INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMPÁ
TECNÓLOGO EM REDES DE COMPUTADORES

ALESSANDRO RIGAMONTI DE OLIVEIRA
LUCAS MATEUS DO ESPIRITO SANTO MOTA

PLATAFORMA DE AUTOAVALIAÇÃO INSTITUCIONAL BASEADA NOS CRITÉRIOS DO SINAES

MACAPÁ – AP
2025

ALESSANDRO RIGAMONTI DE OLIVEIRA
LUCAS MATEUS DO ESPIRITO SANTO MOTA

PLATAFORMA DE AUTOAVALIAÇÃO INSTITUCIONAL BASEADA NOS CRITÉRIOS DO SINAES

Trabalho de Conclusão de Curso apresentado Coordenadoria de Curso em Redes de Computadores do Instituto Federal de Educação, Ciência e Tecnologia do Amapá – Campus Macapá, como requisito parcial para obtenção do título de Tecnólogo em Redes de Computadores.
Orientador: Prof. Me. Olavo Nylander Brito Neto

MACAPÁ – AP
2025

Biblioteca Institucional – IFAP
Dados Internacionais de Catalogação na Publicação (CIP)

O0048p Oliveira, Alessandro Rigamonti de
Plataforma de Autoavaliação Institucional Baseada Nos Critérios do SINAES / Alessandro Rigamonti de Oliveira, Lucas Mateus do Espirito Santo Mota. - Macapá, 2025.
41 f.

Trabalho de Conclusão de Curso (Graduação) -- Instituto Federal de Educação, Ciência e Tecnologia do Amapá, Campus Macapá, Tecnologia em Redes de Computadores, 2025.

Orientador: Me. Olavo Nylander Brito Neto.

1. Desenvolvimento de sistema de autoavaliação Institucional. I. Mota, Lucas Mateus do Espirito Santo. I. Neto, Me. Olavo Nylander Brito, orient. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica do IFAP com os dados fornecidos pelo(a) autor(a).

ALESSANDRO RIGAMONTI DE OLIVEIRA
LUCAS MATEUS DO ESPIRITO SANTO MOTA

PLATAFORMA DE AUTOAVALIAÇÃO INSTITUCIONAL BASEADA NOS CRITÉRIOS DO SINAES

Trabalho de Conclusão de Curso apresentado Coordenadoria de Curso em Redes de Computadores do Instituto Federal de Educação, Ciência e Tecnologia do Amapá – Campus Macapá, como requisito parcial para obtenção do título de Tecnólogo em Redes de Computadores.

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
OLAVO NYLANDER BRITO NETO
Data: 13/02/2026 16:46:36-0300
Verifique em https://validar.iti.gov.br

Prof. Me Olavo Nylander Brito Neto

gov.br Documento assinado digitalmente
CLAYTON JORDAN ESPINDOLA DO NASCIMENT
Data: 13/02/2026 16:20:03-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Clayton Jordan Espindola do Nascimento

gov.br Documento assinado digitalmente
CRISTINA COUTINHO DE OLIVEIRA
Data: 12/02/2026 19:04:02-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Cristina Coutinho de Oliveira

Apresentado em: 18/12/2025.

Conceito/Nota: 90

RESUMO

O presente trabalho aborda o desenvolvimento de um sistema de autoavaliação para os cursos superiores do Instituto Federal do Amapá (IFAP), baseado nos critérios utilizados pelo Sistema Nacional de Avaliação da Educação Superior (SINAES) do MEC. O objetivo é oferecer uma ferramenta que permita à instituição avaliar-se previamente. Assim, fica mais fácil identificar pontos de melhoria e preparar-se para as avaliações regulatórias. Para isso, o sistema foi projetado com tecnologias modernas, como React e Next.js no frontend, PostgreSQL como banco de dados e Prisma.js como ORM. A metodologia adotou as diretrizes do modelo MPS.BR para garantir qualidade e eficiência no desenvolvimento. Como resultado, a ferramenta oferece dashboards interativos e relatórios automatizados, possibilitando uma análise integrada das dimensões avaliadas: Organização Didático-Pedagógica, Corpo Docente e Tutorial, e Infraestrutura. O sistema contribui para o fortalecimento da gestão acadêmica do IFAP, alinhando-se às exigências do MEC e promovendo a melhoria contínua da qualidade educacional.

Palavras-chave: autoavaliação; educação superior; sinaes; mps.br; desenvolvimento de software.

ABSTRACT

This study presents the development of a self-assessment system for higher education programs at the Federal Institute of Amapá (IFAP), based on the criteria defined by the National System for the Evaluation of Higher Education (SINAES) of the Brazilian Ministry of Education (MEC). The objective is to provide a tool enabling the institution to evaluate itself beforehand, identifying areas for improvement and facilitating preparation for regulatory assessments. The system was designed using modern technologies, such as React and Next.js for the frontend, PostgreSQL as the database, and Prisma.js as the ORM. The methodology followed the MPS.BR model guidelines to ensure quality and efficiency in development. The resulting tool features interactive dashboards and automated reports, enabling integrated analysis of the evaluated dimensions: Pedagogical Organization, Faculty and Tutorial Staff, and Infrastructure. This system strengthens IFAP's academic management, aligning with MEC requirements and fostering continuous educational quality improvement.

Keywords: self-assessment; higher education; sinaes; mps.br; software development.

SUMÁRIO

1 INTRODUÇÃO

As políticas públicas voltadas à educação superior no Brasil seguem o princípio constitucional da garantia de padrão de qualidade, estabelecido no artigo 206, inciso VII, da Constituição Federal de 1988. Com base nesse princípio, foi criado em 2004 o Sistema Nacional de Avaliação da Educação Superior (SINAES), com a missão de contribuir para a melhoria contínua da qualidade dos cursos de graduação e das Instituições de Ensino Superior (IES).

Dentro do SINAES, os processos de reconhecimento e renovação de reconhecimento dos cursos envolvem uma avaliação estruturada em três dimensões principais: Organização Didático-Pedagógica, Corpo Docente e Tutorial, e Infraestrutura. Cada uma dessas dimensões tem um peso específico na composição do Conceito de Curso (CC), cuja escala vai de 1 a 5, sendo que notas iguais ou superiores a 3 indicam qualidade considerada satisfatória. Essa avaliação, frequentemente realizada de forma remota ou *in loco* por comissões de especialistas, busca verificar se as informações apresentadas pelos cursos correspondem à realizada observada.

Apesar da importância desse processo, muitas instituições enfrentam desafios para se organizar e atender aos critérios exigidos. Um dos principais entraves é a falta de ferramentas que facilitem a centralização e análise sistemática dos dados institucionais. Essa dificuldade também está presente no Instituto Federal do Amapá (IFAP), onde se percebe uma crescente necessidade de fortalecer a cultura de autoavaliação, especialmente orientada pelos parâmetros definidos pelo MEC.

Diante desse cenário, este Trabalho de Conclusão de Curso propõe o desenvolvimento de uma plataforma digital de autoavaliação voltada ao IFAP. A ideia é criar um sistema que permita à instituição analisar seus cursos com base nos mesmos critérios utilizados nas avaliações externas, contribuindo para identificar fragilidades, propor melhorias e se preparar melhor para os processos regulatórios. A solução conta com tecnologias modernas, como React e Next.js no frontend, PostgreSQL no backend e Prisma como ORM, garantindo desempenho, escalabilidade e facilidade de manutenção.

1.1 Problema da pesquisa

A avaliação dos cursos superiores do Instituto Federal do Amapá (IFAP) enfrenta desafios relacionados à ausência de um sistema eficiente que permita à instituição se autoavaliar com base nos critérios do MEC. Atualmente, o processo de preparação para as visitas avaliativas

é manual e descentralizado, dificultando a análise das dimensões exigidas pelo instrumento de avaliação. Isso resulta em:

a) Dificuldades na coleta e organização de dados;

b) Falta de ferramentas que permitam uma análise integrada das dimensões avaliadas;

c) Riscos de inconformidades com os padrões exigidos pelo MEC;

d) Limitações na identificação prévia de pontos de melhoria;

e) Retrabalho por parte das coordenações.

Diante desse contexto, surge a necessidade de investigar: Como o desenvolvimento de um sistema de autoavaliação baseado nos critérios do MEC pode aprimorar a gestão e preparação do IFAP para os processos avaliativos de seus cursos superiores?

1.2 Justificativa

A análise interna da organização dos cursos superiores constitui um elemento fundamental para que as IES possam identificar fragilidades e planejar ações de melhoria, tanto nos processos de acompanhamento acadêmico quanto nos ciclos de avaliação que antecedem as avaliações regulatórias. No contexto do IFAP, a criação de um sistema específico para a autoavaliação dos cursos superiores se justifica pela necessidade de promover uma preparação mais estruturada e eficiente, baseada em dados consolidados, possibilitando a preservação e a utilização do histórico avaliativo institucional.

A ausência de uma ferramenta integrada limita a capacidade da instituição de acompanhar o desempenho de suas três dimensões avaliativas (Organização Didático-Pedagógica, Corpo Docente e Tutorial, e Infraestrutura) e de implementar ações corretivas de maneira proativa. Além disso, o uso de tecnologias modernas, como React, Next.js, PostgreSQL e Prisma.js, permitirá criar um sistema escalável, responsivo e alinhado às melhores práticas de desenvolvimento de software.

Portanto, este trabalho tem relevância tanto para a melhoria da gestão acadêmica do IFAP quanto para o cumprimento das exigências regulatórias do MEC, contribuindo para a garantia da qualidade educacional dos cursos superiores da instituição junto ao Sistema Nacional de Avaliação da Educação Superior (SINAES) e Diretoria de Avaliação da Educação Superior (DAES).

1.3 Objetivo Geral

Desenvolver um sistema de autoavaliação para os cursos superiores do Instituto Federal do Amapá (IFAP), baseado nos critérios do MEC, com o objetivo de aprimorar a preparação da instituição para os processos avaliativos, junto ao Sistema nacional de Avaliação da Educação Superior (SINAES) e Diretoria de Avaliação da Educação Superior (DAES).

1.4 Objetivos Específicos

a) Analisar os critérios de avaliação do MEC e as necessidades específicas do IFAP para o reconhecimento e renovação de cursos superiores;

b) Projetar uma interface intuitiva e responsiva para o sistema, utilizando tecnologias, com foco na melhoria da experiência do usuário;

c) Implementar o backend e o banco de dados, utilizando das tecnologias que fazem os tais, que possa garantir uma integração eficiente e escalável com o frontend;

d) Desenvolver dashboards interativos e relatórios automatizados que permitam a análise integrada das dimensões avaliativas;

e) Fornecer uma ferramenta que permita ao IFAP identificar deficiências e implementar melhorias antes das avaliações do MEC em processo semiautomatizado.

2 REFERENCIAL TEÓRICO

Segundo o Relatório de Autoavaliação Institucional do IFAP.

> A autoavaliação institucional constitui um processo de análise integral que proporciona o autoconhecimento, o realinhamento, a reestruturação das ações, visando o aperfeiçoamento institucional através dos seguintes eixos: expansão dos resultados acadêmicos, o aperfeiçoamento da qualidade do ensino técnico, da pesquisa e da extensão e o combate à evasão escolar no âmbito do Ifap (Instituto Federal do Amapá, 2015, p.60).

Ainda conforme o documento, a aplicação dos preceitos instituídos pelo Sistema Nacional de Avaliação da Educação Superior (SINAES), criado pela Lei nº 10.861/2013, "tem a finalidade de analisar, oferecer subsídios, fazer recomendações, propor critérios e estratégias para a reformulação dos processos e políticas de avaliação das instituições de Ensino Superior" (Instituto Federal do Amapá, 2015).

Partindo desse contexto institucional, este trabalho adotou o modelo MPS.BR (Melhoria de Processos do Software Brasileiro) como referência no desenvolvimento do novo sistema, de forma a estruturar as etapas do processo de criação do software segundo boas práticas amplamente reconhecidas no setor. Essa integração entre as diretrizes do SINAES e as práticas do MPS.BR visa garantir não apenas a conformidade com as exigências avaliativas, mas também a entrega de uma solução tecnológica eficiente, escalável e centrada no usuário.

Integrar as boas práticas do MPS.BR ao desenvolvimento do novo sistema permitirá que o desenvolvimento do software utilize práticas ágeis e de qualidade que estão alinhadas às tecnologias modernas, como React, Next.js e PostgreSQL que foram adotadas neste trabalho. Essas tecnologias, quando aliadas ao MPS.BR, podem resultar em um sistema mais robusto e escalável, com um foco maior na experiência do usuário e na eficiência do processo de agendamento, atendendo assim às demandas crescentes do ambiente educacional.

3 METODOLOGIA

A metodologia adotada para o desenvolvimento do sistema foi estruturada com o objetivo de atender de forma integral as necessidades institucionais, promovendo a modernização dos processos atualmente utilizados na avaliação dos cursos superiores. O desenvolvimento do sistema foi organizado em quatro etapas principais: a análise de requisitos, definição do sistema, desenvolvimento em si, o teste e implementação, seguindo uma abordagem planejada e colaborativa, conforme os princípios apresentados por Pressman (2020).

3.1 Modelo de processo de software

O desenvolvimento do sistema foi orientado pelo modelo de processo em cascata, originalmente proposto por Winston W. Royce em 1970. Esse modelo se caracteriza por uma abordagem linear e sequencial, estruturada em etapas bem definidas, como análise de requisitos, projeto, implementação, testes e manutenção. Segundo Royce (1970), esse modelo é mais adequado para projetos cujos requisitos são claramente estabelecidos desde as fases iniciais.

A adoção do modelo em cascata neste trabalho também se fundamenta nas contribuições apresentadas por Pressman (2020), que destacam sua aplicabilidade em projetos que demandam maior previsibilidade, controle e estabilidade dos objetivos, especialmente em ambientes institucionais com processos previamente definidos.

O desenvolvimento se baseou em boas práticas presente nos resultados esperados na MPS-BR (Melhoria de Processo do Software Brasileiro), que fala sobre o levantamento, especificação e a validação de requisitos garantindo a qualidade e alinhamento baseado nos padrões nacionais de desenvolvimento de software. Essa combinação resultou um processo bem estruturado, fazendo com que as entregas do sistema tivessem foco completo na resolução do problema identificado.

3.2 Fases do Processo

O desenvolvimento do projeto conforme informado anteriormente foi organizado em fases, conforme o modelo cascata, com atividades especificas que visaram atender os objetivos do projeto:

3.2.1 Análise de requisitos

A análise de requisitos e uma das principais etapas no projeto, pois e nela que será

projetado, documentado e validado as necessidades do sistema, a expectativa dos usuários, e muitos outros pontos essenciais que minimizam retrabalhos ao longo do desenvolvimento, fazendo com que este atenda os objetivos propostos.

Na etapa de análise de requisitos, foi realizado uma abordagem colaborativa com parte da equipe da instituição, com objetivo de compreender de forma sistemática as demandas e limitações relacionadas ao processo de avaliação dos cursos superiores. O primeiro passo foi identificar o principal problema enfrentado pela instituição, que seria a forma como é avaliado atualmente os cursos superiores no Instituto Federal do Amapá (IFAP) pelo Ministério da Educação e da Cultura (MEC), onde verificou-se que o processo é conduzido predominantemente por meio de planilhas, documentos digitais dispersos e em alguns casos registros em formulários físicos, o que torna lento e suscetível a erros. A partir deste diagnóstico foi possível mapear os pontos críticos a serem melhorados com a implementação do sistema.

Com base em todas as informações foi realizado várias reuniões em que foi definido os requisitos funcionais e não funcionais da plataforma, sempre tendo como prioridade a necessidade principal da instituição, fazendo possível a construção de requisitos claros, objetivos e viáveis, que serviram como base nas demais etapas do projeto. Com essa abordagem pratica e colaborativa foi possível garantir que o sistema desenvolvido fosse adequado para atender à "dor" da instituição, melhorado e modernizando o processo que atualmente estava ultrapassado.

3.2.2 Requisitos do Sistema

Os requisitos do sistema verificados foram definidos para garantir que as necessidades da instituição sejam completamente atendidas, modernizando e melhorando o funcionamento atual. Esses requisitos foram divididos em requisitos funcionais e não funcionais.

Segundo Wiegers e Beatty (2013), os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer, enquanto os requisitos não funcionais estabelecem critérios relacionados ao desempenho e à qualidade do sistema, como eficiência e segurança, sendo essenciais para a experiência do usuário.

Com base na análise de requisitos realizada, os requisitos do sistema foram organizados em duas categorias principais: requisitos funcionais e requisitos não funcionais. Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer para atender as necessidades dos usuários e da instituição, enquanto os requisitos não funcionais estabelecem critérios relacionados a qualidade, desempenho, segurança e manutenibilidade do sistema. As tabelas A e B apresentam, respectivamente, os requisitos funcionais:

Tabela 1- Requisitos Funcionais

Tabela A	***REQUISITOS FUNCIONAIS***	
Código	Requisito	Descrição
RF01	Gestão de Anexos	Permitir o envio de anexos pelos coordenadores.
RF02	Indicadores de Avaliação	Permitir a visualização e adição de informações aos indicadores.
RF03	Controle de Visualização	Restringir o acesso de usuários comuns às avaliações
RF04	Autenticação e Acesso	Garantir controle de acesso conforme o perfil do usuário.
RF05	Gerenciamento Administrativo	Permitir o gerenciamento das entidades do sistema.
RF06	Avaliação de Indicadores	Permitir a atribuição de notas e justificativas.
RF07	Avaliação Indireta	Registrar avaliações administrativas indiretas.
RF08	Perfil Acadêmico	Permitir atualização de informações acadêmicas.
RF09	Validação de Dados	Garantir a validação das informações inseridas.
RF10	Geração de Relatórios	Gerar relatórios resumidos das avaliações.
RF11	Sistema de Notificações	Notificar usuários sobre eventos do sistema.

Fonte: Elaborado pelos autores (2025)

Tabela 2 - Requisitos não funcionais

Tabela B	***Requisitos Não Funcionais***	
Código	Requisito	Descrição
RNF01	Responsividade	Compatibilidade com diferentes dispositivos e resoluções de tela.
RNF02	Usabilidade	Interface intuitiva e de fácil navegação.
RNF03	Segurança de Dados	Proteção de confidencialidade e integridade das informações.
RNF04	Performance	Desempenho adequado em acessos simultâneos.
RNF05	Escalabilidade	Suporte ao crescimento de usuários e dados.
RNF06	Manutenibilidade	Facilidade de manutenção e evolução do sistema.

Fonte: Elaborado pelos autores (2025)

3.2.2.1 Casos de uso do sistema

A modelagem dos requisitos do sistema foi realizada por meio da Linguagem de Modelagem Unificada (UML), amplamente utilizada na engenharia de software para a representação e documentação de sistemas. Entre os artefatos da UML, os casos de uso foram adotados para descrever as funcionalidades do sistema a partir da interação entre os atores e o sistema, contribuindo para a identificação e organização dos requisitos funcionais relacionados ao processo de autoavaliação dos cursos.

Os casos de uso foram elaborados com base nos requisitos funcionais identificados e nas

necessidades específicas do IFAP para o processo de autoavaliação. O sistema apresenta três perfis de usuários: Coordenadores, Administradores e Visualizadores, cada um com permissões e funcionalidades específicas.

3.2.2.1.1 Atores do sistema

O coordenador de curso tem a responsabilidade de avaliar os indicadores conforme os critérios do SINAES, ele também gerencia anexos e documentos comprobatórios (PDF, DOCX, JPG) dos indicadores e fornece descrições detalhadas para cada curso vinculado, de acordo com a dimensão avaliativa. Além disso ele pode utilizar um painel específico para poder acompanhar os andamentos das avaliações do curso, ele também deve garantir que suas informações pessoais estejam corretamente preenchidas no sistema, como nome, e-mail, foto de perfil, e dados acadêmicos (formação, vínculo de trabalho, experiência).

O administrador tem um papel mais geral e controlado no sistema. Ele é responsável por gerenciar e cadastrar as dimensões e seus indicadores que vão ser avaliados pelos coordenadores, atribuindo notas (de 0 a 5) e fornecendo justificativas para cada avaliação. Além disso, o administrador gerencia os usuários (coordenadores e visualizadores), cursos e configurações do sistema, além de gerar relatórios consolidados por curso e por dimensão. Ele também tem a função de controlar permissões e acessar todas as funcionalidades do sistema.

O visualizador é um usuário com permissões restritas. Ele pode consultar as avaliações já realizadas, mas não tem permissão para editar os dados. O visualizador tem acesso aos relatórios consolidados, mas apenas em formato somente leitura. Ele também pode visualizar as estatísticas gerais do processo de autoavaliação e, quando autorizado, exportar relatórios em formato PDF.

3.2.2.1.1.1 UC01 - Autenticação e controle de acesso

Tabela 3- Autenticação e controle de acesso

Ator	Todos os usuários.
Descrição	Sistema de login seguro, com diferentes níveis de permissão para cada tipo de usuário.
Pré-condições	O usuário deve possuir credenciais válidas no sistema.
Fluxo Principal	O usuário acessa a plataforma e visualiza a tela de login após isso o sistema oferece autenticação via credenciais após a validação das credenciais ele gera o token JWT com as informações do perfil do usuário e redireciona o usuário para a dashboard principal que mostra os cards com base no cargo

Pós-condições	O usuário é autenticado com sessão ativa e tem acesso apenas às funcionalidades permitidas pelo seu perfil.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.2 UC02 - Gestão de indicadores pelos coordenadores

Tabela 4 - Gestão de indicadores pelos coordenadores

Ator:	Coordenador.
Descrição:	O coordenador pode avaliar e gerenciar suas avalições nos indicadores do SINAES com as devidas documentações
Pré-condições:	O coordenador deve estar autenticado e vinculado a um curso.
Fluxo Principal:	O coordenador acessa o dashboard e seleciona a opção "Avaliações", após isso o sistema lista os indicadores por dimensão (Didático-Pedagógica, Corpo Docente, Infraestrutura), o coordenador seleciona o indicador específico para preenchimento, após isso o sistema exibe um formulário dinâmico com validação fazendo o coordenador preencher a descrição detalhada e fazendo o upload dos anexos fazendo o sistema validar os dados e os arquivos antes de salvar
Fluxo Alternativo:	Se a validação falhar, o sistema exibe mensagens de erro específicas.
Pós-condições:	O indicador é cadastrado e está disponível para avaliação pelos administradores.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.3 UC03 - Upload de arquivos comprovatórios

Tabela 5 - Upload de arquivos comprovatórios

Ator:	Coordenador.
Descrição:	O coordenador pode fazer upload de documentos comprobatórios relacionados aos indicadores.
Pré-condições:	O coordenador está no processo de cadastro de um indicador.
Fluxo Principal:	O coordenador clica na opção "Anexar Documento" no formulário de indicadores, após isso o sistema exibe uma interface de upload, permitindo arrastar e soltar os arquivos, no momento de envio o sistema válido a extensão (PDF, DOCX, JPG), o tamanho e o tipo MIME dos arquivos e logo em seguida se for valido ele exibe o progresso do upload em tempo real e no final o arquivo é armazenado com nome único e indexado no banco de dados.
Fluxo Alternativo:	Se o arquivo for inválido (tipo ou tamanho não permitido), o sistema rejeita e solicita um arquivo correto.
Pós-condições:	O arquivo é anexado ao indicador e armazenado corretamente para futuras consultas.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.4 UC04 - Avaliação de indicadores pelos administradores

Tabela 6 - Avaliação de indicadores pelos administradores

Ator:	Administrador
Descrição:	O administrador avalia os indicadores cadastrados pelos coordenadores, atribuindo notas e justificativas.
Pré-condições:	Os indicadores devem estar cadastrados pelos coordenadores.
Fluxo Principal:	O administrador acessa o dashboard e visualiza os indicadores pendentes para avaliação, em seguida o sistema lista os indicadores por curso, status e dimensão avaliativa, após isso o administrador seleciona o indicador para avaliação e insere a nota (0-5) e justificativa obrigatória, o sistema valida a consistência dos dados e salva a avaliação, e no final envia uma notificação ao coordenador sobre a avaliação concluída.
Pós-condições	O indicador é avaliado, com nota e justificativa registradas no sistema.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.5 UC05 - Geração de relatórios consolidados

Tabela 7 - Geração de relatórios consolidados

Ator:	Administrador, Visualizador, Coordenador
Descrição:	O sistema permite a criação automática de relatórios por curso com métricas e análises.
Pré-condições:	Avaliações concluídas para pelo menos um curso.
Fluxo Principal:	O usuário autorizado acessa o módulo de relatórios, O sistema apresenta filtros (curso, período, dimensão) após isso o usuário configura parâmetros do relatório, o sistema processa dados e calcula médias por dimensão, gera o relatório em formato PDF com gráficos e tabelas e por fim o usuário pode visualizar online ou fazer download.
Pós-condições:	Relatório gerado e disponibilizado para acesso.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.6 UC06 - Dashboard interativo e estatísticas

Tabela 8 - Dashboard interativo e estatísticas

Ator:	Todos os usuários autenticados
Descrição:	O painel de controle exibe métricas em tempo real e indicadores de progresso.
Pré-condições:	O usuário está autenticado no sistema.
Fluxo Principal:	O usuário acessa o dashboard após login, o sistema carrega métricas específicas

	baseadas no perfil do usuário, o dashboard exibe progresso por dimensão, estatísticas gerais e ações recentes, o sistema atualiza dados automaticamente em intervalos regulares e por fim o usuário pode filtrar informações por período e curso.
Pós-condições:	Dashboard carregado com informações atualizadas.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.7 UC07 - Sistema de notificações

Tabela 9 - Sistema de notificações

Ator:	Coordenador, Administrador
Descrição:	O sistema envia alertas para coordenadores e administradores sobre eventos importantes.
Pré-condições:	O usuário está autenticado e eventos relevantes ocorrem no sistema.
Fluxo Principal:	O sistema monitora eventos importantes (novas avaliações, prazos próximos), envia logo em seguida as notificações in-app, O usuário visualiza notificações no header da aplicação e pode acessar o link relacionado à notificação.
Pós-condições:	Usuário informado sobre eventos relevantes.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.8 UC08 - Gestão de usuários e permissões

Tabela 10 - Gestão de usuários e permissões

Ator:	Administrador
Descrição:	O administrador gerencia usuários, cursos e permissões no sistema.
Pré-condições:	O administrador está autenticado com permissões de gestão.
Fluxo Principal:	O administrador acessa painel de gestão de usuários, O sistema lista usuários com filtros por cargo e status, O administrador pode criar, editar ou desativar usuários, O sistema valida hierarquia de permissões e registra auditoria. O sistema envia e-mail de boas-vindas para novos usuários. O sistema registra log de auditoria para todas as ações realizadas.
Pós-condições:	Usuário gerenciado conforme ação realizada e log registrado.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.9 UC09 - Busca e filtros avançados

Tabela 11 - Busca e filtros avançados

Ator:	Todos os usuários autenticados
Descrição:	O sistema permite buscar indicadores, avaliações e relatórios com filtros avançados.
Pré-condições:	Existem dados cadastrados no sistema.
Fluxo Principal:	O usuário acessa funcionalidade de busca, o sistema apresenta opções de filtros (curso, dimensão, status, período), o usuário define critérios de busca, o sistema processa consulta e retorna resultados paginados, o usuário pode refinar busca ou acessar itens encontrados.
Pós-condições:	Resultados da busca exibidos conforme critérios definidos.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.1.10 UC10 - Exportação de dados

Tabela 12 - Exportação de dados

Ator:	Administrador, Visualizador
Descrição:	O sistema permite exportar dados em diferentes formatos para análise externa.
Pré-condições:	O usuário possui permissões de exportação.
Fluxo Principal:	O usuário acessa funcionalidade de exportação, o sistema apresenta opções de formato (PDF, Excel, CSV), logo em seguida o usuário seleciona dados a serem exportados e formato desejado, o sistema processa dados e gera arquivo e disponibiliza download do arquivo gerado.
Pós-condições:	Dados exportados no formato solicitado.

Fonte: Elaborado pelos autores (2025)

3.2.2.1.2 Regras de negócio

Tabela 13 - Regras de negócio

Regra	**Descrição**
RN01 - Controle de Acesso por Perfil	O controle de acesso por perfil está dividido em 3 cargos sendo eles: os Coordenadores que só podem gerenciar indicadores de seus próprios cursos, os administradores que têm acesso completo a todos os dados e funções do sistema e os visualizadores têm acesso somente leitura, podendo consultar as avaliações e relatórios, mas não editar nada.
RN02 - Validação de Indicadores SINAES	A validação dos indicadores deve seguir os critérios oficiais do SINAES, garantindo conformidade com as exigências do MEC, cada dimensão deve ter indicadores obrigatórios preenchidos e os anexos são obrigatórios para comprovação de

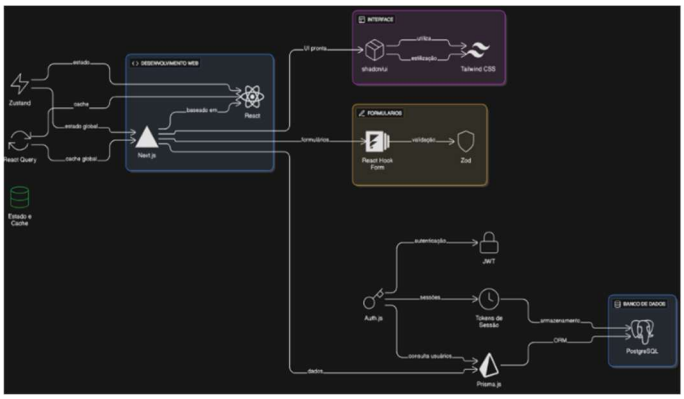
	indicadores, e o sistema deve garantir que isso seja feito.
RN03 - Sistema de Avaliação	As regras do sistema de avaliações determinam que as notas devem estar na escala de 0 a 5, conforme os critérios definidos para a avaliação, as justificativas são obrigatórias para todas as avaliações feitas pelos administradores e apenas administradores podem alterar avaliações concluídas, garantindo controle sobre o processo de avaliação.
RN04 - Auditoria e Rastreabilidade	A auditoria e a rastreabilidade são feita baseada em logs e todas as ações administrativas (como criação, edição ou exclusão de dados) são registradas para garantir rastreabilidade. As Alterações nas avaliações geram notificações para os coordenadores e administradores responsáveis e o histórico completo de modificações é mantido para garantir a transparência no processo.
RN05 - Segurança e Privacidade	A segurança e a privacidade dos Dados sensíveis, como informações pessoais e avaliações, são criptografados para garantir a segurança da plataforma e as sessões expiram automaticamente após um período de inatividade, para proteger a conta dos usuários.

Fonte: Elaborado pelos autores (2025)

3.2.3 Design técnico

Nesta fase, foram cuidadosamente avaliados diversos aspectos técnicos e funcionais, visando garantir a robustez, escalabilidade, segurança e usabilidade da plataforma. A seleção das tecnologias e a concepção da estrutura do sistema foram pautadas pela necessidade de atender aos requisitos identificados, ao mesmo tempo em que se buscava otimizar o desempenho e facilitar a manutenção futura. Na figura 1, o diagrama técnico apresentado foi elaborado pelos autores utilizando a ferramenta Eraser.io, a qual possibilita a visualização todo o fluxo e as integrações entre as tecnologias escolhidas.

Figura 1- Design técnico



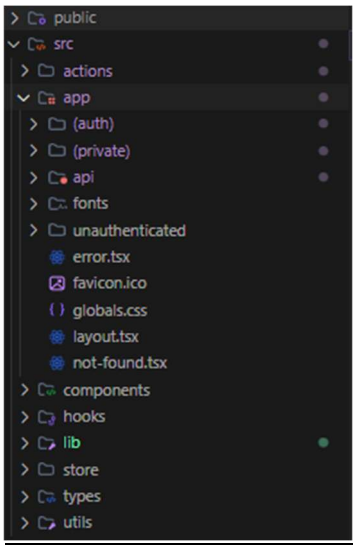
Fonte: Elaborado pelos autores (2025).

3.2.3.1 Desenvolvimento web – front end

Para o desenvolvimento do front end foi utilizada a biblioteca React, mantida pela Meta Platforms, Inc., amplamente empregada na construção de interfaces de usuário dinâmicas e baseadas em componentes, o que favorece a manutenção e a escalabilidade do código (META PLATAFORMS, 2022).

Junto do react foi o utilizado o Next.js que é um framework de código aberto baseado no React, que foi escolhido por oferecer algumas funcionalidades importantes para o desenvolvimento web. Ele permite fazer a renderização do lado do servidor (*Server Side Rendering – SSR*) e também a geração de sites estáticos (*Static Site Generation – SSG*), o que ajuda bastante na questão de performance, tempo de carregamento e na otimização para os mecanismos de busca, conhecido como (*Search Engine Optimization – SEO*) (VERCEL, s.d). A utilização dele foi pensada justamente para ter uma interface que seja mais dinâmica, responsiva e que atenda melhor os usuários. Além disso, a versão 14 do Next.js trouxe uma nova funcionalidade chamada *server actions*, que faz com que o frontend consiga se comunicar de forma mais direta com o backend, simplificando a manipulação dos dados dentro da própria aplicação e facilitando muito o desenvolvimento, já que não precisa criar uma api separada para o back-end. Na figura 2 a organização das pastas com os componentes divididos.

Figura 2 Estrutura de pastas do projeto



Autor: Elaborado pelos autores (2025)

3.2.3.2 Banco de dados

Como banco de dados foi utilizado o PostgreSQL que é um sistema de gerenciamento

de banco de dados relacional de código aberto, amplamente utilizado devido à sua robustez, confiabilidade e capacidade de lidar com grandes volumes de dados. Esse sistema é frequentemente escolhido por instituições que buscam uma alternativa com um conjunto abrangente de funcionalidades e desempenho comparável a bancos de dados proprietários. Segundo Momjian (2001), o PostgreSQL se consolidou como um banco de dados relacional voltado a pessoas que buscam uma solução de código aberto com um conjunto abrangente de recursos e desempenho comparável aos sistemas proprietários. Sua arquitetura suporta transações ACID, o que o torna adequado para aplicações complexas e de missão crítica. A escolha dele se deu principalmente porque a instituição já o utiliza, sendo esse um dos requisitos solicitados por ela.

E para criar uma estrutura solida, e padronizada do banco de dados foi utilizado o Prisma.js que é um ORM (Object-Relational Mapping, em português: Mapeamento objeto-relacional) que foi escolhido por melhorar e simplificar a interação entre a aplicação e o banco de dados. Esse ORM permite com que fazer consultas e manipulações de dados sejam de uma forma intuitiva e padronizada fazendo assim que a estrutura do banco seja menos propensa a erros e reduzindo a necessidade de escrever o SQL manualmente. A combinação entre o PostgreSQL e o Prisma.js oferece uma infraestrutura que além de ser sólida e eficiente para o desenvolvimento de aplicações modernas, proporciona uma maior flexibilidade e escalabilidade na manipulação de dados.

A combinação entre PostgreSQL e Prisma entrega uma infraestrutura sólida, escalável e eficiente. Na figura 3 pode ser visto uma parte inicial do código onde mostra a base do prisma e o modelo de Session que vai criar no banco de dados a tabela session junto das suas propriedades.

Figura 3 – Código inicial do Prisma.js com modelo de tabela Session

Fonte: Elaborado pelos autores (2025).

3.2.3.3 Construção de componentes e estilização

Para a estilização do sistema, foi utilizado o Tailwind CSS, um framework CSS utilitário mantido pela empresa Tailwind Labs, que possibilita a aplicação de estilos diretamente nos elementos da interface por meio de classes utilitárias. Essa abordagem contribui para uma estilização mais ágil e consistente, favorecendo o desenvolvimento de interfaces responsivas e flexíveis, além de reduzir a necessidade de arquivos CSS extensos. A alta capacidade de personalização e o baixo nível de abstração do framework permitem maior controle sobre o layout e o comportamento visual da aplicação (TAILWIND LABS, s.d.).

Com a escolha do tailwind foi verificado também a biblioteca Shadcn/ui que é uma biblioteca de componentes estilizados que utiliza dele e que acelera o desenvolvimento da interface, por permitir maior consistência visual, e inclui vários componentes prontos para uso, fazendo assim com que a necessidade de estilização manual durante o desenvolvimento seja reduzida, aumentando consequentemente a velocidade de desenvolvimento. Conforme descrito em sua documentação, que fala que o shadcn/ui é um conjunto de componentes acessíveis e lindamente projetados. (Shadcn, s.d.). Esse conjunto de ferramentas agilizou significativamente o desenvolvimento da interface, sendo na Figura 4 uma demonstração do componente Card feito pelo shadncn/ui utilizando o tailwind.

Figura 4 - Código do componente Card utilizando Shadcn/UI e Tailwind CSS

```
import * as React from "react"      Alessandro, há 8 meses • feat: initial proj

import { cn } from "@/lib/utils"

const Card = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn(
      "rounded-xl border bg-card text-card-foreground shadow",
      className
    )}
    {...props}
  />
))
Card.displayName = "Card"
```

Fonte: Elaborado pelos autores (2025)

3.2.3.4 Formulários e validação de dados

Para a criação dos formulários foi escolhido duas bibliotecas uma sendo o React Hook Form que melhora a criação e o gerenciamento de formulários pegando os dados e gerenciando de forma eficiente. No sistema foi usado como base para criar todos os formulários e os inputs de forma dinâmica e eficiente, reduzindo a complexidade e melhorando a usabilidade. (React Hook Form, s.d.). Foi utilizado junto de um componente dinâmico que facilita ainda mais a utilização sendo o componente mostrado na figura 5 e sua implementação na figura 6.

Figura 5 - Implementação do componente do React Hook Form

Fonte: Elaborado pelos autores (2025)

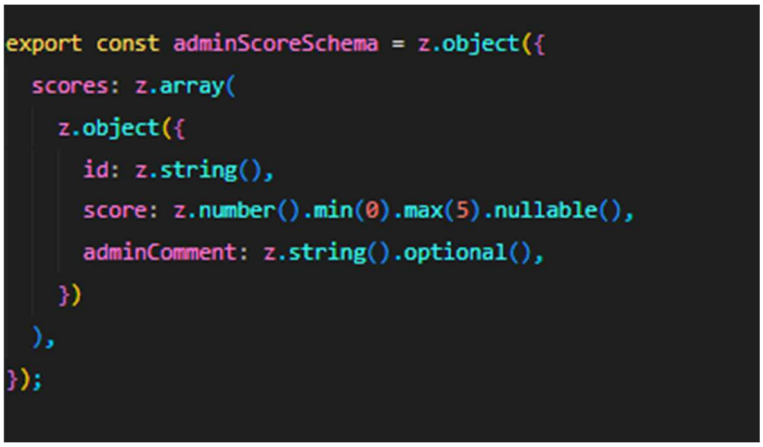
Figura 6 - Código do componente de formulário utilizando React Hook Form

```
<div className="grid grid-cols-3 gap-4">
  <div className="col-span-1">
    <InputForm
      type="number"
      label="Nota"
      name={`scores.${globalIndex}.score`}
      placeholder="0-5"
      className="w-full"
    />
  </div>
  <div className="col-span-3">
    <InputArea
      label="Comentário do Administrador"
      name={`scores.${globalIndex}.adminComment`}
      placeholder="Adicione seu comentário"
    />
  </div>
```

Fonte: Elaborado pelos autores (2025)

Junto do react hook form foi utilizado o Zod que é uma ferramenta/biblioteca que server para validar os *schemas* (estrutura e tipo dos dados) fazendo com que junto do React Hook Form, garanta a precisão e confiabilidade dos dados que são essenciais para a integridade do sistema (Zod, 2024), e durante o processo foi fundamental para a padronização das estruturas e dados. Essa validação pode ser observada na figura 7 onde demonstra uma parte do código onde foi utilizada.

Figura 7- Exemplo de validação utilizando o Zod com React Hook Form



```
export const adminScoreSchema = z.object({
  scores: z.array(
    z.object({
      id: z.string(),
      score: z.number().min(0).max(5).nullable(),
      adminComment: z.string().optional(),
    })
  ),
});
```

Fonte: Elaborado pelos autores (2025).

3.2.3.5 Gerenciamento de estado e cacheamento de dados

O gerenciamento de estado e cache foi pensado para garantir alta performance e eficiência no carregamento dos dados, nisso foram utilizadas duas ferramentas principais o Zustand que é uma solução que foi escolhida para gerenciar os estados globais da aplicação, utilizada de forma que para controlar sessões, configurações e dados. Por ele ser simples de ser implementado ele permite fazer rápidas atualizações no código, alinhando-se ao requisito de usabilidade do sistema (Zustand, 2024). Nas figuras 8 e 9 que estão logo abaixo pode ser observado a sua estrutura de como foi feito e chamado.

Figura 8 - Implementação do zustand

Fonte: Elaborado pelos autores (2025)

Figura 9 - Estrutura básica do zustand

Fonte: Elaborado pelos autores (2025)

Outra ferramenta adotada no desenvolvimento foi o TanStack Query (React Query), uma biblioteca voltada ao gerenciamento de dados assíncronos em aplicações React. Essa biblioteca oferece mecanismos automáticos de cache, sincronização e atualização de dados provenientes de requisições ao servidor, contribuindo para a redução do número de chamadas redundantes e para a melhoria do desempenho da aplicação. Sua utilização é especialmente recomendada em componentes que manipulam listas de dados, uma vez que favorece maior responsividade da interface e uma experiência de uso mais eficiente (TANSTACK, s.d.).

Na Figura 10, é apresentada a integração da biblioteca na aplicação, evidenciando sua atuação na gestão e no cache dos dados.

Figura 10 - Integração React Query

```
export default function CoursesList({ data: courses }: { data: CoursesWithType[] }) {
  const queryClient = useQueryClient()
  const { toast } = useToast()
  const { data } = useQuery({
    initialData: courses,
    queryKey: ["courses"],
    queryFn: () => getAllCourses(),
  });
  const { mutate: deleteQuery } = useMutation({
    mutationFn: async (id: string) => await deleteCourses(id),
    onSuccess() {
      queryClient.invalidateQueries({ queryKey: ["courses"], })
      toast({ title: "Sucesso", description: "Item deletado", "className": "bg-green-500 text-white" });
    },
    onError(le) {
      console.log(le)
    }
  });
```

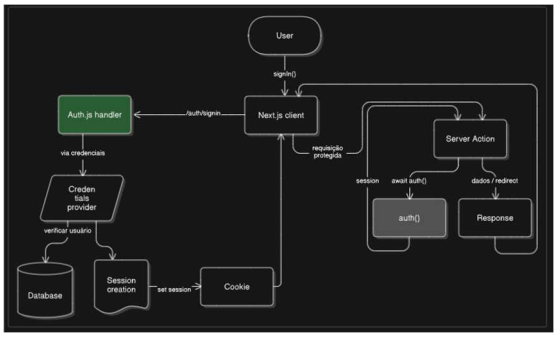
Fonte: Elaborado pelos autores (2025)

3.2.3.6 Autenticação e autorização: auth.js

Para o controle de autenticação e autorização, o sistema utiliza a biblioteca Auth.js, uma solução baseada em APIs Web padrão oferecem mecanismos seguros e flexíveis para o gerenciamento de autenticação em aplicações web. A biblioteca possibilita a autenticação por meio de credenciais locais, como usuário e senha, bem como por provedores externos, a exemplo do Google, utilizando protocolos como OAuth 2.0. Além disso, o Auth.js disponibiliza recursos para o gerenciamento de sessões e para o controle de permissões conforme o perfil dos usuários, como administradores, coordenadores e visualizadores (AUTHJS, s.d.).

Vale destacar que a versão 5.x do Auth.js possui integração aprimorada com as servers actions do Next.js, simplificando significativamente a gestão de sessões e autenticações, como mostrado na figura 11.

Figura 11- Fluxo de Autenticação do Auth.JS



Fonte: Elaborado pelos autores (2025)

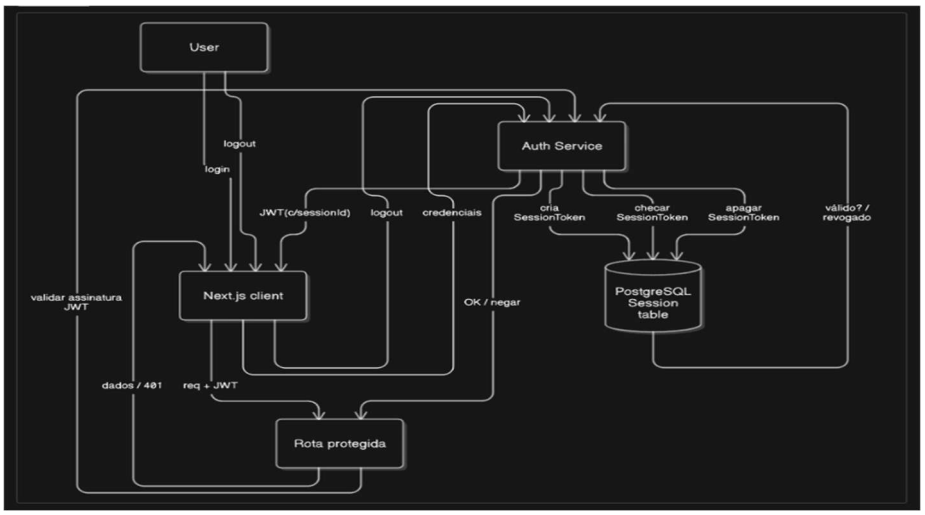
3.2.3.7 Segurança: jwt e tokens de sessão

A segurança do sistema é garantida por uma combinação de dois mecanismos principais além da autenticação do Auth.JS o JWT (JSON Web Tokens) que é utilizado como método seguro para autenticação, gerando tokens assinados digitalmente que contêm informações relevantes, como o ID e o cargo do usuário. Esses tokens são validados em cada requisição feita ao servidor, garantindo a integridade e a segurança dos dados trafegados (JWT, 2024).

Em conjunto do jwt o segundo mecanismo foram os Tokens de Sessão que funcionam como um complemento ao JWT e eles ficam armazenados no banco de dados (PostgreSQL) com prazos de validade configuráveis, como, por exemplo, 24 horas. Esse mecanismo permite revogação imediata do acesso em casos de logout ou bloqueio de conta, além de adicionar uma camada extra de controle e segurança. Essa abordagem atende aos requisitos não funcionais de confidencialidade, integridade e controle rigoroso de acesso.

A junção do uso de JWT com tokens de sessão cria um equilíbrio ideal entre segurança e desempenho, como ilustrado no diagrama da figura 12. Que mostra, em linhas gerais, que ao fazer login o usuário envia suas credenciais ao serviço de autenticação, que cria um registro de sessão no PostgreSQL e devolve ao cliente um JWT assinado; cada requisição subsequente a rotas protegidas traz esse JWT, onde a assinatura é verificada rapidamente e em seguida o servidor confirma no banco se a sessão ainda está ativa só então os dados são liberados; se o usuário faz logout ou a conta é bloqueada, o registro de sessão é revogado e qualquer JWT remanescente perde a validade na próxima verificação, garantindo agilidade no dia a dia e revogação imediata quando necessário.

Figura 12 - Fluxo de segurança com JWT e tokens de sessão



Fonte: Elaborado pelos autores (2025)

3.2.4 Desenvolvimento

Nesta etapa, depois todas as análises e escolhas se iniciou o desenvolvimento, sendo o principal objetivo transformar os requisitos funcionais e não funcionais, bem como as especificações técnicas que foram definidas anteriormente, em um software funcional. O desenvolvimento do sistema foi conduzido de forma estruturada, adotando o modelo cascata e aplicando práticas recomendadas pelo modelo MPS.BR, conforme descrito no Guia Geral (SOFTEX, 2020). Essa abordagem visou assegurar que cada etapa fosse executada de maneira controlada, com rastreabilidade e foco na qualidade do produto.

Dentre as práticas implementadas, destacam-se:

Gerência de Requisitos (GRE): foi realizado um levantamento sistemático dos requisitos funcionais e não funcionais, seguido da documentação e validação junto aos stakeholders, garantindo que as funcionalidades atendam tanto às necessidades institucionais do IFAP quanto aos critérios de avaliação do SINAES.

Gerência de Projetos (GPR): estabeleceu-se um planejamento detalhado, definindo escopo, cronograma, recursos e responsabilidades, permitindo o acompanhamento contínuo do progresso e o controle de desvios em relação ao planejamento inicial.

Aquisição (AQU): as tecnologias empregadas React e Next.js no frontend, PostgreSQL no backend e Prisma.js como ORM, foram selecionadas a partir de critérios técnicos de desempenho, escalabilidade, facilidade de manutenção e compatibilidade com o contexto institucional.

Gerência de Configuração (GCO): utilizou-se controle de versão via Git para código-

fonte e documentação, assegurando a rastreabilidade, integridade e possibilidade de restauração de versões anteriores do projeto.

Garantia da Qualidade (GQA): foram realizadas revisões técnicas e testes automatizados (unitários e de integração) para verificar a conformidade das entregas com os requisitos estabelecidos e padrões definidos.

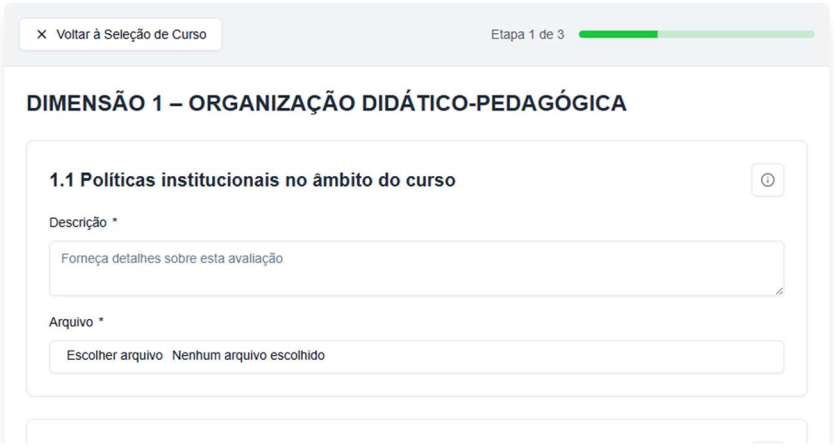
A aplicação dessas práticas contribuiu para a padronização dos processos, prevenção de falhas e melhoria da eficiência no desenvolvimento, resultando em um sistema robusto, escalável e alinhado às demandas regulatórias e acadêmicas da instituição.

3.2.4.1 Implementação das Funcionalidades

No decorrer do desenvolvimento, foram feitas as principais funcionalidades do sistema, tudo baseado nos requisitos funcionais identificados na sessão 3.2.2 (requisitos do sistema). Foram feitas diversas funcionalidades para o desenvolvimento do aplicativo sendo elas:

A avaliação dos coordenadores que foram criados formulários interativos utilizando o react Hook form e o zod para que os coordenadores cadastrem indicadores de avaliação, associando-os a anexos e descrições detalhando seu propósito, promovendo uma documentação clara e organizada.

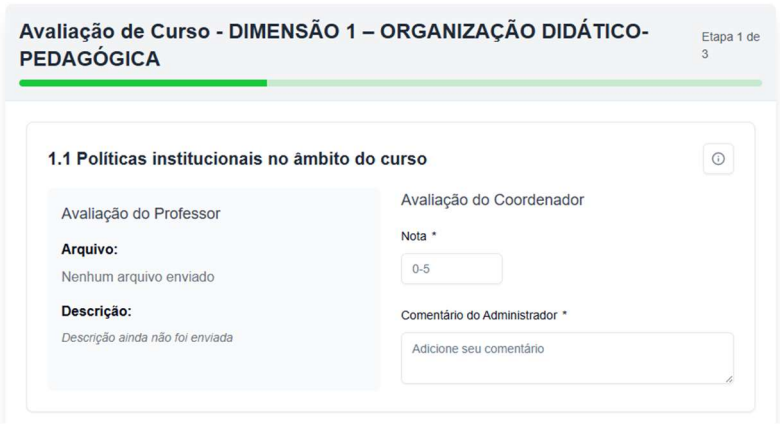
Figura 13 - Formulário de avaliação na visão do coordenador I



Fonte: Elaborado pelos autores (2025)

A avaliação dos Administradores que foi implementada uma funcionalidade que possibilita aos administradores atribuírem notas (de 0 a 5) e justificativas textuais aos indicadores, com validações para assegurar a consistência dos dados inseridos.

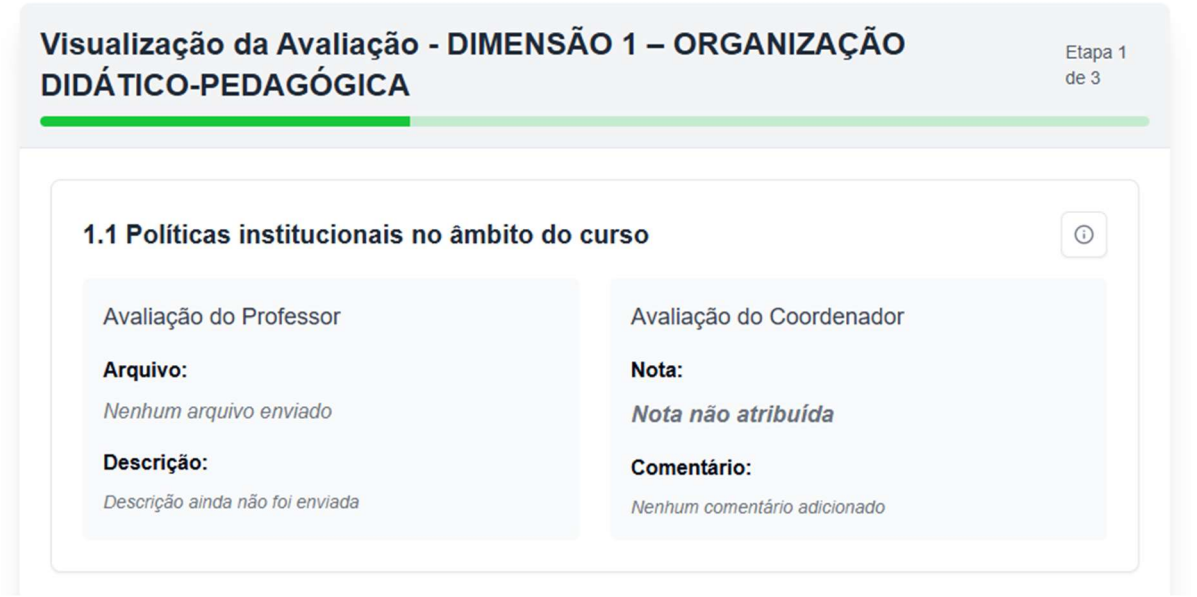
Figura 14 - Formulário de avaliação na visão do coordenador II



Fonte: Elaborado pelos autores (2025)

A visualização restrita que foi criada uma interface específica para usuários sem permissões administrativas ou de coordenação, permitindo apenas a consulta das avaliações, sem opções de edição.

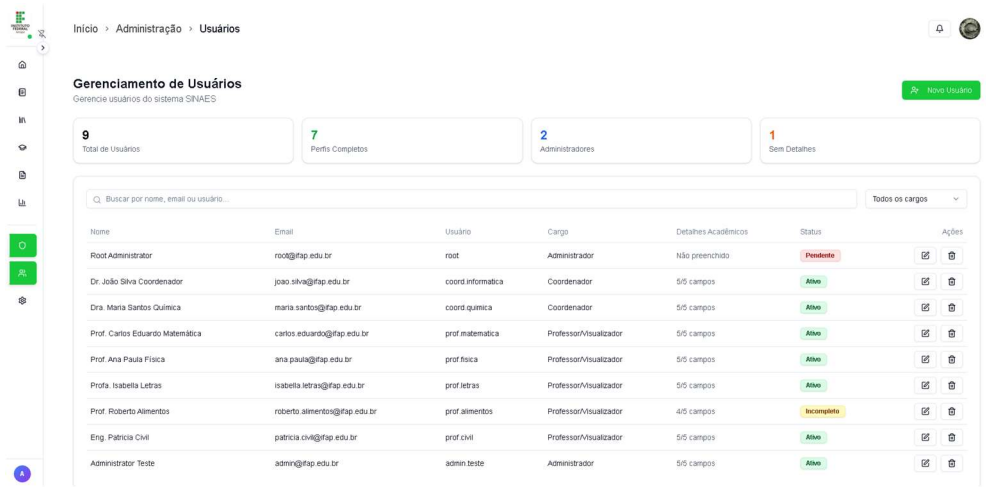
Figura 15 - Visualização restrita para somente leitura do formulário de avaliação



Fonte: Elaborado pelos autores (2025)

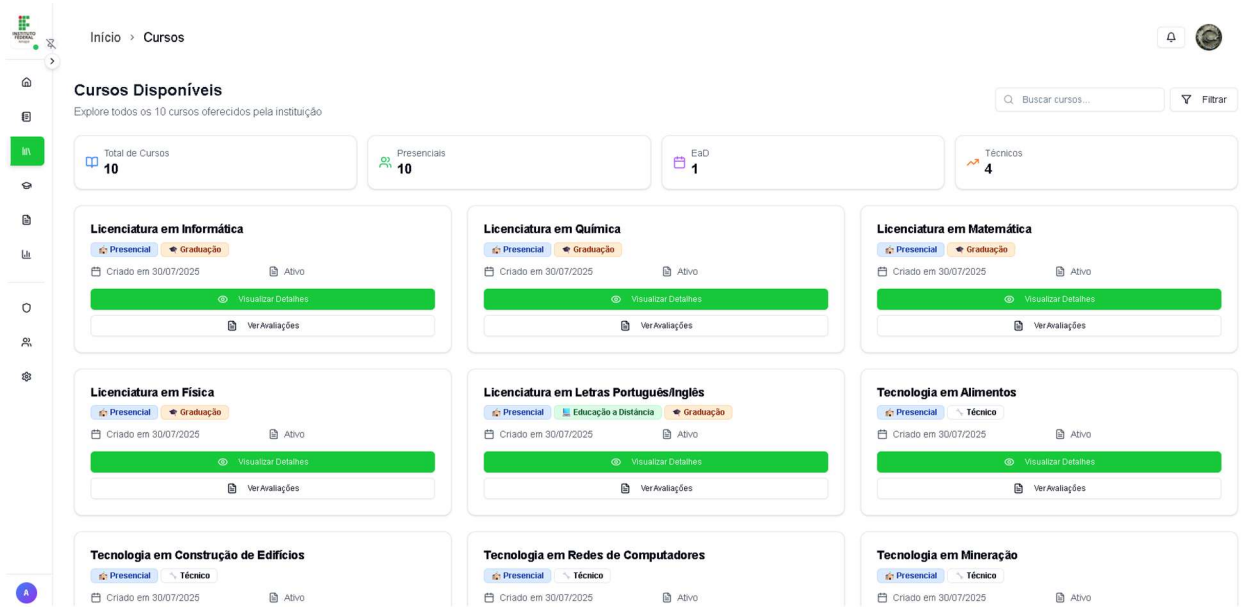
A Gestão de Administrador que nela foi construído interfaces administrativas para o gerenciamento de usuários, cursos e indicadores, com controles de acesso rigorosos para proteger informações sensíveis.

Figura 16 - Gerenciamento de usuários



Fonte: Elaborado pelos autores (2025)

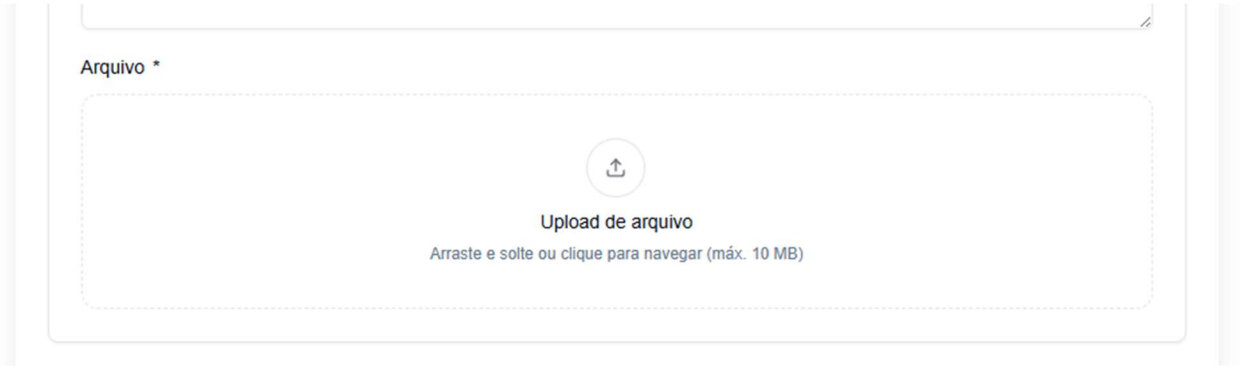
Figura 17 - Cursos disponíveis



Fonte: Elaborado pelos autores (2025)

O upload de anexos que permite aos coordenadores enviarem documentos nos formatos PDF e DOCX, bem como imagens JPG, com validação de extensões e limite de tamanho, atendendo à demanda por digitalização das evidências institucionais.

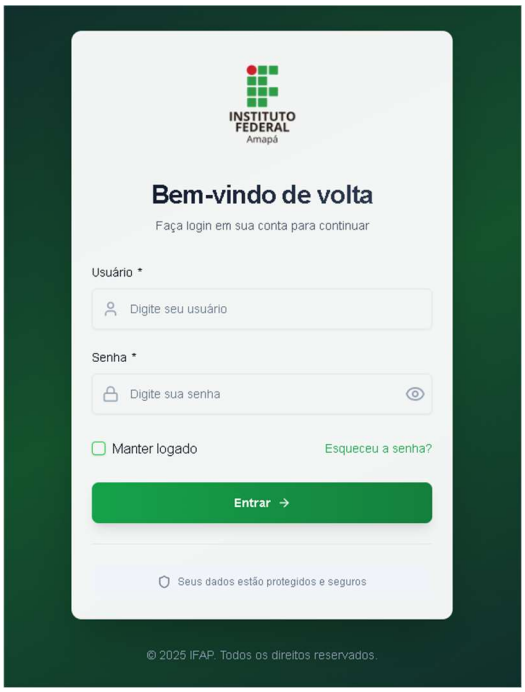
Figura 18 - Upload de anexos



Fonte: Elaborado pelos autores (2025)

A autenticação e Autorização Utilizando do Auth.js (versão 5.x) para gerenciar o login e o controle de acesso baseado em cargos, utilizando JSON Web Tokens (JWT) e tokens de sessão para garantir segurança e flexibilidade.

Figura 19 - Tela de autenticação e autorização de acesso



Fonte: Elaborado pelos autores (2025)

Figura 20 - Código para gerenciar login e o controle de acesso

```
callbacks: {
  async jwt({ token, user }) {
    if (user) {

      token.accessToken = user.accessToken;
      token.refreshToken = user.refreshToken;
      token.accessTokenExpires = Date.now() + 15 * 60 * 1000;
      token.user = {
        ...user
      };

      return token;
    }



    else {

      if (!token.refreshToken) throw new TypeError("Missing refresh_token")
      const revalidateRefresh = await refreshAccessToken(token);
      if (revalidateRefresh?.error) {
        token.error = "RefreshTokenError"
        throw new Error("Refresh token expired");
      }
      const newTokens = revalidateRefresh;
      token.access_token = newTokens?.accessToken
      token.expires_at = Math.floor(
```

Fonte: Elaborado pelos autores (2025)

Os relatórios que foi feito a geração de relatórios automatizados por curso, contendo médias das notas e comentários consolidados, exportáveis em formato PDF para uso offline.

3.2.5 Testes e Implementação

O processo de testes e implementação do sistema foi realizado de forma contínua, com o objetivo de garantir que todos os requisitos funcionais e não funcionais fossem atendidos conforme o planejado. Os testes foram conduzidos ao longo de todo o desenvolvimento, validando cada funcionalidade com base nos requisitos estabelecidos, o que permitiu identificar falhas e fazer ajustes rápidos.

Os testes começaram pela validação dos requisitos funcionais, que garantiram que cada funcionalidade do sistema estivesse operando conforme esperado, como exemplo foram verificados se os coordenadores conseguiam gerenciar os indicadores e se os relatórios gerados estavam corretos, também foi verificado se o controle de acesso estava correto conforme os diferentes tipos de usuários. Para cada requisito, foram realizados testes manuais e automatizados para garantir a precisão das funcionalidades.

Foram feitos testes também para ver as avaliações dos usuários que não participaram do processo do software sobre o sistema no geral, que nos ajudaram a entender se as telas estavam

intuitivas e se todos conseguiam entender o que estava sendo mostrada. Com base nisso, conseguimos melhorar o design do sistema, deixando a experiencia melhor para os usuários.

Na fase de desenvolvimento, corrigimos problemas rapidamente, seguindo as sugestões dos testes. Isso nos permitiu ajustar o sistema de acordo com o que o IFAP esperava. Também fizemos testes para garantir que todas as partes do sistema, como o lado de trás do sistema, o banco de dados e as telas, funcionassem juntas sem problemas e sem erros.

No final, testamos o sistema em um ambiente de uso real, com a ajuda dos usuários que vão usar o sistema. Isso garantiu que tudo funcionasse da maneira certa e que atendesse as necessidades da instituição e as regras do MEC. Essa etapa foi muito importante para entregar um sistema sólido e confiável para o IFAP.

4 RESULTADOS E DISCUSSÕES

O principal objetivo da plataforma criada é auxiliar o IFAP na autoavaliação dos cursos superiores de maneira mais estruturada e eficiente, em conformidade com os padrões do MEC. O sistema foi projetado para simplificar o trabalho dos coordenadores, do procurador institucional e de outros colaboradores, que anteriormente precisavam recorrer a planilhas, documentos distintos e até fichas em papel para gerenciar essas informações.

Com o novo sistema, é possível registrar as avaliações, adicionar documentos, justificar as notas, criar relatórios por curso e receber avisos quando houver alguma mudança. O objetivo foi tornar tudo o mais simples possível, mantendo a segurança dos dados, o que simplifica o trabalho do usuário.

4.1 Resultados positivos

Durante o desenvolvimento e os testes, foi possível perceber que o sistema trouxe vários benefícios importantes, como a centralização das informações fazendo com que tudo esteja em um único lugar, o que simplifica a consulta e previne a perda de dados. A organização também melhorou no quesito de organização pois cada curso tem seu próprio espação nos sistemas além de as avaliações obedecerem a um padrão. Os relatórios gerados ajudam a identificar os pontos fortes e fracos de cada curso melhorando assim a tomada de decisão. A integração do sistema ajudou a reduzir o trabalho pois as informações não precisam ser inseridas várias vezes. E, por fim, o sistema segue os mesmos critérios das avaliações do MEC, o que torna mais fácil a preparação para as visitas e garante que a instituição esteja alinhada com as exigências do ministério.

4.2 Limitações e dificuldades encontradas

Apesar dos bons resultados, algumas dificuldades e limitações apareceram ao longo do projeto. A curva de aprendizado para alguns usuários pode ser um problema, principalmente se não estiverem acostumados com sistemas digitais.

Também houve algumas situações desafiadoras, como entender bem os critérios usados pelo MEC e transformar isso em funcionalidades dentro da plataforma. Foi preciso fazer várias reuniões com pessoas da instituição para entender o processo e adaptar o sistema à realidade do IFAP.

O Controle de permissões e acessos também foi um dos desafios, por existir diversos tipos de usuários (Administradores, Coordenadores, Visualizadores), principalmente no que de garantir que todos eles tivessem o acesso correto e seguro. Isso exigiu ajustes contínuos para evitar falhas de segurança e garantir que somente usuários autorizados tivessem acesso a informações sensíveis.

Além disso, a escalabilidade do sistema passou a ser um aspecto a ser considerado. Apesar de ter sido projetado para atender a um número considerável de usuários, o desempenho pode ser afetado ao longo do tempo, especialmente com o crescimento do volume de dados. Isso implica que, conforme o uso do sistema cresce, serão necessárias melhorias.

4.3 Possíveis melhorias futuras

Pensando em possíveis melhorias futuras, existem algumas que poderiam ser implementadas de uma forma que torne o sistema mais eficiente e acessível, primeiramente ao adicionar vídeos e tutoriais para ajudar novos usuários a se familiarizarem com a plataforma diminuindo assim a curva de aprendizado, além disso, adicionar algumas integrações com outros sistemas como por exemplo o SUAP (Sistema Unificado de Administração Pública), melhoraria a importação de dados que aumentaria a precisão das informações.

Outras melhorias que poderiam ser adicionadas seria a adição de lembretes e alertas que notificaria os operadores sobre os prazos importantes, garantindo assim que nada fosse esquecido ao longo do processo. Por fim, adicionar módulos de acessibilidade, para tornar o sistema utilizável para todos, incluindo pessoas com deficiência e garantindo que a plataforma seja inclusiva e atenda a todas as necessidades dos usuários.

5 CONCLUSÃO

O desenvolvimento da plataforma de autoavaliação para o IFAP veio para resolver um problema que constantemente fazia a autoavaliação ser muito trabalhoso, que era a forma como as informações eram organizadas e avaliadas. Anteriormente, era tudo feito com planilhas, documentos e até fichas em papel, o que deixava o processo lento e sujeito a erros. Agora, com o sistema, tudo fica centralizado, fácil de acessar e seguindo os critérios que o MEC usa nas avaliações.

Usando tecnologias como React, Next.js, PostgreSQL e Prisma.js, foi possível criar um sistema rápido, seguro e que qualquer pessoa com um pouco de prática já consegue usar. A aplicação das diretrizes do MPS.BR ajudou a manter a qualidade do desenvolvimento, fazendo com que cada etapa fosse pensada para atender exatamente o que a instituição precisava.

O sistema trouxe várias funcionalidades que fazem diferença no dia a dia, como a gestão de indicadores, avaliação pelos administradores, relatórios automáticos e um dashboard que mostra tudo de forma clara. Isso facilita muito a identificar o que precisa melhorar e se preparar melhor para as avaliações externas, além de reduzir bastante o retrabalho.

A escolha do modelo cascata, junto com boas práticas do MPS.BR, ajudou a deixar o desenvolvimento mais organizado, mas também permitiu fazer ajustes quando era necessário. Assim foi possível chegar no resultado que a gente queria, sem fugir do planejamento.

No fim, a plataforma cumpriu o objetivo principal e entregou algo que realmente pode melhorar a gestão acadêmica do IFAP, deixando o processo de autoavaliação mais rápido, mais confiável e mais fácil de acompanhar.

REFERÊNCIAS

AUTHJS. **Auth.js documentation**. Disponível em: https://authjs.dev. Acesso em: 10 out. 2024.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML:** guia do usuário. 2. ed. Rio de Janeiro: Campus Elsevier, 2006.

ERASER.IO. **Eraser**. Disponível em: https://www.eraser.io. Acesso em: 10 jul. 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAPÁ. **Relatório da autoavaliação institucional**. Macapá: Comissão Própria de Avaliação (CPA), 2015. Disponível em: https://www.ifap.edu.br/index.php/editais-de-pesquisa/item/download/93_f8c6b25b5a9cc082262ee0fcadf3d5be. Acesso em: 12 ago. 2025.

JWT.IO. **JSON Web Tokens**. Disponível em: https://jwt.io. Acesso em: 10 out. 2024.

META PLATFORMS, Inc. **React**: a biblioteca JavaScript para construção de interfaces de usuário. Disponível em: https://react.dev. Acesso em: 13 maio 2025.

MOMJIAN, Bruce. **PostgreSQL**: introduction and concepts. Boston: Addison-Wesley, 2001.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software**: uma abordagem profissional. 9. ed. Porto Alegre: AMGH Editora, 2020.

REACT HOOK FORM. **React Hook Form**. Disponível em: https://www.react-hook-form.com. Acesso em: 10 out. 2024.

ROYCE, Winston W. Managing the development of large software systems. In: IEEE. **Proceedings of the IEEE WESCON**. Los Angeles, 1970.

SHADCN. **Shadcn UI**. Disponível em: https://ui.shadcn.com. Acesso em: 10 out. 2024.

SOFTEX. **MPS.BR**: guia geral. Associação para Promoção da Excelência do Software Brasileiro, 2020. Disponível em: https://softex.br/mpsbr. Acesso em: 10 out. 2024.

TANSTACK. **TanStack Query documentation**. Disponível em: https://tanstack.com/query/latest. Acesso em: 10 jul. 2025.

TAILWIND LABS. **Tailwind CSS documentation**. Disponível em: https://tailwindcss.com/docs. Acesso em: 10 jul. 2025.

VERCEL INC. **Next.js documentation**. Disponível em: https://nextjs.org/docs. Acesso em: 10 jul. 2025.

WIEGERS, Karl; BEATTY, Joy. **Software requirements**. 3. ed. Redmond: Microsoft Press, 2013.

ZOD. **Zod**. Disponível em: https://zod.dev. Acesso em: 10 out. 2024.

ZUSTAND. **Zustand:** bearbones state-management for React. Disponível em: https://zustand-demo.pmnd.rs. Acesso em: 13 dez. 2024.