

UNIBRASIL  
ENGENHARIA DE SOFTWARE  
PRÁTICAS PROFISSIONAIS

GABRIEL GUERRA  
GUSTAVO COSTA  
GABRYEL ZANELLA

**DOCUMENTAÇÃO DE  
PROJETO DE SOFTWARE COGNITUS**

**Curitiba – Paraná  
2025**

**GABRIEL GUERRA**  
**GUSTAVO COSTA**  
**GABRYEL ZANELLA**

**DOCUMENTAÇÃO DE**  
**PROJETO DE SOFTWARE COGNITUS**

Trabalho para o curso de  
Engenharia de Software  
da UniBrasil na matéria de  
Práticas Profissionais  
Orientador: Leonel da Rocha

**Curitiba – PR**

**2025**

## Sumário

<b>1</b>	<b>Introdução .....</b>	<b>4</b>
<b>2</b>	<b>Justificativa e Objetivos.....</b>	<b>4</b>
<b>3</b>	<b>Tecnologias Utilizadas .....</b>	<b>5</b>
<b>4</b>	<b>DER, MER, etc .....</b>	<b>6</b>
<b>5</b>	<b>Arquitetura de Software .....</b>	<b>7</b>
5.1	Usabilidade .....	7
5.2	Manutenibilidade .....	8
5.3	Segurança.....	8
5.4	Desempenho.....	8
5.5	Portabilidade .....	9
<b>6</b>	<b>O Sistema Interativo.....</b>	<b>9</b>
<b>7</b>	<b>API do Software.....</b>	<b>10</b>
<b>8</b>	<b>Conclusão.....</b>	<b>11</b>

# 1 Introdução

O Sistema de Gestão Acadêmica é uma aplicação web moderna desenvolvida para otimizar a administração de instituições educacionais. Construído com **Blazor WebAssembly**, **.NET 8**, **MudBlazor** e **SQL Server**, o sistema oferece uma interface responsiva e funcionalidades robustas para gerenciar cadastros de alunos, professores, cursos, notas e frequência, além de dashboards analíticos para acompanhamento do desempenho acadêmico. A aplicação é acessível via navegadores modernos, garantindo flexibilidade e usabilidade para administradores, professores e outros usuários autorizados. O objetivo principal é proporcionar uma solução integrada que automatize processos acadêmicos, reduza erros manuais e facilite a tomada de decisões com base em dados.

## 2 Justificativa e Objetivos

### Justificativa

A gestão acadêmica em instituições educacionais frequentemente enfrenta desafios como burocracia, erros em registros manuais e dificuldade na consolidação de informações. Esses problemas impactam a eficiência operacional e a experiência de alunos, professores e administradores. Um sistema web automatizado, com interface amigável e relatórios analíticos, resolve essas questões ao centralizar dados, melhorar a rastreabilidade e oferecer visualizações em tempo real do desempenho acadêmico.

### Objetivos

- **Objetivo Geral:** Desenvolver uma aplicação web para gestão acadêmica que integre cadastros, notas, frequência e análises em uma plataforma única, acessível e segura.
- **Objetivos Específicos:**
  - Implementar operações CRUD (Create, Read, Update, Delete) para entidades como Aluno, Professor, Curso, Nota e Frequência.
  - Disponibilizar dashboards interativos com gráficos para monitoramento de desempenho e frequência.
  - Garantir uma API segmentada por responsabilidades, com suporte a consultas filtradas e paginação.
  - Assegurar usabilidade com uma interface responsiva e intuitiva, utilizando componentes do MudBlazor.
  - Facilitar a manutenção e escalabilidade com uma arquitetura modular e padrões modernos de desenvolvimento.

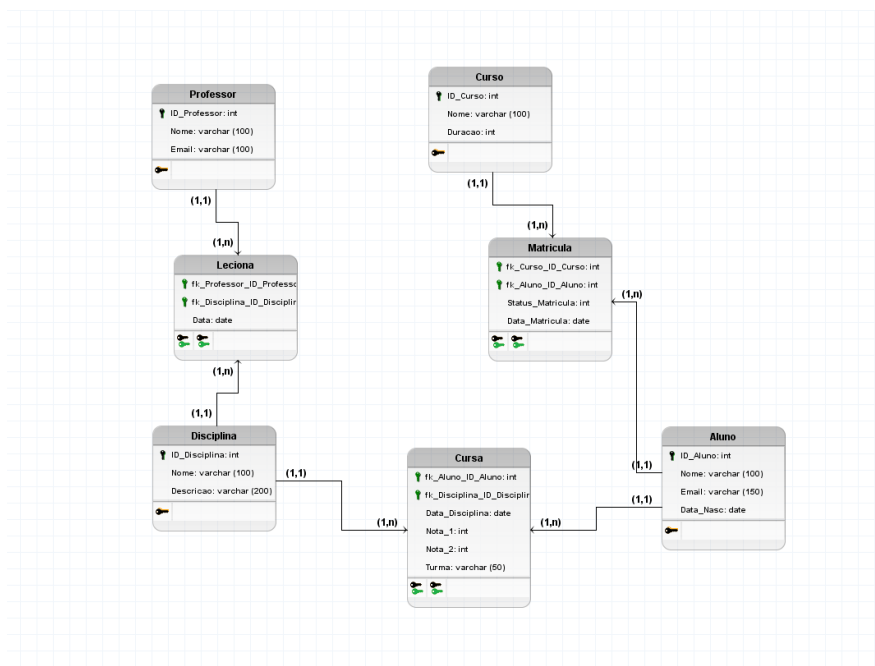
### 3 Tecnologias Utilizadas

O sistema foi desenvolvido com um conjunto de tecnologias modernas, garantindo desempenho, escalabilidade e facilidade de manutenção. Abaixo está a lista detalhada:

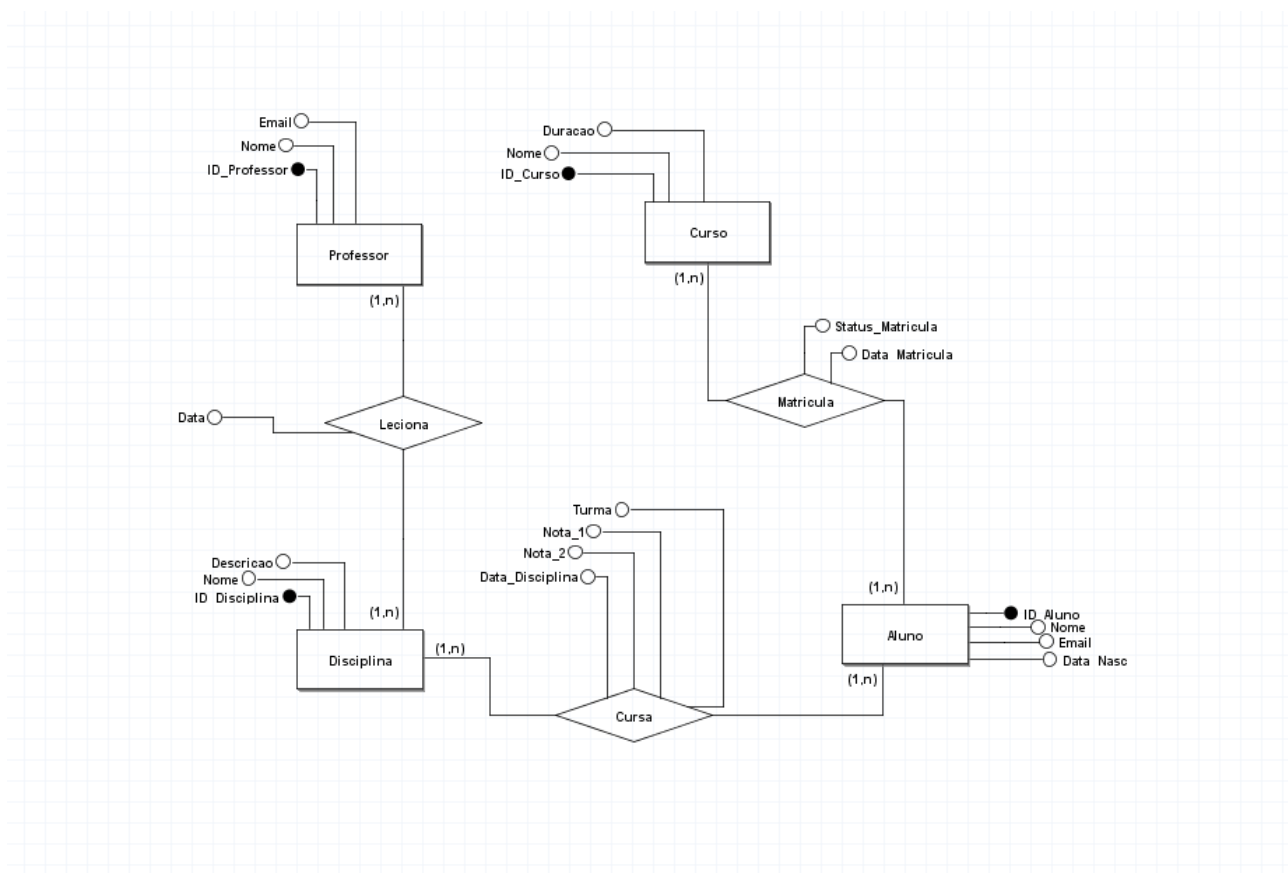
- **.NET 8 e C#:** Framework principal para backend e lógica de negócio, oferecendo suporte multiplataforma e alto desempenho.
- **Blazor WebAssembly:** Tecnologia para o frontend, permitindo a execução de C# diretamente no navegador, com interações rápidas e sem dependência de recarregamento de páginas.
- **MudBlazor:** Biblioteca de componentes UI para Blazor, fornecendo elementos responsivos, acessíveis e personalizáveis, como tabelas, formulários e modais.
- **Entity Framework Core:** ORM (Object-Relational Mapping) para interação com o banco de dados SQL Server, com suporte a mapeamento por atributos e Fluent API.
- **SQL Server:** Banco de dados relacional para armazenamento de dados, garantindo consistência e suporte a consultas complexas.
- **Minimal APIs (.NET 8):** Utilizadas no backend para criar endpoints REST leves e performáticos, com suporte a operações CRUD e paginação.
- **Blazorise.Charts:** Biblioteca para renderização de gráficos nos dashboards, permitindo visualizações como barras, linhas e pizza para análise de dados.
- **Ferramentas de Desenvolvimento:**
  - **IDE:** Visual Studio 2022 ou Rider, compatíveis com .NET 8.
  - **Controle de Versão:** Git, hospedado no GitHub.
  - **Gerenciamento de Banco:** SQL Server Management Studio (SSMS) ou Azure Data Studio.

## 4 DER, MER, etc

### Diagrama Entidade-Relacionamento (DER)



### Modelo Entidade-Relacionamento (MER)



## 5 Arquitetura de Software

O sistema adota uma arquitetura em camadas, separando responsabilidades para facilitar manutenção, testes e escalabilidade. A estrutura do projeto é organizada como segue:

SistemaAcademico/

- |— API/ → Minimal APIs para endpoints REST
- |— Web/ → Projeto Blazor WebAssembly (frontend)
- |— Data/ → Contexto EF Core e camada de acesso a dados (DAL)
- |— Models/ → Classes de entidades e DTOs
- |— Services/ → Camada de serviços para lógica de negócio e consumo da API

- **API:** Contém os endpoints REST implementados com Minimal APIs, organizados por entidade (/api/alunos, /api/cursos, etc.). Suporta autenticação e autorização (sugerida com JWT).
- **Web:** Projeto Blazor WebAssembly, com páginas Razor e componentes MudBlazor para interface.
- **Data:** Inclui o DbContext do EF Core, repositórios e migrações para o banco de dados.
- **Models:** Define entidades (ex.: Aluno, Curso) e DTOs (ex.: AlunoDTO) para transferência de dados.
- **Services:** Camada que orquestra chamadas à API, validações e lógica de negócio no frontend.

### 5.1 Usabilidade

- **Interface:** A interface é responsiva, construída com MudBlazor, oferecendo componentes como tabelas interativas, formulários dinâmicos e modais. A navegação é intuitiva, com menus laterais e breadcrumbs.
- **Dashboards:** Gráficos gerados com Blazorise.Charts exibem métricas como taxas de aprovação, frequência por disciplina e desempenho médio por curso.
- **Acessibilidade:** MudBlazor segue padrões WCAG, garantindo suporte a leitores de tela e navegação por teclado.
- **Feedback ao Usuário:** Notificações visuais (toasts) informam sobre sucesso ou erro em operações.

## 5.2 Manutenibilidade

- **Modularidade:** O código é dividido em camadas (API, Web, Data, etc.), com responsabilidades claras.
- **Padrões:** Uso de DTOs para transferência de dados, injeção de dependência para serviços e repositórios genéricos no EF Core.
- **Evolução do Banco:** Migrações do EF Core permitem atualizações no schema do banco sem perda de dados.
- **Documentação:** Swagger/OpenAPI integrado às Minimal APIs facilita a manutenção dos endpoints.
- **Testabilidade:** A estrutura permite testes unitários (ex.: com xUnit) e de integração.

## 5.3 Segurança

- **Autenticação e Autorização:** Sugerida a implementação de JWT ou integração com IdentityServer para proteger endpoints e restringir acesso por roles (ex.: Admin, Professor, Aluno).
- **Validação de Entrada:** Minimal APIs utilizam validação de modelos com anotações [Required], [Range], etc.
- **Proteção contra Ataques:**
  - **SQL Injection:** Evitado pelo uso de consultas parametrizadas no EF Core.
  - **XSS:** Componentes MudBlazor sanitizam entradas de usuários.
  - **CSRF:** Blazor WebAssembly mitiga CSRF por não depender de cookies padrão.
- **Auditoria:** Sugerida a implementação de logs para rastrear operações sensíveis (ex.: alteração de notas).

## 5.4 Desempenho

- **Frontend:** Blazor WebAssembly reduz requisições ao servidor após o carregamento inicial, garantindo interações rápidas.
- **Backend:** Minimal APIs são leves, com sobrecarga mínima comparada a controllers tradicionais.
- **Banco de Dados:** EF Core utiliza carregamento sob demanda (Include, AsNoTracking) e índices para otimizar consultas.



- **Paginação:** Tabelas com grandes volumes de dados (ex.: lista de alunos) utilizam MudPagination no frontend e queries paginadas no backend.
- **Cache:** Sugerida a implementação de cache em memória ou distribuído (ex.: Redis) para consultas frequentes.

## 5.5 Portabilidade

- **Frontend:** Blazor WebAssembly é compatível com navegadores modernos (Chrome, Firefox, Edge, Safari).
- **Backend:** .NET 8 é multiplataforma, permitindo execução em Windows, Linux ou macOS.
- **Banco de Dados:** SQL Server é o banco padrão, mas o EF Core suporta outros bancos (ex.: PostgreSQL) com ajustes na connection string.
- **Deploy:** A aplicação pode ser hospedada em serviços como Azure, AWS ou servidores locais, com suporte a contêineres (Docker).

## 6 O Sistema Interativo

O Sistema de Gestão Acadêmica foi projetado considerando o contexto de uma instituição Acadêmica educacional. As funcionalidades foram adaptadas para atender às necessidades, como:

- **Gestão de Frequência:** Registro detalhado de presença, essencial para instituições que monitoram assiduidade rigorosamente.
- **Interface Personalizada:** A interface utiliza cores e ícones que remetem à identidade visual da instituição, com suporte a temas claros e escuros via MudBlazor.
- **Integrações Futuras:** O sistema está preparado para integração com APIs de serviços locais, como sistemas de pagamento de mensalidades ou plataformas de ensino a distância.

O sistema suporta cenários típicos de instituições cearenses, como cursos técnicos, graduações e programas de extensão, com flexibilidade para personalização.

## 7 API do Software

A API do sistema foi desenvolvida com **Minimal APIs** no .NET 8, oferecendo endpoints REST para todas as funcionalidades principais. A API é organizada por recurso, com suporte a operações CRUD, filtros, ordenação e paginação. Exemplos de endpoints:

- **/api/alunos:**
  - GET /api/alunos: Lista alunos com filtros (nome, matricula) e paginação.
  - GET /api/alunos/{id}: Retorna detalhes de um aluno.
  - POST /api/alunos: Cria um novo aluno.
  - PUT /api/alunos/{id}: Atualiza dados de um aluno.
  - DELETE /api/alunos/{id}: Exclui um aluno.
- **/api/cursos:**
  - GET /api/cursos: Lista cursos com filtros (nome, duracao).
  - POST /api/cursos: Cria um novo curso.
- **/api/notas:**
  - GET /api/notas?alunoId={id}&disciplinaId={id}: Consulta notas por aluno e disciplina.
  - POST /api/notas: Lança uma nova nota.
- **/api/frequencia:**
  - POST /api/frequencia: Registra presença de um aluno em uma disciplina.
  - GET /api/frequencia?data={data}: Lista frequências por data.
- **Características:**
  - **Formato de Dados:** JSON para requisições e respostas.
  - **Paginação:** Suporte via parâmetros page e pageSize.
  - **Documentação:** Swagger/OpenAPI integrado para exploração dos endpoints.
  - **Autenticação:** Sugerida a integração com JWT para proteger endpoints sensíveis.
  - **Validação:** Modelos validados com anotações e FluentValidation (opcional).

A API é consumida pelo frontend Blazor via serviços injetados, utilizando HttpClient com configuração centralizada.

## **8 Conclusão**

O Sistema de Gestão Acadêmica, em sua versão 1.0, entrega uma solução completa e moderna para administração acadêmica, combinando uma interface web intuitiva, dashboards analíticos e uma API robusta.

A arquitetura em camadas, aliada às tecnologias .NET 8, Blazor WebAssembly e MudBlazor, garante usabilidade, desempenho e potencial de escalabilidade. O sistema atende às necessidades de instituições Educacionais com funcionalidades adaptadas.