora a lógica estrutural do *WordPress* tenha sido preservada no novo *site*, foram realizados aprimoramentos significativos no design visual, na responsividade e na experiência de navegação. A publicação dos dados não ocorreu de forma integral ou única; ao contrário, acompanhou a implementação iterativa das funcionalidades do painel administrativo. Dessa forma, à medida que uma funcionalidade era desenvolvida e validada junto à equipe do Observatório, a migração dos dados relacionados podia ser iniciada, garantindo entregas parciais e contínuas.

Por exemplo, após a conclusão e validação do formulário para criação de *posts* do tipo "Artigo", que corresponde a uma parte significativa das publicações do sistema anterior, iniciou-se a inserção manual desses conteúdos no novo ambiente. Esse processo replicava textos, imagens e metadados provenientes do *WordPress*, sendo aplicado também às categorias de publicação, dados da equipe, mídias e demais conteúdos associados. Essa abordagem incremental possibilitou que os dados migrados passassem a estar disponíveis no *site* público, favorecendo uma transição gradual e validada.

A migração da biblioteca digital, anteriormente gerenciada pelo *plugin Tainacan* no *WordPress*, foi realizada de forma automatizada. Os metadados e arquivos foram exportados em formato *CSV* com campos organizados e padronizados. Com base nesses arquivos, desenvolveu-se um *script* em *Python* para leitura dos dados, envio dos documentos ao *Firebase Storage* e criação das entradas correspondentes no *Firestore*. Essa automação acelerou a transferência de grande volume de conteúdo, minimizou erros humanos e garantiu a consistência entre o sistema antigo e o novo.

Por fim, após a migração e implementação das funcionalidades essenciais, foram realizados testes de usabilidade para avaliar a clareza da navegação, a eficiência dos fluxos de interação e a adequação do sistema às expectativas dos usuários. Essa estratégia possibilitou uma migração segura, contínua e validada, preparando o ambiente para oferecer uma experiência mais moderna, estável e intuitiva.

4.6 Publicação e Lançamento

A publicação do novo *site* do Observatório de Ensino de História e Geografia envolve um processo cuidadoso para garantir a transição segura e eficaz da plataforma. No ano 2025, o sistema migrado encontra-se hospedado em um subdomínio dedicado para testes e validações, acessível pelo endereço <https://curadoria.observatoriogeohistoria.net.br/>. Enquanto isso, o *site* original permanece ativo no domínio principal <https://observatoriogeohistoria.net.br/>.

Essa estratégia permite que as etapas finais de teste de usabilidade, ajustes de funcionalidades e validação da experiência do usuário sejam conduzidas sem impacto para o público geral. Além disso, a migração dos *posts* está sendo realizada de forma gradual, pois trata-se de um processo manual que exige cuidado para assegurar a integridade e fidelidade do conteúdo transferido.

Após a aprovação final das funcionalidades e da migração completa dos conteúdos, está prevista a substituição do *site* antigo pela nova versão desenvolvida em *Flutter*. Essa substituição será realizada de forma planejada, minimizando riscos e assegurando a continuidade do serviço para os usuários do Observatório.

A infraestrutura de hospedagem do novo sistema permanece na *Locaweb*, aproveitando a estrutura contratada na versão anterior, enquanto o gerenciamento do domínio é realizado pela *HostGator*. A configuração do *DNS* está ajustada para permitir a flexibilidade necessária durante esse período de transição, garantindo o acesso correto conforme o estágio de implantação do projeto.

4.7 Análise de Resultados

Durante a migração, desafios significativos foram enfrentados, como a reconstrução completa da plataforma, a adaptação da base de dados e a integração de novos recursos. Apesar dessas dificuldades, os resultados obtidos comprovam o sucesso da implementação e os benefícios de adotar uma arquitetura mais contemporânea e otimizada. Para avaliar objetivamente esses avanços, foram realizados testes com a ferramenta *Pingdom*[4], a partir da localização América do Sul – Brasil – São Paulo. Os principais resultados estão apresentados na Tabela 4.

A Tabela 4 evidencia melhorias expressivas no desempenho da nova versão da plata-

[4] Disponível em: <https://tools.pingdom.com>

Tabela 4 – Comparativo de desempenho entre o site antigo (*WordPress*) e o novo (*Flutter Web*), com testes realizados na ferramenta *Pingdom* a partir da localização América do Sul – Brasil – São Paulo.

Página	Versão	Nota de Performance	Tamanho da Página	Tempo de Carregamento
Sobre	Novo (*Flutter*)	A97	124.9 KB	653 ms
Sobre	Antigo (*WordPress*)	C78	1.9 MB	2.55 s
Experiências Educativas	Novo (*Flutter*)	A97	124.2 KB	526 ms
Experiências Educativas	Antigo (*WordPress*)	C78	1.5 MB	2.15 s
Biblioteca	Novo (*Flutter*)	A97	124.2 KB	527 ms
Biblioteca	Antigo (*WordPress*)	C78	10.4 MB	3.19 s

Fonte: Autoria própria (2025).

forma. O tempo de carregamento médio caiu de **2 a 3 segundos para menos de 700 ms** — ou seja, as páginas agora abrem quase instantaneamente. O tamanho médio das páginas foi reduzido de **1,5–10 MB para aproximadamente 124 KB**, uma diminuição superior a **90%** no volume de dados transferidos. Além disso, o número de arquivos que o navegador precisa solicitar para carregar uma página passou de **85–99 no site antigo** para apenas **4 no novo**, simplificando a comunicação com o servidor.

Essas mudanças resultam em uma experiência de navegação muito mais fluida e estável, com menor consumo de dados, carregamento mais rápido mesmo em conexões lentas e maior responsividade em dispositivos móveis. Do ponto de vista técnico, a nova arquitetura baseada em *Flutter Web* também oferece maior escalabilidade, facilitando futuras atualizações e integrações sem comprometer o desempenho.

Indicadores técnicos de forma simples:

- **Tempo de carregamento:** período entre o clique e a página completamente carregada. Quanto menor, melhor.
- **Tamanho da página:** volume de dados que o navegador precisa baixar. Menos MB significam carregamento mais rápido e menos consumo de internet.
- **Requisições:** número de arquivos (imagens, *scripts*, estilos, fontes) que o navegador precisa buscar. Menos requisições deixam o *site* mais ágil.
- **Nota de performance:** índice geral de eficiência da página. Notas mais altas indicam otimização melhor.
- **Cache do navegador:** cópias locais de arquivos, acelerando visitas futuras.

- **TTFB (*Time to First Byte*):** tempo que o servidor leva para começar a responder. Quanto menor, mais rápido o conteúdo chega ao usuário.

Em síntese, a migração para uma arquitetura baseada em *Flutter Web* demonstrou ganhos expressivos em desempenho, usabilidade e manutenibilidade. O projeto reforça a importância de aliar inovação tecnológica a objetivos educacionais, ampliando o alcance e a eficácia de iniciativas públicas voltadas ao fortalecimento do ensino de História e Geografia. Dessa forma, este trabalho consolida-se como uma contribuição relevante tanto para o avanço tecnológico aplicado à educação quanto para a promoção de práticas pedagógicas mais acessíveis e integradas ao contexto digital atual.

5 Conclusão

O desenvolvimento tecnológico apresentado, ao longo deste trabalho, reforça a importância da modernização de plataformas digitais de acesso livre, sobretudo quando vinculadas a iniciativas acadêmicas que buscam ampliar o alcance da educação pública de qualidade. A migração e reconstrução do *website* do Observatório de Ensino de História e Geografia, agora estruturado com o *framework Flutter* e integrado a um Painel Administrativo seguro e flexível, representam um avanço significativo na disponibilização de conteúdos digitais voltados ao ensino e à pesquisa.

Essa atualização não apenas aprimora o desempenho, a responsividade e a experiência de navegação dos usuários, mas também cria uma base tecnológica sólida para o crescimento futuro da plataforma, incluindo a integração de novas ferramentas e iniciativas associadas. Com essa infraestrutura, o Observatório se consolida como um ambiente digital inovador, colaborativo e alinhado às demandas contemporâneas da educação, promovendo práticas pedagógicas mais dinâmicas, interativas e acessíveis.

Dessa forma, este projeto contribui de maneira efetiva para fortalecer a disseminação do conhecimento nas áreas de História e Geografia, valorizando o papel das tecnologias digitais como instrumentos de transformação no processo de ensino-aprendizagem e no incentivo a práticas pedagógicas inovadoras, dentro e fora do ambiente universitário.

5.1 Trabalhos Futuros

Como perspectivas de trabalhos futuros, destacam-se algumas possibilidades de aprimoramento e expansão da plataforma. Uma delas é a implementação de uma versão PWA, que permitiria o acesso *offline* a conteúdos previamente carregados, além de instalação direta pelo navegador e notificações, ampliando a usabilidade do Observatório.

Outra frente relevante diz respeito à acessibilidade, por meio do desenvolvimento de recursos como ajuste de contraste, aumento de fonte, navegação por teclado e suporte a leitores de tela, garantindo maior inclusão de públicos diversos.

Também se considera fundamental a integração de ferramentas de *analytics*, que pos-

sibilitem acompanhar o desempenho do site, identificar conteúdos mais acessados e orientar decisões sobre melhorias. No âmbito do desenvolvimento de *software*, a adoção de testes automatizados representa um passo importante para assegurar a qualidade do código e prevenir falhas em futuras atualizações.

Ademais, o avanço das pesquisas em inteligência artificial abre caminho para explorar soluções de automação da categorização de conteúdos e da extração de metadados, reduzindo a carga manual da equipe e oferecendo recomendações personalizadas aos usuários.

Por fim, sugere-se ainda o fortalecimento da interoperabilidade com outros repositórios e plataformas educacionais, ampliando a visibilidade, a preservação digital e o intercâmbio de informações em rede.

Referências

(ABED), A. B. de Educação a D. **Definição de Educação a Distância**. 2025. <https://www.abed.org.br/site/conteudo/enciclopedia/educacao-a-distancia/>. Acessado em: 26 jul. 2025. Citado na página 16.

BERGIN, T. A history of the history of programming languages. **Communications of the ACM**, 2007. Disponível em: <https://www.researchgate.net/publication/220423253_A_history_of_the_history_of_programming_languages>. Citado na página 21.

BRASIL. **Decreto nº 5.622, de 19 de dezembro de 2005: Regulamenta o art. 80 da Lei nº 9.394/1996 (LDB) e define a modalidade de Educação a Distância**. 2005. <https://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/decreto/d5622.htm>. Acessado em: 26 jul. 2025. Citado na página 16.

BRASIL.(IBGE), I. B. de Geografia e E. **Indicadores de TIC 2024: acesso à Internet por tipo de equipamento no Brasil**. 2025. Acessado em: 18 ago. 2025. Disponível em: <https://agenciadenoticias.ibge.gov.br/en/agencia-press-room/2184-news-agency/news/44046-for-the-first-time-more-than-half-of-the-population-accesses-the-internet-from-a-tv.html>. Citado na página 48.

BRASIL.INEP. **Resumo técnico do Censo da Educação Superior 2022**. 2022. <https://download.inep.gov.br/publicacoes/institucionais/estatisticas_e_indicadores/resumo_tecnico_censo_educacao_superior_2022.pdf>. Acessado em: 9 out. 2024. Citado na página 13.

BURD, L. **Desenvolvimento de Software para Atividades Educacionais**. Dissertação (Dissertação de Mestrado) — Universidade Estadual de Campinas (UNICAMP), Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, 1999. Acessado em: 21 nov. 2024. Disponível em: <https://web.media.mit.edu/~leob/tese_total.pdf>. Citado 2 vezes nas páginas 19 e 20.

GRáCIO, J. C. A.; MADIO, T. C. d. C. O papel da preservação digital na curadoria digital. In: JORENTE, M. J. V.; SEGUNDO, R. S.; MONTOYA, J. A. F.; MARTíNEZ-ÁVILLA, D.; NAKANO, N. (Ed.). **Curadoria Digital e Gênero na Ciência da Informação**. Marília; São Paulo: Oficina Universitária; Cultura Acadêmica, 2021. p. 163–189. Acessado em: 26 jul. 2025. Disponível em: <https://ebooks.marilia.unesp.br/index.php/lab_editorial/catalog/download/273/2848/4955?inline=1>. Citado 2 vezes nas páginas 19 e 20.

HOWEY, R. A. Understanding software technology. **Knowledge, Technology & Policy**, 2002. Acesso em: 26 jul. 2025. Disponível em: <https://www.researchgate.net/publication/248139550_Understanding_software_technology>. Citado na página 21.

KIRKLAND, S. **FTP Deploy Action**. 2024. Acessado em: 27 jul. 2025. Disponível em: <https://github.com/SamKirkland/FTP-Deploy-Action>. Citado na página 36.

Manifesto Ágil. **Manifesto para Desenvolvimento Ágil de Software**. 2001. <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 14 jul. 2025. Citado na página 23.

MARTIN, R. C. **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. Prentice Hall, 2017. ISBN 0134494326, 9780134494326. Disponível em: <https://books.google.com.br/books/about/Clean_Architecture.html?id=uGE1DwAAQBAJ&redir_esc=y>. Citado na página 29.

MATHARU, G.; MISHRA, A.; SINGH, H.; UPADHYAY, P. Empirical study of agile software development methodologies: A comparative analysis. **ACM SIGSOFT Software Engineering Notes**, 2015. Acessado em: 27 jul. 2025. Disponível em: <https://www.researchgate.net/publication/276342954_Empirical_Study_of_Agile_Software_Development_Methodologies>. Citado na página 23.

MEDIUM. **The History of Flutter: From Origins to Global Adoption**. 2025. <https://alpondith.medium.com/the-history-of-flutter-from-origins-to-global-adoption-ddd5b1873dd3>. Acessado em: 27 jul. 2025. Citado na página 27.

MOHAMED, V.; SHAFFI, A. S. A study on model-view-view model (mvvm) design pattern. **International Journal of Emerging Technology and Advanced Engineering**, 2016. Disponível em: <https://www.researchgate.net/publication/373678906_A_study_on_Model-View-View_Model_MVVM_Design_Pattern/download>. Citado na página 29.

MORAN, J. M. **Personalização do aprendizado: trilhas formativas com curadoria e inteligência artificial**. 2021. Acessado em: 26 jul. 2025. Disponível em: <https://www2.eca.usp.br/moran/2021/04/06/personalizacao-do-aprendizado-trilhas-formativas-com-curadoria-e-inteligencia-artificial/>. Citado na página 17.

MOTA, M. Peixoto da; SANTOS, L. S. d. Os impactos da pandemia na acentuação da desigualdade digital. **Revista Metaxy**, 2024. ISSN 2526-5229. Acessado em: 17 ago. 2025. Disponível em: <https://revistas.ufrj.br/index.php/metaxy/article/view/64691/41117>. Citado na página 16.

OLIVEIRA, T. M. d.; SANTOS, F. V. "caminhando contra o vento, sem lenço e sem documento": Educação básica em tempos de pandemia. **Boletim de Conjuntura (BOCA)**, 2020. ISSN 2675-1488. Acessado em: 26 jul. 2025. Disponível em: <https://revista.ioles.com.br/boca/index.php/revista/article/view/41/36>. Citado 3 vezes nas páginas 18, 19 e 20.

QUADROS, J. M. de; WARPECHOWSKI, M. A utilização de repositórios de objetos de aprendizagem por professores das escolas de capão da canoa. **Revista Científica Trajetória Multicursos**, 2019. Acessado em: 11 out. 2024. Disponível em: <https://cientifica.cnec.br/index.php/trajetoria-multicursos/article/download/286/287/859>. Citado na página 13.

RICHARDS, M. **Software Architecture Patterns: Understanding Common Architecture Patterns and When to Use Them**. O'Reilly Media, 2015. ISBN 1491924241, 9781491924242. Disponível em: <https://books.google.com.br/books/about/Software_Architecture_Patterns.html?id=ZLYtuwEACAAJ&redir_esc=y>. Citado na página 28.

Robert C. Martin. **Clean Code: A Handbook of Agile Software Craftsmanship**. Pearson Education, 2008. ISBN 0136083250, 9780136083252. Disponível em: <https://books.google.com.br/books/about/Clean_Code.html?id=_i6bDeoCQzsC&redir_esc=y>. Citado na página 29.

ROCHA, D. G. da; GOUVEIA, L. M. B. Curadoria de conteúdo para ead: percepções de professores universitários, bibliotecários, avaliadores do mec e gestores. **ETD – Educação Temática Digital**, 2023. Acessado em: 26 jul. 2025. Disponível em: <https://periodicos.sbu.unicamp.br/ojs/index.php/etd/article/view/8667374>. Citado na página 17.

SADALAGE, P. J.; FOWLER, M. **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**. Upper Saddle River, NJ: Addison-Wesle, 2012. ISBN 013303612X, 9780133036121. Disponível em: <https://books.google.com.br/books?id=AyY1a6-k3PIC&hl=pt-BR&source=gbs_navlinks_s>. Citado na página 30.

SANTOS, E. **Formação online na pós-graduação stricto senso: Diários, visual storytelling, narrativas e autorias de uma pesquisadora em movimento nas Cidades e no Ciberespaço**. Pedro & João Editores, 2024. Acessado em: 17 ago. 2025. ISBN 978-65-265-1132-9, 978-65-265-1133-6. Disponível em: <https://pedroejoaoeditores.com.br/wp-content/uploads/2024/04/EBOOK_Formacao-online-na-pos-graduacao-stricto-senso.pdf>. Citado na página 17.

SOUSA, R. F. d.; LAGE, E. F. R. Educação e pandemia da covid-19: a importância das tecnologias de comunicação e informação neste cenário. **Revista Sítio Novo**, 2025. Acessado em: 17 ago. 2025. Disponível em: <https://www.researchgate.net/publication/390478446_Educacao_e_pandemia_da_covid-19_a_importancia_das_tecnologias_de_comunicacao_e_informacao_neste_cenarioEducation_and_the_Covid-19_Pandemic_the_importance_of_communication_and_information_technologies_>. Citado na página 16.

TAMANINI, P. A.; SOUZA, M. do S. As tecnologias digitais no ensino de história no brasil: Um mapeamento das pesquisas acadêmicas. **Revista Docência e Cibercultura**, 2019. Acessado em: 11 out. 2024. Disponível em: <https://www.e-publicacoes.uerj.br/re-doc/article/view/36814/27813>. Citado na página 13.

UNESCO. **Educação: Do fechamento das escolas à recuperação**. 2020. <https://www.unesco.org/pt/covid-19/education-response>. Acessado em: 9 out. 2024. Citado na página 12.