INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

CAMPUS GUARABIRA

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

MARIA ALICE DA SILVA TAVARES
VICTOR GUSTAVO CAVALCANTI DA SILVA

RELATÓRIO TÉCNICO

Projeto Integrador em Sistemas para Internet (PISI): Sistema de Assessoria Lúdica (SAC)

GUARABIRA – PB
2026

MARIA ALICE DA SILVA TAVARES
VICTOR GUSTAVO CAVALCANTI DA SILVA

RELATÓRIO TÉCNICO

Projeto Integrador em Sistemas para Internet (PISI): Sistema de Assessoria Lúdica (SAC)

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso Superior de Tecnologia em Sistemas para Internet, no âmbito do IFPB – Campus Guarabira, em cumprimento às exigências parciais para obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Rodrigo Leone Alves, Dr.
Coorientador: Rhavy Maia Guedes, Me.

GUARABIRA – PB
2026

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IFPB - GUARABIRA

T231r Tavares, Maria Alice da Silva

Relatório técnico: Projeto Integrador em Sistemas para Internet (PISI): Sistema de Assessoria Lúdica (SAC) / Maria Alice da Silva Tavares; Victor Gustavo Cavalcanti da Silva.- Guarabira, 2026.

51f.; il.; color.

Trabalho de Conclusão de Curso (Tecnólogo em Sistemas para Internet). – Instituto Federal da Paraíba, Campus Guarabira. 2026.

"Orientação: Prof. Dr. Rodrigo Leone Alves
Coorientação: Prof. Me. Rhavy Maia Guedes"

Referências.

1. Sistema para internet 2. Método montessori. 3. Gestão escolar. 4. Desenvolvimento web. 5. Assessoria lúdica. I. Título.

CDU 004.4(0.067)

Elaborada por Ana Carine da Costa Gonçalves - CRB 000676

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA

ATA 21/2026 - CCSTSI/DDE/DG/GB/REITORIA/IFPB

ATA DE APRESENTAÇÃO E DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

CCS de Tecnologia em Sistemas para Internet

Aos 06 de agosto de 2026, às 10:00, no Laboratório de Informática, reuniram-se os membros da banca avaliadora, Rodrigo Leone Alves (Orientador), Rhavy Maia Guedes (Coorientador), Tannissa Luanna Cardoso de Araujo (Examinador Interno), Daniel Marques Targino Guimarães (Examinador Interno), para avaliarem a apresentação do *Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet* (Relatório Final do Projeto Integrador em Sistemas para Internet - PISI) do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), *campus* Guarabira, desenvolvido pela aluno(a) **Maria Alice da Silva Tavares , matrícula de nº 202213810009** intitulado "Sistema de Assessoria Lúdica (SAC)", protocolado para apresentação de acordo com os requisitos expostos no Projeto Pedagógico de Curso de Tecnologia em Sistemas para Internet. Após a apresentação, a banca apresentou, por unanimidade, pareceres a favor da aprovação do trabalho. Desta forma, o Trabalho de Conclusão de Curso foi **aprovado** e definiu-se a **nota final 91 (Noventa e Um)**.

Nada mais havendo a tratar, às 16:45, encerraram-se os trabalhos, determinando a lavratura desta ata, que, após lida e considerada conforme, será assinada pelos presentes. Eu, Rodrigo Leone Alves, lavrei a presente ata.

Guarabira/PB, em 10 de agosto de 2026.

Documento assinado eletronicamente por:

- **Rodrigo Leone Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 12:17:50.
- **Tannissa Luanna Cardoso de Araujo**, **PEDAGOGO-AREA**, em 10/08/2026 12:49:15.
- **Daniel Marques Targino Guimaraes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 10/08/2026 22:38:55.

Este documento foi emitido pelo SUAP em 10/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 919915
Código de Autenticação: ae07ec88ff

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS GUARABIRA

ATA 22/2026 - CCSTSI/DDE/DG/GB/REITORIA/IFPB

ATA DE APRESENTAÇÃO E DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

CCS de Tecnologia em Sistemas para Internet

Aos 06 de agosto de 2026, às 10:00, no Laboratório de Informática, reuniram-se os membros da banca avaliadora, Rodrigo Leone Alves (Orientador), Rhavy Maia Guedes (Coorientador), Tannissa Luanna Cardoso de Araujo (Examinador Interno), Daniel Marques Targino Guimarães (Examinador Interno), para avaliarem a apresentação do *Trabalho de Conclusão do Curso Superior de Tecnologia em Sistemas para Internet* (Relatório Final do Projeto Integrador em Sistemas para Internet - PISI) do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), *campus* Guarabira, desenvolvido pelo(a) aluno(a) **Victor Gustavo Cavalcanti da Silva, matrícula de nº 202213810011** intitulado "Sistema de Assessoria Lúdica (SAC)", protocolado para apresentação de acordo com os requisitos expostos no Projeto Pedagógico de Curso de Tecnologia em Sistemas para Internet. Após a apresentação, a banca apresentou, por unanimidade, pareceres a favor da aprovação do trabalho. Desta forma, o Trabalho de Conclusão de Curso foi **aprovado** e definiu-se a **nota final 91 (Noventa e Um)**.

Nada mais havendo a tratar, às 16:45, encerraram-se os trabalhos, determinando a lavratura desta ata, que, após lida e considerada conforme, será assinada pelos presentes. Eu, Rodrigo Leone Alves, lavrei a presente ata.

Guarabira/PB, em 10 de agosto de 2026.

Documento assinado eletronicamente por:

- **Rodrigo Leone Alves**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 12:20:11.
- **Tannissa Luanna Cardoso de Araujo**, **PEDAGOGO-AREA**, em 10/08/2026 12:49:24.
- **Rhavy Maia Guedes**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 10/08/2026 15:41:42.
- **Daniel Marques Targino Guimaraes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 10/08/2026 22:37:51.

Este documento foi emitido pelo SUAP em 10/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 919918
Código de Autenticação: 7b1efc3e60

A Deus, meu sopro de vida. Ao meu parceiro de alma, de vida e também deste TCC. Aos meus pais, meus alicerces que sempre me incentivaram e fizeram de tudo para que eu só precisasse me preocupar com os estudos. À minha irmã, inestimável melhor amiga e minha outra metade na vida. A todos os professores, amigos e colegas de turma.

Chegar aqui não foi só um feito acadêmico, foi escolher lutar pela vida mesmo mal aguentando viver. Bem, eu estou viva, eu estou aqui e esse momento é meu meu e de todos os que continuam lutando por seus sonhos, mesmo enfrentando problemas psicológicos.

Maria Alice

Para a minha colega de faculdade e, hoje, companheira de vida àquela que viu o pior de mim e mesmo assim decidiu ficar.

Para os meus camaradas, que juntos proporcionaram momentos divertidos em meio à dureza da rotina. Para os meus pais, que, mesmo com todas as preocupações do mundo, nunca deixaram de acreditar na minha trajetória. Para o meu irmão, aquele a quem jurei proteger de todas as maldades do mundo. E para todos os professores, amigos e colegas a quem tive o privilégio de conhecer.

Cada um de vocês é parte desse pequeno trecho da minha vida, que para sempre ficará guardado em minhas memórias.

Victor Gustavo

Agradecimentos

Agradecimentos de Maria Alice

Olhando para trás, reconheço que ultrapassar cada obstáculo só foi possível porque não caminhei só. Aos que construíram diretamente este trabalho e sustentaram meus passos, a minha mais profunda gratidão.

Ao amor da minha vida e parceiro de jornada, Victor Gustavo, por me acolher nos dias mais difíceis, ensinar-me com dedicação e construir este projeto ao meu lado. À minha irmã e melhor amiga, Julia Tavares, minha eterna inspiração, e me apresentou a área de tecnologia. Ao mestre e coorientador Rhavy Maia, minha maior referência acadêmica, expresso minha gratidão imensurável: por sua paciência genuína, por despertar minha paixão pelo JavaScript, por sua orientação brilhante na elaboração desta documentação e por seu olhar humano e atento, sempre acreditando no meu potencial e oferecendo o suporte necessário em cada desafio.

Às minhas colegas e amigas de curso, Nielen Jhose e Maria Vitória: obrigada por toda cumplicidade e carinho nesta jornada acadêmica. Sem vocês, esse caminho não teria a mesma graça.

Agradeço imensamente à minha psicóloga Maria Clarissa Luna e à minha psicopedagoga Layse Costa, por me acolherem com sabedoria, cuidando da minha saúde mental e do meu desenvolvimento acadêmico com um zelo transformador. Aos meus coordenadores de extensão, Líbna Naftali e Augusto Viana, meus "segundos pais", o meu sincero obrigado pelo afeto, escuta atenta e acolhimento em cada etapa.

Aos meus professores, expresso minha mais profunda admiração. Ao mestre José Barros, por sua inspiradora vivência cristã, entusiasmo contagiante e sensibilidade técnica; ao mestre Elenilson Vieira, pelo incentivo constante e por ter me confiado a inesquecível oportunidade de vivenciar minha primeira experiência lecionando; ao mestre Lucas Souza, por sua didática impecável, suporte diário e por transformar qualquer resistência inicial em profunda admiração; à Dra. Gabriela Guedes, nossa coordenadora extraordinária, pelo brilhantismo e apoio humano imensurável; ao mestre Erick Sharlss, por sua energia leve e por tornar o conhecimento algo tão dinâmico e cativante; ao Dr. Rodrigo Leone, por seu

ensino de excelência e rigor acadêmico; ao mestre Daniel Marques, por sua dedicação e contribuição direta para minha formação profissional; e a todos os demais professores que compõem este corpo docente extraordinário, meu muito obrigada por cada ensinamento compartilhado.

Minha gratidão à Gleizy Fortunato (CLAI), por conduzir minha acessibilidade e inclusão no campus com tanto carinho e responsabilidade. Estendo meus agradecimentos ao segurança Tiago Gomes, às queridas Luciana Almeida e Joseilda da Silva, ao porteiro Joel de Oliveira e aos motoristas Fabiano Lima e Johandson Lima, que com gentileza e humanidade tornaram minha rotina acadêmica muito mais leve.

A toda a comunidade e equipe do IFPB, minha eterna gratidão e o maior orgulho por fazer parte desta instituição.

Agradecimentos de Victor Gustavo

É difícil definir em palavras devido aos limites físicos deste humilde documento e ao meu modesto domínio da escrita o quão grato estou por estar vivendo este momento. Sempre duvidei que um dia iria viver isso...

Inicialmente, eu gostaria de agradecer a todos aqueles que contribuíram diretamente para a construção deste documento. Muito obrigado, Alice Tavares, meu amor e companheira, por ter feito parte deste projeto e por caminhar ao meu lado na vida! Muito obrigado, Lucas Santos (História UEPB), por ter cedido um pouco do seu tempo para a revisão deste trabalho e um pouco do seu conhecimento nas indicações literárias; sem as suas reflexões e provocações, tenho certeza de que eu continuaria cego pela minha ignorância. Agradeço também ao mestre Rhavy Maia, por ter colaborado como coorientador deste projeto sua experiência e seus conselhos foram fundamentais para a qualidade técnica deste trabalho.

Aos que estiveram comigo em outros projetos: Me. Elenilson Vieira (FIC Digital), Esp. Pedro Gabi (PI Digital) e Me. José Barros (Monitoria), obrigado por terem acreditado no meu potencial mesmo quando eu mesmo não o fiz.

Aos professores Me. Lucas Souza, Dra. Gabriela Guedes, Me. Erick Sharlls, Dr. Rodrigo Leone, Me. Daniel Marques e aos demais que compõem o excelente corpo docente do Instituto Federal, só tenho a agradecer por cada segundo que pude estar com vocês. Cada um de vocês é parte do profissional que estou me tornando. Obrigado por toda entrega, esforço e comprometimento em serem os melhores!

Resumo

O modelo de ensino tradicional frequentemente desconsidera as particularidades e os ritmos individuais de aprendizagem dos estudantes. Em contrapartida, a Pedagogia Montessori promove a autonomia e o acompanhamento qualitativo contínuo do desenvolvimento infantil. Contudo, em assessorias lúdicas orientadas por essa abordagem, a dependência de processos manuais na gestão de matrículas e nos registros pedagógicos gera gargalos operacionais e dificulta a comunicação transparente com as famílias. Este trabalho apresenta o desenvolvimento do Sistema de Assessoria Lúdica (SAC), uma plataforma web integrada concebida para otimizar rotinas administrativas e sistematizar o acompanhamento pedagógico. Metodologicamente, enquadra-se como uma pesquisa aplicada de natureza tecnológica, apoiada no gerenciamento ágil com Kanban e práticas de engenharia de software do Extreme Programming. Como resultados, consolidou-se a especificação de requisitos, a modelagem de dados relacional e a construção de uma suíte de testes automatizados de contrato e integração. Espera-se que a implantação da plataforma proporcione maior agilidade na homologação de matrículas e fortaleça o engajamento com as famílias por meio do acompanhamento transparente do desenvolvimento infantil.

Palavras-chave: Método Montessori, Gestão Escolar, Assessoria Lúdica, Test-Driven Development, Desenvolvimento Web.

Abstract

Traditional educational models often disregard individual student learning styles and paces. In contrast, Montessori Pedagogy promotes autonomy and continuous qualitative monitoring of child development. However, play-based educational centers following this approach face operational bottlenecks and communication gaps with families due to reliance on manual administrative and pedagogical tracking processes. This work presents the development of the Playful Advisory System (SAC), an integrated web platform designed to streamline administrative routines and systematize pedagogical tracking. Methodologically, it is categorized as applied research of a technological nature utilizing Kanban agile workflow integrated with Extreme Programming software engineering practices. Results include the formal specification of requirements, a relational data model, and an automated API contract and integration testing suite. Upon full deployment, the platform is expected to increase operational efficiency in enrollment approvals and strengthen family engagement through transparent tracking of child development.

Keywords: Montessori Method, School Management, Playful Advisory, Test-Driven Development, Web Development.

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

API – Application Programming Interface;
AWS – Amazon Web Services;
BLOB – Binary Large Objects;
CI/CD – Continuous Integration / Continuous Deployment;
DOM – Document Object Model **E2E** – End-to-End;
HTML – HyperText Markup Language;
HTTP – Hypertext Transfer Protocol;
HTTPS – HTTP Secure;
JS – JavaScript;
LGPD – Lei Geral de Proteção de Dados;
MER – Modelo Entidade-Relacionamento;
MVC – Model-View-Controller;
ORM – Object Relational Model;
PISI – Projeto Integrador em Sistemas Para Internet;
PMI – Project Management Institute;
RBAC – Role-Based Access Control;
RF – Requisito Funcional;
RNF – Requisito Não Funcional;
ROI – Return on Investment;
S3 – Simple Storage Service;
SAC – Sistema de Assessoria Lúdica;
SDK – Software Development Kit;
SPA – Single Page Application;
SQL – Structured Query Language;
TDD – Test-Driven Development;
VU – Virtual Users;
UUID – Universally Unique Identifier;
WIP – Work in Progress;
XP – Extreme Programming.

Sumário

1 Introdução

Por décadas, o modelo tradicional de ensino tem sido aplicado sob uma lógica excessivamente rígida, marcada pela ênfase na memorização, pelo controle disciplinar estrito e pela escassa participação discente. Segundo Freire (2019), em sua obra *Pedagogia do Oprimido*, esse modelo se estrutura como uma "educação bancária", na qual o professor deposita conteúdos em um aluno tratado como receptor passivo, e não como sujeito ativo do próprio conhecimento. Em muitas práticas vigentes, o professor aparece como centro transmissor e exclusivo do conhecimento, em aulas predominantemente expositivas e pouco práticas; o erro é tratado como mecanismo de punição; e o ritmo de ensino é homogeneizado, desconsiderando particularidades e tempos dos estudantes. Esse cenário aproxima-se do processo de docilização dos corpos descrito por Foucault (1977), para quem as instituições disciplinares – entre elas a escola – organizam-se em torno da vigilância constante e da normalização do comportamento dos indivíduos. Na prática, isso se manifesta na padronização de comportamentos e formas de participação, na disciplina como gestão do cotidiano escolar e na redução do espaço para questionamento e elaboração autoral por parte dos estudantes.

O método Montessoriano, por sua vez, adota uma lógica diferente. De acordo com Montessori (1949; 2013), o aluno assume o papel de protagonista de seu aprendizado, agindo de forma ativa por meio da autoeducação e da interação empírica com o ambiente preparado; nesse modelo, o aluno vivencia um desenvolvimento integral (holístico), que respeita suas necessidades físicas, cognitivas e psicológicas.

No entanto, a viabilização de uma assessoria lúdica orientada por esses princípios pedagógicos enfrenta sérios desafios de ordem operacional quando desprovida de infraestrutura tecnológica adequada. No caso analisado neste trabalho, a assessoria lúdica utilizada como referência empírica tem rotinas administrativas, processos de matrícula e gestão cadastral dependentes de intervenções manuais constantes, o que pode gerar inconsistências nos registros e aumentar a vulnerabilidade a erros. No âmbito pedagógico, a ausência de uma ferramenta automatizada para a elaboração de relatórios dificulta a consolidação de uma documentação padronizada e transparente sobre o progresso individual de cada aluno. Assim, famílias e responsáveis recebem informações fragmentadas e/ou ambíguas, o que abre margem para interpretações errôneas e inviabiliza a autêntica parceria escola-família que Freire (1997) descreve como fundada no diálogo horizontal entre os sujeitos do processo

educativo. Desse modo, compromete-se também a orientação individual do desenvolvimento da criança defendida por Montessori (1949; 2013), para quem o acompanhamento pedagógico depende de observação e registro contínuos das conquistas de cada aluno.

1.1 Justificativa

A necessidade deste trabalho fundamenta-se na vulnerabilidade operacional enfrentada por assessorias lúdicas que dependem de processos manuais. A ausência de uma infraestrutura tecnológica adequada gera inconsistências administrativas, riscos à integridade dos dados e dificulta a padronização de relatórios pedagógicos. Essa carência técnica compromete a aplicação prática dos princípios de Montessori e Freire, uma vez que a fragmentação das informações impede o acompanhamento contínuo do desenvolvimento individual do aluno e obstrui o diálogo transparente entre escola e família. Portanto, o desenvolvimento de um sistema de gestão integrado justifica-se pela urgência em automatizar rotinas para garantir eficiência administrativa e promover uma parceria educativa sólida, baseada em dados centralizados e acessíveis.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema de gestão de dados para **Assessoria Lúdica** que automatize rotinas e processos administrativos, além de facilitar a troca de informações entre professores e responsáveis por meio de um banco de dados centralizado.

1.2.2 Objetivos Específicos

- Automatizar o processo de matrícula, reduzindo o tempo de resposta e a carga manual de trabalho administrativo;
- Implementar um banco de dados robusto para o armazenamento e recuperação de relatórios pedagógicos detalhados;
- Criar uma interface de feedback para os pais, promovendo a transparência e a melhoria contínua dos serviços lúdicos;
- Garantir a integridade e segurança dos dados sensíveis de alunos e responsáveis em conformidade com as boas práticas de desenvolvimento.

1.3 Estrutura do Trabalho

O presente trabalho está estruturado em seis capítulos. O Capítulo 2 apresenta a fundamentação teórica, abordando a Pedagogia Montessori, a comunicação escola-família e os sistemas de informação aplicados à gestão educacional. O Capítulo 3 descreve a metodologia de desenvolvimento, incluindo a caracterização da pesquisa, o fluxo de trabalho com Kanban e XP, e os procedimentos técnicos adotados. O Capítulo 4 detalha a especificação dos requisitos funcionais e não funcionais do sistema. O Capítulo 5 expõe os resultados alcançados, abrangendo a modelagem do banco de dados, os diagramas de casos de uso, a infraestrutura de armazenamento, os testes automatizados e as interfaces da aplicação. Por fim, o Capítulo 6 traz as considerações finais e as perspectivas para trabalhos futuros.

2 Fundamentação Teórica

O presente capítulo tem como objetivo apresentar os fundamentos teóricos que sustentam esta pesquisa, abordando os desafios da gestão no modelo pedagógico Montessori, a importância do acompanhamento do desenvolvimento infantil e o papel da digitalização de processos em assessorias lúdicas. Inicialmente, discute-se a Pedagogia Montessori e a relevância da observação e do acompanhamento qualitativo contínuo. Em seguida, aborda-se a importância da transparência e do diálogo entre a instituição e as famílias. Por fim, apresenta-se o papel dos sistemas de informação na automação de processos administrativos e no gerenciamento pedagógico.

2.1 A Pedagogia Montessori e o Acompanhamento do Desenvolvimento Infantil

A Pedagogia Montessori, desenvolvida por Maria Montessori no início do século XX, fundamenta-se no respeito ao ritmo natural de aprendizagem da criança, na promoção da autonomia e no uso de um ambiente preparado para estimular a autoeducação, conforme descreve a própria autora em *Pedagogia Científica: A Descoberta da Criança* (MONTESSORI, 2013). Em *Mente Absorvente*, Montessori (1949) defende que, nesse modelo, o educador atua como um observador e guia, cuja função principal é acompanhar e registrar o progresso individual dos alunos, em vez de aplicar avaliações padronizadas ou quantitativas.

O acompanhamento pedagógico montessoriano exige a elaboração constante de registros qualitativos que detalhem a interação da criança com os materiais e suas conquistas cognitivas e motoras. Ao analisarem esse tipo de acompanhamento, Santos e Lima (2020) argumentam que a documentação pedagógica contínua é a principal ferramenta para mapear a evolução do estudante, permitindo intervenções personalizadas que respeitem suas necessidades individuais.

Contudo, a execução manual desses registros e relatórios em instituições que adotam essa abordagem costuma gerar sobrecarga operacional para os educadores e risco de perda de informações ao longo do tempo. Assim, torna-se necessário o apoio de meios estruturados para garantir que as observações diárias sejam armazenadas de forma segura e

acessível.

2.2 A Comunicação Escola-Família e a Gestão Transparente

A parceria entre os educadores e os responsáveis é apontada pela literatura educacional como um fator determinante para o desenvolvimento integral da criança. Em *Pedagogia da Autonomia*, Freire (1997) sustenta que a prática educativa autêntica pressupõe o diálogo transparente e horizontal entre todos os sujeitos envolvidos no processo de formação, superando barreiras burocráticas que afastam as famílias do cotidiano escolar.

Nas assessorias lúdicas e instituições de ensino complementar, a comunicação com os pais frequentemente ocorre por meio de canais informais ou relatórios impressos entregues de forma esporádica. Ao investigarem esse cenário, Silva e Oliveira (2021) apontam que esse formato fragmentado pode acarretar ruídos na interpretação do progresso do aluno e atrasos no envio de *feedbacks* cruciais.

Nesse contexto, a disponibilização de canais diretos e organizados para o envio de relatos, áudios e atualizações diárias aproxima os responsáveis da rotina pedagógica, fortalecendo a relação de confiança com a instituição e assegurando que os pais acompanhem em tempo real as etapas de desenvolvimento de seus filhos.

2.3 Sistemas de Informação em Assessorias Lúdicas e Instituições de Ensino

A adoção de plataformas digitais e sistemas web de gestão centralizada tem se consolidado como uma solução indispensável para otimizar rotinas administrativas e pedagógicas em ambientes educacionais. Segundo Almeida (2022), a digitalização de processos reduz a incidência de erros manuais, elimina o retrabalho na consolidação de dados cadastrais e confere agilidade ao fluxo de matrículas e gestão de turmas.

No contexto de uma assessoria lúdica orientada por metodologias ativas, o uso de um sistema de informação integrado possibilita:

- **Centralização do Histórico Pedagógico:** Registro cronológico de observações, relatórios e arquivos multimídia associados ao perfil do aluno, garantindo a rastreabilidade da sua evolução;
- **Automação do Fluxo Administrativo:** Digitalização dos processos de solicitação, análise e homologação de matrículas, diminuindo o tempo de resposta e organizando o fluxo de documentos;

- **Controle de Acesso e Privacidade (RBAC):** Restrição do acesso a dados sensíveis de acordo com o perfil do usuário (Administrador, Professor e Responsável), assegurando a privacidade das informações e a conformidade com boas práticas de segurança.

3 Metodologia

Este capítulo apresenta a caracterização da pesquisa e a metodologia adotada para o desenvolvimento do sistema de gestão da assessoria lúdica, detalhando o processo de trabalho, as práticas de engenharia de software e os procedimentos técnicos empregados.

3.1 Caracterização da Pesquisa

O presente trabalho enquadra-se como uma pesquisa aplicada de natureza tecnológica. Caracteriza-se como aplicada por buscar a solução de um problema prático do cotidiano operacional e pedagógico de uma assessoria lúdica, mediante o desenvolvimento de uma aplicação web. O desenvolvimento integra análise das rotinas pedagógicas sob a ótica da Metodologia Montessori, levantamento de requisitos junto aos atores envolvidos, implementação segundo padrões de engenharia de software (MVC, TDD) e validação por testes automatizados.

3.2 Metodologia de Desenvolvimento de Software

Para o gerenciamento do projeto e a execução das etapas de desenvolvimento, adotou-se a metodologia **Kanban** integrada a princípios e práticas do **Extreme Programming (XP)**.

O Kanban foi utilizado para garantir o fluxo contínuo de trabalho e a visualização clara do progresso das tarefas por meio de um quadro de acompanhamento dividido nas etapas de *Backlog*, *A Fazer*, *Em Desenvolvimento*, *Em Teste* e *Concluído*. Essa abordagem permitiu limitar o trabalho em andamento (*Work in Progress* – WIP), reduzindo gargalos e otimizando o tempo de entrega das funcionalidades.

Ademais, incorporaram-se práticas do XP voltadas à garantia de qualidade do código e do produto final, tais como:

- **Desenvolvimento Orientado a Testes (TDD):** Escrita prévia de testes automatizados antes da implementação das regras de negócio, assegurando maior confiabilidade ao sistema;

- **Refatoração Contínua:** Melhoria constante da estrutura interna do código sem alterar seu comportamento externo, mantendo a simplicidade e clareza da solução;
- **Entregas Frequentes e Pequenas:** Disponibilização incremental dos módulos do sistema (Administrativo, Pedagógico e Interatividade com os Pais) para validação rápida.

3.3 Procedimentos Técnicos

Os procedimentos técnicos empregados no desenvolvimento da plataforma abrangeram as seguintes etapas:

3.3.1 Levantamento e Especificação de Requisitos

Mapeamento dos fluxos de trabalho da assessoria lúdica (matrículas, acompanhamento pedagógico e comunicação com os pais) para a definição e detalhamento formal dos Requisitos Funcionais (RF) e Não-Funcionais (RNF).

3.3.2 Arquitetura e Modelagem do Sistema

Adoção do padrão arquitetural *Model-View-Controller* (MVC) para assegurar a separação de responsabilidades. Nesta etapa, elaborou-se o Modelo Entidade-Relacionamento (MER) para estruturação do banco de dados relacional e os diagramas de Casos de Uso para representação das interações dos usuários com a plataforma.

3.3.3 Implementação

A construção do *back-end* utilizou a linguagem Python sob o microframework Flask, estruturado segundo a arquitetura em camadas (*Controller-Service-Model-Schema*). Para a persistência de dados no PostgreSQL, adotou-se a biblioteca de mapeamento objeto-relacional (ORM) SQLAlchemy com chaves primárias baseadas em identificadores únicos universais (UUIDv4) e suporte a exclusão lógica (*soft delete*). O gerenciamento de migrações de banco de dados foi conduzido pelo Alembic. A validação e serialização das requisições e respostas são gerenciadas pela biblioteca Marshmallow, enquanto o Flasgger/Swagger provê a documentação interativa das rotas RESTful.

Para a manipulação e armazenamento de arquivos de áudio e anexos do módulo de monitoramento, integrou-se o SDK `boto3` à ferramenta Kumo para emular localmente o serviço de armazenamento S3, viabilizando o consumo seguro das mídias no *front-end* por meio de URLs pré-assinadas (*Presigned URLs*). O ambiente de desenvolvimento e execução foi padronizado em contêineres Docker gerenciados via Docker Compose, utilizando

o gerenciador de pacotes `uv` no *Dockerfile* para garantir construções de imagem rápidas e determinísticas.

No *front-end*, desenvolveu-se uma Aplicação de Página Única (*Single Page Application* – SPA) utilizando ReactJS com o empacotador Vite. A navegação entre os módulos do sistema é gerenciada pelo React Router DOM, o Bootstrap assegura a usabilidade responsiva da interface e a biblioteca Zod realiza a validação de esquemas no lado do cliente antes do envio à API.

3.3.4 Tecnologias Web e Arquitetura do Ecossistema

A concepção e a estruturação de sistemas web modernos exigem a escolha de um ecossistema tecnológico que garanta escalabilidade, modularidade, manutenibilidade e alta disponibilidade. No âmbito do desenvolvimento do Sistema de Assessoria Lúdica (SAC), adotou-se uma arquitetura distribuída e desacoplada, fundamentada no paradigma cliente-servidor e ancorada em padrões consolidados de engenharia de software.

3.3.4.1 Arquitetura REST e Comunicação via JSON

O padrão arquitetural REST (*Representational State Transfer*), introduzido por Fielding (2000) em sua tese de doutorado sobre estilos arquiteturais para sistemas distribuídos, estabelece um conjunto de restrições para a construção de serviços web fracamente acoplados, escaláveis e sem retenção de estado (*stateless*). Segundo o autor, em uma API RESTful, os recursos do sistema são identificados de forma única por URIs (*Uniform Resource Identifiers*) e manipulados por meio dos métodos e verbos padronizados do protocolo HTTP (`GET`, `POST`, `PUT`, `PATCH` e `DELETE`).

A troca de dados entre a camada de apresentação (*front-end*) e o servidor (*back-end*) ocorre por meio do formato JSON (*JavaScript Object Notation*). Conforme explica Flanagan (2020), trata-se de um padrão leve e independente de linguagem para estruturação e intercâmbio de dados, cuja sintaxe baseada em pares de chave-valor facilita o processamento computacional e reduz a sobrecarga de transferência na rede.

3.3.4.2 Aplicações de Página Única (SPA) com ReactJS

Para a construção da interface gráfica, adotou-se o modelo de Aplicação de Página Única (*Single Page Application* – SPA). Ao contrário do modelo tradicional de navegação web, no qual cada ação resulta no recarregamento completo da página pelo servidor, as SPAs carregam uma única estrutura HTML inicial e atualizam dinamicamente apenas os componentes modificados da tela via requisições assíncronas em segundo plano à API.

A implementação do *front-end* utiliza a biblioteca ReactJS. Conforme descrevem Pressman e Maxim (2021), o React fundamenta-se no conceito de arquitetura orientada a

componentes reutilizáveis e no uso de um DOM Virtual (*Virtual DOM*), uma representação em memória da árvore de elementos do navegador; quando o estado da aplicação se altera, o React calcula as diferenças de forma otimizada e atualiza exclusivamente os elementos estritamente necessários na interface, assegurando fluidez e alta performance para os usuários.

3.3.4.3 Microframeworks Python (Flask) e ORM SQLAlchemy

No ecossistema do *back-end*, a escolha da linguagem Python uniu legibilidade de código, concisão e suporte a bibliotecas consolidadas. A construção da API RESTful fundamentou-se no microframework Flask, caracterizado por sua estrutura enxuta, modular e flexível. Diferentemente de *frameworks* monolíticos, o Flask permite integrar apenas os componentes necessários ao projeto, mantendo a aplicação leve e de simples manutenção.

Para a camada de persistência no banco de dados relacional PostgreSQL, empregou-se a biblioteca de Mapeamento Objeto-Relacional (*Object-Relational Mapping* – ORM) SQLAlchemy. Como explica Sommerville (2019), a ORM atua como uma camada de abstração entre o modelo orientado a objetos da aplicação e a estrutura relacional do banco de dados, mitigando o problema da incompatibilidade de impedância (*impedance mismatch*); além de simplificar a construção de consultas, a ORM previne vulnerabilidades críticas de segurança, como ataques de injeção de SQL (*SQL Injection*).

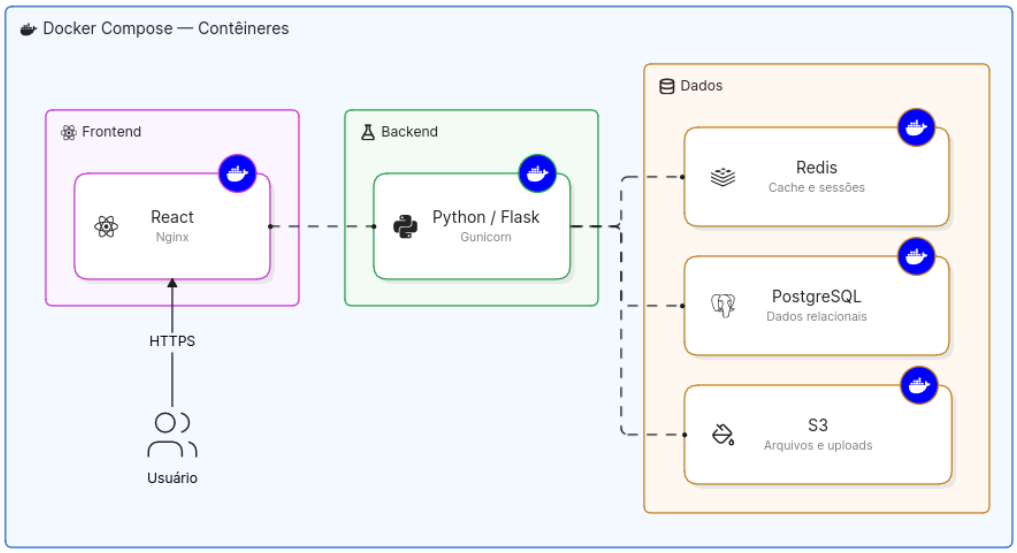
3.3.4.4 Conteinerização de Ambientes com Docker

Para assegurar a reprodutibilidade do ambiente de desenvolvimento e a homogeneidade entre as etapas de teste e execução final, utilizou-se a tecnologia de conteinerização Docker. Segundo Tanenbaum e Bos (2016), ao contrário da virtualização tradicional baseada em hipervisores pesados, os contêineres Docker compartilham o mesmo núcleo (*kernel*) do sistema operacional hospedeiro, empacotando a aplicação e todas as suas dependências em unidades isoladas e de rápida inicialização.

Com o auxílio da ferramenta Docker Compose, definiu-se a orquestração dos múltiplos serviços que compõem a solução a API em Flask, a instância do banco de dados PostgreSQL e o emulador de armazenamento de objetos Kumo (S3) , garantindo a inicialização padronizada do ecossistema por meio de uma única configuração declarativa.

A implementação do sistema proposto adota essa abordagem ao utilizar uma arquitetura baseada no padrão Model-View-Controller (MVC) com o microframework Flask em Python no back-end, banco de dados PostgreSQL para persistência de dados e conteinerização via Docker. Essa estrutura tecnológica assegura escalabilidade, confiabilidade no armazenamento dos relatórios e facilidade na manutenção do software.

Figura 1: Diagrama de Arquitetura Geral do Ecossistema.



3.3.5 Estratégia de Testes

A garantia de qualidade e estabilidade da aplicação fundamentou-se na prática do *Test-Driven Development* (TDD), seguindo o ciclo *Red-Green-Refactor* (BECK, 2002). A suíte de testes automatizados foi construída utilizando o *framework* `pytest`, cobrindo diferentes níveis da arquitetura:

- **Testes Unitários e de Integração com Banco de Dados:** Validação das regras de negócio isoladas no *back-end* e verificação da persistência e integridade referencial das entidades manipuladas via ORM SQLAlchemy no PostgreSQL;
- **Testes de Contrato e API:** Cobertura sistemática dos *endpoints* RESTful para validar o comportamento dos verbos HTTP (POST, GET, PATCH, DELETE). Essa camada assegura a correta validação dos esquemas de requisição (*payloads*), limites de upload e suporte a formatos de arquivo, tratamento de exceções para parâmetros inválidos ou malformados, paginação, idempotência em exclusões lógicas (*soft delete*) e a aplicação estrita do controle de acesso por permissões (RBAC).

4 Especificação de Requisitos

Esta seção apresenta o mapeamento e o detalhamento formal dos requisitos funcionais e não-funcionais do sistema, abrangendo as dimensões lúdica, administrativa e pedagógica da aplicação.

4.1 Requisitos Funcionais (RF)

Tabela 1: Requisitos Funcionais (RF)

ID	Nome	Descrição
RF01	Registro de Feedback Multimídia	O sistema deve permitir que o tutor realize o registro do progresso da criança utilizando texto corrido ou gravação de áudio, associando os dados à data atual e ao perfil do aluno.
RF02	Visualização da Linha do Tempo	O sistema deve exibir, de forma cronológica inversa, todos os registros de texto e áudio associados a um determinado aluno para o acompanhamento dos pais e tutores.
RF03	Controle de Acesso (RBAC)	O sistema deve restringir e validar o acesso às funcionalidades e dados da plataforma de acordo com o perfil atribuído ao usuário autenticado, segregando as permissões em três níveis: Administrador, Professor e Responsável.
RF04	Alocação e Gestão de Turmas	O sistema deve permitir que a administração crie turmas, defina limites de vagas por sala de atividade e vincule os alunos matriculados e seus respectivos tutores a essas turmas.

ID	Nome	Descrição
RF05	Emissão de Relatório Pedagógico	O sistema deve permitir que os tutores preencham e emitam relatórios formais sobre o desempenho lúdico e a evolução pedagógica do aluno com base na metodologia Montessori, consolidando as observações do período.
RF06	Integração de Links do WhatsApp	O sistema deve ser capaz de emitir links diretos para conversas particulares com os responsáveis no WhatsApp (wa.me), gerados com base nos números de telefones cadastrados previamente no sistema.
RF07	Criação de disponibilização de eventos	O sistema deve permitir a criação, visualização, atualização e exclusão lógica de eventos (Reuniões, *Workshops*, Comunicados), restringindo essas operações aos usuários com perfil Administrador. Cada evento contém título, descrição, imagens ilustrativas, data e hora.
RF08	Gerenciamento de Responsáveis	O sistema deve permitir a criação, visualização, atualização e exclusão lógica de registros de responsáveis (pais e tutores legais), validando dados obrigatórios e restrições de vínculo.
RF09	Gerenciamento de Tutores	O sistema deve permitir a criação, visualização, atualização e exclusão lógica de registros de tutores (professores e educadores), atribuindo credenciais, perfil de acesso e vínculos com turmas/alunos.
RF10	Gerenciamento de Alunos	O sistema deve permitir a criação (via aprovação de matrícula), visualização do histórico, atualização cadastral e exclusão lógica com manutenção do histórico pedagógico dos registros de alunos.

4.2 Requisitos Não Funcionais (RNF)

Tabela 2: Requisitos Não-Funcionais (RNF)

ID	Nome	Descrição
RNF01	Usabilidade	A interface de registro de progresso operada pelo tutor deve concluir a inserção e o salvamento dos dados em no máximo três interações na tela.
RNF02	Desempenho	A reprodução dos áudios na linha do tempo deve iniciar em até dois segundos após o acionamento do comando pelos responsáveis, utilizando a tecnologia de transmissão por fluxo contínuo.
RNF03	Compatibilidade	O sistema deve ser compatível com navegadores web modernos (Google Chrome 120+, Apple Safari 16+, Mozilla Firefox 120+) e aplicações móveis (iOS 12+ e Android 8+) mantendo plena funcionalidade em todos os ambientes sem degradação de recursos ou experiência do usuário.
RNF04	Manutenibilidade	O código-fonte deve seguir padrões de documentação técnica e arquitetura modular, permitindo atualizações e correções de componentes individuais sem impacto em outros módulos ou na disponibilidade da plataforma.
RNF05	Segurança e Privacidade	O tráfego de dados e arquivos deve ocorrer via protocolo criptografado HTTPS, e as regras de controle de acesso devem impedir a visualização dos dados por qualquer usuário não autenticado ou sem vínculo direto de tutela com o aluno.
RNF06	Escalabilidade do Banco de Dados	O banco de dados centralizado deve ser estruturado para suportar o crescimento volumétrico histórico, garantindo o armazenamento persistente e a recuperação ágil de relatórios pedagógicos e cadastros sem degradação de tempo de resposta.
RNF07	Disponibilidade	O ecossistema de serviços da plataforma, especialmente os módulos de matrícula online e administrativo, deve manter uma taxa de disponibilidade mínima de 99,5% do tempo de operação.

ID	Nome	Descrição
RNF08	Confiabilidade	O sistema deve implementar backup automático diário dos dados pedagógicos com mecanismo de recuperação garantindo restauração completa em no máximo 4 horas em caso de falha crítica ou perda de dados.
RNF09	Portabilidade	A plataforma deve ser hospedável em ambientes cloud múltiplos (AWS, Microsoft Azure, Google Cloud) com containerização via Docker, permitindo migração entre provedores sem modificação do código-fonte ou perda de funcionalidades.
RNF10	Recuperabilidade e Conformidade	O sistema de autenticação e sessões deve manter registros de auditoria completos por período mínimo de 12 meses para fins de compliance regulatório, rastreabilidade de acessos e investigação de incidentes de segurança.

5 Resultados e Discussão

Neste capítulo são apresentados os resultados alcançados durante o desenvolvimento e a validação do Sistema de Assessoria Lúdica (SAC), abrangendo a modelagem do banco de dados, os diagramas de casos de uso, a infraestrutura de armazenamento e os resultados obtidos na suíte de testes automatizados.

5.1 Diagramas e Modelagem

5.1.1 Modelo Entidade Relacionamento

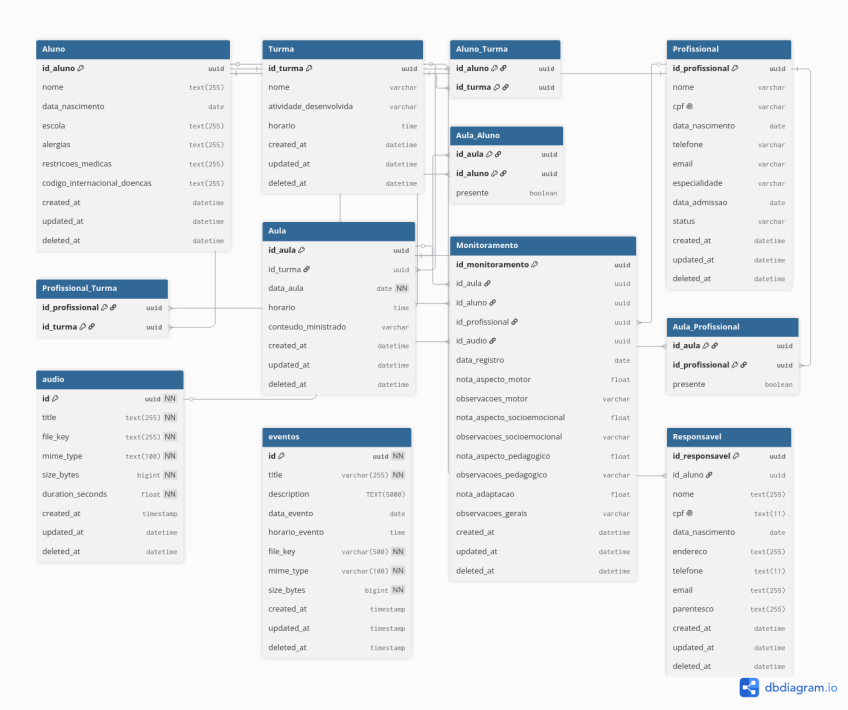
Para garantir o suporte adequado às regras de negócio e a persistência dos dados administrativos e pedagógicos, estruturou-se um Modelo Entidade-Relacionamento (MER) relacional. O esquema do banco de dados organiza-se em três eixos principais de domínio:

- **Núcleo Cadastral e de Gestão de Turmas:** Composto pelas entidades `Aluno`, `Responsavel`, `Profissional` e `Turma`.
  - A entidade `Responsavel` possui vínculo direto (1:N) com a entidade `Aluno`, registrando os dados de contato e parentesco.
  - As tabelas associativas `Aluno_Turma` e `Profissional_Turma` implementam o relacionamento de muitos-para-muitos (N:N) entre alunos, profissionais e suas respectivas turmas de atividades lúdicas.
- **Gestão de Aulas e Frequência:** Centrado na entidade `Aula`, que relaciona-se com as turmas. A presença diária é registrada por meio das tabelas associativas `Aula_Aluno` e `Aula_Profissional`, garantindo o histórico de assiduidade de alunos e tutores.
- **Acompanhamento Pedagógico e Multimídia:** Estruturado pela entidade `Monitoramento` que centraliza as avaliações qualitativas do desenvolvimento infantil ligadas a uma aula, aluno e profissional específicos.
  - A entidade `Monitoramento` contempla métricas e observações para os aspectos *motor*, *sócio-emocional*, *pedagógico* e de *adaptação*, respeitando as diretrizes de observação montessoriana.

– Associa-se opcionalmente à entidade `audios`, permitindo o vínculo de arquivos de voz gravados pelos tutores com armazenamento de metadados (*file_key*, *mime_type*, *size_bytes* e *duration_seconds*).

Todas as tabelas principais utilizam identificadores únicos universais (`UUID`) como chave primária, além de incorporarem campos padronizados de auditoria (`created_at`, `updated_at`) e suporte a exclusão lógica (`deleted_at`). A Figura 2 ilustra o diagrama completo do Modelo Entidade-Relacionamento do sistema.

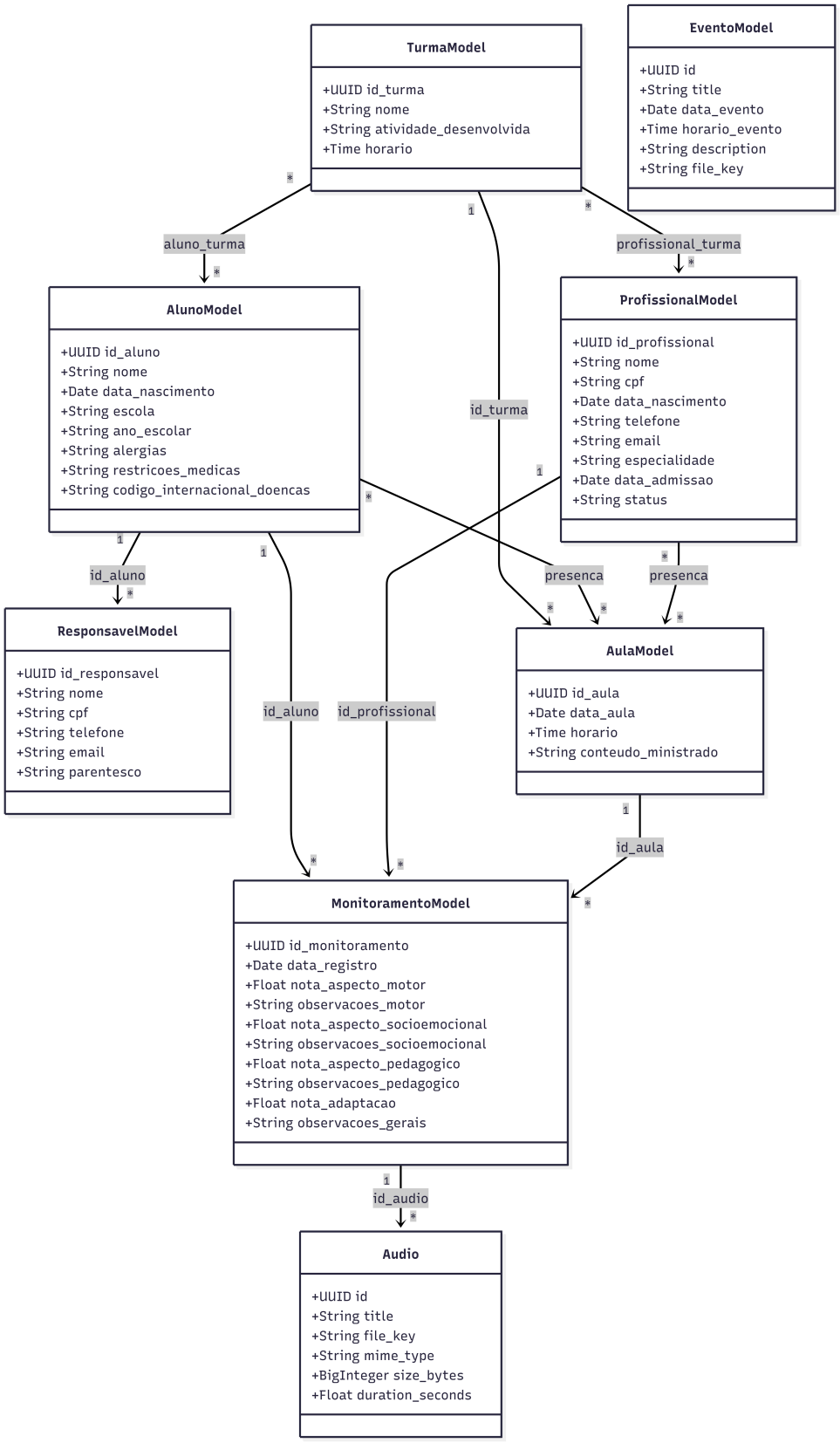
Figura 2: Modelo Entidade Relacionamento (MER) do Sistema.



5.1.2 Diagrama de Classes

Para representar a estrutura estática do sistema e o mapeamento dos conceitos do domínio em componentes orientados a objetos, elaborou-se o Diagrama de Classes. A modelagem abrange as entidades do sistema, seus atributos, métodos e as relações de associação, agregação e herança existentes.

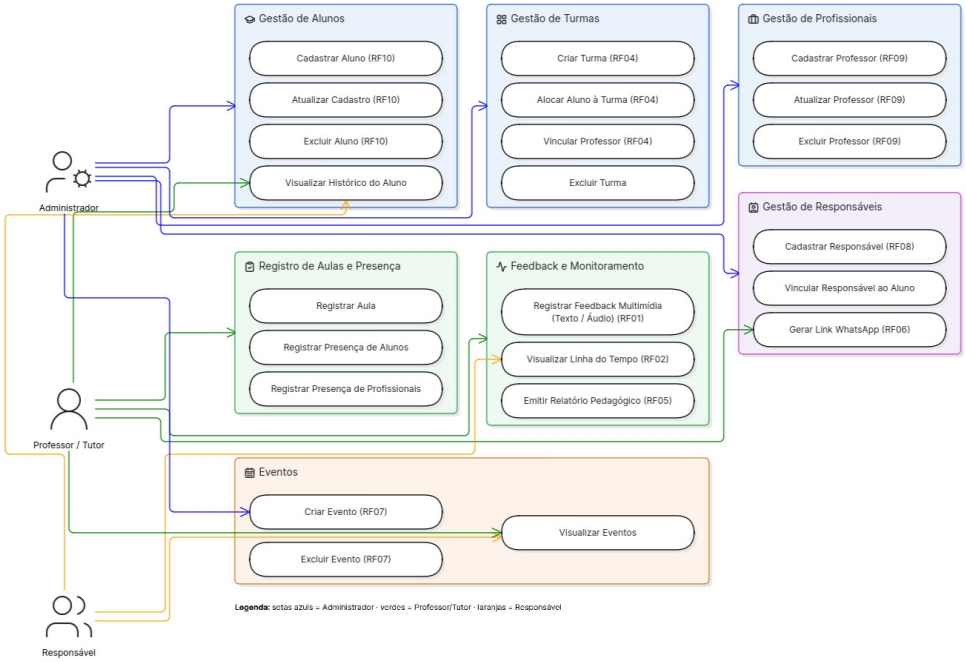
Figura 3: Diagrama de classes com as principais entidades do sistema



5.1.3 Casos de Uso

Para especificar os requisitos funcionais sob a perspectiva dos atores que interagem com a solução, elaborou-se o Diagrama de Casos de Uso. A modelagem delimita as fronteiras do sistema e ilustra as principais funcionalidades e interações disponíveis para cada perfil de usuário.

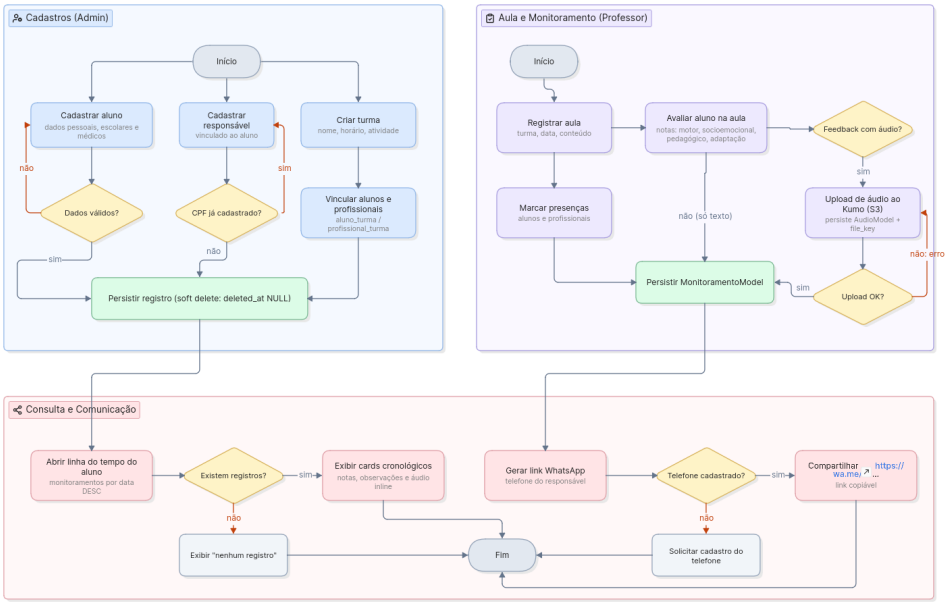
Figura 4: Diagrama de Casos de Uso com algumas das principais funções do sistema



5.1.4 Diagrama de atividade

Para detalhar o fluxo operacional e o comportamento dinâmico dos processos do sistema, elaborou-se o Diagrama de Atividades. A modelagem descreve a sequência de etapas, tomadas de decisão e caminhos de execução necessários para a realização das operações da aplicação.

Figura 5: Diagrama de atividade com algumas das principais atividades do sistema



5.2 Infraestrutura e Armazenamento de Objetos (BLOB)

Para atender aos requisitos RNF03 e RF01, referentes ao upload, compressão e disponibilização de arquivos de mídia (áudios e anexos), adotou-se o uso de ferramentas locais para o armazenamento de objetos do tipo BLOB (*Binary Large Objects*). Por se tratar de uma solução de simples implementação, baixo custo e visando garantir a independência de serviços externos de nuvem durante as etapas de desenvolvimento, testes e execução local containerizada via Docker Compose, utilizou-se a ferramenta **Kumo** para emular localmente o serviço de armazenamento S3. A integração com o *back-end* ocorreu de forma transparente através do SDK `boto3` em Python, permitindo a geração segura de URLs pré-assinadas e a gestão eficiente do ciclo de vida dos arquivos de áudio.

5.3 Garantia de Qualidade e Suíte de Testes Automatizados

Em conformidade com a prática de *Test-Driven Development* (TDD) adotada na metodologia do projeto, a qualidade do código, a conformidade da API e a integridade dos dados foram asseguradas por meio de uma robusta suíte de testes automatizados com o *framework* `pytest`.

Tabela 3: Distribuição e Cobertura da Suíte de Testes Automatizados.

Categoria	**Qtd. Testes**	**Escopo Principal de Validação**
API e Contratos	204 testes	Validação de verbos HTTP, esquemas de *payloads*, regras de acesso (RBAC), paginação e tratamento de exceções.
Banco de Dados	199 testes	Validação de regras de negócio, persistência via ORM SQLAlchemy e integridade referencial no PostgreSQL.
Total	**403 testes**	Cobertura combinada das camadas de *back-end* e persistência.

5.3.1 Considerações Sobre Testes de Ponta a Ponta (E2E)

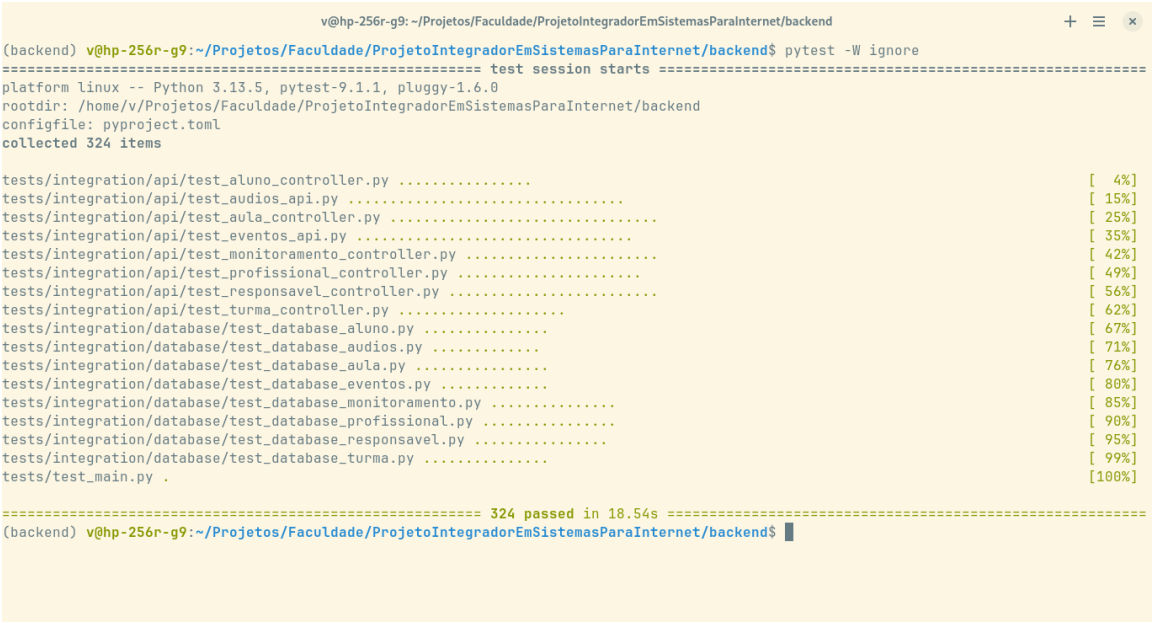
Embora a suíte de testes contemple uma ampla cobertura nas camadas de API e integração com o banco de dados, optou-se fundamentadamente pela não implementação de testes automatizados de Ponta a Ponta (*End-to-End* – E2E) no escopo atual do projeto.

Essa decisão baseou-se nos princípios da Pirâmide de Testes e no Retorno sobre Investimento (*ROI*) da automação de software. Testes E2E exigem a alocação de navegadores automatizados e apresentam alto tempo de execução e manutenção, além de susceptibilidade a falsos negativos decorrentes de latência de renderização da interface.

Dessa forma, priorizou-se garantir a estabilidade e a corretude do sistema na camada de serviços: os **204 testes de contrato e API** asseguraram a validação rigorosa dos *payloads*, regras de negócio e controle de acesso (RBAC), enquanto a validação de formulários no *front-end* foi garantida por esquemas estritos no cliente via Zod. A validação das fluxos de interface foi conduzida por meio de testes funcionais manuais e validações de usabilidade baseadas nos casos de uso modelados.

Na figura abaixo é possível visualizar a execução completa de todos os testes de API e integração com o banco de dados utilizando o `pytest`.

Figura 6: Resultado da execução completa da suíte de testes automatizados no Pytest.



5.4 Testes de Carga

Utilizando a ferramenta **Grafana k6** foram realizadas três baterias de testes nos principais *endpoints* do sistema, mais precisamente das entidades de Responsável, Profissional e Aluno. Cada teste tem como objetivo simular um uso padrão do sistema, realizando operações de listagem, inserção, remoção e atualização das informações.

5.4.1 Ambiente de Execução

Os testes foram executados em um ambiente containerizado utilizando **Docker Compose**, com a seguinte configuração de hardware e stack de serviços:

Item	**Especificação**
CPUs (lógicos)	8
Memória RAM	23 GiB total (14 GiB disponíveis durante teste)
Sistema Operacional / Kernel	Linux (Debian 13) kernel 6.12.96+deb13-amd64
Python	3.13.5

A **stack de serviços** foi composta pelos seguintes componentes:

Serviço	**Imagem**	**Porta**	**Papel**
`db`	`postgres:15-alpine`	5432	Banco de dados relacional
`redis`	`redis`	6379	Cache em memória

Serviço	Imagem	Porta	Papel
`kumo`	`sivchari/kumo:0.26.0`	8080 (→4566)	Storage de arquivos
`web`	build customizado	5000	API Flask

5.4.2 Configuração dos Testes

5.4.2.1 Casos de Carga e Thresholds

Os testes foram estruturados em três cenários de carga, utilizando o executor **ramping-vus** do k6 com três estágios: ramp up (10 segundos até 25% da meta), carga constante e ramp down (10 segundos). A tabela a seguir sintetiza os casos de teste:

Caso	LOAD_CASE	VUs Alvo	Duração da Carga Constante
Baixo	100	100	60s
Médio	1000	1000	120s
Alto	10000	10000	180s

Os **thresholds** de aceitação foram aplicados uniformemente aos três casos:

- **Taxa de falhas**: `http_req_failed` $< 0{,}01$ (máximo 1% de falhas);
- **Latência P95**: `http_req_duration` P95 < 1000 ms.

5.4.2.2 Fluxos de Teste

Cada usuário virtual executava um fluxo completo de **CRUD** (Create, Read, Update, Delete) distribuído entre três recursos principais:

- **Alunos**: POST (criação), GET (listagem), GET (por ID), PUT, PATCH, DELETE;
- **Responsáveis**: POST (com FK de aluno existente), GET (listagem), GET (por ID), PUT, PATCH, DELETE, GET (whatsapp);
- **Profissionais**: POST, GET (listagem), GET (por ID), PUT, PATCH, DELETE.

Os dados foram gerados dinamicamente via helper script (`helpers.js`), incluindo CPF aleatório de 11 dígitos (único por requisição), telefone válido, e busca/criação do aluno base para suportar a chave estrangeira do responsável.

5.4.2.3 Configuração dos Servidores Testados

Bateria 1 Servidor de Desenvolvimento (Werkzeug):
Configuração padrão do Flask com servidor de desenvolvimento:

```
CMD ["flask", "run", "--host", "0.0.0.0", "--debug"]
```

Execução em um único processo sem concorrência real (apenas threads de desenvolvimento).

Bateria 2 Gunicorn (Workers × Threads):
Configuração otimizada com Gunicorn em ambiente containerizado:

Parâmetro	**Valor**
`bind`	`0.0.0.0:5000`
`workers` (WEB_CONCURRENCY)	4
`worker_class`	`gthread`
`threads` (WEB_THREADS)	8
`timeout`	60s
`keepalive`	5s
`max_requests`	2000 (com jitter de 100)

A concorrência total alcançada foi de **32 conexões simultâneas** (4 workers × 8 threads).

5.4.3 Resultados Bateria 1: Werkzeug (Servidor de Desenvolvimento)

Os testes iniciais com o servidor de desenvolvimento (Werkzeug) revelaram limitações severas de escalabilidade. A tabela abaixo sintetiza o desempenho observado:

Métrica	**100 VUs**	**1000 VUs**	**10000 VUs**
Throughput	∼227 req/s	Colapso	Colapso
Latência P95	∼125 ms	60 s (timeout)	60 s (timeout)
Taxa de Falhas	∼0%	16,4%	65,3%

O servidor de desenvolvimento (execução em único processo) não suportou os cenários de 1000+ VUs. A fila de requisições excedeu o timeout padrão de 60 segundos do k6, resultando em falhas massivas. Este comportamento foi esperado, confirmando que o servidor Werkzeug é inadequado para testes de carga reais.

5.4.4 Resultados Bateria 2: Gunicorn (Produção)

5.4.4.1 Resumo Geral

A configuração com Gunicorn (4 workers × 8 threads) demonstrou desempenho significativamente superior. O resumo geral dos três cenários é apresentado abaixo:

Métrica	**100 VUs**	**1000 VUs**	**10000 VUs**
Requisições Processadas	20.900	73.351	31.749
Throughput	248,8 req/s	431,5 req/s	138,0 req/s
Latência P50	∼4 ms	∼780 ms	2256 s
Latência P95	∼9 ms	**∼3,3 s**	∼60 s (timeout)
Latência Máxima	60 ms	∼19 s	∼60 s
Taxa de Sucesso	100%	99,996%	55,1%
Erros Registrados	nenhum	3 (status 0)	∼12.000 timeouts + 9 HTTP 500

Análise dos Cenários:

- **100 VUs**: Desempenho saudável e dentro de todos os thresholds. Latência P95 inferior a 10 ms, taxa de sucesso de 100%. O sistema opera com folga de capacidade;
- **1000 VUs**: Ponto de ruptura de latência, mas mantendo estabilidade funcional. Nenhum timeout crítico (máximo ∼19 s), com 99,996% das requisições respondidas com sucesso. No entanto, a latência P95 (∼3,3 s) ultrapassa o threshold de 1 segundo, indicando **degradação de desempenho**. O throughput estabiliza em ∼431 req/s, sugerindo saturação das 32 conexões do Gunicorn;
- **10000 VUs**: Colapso de estabilidade, com apenas 55,1% de sucesso. Throughput cai abaixo do cenário médio (138 req/s), resultando em ∼12.000 timeouts de conexão e 9 erros HTTP 500 (indicando possível estouro de recursos no Postgres). Embora superior ao Werkzeug (65,3% de falha), o resultado é inaceitável para produção.

5.4.4.2 Latência por Endpoint 100 VUs

Endpoint	**n**	**Média (ms)**	**P50 (ms)**	**P95 (ms)**	**P99 (ms)**	**Máx (ms)**
`GET /alunos/{id}`	4160	5,08	4,71	8,90	13,28	49,51

Endpoint	n	Média (ms)	P50 (ms)	P95 (ms)	P99 (ms)	Máx (ms)
GET/ /responsaveis/ {id}	4160	5,07	4,70	9,11	13,79	50,13
GET/ /profissionais/ {id}	4160	5,34	4,92	9,81	14,81	35,19
GET/ /alunos	2180	3,89	3,49	6,79	10,24	60,49
GET/ /responsaveis	2080	4,39	4,17	8,36	11,34	31,67
GET/ /profissionais	2080	4,26	4,17	8,07	11,13	17,44
GET/ /alunos/ {id}/ responsaveis	1040	2,76	2,36	5,02	7,77	18,33
GET/ /responsaveis/ {id}/ whatsapp	1040	1,65	1,20	3,52	6,23	35,23

Observação: No cenário de 100 VUs, todos os endpoints demonstram latências muito baixas e uniformes, indicando que o sistema não está sob pressão.

5.4.4.3 Latência por Endpoint 1000 VUs

Endpoint	n	Média (ms)	P50 (ms)	P95 (ms)	P99 (ms)	Máx (ms)
GET/ /alunos/ {id}	15154	1042	706	3246	4188	18915
GET/ /responsaveis/ {id}	14485	1134	782	3264	4212	18662
GET/ /profissionais/ {id}	13363	1177	812	3277	4243	18999
GET/ /alunos	8581	1007	675	3185	4165	4792

Endpoint	n	Média (ms)	P50 (ms)	P95 (ms)	P99 (ms)	Máx (ms)
GET/ /responsaveis	7487	1141	787	3378	4237	18951
GET/ /profissionais	6927	1211	845	3342	4234	18936
GET/ /alunos/ {id}/ responsaveis	3687	1105	750	3265	4211	18637
GET/ /responsaveis/ {id}/ whatsapp	3667	1106	736	3280	4194	18955

Taxas de erro: 3 requisições (POST /responsaveis) com status 0, resultando em **99,996% de sucesso**.

Observação: A latência P95 aumentou aproximadamente 200 vezes em relação ao cenário de 100 VUs, consolidando o gargalo nas 32 conexões disponíveis do Gunicorn ($\sim$13 req/s por conexão).

5.4.4.4 Latência por Endpoint 10000 VUs

Endpoint	n	Média (ms)	P50 (ms)	P95 (ms)	P99 (ms)	Máx (ms)
GET/ /alunos	17628	38933	56195	60001	60003	60016
GET/ /alunos/ {id}	11535	33701	30542	60001	60002	60016
GET/ /responsaveis	1160	27640	22149	60001	60001	60007
GET/ /responsaveis/ {id}	660	37008	34460	60001	60001	60014
GET/ /alunos/ {id}/ responsaveis	287	38990	34352	60001	60002	60007

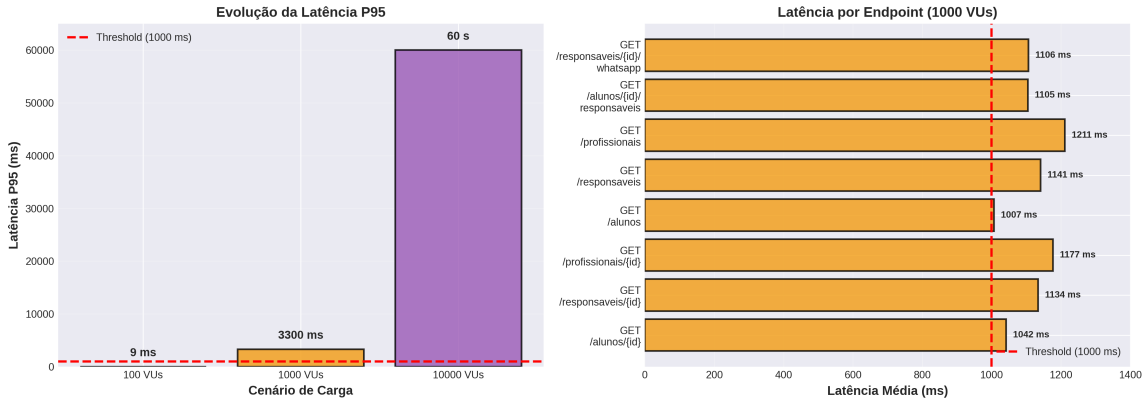
Endpoint	n	Média (ms)	P50 (ms)	P95 (ms)	P99 (ms)	Máx (ms)
GET/ /profissionais	245	48177	60000	60001	60003	60008
GET/ /responsaveis/ {id}/ whatsapp	216	45471	60000	60001	60001	60008
GET/ /profissionais/ {id}	18	20421	18878	35548	35548	50518

5.4.5 Visualização Gráfica dos Resultados

Figura 7: Comparativo de Throughput entre Werkzeug e Gunicorn nos três cenários de carga.

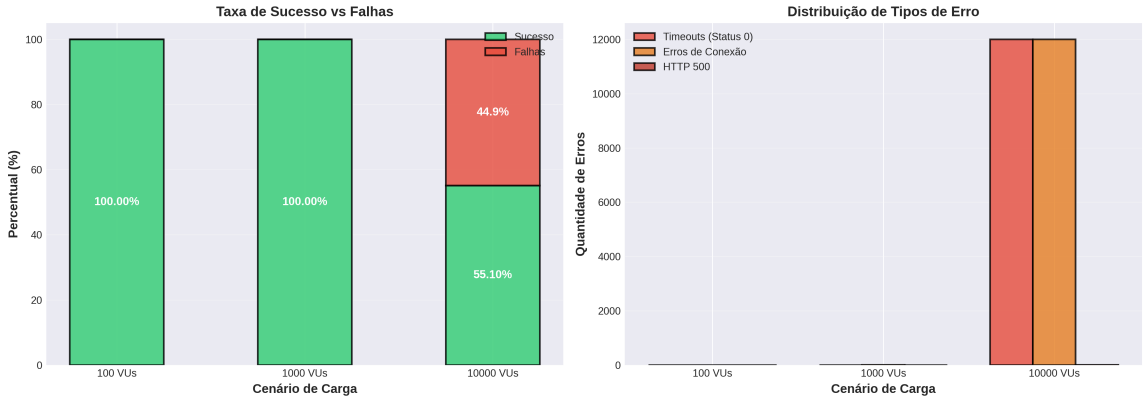
Fonte: Elaborado pelos autores (2026).

Figura 8: Evolução da latência P95 em função da carga (Gunicorn 4 workers × 8 threads).



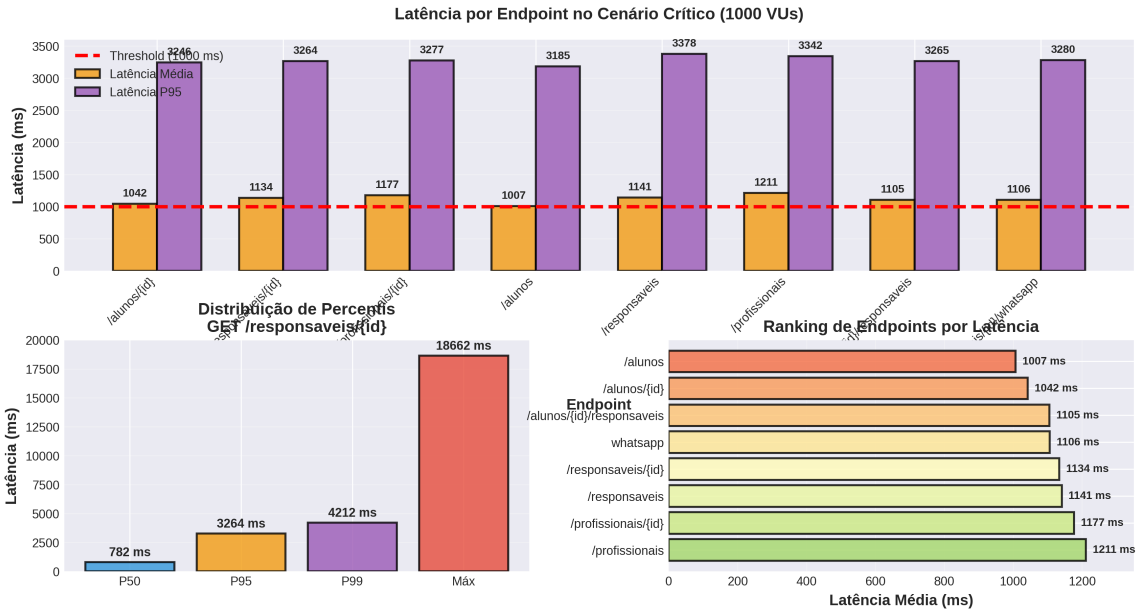
Fonte: Elaborado pelos autores (2026).

Figura 9: Taxa de sucesso e distribuição de erros por cenário.



Fonte: Elaborado pelos autores (2026).

Figura 10: Latência por endpoint no cenário de 1000 VUs (ponto crítico de degradação).



Fonte: Elaborado pelos autores (2026).

5.4.6 Interpretação e Conclusões

5.4.6.1 Capacidade de Suporte

Os resultados indicam os seguintes pontos críticos de desempenho:

- **Até 100 VUs**: Sistema opera em regime saudável, com latências aceitáveis (P95 $<$ 10 ms) e sem degradação. Recomenda-se este cenário como carga padrão de operação;
- **100 a 1000 VUs**: Zona de degradação aceitável. O sistema mantém funcionalidade (99,996% de sucesso) mas com latência P95 elevada ($\sim$3,3 s). O threshold de 1 segundo é superado, indicando necessidade de otimizações;
- **Acima de 1000 VUs**: Colapso progressivo do sistema. Em 10000 VUs, apenas 55,1% das requisições são processadas com sucesso. Não é recomendado operar nesta faixa sem escalamento horizontal ou otimizações críticas.

5.4.6.2 Gargalos Identificados

A análise dos resultados revelou os seguintes gargalos:

1. **Concorrência limitada do Gunicorn**: 32 conexões simultâneas (4 workers $\times$ 8 threads) constituem o gargalo primário. Em 1000 VUs, cada conexão processa $\sim$13 requisições por segundo, criando fila de espera;

2. **Custo de processamento por requisição**: As operações CRUD incluem validações, I/O com banco de dados e possíveis lazy loads. Auditar N+1 queries seria prioritário;

3. **Limite de conexões do Postgres**: No cenário de 10000 VUs, os 9 erros HTTP 500 sugerem provável estouro do pool de conexões do banco de dados;

4. **Ausência de cache**: Endpoints de leitura (GET) poderiam beneficiar-se de Redis, já presente no stack, com TTL apropriado.

5.4.6.3 Recomendações de Melhoria

Para incrementar a capacidade de suporte do sistema, recomenda-se (em ordem de prioridade):

1. **Aumentar concorrência do Gunicorn**: Re-rodar o cenário de 1000 VUs com `WEB_CONCURRENCY=8` e `WEB_THREADS=8` (64 conexões) para validar redução proporcional da latência P95;

2. **Auditoria de N+1 queries**: Revisar os serviços de Alunos, Responsáveis e Profissionais para identificar lazy loads em laços, aplicando `joinedload`/`selectinload` do SQLAlchemy;

3. **Otimizar índices do Postgres**: Validar índices nas colunas filtradas (`cpf`, `email`, `deleted_at`, chaves estrangeiras);

4. **Implementar caching com Redis**: Endpoints de leitura (GET listas) podem cache com TTL curto, aproveitando o serviço Redis já presente;

5. **Investigar erros HTTP 500**: Conferir logs do Gunicorn e Postgres no cenário de 10000 VUs para confirmar estouro de conexões ou deadlocks;

6. **Escalamento horizontal**: Para suportar além de 1000 VUs, considerar múltiplas instâncias da API atrás de um balanceador de carga (traefik/nginx).

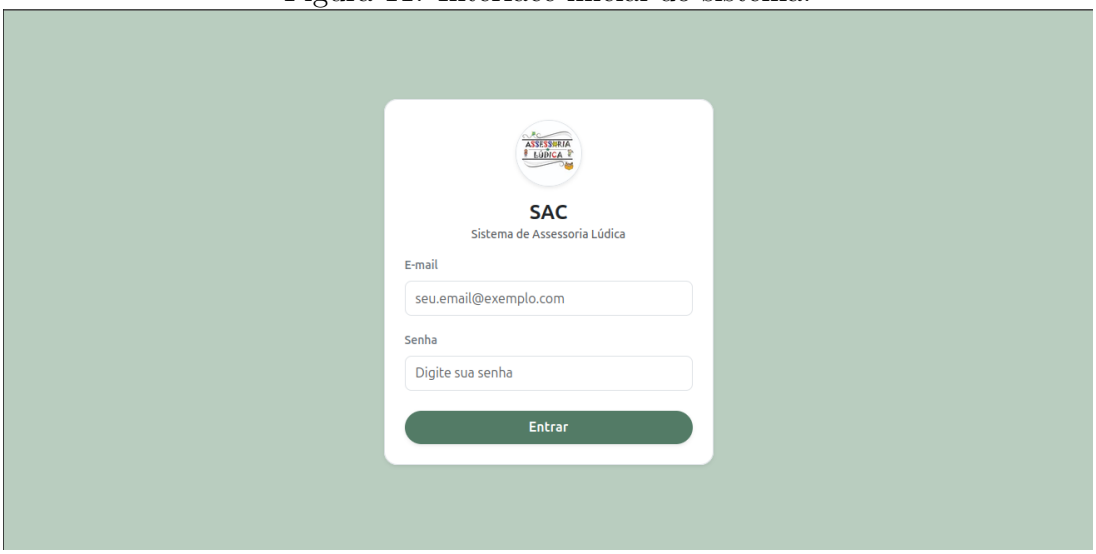
Em conclusão, o sistema demonstra desempenho adequado para operação em cenários de até **100 VUs simultâneos**. Incrementos de carga até 1000 VUs são toleráveis do ponto de vista funcional (mantendo sucesso $> 99\%$), porém com degradação visível de latência. Cargas acima de 1000 VUs exigem otimizações substanciais ou escalamento horizontal para viabilização em produção.

5.5 Interfaces da Aplicação

Para evidenciar os resultados práticos obtidos com a implementação do sistema, apresentam-se a seguir as principais interfaces desenvolvidas para os módulos da plataforma.

O primeiro contato do usuário com a aplicação ocorre no momento do login. A depender das credenciais inseridas, o usuário pode acessar o sistema como Administrador, Tutor ou Responsável.

Figura 11: Interface inicial do sistema.

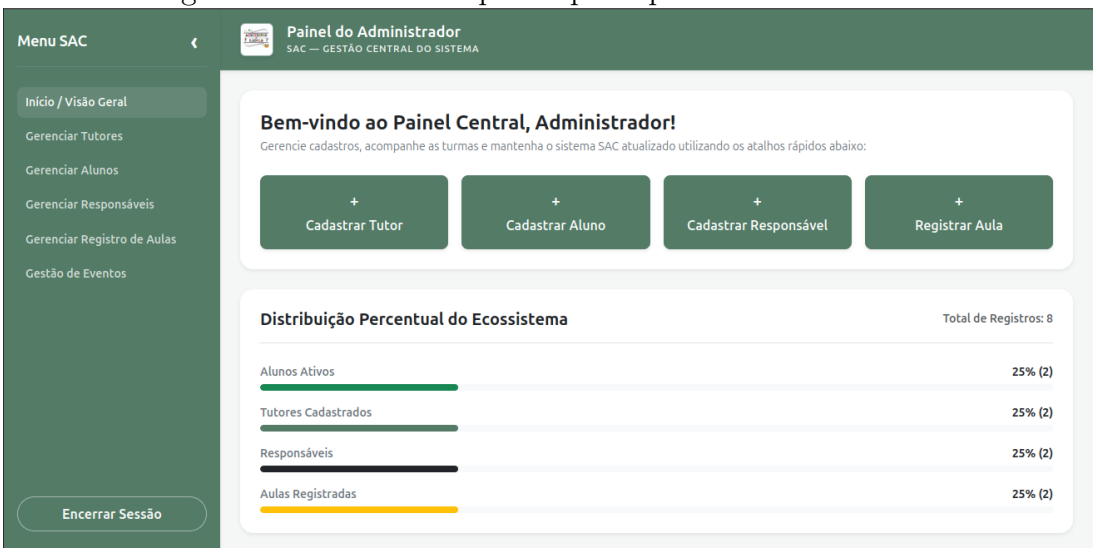


Fonte: Elaborado pelos autores (2026).

5.5.1 Módulo do Administrador

No painel do administrador, existem vários botões de acesso rápido para o cadastro de novos tutores, alunos e responsáveis no sistema. Além disso, é possível visualizar rapidamente diversas estatísticas relevantes, como o total de alunos ativos na assessoria, tutores, responsáveis e aulas registradas.

Figura 12: Interface do painel principal do administrador.

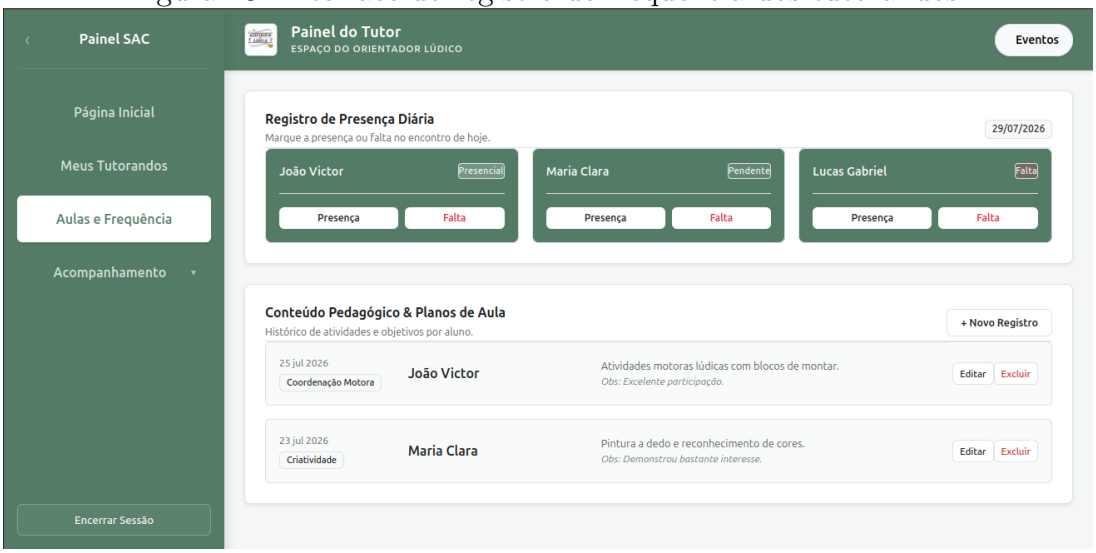


Fonte: Elaborado pelos autores (2026).

5.5.2 Módulo do Tutor

Do lado do tutor, é possível visualizar os alunos (tutorandos) disponíveis para acompanhamento, bem como realizar o registro de aulas e da frequência dos discentes.

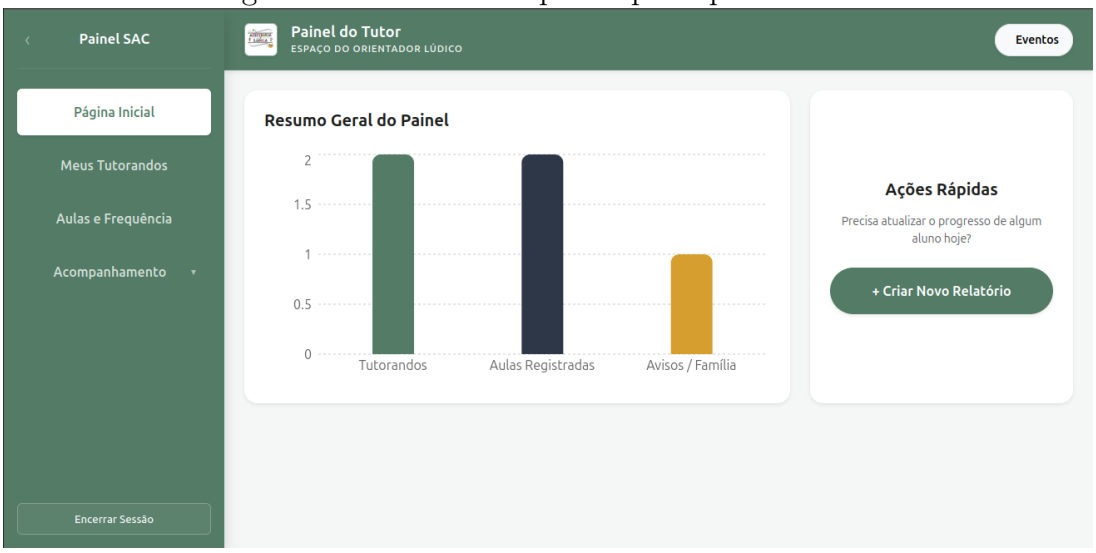
Figura 13: Interface de registro de frequência dos tutorandos.



Fonte: Elaborado pelos autores (2026).

Na página inicial do tutor, é possível obter um gráfico resumido das principais estatísticas do sistema, como o número total de tutorandos acompanhados, o número de aulas registradas e a quantidade de avisos dos familiares. Além disso, é possível realizar um registro rápido de relatório por meio do botão **Criar Novo Relatório**.

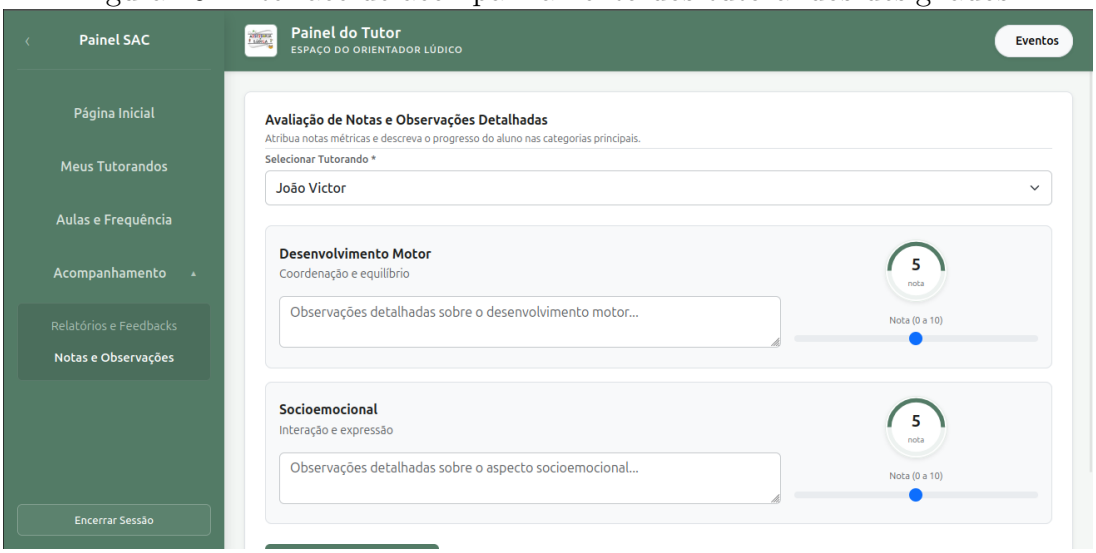
Figura 14: Interface do painel principal do tutor.



Fonte: Elaborado pelos autores (2026).

No painel lateral, é possível realizar alguns registros de notas no sistema. A plataforma avalia o desenvolvimento motor e o desenvolvimento socioemocional com notas de 0 a 10, contando também com campos de texto livre para possíveis observações.

Figura 15: Interface de acompanhamento dos tutorandos designados.

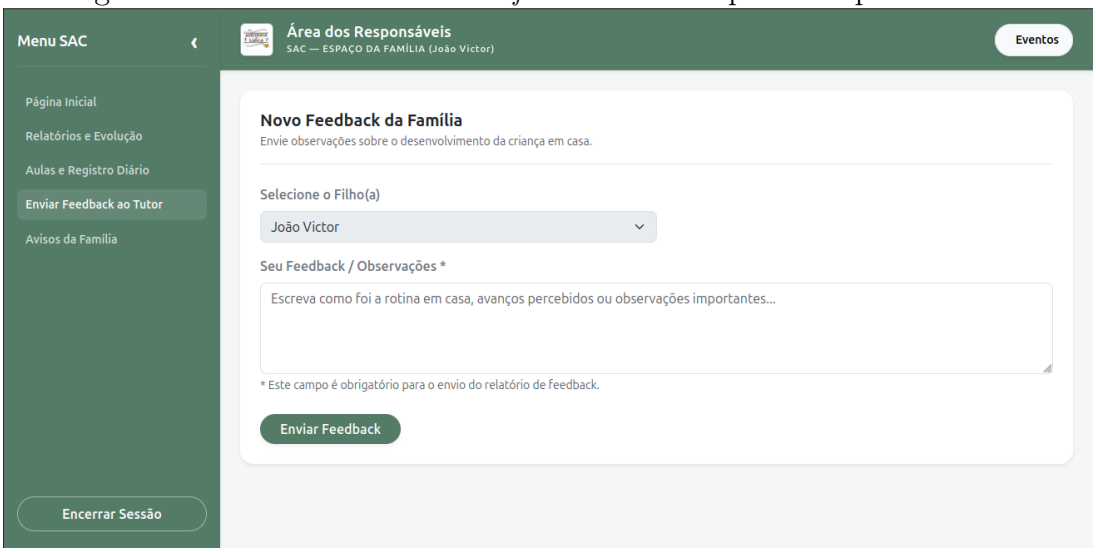


Fonte: Elaborado pelos autores (2026).

5.5.3 Módulo do Responsável

Ao acessar o sistema como responsável, é possível realizar diversas tarefas, tais como: acompanhar a evolução do aluno, visualizar os respectivos relatórios emitidos pelos tutores, realizar o registro diário de aulas e enviar feedbacks e avisos para os profissionais.

Figura 16: Interface de envio de *feedbacks* do responsável para o tutor.



Fonte: Elaborado pelos autores (2026).

Semelhante às outras telas vistas anteriormente, o painel do responsável permite visualizar rapidamente o número de relatórios, aulas e feedbacks realizados pelos profissionais da assessoria. Além disso, também é possível enviar um feedback rápido para o tutor, bem como realizar o registro de momentos lúdicos em casa no menu localizado abaixo do gráfico.

Figura 17: Interface do painel principal do responsável.

Fonte: Elaborado pelos autores (2026).

6 Considerações Finais

O desenvolvimento do Sistema de Assessoria Lúdica (SAC) alcançou os objetivos propostos ao projetar e implementar uma solução tecnológica capaz de automatizar processos administrativos e sistematizar o acompanhamento pedagógico orientado pela Metodologia Montessori. A arquitetura construída assegurou a separação de responsabilidades, o armazenamento persistente e seguro de mídias e relatórios, e a validação contínua das regras de negócio por meio de testes automatizados.

6.1 Trabalhos Futuros

Como **trabalhos futuros**, pretende-se aprimorar a responsividade e a usabilidade da interface em dispositivos móveis, visando otimizar a experiência de uso dos tutores no ambiente operacional. Ademais, prevê-se a realização de testes formais de usabilidade e aceitação com os usuários finais (tutores e pais), bem como a expansão total da cobertura de testes. Almejamos também realizar o *deploy* e a implantação da plataforma em um ambiente real de produção. Para dar suporte a essa infraestrutura, planeja-se a estruturação de uma *pipeline* de CI/CD (*Continuous Integration / Continuous Deployment*), garantindo a execução automatizada da suíte de testes e a entrega contínua de atualizações de forma segura e ágil.

No escopo de evolução funcional, almeja-se a implementação da transcrição automática dos áudios pedagógicos para texto e a integração de um serviço para envio de notificações automatizadas via WhatsApp. Por fim, como compromisso ético e legal inerente ao tratamento de dados de menores de idade, estabelece-se como meta a adequação formal do sistema às diretrizes da Lei Geral de Proteção de Dados (LGPD – Lei nž 13.709/2018), mediante a implementação de políticas de privacidade transparentes, gestão explícita do consentimento dos responsáveis e rotinas para a garantia dos direitos dos titulares de dados.

Referências

ALMEIDA, Carlos Eduardo. **Sistemas de informação aplicados à gestão escolar**. São Paulo: Ed. Blucher, 2022.

BECK, Kent. **Test-Driven Development: by example**. Boston: Addison-Wesley, 2002.

BRASIL. **Lei nž 13.709, de 14 de agosto de 2018**. Lei Geral de Proteção de Dados Pessoais (LGPD). Brasília, DF: Diário Oficial da União, 2018.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation, University of California, Irvine, 2000.

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 7. ed. Porto Alegre: Bookman, 2020.

FOUCAULT, Michel. **Vigiar e Punir: o nascimento da prisão**. Petrópolis: Vozes, 1977.

FREIRE, Paulo. **Pedagogia da autonomia: saberes necessários à prática educativa**. Rio de Janeiro: Paz e Terra, 1997.

FREIRE, Paulo. **Pedagogia do oprimido**. Rio de Janeiro: Paz e Terra, 2019.

MONTESSORI, Maria. **Mente absorvente**. Rio de Janeiro: Nórdica, 1949.

MONTESSORI, Maria. **Pedagogia Científica: A Descoberta da Criança**. Campinas: Kíron, 2013.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: uma abordagem profissional**. 9. ed. Porto Alegre: AMGH, 2021.

PROJECT MANAGEMENT INSTITUTE (PMI). **A Guide to the Project Management Body of Knowledge (PMBOK Guide)**. 7th ed. USA: Project Management Institute, 2021.

SANTOS, Mariana Aranha; LIMA, Ricardo Fernandes. A importância da documentação pedagógica na educação infantil. **Revista Brasileira de Estudos Pedagógicos**, v. 101, n. 258, p. 312-330, 2020.

SILVA, João Pedro; OLIVEIRA, Maria Cláudia. Comunicação escola-família intermediada por tecnologias digitais. **Educação & Sociedade**, v. 42, e23910, 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson Education do Brasil, 2019.

TANENBAUM, Andrew S.; BOS, Herbert. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson Education do Brasil, 2016.