Estrutura da Solution

|  |  |
| --- | --- |
| SchoolManagement.API/  ├── Controllers/  │ ├── EscolaController.cs  │ ├── MensalidadeController.cs  │ ├── ParcelaController.cs  │ └── AuthController.cs  ├── Middleware/  │ └── ExceptionHandlingMiddleware.cs  ├── Program.cs  ├── appsettings.json |  |
| SchoolManagement.Application/  ├── Interfaces/  │ ├── IEscolaService.cs  │ ├── IParcelaService.cs  ├── Services/  │ ├── EscolaService.cs  │ ├── ParcelaService.cs  ├── DTOs/  │ ├── EscolaDto.cs  │ ├── ParcelaDto.cs  │ └── LoginDto.cs |  |
| SchoolManagement.Domain/  ├── Entities/  │ ├── Escola.cs  │ ├── Mensalidade.cs  │ ├── Aluno.cs  │ └── Parcela.cs  ├── Enums/  │ └── StatusPagamento.cs  ├── Interfaces/  │ ├── IEscolaRepository.cs  │ ├── IParcelaRepository.cs  ├── Rules/  │ └── ParcelaRules.cs |  |
| SchoolManagement.Infrastructure/  ├── Data/  │ ├── AppDbContext.cs  │ └── SeedData.cs  ├── Repositories/  │ ├── EscolaRepository.cs  │ ├── ParcelaRepository.cs  ├── Migrations/ |  |
| SchoolManagement.Tests.Unit/  ├── Services/  │ ├── EscolaServiceTests.cs  │ └── ParcelaServiceTests.cs  ├── Rules/  │ └── ParcelaRulesTests.cs |  |
| SchoolManagement.Tests.Integration/  ├── API/  │ ├── EscolaControllerTests.cs  │ └── ParcelaControllerTests.cs  ├── Setup/  │ └── TestServerFactory.cs |  |
| SchoolManagement.Factory/  ├── ParcelaFactory.cs |  |
| SchoolManagement.WebApp/  ├── pages/  │ ├── login.tsx  │ ├── admin.tsx  │ └── responsavel.tsx  ├── components/  │ ├── EscolaList.tsx  │ ├── MensalidadeList.tsx  │ ├── ParcelaList.tsx  ├── services/  │ └── api.ts  ├── context/  │ └── AuthContext.tsx |  |

Responsabilidades

|  |
| --- |
| SchoolManagement.API   * Expor endpoints REST * Configurar injeção de dependência * Middleware para tratamento de erros * Autenticação local |
| SchoolManagement.Application   * Orquestrar regras de negócio * Transformar entidades em DTOs * Validar dados de entrada * Gerenciar autenticação e perfi |
| SchoolManagement.Domain   * Modelar entidades e relacionamentos * Definir interfaces de repositórios * Implementar regras de negócio (ex: cálculo de juros) |
| SchoolManagement.Infrastructure   * Configurar EF Core e SQL Server * Implementar repositórios * Gerenciar migrations e seed |
| SchoolManagement.Tests.Unit   * Testar regras de negócio e serviços isoladamente * Usar NUnit + Moq para mocks |
| SchoolManagement.Tests.Integration   * Testar endpoints com banco real ou em memória * Validar fluxo completo de pagamento |
| SchoolManagement.Factory   * Criar parcelas com lógica de vencimento * Calcular valor pago com juros |
| SchoolManagement.WebApp   * Tela de login com seleção de perfil * Página de administrador: lista escolas e mensalidades * Página de responsável: lista parcelas e botão "Marcar como paga" * Mensagens de erro/sucesso visíveis |

🔌 Backend (.NET + EF Core)

📁 Entidades principais

* Escola: Id, Nome
* Mensalidade: Id, EscolaId, Valor, DataVencimento
* Aluno: Id, Nome, EscolaId
* Parcela: Id, AlunoId, MensalidadeId, Valor, DataVencimento, Status, ValorPago

🔄 Endpoints REST

|  |  |  |
| --- | --- | --- |
| /api/escolas |  |  |
| /api/escolas/{id}/mensalidades |  |  |
| /api/alunos/{id}/parcelas |  |  |
| /api/parcelas/{id}/pagar |  |  |

📐 Regras de negócio

* Status:
  + "pago": ValorPago > 0
  + "não pago": ValorPago == 0 e DataVencimento >= hoje
  + "atraso": ValorPago == 0 e DataVencimento < hoje
* Juros: valorJuros = diasAtraso \* 0.01 \* valorInicial
* ValorPago = valorInicial + valorJuros

🧪 Testes

* NUnit para testes unitários (serviços, regras de negócio)
* Testes de integração com banco local e chamadas HTTP

🧰 Técnicas e padrões

* Injeção de dependência com Microsoft.Extensions.DependencyInjection
* Repositórios com interfaces no domínio
* Factory para criação de parcelas e cálculo de pagamento
* Tratamento de erros com middleware customizado e mensagens amigáveis

🖥️ Frontend (React + Next.js)

🔐 Autenticação

* Tela de login simples com seleção de perfil (Administrador ou Responsável)
* Autenticação local com email/senha
* Redirecionamento por perfil

🧭 Páginas

|  |  |
| --- | --- |
|  |  |
|  |  |

✅ UX

* Mensagens de sucesso/erro visíveis após cada ação
* Feedback visual para carregamento e validação

🔧 Tecnologias

* Next.js com Pages Router
* Fetch API para chamadas aos endpoints
* Context API ou Redux para estado de autenticação
* TailwindCSS ou styled-components para estilo

📌 Extras

* CI/CD com GitHub Actions ou Azure DevOps
* Docker para ambiente local
* Swagger para documentação da API