UNIVERSIDADE PAULISTA

CURSOS

CST em Análise e Desenvolvimento de Sistemas

PIM

Projeto Integrado Multidisciplinar

2º Semestre - 2025

Nome: Felipe Herrera Felix RA: T118DG8

Sumário

1. Introdução

O presente Projeto Integrado Multidisciplinar (PIM) concentra-se no **Desenvolvimento de um Sistema Acadêmico Colaborativo com Apoio de IA**. A iniciativa propõe a criação de uma solução integrada para apoiar a gestão de turmas, alunos, aulas e atividades em uma instituição de ensino. Atualmente, muitos controles são realizados de maneira fragmentada, utilizando planilhas, e-mails e aplicativos de mensagens.

O sistema a ser desenvolvido buscará centralizar essas operações, incorporando funcionalidades de colaboração e explorando ativamente práticas de engenharia de *software* ágil. A implementação utilizará linguagens como Python e C, e será projetado para funcionar em uma rede local simples, aplicando o conceito de arquitetura cliente-servidor. Além disso, o projeto integrará o uso de recursos de Inteligência Artificial para enriquecer suas funcionalidades e incluirá diretrizes de sustentabilidade, como a eliminação do uso de papel.

2. Justificativa

A necessidade de otimizar os processos de gestão acadêmica e promover um ambiente de aprendizado mais eficiente e sustentável impulsiona o desenvolvimento deste projeto. Atualmente, a descentralização dos controles — com o uso de múltiplas ferramentas como planilhas e e-mails — gera ineficiência, dificulta o acompanhamento centralizado e, em muitos casos, resulta na dependência excessiva de materiais impressos.

O desenvolvimento de um sistema acadêmico colaborativo é crucial, pois endereça diretamente a demanda por uma solução que centralize o cadastro de turmas e alunos, o registro de aulas e o diário eletrônico, além de gerenciar o upload e a consulta de atividades. A incorporação de Inteligência Artificial (IA) é justificada pela possibilidade de oferecer funcionalidades inovadoras, como sugestões automatizadas ou apoio na geração de conteúdo técnico. Por fim, a adesão a medidas sustentáveis, como a eliminação do uso de

papel, alinha o projeto às práticas de educação ambiental e responsabilidade social.

3. Objetivo Geral e Específico

3.1. Objetivo Geral

Projetar e implementar um sistema acadêmico integrado que permita gerenciar turmas, alunos, aulas e atividades, com funcionalidades de colaboração, explorando práticas de engenharia de *software* ágil e o uso de recursos de Inteligência Artificial.

3.2. Objetivos Específicos

- Aplicar a metodologia de engenharia de software ágil para organizar sprints, backlog e o acompanhamento do projeto.
- Implementar algoritmos e estruturas de dados em Python aplicados a funcionalidades de busca, ordenação e relatórios.
- Desenvolver módulos críticos em linguagem C estruturada para compreensão da base de sistemas mais próximos do hardware.
- Criar modelos de análise e projeto de sistemas, incluindo a elaboração de diagramas UML (Caso de Uso, Classes, Sequência).
- Aplicar conceitos de redes de computadores e sistemas distribuídos, garantindo que o sistema funcione em rede local, explorando o modelo cliente-servidor para acesso multiusuário simultâneo.
- Utilizar recursos de Inteligência Artificial no desenvolvimento para enriquecer as funcionalidades do sistema.
- Realizar pesquisa de tecnologias emergentes para propor inovações aplicáveis ao sistema.
- Incluir recomendações de educação ambiental, como o uso de relatórios digitais em substituição ao papel.

4. Disciplinas Contempladas

O desenvolvimento do Sistema Acadêmico Colaborativo com Apoio de IA integra e aplica os conhecimentos fundamentais adquiridos nas diversas disciplinas do semestre, garantindo uma solução robusta, segura e funcional. Este capítulo estabelece a relação direta entre o projeto e os conteúdos de Redes de Computadores e Sistemas Distribuídos e Engenharia de Software Ágil, servindo como base teórica e prática para as decisões de implementação.

4.1 Redes de Computadores e Sistemas Distribuídos

O sistema acadêmico será implementado com base na arquitetura cliente-servidor e projetado para operar em uma Rede Local (LAN) simples. Essa abordagem é fundamental para permitir o acesso simultâneo de diferentes usuários e máquinas, conforme os requisitos do projeto.

O sistema será utilizado por dois grupos principais de usuários: **professores** e **alunos**.

- **Professores:** Responsáveis pelo gerenciamento de turmas, registro de aulas, diário eletrônico e *upload* de atividades.
- **Alunos:** Responsáveis por consultar turmas e notas, futuramente realizar o *download* e consulta de atividades.

A estimativa inicial de hardware considera a implantação em uma instituição de pequeno a médio porte, com aproximadamente 20 computadores (incluindo máquinas de gestão/secretaria e pontos de acesso fixos) e a rede dando suporte a um número flutuante de dispositivos pessoais de alunos e professores (notebooks, tablets e smartphones).

A topologia ideal para este ambiente é a do tipo Estrela, onde todos os dispositivos se conectam a um ponto central (como um switch gerenciado ou um servidor dedicado), facilitando o gerenciamento, a identificação de falhas e a implementação de segurança. Na Figura 1 é apresentada a organização e disposição da rede adotada.

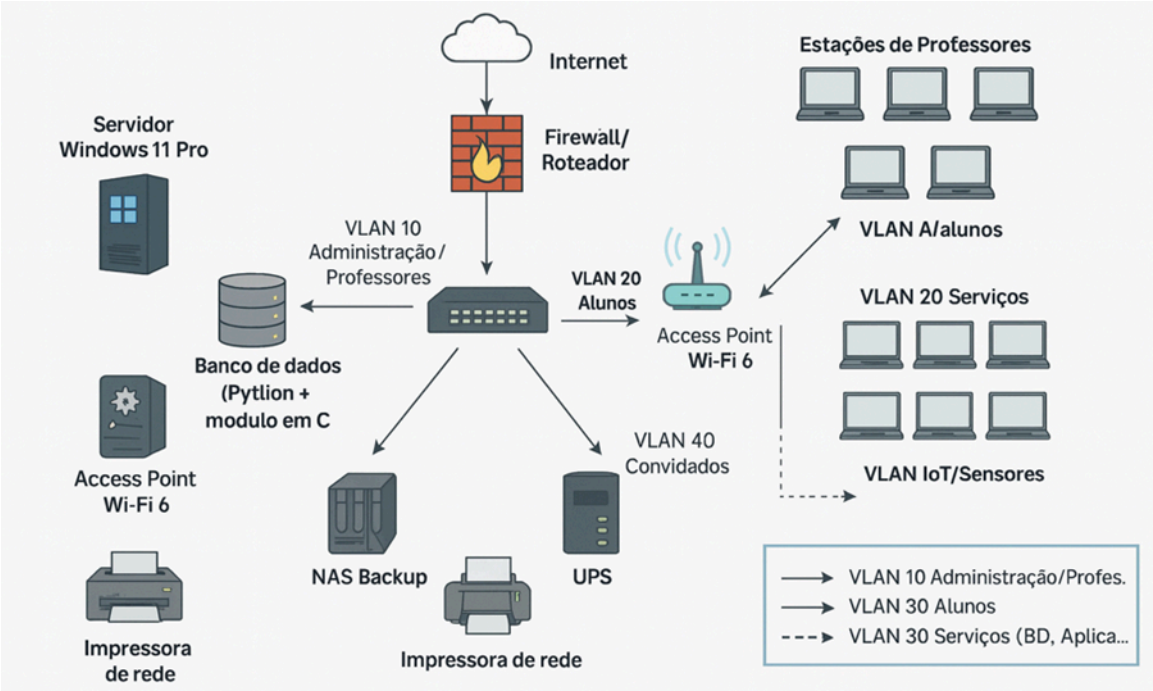


Figura 1 – Topologia Estrela adotada para integração do sistema

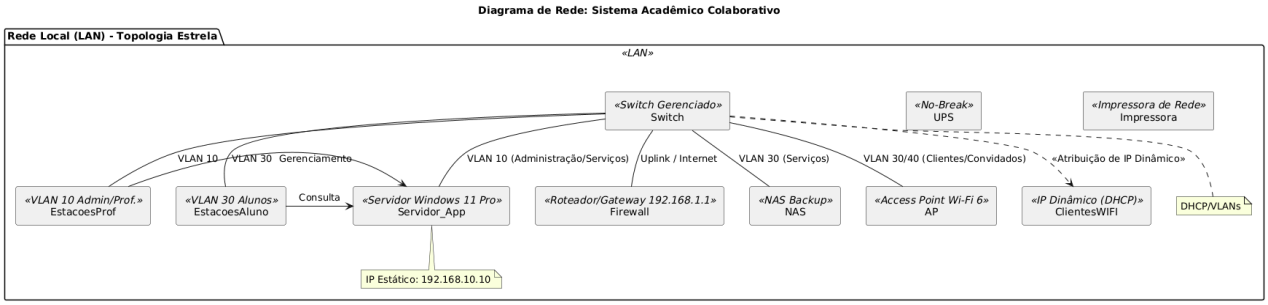


Figura 2 – Diagrama de rede: Sistema acadêmico colaborativo

Recomenda-se a adoção do sistema operacional **Windows 11 Pro** para os servidores e estações de gerenciamento, em virtude de seus recursos avançados de segurança, como o **BitLocker** para criptografia de unidade, políticas de atualização automatizadas e isolamento de credenciais. O sistema também oferece suporte nativo à **virtualização** e mecanismos de administração centralizada, como o **Active Directory** para controle de domínio, políticas de

grupo e gerenciamento remoto, o que favorece a padronização e o controle dos dispositivos na rede.

A proteção dos ambientes será garantida por meio da suíte **Norton 360 Advanced**, que oferece defesa em tempo real, **firewall** integrado, verificação de mensagens eletrônicas e proteção de navegação, formando uma camada robusta de segurança contra ameaças digitais. Além disso, propõe-se o uso de um **switch gerenciado**, que possibilita a criação de **VLANs** (redes locais virtuais) para segmentação do tráfego entre os diferentes perfis de usuários, como docentes, discentes, serviços internos e visitantes. Essa configuração permite maior controle de acesso, monitoramento e segurança da rede local.

Os princípios de **sistemas distribuídos** orientam o desenvolvimento do projeto, de modo que os módulos de software, como **frontend**, **backend** e **banco de dados**, operem de forma integrada na rede local, com suporte à **conectividade remota** segura para manutenção e suporte técnico.

A **Tabela 1** apresenta a relação detalhada dos equipamentos e serviços adotados, incluindo informações sobre o fabricante, quantidade, descrição técnica, preço total, preço unitário e fornecedor. Esses dados servem como base para o controle financeiro, a rastreabilidade dos recursos e o planejamento de capacidade da infraestrutura

Tabela 1. Registro de valores de equipamentos e serviços implementados

Fab.	Qtd.	Descrição	Preço Total	Preço Unitário	Fornecedor
Dell	3	Computador Intel Core i5, 16GB RAM, SSD 512GB	R$ 9.597,00	R$ 3.199,00	Dell Brasil
TP-Link	2	Switch Gigabit 24 portas TL-SG1024	R$ 1.198,00	R$ 599,00	Kabum
TP-Link	2	Access Point Wi-Fi 6 AX1800 Dual Band	R$ 1.198,00	R$ 599,00	Kabum
Microsoft	3	Licença Windows 11 Pro	R$ 2.697,00	R$ 899,00	Microsoft Store
Western Digital	1	NAS Backup 4TB WD My Cloud	R$ 1.499,00	R$ 1.499,00	Amazon
APC	1	No-Break (UPS) 1000VA	R$ 1.199,00	R$ 1.199,00	Terabyte
Norton	5	Licenças Norton 360 Advanced (Proteção Total)	R$ 1.095,00	R$ 219,00	Norton
AWS	1	Serviços em Nuvem (Backup, Banco de Dados e VPN)	R$ 458,00	R$ 458,00	Amazon Web Services
Total	—	—	**R$ 20.435,00**	—	—

4.2 Engenharia de Software Ágil e Análise e Projeto de Sistemas

A disciplina de Engenharia de Software Ágil constitui a base do processo de desenvolvimento, assegurando a organização das atividades, a entrega contínua de valor e a adaptação a mudanças de requisitos. Para este projeto, adota-se a metodologia Scrum, utilizada no planejamento, execução e acompanhamento das etapas de desenvolvimento.

O levantamento de requisitos foi realizado a partir da análise do cenário atual da instituição de ensino, caracterizado por controles descentralizados e não integrados, como o uso de planilhas e e-mails. Identificaram-se como requisitos essenciais o cadastro de turmas e alunos, o registro de aulas, o diário eletrônico, o upload e a consulta de atividades, além do funcionamento distribuído em rede. A disciplina de Análise e Projeto de Sistemas fundamenta a elaboração dos diagramas UML, como Casos de Uso, Classes e Sequência, permitindo a visualização da estrutura e do comportamento do sistema antes do início da codificação.

O ciclo de desenvolvimento será estruturado em Sprints de curta duração, com periodicidade de duas semanas. Cada ciclo contemplará duas fases principais:

1. **Planejamento da Sprint (Sprint Planning):** etapa em que a equipe define as funcionalidades prioritárias a serem desenvolvidas e elabora o **Sprint Backlog**, contendo as tarefas e seus respectivos critérios de aceitação. O acompanhamento do progresso será realizado por meio do **Quadro Kanban** apresentado na Figura 2.
2. **Revisão da Sprint (Sprint Review):** fase em que a equipe apresenta o incremento de software desenvolvido às partes interessadas, como o professor orientador ou o coordenador do projeto, a fim de obter feedback e validar os resultados parciais.

Figura 2 – Quadro Kankan do Sistema Educativo

Além dessas etapas, serão realizadas reuniões de acompanhamento e retrospectivas, visando à melhoria contínua do processo e à manutenção da qualidade das entregas de acordo com a Definição de Pronto (*Definition of Done)* e os critérios de aceitação previamente estabelecidos.

4.3 Algoritmos e Estruturas de Dados Python e Programação Estruturada em C

A funcionalidade essencial do Sistema Acadêmico Colaborativo depende diretamente da aplicação eficiente dos conceitos de programação.

Aplicação em Python A linguagem Python será utilizada para a construção da lógica principal do sistema e das funcionalidades mais complexas voltadas ao usuário, como a geração de relatórios e a manipulação de grandes volumes de dados de turmas e notas. A disciplina de Algoritmos e Estruturas de Dados Python é aplicada na implementação de:

- **Busca e Ordenação:** Algoritmos serão empregados para realizar buscas rápidas por alunos ou turmas e para ordenar listas de notas ou frequências.
- **Estruturas de Dados:** Embora o projeto pressupunha a manipulação de dados em arquivos JSON, o conhecimento aprofundado de estruturas como listas e dicionários em Python será crucial para otimizar o acesso e o armazenamento temporário de dados em memória.

Aplicação em C Estruturado A disciplina de **Programação Estruturada em C** é integrada ao projeto com o objetivo de desenvolver um módulo crítico do sistema. Esta escolha visa proporcionar uma compreensão da base de sistemas mais próximos do *hardware*. O módulo em C será responsável pela criação de um arquivo de dados contendo informações de alunos e notas, que será posteriormente lido e exibido pela interface em Python, simulando a comunicação entre diferentes componentes do sistema.

4.4 Inteligência Artificial e Pesquisa, Tecnologia e Inovação

A inovação proposta para o sistema acadêmico reside na integração de recursos de **Inteligência Artificial (IA)**, um requisito central do projeto.

- **Inteligência Artificial:** A aplicação da IA não se limita à programação de algoritmos complexos de aprendizado de máquina, mas abrangerá o uso de ferramentas para aprimorar a eficiência do trabalho. As evidências de aplicação se darão, por exemplo, na utilização de *prompts* de IA para auxiliar na geração de textos técnicos para o manual do sistema, na criação de mensagens automáticas de *feedback*. O objetivo é explorar

como as ferramentas de IA podem apoiar a rotina de desenvolvimento e a experiência do usuário.

- **Pesquisa, Tecnologia e Inovação:** Esta disciplina orienta a busca por tecnologias emergentes e a proposição de diferenciais competitivos ao sistema. O uso da biblioteca Streamlit para a interface, a manipulação de dados em JSON e o protocolo de comunicação em rede local são exemplos de escolhas tecnológicas embasadas em pesquisa. A inovação e a criatividade no projeto são critérios de avaliação e serão evidenciadas pelo diferencial proposto pelo grupo e pela forma como a arquitetura cliente-servidor é demonstrada no ambiente de rede.

4.5 Educação Ambiental

A disciplina de **Educação Ambiental** é integrada ao projeto por meio da proposição de medidas sustentáveis no sistema. O cenário do trabalho estabelece que um dos objetivos é a eliminação do uso de papel pelos professores. Desta forma, o sistema contribuirá para:

- **Sustentabilidade Digital:** Implementando funcionalidades que substituam a necessidade de materiais impressos, como o diário eletrônico e a consulta de atividades exclusivamente em formato digital.
- **Métricas de Sustentabilidade:** O sistema pode incluir relatórios digitais que substituem o papel e, futuramente, métricas para acompanhamento da redução do consumo de recursos

4.6 Quadro Kanban

https://trello.com/b/DOQHJvkS/pimsegundosemestre

4.7 Ferramentas de Gestão, Modelagem e Prototipação

O desenvolvimento do Sistema Acadêmico Colaborativo foi estruturado sob os princípios da Engenharia de Software Ágil, exigindo um conjunto robusto de

ferramentas para dar suporte desde o planejamento até a entrega do código-fonte. A adoção dessas tecnologias foi crucial para manter a transparência, a rastreabilidade e a adaptabilidade do projeto, em linha com a metodologia Scrum. Para garantir a gestão, a modelagem e a prototipação eficazes do projeto, foram empregadas diversas ferramentas que suportam a metodologia ágil adotada e a criação dos artefatos técnicos do sistema.

Ferramentas Essenciais Utilizadas

Trello (Quadro Kanban): Utilizado como ferramenta central de gestão visual, o Trello permitiu a criação de um Quadro Kanban dinâmico para as *sprints* do projeto. Nele, o *backlog* e as tarefas foram organizados nas colunas "A Fazer", "Em Progresso", "Em Revisão/Testes" e "Concluído", facilitando o acompanhamento visual do fluxo de trabalho e o alinhamento de toda a equipe.

Astah Professional (UML): A fase de Análise e Projeto de Sistemas foi fundamentada no Astah, onde foram criados os diagramas UML essenciais (Caso de Uso, Classes e Sequência). Essa modelagem serviu como um mapa arquitetônico claro, permitindo a compreensão da estrutura do sistema antes da codificação, garantindo que as classes Python e a comunicação entre o módulo C e o servidor estivessem bem definidas.

Figma (Prototipação): O Figma foi empregado para a prototipação de alta fidelidade das interfaces do sistema (desktop, web e mobile). Essa etapa foi vital para refinar a usabilidade e a arquitetura da informação em um ambiente colaborativo e de baixo custo, garantindo que a implementação final, fosse em Streamlit, atendesse às expectativas do usuário final.

GitHub (Controle de Versão): O GitHub foi a plataforma de controle de versão escolhida, sendo indispensável para gerenciar o código-fonte em Python e C. Ele assegurou a colaboração segura e assíncrona entre os membros da equipe, registrando o histórico completo de modificações e facilitando a integração do trabalho de programação estruturada e orientada a objetos.

Ferramenta	Uso e Aplicação no Projeto	Disciplina Contemplada
Trello (Quadro Kanban)	Utilizado para a gestão visual e transparente das tarefas, suportando a metodologia Scrum. Permitiu o acompanhamento do backlog e a organização das atividades nas colunas "A Fazer", "Em Progresso", "Em Revisão/Testes" e "Concluído" ao longo das sprints.	Engenharia de Software Ágil
Astah Professional (ou ferramenta similar)	Ferramenta utilizada para a modelagem dos diagramas UML exigidos (Casos de Uso, Classes e Sequência). Serviu como base visual para a compreensão da estrutura e do comportamento do sistema antes da codificação.	Análise e Projeto de Sistemas
Figma	Empregado para a criação do design visual e prototipação das telas de desktop, web e mobile, definindo a usabilidade e a arquitetura de informação antes da implementação com Tkinter ou Streamlit.	Análise e Projeto de Sistemas
GitHub	Plataforma de controle de versão essencial para o gerenciamento do código-fonte em Python e C. Garantiu a colaboração segura entre os membros do grupo e a rastreabilidade de todas as modificações realizadas no sistema.	Engenharia de Software Ágil

4.8 Evidências da Aplicação Estratégica da Inteligência Artificial (IA)

A inovação do Sistema Acadêmico reside, em parte, na integração de recursos de Inteligência Artificial, conforme previsto no escopo. Em vez de focar na construção de modelos de Machine Learning complexos (escopo de projetos futuros), utilizamos ferramentas de **IA Generativa** para aprimorar a documentação e a experiência de uso do sistema. Essa abordagem otimizou o tempo de desenvolvimento e agregou valor aos artefatos de entrega, transformando a IA em uma ferramenta de otimização de processos.

A seguir, são detalhadas as evidências de aplicação da IA no projeto:

- **Geração de Textos Técnicos para o Manual do Sistema:**
    - **Propósito:** Utilização de ferramentas de IA (via *prompts* específicas) para auxiliar na elaboração de trechos padronizados e tecnicamente corretos para o manual de uso e o guia técnico do sistema. Isso garantiu clareza e profissionalismo na documentação.
- **Criação de Mensagens Automáticas de Feedback:**
    - **Propósito:** Desenvolvimento de mensagens concisas e informativas a serem exibidas para o usuário em cenários específicos, como confirmação de cadastro de turmas, erros de comunicação na rede ou alertas de aprovação/reprovação. Essas mensagens melhoram a comunicação com o usuário.
    - **Exemplo de Prompt Utilizado:** "Crie uma mensagem automática de sucesso para o aluno após a consulta de suas notas, incentivando o bom desempenho."

Essa integração demonstra o compromisso do projeto com a Pesquisa, Tecnologia e Inovação, alinhando-se aos requisitos disciplinares.

5. Contextualização do Caso

Uma instituição de ensino necessita de um sistema colaborativo para apoiar professores e alunos no gerenciamento de turmas, aulas e atividades. Atualmente, controles são realizados de forma descentralizada (planilhas, e-mails, mensagens em aplicativos).

O sistema deve permitir cadastro de turmas e alunos, registro de aulas e diário eletrônico, upload e consulta de atividades, e módulos distribuídos em uma rede. Um dos objetivos é a eliminação do uso de papel pelos professores como medidas sustentáveis.

O sistema deverá ser projetado para funcionar em uma rede local simples (LAN), aplicando o conceito de cliente-servidor. Os alunos deverão demonstrar como diferentes usuários podem acessar e interagir com o sistema em máquinas distintas. Simulações ou testes em laboratório de redes podem ser utilizados para validação.

6. Manual de Uso do Sistema

6.1 Introdução

O **Sistema Acadêmico Colaborativo 2.0** é uma aplicação web desenvolvida em Python (framework Streamlit) para gestão de dados acadêmicos, turmas e notas. Sua arquitetura cliente-servidor e a integração com um módulo em Linguagem C demonstram o domínio das disciplinas do projeto.

O sistema possui três perfis de acesso: **Aluno**, **Professor** e **Administrador** (Admin), cada um com funcionalidades específicas.

6.2. Requisitos de Instalação e Execução

O sistema foi desenvolvido em Python e utiliza o framework Streamlit para a interface gráfica.

Requisito	Detalhes
Linguagem	Python 3.x
Frameworks	streamlit, cryptography
Módulo C	Binário modulo_c.exe (necessário para a integração)
Execução	Via comando streamlit run [nome_do_arquivo].py

Passos para Execução:

1. Garanta que o ambiente Python e as bibliotecas necessárias (pip install streamlit cryptography) estejam instalados.
2. Coloque o arquivo Python principal (main.py, por exemplo) e o binário C (modulo_c.exe) no mesmo diretório.

Execute o comando no terminal:
Bash
streamlit run [nome_do_arquivo].py

3. O sistema abrirá automaticamente no navegador web padrão.

6.3. Acesso e Cadastro

Ao iniciar o sistema, o usuário é direcionado à tela de **Login / Cadastro Novo**.

6.4. Cadastro de Novo Usuário

O cadastro é realizado na barra lateral (Sidebar). É necessário fornecer:

- **Usuário novo:** O login de acesso.
- **Senha nova:** A senha será armazenada de forma segura (criptografia SHA256).
- **Nome Completo:** O nome será armazenado de forma criptografada (Fernet) para segurança.
- **Tipo:** Selecione o perfil de acesso (**aluno**, **professor** ou **admin**).

> ⚠️ **Nota de Segurança:** O nome e o Registro Acadêmico (RA) do aluno são criptografados usando o algoritmo **Fernet**, com uma chave gerada e armazenada localmente no arquivo key.bin, garantindo a confidencialidade dos dados sensíveis.

6.5. Login

O usuário deve inserir suas credenciais nos campos **Seu Usuário** e **Sua Senha** e clicar em **Entrar**. Após o login, o sistema redireciona o usuário para a área específica de seu perfil.

6.6. Funcionalidades do Perfil: Professor e Administrador

Professores e Administradores acessam o **Área de [PERFIL]** e utilizam o menu lateral (sidebar) para navegar entre as funções:

Menu	Funcionalidade	Perfil
Dashboard	Visão geral do sistema (total de alunos, turmas).	Professor / Admin
Cadastrar Turmas	Criação de novas turmas.	Professor / Admin

Gestão de Turmas	Matrícula de alunos em turmas existentes.	Professor / Admin
Diário Eletrônico (Notas C)	Lançamento e edição de notas, com execução do módulo C.	Professor / Admin
Gerenciar Usuários	Exclusão permanente de usuários.	Admin (Apenas)

6.7. Diário Eletrônico (Lançar Notas)

Esta é a principal função para o lançamento das notas e demonstração da **Integração Python-C**.

1. Selecione a **Matéria para editar** na caixa de seleção.
2. A **Tabela de Notas** será carregada, permitindo a edição direta dos campos **NP1**, **NP2** e **PIM** (o PIM é a nota global para todas as matérias).
3. Após a edição, clique em **Salvar Tudo e Chamar o C**.

💡 **Integração Python-C:** Ao clicar no botão de salvar, o sistema realiza os seguintes passos:

1. Salva as notas no arquivo JSON (dados.json).
2. Cria um arquivo binário (dados_notas.dat) com o RA, NP1, NP2 e PIM dos alunos.
3. Cria um arquivo de texto (ras_para_c.txt) com a lista de RAs.
4. Executa o binário **C** (./modulo_c.exe), que lê o arquivo ras_para_c.txt e sincroniza o binário, garantindo que apenas

os RAs válidos do sistema Python permaneçam no arquivo binário.

6.8. Funcionalidades do Perfil: Aluno

O aluno acessa a Área do Aluno para consultar seus dados acadêmicos e notas.

1. Dados: Exibe o ID da Turma do aluno.
2. Minhas Notas:
    - O aluno pode selecionar a matéria para ver o detalhamento da nota.
    - Exibe as notas NP1, NP2 e a nota PIM (global).
3. Cálculo da Média: O sistema calcula automaticamente a Média Final da Matéria e o Status (Aprovado ≥ 7 ou Reprovado), seguindo a regra de ponderação da instituição (NP1 x 4 + PIM x 2 + NP2 x 4).
4. Média Geral: O sistema exibe a Média Aritmética de Tudo, calculada a partir da média final de todas as matérias.

7. Conclusão

O presente Projeto Integrado Multidisciplinar (PIM) alcançou com sucesso o objetivo geral de projetar e implementar um sistema acadêmico integrado que permite gerenciar turmas, alunos, aulas e atividades, incorporando práticas de engenharia de software ágil e o uso de recursos de Inteligência Artificial. A solução desenvolvida responde diretamente à problemática da descentralização dos controles em instituições de ensino, que utilizam múltiplas ferramentas como planilhas e e-mails, gerando ineficiência no gerenciamento.

A concretização do sistema foi possível pela integração bem-sucedida de todas as disciplinas contempladas. A Engenharia de Software Ágil e a Análise e Projeto de Sistemas nortearam a organização do projeto em sprints e a modelagem estrutural através de diagramas UML. O funcionamento multiusuário

e a arquitetura cliente-servidor foram estabelecidos pelos conceitos de Redes de Computadores e Sistemas Distribuídos, garantindo a conectividade em rede local.

No âmbito da programação, a eficiência e a lógica do backend foram garantidas pela aplicação de Algoritmos e Estruturas de Dados Python. Conforme requisito, o desenvolvimento de um módulo crítico em C Estruturado ofereceu a compreensão da base de sistemas mais próximos do hardware e simulou a comunicação entre componentes de diferentes linguagens. A diferenciação e inovação do sistema foram evidenciadas pela incorporação da Inteligência Artificial na geração de prompts de apoio e textos técnicos.

Finalmente, a inclusão de diretrizes da Educação Ambiental, através da eliminação do uso de papel, alinha o projeto às práticas de responsabilidade social e sustentabilidade, transformando o sistema em um ativo eficiente e ecologicamente consciente. O resultado é um sistema funcional que atende aos requisitos essenciais e demonstra a capacidade do grupo em aplicar o conhecimento multidisciplinar na resolução de um problema real.

8. Trabalhos futuros

O desenvolvimento do Sistema Acadêmico Colaborativo com Apoio de IA, embora funcional e robusto em sua arquitetura de rede local (LAN), representa a fundação para expansões e melhorias. Sugerem-se os seguintes caminhos para Trabalhos Futuros, visando otimizar a escalabilidade, segurança e o conjunto de funcionalidades do sistema:

Expansão para Arquitetura em Nuvem (Cloud Computing): Migração do ambiente para uma arquitetura baseada em nuvem, utilizando serviços como AWS ou Azure, para garantir maior disponibilidade, escalabilidade e acesso remoto seguro, superando as limitações de uma rede local.

Desenvolvimento de Aplicações Nativas: Implementação de aplicativos móveis nativos (Android/iOS), conforme sugerido nas observações gerais, para otimizar a experiência do usuário, especialmente para consulta de atividades e registro de frequência fora do ambiente desktop.

Aprimoramento da Inteligência Artificial e Predição: Evolução dos prompts de apoio para a integração de modelos de Machine Learning mais avançados. Isso permitiria a criação de funcionalidades preditivas, como a identificação precoce de alunos em risco de evasão ou a sugestão personalizada de trilhas de aprendizado baseadas na análise de padrões de desempenho.

Reforço da Segurança e Gestão de Identidades: Implementação de autenticação de dois fatores e integração com serviços de diretório, como o Active Directory ou LDAP, para reforçar a segurança e a gestão centralizada das credenciais de acesso de professores e alunos.

Metrificação de Sustentabilidade no Dashboard: Criação de um painel de controle (dashboard) dedicado para a mensuração e exibição das métricas de sustentabilidade, quantificando a redução do consumo de papel e energia por meio dos relatórios digitais, como aprofundamento das diretrizes de Educação Ambiental.

9. Referências

DE VIT, Antônio Rodrigo Delepiane; BELO, Sidnei Renato Silveira. ***Introdução à programação e estruturas de dados fundamentais com Python para programadores C.*** Belo Horizonte, MG: Synapse Editora, 2023. Disponível em: https://www.editorasynapse.org/wp-content/uploads/2023/04/Livro_Python_v1.pdf. Acesso em: 24 out. 2025.

MENEZES, Priscylla Karoline de. ***Educação ambiental.*** Recife: Ed. UFPE, 2021. Disponível em:

https://repositorio.ufpe.br/bitstream/123456789/49421/1/Educa%C3%A7%C3%A3o%20ambiental.pdf. Acesso em: 24 out. 2025.

PRIKLADNICKI, Rafael; WILLI, Renato; MILANI, Fabiano. ***Métodos Ágeis para Desenvolvimento de Software*.** Porto Alegre: Bookman, 2014.

SILVA, Todman Reis da; MEDEIROS, Marcus Vinicius Batella; MEDEIROS, Glaucia Rodrigues Nascimento. Gestão de Riscos no Framework SCRUM **Utilizando Análise SWOT.** *Revista de Humanidades, Tecnologia e Cultura*, v. 10, n. 2, dez. 2021. Disponível em: https://bkpsitecpsnew.blob.core.windows.net/uploadsitecps/sites/51/2024/08/22-METODOLOGIAS-AGEIS-NO-DESENVOLVIMENTO-DE-SOFTWARES-UMA-REVISAO-BIBLIOGRAFICA.pdf. Acesso em: 24 out. 2025.

TANNENBAUM, Andrew S.; WETHERALL, David J. ***Redes de Computadores*.** 5. ed. Rio de Janeiro: Prentice Hall, 2011.

WAZLAWICK, Raul Sidnei. ***Análise e projeto de sistemas de informação orientados a objetos*.** 2. ed. Rio de Janeiro: Elsevier, 2011.

10. Anexos

10.1. Acesso ao GitHub

https://github.com/Github-FelipeFelix/PIM_UNIP_2SEMESTRE

10.2. Código sistema em Python

```
import json

import os

import secrets

import struct
```

```python
import subprocess

import streamlit as st

from cryptography.fernet import Fernet, InvalidToken

from collections import defaultdict # Deixa a importação
inútil


# --- Paths e configs ---

base_path = os.path.dirname(os.path.abspath(__file__))

dados_path = os.path.join(base_path, 'dados.json')

chave_path = os.path.join(base_path, 'key.bin')


# RAs fixos

ras_fixos = {0: 1234567, 1: 9876543, 2: 1122334}


# Materias do curso

materias = [

    "PIM",

    "ENG_SOFT_AGIL",

    "ALGOR_ESTRUT_PYTHON",

    "PROG_ESTRUT_C",

    "ANALISE_PROJ_SIST",

    "PESQ_TEC_INOV",

    "EDU_AMBIENTAL",

    "REDES_DISTRIB",
```

```
    "INTELIGENCIA_ART",
    "ESTUDOS_DISCIPLINARES",
]


# --- Funções auxiliares de Segurança/Dados ---
def pega_chave():
    if not os.path.exists(chave_path):
        k = Fernet.generate_key()
        with open(chave_path, 'wb') as f:
            f.write(k)
        return k
    with open(chave_path, 'rb') as f:
        return f.read()


def cripto(texto):
    f = Fernet(pega_chave())
    return f.encrypt(texto.encode()).decode()


def decripto(texto):
    f = Fernet(pega_chave())
    try:
        # Tive que colocar o padding aqui, senão dá pau
        texto_pad = texto + '=' * (-len(texto) % 4)
        return f.decrypt(texto_pad.encode()).decode()
```

```
    except InvalidToken:
        return None
    except Exception:
        return None


def hash_senha(s):
    import hashlib
    return hashlib.sha256(s.encode()).hexdigest()


def notas_vazias():
    # Cria a estrutura de notas vazia
    n = {}
    for m in materias:
        if m == "PIM":
            n[m] = 0.0
        else:
            n[m] = {"NP1": 0.0, "NP2": 0.0}
    return n


def carrega_dados():
    if os.path.exists(dados_path):
        with open(dados_path, 'r') as f:
            return json.load(f)
    return {'usuarios': {}, 'alunos': [], 'turmas': []}
```

```
def salva_dados(d):
    with open(dados_path, 'w') as f:
        json.dump(d, f, indent=4)

# --- Funções de Lógica e Regras de Negócio ---
def login_ok(u, s):
    d = carrega_dados()
    if u in d['usuarios'] and d['usuarios'][u]['hash'] ==
hash_senha(s):
        return d['usuarios'][u]['tipo']
    return None

def cad_user(u, s, nome, tipo='aluno'):
    d = carrega_dados()
    if u in d['usuarios']:
        return False
    d['usuarios'][u] = {'hash': hash_senha(s), 'tipo': tipo}

    if tipo == 'aluno':
        # Conta quantos alunos tem para dar o RA
        cont = len([a for a in d['alunos'] if
a['usuario_login'] not in ['admin','professor']])
        ra = ras_fixos.get(cont,
secrets.randbelow(9000000)+1000000)
```

```
        d['alunos'].append({
            'usuario_login': u,
            'nome_criptografado': cripto(nome),
            'ra_criptografado': cripto(str(ra)),
            'turma_id': 'Nenhuma',
            'notas_c': notas_vazias()
        })
    salva_dados(d)
    return True


def cadastra_turma(nome):
    d = carrega_dados()
    if any(t['nome'] == nome for t in d['turmas']):
        return False
    d['turmas'].append({'id': secrets.token_hex(3), 'nome':
nome})
    salva_dados(d)
    return True


def matricula_aluno(u_login, t_id):
    d = carrega_dados()
    for a in d['alunos']:
        if a['usuario_login'] == u_login:
            a['turma_id'] = t_id
```

```
            salva_dados(d)
            return True
    return False

def apagar_user(u):
    d = carrega_dados()
    if u not in d['usuarios']:
        return False, "Usuário não existe"
    del d['usuarios'][u]
    # Filtra os alunos (usa list comprehension porque é "moderno")
    d['alunos'] = [a for a in d['alunos'] if a['usuario_login'] != u]
    salva_dados(d)
    return True, f"{u} apagado."

# --- Integração C ---
def envia_para_c():
    d = carrega_dados()
    lista_ras = []
    arq_bin = 'dados_notas.dat'
    # Estrutura: int (RA), float (NP1), float (NP2), float (PIM)
    struct_bin = struct.Struct('i f f f')
    try:
```

```
        with open(arq_bin, 'wb') as f:
            for a in d['alunos']:
                # Pega RA, se der erro, pula (tem que ser
número)
                ra = decripto(a['ra_criptografado'])
                if not ra or not ra.isdigit():
                    continue
                lista_ras.append(ra)

                # Matéria que vai pro C
                np1 = float(a['notas_c'].get('PROG_ESTRUT_C',
{}).get('NP1') or 0.0)
                np2 = float(a['notas_c'].get('PROG_ESTRUT_C',
{}).get('NP2') or 0.0)
                pim = float(a['notas_c'].get('PIM') or 0.0)

                f.write(struct_bin.pack(int(ra), np1, np2,
pim))
    except Exception as e:
        return False, f"Erro criando binário: {e}"

    if not lista_ras:
        return False, "Nenhum RA para enviar."
```

```
    with open(os.path.join(base_path,'ras_para_c.txt'),'w') as
f:
        f.write('\n'.join(lista_ras))

    try:
        subprocess.run(['./modulo_c.exe'], check=True)
        return True, "Módulo C executado."
    except FileNotFoundError:
        return False, "Executável C não encontrado"
    except Exception as e:
        return False, f"Erro ao rodar C: {e}"

# --- Funções de Interface Auxiliares (UI) ---

def ui_gestao_turmas():
    st.subheader("Matricular Alunos")
    d = carrega_dados()
    turmas = d['turmas']

    # Faz um loop pra pegar quem não tá matriculado
    alunos_sem_turma = [decripto(a['nome_criptografado']) for
a in d['alunos'] if a['turma_id'] == 'Nenhuma']

    if alunos_sem_turma and turmas:
```

```
        aluno_selecionado = st.selectbox("Aluno para
Matricular:", alunos_sem_turma)

        turma_selecionada = st.selectbox("Turma para Alocar:",
[t['nome'] for t in turmas])



        # Pega o ID

        t_id = next((t['id'] for t in turmas if t['nome'] ==
turma_selecionada), None)



        if st.button("Matricular!"):

            # Pega o login do aluno

            u_login = next((a['usuario_login'] for a in
d['alunos'] if decripto(a['nome_criptografado']) ==
aluno_selecionado), None)



            if matricula_aluno(u_login, t_id):

                st.success(f"Beleza, {aluno_selecionado} agora
tá em {turma_selecionada}!")

                st.rerun()

            else:

                st.error("Erro na matrícula.")

    else:

        st.warning("Não tem alunos sem turma ou turmas
cadastradas.")



def ui_diario_eletronico():
```

```
    st.subheader("Diário Eletrônico: Lançar Notas")
    st.info("Só edite NP1, NP2 e PIM. O PIM é a nota global do projeto.")

    d = carrega_dados()
    materia_editar = st.selectbox("Matéria para editar:", [m for m in materias if m != "PIM"])
    tabela_notas = []

    for a in d['alunos']:
        nome = decripto(a['nome_criptografado'])
        ra = decripto(a['ra_criptografado'])
        notas = a.get('notas_c', notas_vazias())

        # Pega notas com conversão simples
        np1 = float(notas.get(materia_editar, {}).get('NP1') or 0.0)
        np2 = float(notas.get(materia_editar, {}).get('NP2') or 0.0)
        pim = float(notas.get('PIM') or 0.0)

        # Média UNIP: NP1*4 + NP2*4 + PIM*2
        media = ((np1 * 4) + (pim * 2) + (np2 * 4)) / 10
```

```
        status = "Aprovado" if media >= 7.0 else "Reprovado"
if media > 0 else "Sem Nota"



        tabela_notas.append({

            'Nome': nome,

            'RA': ra,

            'NP1': np1,

            'NP2': np2,

            'PIM': pim,

            'Média': f"{media:.2f}",

            'Status': status,

            'login_aluno': a['usuario_login']

        })



    st.markdown("### Tabela de Notas (Editar)")

    tabela_editada = st.data_editor(

        tabela_notas,

        column_config={

            "Nome":
st.column_config.TextColumn(disabled=True),

            "RA": st.column_config.TextColumn(disabled=True),

            "Média":
st.column_config.TextColumn(disabled=True),

            "Status":
st.column_config.TextColumn(disabled=True),
```

```
            "login_aluno":
st.column_config.TextColumn(disabled=True),
            "NP1":
st.column_config.NumberColumn(min_value=0.0, max_value=10.0,
format="%.2f"),
            "NP2":
st.column_config.NumberColumn(min_value=0.0, max_value=10.0,
format="%.2f"),
            "PIM":
st.column_config.NumberColumn(min_value=0.0, max_value=10.0,
format="%.2f"),
        },
        hide_index=True,
        num_rows="dynamic"
    )


    st.markdown("---")
    if st.button("Salvar Tudo e Chamar o C"):
        # Salva no JSON
        for linha in tabela_editada:
            for aluno in d['alunos']:
                if aluno['usuario_login'] ==
linha['login_aluno']:

                    # Garante que tem a estrutura de notas
                    aluno.setdefault('notas_c',
notas_vazias())
```

```
                    # Pega os valores da tabela
                    np1_novo = float(linha.get('NP1') or 0.0)
                    np2_novo = float(linha.get('NP2') or 0.0)
                    pim_novo = float(linha.get('PIM') or 0.0)

                    # Salva
                    aluno['notas_c'][materia_editar]['NP1'] =
np1_novo
                    aluno['notas_c'][materia_editar]['NP2'] =
np2_novo
                    aluno['notas_c']['PIM'] = pim_novo

                    # Recalcula e salva a média (dic extra)
                    media_final = ((np1_novo * 4) + (pim_novo
* 2) + (np2_novo * 4)) / 10
                    aluno['notas_c'].setdefault('medias_calc',
{})
aluno['notas_c']['medias_calc'][materia_editar] = media_final

                    break

        salva_dados(d)
        st.success("Notas salvas. Beleza!")
```

```
        # Chama a integração C
        sucesso, msg = envia_para_c()

        if sucesso:
            st.success(f"C: {msg}")
        else:
            st.error(f"C FALHOU: {msg}")
        st.rerun()

def ui_gerenciar_usuarios(d, user_atual):
    st.subheader("APAGAR Usuários")

    lista_excluir = [u for u in d['usuarios'].keys() if u != user_atual]

    if not lista_excluir:
        st.warning("Só tem você aqui.")
        return

    usuario_excluir = st.selectbox("Quem apagar?", lista_excluir)

    st.error(f"CUIDADO: Apagar '{usuario_excluir}' é para sempre.")
```

```
    if st.button(f"APAGAR TUDO ({usuario_excluir})"):
        sucesso, mensagem = apagar_user(usuario_excluir)
        if sucesso:
            st.success(mensagem)
        else:
            st.error(mensagem)
        st.rerun()

# --- Funções de Interface Principal ---

def tela_prof_admin(tipo):
    st.title(f"Área de {tipo.upper()}:
{st.session_state['usuario']}")
    d = carrega_dados()

    menu = ["Dashboard", "Cadastrar Turmas", "Gestão de
Turmas", "Diário Eletrônico (Notas C)"]
    if tipo == 'admin':
        menu.append("Gerenciar Usuários")

    escolha = st.sidebar.selectbox("O que fazer?", menu)

    if escolha == "Dashboard":
```

```
        st.subheader("Visão Geral")

        if tipo == 'professor':

            st.info("DICA: Use o diário eletrônico para lançar
as notas. É mais fácil que papel.")

        else:

            st.info("DICA: A arquitetura Cliente-Servidor é
boa porque o processamento pesado fica no Servidor.")


        st.metric("Total de Alunos", len(d['alunos']))

        st.metric("Total de Turmas", len(d['turmas']))


    elif escolha == "Cadastrar Turmas":

        st.subheader("Criar Turma")

        nome_t = st.text_input("Nome da Turma:")

        if st.button("Criar Turma"):

            if cadastra_turma(nome_t):

                st.success(f"Turma '{nome_t}' criada.")

            else:

                st.error("Turma já existe.")


        turmas_existentes = d['turmas']

        st.markdown("---")

        st.text("Turmas Existentes:")

        for t in turmas_existentes:
```

```
            st.write(f"- {t['nome']} (ID: {t['id']})")


    elif escolha == "Gestão de Turmas":
        ui_gestao_turmas()


    elif escolha == "Diário Eletrônico (Notas C)":
        ui_diario_eletronico()


    elif escolha == "Gerenciar Usuários":
        ui_gerenciar_usuarios(d, st.session_state['usuario'])



def tela_aluno():
    st.title(f"Área do Aluno: {st.session_state['usuario']}")
    st.sidebar.header("Menu")


    d = carrega_dados()
    aluno = next((a for a in d['alunos'] if a['usuario_login']
== st.session_state['usuario']), None)


    if not aluno:
        st.error("Seu registro não foi achado. Fale com o
professor.")
        return
```

```
    st.subheader("Dados")
    st.metric("ID da Turma", aluno['turma_id'])

    st.markdown("---")
    st.subheader("Minhas Notas")

    notas = aluno.get('notas_c', notas_vazias())

    materia_escolhida = st.selectbox("Ver a nota de:",
materias.copy())

    st.markdown("---")

    if materia_escolhida == "PIM":
        pim_nota = float(notas.get("PIM") or 0.0)
        st.metric("Nota do PIM", f"{pim_nota:.2f}")

    else:
        info = notas.get(materia_escolhida, {"NP1": 0.0,
"NP2": 0.0})

        np1 = float(info.get("NP1") or 0.0)
        np2 = float(info.get("NP2") or 0.0)
```

```
        pim = float(notas.get("PIM") or 0.0)


        st.metric("NP1", f"{np1:.2f}")
        st.metric("NP2", f"{np2:.2f}")


        # Média UNIP 4/2/4
        media = ((np1 * 4) + (pim * 2) + (np2 * 4)) / 10


        st.metric("Média Final da Matéria", f"{media:.2f}")


        status = "Aprovado" if media >= 7 else "Reprovado"
        st.metric("Status", status)


    # Cálculo da média semestral geral:
    st.markdown("---")


    if materia_escolhida != "PIM":
        pim_global = float(notas.get("PIM") or 0.0)
        medias_materias = []
        for materia, info in notas.items():
            if materia == "PIM" or not isinstance(info, dict):
                continue


            n1 = float(info.get("NP1") or 0.0)
```

```
            n2 = float(info.get("NP2") or 0.0)


            media_materia = ((n1 * 4) + (pim_global * 2) + (n2
* 4)) / 10
            medias_materias.append(media_materia)


        if medias_materias:
            media_geral = sum(medias_materias) /
len(medias_materias)
            st.subheader("Média Geral")
            st.metric("Média Aritmética de Tudo",
f"{media_geral:.2f}")


    st.info("DICA: Use o sistema direito para garantir que os
dados acadêmicos estão certos.")


# --- Main ---
def main():
    if 'logado' not in st.session_state:
        st.session_state['logado'] = False


    st.sidebar.button("Sair (Logout)", on_click=lambda:
st.session_state.update({'logado': False, 'usuario': None,
'tipo': None}))


    if st.session_state.get('logado'):
```

```
        if st.session_state['tipo'] in ['professor', 'admin']:
            tela_prof_admin(st.session_state['tipo'])
        elif st.session_state['tipo'] == 'aluno':
            tela_aluno()
    else:
        # Tela de Login/Cadastro na página principal
        st.title("Sistema Acadêmico Colaborativo 2.0")
        st.subheader("Login / Cadastro Novo")

        # Cadastro na Sidebar
        st.sidebar.header("Novo Cadastro")
        u_novo = st.sidebar.text_input("Usuário novo:")
        s_nova = st.sidebar.text_input("Senha nova:",
type="password")
        n_completo = st.sidebar.text_input("Nome Completo:")
        tipo_user = st.sidebar.radio("Tipo:", ['aluno',
'professor', 'admin'])

        if st.sidebar.button("Registrar"):
            if u_novo and s_nova and n_completo:
                if cad_user(u_novo, s_nova, n_completo,
tipo_user):
                    st.sidebar.success(f"Beleza, {u_novo}
agora tá cadastrado!")
                else:
```

```
                    st.sidebar.error("Usuário já existe.")
            else:
                st.sidebar.warning("Preencha TUDO para
cadastrar.")

        st.markdown("---")

        # Login
        u_login = st.text_input("Seu Usuário:")
        s_login = st.text_input("Sua Senha:", type="password")

        if st.button("Entrar"):
            tipo = login_ok(u_login, s_login)
            if tipo:
                st.session_state['logado'] = True
                st.session_state['usuario'] = u_login
                st.session_state['tipo'] = tipo
                st.rerun()
            else:
                st.error("Usuário ou senha errados.")

if __name__ == '__main__':
    main()
```

10.3 Código sistema em C

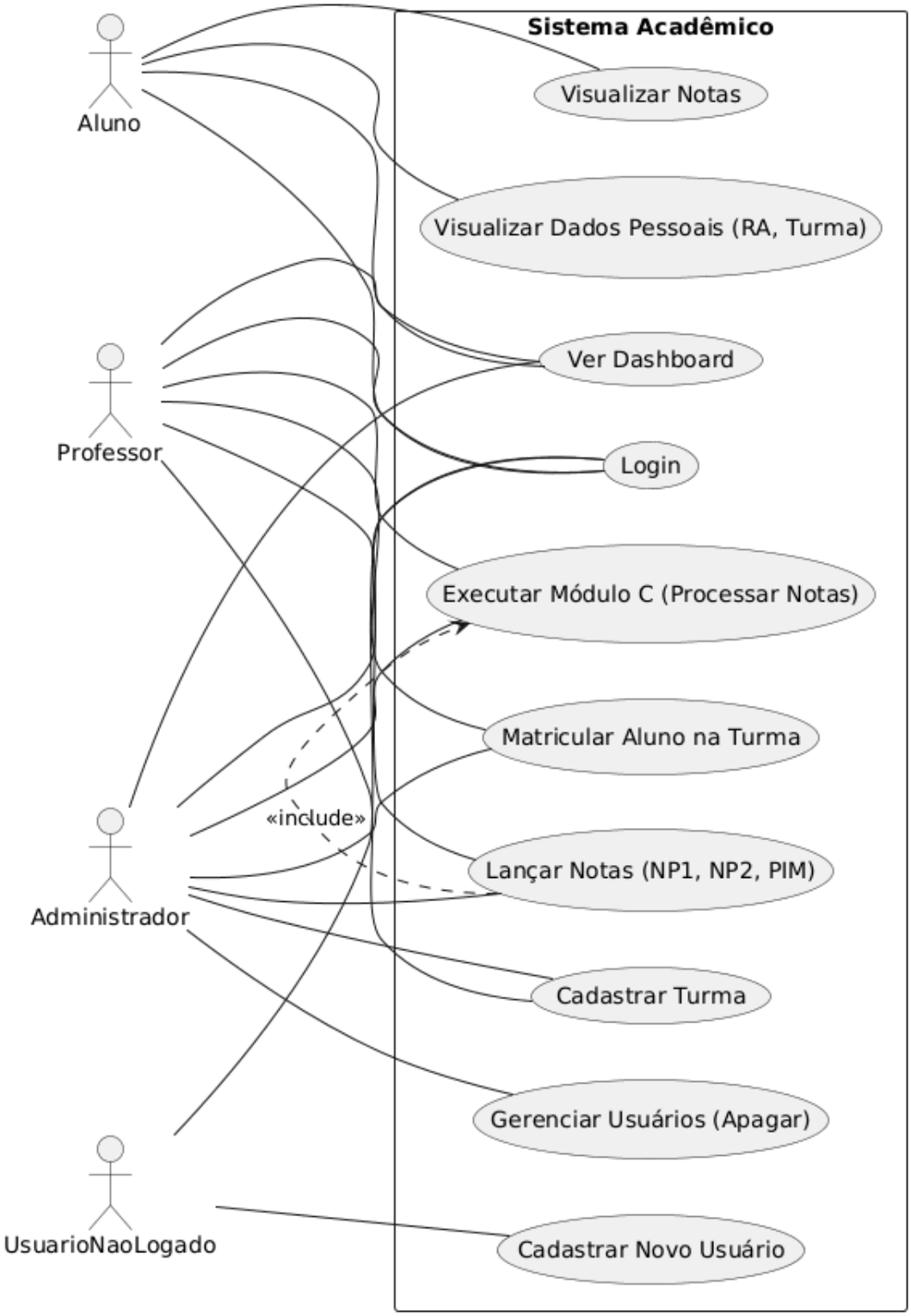
```c
// Arquivo: modulo_c.c
// Função: Atualizar notas do arquivo binário com a lista de RAs
// Objetivo: Garantir que o binário tenha só os RAs do texto

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINHA 15
#define ARQ_BIN "dados_notas.dat"
#define ARQ_RAS "ras_para_c.txt"
#define ARQ_TEMP "dados_temp.dat"

// Estrutura do aluno
struct Notas {
    int ra;
    float np1;
    float np2;
    float pim;
};

int main() {
```

```
    FILE *f_ras, *f_bin, *f_temp;

    struct Notas aluno;

    struct Notas novo_aluno;

    char linha[MAX_LINHA];

    int ra_texto;


    f_ras = fopen(ARQ_RAS, "r");

    if (!f_ras) return 0; // se nao existe lista, sai


    f_bin = fopen(ARQ_BIN, "rb");

    f_temp = fopen(ARQ_TEMP, "wb");


    int bin_existe = (f_bin != NULL);


    while (fgets(linha, sizeof(linha), f_ras)) {

        linha[strcspn(linha, "\n")] = 0; // tira \n

        ra_texto = atoi(linha);


        int achei = 0;


        if (bin_existe) {

            rewind(f_bin);

            while (fread(&aluno, sizeof(struct Notas), 1,
f_bin)) {
```

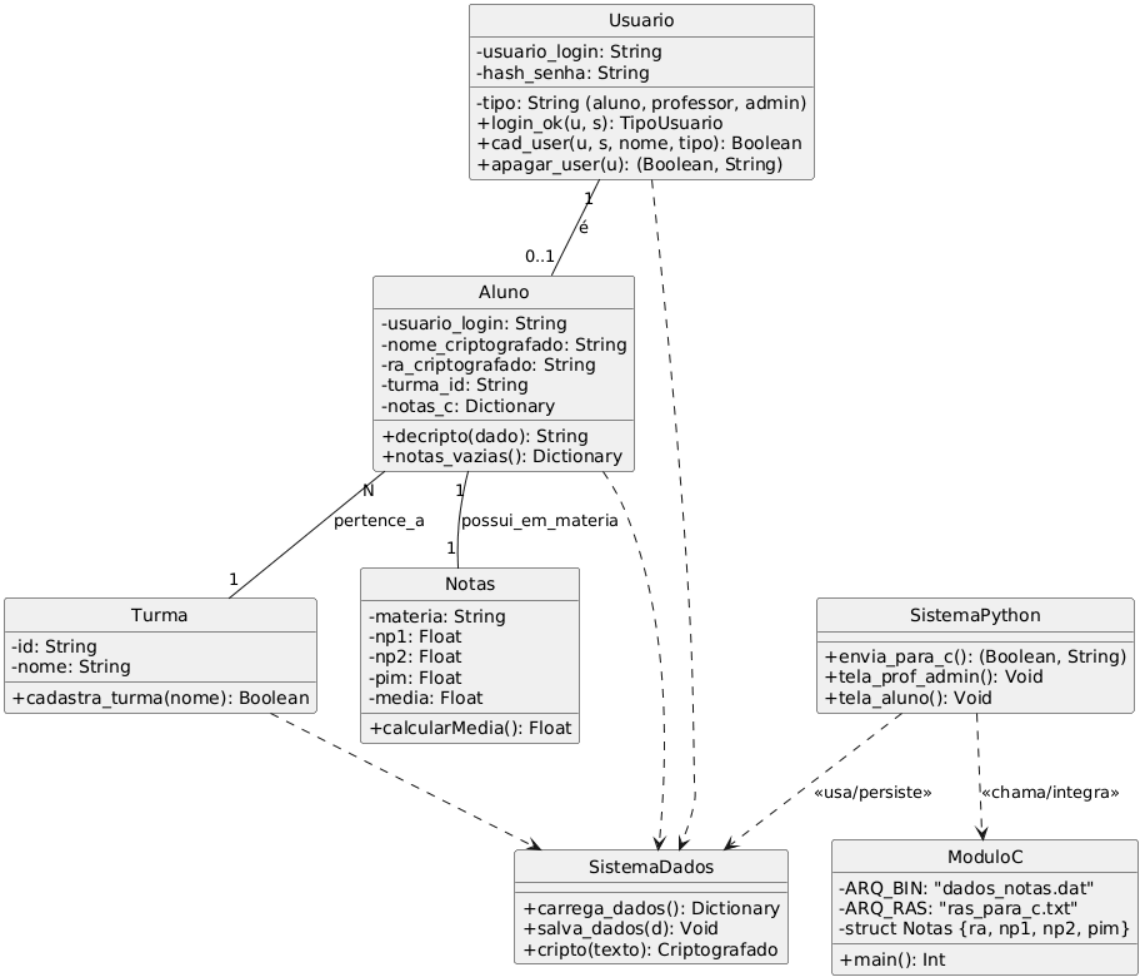
```
                if (aluno.ra == ra_texto) {
                    fwrite(&aluno, sizeof(struct Notas), 1,
f_temp);
                    achei = 1;
                    break;
                }
            }
        }

        if (!achei) {
            novo_aluno.ra = ra_texto;
            novo_aluno.np1 = 0;
            novo_aluno.np2 = 0;
            novo_aluno.pim = 0;
            fwrite(&novo_aluno, sizeof(struct Notas), 1,
f_temp);
        }
    }

    fclose(f_ras);
    if (f_bin) fclose(f_bin);
    fclose(f_temp);

    rename(ARQ_TEMP, ARQ_BIN);
```

```
    return 0;
}
```

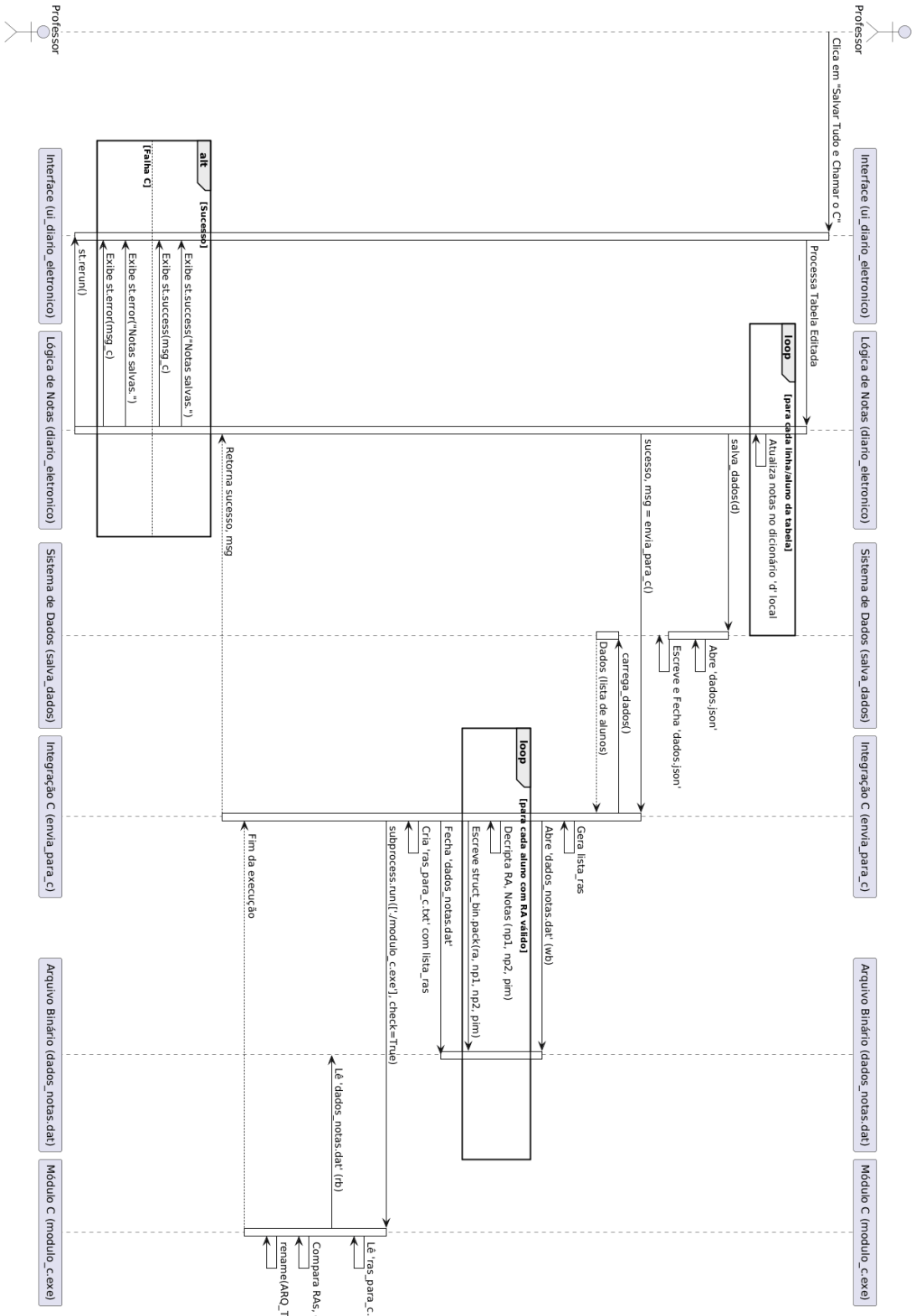
10.4. Diagrama de caso de uso



10.5. Diagrama de classe



10.6. Diagrama de sequência



10.7. Protótipo Desktop

10.8. Protótipo mobile

10.9.Slides apresentação

PIM – Projeto Integrado Multidisciplinar

2025

UNIP UNIVERSIDADE PAULISTA

2025

O problema: Gestão Descentralizada

Fragmentação: Controles realizados de forma descentralizada em planilhas, e-mails e aplicativos de mensagens.

Ineficiência: Dificuldade no acompanhamento centralizado, gerando retrabalho e falta de padronização.

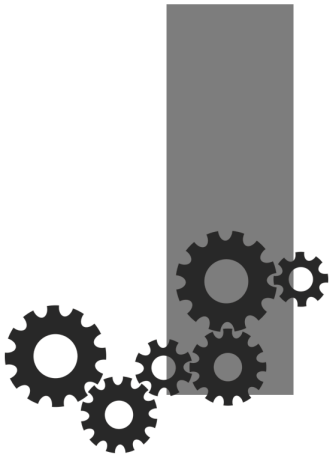
Falta de Sustentabilidade: Dependência excessiva de diários de classe, relatórios e atividades em papel.

A solução: Plataforma Integrada

Gestão acadêmica: Centraliza o cadastro de turma, alunos, o registro de aulas e o diário eletrônico.

Arquitetura cliente-servidor: Permite o acesso simultâneo de professores e alunos em maquinas distintas na rede local (LAN).

Foco em sustentabilidade: Elimina o uso de papel, movendo todos os relatórios e diários para o formato digital.



2025

Pilares Tecnológicos do projeto

Desenvolvimento hibrido (Python + C)

- Python (Streamlit): Utilizado para a interface principal do sistema (web app), gestão de usuários e criptografia de dados (JSON)
- Linguagem C: Desenvolve um módulo crítico de baixo nível, responsável pelo processamento e persistência binaria das notas.

Pilares Tecnológicos do projeto

Gestão ágil e inovação

- Eng. De Software: Gestão do projeto com metodologia Scrum, Trello (Quadro Kanban) e modelagem UML (Astah).
- Inteligência artificial: Usada como ferramenta de apoio para otimização de código e geração de documentação técnica.

Obrigado!

FICHA DE CONTROLE DO PIM

Grupo Nº

Ano 2° Semestre 2025

Período: Noturno

Orientador: Karhyne Assis

Tema:

Alunos:

RA	Nome	E-mail	Curso	Visto do aluno

Registros:

Data / Sprint	Atividades e Entregas	Observações Técnicas
Sprint 1 – 10/09/2025	Reunião inicial e levantamento do **backlog do produto** com base nas necessidades de professores e alunos.	Todos os membros participaram do brainstorming sobre o escopo e objetivos do projeto PIM.
Sprint 2 – 17/09/2025	Definição da **metodologia ágil (Scrum)**, requisitos funcionais e não funcionais. Início do backlog e **priorização das user stories**.	Foram definidos critérios de aceitação e épicos principais no **Trello**.
Sprint 3 – 25/09/2025	Divisão de funções e criação dos canais de comunicação (**Discord** e **Trello**). Organização do **Product Backlog**.	Papéis definidos: Scrum Master, Product Owner e equipe de desenvolvimento.
Sprint 4 – 08/10/2025	Definição da **arquitetura de rede** e dos equipamentos necessários para suporte à aplicação.	Identificação dos componentes de infraestrutura e ambientes de teste.
Sprint 5 – 17/10/2025	Revisão de tarefas realizadas e **retrospectiva** da equipe. Atualização do backlog com ajustes de prioridade.	Avaliação de desempenho do time e planejamento para a prototipação.

Sprint 6 – 20/10/2025	Criação dos **protótipos de interface** no Figma e modelagem inicial dos **diagramas UML** (casos de uso e classes).	Diagramas desenvolvidos no Astah para representar as interações e entidades do sistema.
Sprint 7 – 22/10/2025	Início da **codificação do sistema**, com base no backlog priorizado.	Implementação das principais funcionalidades de cadastro e autenticação.
Sprint 8 – 23/10/2025	Revisão de progresso e coleta de feedback sobre o desenvolvimento. Atualização do **Sprint Backlog**.	Ajustes de código e correção de bugs identificados.
Sprint 9 – 24/10/2025	Preparação e entrega da **documentação técnica**, dos **diagramas UML** finais e dos **slides de apresentação**.	Conclusão do design e revisão dos requisitos atendidos.
Sprint 10 – 25/10/2025	**Testes de sistema e validação** das funcionalidades implementadas (testes de caixa preta).	Registro de bugs e resultados de testes no backlog de melhoria.
Sprint 11 – 26/10/2025	**Encerramento do projeto**, revisão geral e apresentação final.	Entrega completa do sistema, documentação e relatório final.

- **Backlog do produto:** estruturado no Trello, dividido por épicos e user stories.
- **Metodologia:** Scrum com sprints semanais e retrospectivas ao final de cada ciclo.
- **Ferramentas utilizadas:** Astah (UML), Figma (protótipos), Trello (kanban), Discord (comunicação), GitHub (versão de código).
- **Diagramas UML criados:**
  - Diagrama de Casos de Uso
  - Diagrama de Classes
  - Diagrama de Sequência