**UNIVERSIDADE PAULISTA**

**ICET - INSTITUTO DE CIÊNCIAS EXATAS E TECNOLOGIA**

**CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**PROJETO INTEGRADO MULTIDISCIPLINAR PIM II**

# DESENVOLVIMENTO DE UM SISTEMA ACADÊMICO COLABORATIVO COM APOIO DE IA



| Nomes | R.A |
|---|---|
| Arthur de Lima Ferreira | R661881 |
| Felipe Augusto Silva de Faria | H719BH9 |
| Gabriel de Sousa Ferreira | R869067 |
| Gabrielle Valéria da Silva Souza | R869DD5 |
| Santiago dos Santos Pacheco | R8681C9 |
| Vinícius Machado de Carvalho | R870HA8 |



**SÃO JOSÉ DOS CAMPOS – SP**

**NOVEMBRO/2025**

| Nomes | R.A |
|---|---|
| Arthur de Lima Ferreira | R661881 |
| Felipe Augusto Silva de Faria | H719BH9 |
| Gabriel de Sousa Ferreira | R869067 |
| Gabrielle Valéria da Silva Souza | R869DD5 |
| Santiago dos Santos Pacheco | R8681C9 |
| Vinícius Machado de Carvalho | R870HA8 |


# DESENVOLVIMENTO DE UM SISTEMA ACADÊMICO COLABORATIVO COM APOIO DE IA

Projeto Integrado Multidisciplinar (PIM) desenvolvido como exigência parcial dos requisitos obrigatórios à aprovação semestral no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da UNIP (Universidade Paulista), orientado pelo corpo docente do curso.

**SÃO JOSÉ DOS CAMPOS – SP**

**NOVEMBRO/2025**

## RESUMO

O presente trabalho consistiu no desenvolvimento de um sistema acadêmico colaborativo com apoio de Inteligência Artificial (IA), voltado para a centralização e modernização da gestão escolar. O projeto teve como objetivo principal integrar, em uma única plataforma, o controle de alunos, professores, turmas e notas, promovendo eficiência administrativa e sustentabilidade por meio da substituição de relatórios impressos por relatórios digitais automatizados. A fundamentação teórica baseou-se em conceitos de Engenharia de Software Ágil (Scrum), Programação Estruturada em C, Estruturas de Dados em Python, Redes de Computadores e Inteligência Artificial Aplicada, aliando teoria e prática no contexto educacional. A metodologia adotada foi de caráter exploratório e descritivo, com aplicação da metodologia ágil Scrum para o planejamento e acompanhamento das sprints. O sistema foi implementado em arquitetura cliente-servidor, utilizando C para o backend e Python para a interface e geração de relatórios, incluindo uma IA Manual (offline) e uma IA generativa com a API Gemini (online). Os testes realizados confirmaram o correto funcionamento dos módulos e a integração entre as linguagens, demonstrando estabilidade e precisão na análise de dados. Concluiu-se que o sistema atendeu plenamente aos objetivos propostos, oferecendo uma solução tecnológica eficiente, sustentável e coerente com os princípios da educação digital.



**Palavras-chave:** Sistema acadêmico; Inteligência Artificial; Engenharia de Software; Sustentabilidade; Scrum.

**SUMÁRIO**

# 1 INTRODUÇÃO

A evolução tecnológica tem transformado profundamente o cenário educacional, ampliando o acesso à informação e promovendo novos métodos de ensino e gestão. Entretanto, muitas instituições de ensino ainda enfrentam dificuldades na centralização de dados acadêmicos e no uso eficiente de ferramentas digitais.

A ausência de sistemas integrados resulta em falhas administrativas**,** duplicidade de informações e dificuldade de acompanhamento do desempenho estudantil**,** comprometendo a eficiência pedagógica.

Diante dessa realidade, o presente projeto propõe o desenvolvimento de um sistema acadêmico colaborativo com apoio de Inteligência Artificial (IA)**,** capaz de gerenciar turmas, alunos, professores e atividades em uma única plataforma. O sistema também busca modernizar a gestão escolar, aplicando práticas de Engenharia de Software Ágil (Scrum) e promovendo a sustentabilidade digital por meio da substituição de relatórios impressos por relatórios digitais automatizados. Além disso, o projeto foi desenvolvido em arquitetura cliente-servidor, integrando módulos desenvolvidos em C (backend) e Python (frontend)**,** com suporte a rede local (LAN) e análises inteligentes realizadas tanto por uma IA Manual (offline) quanto pela API Gemini (Google)**.**

## 1.1 Justificativa

O sistema acadêmico colaborativo foi desenvolvido para centralizar e modernizar a gestão escolar, integrando alunos, professores e administradores em uma única plataforma. Muitas instituições ainda utilizam métodos manuais ou descentralizados, o que gera falhas de controle e retrabalho.

A proposta apresenta uma solução inteligente e sustentável, com apoio de Inteligência Artificial para análise de desempenho e geração de relatórios digitais, substituindo processos impressos e promovendo a educação ambiental e o uso eficiente da tecnologia no ambiente acadêmico.

## 1.2 Objetivo Geral

Desenvolver um sistema acadêmico colaborativo, com apoio de Inteligência Artificial, que possibilite o gerenciamento integrado de turmas, alunos, aulas e atividades,

gerando relatórios digitais automatizados e operando em rede local (cliente-servidor), utilizando princípios da Engenharia de Software Ágil (Scrum).

### 1.3 Objetivos Específicos

Para alcançar o propósito central deste trabalho, foi necessário detalhar o objetivo geral em ações mais detalhadas. Assim, os objetivos específicos foram definidos com o intuito de orientar o desenvolvimento do projeto, assegurando uma abordagem clara e organizada.

- Aplicar a metodologia ágil Scrum para planejar e acompanhar as etapas do desenvolvimento, por meio de sprints, backlog e revisões.
- Implementar módulos críticos em C estruturado, utilizando estruturas de decisão, repetição, funções e manipulação de arquivos .csv para persistência de dados.
- Desenvolver, em Python, a interface gráfica e módulos de exibição de relatórios, integrando com os arquivos manipulados pelo backend em C.
- Aplicar técnicas de Inteligência Artificial Manual (offline) e IA generativa com API Gemini (online) para análise de desempenho acadêmico e geração de relatórios personalizados.
- Modelar o sistema por meio de diagramas UML (casos de uso, classes e sequência) e Diagramas de Fluxo de Dados (DFDs), representando os processos de entrada e saída de dados.
- Estruturar o sistema em arquitetura cliente-servidor, permitindo acesso multiusuário em ambiente de rede local (LAN).
- Adotar estratégias de sustentabilidade digital, substituindo relatórios impressos por relatórios digitais automatizados e mantendo toda a documentação em formato eletrônico.

### 1.4 Metodologia Aplicada

O projeto foi desenvolvido com base na metodologia ágil Scrum, adotando uma abordagem iterativa e incremental. As atividades foram organizadas em sprints, com planejamento, execução, revisão e retrospectiva a cada ciclo. Durante o processo, o Product Backlog foi constantemente atualizado, garantindo alinhamento entre os objetivos técnicos e pedagógicos.

A pesquisa teve caráter aplicado, exploratório e descritivo, combinando referências teóricas e experimentação prática. As linguagens C e Python foram utilizadas de forma integrada, o C responsável pelo processamento e armazenamento dos dados, e o Python pela interface gráfica e execução das rotinas inteligentes.

O sistema foi modelado utilizando diagramas UML e DFDs, estruturado em arquitetura cliente-servidor, testado em ambiente de rede local simulada e validado por meio de casos de teste funcionais e integrados. Por fim, foram incorporadas práticas de sustentabilidade digital e educação ambiental, com o uso exclusivo de documentação e relatórios eletrônicos, eliminando impressões e reduzindo o impacto ambiental do desenvolvimento.

# 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos, métodos e fundamentos teóricos que sustentaram o desenvolvimento do sistema acadêmico colaborativo.

## 2.1 Programação Estruturada em C

A linguagem C, criada por Dennis Ritchie na década de 1970 nos laboratórios Bell, é considerada uma das mais influentes no campo da computação, servindo de base para diversas linguagens modernas, como C++, C# e Java (KERNIGHAN; RITCHIE, 1988). Por sua flexibilidade, eficiência e proximidade com o hardware, tornou-se amplamente utilizada tanto em sistemas operacionais quanto em softwares de aplicação.

No contexto educacional, a utilização de C possibilita compreender conceitos fundamentais de programação estruturada, como decisões, repetições, modularização, manipulação de arquivos e estruturas de dados, contribuindo para a formação lógica e para a prática de boas técnicas de desenvolvimento.

### Estruturas de Decisão e Repetição

As estruturas de decisão permitem que o programa execute diferentes blocos de instruções de acordo com condições lógicas. Os comandos if, else e switch são amplamente empregados para controlar o fluxo do programa (DEITEL; DEITEL, 2016).

Já as estruturas de repetição possibilitam a execução iterativa de instruções, reduzindo redundâncias no código e facilitando operações sobre conjuntos de dados. Os laços for, while e do-while representam os principais mecanismos de iteração em C, sendo essenciais para a construção de algoritmos eficientes (JOYCE, 2015).

### Funções e Modularização

A modularização é um dos pilares da programação estruturada, permitindo dividir um programa em funções menores e independentes. Isso facilita a manutenção, a reutilização de código e o trabalho colaborativo em projetos maiores. Em C, as funções podem ser declaradas antes ou depois da função principal main, além de poderem ser

armazenadas em arquivos separados, promovendo organização e clareza (KERNIGHAN; RITCHIE, 1988).

Segundo Dale, Weems e Headington (2017), a utilização de funções em C é essencial para a compreensão da lógica de decomposição de problemas e da abstração de procedimentos, habilidades fundamentais para o desenvolvimento de sistemas escaláveis.

**Manipulação de Arquivos**

A manipulação de arquivos em C possibilita a persistência de dados, garantindo que informações sejam salvas mesmo após a finalização do programa. As operações básicas incluem abertura (fopen), leitura (fscanf, fgets), escrita (fprintf, fputs) e fechamento (fclose) de arquivos (DEITEL; DEITEL, 2016).

Essa funcionalidade é especialmente importante em sistemas acadêmicos, nos quais dados de alunos, turmas e atividades precisam ser armazenados e recuperados com confiabilidade. Como destaca Prata (2015), a manipulação de arquivos em C fornece uma base sólida para compreender o funcionamento de bancos de dados e sistemas de persistência em níveis mais avançados.

**Estruturas (Structs) e Alocação Dinâmica**

As structs permitem agrupar diferentes tipos de dados em uma única entidade lógica, representando registros mais próximos da realidade do problema modelado. Essa característica é essencial para sistemas que necessitam manipular informações complexas, como dados acadêmicos compostos por nomes, notas e frequências (JOYCE, 2015).

A alocação dinâmica de memória, por sua vez, é realizada por meio das funções malloc, calloc, realloc e free, permitindo ao programa reservar e liberar memória durante a execução. Esse recurso garante flexibilidade e eficiência, especialmente em situações em que a quantidade de dados não é conhecida previamente (PRATA, 2015).

Segundo Dale, Weems e Headington (2017), a combinação de structs e alocação dinâmica é fundamental para o desenvolvimento de sistemas que lidam com grandes volumes de dados e exigem manipulação eficiente de memória.

## 2.2 Engenharia de Software Ágil

A Engenharia de Software Ágil representa uma abordagem moderna para o desenvolvimento de sistemas, buscando responder de maneira rápida e eficiente às constantes mudanças de requisitos e necessidades dos usuários. Diferente dos modelos tradicionais, que priorizam planejamento rígido e documentação extensa, as metodologias ágeis favorecem a colaboração, a adaptação contínua e entregas incrementais de valor (PRESSMAN; MAXIM, 2021).

De acordo com Sommerville (2019), a agilidade no desenvolvimento de software está diretamente relacionada à capacidade de lidar com incertezas e de incorporar melhorias contínuas, mantendo a qualidade do produto final.

### Conceitos Fundamentais

A agilidade em engenharia de software baseia-se no Manifesto Ágil, publicado em 2001, que estabelece quatro valores principais:

1. Indivíduos e interações mais que processos e ferramentas.
2. Software em funcionamento mais que documentação abrangente.
3. Colaboração com o cliente mais que negociação de contratos.
4. Responder a mudanças mais que seguir um plano (BECK et al., 2001).

Esses valores se desdobram em princípios que orientam práticas como comunicação efetiva, simplicidade no design, entregas frequentes e foco contínuo na satisfação do usuário final.

### Metodologia Scrum

Entre as metodologias ágeis, o Scrum é uma das mais aplicadas em projetos de software. Segundo Schwaber e Sutherland (2020), o Scrum organiza o trabalho em

sprints (ciclos curtos de desenvolvimento, geralmente de 1 a 4 semanas), nas quais uma parte funcional do sistema é planejada, implementada e entregue.

O Scrum possui três papéis principais:

- Product Owner: responsável pelo backlog do produto e pelo alinhamento com as necessidades do cliente.
- Scrum Master: garante a aplicação correta da metodologia e remove impedimentos do time.
- Development Team: equipe responsável por desenvolver o produto de forma colaborativa.

Os principais eventos do Scrum incluem o Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective, que estruturam a comunicação e o acompanhamento do progresso.

**Levantamento de Requisitos (Funcionais e Não Funcionais)**

O levantamento de requisitos é etapa essencial em Engenharia de Software, uma vez que define o que o sistema deve fazer (requisitos funcionais) e como o sistema deve se comportar (requisitos não funcionais).

- Requisitos Funcionais (RF): descrevem funcionalidades específicas, como cadastro de alunos, registro de aulas ou geração de relatórios acadêmicos.
- Requisitos Não Funcionais (RNF): estabelecem restrições de desempenho, segurança, usabilidade e confiabilidade (SOMMERVILLE, 2019).

No contexto do sistema acadêmico colaborativo, os requisitos foram definidos a partir da análise das necessidades da instituição de ensino e da integração com práticas de sustentabilidade e Inteligência Artificial.

**Artefatos de Engenharia de Software**

Durante o desenvolvimento ágil, diversos artefatos são produzidos para apoiar a documentação e comunicação do projeto. No Scrum, destacam-se:

- Product Backlog: lista priorizada de funcionalidades e melhorias desejadas.
- Sprint Backlog: conjunto de tarefas selecionadas para uma sprint específica.
- Incremento: versão funcional do sistema entregue ao final de cada sprint.

Além disso, no campo da Engenharia de Software, utilizam-se diagramas UML (casos de uso, classes, sequência) e documentos de requisitos (como o SRS – Software Requirement Specification) para garantir clareza, consistência e rastreabilidade das funcionalidades do sistema (PRESSMAN; MAXIM, 2021).

## 2.3 Estruturas de Dados em Python

O Python consolidou-se como uma das linguagens mais utilizadas no mundo, destacando-se por sua simplicidade, legibilidade e ampla comunidade de desenvolvedores. Criada por Guido van Rossum em 1991, a linguagem possui sintaxe clara e suporte a múltiplos paradigmas, sendo aplicada em ciência de dados, desenvolvimento web, automação, inteligência artificial e ensino de programação (RAMALHO, 2015).

No contexto deste projeto, o Python é empregado para manipulação de dados, integração com módulos em C, geração de relatórios e construção de interfaces de visualização. Sua flexibilidade e vasta biblioteca padrão tornam-no uma ferramenta essencial para complementar a programação estruturada e possibilitar recursos avançados.

### Estruturas de Decisão e Repetição

As estruturas de decisão em Python (if, elif, else) permitem a construção de fluxos condicionais de maneira simples e intuitiva. Já as estruturas de repetição, como os laços for e while, possibilitam a execução de blocos de código de forma iterativa, sendo amplamente utilizadas na manipulação de coleções de dados, como listas, tuplas e dicionários (LUTZ, 2013).

De acordo com Guttag (2016), a clareza e a simplicidade da sintaxe de Python facilitam o aprendizado de conceitos fundamentais de lógica e algoritmos, o que a torna uma linguagem adequada para projetos acadêmicos e prototipagem rápida.

### Manipulação de Dados e Integração com C

Uma das vantagens do Python é sua capacidade de manipular dados de forma eficiente, utilizando estruturas como listas, conjuntos, dicionários e tuplas, que oferecem flexibilidade e operações nativas de busca, filtragem e ordenação.

Além disso, o Python pode se integrar com programas em C por meio de bibliotecas como ctypes e Cython, permitindo a execução de rotinas críticas em baixo nível, ao mesmo tempo, em que mantém a simplicidade de sua sintaxe em alto nível. Essa integração é relevante em sistemas híbridos, nos quais o desempenho de C é combinado com os recursos de abstração e produtividade de Python (BEAZLEY, 2009).

### Relatórios e Visualizações (Interface)

A geração de relatórios digitais e visualizações é um dos pontos centrais do uso de Python neste projeto. Bibliotecas como Pandas e Matplotlib permitem processar dados acadêmicos, gerar gráficos e exportar relatórios em formatos como PDF e CSV, tornando os resultados acessíveis e úteis para professores e alunos (MCKINNEY, 2017).

Além disso, ferramentas como Tkinter e PyQt possibilitam o desenvolvimento de interfaces gráficas que permitem a interação direta do usuário com o sistema, ampliando a usabilidade e facilitando a interpretação dos dados. Dessa forma, o Python contribui para a construção de uma camada de visualização intuitiva e eficiente, essencial em sistemas de apoio acadêmico.

## 2.4 Análise e Projeto de Sistemas

A Análise e Projeto de Sistemas é uma etapa essencial no ciclo de desenvolvimento de software, responsável por transformar as necessidades do usuário em modelos que orientam a implementação. A análise busca compreender o que o sistema deve realizar, enquanto o projeto define como essas funcionalidades serão implementadas em termos técnicos (SOMMERVILLE, 2019).

Segundo Pressman e Maxim (2021), essa fase é fundamental para garantir que o produto final atenda às expectativas do cliente, ao mesmo tempo em que seja tecnicamente viável, escalável e sustentável.

**UML – Casos de Uso, Classes e Sequência**

A Unified Modeling Language (UML) é um padrão amplamente utilizado para a modelagem de sistemas orientados a objetos. Criada por Booch, Rumbaugh e Jacobson na década de 1990, a UML possibilita a representação gráfica de diferentes perspectivas do sistema, facilitando a comunicação entre desenvolvedores, clientes e gestores (BOOCH; RUMBAUGH; JACOBSON, 2005).

- Diagramas de Casos de Uso: representam as interações entre os usuários (atores) e o sistema, descrevendo as funcionalidades que devem ser oferecidas.
- Diagramas de Classes: modelam a estrutura estática do sistema, detalhando atributos, métodos e relações entre entidades.
- Diagramas de Sequência: ilustram o fluxo de mensagens e interações ao longo do tempo, destacando como os objetos colaboram para atender a um caso de uso (FOWLER, 2004).

Esses diagramas são aplicados no projeto acadêmico como forma de documentar requisitos e organizar a arquitetura lógica do sistema.

**Especificação de Requisitos**

A Especificação de Requisitos de Software (SRS) é o documento que formaliza as funcionalidades e restrições de um sistema. Ela diferencia requisitos funcionais, que descrevem serviços e comportamentos do software, de requisitos não funcionais, que estabelecem restrições de desempenho, usabilidade, segurança e conformidade legal (SOMMERVILLE, 2019).

No caso do sistema acadêmico colaborativo, os requisitos incluem desde funcionalidades básicas, como cadastro de alunos e registro de aulas, até aspectos avançados, como uso de Inteligência Artificial e relatórios digitais sustentáveis. Essa

formalização contribui para reduzir ambiguidades e orientar a implementação e os testes.

### Arquitetura Cliente-Servidor

A arquitetura cliente-servidor é um modelo clássico de sistemas distribuídos, no qual o servidor centraliza dados e serviços, enquanto os clientes acessam e interagem com esses recursos por meio da rede. Essa abordagem garante organização, escalabilidade e segurança no acesso às informações (TANENBAUM; VAN STEEN, 2017).

De acordo com Sommerville (2019), essa arquitetura é apropriada para aplicações que exigem múltiplos acessos simultâneos, como no caso de um ambiente acadêmico, no qual professores e alunos utilizam o sistema de forma concorrente. Além disso, sua implementação favorece a integração de diferentes tecnologias, como módulos em C para manipulação de arquivos e rotinas em Python para relatórios e análises.

## 2.5 Redes de Computadores e Sistemas Distribuídos

As redes de computadores são essenciais para o funcionamento de sistemas distribuídos, permitindo a comunicação e o compartilhamento de recursos entre múltiplos usuários. Em um ambiente acadêmico, possibilitam que professores e alunos acessem dados e funcionalidades de forma simultânea, tornando a gestão de atividades mais eficiente.

Segundo Tanenbaum e Wetherall (2011), uma rede é composta por dispositivos interconectados que trocam informações por meio de protocolos bem definidos. Quando associadas a sistemas distribuídos, essas redes tornam-se a base para arquiteturas que garantem escalabilidade, confiabilidade e colaboração entre usuários.

### Fundamentos de Redes

As redes de computadores podem ser classificadas de acordo com sua abrangência, como LAN (Local Area Network), MAN (Metropolitan Area Network) e WAN (Wide Area Network). No caso do sistema acadêmico colaborativo, a escolha por uma LAN é

adequada, pois permite a comunicação em um ambiente restrito, como laboratórios e salas de aula (STALLINGS, 2017).

Protocolos de comunicação, como o TCP/IP, são fundamentais para garantir a entrega confiável dos pacotes de dados. O modelo OSI, por sua vez, organiza as funções da rede em sete camadas, padronizando a comunicação entre sistemas heterogêneos (KUROSE; ROSS, 2017).

**Topologia da Rede Local**

A topologia de rede define a forma como os dispositivos estão conectados. As mais comuns são: barramento, estrela, anel e malha. Em ambientes acadêmicos, a topologia em estrela é a mais utilizada, pois conecta todos os dispositivos a um ponto central (switch ou roteador), garantindo facilidade de configuração e manutenção (STALLINGS, 2017).

Para o sistema acadêmico, essa configuração assegura estabilidade e desempenho adequado para múltiplos acessos simultâneos em rede local.

**Configuração e Serviços de Redes**

A configuração de uma rede envolve a definição de endereçamento IP, máscara de sub-rede e gateways, além da escolha entre endereços estáticos ou atribuídos dinamicamente por meio do DHCP.

Serviços como DNS (Domain Name System) permitem a tradução de nomes de host em endereços IP, enquanto o DHCP (Dynamic Host Configuration Protocol) automatiza a distribuição de endereços de rede. Esses recursos tornam a administração da rede mais simples e eficiente (KUROSE; ROSS, 2017).

No projeto, a correta configuração desses serviços garante que os usuários possam acessar o sistema acadêmico colaborativo sem falhas de comunicação, assegurando usabilidade e desempenho em ambiente de laboratório.

## 2.6 Educação Ambiental

A Educação Ambiental busca promover a conscientização sobre a importância de práticas sustentáveis no cotidiano, estimulando mudanças de comportamento que minimizem impactos ambientais. No campo da tecnologia, a adoção de soluções digitais e a substituição de processos manuais por eletrônicos representam uma estratégia relevante para reduzir o consumo de recursos naturais.

Segundo Dias (2015), a Educação Ambiental deve ser entendida como um processo contínuo e participativo, que visa formar cidadãos críticos e responsáveis quanto à preservação do meio ambiente. Nesse sentido, o uso consciente da tecnologia pode ser um aliado no desenvolvimento de práticas mais sustentáveis.

### Conceitos Fundamentais

A Educação Ambiental, regulamentada no Brasil pela Lei nº 9.795/1999, integra os processos educacionais formais e não formais, orientando ações voltadas à preservação do meio ambiente e à promoção do desenvolvimento sustentável (BRASIL, 1999).

Do ponto de vista tecnológico, o desafio está em conciliar inovação com responsabilidade socioambiental, incentivando o uso de ferramentas digitais que reduzam desperdícios e promovam eficiência (JACOBI, 2003).

### Adoção de Relatórios Digitais x Papel

Um dos principais impactos ambientais das instituições de ensino está relacionado ao uso intensivo de papel em relatórios, provas e documentos administrativos. A substituição por relatórios digitais contribui para a redução significativa do consumo de papel, diminuindo custos e preservando recursos naturais (BARBIERI, 2011).

No contexto do sistema acadêmico colaborativo, os relatórios digitais substituem versões impressas, além de oferecerem maior acessibilidade, segurança e facilidade de armazenamento.

### Sustentabilidade no Uso de Sistemas

Além da eliminação do papel, a sustentabilidade no uso de sistemas computacionais envolve práticas como:

- otimização de código para reduzir consumo de energia;
- utilização de servidores compartilhados ou em nuvem para evitar desperdício de infraestrutura;
- incentivo ao acesso digital em vez de impressões desnecessárias.

Segundo Sachs (2008), a sustentabilidade tecnológica deve estar alinhada ao conceito de desenvolvimento sustentável, que integra crescimento econômico, inclusão social e proteção ambiental. Assim, o sistema proposto não apenas atende a demandas acadêmicas, mas também contribui para objetivos ambientais mais amplos.

## 2.7 Inteligência Artificial

A Inteligência Artificial (IA) é uma das áreas mais dinâmicas da ciência da computação, voltada para o desenvolvimento de sistemas capazes de realizar tarefas que normalmente exigem inteligência humana, como aprendizado, raciocínio e tomada de decisão. No contexto educacional, a IA pode oferecer recursos inovadores para análise de dados, personalização de ensino e otimização de processos acadêmicos.

### Conceitos e Histórico

O termo Inteligência Artificial foi formalizado em 1956, durante a Conferência de Dartmouth, quando pesquisadores como John McCarthy e Marvin Minsky propuseram o desenvolvimento de máquinas capazes de simular aspectos da inteligência humana (RUSSELL; NORVIG, 2021).

Inicialmente, a IA concentrou-se em sistemas baseados em regras e raciocínio simbólico. Com o avanço tecnológico, especialmente a partir da década de 2010, surgiram aplicações práticas de aprendizado de máquina e redes neurais profundas, possibilitando avanços em reconhecimento de fala, visão computacional e processamento de grandes volumes de dados (GOODFELLOW; BENGIO; COURVILLE, 2016).

### Técnicas Aplicadas ao Sistema

No projeto do sistema acadêmico colaborativo, a IA pode ser aplicada em diferentes frentes:

- Análise de dados acadêmicos: identificação de padrões de desempenho de alunos e turmas;
- Recomendações automáticas: sugestão de atividades de reforço com base no histórico do estudante;
- Otimização de buscas e consultas: algoritmos inteligentes para localizar rapidamente informações relevantes.

Segundo Alpaydin (2021), o uso de técnicas de aprendizado supervisionado e não supervisionado permite construir modelos que aprendem a partir de dados históricos, tornando os sistemas mais adaptativos e úteis.

**IA no Apoio ao Desempenho Acadêmico**

A aplicação da IA em ambientes educacionais contribui para monitoramento e apoio pedagógico. Professores podem utilizar relatórios gerados por algoritmos inteligentes para identificar alunos em risco de evasão ou com baixo desempenho, permitindo intervenções mais rápidas e eficazes.

Além disso, a IA favorece a personalização do ensino, adaptando conteúdos e exercícios ao perfil de cada estudante. De acordo com Holmes, Bialik e Fadel (2019), essa personalização aumenta o engajamento e melhora a aprendizagem, promovendo um processo educacional mais inclusivo e eficiente.

## 2.8 Pesquisa, Tecnologia e Inovação

A integração entre pesquisa, tecnologia e inovação é essencial para o avanço da sociedade do conhecimento, promovendo soluções capazes de transformar realidades sociais, educacionais e econômicas. Segundo a Organização para a Cooperação e Desenvolvimento Econômico (OCDE, 2015), a inovação não se limita a novos produtos ou serviços, mas envolve também melhorias em processos, modelos de gestão e formas de interação.

No contexto acadêmico, a inovação tecnológica fortalece a formação de profissionais capazes de lidar com os desafios da transformação digital, estimulando práticas de ensino-aprendizagem mais dinâmicas e inclusivas.

**Tecnologias Emergentes**

As tecnologias emergentes caracterizam-se por seu potencial disruptivo e por moldarem novos paradigmas. Entre elas, destacam-se a Inteligência Artificial (IA), computação em nuvem, big data, Internet das Coisas (IoT) e blockchain (SCHWAB, 2016).

Essas tecnologias oferecem novas possibilidades para o setor educacional, como plataformas adaptativas, análise de grandes volumes de dados acadêmicos e ambientes virtuais de aprendizagem mais interativos.

**Inovações Aplicáveis ao Sistema**

O sistema acadêmico colaborativo proposto incorpora inovações diretamente aplicáveis, como:

- relatórios digitais inteligentes, substituindo relatórios impressos;
- uso de IA para análise de desempenho e personalização de apoio ao estudante;
- armazenamento centralizado com acesso remoto em arquitetura cliente-servidor;
- interface intuitiva que facilita a colaboração entre professores, alunos e gestores.

Essas inovações alinham-se à visão de inovação aberta de Chesbrough (2006), na qual soluções acadêmicas e tecnológicas podem ser cocriadas e aprimoradas em ambientes colaborativos.

**Tendências Futuras**

As tendências futuras apontam para a intensificação do uso de tecnologias digitais na educação, como sistemas de tutoria inteligente, realidade aumentada e virtual, além de plataformas baseadas em aprendizado adaptativo. Christensen, Horn e Johnson

(2011) destacam que a inovação disruptiva no setor educacional pode gerar modelos de ensino mais personalizados e acessíveis, rompendo barreiras tradicionais.

Nesse cenário, o projeto do sistema acadêmico colaborativo antecipa essas tendências ao adotar recursos de IA, práticas sustentáveis e ferramentas de colaboração digital, contribuindo para um modelo educacional mais moderno, eficiente e sustentável.

# 3 DESENVOLVIMENTO DO SISTEMA

Este capítulo detalhar como o sistema foi concebido, modelado, desenvolvido e testado.

## 3.1 Propósito

O sistema acadêmico colaborativo foi desenvolvido com o propósito de centralizar e modernizar a gestão acadêmica, unificando o controle de turmas, alunos, aulas e avaliações em uma única plataforma. O projeto visa otimizar processos pedagógicos e administrativos, garantindo maior eficiência e acessibilidade das informações. Além disso, adota a metodologia ágil Scrum para um desenvolvimento colaborativo e contínuo, e aplica princípios de sustentabilidade ao substituir o uso de papel por relatórios digitais automatizados.

## 3.2 Escopo do Sistema

O sistema tem como escopo principal o gerenciamento centralizado das atividades acadêmicas de uma instituição de ensino, permitindo o cadastro de alunos, professores e turmas, bem como o registro de atividades e notas. O sistema também oferece relatórios digitais inteligentes, gerados com o apoio de Inteligência Artificial, para análise de desempenho e acompanhamento pedagógico. Sua estrutura foi projetada para operar em rede local (LAN), possibilitando o acesso simultâneo de múltiplos usuários em diferentes máquinas. A Tabela 1, mostra o resumo do escopo.

Entretanto, o sistema não contempla funcionalidades financeiras, como controle de mensalidades ou despesas, nem recursos de diário físico ou integração com sistemas externos, mantendo o foco exclusivo na gestão acadêmica e pedagógica.

Tabela 1 - Escopo Funcional do Sistema

| **Área** | **Função Principal** | **Atores Envolvidos** |
|---|---|---|
| Acadêmica | Gerenciar alunos, professores e turmas | Administrador, Professor |
| Avaliação | Registrar atividades e notas | Professor |
| Relatórios e IA | Gerar relatórios digitais e análises de desempenho | Admin, Professor, Aluno |
| Rede Local (LAN) | Permitir acesso simultâneo em múltiplas máquinas | Todos os usuários |

Fonte: Os Autores (2025).

## 3.3 Metodologia de Desenvolvimento

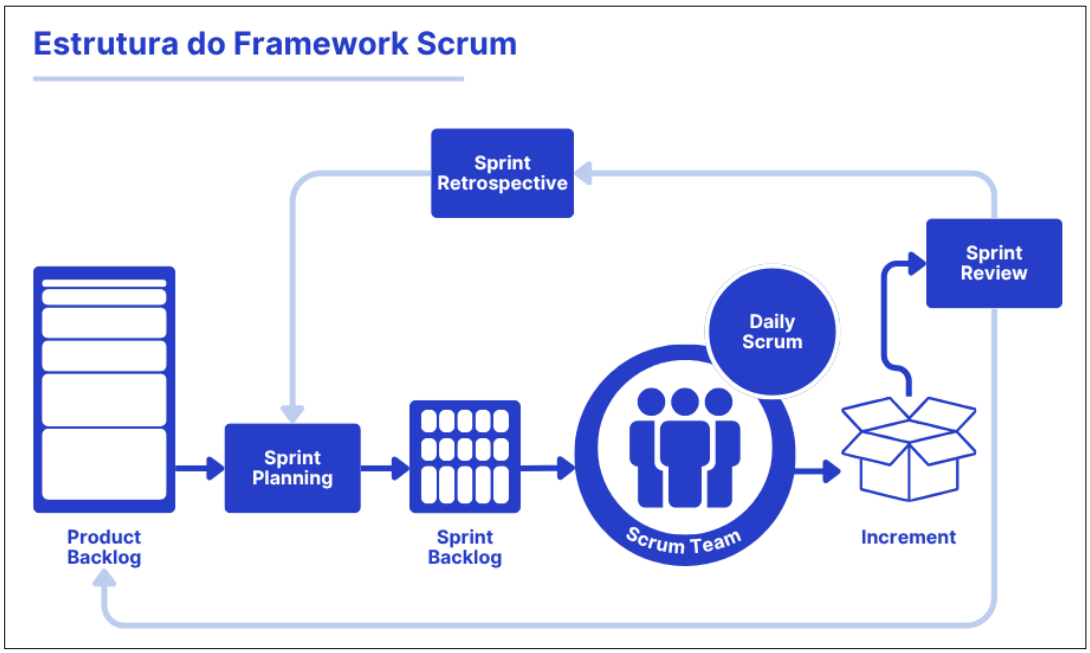
O desenvolvimento do sistema foi conduzido com base na metodologia ágil Scrum, que proporcionou uma organização eficiente das atividades, promovendo a colaboração contínua entre os membros da equipe e entregas incrementais de valor a cada etapa. Essa abordagem permitiu flexibilidade nas decisões, acompanhamento constante do progresso e melhor adaptação às necessidades do projeto, garantindo um processo de desenvolvimento dinâmico e bem estruturado.

### Abordagem de Desenvolvimento Ágil

A escolha da metodologia Scrum ocorreu devido à sua capacidade de organizar projetos de software de forma iterativa e incremental, facilitando o acompanhamento e a entrega gradual de resultados. Essa abordagem valoriza a colaboração do time, o feedback constante e o planejamento em ciclos curtos (sprints), o que contribuiu para o alinhamento entre todos os membros e para a rápida identificação e resolução de possíveis impedimentos. Na Figura 1, podemos observar a estrutura do framework Scrum, que nos ajudou na aplicação do método.

O método também garantiu maior transparência e controle sobre o andamento das tarefas, permitindo que as funcionalidades do sistema fossem desenvolvidas de forma contínua, revisadas e aperfeiçoadas em cada ciclo.

Figura 1 - Estrutura do Framework Scrum




Fonte: Adaptado de Schwaber e Sutherland (2020).

## Aplicação da Metodologia Scrum

O time adotou a estrutura clássica do Scrum, com papéis bem definidos que garantiram a organização e o comprometimento de cada integrante ao longo do projeto, como mostra a Tabela 2.

Tabela 2 - Papéis Scrum e Responsabilidades

| Papel | Responsabilidade | Membro Responsável |
|---|---|---|
| **Product Owner (PO)** | Definir prioridades, gerenciar o Product Backlog e alinhar requisitos com as necessidades do cliente. | Gabrielle Souza |
| **Scrum Master (SM)** | Garantir a correta aplicação da metodologia, facilitar a comunicação e remover impedimentos. | Gabriel Sousa |
| **Time de Desenv (Dev Team)** | Implementar o sistema, realizar testes e entregar incrementos funcionais ao final de cada Sprint. | Arthur Ferreira, Felipe Faria, Santiago Pacheco, Vinícius Carvalho |

Fonte: Os Autores (2025).

Durante o processo, foram utilizados os principais artefatos do Scrum, como o Product Backlog, o Sprint Backlog e o Incremento. Os eventos seguiram o ciclo completo do framework, a Tabela 3 representa os Sprints e Entregas.

Tabela 3 -Sprints e Entregas

| Sprint | Período | Foco Principal | Entregáveis (Incrementos) |
|---|---|---|---|
| **Sprint 1** | 01/09 – 15/09 | Definição do escopo, levantamento de requisitos e fundamentação teórica. | Documento SRS, Product Backlog inicial, cronograma de trabalho. |
| **Sprint 2** | 16/09 – 02/10 | Modelagem do sistema e desenvolvimento dos módulos em C. | Diagramas UML, DFDs, arquitetura do sistema e módulos de backend. |
| **Sprint 3** | 03/10 – 01/11 | Integração entre C e Python, aplicação da IA e testes finais. | Sistema funcional, relatórios digitais com IA e plano de testes. |

Fonte: Os Autores (2025).

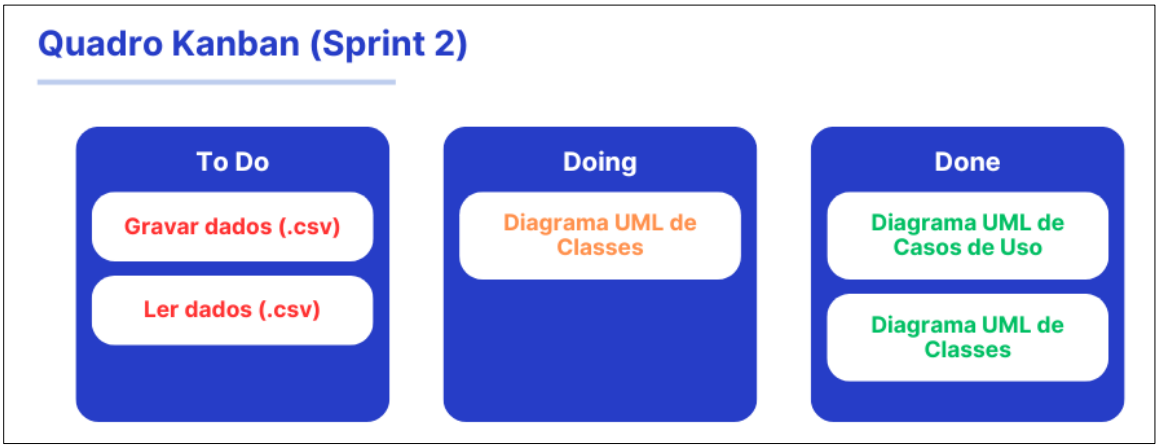
### Ferramentas e Organização das Sprints

Durante o desenvolvimento, foram utilizadas diversas ferramentas que apoiaram a organização e execução das tarefas:

- GitHub: controle de versão e armazenamento do código-fonte;
- Visual Studio Code: ambiente de desenvolvimento integrado (IDE);
- Python e C Compiler: linguagens utilizadas na implementação;
- Google Docs e Planilhas: documentação e registro de progresso;
- Kanban: acompanhamento visual das tarefas por sprint.

A equipe organizou as atividades de cada Sprint em um quadro Kanban, dividido em colunas To Do, Doing e Done, o que facilitou o controle das pendências e a visualização das tarefas concluídas, como exemplifica a Figura 2.

Figura 2 - Exemplo Quadro Kanban (Sprint 2)




Fonte: Os Autores (2025).

## 3.4 Requisitos do Sistema

Os requisitos do sistema foram definidos para orientar o desenvolvimento do projeto, assegurando que as funcionalidades implementadas atendessem às necessidades da gestão acadêmica de forma eficiente, organizada e alinhada às boas práticas de engenharia de software. Eles foram classificados em requisitos funcionais, que descrevem o que o sistema deve fazer, e requisitos não funcionais, que tratam de aspectos de desempenho, segurança, arquitetura e sustentabilidade.

### Requisitos Funcionais (RF)

Os requisitos funcionais estabelecem as principais operações que o sistema deve realizar, garantindo o gerenciamento completo das atividades acadêmicas e administrativas da instituição.

- RF01: O sistema deve permitir o cadastro de alunos, professores e turmas.
- RF02: O sistema deve registrar as atividades.
- RF03: O sistema deve registrar as notas dos alunos.
- RF04: O sistema deve gerar relatórios digitais de desempenho, frequência e notas com IA.
- RF05: O sistema deve operar em rede local (LAN), permitindo a comunicação entre diferentes computadores.
- RF06: O sistema deve permitir acesso simultâneo de múltiplos usuários conforme o tipo de perfil (administrador, professor, aluno).

**Requisitos Não Funcionais (RNF)**

Os requisitos não funcionais definem as condições técnicas e operacionais necessárias para o correto funcionamento do sistema, abordando aspectos como desempenho, compatibilidade, segurança e sustentabilidade.

- RNF01: O sistema deve ser baseado em arquitetura cliente-servidor.
- RNF02: O sistema deve ter parte do desenvolvimento em C estruturado (com estruturas de decisão, repetição, funções e manipulação de arquivos).
- RNF03: O sistema deve utilizar Python para consultas, relatórios e Front-end.
- RNF04: O sistema deve ser desenvolvido utilizando método ágil (Scrum).
- RNF05: O sistema deve responder às operações em tempo aceitável.
- RNF06: O sistema deve permitir o funcionamento em máquinas de laboratório acadêmico com configuração mínima: processador dual-core 2.0 GHz, 4 GB de RAM, 250 GB de armazenamento, sistema operacional Windows 10 ou superior.
- RNF07: O sistema deve possuir controle de acesso por tipo de usuário (professor, aluno e administrador).
- RNF08: O sistema deve considerar aspectos de segurança da informação e proteção de dados (LGPD).
- RNF09: O sistema deve priorizar soluções digitais que reduzam o consumo de papel.

## Diagrama de Casos de Uso

O Diagrama de Casos de Uso apresenta de forma visual as interações entre os diferentes usuários e o sistema, ilustrando as funcionalidades disponíveis para cada perfil. Ele demonstra como administradores, professores e alunos se relacionam com as principais operações do sistema, como o gerenciamento de cadastros, lançamento de notas, geração de relatórios e consultas, como ilustra a Figura 3.

Figura 3 - Diagrama de Casos de Uso do Sistema Acadêmico


Fonte: Os Autores (2025).

## 3.5 Arquitetura e Modelagem do Sistema

Esta seção apresenta a estrutura técnica e lógica adotada para o desenvolvimento do sistema acadêmico colaborativo, bem como sua modelagem conceitual e estrutural. A arquitetura proposta segue o modelo cliente-servidor, integrando módulos desenvolvidos em C e Python, além de contemplar um ambiente de rede local que possibilitaria o funcionamento simultâneo entre diferentes usuários.

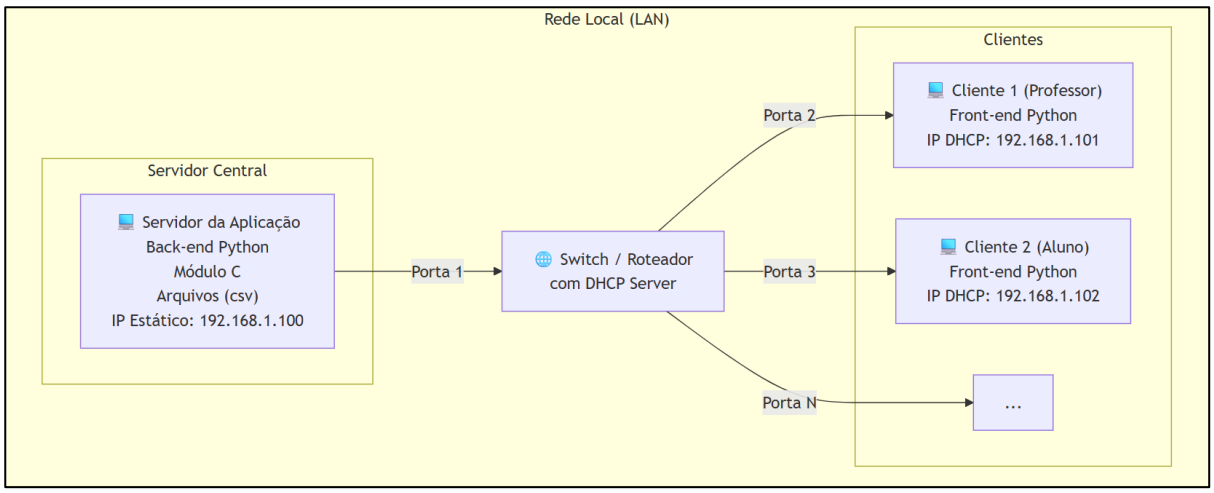
### Caracterização do Ambiente

O ambiente de execução proposto para o sistema baseia-se em uma rede local (LAN), composta por um servidor central e máquinas clientes interconectadas. Nessa configuração, o servidor seria responsável por armazenar os dados e gerenciar as requisições dos usuários, enquanto os clientes acessariam o sistema para realizar consultas, cadastros e visualizações.

O modelo prevê o uso de endereçamento IP fixo para comunicação entre os dispositivos e compartilhamento de arquivos via protocolo interno. Embora o sistema tenha sido projetado para esse tipo de ambiente, sua execução e testes foram realizados de forma simulada, representando como funcionaria em um laboratório acadêmico real.

A Figura 4 representa a Topologia Estrela, a arquitetura física dominante em LANs. Essa estrutura suporta o modelo Cliente-Servidor, onde todos os dispositivos se conectam a um ponto central para estruturar e controlar a troca de dados na rede.

Figura 4 - Topologia de Rede Local (LAN) Cliente-Servidor em Estrela



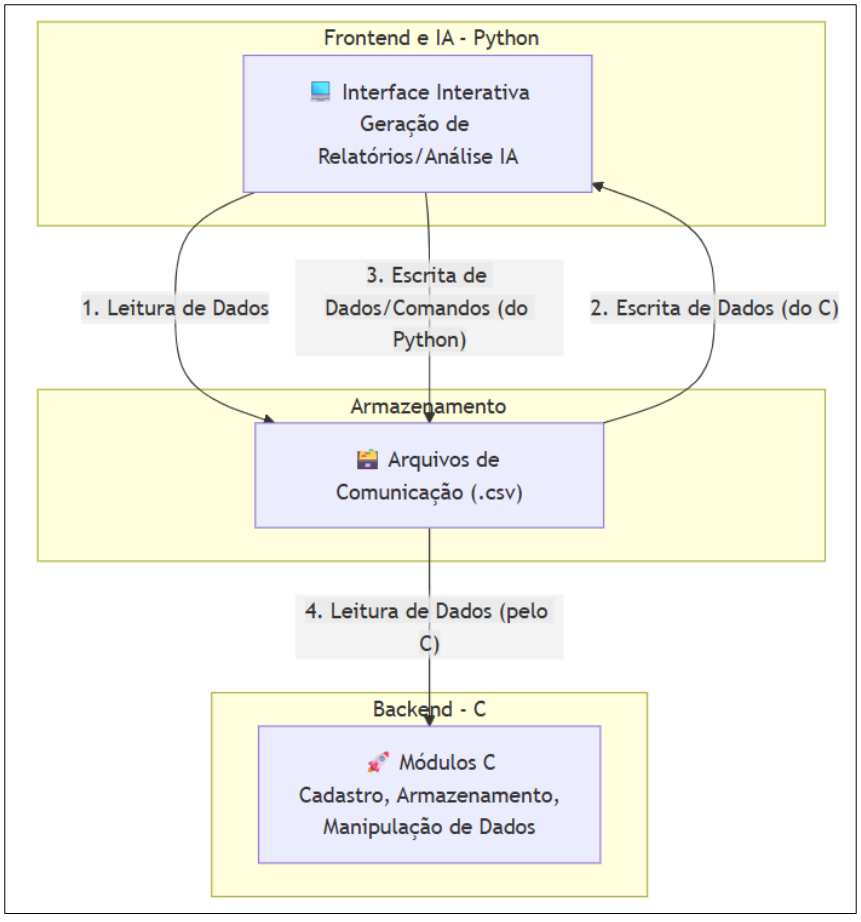

Fonte: Os Autores (2025).

## Arquitetura do Sistema

A arquitetura do sistema foi planejada de forma modular e integrada, unindo o desempenho da linguagem C com a flexibilidade e usabilidade do Python. Essa integração garante uma separação clara entre o processamento lógico e a camada de apresentação, favorecendo a manutenção e a escalabilidade do sistema. Como mostra a Figura 5.

- Backend (C): responsável pelos módulos de cadastro, armazenamento e manipulação de arquivos, utilizando estruturas, funções e controle de dados em formato .csv.
- Frontend e Inteligência Artificial (Python): encarregados da interface interativa, geração de relatórios digitais e análise inteligente de desempenho dos alunos.
- Comunicação: ocorre por meio da leitura e escrita de arquivos .csv, permitindo o intercâmbio de informações entre as linguagens e a execução em ambiente de rede local simulada.

Figura 5 - Arquitetura Geral do Sistema



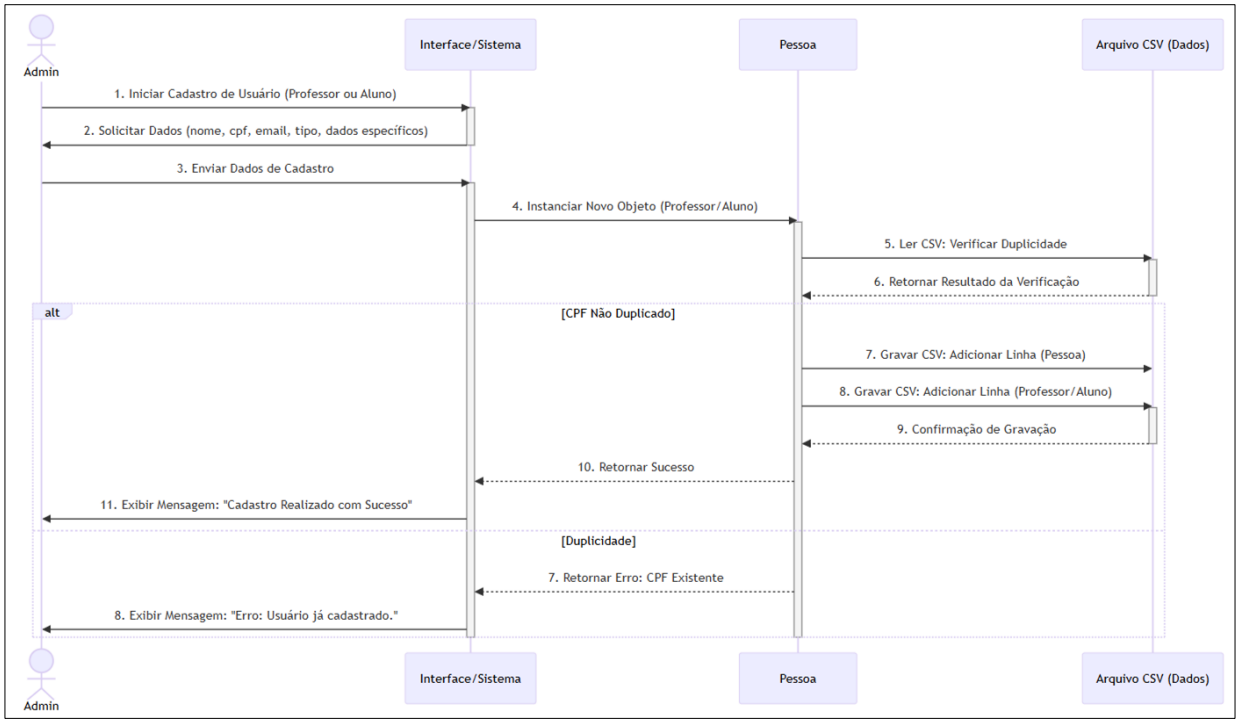

Fonte: Os Autores (2025).

## Modelagem do Sistema

A modelagem do sistema foi realizada por meio de diagramas UML, que representam a estrutura e o comportamento das principais entidades do projeto. Esses diagramas auxiliam na compreensão da lógica interna, da interação entre componentes e do fluxo de dados durante a execução do sistema.

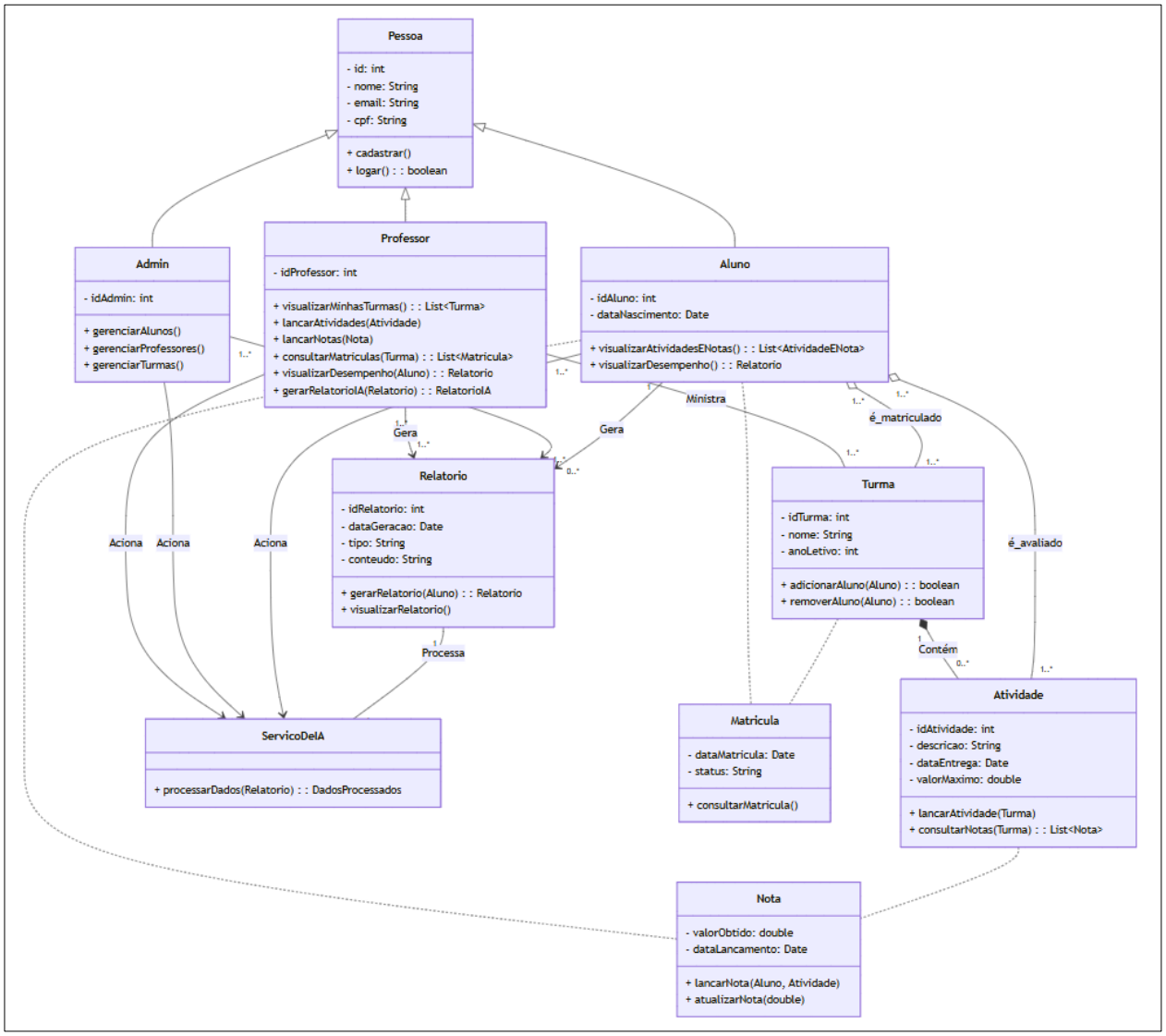
- Diagrama de Sequência: ilustra o fluxo de comunicação entre os objetos, destacando o processo de cadastro, na Figura 6, temos o Diagrama de Sequência de Cadastro de Usuário, todos os diagramas encontram-se disponíveis publicamente no repositório do projeto no GitHub, conforme indicado no Anexo A.
- Diagrama de Classes UML: demonstra as principais classes do sistema e seus relacionamentos, como Aluno, Professor, Turma, Atividade e Relatório, na Figura 7, podemos observar esse diagrama.

Figura 6 - Diagrama de Sequência: Cadastro de Usuário




Fonte: Os Autores (2025).

Figura 7 - Diagrama de Classes (Geral)



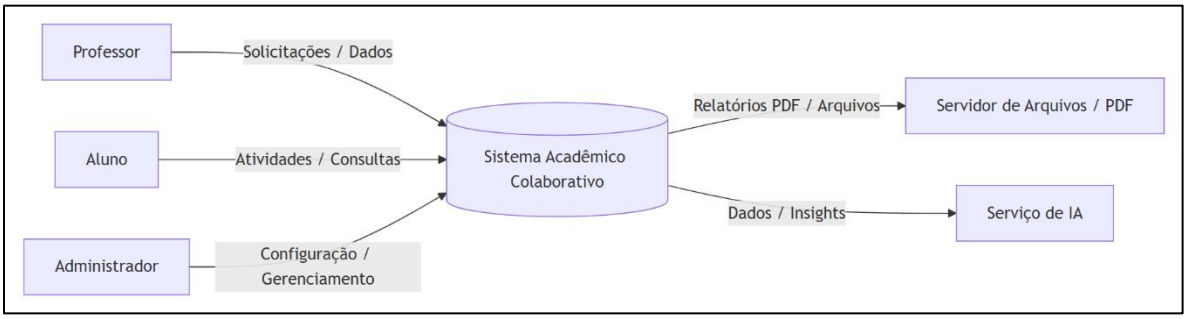

Fonte: Os Autores (2025).

## Diagramas de Fluxo de Dados (DFD)

Os Diagramas de Fluxo de Dados (DFDs) representam o caminho percorrido pelas informações dentro do sistema, descrevendo de forma clara como os dados são inseridos, processados e armazenados. Eles complementam a modelagem do sistema ao demonstrar a interação entre os usuários (atores externos), os processos internos e os arquivos de dados utilizados.

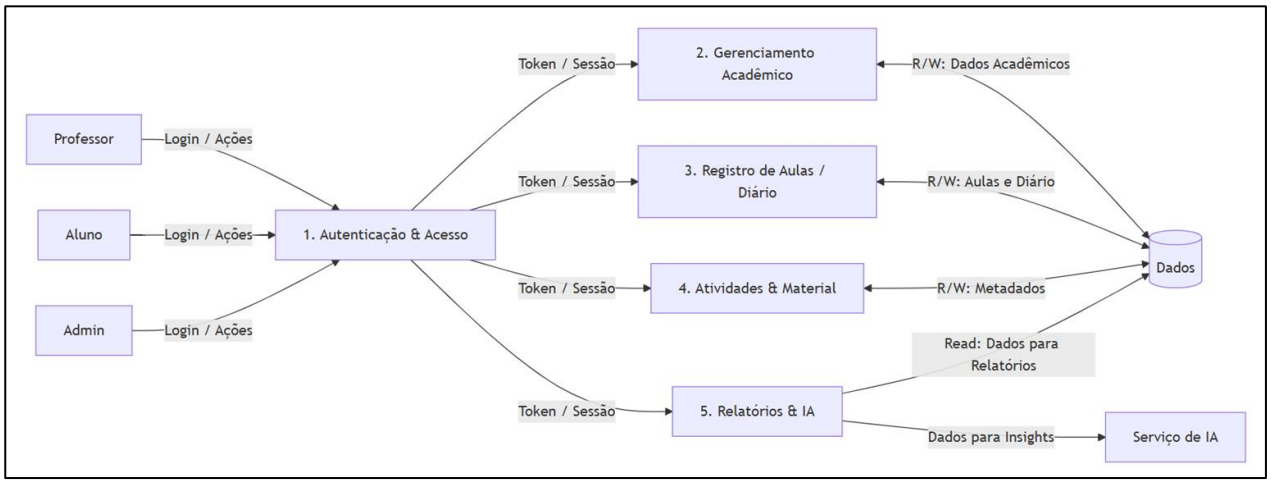
No sistema acadêmico colaborativo, os DFDs foram elaborados em três níveis, como ilustra as Figuras 8, 9 e 10:

Figura 8 - DFD Nível 0 (Contexto)



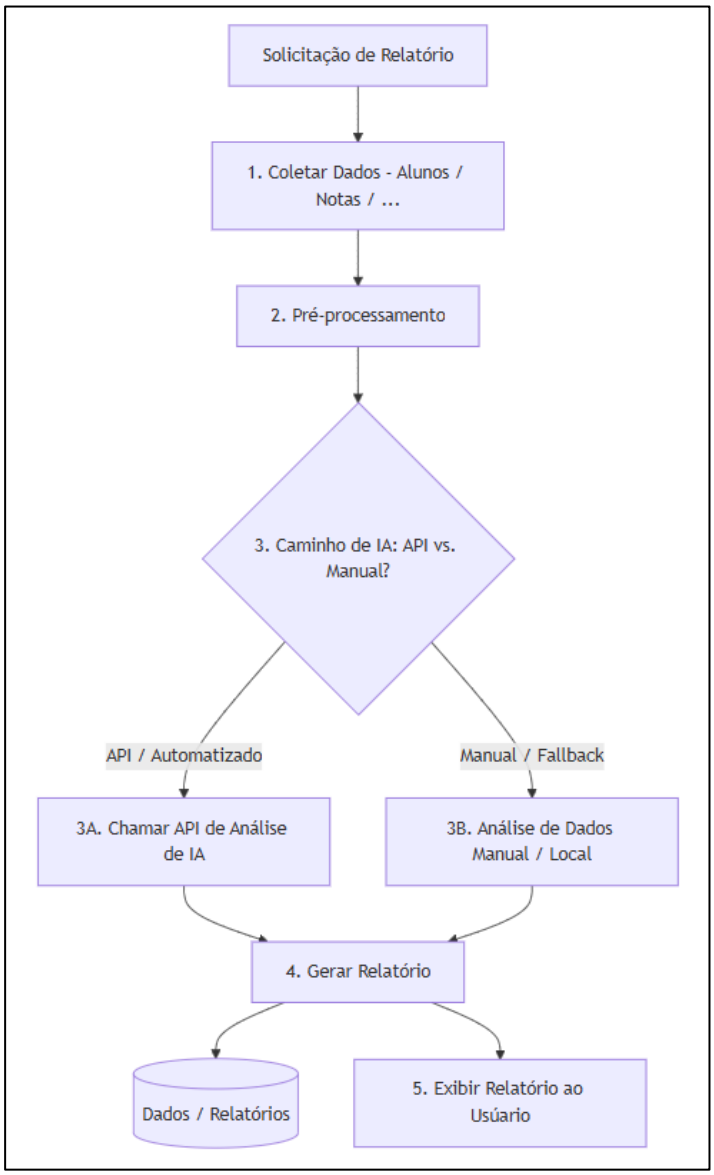

Fonte: Os Autores (2025).

Figura 9 - DFD Nível 1 (Processos Principais)




Fonte: Os Autores (2025).

Figura 10 - DFD Nível 2 (Processo de Relatórios e IA)




Fonte: Os Autores (2025).

## 3.6 Implementação

A implementação do sistema acadêmico colaborativo foi estruturada de forma modular, integrando as linguagens C e Python. Essa integração garante a separação entre as camadas de processamento lógico e apresentação gráfica, favorecendo a escalabilidade e a manutenção do sistema.

A estrutura de diretórios foi organizada da seguinte forma:

🗁 backend_c/
  ├── dados.c
  ├── estruturas.h

```
├── main.c
└── dados/ (pasta com os arquivos .csv)

🗁 frontend_python/
├── ai_module.py
├── app_gui.py
└── data_manager.py
```

Essa organização garante uma separação clara entre as responsabilidades de cada linguagem: enquanto o C atua no controle e persistência dos dados, o Python é responsável pela interface e geração de relatórios inteligentes.

**Módulos em C**

Os módulos em C formam o núcleo lógico e funcional do sistema, responsáveis pelo cadastro, armazenamento e manipulação dos dados de alunos, professores e turmas. Essa camada também realiza toda a persistência de informações, utilizando arquivos .csv como base de dados.

O arquivo estruturas.h define as estruturas e tipos personalizados utilizados em todo o sistema, enquanto o dados.c implementa as funções operacionais (cadastro, busca, listagem e atualização). Por fim, o main.c coordena a execução dos módulos e a interação com o usuário em modo terminal.

A Figura 11 apresenta um exemplo da função cadastrarAluno() contida no arquivo dados.c, responsável por armazenar novos registros de alunos no arquivo alunos.csv.

Figura 11 - Exemplo de função de cadastro em C (dados.c)

```c
void cadastrar_aluno() {
    Aluno *novo_aluno = (Aluno *)alocar_novo_registro_generico((void **)&alunos, &num_alunos, sizeof(Aluno));

    novo_aluno->id = obter_proximo_id("alunos.csv", sizeof(Aluno));

    printf("\n--- CADASTRO DE ALUNO ---\n");
    printf("Nome: "); scanf(" %99[^\n]", novo_aluno->nome); limpa_buffer();
    printf("Matricula: "); scanf(" %19s", novo_aluno->matricula); limpa_buffer();
    printf("CPF: "); scanf(" %14s", novo_aluno->cpf); limpa_buffer();
    printf("Login: "); scanf(" %49s", novo_aluno->login); limpa_buffer();
    printf("Senha: "); scanf(" %49s", novo_aluno->senha); limpa_buffer();

    // CRIPTOGRAFIA: Criptografa a senha antes de salvar no array
    criptografar_string(novo_aluno->senha);

    if (salvar_dados_csv("alunos.csv", alunos, num_alunos, sizeof(Aluno), ACESSO_ALUNO)) {
        printf("✅ Aluno %s cadastrado com sucesso (ID: %d).\n", novo_aluno->nome, novo_aluno->id);
    } else {
        printf("❌ Erro ao salvar dados do aluno.\n");
    }
}
```

Fonte: Os Autores (2025).

### Módulos em Python

Os módulos Python compõem a camada de apresentação e inteligência, sendo responsáveis pela interface gráfica, leitura dos dados e aplicação da IA.

- data_manager.py: gerencia a leitura dos arquivos .csv criados pelo C.
- app_gui.py: implementa a interface gráfica, permitindo que o usuário acesse as funções do sistema de forma intuitiva.
- ai_module.py: concentra os algoritmos de inteligência artificial, com duas abordagens distintas: uma IA manual (offline) e uma IA baseada em API (Gemini, Google).

A Figura 12 apresenta um trecho de código do módulo app_gui.py, que implementa a classe LoginFrame, responsável por renderizar a tela de login do sistema. Essa classe foi construída com o uso da biblioteca CustomTkinter (CTk), proporcionando uma interface moderna, responsiva e com design consistente.

Figura 12 - Exemplo de interface gráfica (LoginFrame em app_gui.py)

```
class LoginFrame(ctk.CTkFrame):
    def __init__(self, master, callback_sucesso):
        super().__init__(master, fg_color=LIGHT_GRAY_BG)
        self.callback_sucesso = callback_sucesso
        self.master_app = master

        self.grid_columnconfigure(0, weight=1)
        self.grid_rowconfigure(0, weight=1)

        # Card de Login
        self.login_card = ctk.CTkFrame(self, width=700, height=450, corner_radius=15, fg_color=CARD_BG)
        self.login_card.grid(row=0, column=0, sticky="")

        self.auth_panel = ctk.CTkFrame(self.login_card, fg_color="transparent")
        self.auth_panel.grid(row=0, column=0, sticky="nsew", padx=50, pady=50)
        self.auth_panel.grid_columnconfigure(0, weight=1)

        ctk.CTkLabel(self.auth_panel, text="SISTEMA DE ACESSO", font=ctk.CTkFont(size=20, weight="bold"),
                     text_color=PRIMARY_BLUE).grid(row=0, column=0, pady=(0, 20))

        self.acesso_var = ctk.StringVar(value="aluno")
        self.seg_button = ctk.CTkSegmentedButton(self.auth_panel, variable=self.acesso_var,
                                                 values=["aluno", "professor", "admin"], width=250, height=30)
        self.seg_button.grid(row=1, column=0, pady=10)

        self.login_entry = ctk.CTkEntry(self.auth_panel, placeholder_text="Login", width=250, height=40)
        self.login_entry.grid(row=2, column=0, pady=10)

        self.senha_entry = ctk.CTkEntry(self.auth_panel, placeholder_text="Senha", show="*", width=250, height=40)
        self.senha_entry.grid(row=3, column=0, pady=10)

        self.login_button = ctk.CTkButton(self.auth_panel, text="ENTRAR", command=self.tentar_login, width=250,
                                          height=40, fg_color=PRIMARY_BLUE)
        self.login_button.grid(row=4, column=0, pady=(20, 10))

        self.exit_button = ctk.CTkButton(self.auth_panel, text="SAIR", command=self.master_app.destroy, width=250,
                                         height=40, fg_color=ERROR_RED, hover_color="#B20C2D")
        self.exit_button.grid(row=5, column=0, pady=(10, 10))

        self.info_label = ctk.CTkLabel(self.auth_panel, text="", text_color=ERROR_RED)
        self.info_label.grid(row=6, column=0, pady=5)
```

Fonte: Os Autores (2025).

**Persistência dos Dados**

A persistência dos dados é realizada exclusivamente pelos módulos em C, que utilizam arquivos CSV como repositório de informações. Essa abordagem dispensa o uso de bancos de dados complexos, garantindo simplicidade, compatibilidade e portabilidade entre diferentes ambientes de execução.

Cada operação de cadastro, listagem ou atualização é refletida diretamente nos arquivos armazenados na pasta *dados/*, que concentram informações de alunos, professores e turmas. Os arquivos seguem um formato padronizado (ex.: ID,Nome,Nota), possibilitando fácil leitura e integração com o Python.

Essa estratégia assegura a integridade das informações e mantém o sistema funcional mesmo em ambientes offline, além de facilitar futuras expansões para bancos de dados relacionais, caso necessário.

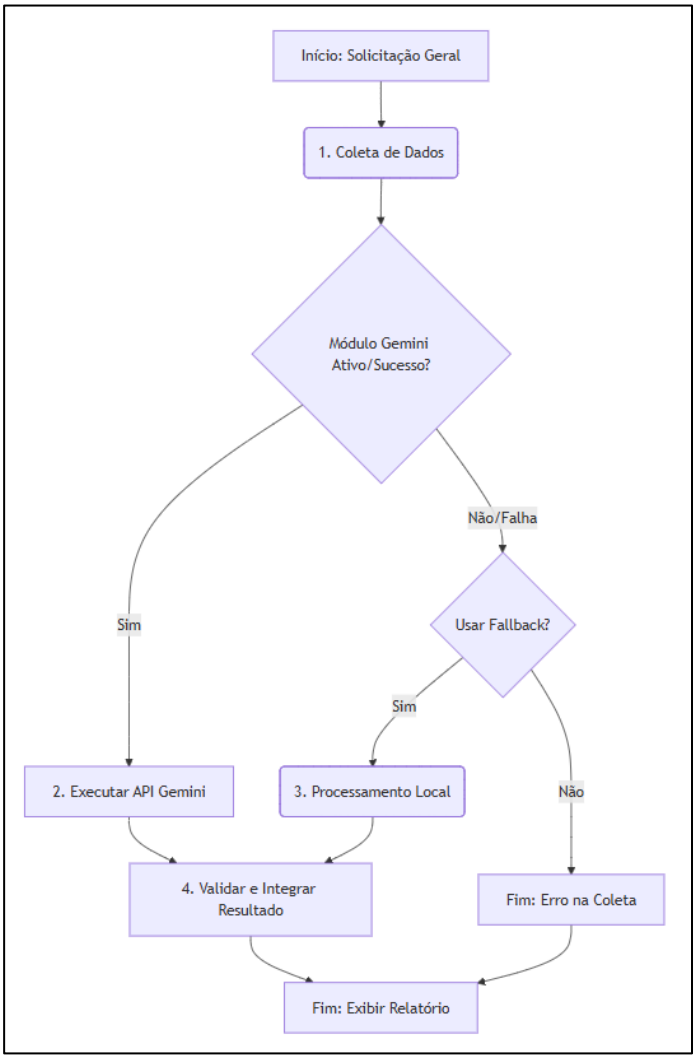
## 3.7 Aplicação de Inteligência Artificial

A aplicação de Inteligência Artificial (IA) no sistema acadêmico colaborativo tem como principal objetivo analisar o desempenho dos alunos e gerar relatórios automáticos de acompanhamento pedagógico. Essa funcionalidade amplia a capacidade de diagnóstico da instituição, oferecendo informações relevantes que auxiliam na tomada de decisões e na identificação de padrões de aprendizado.

A IA foi implementada em duas abordagens complementares: uma IA Manual (offline) e uma IA com integração à API Gemini (online).

O funcionamento segue um fluxo condicional: o sistema tenta primeiramente se conectar à API Gemini; caso não haja acesso à internet, ele ativa automaticamente a IA Manual para realizar as análises de forma local. Esse processo é ilustrado na Figura 13, que apresenta o fluxograma de decisão da IA.

Figura 13 - Fluxograma de funcionamento da Inteligência Artificial




Fonte: Os Autores (2025).

## IA Manual (Offline)

A IA Manual foi desenvolvida para funcionar sem dependência de rede, processando os dados diretamente a partir dos arquivos gerados pelos módulos em C (dados/alunos.csv, ...).

Essa versão realiza cálculos estatísticos simples, como a média geral da turma, o desempenho individual e a detecção de alunos com notas abaixo da média.

Com base nesses resultados, o sistema exibe mensagens interpretativas, classificando o desempenho como satisfatório, regular ou insatisfatório.

Esse modo garante a continuidade do funcionamento do sistema mesmo em ambientes desconectados, sendo especialmente útil em laboratórios locais e instituições com infraestrutura limitada.

**IA com API Gemini (Online)**

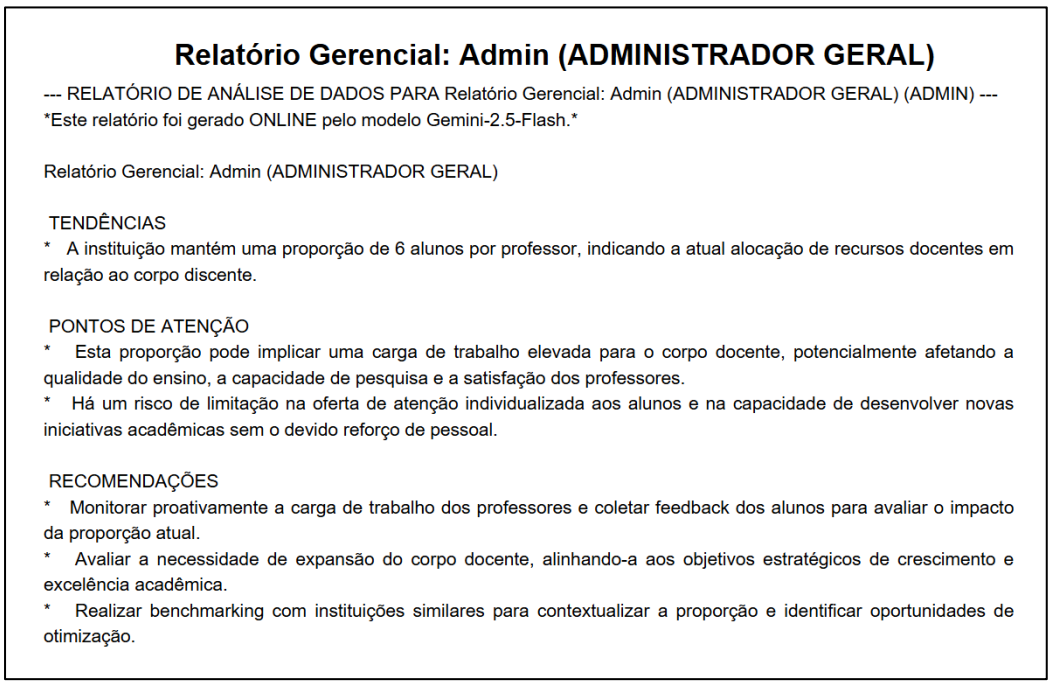
Quando há conexão com a internet, o sistema utiliza a API Gemini, modelo de IA generativa desenvolvido pela Google, para realizar uma análise mais detalhada e descritiva.

Através dessa integração, o sistema envia as notas e informações de desempenho dos alunos para a API, que retorna relatórios em linguagem natural, contendo observações sobre o rendimento médio da turma, sugestões de reforço e possíveis causas de baixo desempenho.

Essa abordagem oferece relatórios personalizados, com uma linguagem próxima à humana, tornando as análises mais acessíveis e compreensíveis tanto para professores quanto para administradores, e para os próprios alunos.

A Figura 14 apresenta um exemplo real de relatório gerado automaticamente pelo modelo Gemini.

Figura 14 - Exemplo de Relatório Gerado pela IA (API Gemini)



**Relatório Gerencial: Admin (ADMINISTRADOR GERAL)**

--- RELATÓRIO DE ANÁLISE DE DADOS PARA Relatório Gerencial: Admin (ADMINISTRADOR GERAL) (ADMIN) ---
*Este relatório foi gerado ONLINE pelo modelo Gemini-2.5-Flash.*

Relatório Gerencial: Admin (ADMINISTRADOR GERAL)

TENDÊNCIAS
* A instituição mantém uma proporção de 6 alunos por professor, indicando a atual alocação de recursos docentes em relação ao corpo discente.

PONTOS DE ATENÇÃO
* Esta proporção pode implicar uma carga de trabalho elevada para o corpo docente, potencialmente afetando a qualidade do ensino, a capacidade de pesquisa e a satisfação dos professores.
* Há um risco de limitação na oferta de atenção individualizada aos alunos e na capacidade de desenvolver novas iniciativas acadêmicas sem o devido reforço de pessoal.

RECOMENDAÇÕES
* Monitorar proativamente a carga de trabalho dos professores e coletar feedback dos alunos para avaliar o impacto da proporção atual.
* Avaliar a necessidade de expansão do corpo docente, alinhando-a aos objetivos estratégicos de crescimento e excelência acadêmica.
* Realizar benchmarking com instituições similares para contextualizar a proporção e identificar oportunidades de otimização.

Fonte: Os Autores (2025).

Essa abordagem com a API Gemini representa um avanço significativo em relação à análise tradicional, pois fornece interpretações dinâmicas e relatórios gerenciais em tempo real, transformando dados brutos em insights estratégicos.

Combinada à IA Manual, a solução garante flexibilidade operacional e inteligência adaptativa, funcionando de forma eficiente tanto em modo offline quanto online, consolidando o sistema como uma ferramenta moderna de apoio à gestão educacional.

### 3.8 Estratégias de Sustentabilidade

A adoção de práticas sustentáveis no desenvolvimento do sistema acadêmico colaborativo está diretamente alinhada aos princípios de Educação Ambiental e à promoção de um uso mais consciente dos recursos tecnológicos.

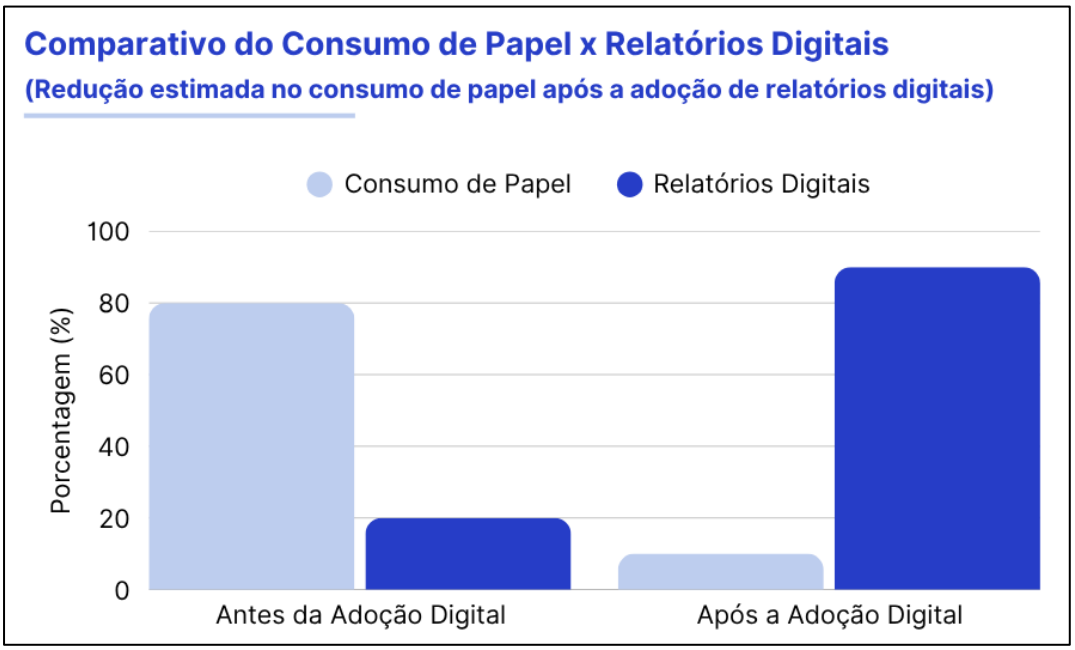
O projeto foi concebido com o propósito de reduzir o impacto ambiental gerado por processos administrativos tradicionais, priorizando soluções digitais e otimizadas.

A principal estratégia implementada foi a substituição do uso de papel por relatórios digitais automatizados, desenvolvidos e armazenados integralmente dentro do sistema. Essa decisão permitiu eliminar a necessidade de impressões físicas para cadastros, boletins e análises de desempenho, resultando em menor consumo de papel e redução de resíduos sólidos.

Além disso, toda a documentação do projeto, testes e relatórios de desenvolvimento foram produzidos e compartilhados em formato digital, promovendo a conscientização ambiental e reforçando o compromisso do grupo com a sustentabilidade tecnológica.

A Figura 15 ilustra o comparativo entre o consumo estimado de papel em processos convencionais e o impacto reduzido obtido com a adoção dos relatórios digitais, evidenciando a efetividade da solução implementada.

Figura 15 - Comparativo do Consumo de Papel x Relatórios Digitais




Fonte: Os Autores (2025).

Embora a principal iniciativa de sustentabilidade já tenha sido incorporada, outras ações futuras podem fortalecer ainda mais a responsabilidade ambiental do sistema.

Entre elas, destaca-se o uso eficiente de energia, ainda não implementado, mas considerado essencial para versões futuras. Essa melhoria incluiria recursos como modo escuro, desligamento automático em inatividade e otimização de servidores, contribuindo para um menor consumo energético e ampliando o compromisso ecológico do projeto.

Dessa forma, o sistema acadêmico colaborativo não apenas moderniza a gestão educacional, mas também atua como um exemplo prático de Tecnologia da Informação sustentável, integrando inovação digital e consciência ambiental em um mesmo propósito.

## 3.9 Testes e Homologação

A fase de testes e homologação teve como objetivo validar o correto funcionamento dos módulos do sistema e garantir o atendimento aos requisitos funcionais e não funcionais definidos durante o desenvolvimento.

Essa etapa foi essencial para verificar a integração entre as linguagens C e Python, o comportamento da Inteligência Artificial, a persistência dos dados em arquivos CSV e a interação do sistema em rede local simulada.

O plano de testes foi elaborado com base na verificação de requisitos, utilizando casos de teste unitários e integrados, que avaliaram tanto as funcionalidades individuais (cadastro, leitura e gravação de dados) quanto o funcionamento conjunto entre backend e frontend.

Foram também testados aspectos de usabilidade e acessibilidade, assegurando que a interface gráfica apresentasse comportamento estável e intuitivo durante a navegação do usuário.

### Casos de Teste

Os casos de teste foram estruturados de forma a abranger as principais funções do sistema, desde o cadastro de informações até a geração automática de relatórios pela IA e a execução em rede local. Cada caso foi documentado com o cenário de execução, resultado esperado e status final, conforme demonstrado na Tabela 4.

Tabela 4 - Casos de Teste e Resultados

| Caso | Ação Testada | Resultado Esperado | Status |
|---|---|---|---|
| **CT01** | Cadastro de aluno | Dados gravados corretamente no arquivo alunos.csv | OK |
| **CT02** | Geração de relatório pela IA (API Gemini) | Relatório gerado com sucesso e exibido na interface | OK |
| **CT03** | IA Manual (modo offline) | Análise local executada e relatório textual exibido | OK |
| **CT04** | Leitura e exibição de dados (Python) | Dados lidos corretamente dos arquivos CSV | OK |
| **CT05** | Interface de login e autenticação | Acesso concedido conforme credenciais válidas | OK |
| **CT06** | Listagem de alunos cadastrados | Dados apresentados corretamente na interface | OK |

Fonte: Os Autores (2025).

**Resultados Obtidos**

Os resultados obtidos durante a fase de testes demonstraram que todas as funcionalidades principais atenderam aos requisitos estabelecidos, apresentando comportamento estável e coerente com o esperado.

A integração entre os módulos em C (responsáveis pela persistência dos dados) e o frontend em Python (responsável pela interface e relatórios) foi validada com sucesso, sem ocorrência de erros críticos.

Durante a homologação, o sistema foi executado em um ambiente simulado de rede local (LAN), garantindo compatibilidade e desempenho satisfatório mesmo em múltiplas instâncias.

As funcionalidades de Inteligência Artificial, tanto na versão manual quanto na integração com a API Gemini, também apresentaram resultados corretos, com geração de relatórios coerentes e informativos.

Assim, o processo de testes confirmou que o sistema acadêmico colaborativo está totalmente funcional, estável e pronto para uso, cumprindo integralmente os objetivos definidos no projeto.

# 4 CONSIDERAÇÕES FINAIS

O desenvolvimento do Sistema Acadêmico Colaborativo proporcionou uma experiência prática e interdisciplinar, aplicando conceitos de Engenharia de Software, Programação Estruturada, Redes de Computadores e Inteligência Artificial. O projeto atingiu seu objetivo de centralizar e modernizar a gestão acadêmica, reunindo o cadastro de alunos, professores, turmas e notas em uma única plataforma, com relatórios digitais inteligentes que substituem processos manuais e reduzem o uso de papel.

A integração das linguagens C e Python mostrou-se eficiente: o C foi responsável pela manipulação e armazenamento dos dados, enquanto o Python cuidou da interface e das análises automatizadas, utilizando IA Manual (offline) e API Gemini (online). Essa combinação garantiu desempenho, escalabilidade e usabilidade em ambiente local.

A adoção da metodologia ágil Scrum permitiu uma organização eficiente das sprints, entregas contínuas e trabalho colaborativo, fortalecendo o aprendizado prático e o gerenciamento de tarefas. Os testes realizados comprovaram o funcionamento estável do sistema e o cumprimento dos requisitos propostos.

O projeto também se destacou pela sustentabilidade digital, ao substituir relatórios impressos por versões eletrônicas e reduzir o impacto ambiental. Todo o código-fonte e documentação técnica foram mantidos no repositório GitHub oficial, garantindo versionamento e transparência (conforme Anexo A).

Conclui-se que o sistema atendeu aos objetivos técnicos e pedagógicos, apresentando uma solução eficiente, acessível e ecologicamente responsável. Como aprimoramento futuro, recomenda-se o avanço da IA e a migração para banco de dados relacional, ampliando o potencial de aplicação em instituições de maior porte.

# 5 REFERÊNCIAS


ALPAYDIN, E. Introdução ao Aprendizado de Máquina. 4.ed. Cambridge: MIT Press, 2021.

BARBIERI, JC Gestão Ambiental Empresarial. 3.ed. São Paulo: Saraiva, 2011.

BECK, K. et al. Manifesto Ágil. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 15 de set de 2025.

BOOCH, G.; RUMBAUG, J.; JACOBSON, I. Guia do usuário da linguagem de modelagem unificada. 2. ed. Boston: Addison-Wesley, 2005.

BRASIL. Lei nº 9.795, de 27 de abril de 1999. Dispõe sobre a educação ambiental. Disponível em: http://www.planalto.gov.br. Acesso em: 20 de setembro de 2025.

CHESBROUGH, H. Inovação aberta: o novo imperativo para criar e lucrar com a tecnologia. Boston: Harvard Business School Press, 2006.

CHRISTENSEN, CM; CHIFRE, MB; JOHNSON, CW Inovação na Sala de Aula: como a inovação disruptiva mudará a forma de aprender. Porto Alegre: Bookman, 2011.

DALE, N.; WEEMS, C.; HEADINGTON, M. Programação e Resolução de Problemas com C. 8ª ed. Burlington: Jones & Bartlett Learning, 2017.

DEITEL, H.M.; DEITEL, PJ C: Como Programar. 8ª ed. Boston: Pearson, 2016.

DIAS, GF Educação Ambiental: princípios e práticas. 12. ed. São Paulo: Gaia, 2015.

FOWLER, M. UML destilada: um breve guia para a linguagem de modelagem de objetos padrão. 3.ed. Boston: Addison-Wesley, 2004.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Aprendizagem Profunda. Cambridge: MIT Press, 2016.

GUTTAG, J. Introdução à Computação e Programação Usando Python. 2. ed. Cambridge: MIT Press, 2016.

HOLMES, W.; BIALIK, M.; FADEL, C. Inteligência Artificial na Educação: Promessas e Implicações para o Ensino e a Aprendizagem. Boston: Centro para Redesenho Curricular, 2019.

JACOBI, PR Educação Ambiental, cidadania e sustentabilidade. Cadernos de Pesquisa, n. 118, pág. 189-205, 2003.

JOYCE, F. Iniciando em C: Do Iniciante ao Profissional. 5ª ed. Nova York: Apress, 2015.

KERNIGHAN, BW; RITCHIE, DM A Linguagem de Programação C. 2ª ed. Penhascos de Englewood: Prentice Hall, 1988.

KUROSE, JF; ROSS, KW Redes de Computadores e a Internet: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2017.

LUTZ, M. Aprendendo Python. 5. ed. Sebastopol: O'Reilly Media, 2013.

MCKINNEY, W. Python para análise de dados. 2. ed. Sebastopol: O'Reilly Media, 2017.

OCDE. Manual de Oslo: Diretrizes para Coleta e Interpretação de Dados de Inovação. 3.ed. Paris: Publicação OCDE, 2015.

PRATA, S.C Primer Plus. 6ª ed. Addison-Wesley, 2015.

PRESSMAN, RS; MAXIM, BR Engenharia de Software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.

RAMALHO, L. Python fluente. 1.ed. Sebastopol: O'Reilly Media, 2015.

RUSSELL, S.; NORVIG, P. Inteligência Artificial: Uma Abordagem Moderna. 4.ed. Boston: Pearson, 2021.

SACHS, I. Caminhos para o Desenvolvimento Sustentável. Rio de Janeiro: Garamond, 2008.

SCHWAB, K. A Quarta Revolução Industrial. São Paulo: Édipro, 2016.

SCHWABER, K.; SUTHERLAND, J. O Guia Scrum. 2020. Disponível em: https://scrumguides.org/. Acesso em: 15 de set de 2025.

SOMMERVILLE, I. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

STALLINGS, W. Dados e Comunicação por Computador. 10. ed. Boston: Pearson, 2017.

TANENBAUM, AS; VAN STEEN, M. Sistemas Distribuídos: Princípios e Paradigmas. 2. ed. Alto Rio Saddle: Pearson, 2017.

TANENBAUM, AS; WETHERALL, DJ Redes de Computadores. 5. ed. São Paulo: Pearson, 2011.

# 6 ANEXOS

ANEXO A – Repositório do Projeto

Todos os documentos complementares relacionados ao desenvolvimento deste trabalho — incluindo a Documentação Scrum, o SRS (Especificação de Requisitos de Software), os Diagramas UML e DFDs, os Planos de Testes, bem como o código-fonte completo do Sistema Acadêmico Colaborativo com Apoio de IA, foram armazenados e organizados em um repositório público no GitHub.

O repositório contém toda a estrutura do projeto, incluindo os diretórios backend_c e frontend_python, bem como os arquivos de documentação, relatórios e diagramas utilizados durante as etapas de planejamento, modelagem, implementação e testes.

Link do repositório: https://github.com/GabyValeria/pimII-2025-sistema-academico

Fonte: Os Autores (2025).

# FICHA DE CONTROLE DO PIM

**Ano:** 2025 **Período:** 2° **Coordenador:** Professor Roberto Cordeiro Waltz

**Tema:** Desenvolvimento de um Sistema Acadêmico Colaborativo com Apoio de IA

**Alunos:**

| RA | Nome | E-mail | Curso | Visto do Aluno |
|---|---|---|---|---|
| R661881 | Arthur de Lima Ferreira | arthur.ferreira29@aluno.unip.br | CST em ADS | |
| H719BH9 | Felipe Augusto Silva de Faria | felipe.faria22@aluno.unip.br | CST em ADS | |
| R869067 | Gabriel de Sousa Ferreira | gabriel.ferreira204@aluno.unip.br | CST em ADS | |
| R869DD5 | Gabrielle Valéria da Silva Souza | gabrielle.sou@aluno.unip.br | CST em ADS | |
| R8681C9 | Santiago dos Santos Pacheco | santiago.pacheco@aluno.unip.br | CST em ADS | |
| R870HA8 | Vinícius Machado de Carvalho | vini.machado@aluno.unip.br | CST em ADS | |

**Registros**

| Data do Encontro | Observações |
|---|---|
| 01/09/2025 | Criamos o repositório no GitHub, definimos o escopo inicial do projeto com principais funcionalidades e entregas, e alinhamos responsabilidades da equipe para melhor organização. |
| 06/09/2025 | Configuramos todos os recursos do projeto para desenvolvimento, modelagem, gestão, rede e documentação, garantindo a base para o PIM II. |
| 15/09/2025 | Elaboramos o referencial teórico do projeto, reunindo as principais bases conceituais e autores que fundamentam o desenvolvimento do trabalho. E o SRS (Especificação de Requisitos de Software). |
| 26/09/2025 | Elaboramos os DFDs com os fluxos de dados entre processos e fontes externas, além dos Diagramas UML, de Arquitetura e de Rede. |
| 02/10/2025 | Realizamos a validação dos DFDs, Diagramas UML, Arquitetura do Sistema e Diagrama de Rede junto ao professor Cordeiro, obtendo alinhamento e aprovação para prosseguimento das próximas etapas do projeto. |
| 04/11/2025 até 07/11/2023 | Mostramos o sistema, diagramas e documentação aos professores. Realizados ajustes imediatos e alinhamento conforme os *feedbacks* recebidos. |
| 10/11/2025 | Revisão geral do projeto. Finalização da documentação, ajustes no código e verificação da conformidade com as normas ABNT. |