📚 Plataforma de Estudos Universitários: Documentação da Arquitetura

Este documento detalha a arquitetura e o funcionamento da Plataforma de Estudos Universitários, servindo como guia de referência para o projeto. Ele aborda a modelagem das entidades principais, o módulo de colaboração e os fluxos de processos essenciais.

---

1. Visão Geral e Objetivo da Aplicação

A Plataforma de Estudos Universitários é projetada para ser uma ferramenta centralizada que auxilia estudantes na **organização eficiente da vida acadêmica**. Seu objetivo principal é gerenciar disciplinas, tarefas e anotações, com a funcionalidade chave de **colaboração em atividades específicas**. A arquitetura foi pensada de forma modular, separando o essencial da organização individual das funcionalidades de compartilhamento.

---

2. Modelagem e Arquitetura do Sistema

A modelagem do sistema é realizada utilizando **Diagramas de Classes UML** para a estrutura estática (entidades, atributos, métodos e relações) e **Diagramas de Atividades UML** para os fluxos de processos dinâmicos.

2.1. Diagrama de Classes: Core do Sistema (Domínio Principal)

Este diagrama representa as entidades fundamentais da aplicação e as regras de negócio primárias para a organização individual do estudante.

Classes Principais:

- `Usuário`
    - **Função:** A entidade central e ativa do sistema. O `Usuário` inicia a **criação** de todas as demais entidades (disciplinas, tarefas, anotações), refletindo a responsabilidade de iniciar ações na plataforma.
    - **Propriedade:** Todas as `Disciplinas`, `Tarefas` e `Anotações` criadas são **necessariamente vinculadas** a um único `Usuário`. Um `Usuário` pode criar `0..*` instâncias de cada uma dessas entidades.
    - **Atributos:** `id`, `nome`, `email`, `senhaHash`, `perfil`.
    - **Métodos:** `+ criarDisciplina()`, `+ criarTarefa()`, `+ criarAnotacao()` (e suas variações específicas), além de métodos de gestão de perfil.
- `Disciplina`
    - **Função:** Atua como um **contêiner lógico** para organizar o conteúdo de uma matéria específica.
    - **Relacionamento:** É criada e gerida por um `Usuário` (`1 Usuário` possui `0..* Disciplinas`). A `Disciplina` não possui métodos de criação, pois seu papel é de agregação passiva.

- `Tarefa`
    - **Função:** Representa um compromisso ou atividade específica.
    - **Relacionamento com `Disciplina`:** Pode ou não estar vinculada a uma `Disciplina`. A associação é: `Disciplina 0..* -- 0..1 Tarefa`. Isso permite `Tarefas` avulsas ou ligadas a uma única `Disciplina`.
    - **Tipos de Tarefa:** Possui um atributo `tipoTarefa` (do Enum `TipoTarefa` como `Trabalho`, `Prova`, `Leitura`, `Projeto`, etc.), permitindo diferenciar atividades sem criar classes separadas.
    - **Propriedade:** Cada `Tarefa` é criada e pertence a um `Usuário` (`1 Usuário` cria `0..* Tarefas`).
- `Anotação`
    - **Função:** Permite o registro flexível de informações, notas de aula, resumos ou ideias.
    - **Relacionamento Flexível:** Pode ser vinculada a `0..1 Disciplina` **E/OU** `0..1 Tarefa`. Isso permite anotações de aula, anotações de trabalho, anotações que servem a ambos, ou anotações "soltas".
    - **Propriedade:** Cada `Anotação` é criada e pertence a um `Usuário` (`1 Usuário` cria `0..* Anotações`).

Enums Utilizados (Neste Módulo):

- `Perfil`
- `StatusDisciplina`
- `StatusTarefa`
- `TipoTarefa`

2.2. Diagrama de Classes: Módulo de Colaboração e Permissões

Este diagrama detalha as classes e as regras que regem o compartilhamento de recursos e as permissões de acesso, complementando o core do sistema.

Classes do Módulo de Colaboração:

- `ConviteColaboracao`
    - **Função:** Gerencia o ciclo de vida dos convites para colaborar em recursos específicos.
    - **Relacionamentos com `Usuário`:**
        - Como **Remetente**: `1 Usuário envia 0..* ConvitesColaboracao`.
        - Como **Destinatário**: `0..* Usuários recebe 0..1 ConviteColaboracao`.
    - **Alvo do Convite:** Pode ter como alvo uma `Tarefa` ou uma `Anotação` (`0..*` do alvo para `0..1` no convite), especificado pelo atributo `tipoRecurso` (do Enum `TipoRecursoColaboracao`).
    - **Atributos Chave:** `idRemetente`, `emailDestinatario`, `idDestinatario`, `idRecurso`, `tipoRecurso`, `status` (do Enum `StatusConvite`).
- `ColaboracaoTarefa`
    - **Função:** Classe associativa que representa a permissão concedida a um `Usuário` sobre uma `Tarefa` específica.

    - **Relacionamento:** Conecta `Usuário` e `Tarefa` (`1 Usuário -- 0..* ColaboracaoTarefa -- 1 Tarefa`).
    - **Atributos Chave:** `idUsuario`, `idTarefa`, `permissao` (do Enum `Permissao`).

- `ColaboracaoAnotacao`
    - **Função:** Classe associativa que representa a permissão concedida a um `Usuário` sobre uma `Anotação` específica.
    - **Relacionamento:** Conecta `Usuário` e `Anotação` (`1 Usuário -- 0..* ColaboracaoAnotacao -- 1 Anotação`).
    - **Atributos Chave:** `idUsuario`, `idAnotacao`, `permissao` (do Enum `Permissao`).

Enums Específicos da Colaboração:

- `TipoRecursoColaboracao`: (`TAREFA`, `ANOTACAO`) – Define o tipo de recurso alvo do convite/colaboração.
- `StatusConvite`: (`PENDENTE`, `ACEITO`, `RECUSADO`, `EXPIRADO`) – Define o estado de um convite.
- `Permissao`: (`VISUALIZAR`, `EDITAR`) – Define o nível de acesso de um colaborador a um recurso.

Regras de Acesso e Restrições (Constraint Crucial):

Uma nota explícita no diagrama detalha a regra fundamental de acesso restrito para usuários convidados:

> **{Constraint:** Usuários convidados (via `ColaboracaoTarefa`/`ColaboracaoAnotacao`) **não têm acesso direto à** `Disciplina`. O acesso é estritamente limitado ao **Recurso compartilhado** (`Tarefa` ou `Anotação`) e a seus dependentes diretos (e.g., `Anotações` vinculadas à `Tarefa` compartilhada).
>
> - Se uma `Tarefa` é compartilhada, o convidado acessa a `Tarefa` e **APENAS** as `Anotações` que estão vinculadas *diretamente* a essa `Tarefa`.
> - Se uma `Anotação` é compartilhada, o convidado acessa **SOMENTE** essa `Anotação` em si.
> - Em ambos os cenários, não há visibilidade ou acesso à `Disciplina` à qual a `Tarefa` ou `Anotação` possam pertencer. O acesso é sempre "da camada da Tarefa para baixo" ou "somente a Anotação em si".**}**

2.3. Diagrama de Atividades: Fluxo de Convite e Permissões

Este diagrama ilustra o processo dinâmico de como um convite de colaboração é enviado, processado e respondido, bem como o fluxo de gerenciamento de permissões.

Participantes (Swimlanes):

- `Usuário Convidante`: O ator que inicia o processo de compartilhamento.
- `Sistema`: A própria aplicação, responsável pelo processamento lógico, banco de dados e comunicação.
- `Usuário Convidado`: O ator que recebe e responde ao convite.

Fluxo do Convite:

1. **Início:** `Usuário Convidante` inicia a ação de compartilhar um recurso.

2. `Usuário Convidante` informa o e-mail do convidado e o recurso (`Tarefa` ou `Anotação`) a ser compartilhado.
3. `Sistema` processa o convite (verifica usuário, cria registro `ConviteColaboracao` como "Pendente").
4. `Sistema` **notifica** o `Usuário Convidado` sobre o convite (via e-mail ou notificação in-app).
5. `Usuário Convidado` acessa o convite (clica no link/notificação).
6. `Sistema` redireciona para uma página de resposta.
7. `Usuário Convidado` decide: **Aceitar** ou **Recusar**.
8. **Caminho "Aceitar":** `Sistema` atualiza o `ConviteColaboracao` para "Aceito" e cria uma entrada em `ColaboracaoTarefa` ou `ColaboracaoAnotacao` (com permissão inicial).
9. **Caminho "Recusar":** `Sistema` atualiza o `ConviteColaboracao` para "Recusado".
10. `Sistema` notifica o `Usuário Convidante` sobre a resposta (opcional), fechando o ciclo.
11. **Fim:** O fluxo do convite é concluído.

Fluxo de Gerenciamento de Permissões (Complementar):

Um fluxo adicional (ou seção do mesmo diagrama) detalha como o `Usuário Convidante` pode, posteriormente, **visualizar os colaboradores**, **alterar suas permissões** (entre `VISUALIZAR` e `EDITAR`) ou **remover** o acesso de um colaborador, com o `Sistema` processando e notificando as alterações.

---

3. Princípios de Design e Boas Práticas

A modelagem do sistema adere a princípios de design orientado a objetos:

- **Princípio da Responsabilidade Única (SRP):** As classes são projetadas com responsabilidades claras (ex: `Usuário` para iniciação de ações, `Disciplina` como agregador, `ColaboracaoX` para gerenciar permissões).
- **Encapsulamento:** Atributos são privados (`-`), e o acesso e modificação são controlados por métodos públicos (`+`).
- **Modularidade:** A separação das funcionalidades em módulos (core vs. colaboração) e a representação em diagramas distintos demonstram uma arquitetura organizada e extensível.
- **Clareza e Rastreabilidade:** O uso consistente de Enums, multiplicidades e rótulos de associação garante que as regras de negócio sejam explícitas e de fácil compreensão no modelo.

Este documento consolidado fornece um panorama completo do sistema, suas funcionalidades e a lógica por trás de sua modelagem, sendo um recurso inestimável para qualquer pessoa que precise entender ou trabalhar no projeto.