PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

COURSI:
Criação de uma Plataforma de Cursos Online

João Vitor dos Santos Couto

GOIÂNIA
2025

JOÃO VITOR DOS SANTOS COUTO

COURSI:
Criação de uma Plataforma de Cursos Online

Trabalho de conclusão de curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás como parte dos requisitos para obtenção do título de Bacharel em Ciências da Computação.

Orientador: Fernando Gonçalves Abadia

Goiânia
2025

JOÃO VITOR DOS SANTOS COUTO

COURSI:
Criação de uma Plataforma de Cursos Online

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás em ____/____/_____.

_____
Prof. Me. Luiz Álvaro de Oliveira Junior
Coordenador do Trabalho de Conclusão de Curso

Banca Examinadora:

_____
Prof. Me. Fernando Gonçalves Abadia

_____
Prof. Me. Rafael Leal Martins

_____
Prof. Me. Anibal Santos Jukemura

GOIÂNIA
2025

RESUMO

A crescente demanda por plataformas de ensino a distância, impulsionada pela popularização de tecnologias web e pela necessidade de flexibilização no acesso ao conhecimento, motiva o desenvolvimento de soluções robustas para gestão e distribuição de conteúdo educacional digital. Este trabalho apresenta o desenvolvimento de uma plataforma de cursos online voltada à criação, edição, publicação e consumo de conteúdos audiovisuais, integrando tecnologias modernas utilizadas amplamente na indústria de software. A solução é composta por um backend desenvolvido em Node.js com Express, banco de dados MongoDB, autenticação baseada em papéis e comunicação com serviços da AWS, especialmente o Amazon S3 para armazenamento seguro de vídeos. O frontend foi construído com Next.js, empregando renderização híbrida e componentes *client-side* para garantir interatividade e desempenho. Um diferencial da plataforma é o fluxo otimizado de upload de vídeos, que utiliza URLs assinadas e só realiza o envio efetivo ao servidor após a confirmação da publicação do curso, evitando arquivos órfãos e garantindo integridade dos dados. O projeto apresenta uma arquitetura modular, escalável e alinhada às boas práticas do mercado, demonstrando sua viabilidade técnica e utilidade prática no contexto de ensino digital.

Palavras-chave: Ensino a distância. Desenvolvimento web. Next.js. AWS. Plataforma educacional.

ABSTRACT

The growing demand for distance learning platforms, driven by the widespread adoption of web technologies and the need for flexible access to knowledge, has motivated the development of robust solutions for managing and distributing digital educational content. This work presents the development of an online course platform designed for the creation, editing, publishing, and consumption of audiovisual materials, integrating modern technologies widely used in the software industry. The solution includes a backend built with Node.js and Express, a MongoDB database, role-based authentication, and integration with AWS services, particularly Amazon S3, for secure video storage. The frontend was developed using Next.js, employing hybrid rendering and client-side components to ensure interactivity and high performance. A key feature of the platform is its optimized video upload workflow, which uses pre-signed URLs and performs the actual upload only after the course publication is confirmed, preventing orphaned files and ensuring data integrity. The project adopts a modular and scalable architecture, aligned with current industry best practices, demonstrating its technical feasibility and practical applicability within the context of digital education.

Keywords: Distance learning. Web development. Next.js. AWS. Educational platform.

LISTA DE SIGLAS

API	*Application Programming Interface*
AWS	*Amazon Web Services*
CRUD	*Create, Read, Update and Delete*
EAD	Educação a Distância
HTTP	*Hypertext Transfer Protocol*
ISR	*Incremental Static Regeneration*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
NPM	*Node Package Manager*
RBAC	*Role-Based Access Control*
REST	*Representational State Transfer*
SDK	*Software Development Kit*
S3	*Simple Storage Service*
SSR	*Server-Side Rendering*
SSG	Static Site Generation
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
VS Code	Visual Studio Code
W3C	World Wide Web Consortium

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O ensino online consolidou-se como uma das principais modalidades educacionais, impulsionado pela popularização de tecnologias digitais e pela necessidade crescente de flexibilização do processo de aprendizagem. O cenário da pandemia, especificamente, acelerou o uso de plataformas digitais e ferramentas de ensino remoto, promovendo uma transformação no modo como aprendemos e ensinamos (PAIVA, 2021).

Neste contexto, o presente trabalho tem como objetivo apresentar o desenvolvimento de uma plataforma completa de cursos online, implementada utilizando tecnologias modernas como Node.js, Express, MongoDB, Next.js e AWS S3. A plataforma permite a criação, edição e publicação de cursos por instrutores, além de possibilitar a visualização e consumo de conteúdos pelos estudantes.

1.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver uma plataforma web completa de cursos online que permita a criação, gerenciamento e consumo de conteúdos educacionais em formato de vídeo, garantindo segurança, escalabilidade e uma experiência de usuário simples e eficiente.

1.2 Objetivos específicos

- Desenvolver um backend seguro utilizando Node.js e Express.
- Implementar autenticação baseada em JWT utilizando cookies HttpOnly.
- Construir um frontend dinâmico com Next.js, com páginas protegidas e integração com o backend.
- Criar um sistema completo de upload seguro de vídeos utilizando URLs assinadas da AWS.
- Integrar banco de dados MongoDB com modelagem adequada das entidades.
- Implementar fluxo de CRUD de cursos: criação, edição, exclusão e visualização.
- Documentar todos os procedimentos técnicos de forma clara e detalhada.

1.3 Justificativa

A escolha do desenvolvimento de uma plataforma de cursos online se justifica pela crescente demanda por soluções de ensino a distância, principalmente após a expansão do uso de tecnologias digitais no ambiente educacional. Plataformas como Udemy e Coursera demonstram que o modelo de cursos online democratiza o acesso ao conhecimento, permitindo que estudantes aprendam de forma flexível e instrutores distribuam conteúdo de maneira escalável.

Além disso, do ponto de vista tecnológico, o projeto é relevante pois integra diversas tecnologias modernas amplamente utilizadas no mercado, como Next.js, Node.js e AWS S3. A implementação de upload seguro, autenticação robusta e arquitetura distribuída reflete práticas reais encontradas em aplicações profissionais, contribuindo para a formação técnica e prática no desenvolvimento de sistemas web.

1.4 Metodologia

A metodologia utilizada neste trabalho baseia-se em um processo incremental de desenvolvimento, permitindo que funcionalidades fossem planejadas, implementadas e validadas progressivamente. Este método possibilitou ajustes contínuos de acordo com os requisitos identificados ao longo da evolução da plataforma.

- Levantamento de requisitos: identificação das funcionalidades essenciais, como cadastro de usuários, criação de cursos, upload de vídeos e gerenciamento de conteúdos.
- Estudo das tecnologias: análise detalhada das ferramentas escolhidas, como Express para backend, Next.js para frontend e AWS S3 para armazenamento de vídeos.
- Modelagem do sistema: criação dos modelos de dados, definição das entidades e estrutura geral da aplicação.
- Implementação incremental: desenvolvimento das funcionalidades em etapas, começando pela autenticação, depois cursos, upload de vídeos e por fim a integração geral do sistema.
- Testes: utilização de ferramentas como Postman e testes manuais no frontend para validar fluxos críticos.

- Documentação: registro de todo o processo de desenvolvimento, decisões técnicas e justificativas.

Essa metodologia incremental permite maior controle sobre o processo, facilitando ajustes e garantindo que cada etapa seja validada antes de avançar para a próxima.

1.4.1 Tecnologias Utilizadas

- Backend: Node.js, Express.js, JWT, Mongoose, MongoDB.
- Frontend: Next.js 14, React, TailwindCSS, shadcn/ui.
- Infraestrutura: AWS S3, CloudFront.
- Ferramentas auxiliares: Git, VS Code, Postman.

1.5 Organização do Trabalho

Este trabalho está estruturado da seguinte forma:

- O Capítulo 1 apresenta a introdução, objetivos, justificativa, metodologia e organização geral da monografia.
- O Capítulo 2 contém o embasamento teórico necessário, abordando conceitos sobre EAD, arquitetura web, tecnologias utilizadas e fundamentos do desenvolvimento web moderno.
- O Capítulo 3 descreve os materiais utilizados e métodos aplicados, explicando as ferramentas e tecnologias empregadas.
- O Capítulo 4 apresenta o desenvolvimento completo do sistema, detalhando arquitetura, funcionalidades, fluxos, telas e código.
- O Capítulo 5 mostra os resultados obtidos e a análise final da plataforma desenvolvida.
- Por fim, são apresentadas as Considerações Finais e as Referências utilizadas no trabalho.

Encerrando este capítulo, destaca-se que o desenvolvimento da plataforma oferece não apenas um produto funcional, mas também uma experiência de integração de múltiplas tecnologias modernas, permitindo compreender desafios comuns em aplicações reais e as soluções adotadas para cada um deles.

2. FUNDAMENTAÇÃO TEÓRICA

O presente capítulo tem como objetivo apresentar os fundamentos teóricos que sustentam o desenvolvimento da plataforma de cursos online proposta neste trabalho. São abordados conceitos essenciais sobre Educação a Distância (EAD) e Experiência do Usuário (UX) em ambientes educacionais, bem como os princípios de arquitetura web moderna que orientaram as decisões de projeto. Além disso, são exploradas as tecnologias utilizadas na implementação da solução Node.js, Express, MongoDB, Next.js e Amazon AWS S3, detalhando seus conceitos, características técnicas e papel dentro da aplicação. Por fim, são discutidos aspectos relacionados à segurança em aplicações web e ao processo de upload seguro de vídeos, assegurando a integridade e a escalabilidade do sistema desenvolvido.

2.1 Educação a Distância (EAD)

A Educação a Distância (EAD) consolidou-se como uma das formas mais relevantes de aprendizagem no contexto contemporâneo, impulsionada pelo avanço de tecnologias de rede, plataformas digitais e ambientes virtuais interativos. De acordo com Moore e Kearsley (2012), o EAD caracteriza-se pela separação física entre estudante e instrutor, unida por recursos tecnológicos que permitem comunicação síncrona e assíncrona. Essa modalidade proporciona flexibilidade temporal e espacial, ampliando o acesso ao conhecimento para públicos que, por diferentes motivos, não conseguem participar de ambientes presenciais tradicionais.

O crescimento das plataformas EAD está intimamente relacionado à expansão da internet de alta velocidade e às mudanças no comportamento do consumidor digital. Em países como o Brasil, a ascensão de plataformas como Udemy, Coursera, Alura e Hotmart evidenciou novas formas de comercialização de conhecimento, permitindo que instrutores independentes alcancem audiências globais. A pandemia de COVID-19, segundo dados da UNESCO (2020), acelerou ainda mais esse processo, tornando o ambiente virtual um espaço central para atividades educacionais.

Nesse contexto, surgem demandas por soluções tecnológicas personalizadas, capazes de unir robustez técnica, boa experiência do usuário e ferramentas de gestão adequadas. A plataforma desenvolvida neste projeto insere-se nessa tendência,

atendendo tanto instrutores que desejam publicar seus conteúdos quanto estudantes em busca de capacitação acessível.

2.2 Experiência do Usuário (UX) em Plataformas Educacionais

A Experiência do Usuário (UX) desempenha papel fundamental na eficácia de plataformas digitais, sobretudo em ambientes educacionais. Conforme Norman (2013), a UX está relacionada às percepções, emoções, comportamentos e reações do usuário durante a interação com um produto ou sistema. Em plataformas educacionais, uma boa experiência é decisiva para melhorar a retenção do conteúdo, minimizar frustrações durante o uso e garantir continuidade no processo de aprendizagem.

Garrett (2011) destaca que sistemas complexos devem ser projetados com base em cinco camadas conceituais: estratégia, escopo, estrutura, esqueleto e superfície visual. Aplicando esse referencial ao contexto EAD, observa-se que a clareza de navegação, hierarquia visual bem definida, tipografia legível, responsividade e acessibilidade são elementos essenciais para que alunos consigam consumir conteúdo audiovisual sem distrações ou dificuldades técnicas.

Além disso, plataformas educacionais devem considerar princípios específicos, como:

- Minimização da carga cognitiva, evitando excesso de elementos visuais;
- Feedback imediato, principalmente em ações como upload de vídeos e salvamento de cursos;
- Consistência visual, reduzindo a necessidade de aprendizado da interface;
- Acessibilidade, garantindo que pessoas com deficiências possam navegar de forma adequada (W3C, 2023);
- Engajamento, com elementos como progresso, módulos organizados e interface convidativa.

A plataforma desenvolvida adotou tais princípios ao utilizar componentes visuais padronizados no Next.js, estilização responsiva e feedbacks visuais durante operações críticas, como autenticação e envio de vídeos.

2.3 Arquitetura Web Moderna

A evolução da computação em nuvem e dos ambientes JavaScript transformou profundamente a arquitetura de aplicações web. Richards (2015) destaca que sistemas modernos adotam estruturas distribuídas, combinando serviços independentes que se comunicam por APIs. Esses modelos priorizam desempenho, tolerância a falhas e escalabilidade horizontal, algo essencial para plataformas que manipulam conteúdos audiovisuais.

A arquitetura da plataforma em questão foi projetada seguindo o paradigma de aplicação distribuída, composta por quatro camadas principais detalhadas a seguir.

- Camada de Interface (Frontend): Desenvolvida em Next.js, responsável pela interação com o usuário. Ela consome APIs REST e exibe os dados de forma dinâmica, utilizando renderização híbrida (SSR + SSG) para melhorar o tempo de carregamento.
- Camada de Aplicação (Backend): Construída em Node.js com Express, essa camada gerencia lógica de negócios, autenticação, permissões e comunicação com o banco de dados.
- Camada de Dados (Banco MongoDB): Armazena documentos estruturados contendo as informações de cursos, aulas e usuários.
- Armazenamento de Arquivos (AWS S3): Responsável por armazenar grandes volumes de vídeos com segurança, durabilidade e baixo custo.

A utilização combinada dessas camadas reflete uma arquitetura moderna, robusta e amplamente aplicada na indústria.

2.4 Node.js: Conceitos Fundamentais

Node.js é um ambiente de execução JavaScript criado por Ryan Dahl em 2009, construído sobre o motor V8 do Google Chrome. Sua principal inovação está na adoção de um modelo assíncrono e orientado a eventos, que elimina o bloqueio de operações de entrada e saída e permite que o servidor lide com milhares de requisições simultâneas sem degradação significativa de desempenho. De acordo com Teixeira (2019), esse modelo não bloqueante tornou o Node.js especialmente eficiente em aplicações que necessitam manipular múltiplas conexões concorrentes, como APIs REST, serviços de streaming e aplicações em tempo real.

A literatura técnica aponta que o event loop, mecanismo interno responsável por orquestrar tarefas assíncronas, é um dos elementos centrais para o desempenho do Node.js (Shanahan, 2018). Esse recurso possibilita que chamadas de rede, consultas a banco de dados ou operações de leitura de arquivos sejam executadas sem interromper o fluxo principal da aplicação, garantindo escalabilidade tanto vertical quanto horizontal. Segundo a documentação oficial (NODE.JS FOUNDATION, 2024), a eficiência desse modelo se deve ao fato de que apenas uma thread principal é utilizada para coordenar tarefas assíncronas despachadas para workers internos, reduzindo o overhead característico de arquiteturas baseadas em múltiplas threads.

Outro fator determinante para a popularidade do Node.js é seu extenso ecossistema de bibliotecas disponibilizadas no NPM, atualmente considerado o maior repositório de pacotes do mundo. Essa característica permite que desenvolvedores integrem soluções prontas para autenticação, segurança, criptografia, manipulação de arquivos, validação de dados e diversas outras funcionalidades, acelerando o desenvolvimento e garantindo consistência no backend.

No contexto deste projeto, o Node.js desempenhou papel fundamental ao implementar o núcleo da lógica de negócio da plataforma de cursos. Ele foi responsável por validar permissões de alunos e instrutores, controlar o fluxo de criação, edição e exclusão de cursos e aulas, além de gerenciar autenticação baseada em JWT armazenado em cookies HttpOnly — abordagem recomendada por práticas modernas de segurança web. O backend desenvolvido também integrou-se ao AWS SDK para geração de URLs temporárias de upload e download de vídeos no S3, garantindo um fluxo seguro e eficiente para manipulação de arquivos. Essas capacidades demonstraram a adequação do Node.js para aplicações que demandam alto desempenho em I/O e integração com serviços externos em escala.

De forma geral, o uso do Node.js contribuiu para que o backend da plataforma fosse leve, modular e escalável, ao mesmo tempo em que manteve um padrão de segurança adequado para manipulação de dados sensíveis. Sua arquitetura orientada a eventos se mostrou compatível com as exigências operacionais da plataforma, confirmando sua relevância e justificando sua adoção como tecnologia principal no servidor.

2.5 Express: Framework HTTP para APIs Web

Express é um dos frameworks mais utilizados no ecossistema Node.js, oferecendo uma abordagem minimalista, flexível e intuitiva para construção de APIs. O Express estrutura aplicações em middlewares, que podem ser encadeados para realizar funções específicas, como autenticação, validação e roteamento.

Na plataforma, o Express desempenhou funções fundamentais:

- Organização das rotas: Cada funcionalidade — cursos, usuários, uploads, matrículas — é agrupada em um módulo de rotas próprio.
- Middleware de autenticação: O arquivo *authMiddleware.ts* intercepta requisições, valida tokens e injeta os dados do usuário no objeto req.
- Controle de permissões (RBAC): As rotas validam o papel(*role*) do usuário, permitindo apenas que instrutores criem e editem cursos.
- Manipulação de erros: O backend aplica validação de entrada, retornando mensagens claras ao frontend para feedback ao usuário.
- Integração com MongoDB via Mongoose: O Express atua como intermediário entre o banco de dados e o frontend, persistindo e retornando documentos.

Assim, o Express é o alicerce que garante organização, clareza e segurança ao backend.

2.6 MongoDB: Estrutura de Dados e Flexibilidade Documental

MongoDB é um banco de dados NoSQL orientado a documentos, amplamente utilizado em aplicações que necessitam de flexibilidade e escalabilidade. Chodorow (2013) afirma que documentos em JSON permitem modelar dados de forma natural para aplicações modernas, sem a rigidez de esquemas relacionais.

A plataforma utiliza MongoDB para armazenar dados de usuários, cursos, aulas e matrículas. A modelagem da entidade Curso, por exemplo, utiliza listas aninhadas representando as aulas do curso:

Código 1: Estrutura das aulas dentro da entidade “Curso” no MongoDB.

```
//…
aulas: [
  {
    id: string;
    titulo: string;
    url: string;
  }
]
```

Fonte: MongoDB cmd. Autoria própria (2025).

Essa estrutura facilita a manipulação das aulas no backend e no frontend, evitando junções complexas e mantendo boa performance.

2.7 Next.js: Framework para Interfaces Modernas

Next.js é um framework moderno criado sobre o React com o objetivo de facilitar o desenvolvimento de interfaces web de alto desempenho e estruturação simplificada. De acordo com Vercel (2024), mantenedora oficial do framework, o Next.js adota uma abordagem híbrida de renderização que combina diferentes estratégias, como Server-Side Rendering (SSR), Static Site Generation (SSG) e Incremental Static Regeneration (ISR), permitindo ao desenvolvedor balancear desempenho, segurança e atualização dinâmica de conteúdo conforme as necessidades da aplicação. Essa flexibilidade arquitetural é um dos fatores que têm colocado o Next.js entre as ferramentas mais utilizadas no desenvolvimento frontend contemporâneo.

Outro aspecto importante é o sistema de roteamento baseado em diretórios, especialmente após a introdução do App Router nas versões mais recentes. A existência nativa de componentes server-side também melhora a segurança da aplicação, evitando a exposição de dados sensíveis na camada do cliente e possibilitando a pré-renderização eficiente de páginas.

No que diz respeito ao desempenho, o Next.js incorpora automaticamente técnicas como minificação, tree-shaking, code splitting e otimização de imagens, garantindo que apenas o essencial seja transmitido ao navegador (VERCEL, 2024). Tais otimizações foram fundamentais para que a plataforma desenvolvida neste trabalho apresentasse tempos de carregamento reduzidos e navegação fluida, especialmente nas

páginas que lidam com conteúdo dinâmico, como listagem de cursos, formulários de criação e visualização de aulas.

Além disso, a capacidade de integrar facilmente mecanismos de autenticação, como cookies HttpOnly enviados automaticamente em requisições ao backend, tornou o Next.js particularmente adequado para atender às exigências de segurança da plataforma. A modularidade dos componentes facilitou a criação de elementos reutilizáveis, como cartões de cursos, formulários e elementos de layout, promovendo um desenvolvimento mais rápido e consistente.

Por fim, a previsibilidade oferecida pela organização do framework e sua compatibilidade com padrões contemporâneos de engenharia de software contribuíram de forma significativa para que a aplicação desenvolvida atingisse um elevado grau de qualidade e manutenibilidade. A estrutura clara e a harmonia com bibliotecas do ecossistema React garantiram um ambiente ideal não apenas para a implementação inicial, mas também para a expansão futura da plataforma.

2.8 Amazon AWS S3: Armazenamento em Nuvem

O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento em nuvem oferecido pela Amazon Web Services (AWS) e amplamente utilizado na indústria por sua escalabilidade, segurança, disponibilidade e durabilidade. Criado para armazenar objetos de qualquer natureza (imagens, vídeos, documentos, backups e arquivos estáticos), o S3 tornou-se uma solução padrão em sistemas que necessitam lidar com grandes quantidades de dados distribuídos.

Segundo a própria Amazon (AWS, 2023), o S3 oferece durabilidade de 99.999999999% (11 noves), graças ao armazenamento redundante em múltiplas zonas de disponibilidade. Isso significa que os dados são replicados internamente em diferentes locais físicos, o que reduz consideravelmente o risco de perda permanente de informações, mesmo em casos de falha catastrófica em um data center.

Além de durabilidade, o S3 oferece:

2.8.1 Escalabilidade automática

Não há necessidade de gerenciar servidores ou sistemas de arquivos. O S3 amplifica sua capacidade automaticamente para atender a qualquer demanda, seja para

poucos arquivos ou petabytes de dados. Essa característica é crucial para plataformas educacionais que manipulam vídeos, cujo tamanho pode crescer rapidamente conforme novos cursos são adicionados.

2.8.2 URLs pré-assinadas (Pre-signed URLs)

Um dos recursos centrais utilizados no projeto é o uso de URLs pré-assinadas, que são links temporários gerados pelo backend permitindo upload de vídeos diretamente do navegador para o S3 e download temporário de vídeos sem expor o bucket ao público.

Esse mecanismo traz diversas vantagens como reduzir a carga no backend, pois o servidor não recebe arquivos grandes e apenas gera permissões temporárias. Diminui custos de infraestrutura, como o tráfego não passa pelo backend, o consumo de banda do servidor é drasticamente reduzido. As URLs expiram automaticamente após o período configurado (1 hora, no caso do projeto), impedindo acessos indevidos e aumentando a segurança. Também evita que o frontend conheça credenciais da AWS O cliente apenas recebe a URL temporária, mantendo o ambiente seguro.

2.8.3 Integração com o projeto

No contexto da plataforma desenvolvida, o S3 atua como o principal repositório de vídeos das aulas dos cursos. O fluxo funciona da seguinte forma:

1. O instrutor acessa a página de criação ou edição do curso.
2. Ao selecionar um vídeo, o frontend solicita ao backend uma URL assinada.
3. O backend verifica se o usuário tem papel de instrutor, utilizando o middleware de autenticação.
4. O backend gera uma URL temporária via AWS SDK (@aws-sdk/client-s3).
5. O frontend envia o vídeo diretamente ao S3 utilizando o método PUT.
6. A chave do arquivo (key) — não o vídeo — é salva no banco MongoDB.
7. Quando o curso é excluído, o backend remove do banco os dados do curso e executa `DeleteObjectCommand` para apagar o vídeo correspondente.

Esse fluxo combina desempenho, segurança, baixo custo e boa experiência ao usuário. A escolha do S3 foi estratégica para garantir escalabilidade real, já que os vídeos podem se tornar rapidamente o maior consumidor de recursos e banda de qualquer plataforma educacional.

2.9 Segurança em Aplicações Web

A segurança em aplicações web é um fator essencial no desenvolvimento de sistemas modernos, principalmente em plataformas que manipulam dados sensíveis, como credenciais de usuários e conteúdos pagos. Segundo a OWASP Foundation (2022), vulnerabilidades relacionadas a autenticação, injeção de dados, acesso indevido e exposição de informações continuam entre os principais riscos em aplicações distribuídas. Assim, a adoção de boas práticas de segurança se torna indispensável para garantir a integridade, confidencialidade e disponibilidade dos dados.

No desenvolvimento da plataforma de cursos online, diversas medidas foram implementadas para mitigar riscos e seguir padrões recomendados. Uma dessas medidas foi o armazenamento do token JWT em cookies do tipo HttpOnly, impedindo que scripts maliciosos no navegador tenham acesso direto ao token, conforme recomendado pela OWASP para prevenção de ataques de roubo de sessão. Além disso, o backend emprega um middleware de autenticação que valida a assinatura do token, verifica sua integridade e impede o acesso de usuários não autenticados às rotas protegidas.

Outro elemento importante da arquitetura de segurança foi a implementação de um controle de permissões baseado em papéis (RBAC). De acordo com Sandhu et al. (2011), modelos RBAC são eficazes para limitar acessos e garantir que cada usuário tenha apenas os privilégios necessários para sua função. Esse mecanismo assegura que os instrutores possam criar, editar e excluir cursos, enquanto alunos têm seu acesso restrito apenas ao consumo de conteúdos liberados.

O sistema também incorpora validações rigorosas de entrada, incluindo verificação de tipos de arquivos, tamanho de vídeos, formatos aceitos e sanitização de inputs textuais. Essas práticas reduzem a superfície de ataque relacionada à uploads maliciosos, scripts embutidos ou dados inesperados. No nível da infraestrutura, foram aplicadas políticas restritivas no Amazon S3, garantindo que apenas usuários autorizados possam enviar ou excluir arquivos, enquanto a aplicação não permite uploads diretos que não passem pelo processo de validação da API.

Por fim, a eliminação automática de arquivos órfãos — vídeos enviados mas não associados a cursos — reduz riscos de exposição de conteúdo e impede o acúmulo desnecessário de dados sensíveis na nuvem. Esse conjunto de medidas consolida uma arquitetura robusta, que segue diretrizes modernas de segurança e se adequa às exigências de uma plataforma de ensino baseada em vídeo.

2.10 Upload Seguro de Vídeos

O upload de vídeos é uma funcionalidade central na plataforma, e sua implementação exigiu cuidados especiais para garantir eficiência, segurança e escalabilidade. Em vez de enviar os arquivos diretamente para o backend, foi adotada uma arquitetura baseada em URLs pré-assinadas, um modelo amplamente recomendado por provedores de nuvem como a Amazon Web Services (AWS, 2023). Esse mecanismo permite que o navegador envie arquivos diretamente ao Amazon S3, enquanto o servidor permanece responsável apenas pela geração e validação das URLs temporárias.

O fluxo funciona da seguinte forma: inicialmente, o instrutor solicita ao backend uma URL para upload. O servidor verifica o papel do usuário, valida suas permissões e então solicita ao S3 uma URL pré-assinada com tempo de expiração limitado. Apenas após essa verificação o frontend recebe a URL, que é utilizada pelo navegador para enviar o vídeo diretamente ao bucket S3 por meio de uma requisição HTTP PUT. Esse arquivo se torna acessível temporariamente apenas para operações de upload e não pode ser lido publicamente, cumprindo recomendações de segurança da AWS.

Segundo a documentação oficial da Amazon (AWS, 2023), esse modelo reduz custos operacionais e riscos de segurança, pois evita o tráfego de arquivos grandes pelo backend e diminui significativamente a superfície de ataque da aplicação. Durante os testes deste projeto, verificou-se que o uso de URLs pré-assinadas ofereceu ganhos substanciais de desempenho e eliminou gargalos que seriam inevitáveis caso o envio dependesse do processamento do servidor.

Além disso, a expiração automática da URL impede usos indevidos, já que ela apenas pode ser utilizada dentro do intervalo determinado no momento de sua criação. A plataforma também valida os tipos de arquivos e limita o tamanho permitido para upload, garantindo que apenas vídeos em formatos adequados possam ser enviados. Esses mecanismos demonstram que a solução adotada não apenas otimiza o desempenho da aplicação, mas também atende às melhores práticas de segurança recomendadas para serviços de armazenamento em nuvem.

3. MATERIAIS E MÉTODOS

O desenvolvimento da plataforma de cursos online apresentada neste trabalho exigiu a utilização de um conjunto integrado de tecnologias, ferramentas e metodologias capazes de atender aos requisitos de escalabilidade, segurança, desempenho e facilidade de uso. Este capítulo descreve, de maneira aprofundada e contínua, tanto os materiais empregados quanto os métodos adotados ao longo do processo de criação do sistema, incluindo as etapas de planejamento, modelagem, implementação e validação. A abordagem aqui apresentada busca evidenciar a relação direta entre as escolhas tecnológicas e as necessidades práticas do projeto, bem como demonstrar como cada componente contribuiu de forma significativa para o resultado final.

A seleção dos materiais utilizados foi orientada pelos objetivos funcionais do sistema, considerando especialmente a necessidade de criar uma solução moderna, robusta e alinhada às práticas atuais do desenvolvimento web. Para isso, optou-se pela utilização do Node.js no backend, uma vez que sua arquitetura baseada no motor V8 e seu modelo de operações assíncronas oferecem alto desempenho, especialmente em aplicações orientadas a serviços e processamento de requisições. A adoção do Express como framework principal complementou essa escolha, fornecendo uma estrutura organizada e minimalista para a criação das rotas, middlewares, validações e lógica de negócio. A combinação dessas tecnologias permitiu construir uma API segura, modular e eficiente, capaz de atender às exigências do sistema de cursos, matrículas e gerenciamento de usuários.

No frontend, a escolha do Next.js foi motivada pela necessidade de oferecer uma experiência fluida aos usuários, garantindo carregamento rápido, interface reativa e mecanismos adequados de proteção de rotas. O Next.js, por possibilitar renderização híbrida e componentização avançada, facilitou a implementação de páginas destinadas a instrutores, estudantes e administradores, além de permitir a integração direta com a API do backend por meio de requisições autenticadas. O uso de componentes reutilizáveis e padrões visuais consistentes garantiu ao sistema um caráter profissional e coerente, ao mesmo tempo em que possibilitou ao usuário final navegar pela plataforma com naturalidade e previsibilidade.

Para armazenamento dos dados, o MongoDB foi escolhido por sua flexibilidade e aderência à estrutura da aplicação. Por ser um banco de dados NoSQL orientado a documentos JSON, o MongoDB permitiu modelar entidades como “Curso” e “Aula” de

maneira intuitiva, incluindo arrays aninhados de objetos representando cada aula pertencente a um curso. Essa característica facilitou o processo de criação e atualização dos conteúdos enviados pelos instrutores, assim como a recuperação eficiente dos dados para apresentação no frontend. Ademais, o uso do Mongoose como camada de abstração permitiu garantir integridade (por meio de esquemas), validações e operações mais seguras no banco.

Como a plataforma depende de vídeos enviados pelos instrutores, tornou-se essencial adotar um mecanismo de armazenamento que fosse escalável, seguro e de alta durabilidade. Por essa razão, escolheu-se o Amazon S3, que oferece armazenamento distribuído com durabilidade de 11 noves e a possibilidade de integração por meio de URLs pré-assinadas. Essa abordagem permitiu que os uploads de vídeos fossem realizados diretamente do navegador para a nuvem, sem sobrecarregar o backend e reduzindo consideravelmente o consumo de banda do servidor. Além disso, as URLs assinadas garantem segurança ao limitar o tempo de validade do link, impedindo que arquivos possam ser enviados ou baixados fora do período permitido. O backend, nesse contexto, atua apenas como intermediário para autenticar o instrutor e solicitar ao S3 a emissão da URL temporária, garantindo que somente usuários autorizados possam enviar conteúdos.

No processo de desenvolvimento, diversas ferramentas desempenharam papel fundamental. O Visual Studio Code foi utilizado como ambiente de desenvolvimento integrado, oferecendo suporte a extensões, depuração e formatação de código. O Postman foi utilizado para testar as rotas da API, validar respostas, analisar falhas e garantir que o backend se comportava conforme o esperado. O Git e o GitHub foram empregados para controle de versão, colaborando para organização do código e registro do histórico de alterações.

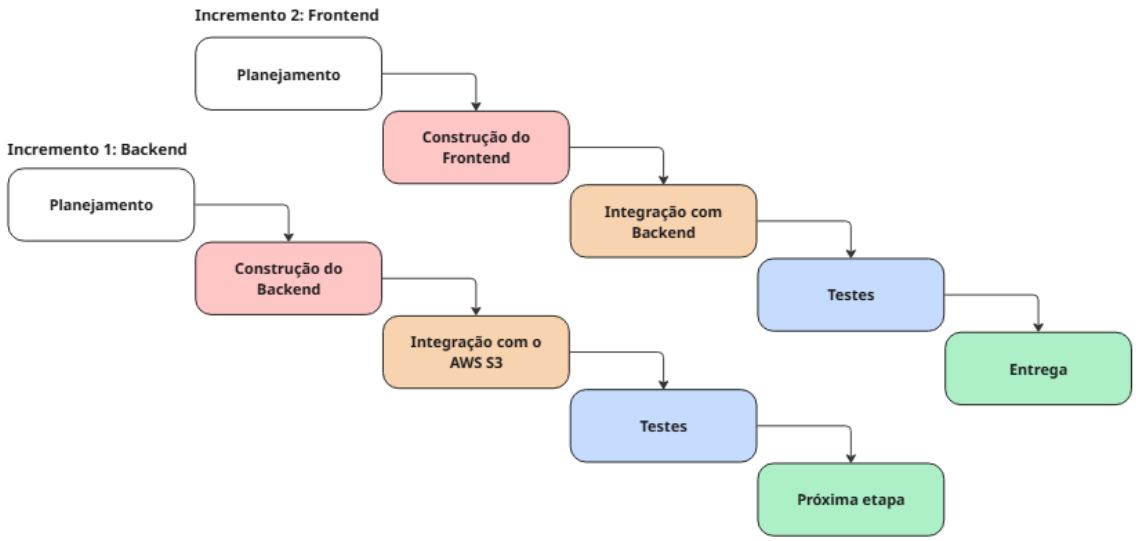
O método de desenvolvimento adotado seguiu uma abordagem incremental, inspirada nos princípios das metodologias ágeis. Essa escolha se mostrou apropriada para o projeto, dado que a plataforma evoluiu gradualmente desde requisitos essenciais até funcionalidades mais complexas. Na primeira etapa, trabalhou-se no levantamento de requisitos, definindo claramente o escopo mínimo necessário para que a plataforma fosse funcional e coerente, incluindo mecanismos de autenticação, criação de cursos e armazenamento de vídeos. A etapa seguinte consistiu no estudo das tecnologias selecionadas, buscando compreender profundamente suas características e limitações, garantindo escolhas adequadas e justificadas.

A modelagem do sistema foi elaborada com foco na simplicidade e eficiência. Foram definidos os modelos de dados que representam usuários, cursos, aulas e matrículas, cada um deles estruturado de forma a facilitar tanto a persistência quanto a leitura dos dados pelo backend e pelo frontend. A divisão em módulos garantiu uma organização clara do backend, separando responsabilidades entre rotas, modelos, utilitários e middlewares. A modelagem do frontend seguiu o padrão de organização do Next.js App Router, distribuindo as páginas em pastas que representavam ações específicas, como criação de cursos, visualização de aulas e gestão de perfil.

A implementação ocorreu de forma progressiva. Primeiramente, foi construída a API responsável pelo cadastro e login de usuários, incluindo a configuração do JWT em cookies seguros. Em seguida, foram implementadas as funcionalidades relativas a cursos, permitindo que instrutores criassem, editassem e excluíssem conteúdos. Na etapa subsequente, desenvolveu-se a integração com o AWS S3, garantindo que vídeos pudessem ser enviados e recuperados com segurança. Foram realizados testes manuais utilizando o Postman, garantindo que as rotas estavam funcionando corretamente e que os mecanismos de validação não permitiam comportamentos indesejados. A implementação da interface gráfica seguiu após a validação dos testes do backend, com a criação das telas de login e registro primeiramente e seguindo com as demais funcionalidades.

O diagrama mostrado na figura 1 ilustra o fluxo de desenvolvimento de cada etapa:

Figura 1 - Diagrama incremental de desenvolvimento da plataforma



Fonte: Autoria própria (2025).

Por fim, a metodologia adotada também envolveu uma etapa contínua de documentação, registrada em cada fase do projeto. Essa documentação inclui diagramas de arquitetura, descrições das entidades, explicações sobre os fluxos funcionais e justificativas das decisões técnicas, permitindo que o projeto seja compreendido em profundidade por leitores e avaliadores. A abordagem metodológica adotada demonstrou-se adequada ao garantir organização, agilidade e clareza durante o desenvolvimento da plataforma.

4. DESENVOLVIMENTO DA PLATAFORMA

O desenvolvimento da plataforma de cursos online foi realizado de forma estruturada, progressiva e alinhada com as necessidades identificadas na fase de levantamento de requisitos. Neste capítulo, apresenta-se a descrição completa da construção do sistema, passando pela definição da arquitetura, implementação do backend e do frontend, integração com o banco de dados, configuração do serviço de armazenamento em nuvem e construção dos fluxos funcionais que permitem o uso eficiente da plataforma por alunos e instrutores. Todo o processo foi conduzido com foco em desempenho, segurança, experiência do usuário e escalabilidade, de forma que a solução final atendesse às demandas de um sistema de ensino moderno.

A arquitetura adotada no projeto foi inteiramente baseada na separação entre frontend e backend, seguindo os princípios da arquitetura web moderna discutidos anteriormente. Essa separação permitiu que cada camada fosse desenvolvida de forma independente, favorecendo a modularidade e facilitando a manutenção e evolução do sistema. O backend, construído com Node.js e Express, ficou responsável pela regra de negócio, autenticação, controle de acesso e fornecimento de dados para o frontend. Já o frontend, desenvolvido em Next.js, organizou a apresentação visual, a navegação entre páginas, o consumo das APIs e a interação do usuário com a plataforma.

A organização de pastas simplificada do projeto se encontra conforme mostrado na figura 2:

Figura 2 - Estrutura simplificada do projeto

```
backend/
  ├── routes/
  ├── models/
  └── middleware/
frontend/
  ├── courses/
  ├── dashboard/
  ├── login/
  ├── my-courses/
  ├── profile/
  └── register/
```

Fonte: Autoria própria (2025).

4.1 Backend

A primeira etapa significativa do desenvolvimento foi a implementação do mecanismo de autenticação. Como a plataforma possui papéis distintos, como aluno, instrutor e administrador, tornou-se necessário criar um sistema seguro, confiável e capaz de proteger áreas sensíveis. Para isso, utilizou-se JWT armazenado em cookies HttpOnly, impedindo que scripts maliciosos acessassem o token. O backend passou a validar o cookie em todas as rotas protegidas por meio de um middleware centralizado, que verificava também o papel do usuário para definir se ele tinha permissão para visualizar ou alterar determinados conteúdos. Esse mecanismo estabeleceu uma base sólida sobre a qual todo o resto da plataforma foi construído:

Código 2: Middleware de autenticação no backend.

```
export const authMiddleware = (req: AuthRequest, res: Response, next: NextFunction)
=> {
  const token = req.cookies?.authToken;

  if (!token) {
    return res.status(401).json({ message: 'Não autorizado: Token não fornecido' });
  }

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET as string);
    req.user = decoded;
    next();
  } catch (error) {
    res.status(401).json({ message: 'Não autorizado: Token inválido' });
  }
};
```

Fonte: Arquivo authMiddleware.ts. Autoria própria (2025).

Após a autenticação, iniciou-se o desenvolvimento das funcionalidades específicas dos instrutores, já que eles são os responsáveis pela criação e gerenciamento dos cursos. O backend recebeu rotas dedicadas à criação, edição e exclusão de cursos, com validações para garantir que apenas instrutores pudessem realizar essas ações. A modelagem dos cursos foi elaborada com o objetivo de permitir uma composição flexível, em que cada curso possui diversas aulas, cada aula associada a um vídeo enviado pelo instrutor. Essa relação foi modelada no banco MongoDB utilizando arrays aninhados, o que refletiu naturalmente a organização lógica do conteúdo educacional.

O processo de upload de vídeos foi uma das partes mais importantes e também mais complexas do desenvolvimento. Hospedar arquivos de vídeo diretamente no servidor traria problemas de performance, escalabilidade e custo. Por isso, integrou-se a plataforma ao Amazon S3, permitindo que os vídeos fossem enviados diretamente do navegador para a nuvem. Esse processo começou com a criação de uma rota no backend que, ao ser acessada por um instrutor autenticado, gerava uma URL pré-assinada. Essa URL permitia que o arquivo fosse enviado diretamente para o AWS S3 sem passar pelo servidor, resultando em menor consumo de banda e muito mais eficiência. O backend não armazena o vídeo em si, mas apenas a chave (key) retornada pela AWS, que identifica o arquivo dentro do bucket. Essa chave passa a ser armazenada junto aos dados da aula no banco de dados. Dessa forma, o backend e o frontend sabem qual vídeo corresponde a qual aula sem nunca manipular o arquivo em si. Estes trechos de código mostram na prática a geração da URL pré-assinada para upload para a AWS (backend) e envio direto para o S3:

Código 3.1: Rota de upload com URL pré-assinada (Backend).

```
router.post("/", authMiddleware, async (req: AuthRequest, res) => {
  try {
    if (req.user.role !== "instrutor") {
      return res.status(403).json({ message: "Apenas instrutores podem enviar vídeos"
});
    }

    const { fileName, contentType } = req.body;

    if (!fileName || !contentType) {
      return res.status(400).json({ message: "Nome do arquivo e contentType são
obrigatórios" });
    }

    if (!contentType.startsWith("video/")) {
      return res.status(400).json({ message: "Apenas arquivos de vídeo são
permitidos" });
    }

    const key = `aulas/${Date.now()}-${fileName}`;

    const command = new PutObjectCommand({
      Bucket: getRequiredEnv("AWS_S3_BUCKET"),
      Key: key,
      ContentType: contentType,
    });

    const uploadUrl = await getSignedUrl(s3, command, { expiresIn: 3600 });
```

```
    res.json({
      uploadUrl,
      key,
    });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Erro ao gerar URL de upload", error });
  }
});
```

Fonte: Arquivo courses.ts. Autoria própria (2025).

]Código 3.2: Método do api-client que chama a rota no Backend (Frontend).

```
  static async uploadVideo(fileName: string, contentType: string) {
    return this.post<{ uploadUrl: string; key: string }>("/upload", {
      fileName,
      contentType,
    });
  }
```

Fonte: Arquivo api-client.ts. Autoria própria (2025).

A implementação dessa funcionalidade também exigiu cuidados adicionais. Por exemplo, identificou-se que enviar o vídeo automaticamente ao selecionar o arquivo poderia gerar arquivos órfãos no S3 caso o instrutor desistisse da criação do curso ou ocorresse algum erro. Assim, esse comportamento foi ajustado: o frontend agora somente solicita URLs de upload e envia vídeos quando o usuário confirma o envio explicitamente. Além disso, a exclusão de cursos passou a incluir a remoção segura dos vídeos associados, utilizando o comando de exclusão do AWS SDK, garantindo que não fossem acumulados arquivos desnecessários.

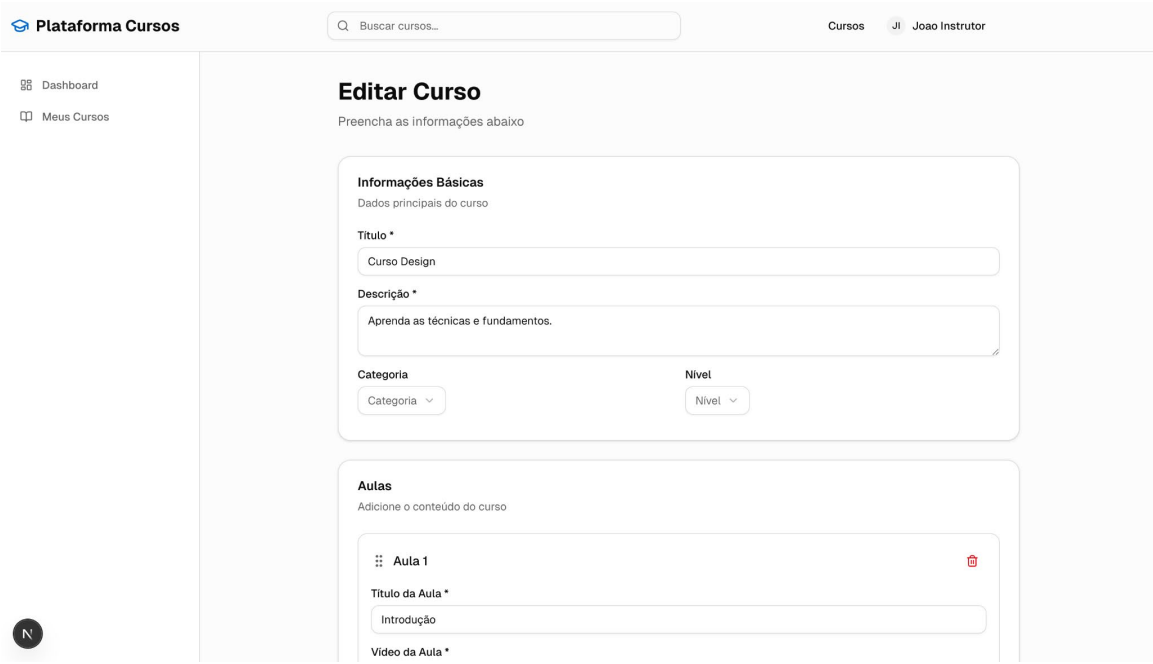
Com o backend funcionando e integrado ao S3, foi possível iniciar a construção das páginas do frontend. A interface do sistema foi projetada em Next.js utilizando o App Router, proporcionando organização clara entre componentes cliente e servidor. A página inicial apresenta informações gerais da plataforma e incentiva o usuário a realizar login. Após autenticado, se o usuário for instrutor, ele é redirecionado ao painel de controle, onde pode visualizar seus cursos existentes, criar novos ou editar os já cadastrados.

4.2 Frontend

A página de criação de cursos possui um formulário completo, contendo campos para título, descrição, categoria, nível e lista de aulas. Cada aula possui seu próprio campo para título e um componente responsável pelo envio do vídeo. Esse componente de upload foi construído com atenção especial à experiência do usuário, incluindo barra de progresso, mensagens informativas e feedback visual para erros ou sucesso. A página de edição segue a mesma interface da criação, porém pré-carregada com os dados já existentes, permitindo que o instrutor modifique qualquer parte do curso sem precisar recomeçar do zero.

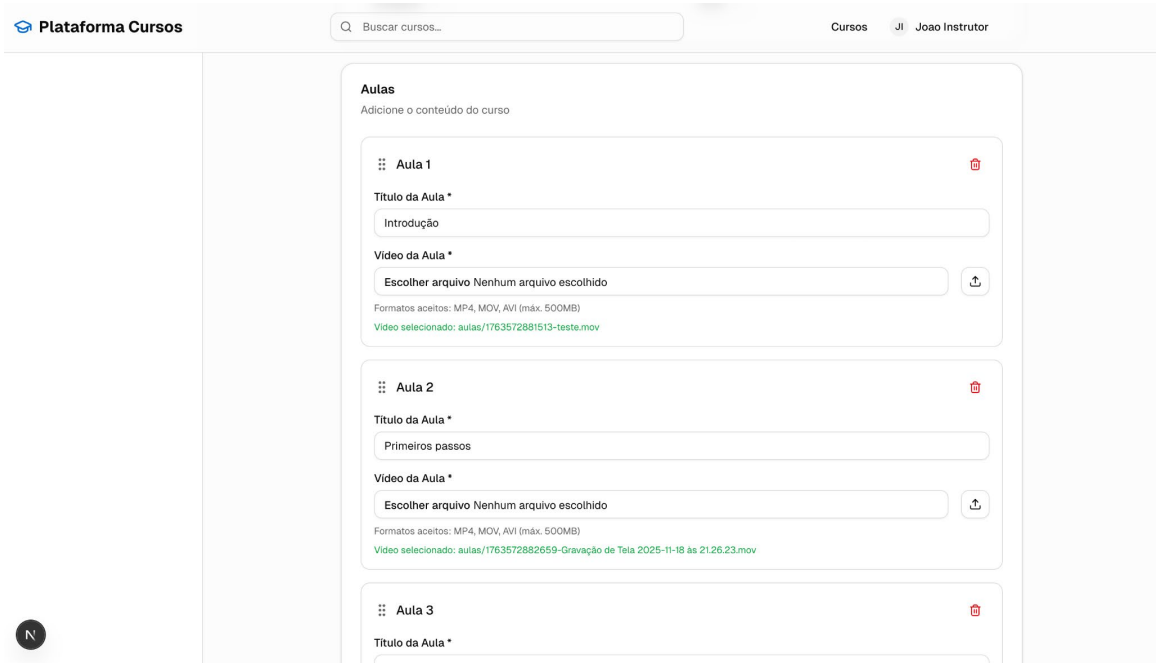
As figuras 3 e 4, a seguir, demonstram como a interface está implementada para atender às funcionalidades citadas anteriormente:

Figura 3 - Tela de edição de curso com o instrutor logado



Fonte: Coursi. Autoria própria (2025).

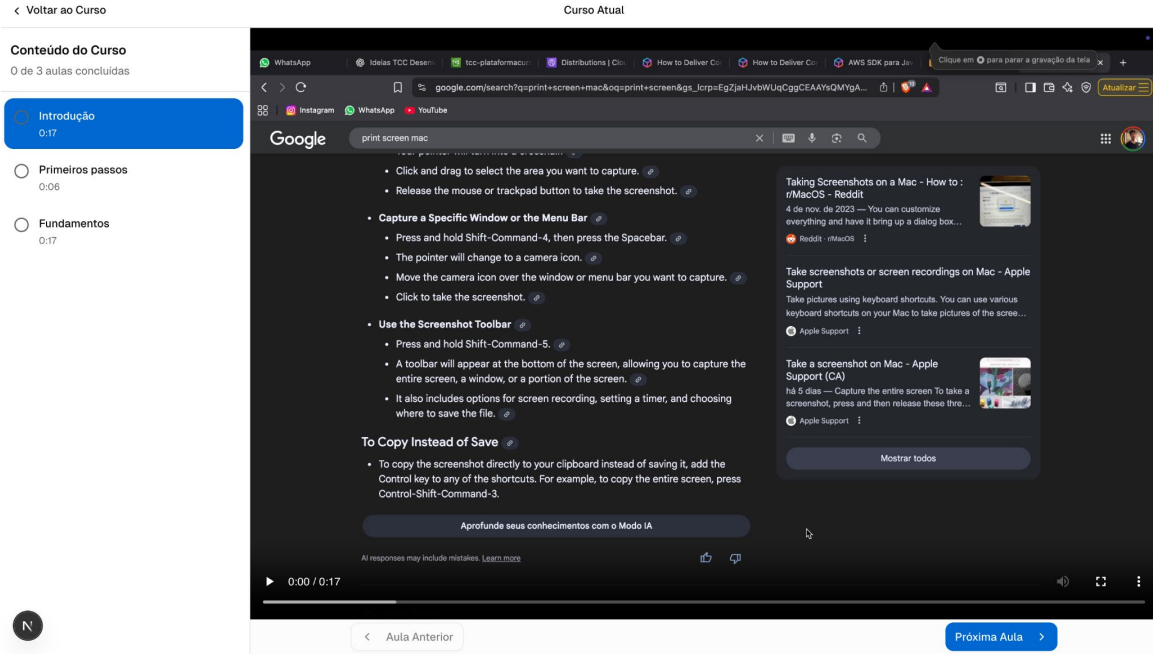
Figura 4 - Tela de edição de curso com o instrutor logado, extensão da imagem anterior



Fonte: Coursi. Autoria própria (2025).

Para os estudantes, foi criada uma página dedicada à visualização das aulas, acessível somente se o aluno estiver inscrito no curso correspondente. Quando o aluno acessa a página, o backend gera URLs temporárias para reprodução dos vídeos, garantindo que as aulas não possam ser acessadas fora da plataforma. Essa estratégia impede o download público dos vídeos e protege o conteúdo dos instrutores. A figura 5 mostra a visualização das aulas para o aluno inscrito no curso:

Figura 5 - Tela do conteúdo do curso com o aluno logado



Fonte: Print da tela “Meus Cursos”. Autoria própria (2025).

Durante todo o processo, foram realizados testes manuais e ajustes contínuos. A integração entre frontend e backend foi validada a cada nova funcionalidade. Testes com o Postman ajudaram a corrigir inconsistências nas respostas da API, enquanto os testes no navegador permitiram identificar falhas de layout, permissões incorretas ou problemas de UX. Essa abordagem incremental garantiu que cada módulo do sistema estivesse funcional antes de avançar para o próximo, resultando em maior estabilidade e menor acúmulo de erros.

Por fim, a plataforma foi documentada detalhadamente, incluindo diagramas de arquitetura, descrições de rotas, justificativas técnicas e explicações sobre decisões de design. Toda essa estrutura não apenas facilita a compreensão do sistema como também contribui para sua futura evolução, permitindo que novos desenvolvedores compreendam rapidamente sua lógica interna.

Com isso, o desenvolvimento da plataforma se consolidou como um processo disciplinado, fundamentado e tecnicamente consistente, resultando em um sistema completo, funcional e aderente às necessidades de uma plataforma moderna de cursos online.

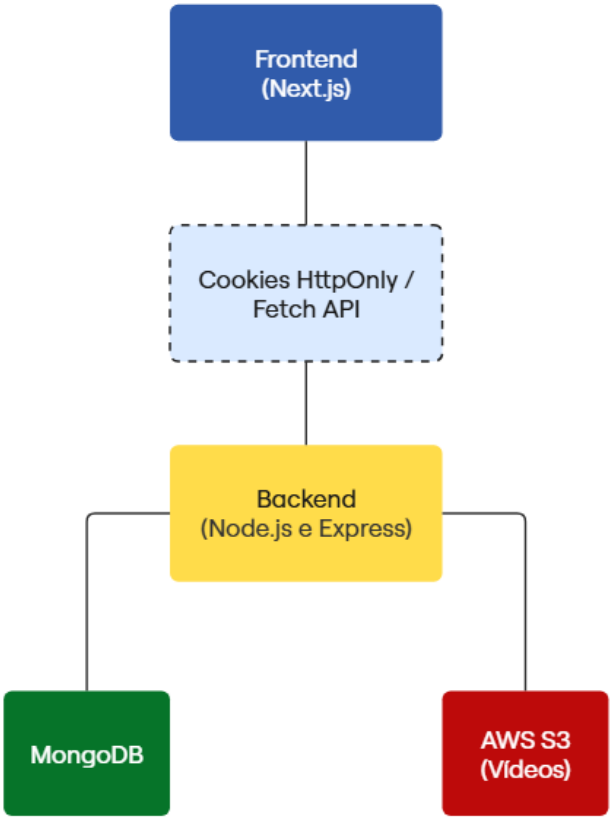
4.3 Arquitetura da plataforma

A arquitetura da plataforma foi projetada com o objetivo de garantir organização, escalabilidade, segurança e clareza na divisão de responsabilidades entre as camadas do sistema. A solução adotada combina princípios de arquitetura distribuída com padrões modernos de desenvolvimento web, resultando em um ambiente robusto para criação, gerenciamento e consumo de cursos online. A plataforma opera por meio de uma comunicação baseada em APIs REST entre o frontend, implementado em Next.js, e o backend, desenvolvido em Node.js com Express, utilizando MongoDB para persistência de dados e Amazon S3 para armazenamento de vídeos.

A decisão de separar claramente o frontend do backend não apenas facilita a manutenção e evolução da aplicação, como também permite que cada camada seja escalada de forma independente. O backend, por exemplo, pode ser ampliado para lidar com alto volume de requisições simultâneas sem afetar a camada visual, enquanto o frontend pode ser otimizado para oferecer melhor experiência ao usuário utilizando renderização híbrida (Server-Side Rendering e componentes cliente-servidor). Essa separação modular é uma característica essencial de arquiteturas modernas e contribui diretamente para a longevidade e flexibilidade do sistema.

Assim, a arquitetura final da plataforma pode ser compreendida como uma integração harmoniosa entre quatro grandes componentes: o frontend (Next.js), o backend (Node.js/Express), o banco de dados (MongoDB) e o armazenamento em nuvem (AWS S3). Cada componente desempenha um papel distinto, mas interdependente, contribuindo para uma solução escalável, segura e de fácil manutenção. A figura 6 ilustra de forma simplificada o fluxo de comunicação entre esses módulos, evidenciando as interações essenciais que sustentam o funcionamento da plataforma:

Figura 6 - Arquitetura geral da plataforma



Fonte: Autoria própria (2025).

5. RESULTADOS OBTIDOS

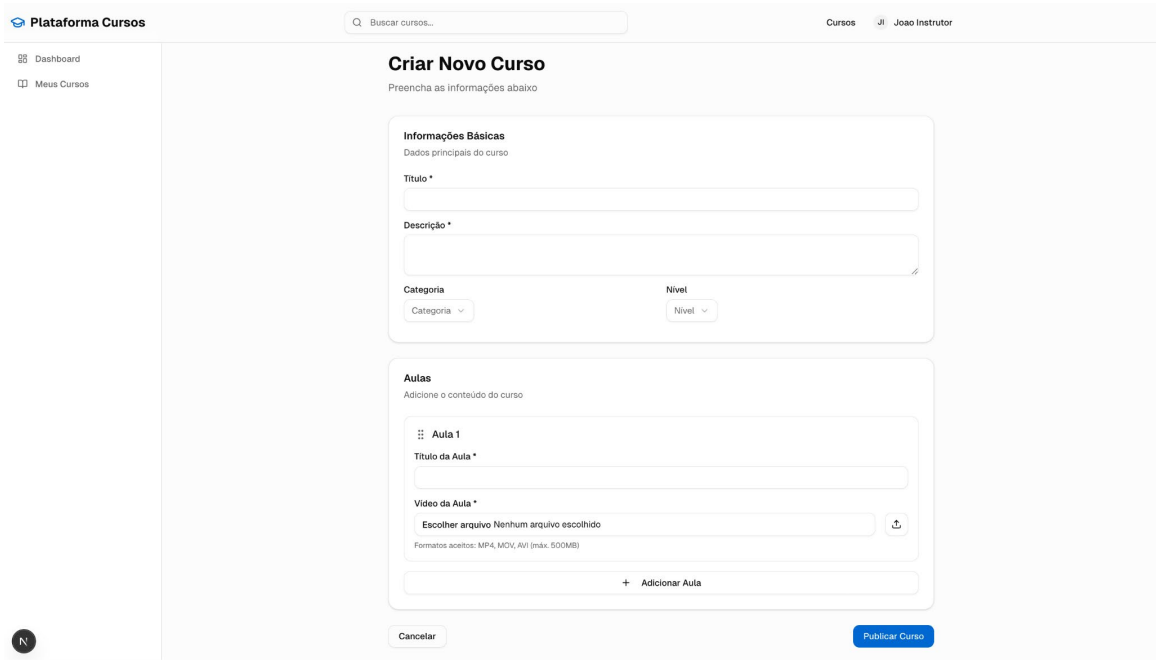
O desenvolvimento da plataforma de cursos online permitiu alcançar um conjunto significativo de resultados práticos, que demonstram tanto a viabilidade técnica da solução quanto seu potencial como ferramenta educacional. Ao longo do processo, foram realizadas implementações, testes e ajustes que possibilitaram validar o funcionamento dos principais módulos da aplicação. Este capítulo apresenta uma análise detalhada desses resultados, incluindo comparações com o cenário inicial, aspectos que funcionaram conforme o esperado, pontos que ainda requerem melhorias e uma visão geral da eficácia da plataforma em seu estado atual.

Desde o início do projeto, um dos objetivos centrais era criar uma solução moderna, segura e funcional que permitisse a instrutores gerenciarem cursos e a alunos consumirem conteúdos em vídeo de forma organizada. Durante os testes finais, verificou-se que todas as funcionalidades essenciais foram implementadas com sucesso. A autenticação utilizando JWT e cookies HttpOnly demonstrou-se eficaz, permitindo separar com clareza as permissões de alunos, instrutores e administradores. As rotas protegidas bloquearam corretamente acessos não autorizados, garantindo o isolamento das áreas sensíveis da plataforma. A interface, construída com Next.js, apresentou bom desempenho e rápida navegação entre as telas, proporcionando uma experiência fluida tanto para alunos quanto para instrutores.

Os testes demonstraram também que o fluxo de criação de cursos funciona de maneira completa e intuitiva. O instrutor pode informar título, descrição, categoria e nível, além de adicionar múltiplas aulas. Um resultado particularmente relevante foi o funcionamento correto do upload de vídeos utilizando URLs pré-assinadas da AWS. A adoção desse mecanismo trouxe benefícios imediatos, como redução de carga no backend e maior velocidade no envio de arquivos, comprovando que a escolha arquitetural foi acertada. Durante os testes de upload, verificou-se que vídeos de diferentes tamanhos puderam ser enviados sem impacto significativo no desempenho da aplicação, validando a eficiência do método.

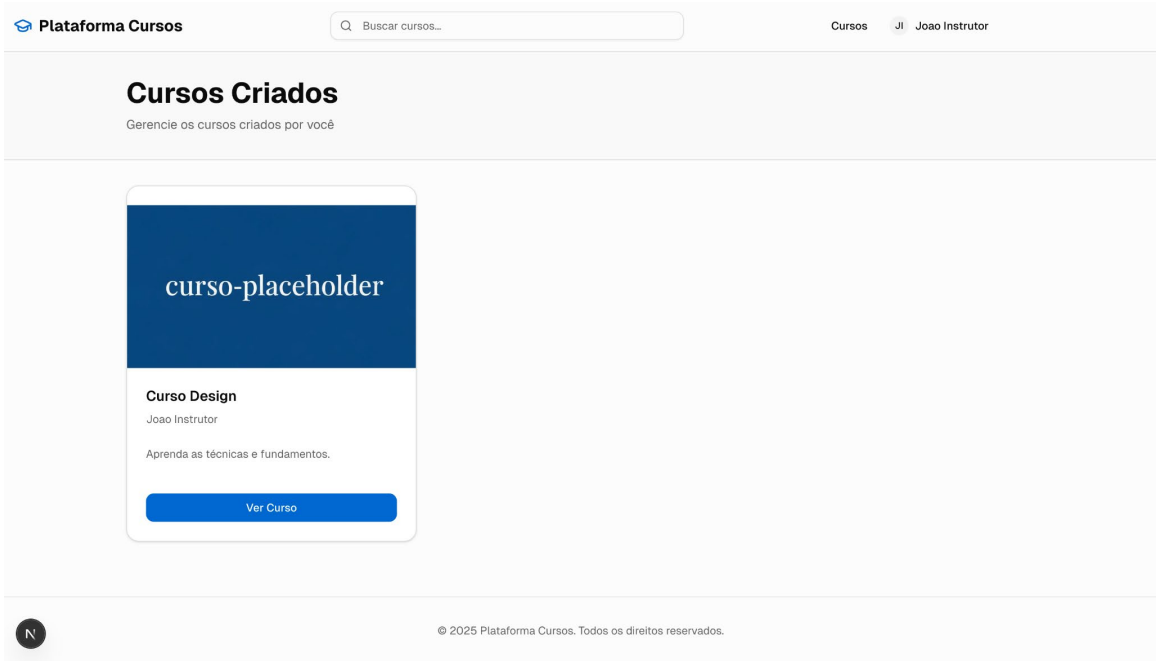
As figuras 7, 8 e 9 reforçarão visualmente a qualidade da interface e a conclusão bem-sucedida das implementações, por exemplo:

Figura 7 - Tela de criação de cursos com formulário completo, com instrutor logado



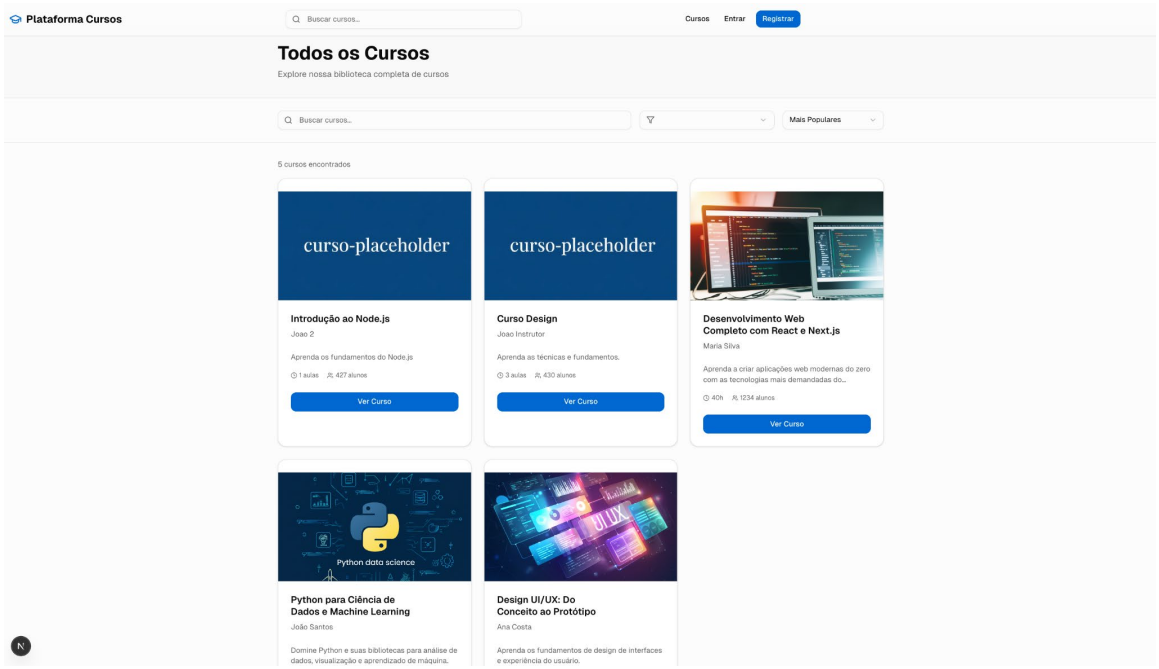
Fonte: Coursi. Autoria própria (2025).

Figura 8 - Tela de listagem de cursos do instrutor



Fonte: Coursi. Autoria própria (2025).

Figura 9 - Tela “Todos os Cursos” da plataforma



Fonte: Coursi. Autoria própria (2025).

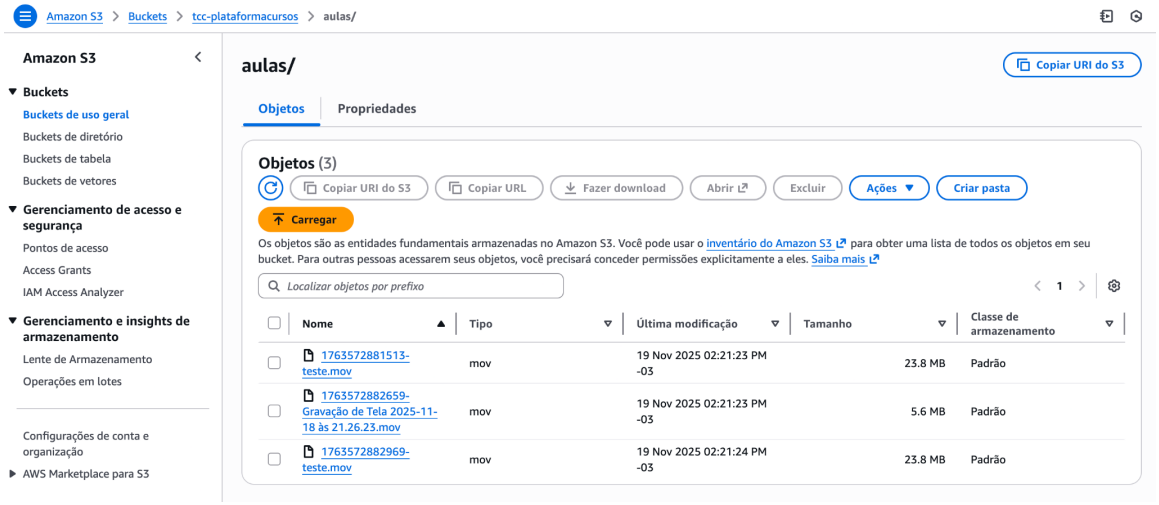
Uma comparação entre o estado inicial planejado e o sistema atual mostra que os principais requisitos foram atendidos. No início do projeto, havia apenas a definição conceitual de uma plataforma modular com três perfis de usuário e upload de vídeos. No estado atual, a plataforma inclui:

- Autenticação funcional baseada em JWT;
- Controle de acesso por papéis;
- CRUD completo de cursos;
- Upload seguro de vídeos com AWS S3;
- Exclusão de vídeos vinculada à exclusão de cursos;
- Página do aluno com player de vídeo;
- Interface responsiva e organizada.

Outro resultado relevante foi a integração bem-sucedida entre os componentes que compõem a arquitetura geral. O backend comunica-se com o banco MongoDB sem inconsistências, enquanto o frontend consome a API utilizando cookies seguros. A geração de URLs de upload e download funciona adequadamente, garantindo integridade e segurança no tráfego de dados entre os serviços.

A figura 10 mostra como os conteúdos aparecem no painel do S3, cada vídeo é um recurso em nuvem fornecido sob demanda:

Figura 10 - Painel de objetos do bucket S3



Fonte: Buckets Amazon S3 (2025).

No entanto, alguns aspectos identificados durante os testes indicam oportunidades de melhoria para expansões futuras. Uma das possíveis evoluções seria a implementação de um sistema de transcodificação de vídeos, para garantir compatibilidade universal no player e ajustar automaticamente a qualidade do arquivo. Outra sugestão é a inclusão de um sistema de progressão do aluno, permitindo acompanhar quais aulas já foram assistidas. Além disso, funcionalidades como comentários, avaliações de cursos ou fóruns internos poderiam aumentar o engajamento dos usuários, tornando a plataforma ainda mais completa.

Em termos de desempenho, a plataforma demonstrou capacidade de lidar com múltiplas operações simultâneas sem quedas perceptíveis. A renderização das páginas foi rápida e o uso de componentes client/server do Next.js contribuiu para isso. O player de vídeo, utilizando URLs temporárias do S3, funcionou sem interrupções, mesmo em conexões intermediárias. Essas observações indicam que a solução desenvolvida possui potencial para uso real e pode ser facilmente evoluída conforme a necessidade.

Em síntese, os resultados obtidos demonstram que a plataforma alcançou seus objetivos centrais e se apresentou funcional, segura e coerente com as exigências técnicas de um sistema moderno de cursos online. O conjunto de testes realizados evidencia que a implementação foi bem-sucedida e fornece uma base sólida para futuras

aprimorações. O projeto atingiu um nível de maturidade que confirma a viabilidade técnica e conceitual da proposta, finalizando esta etapa com resultados satisfatórios e alinhados ao planejamento inicial.

CONSIDERAÇÕES FINAIS

Os objetivos estabelecidos no início do projeto foram plenamente atingidos, o desenvolvimento da plataforma de cursos permitiu alcançar resultados amplos e significativos, tanto do ponto de vista técnico quanto acadêmico. Ao longo deste trabalho, foi possível aplicar de maneira prática os conceitos estudados sobre arquitetura web, desenvolvimento full stack, armazenamento em nuvem, segurança, modelagem de dados e boas práticas de engenharia de software. A construção do sistema demonstrou não apenas a viabilidade da solução proposta, mas também a eficácia das tecnologias selecionadas para atender às necessidades de um ambiente educacional moderno baseado em vídeo.

O sistema final contempla uma estrutura funcional composta por autenticação segura baseada em JWT, controle de acesso por papéis, criação e gerenciamento de cursos por instrutores, upload seguro de vídeos via URLs pré-assinadas da AWS, armazenamento eficiente no MongoDB e interface responsiva desenvolvida em Next.js. O fluxo completo desde o cadastro do instrutor, criação de cursos, envio de aulas, matrícula do aluno e visualização dos vídeos, foi implementado de forma coerente, validando a proposta inicial de construir uma plataforma robusta, escalável e alinhada às práticas utilizadas pela indústria.

Entre os pontos positivos do projeto, destaca-se a arquitetura modular adotada, que separa claramente as responsabilidades entre frontend, backend e serviços externos. Essa decisão permitiu maior organização do código, facilidade de manutenção e possibilidade de evolução futura. Outro aspecto positivo foi a integração bem-sucedida com a AWS S3, que se mostrou uma solução eficiente para o armazenamento de vídeos, eliminando a necessidade de trafegar arquivos grandes pelo backend e garantindo segurança e rapidez nos uploads. A utilização de cookies HttpOnly e permissões baseadas em papéis contribuiu significativamente para a segurança geral da plataforma.

Além disso, a adoção do Next.js proporcionou uma experiência de uso fluida, com carregamento otimizado de páginas e uma interface moderna capaz de atender às expectativas de usuários de plataformas educacionais contemporâneas. O uso de componentes reutilizáveis e organizados permitiu um desenvolvimento escalável e facilitou a manutenção do código no frontend.

Apesar dos avanços conquistados, o projeto apresenta algumas limitações inerentes tanto ao escopo quanto ao tempo disponível. Uma das principais limitações é a

ausência de um sistema de transcodificação e compressão automática dos vídeos enviados, o que implicaria maior compatibilidade entre navegadores e melhor adaptação às diferentes velocidades de conexão dos alunos. Outra limitação identificada foi a falta de um módulo de acompanhamento pedagógico, como trilhas de progresso, avaliações, métricas de engajamento ou funcionalidades de interação entre alunos e instrutores. Da mesma forma, os mecanismos de busca e categorização de cursos ainda podem ser aprimorados para proporcionar melhor navegabilidade em um cenário com grande quantidade de conteúdos.

Essas limitações, entretanto, abrem espaço para uma série de possibilidades de expansão que podem ser consideradas em trabalhos futuros. Entre elas, destaca-se a implementação de um sistema de streaming mais avançado, possivelmente integrado com serviços como AWS Elastic Transcoder ou AWS MediaConvert, permitindo entregas adaptativas (HTTP *Live Streaming*) e melhorando a experiência de visualização. Outra melhoria relevante seria a criação de funcionalidades sociais, como comentários, avaliações ou fóruns para cada curso, aumentando o engajamento dos alunos.

Também seria interessante desenvolver um painel administrativo completo, incluindo gráficos de acesso, métricas de desempenho dos cursos e ferramentas de moderação. A adição de um sistema de certificação automática, geração de PDF e registro de conclusão de módulos também ampliaria o valor pedagógico da plataforma. Por fim, integrar pagamentos online possibilitaria que instrutores comercializassem seus cursos diretamente pela plataforma, transformando o sistema em um produto prontamente utilizável no mercado.

Em síntese, o projeto alcançou seus objetivos centrais e entregou uma solução funcional, segura e sólida do ponto de vista técnico. A plataforma se demonstra totalmente capaz de evoluir e incorporar novos módulos, e sua arquitetura robusta fornece uma base apropriada para expansões futuras. Os conhecimentos adquiridos ao longo do desenvolvimento reforçam a importância de dominar tecnologias modernas, compreender boas práticas de engenharia de software e aplicar metodologias adequadas ao desenvolvimento de soluções reais e escaláveis.

REFERÊNCIAS

AMAZON WEB SERVICES. ***Amazon S3 Documentation.*** 2023. Disponível em: https://docs.aws.amazon.com/s3. Acesso em: 25 nov. 2025.

AWS. ***Amazon S3 Pre-signed URLs Documentation.*** 2023. Disponível em: https://docs.aws.amazon.com/AmazonS3/latest/userguide/ShareObjectPreSignedURL.html. Acesso em: 25 nov. 2025.

CHODOROW, K. ***MongoDB****: The Definitive Guide*. 2. ed. Sebastopol: O'Reilly, 2013.

GARRETT, J. J. ***The Elements of User Experience:*** *user-centered design for the Web and beyond*. 2. ed. Berkeley: New Riders, 2011.

MOORE, M. G.; KEARSLEY, G. ***Educação a Distância****: uma visão integrada*. 3. ed. São Paulo: Cengage Learning, 2012.

NODE.JS FOUNDATION. ***Node.js Documentation.*** 2024. Disponível em: https://nodejs.org/en/docs/. Acesso em: 25 nov. 2025.

NORMAN, D. A. ***The Design of Everyday Things.*** Revised and expanded edition. New York: Basic Books, 2013.

OWASP FOUNDATION. *OWASP* ***Top 10:*** *The Ten Most Critical Web Application Security Risks*. 2022. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 27 nov. 2025.

PAIVA, S. ***Ensino a distância frente à pandemia COVID-19***. *REEDUC*, v. 7, n. 1, 2021. Disponível em: https://www.revista.ueg.br/index.php/reeduc/article/view/11064. Acesso em: 12 nov. 2025.

RICHARDS, M. ***Software Architecture Patterns***. Sebastopol: O'Reilly Media, 2015.

SANDHU, R.; COYNE, E.; FEINSTEIN, H.; YOUMAN, C. ***Role-based access control models***. *IEEE Computer*, v. 29, n. 2, 2011.

SHANAHAN, T. ***The Node.js Event Loop:*** *Understanding Asynchronous Architecture*. San Francisco: JSBooks, 2018.

TEIXEIRA, A. ***Node.js:*** *Aplicações Web Reativas*. Rio de Janeiro: Alta Books, 2019.

TEIXEIRA, J. ***Desenvolvimento Web Moderno com Node.js*****.** São Paulo: Editora CodeMaster, 2019.

UNESCO. ***Education:*** *From disruption to recovery*. Paris, 2020.

VERCEL. ***Next.js Documentation***. 2024. Disponível em: https://nextjs.org/docs. Acesso em: 25 nov. 2025.

W3C – WORLD WIDE WEB CONSORTIUM. ***Web Content Accessibility Guidelines (WCAG) 2***. 2023. Disponível em: https://w3.org. Acesso em: 25 nov. 2025.