MBA DevXpert Full Stack .NET

Plataforma Educacional Distribuída com Microsserviços REST

Título do Projeto

Plataforma Educacional Distribuída com Microsserviços REST

Objetivo

Desenvolver uma evolução do projeto do Módulo 3, transformando a aplicação monolítica em um conjunto de APIs independentes, distribuídas por contexto de negócio (Bounded Context), utilizando boas práticas de arquitetura de microsserviços, comunicação resiliente entre serviços, mensageria e segurança baseada em JWT. O projeto deve simular um cenário mais realista de um ecossistema distribuído, com integração entre serviços via HTTP e eventos.

Descrição Geral

A plataforma educacional agora é composta por diversas APIs independentes que se comunicam entre si para executar os fluxos de negócio. Cada API representa um Bounded Context (BC) com banco de dados isolado e responsabilidade clara.

Os principais componentes do sistema são:

- **APIs RESTful**: Uma para cada contexto de negócio.
- **Mensageria**: Para comunicação assíncrona entre serviços.
- **Autenticação centralizada**: API Auth que emite JWTs. As demais APIs apenas validam os tokens.
- **Front-end**: Pode consumir as APIs diretamente ou via um BFF (Backend for Frontend) que orquestra as chamadas.

Bounded Contexts e APIs

Cada BC se torna uma API isolada com seu próprio repositório, banco de dados e escopo.

Auth API

- Cadastro de usuários (Aluno/Admin)
- Login e emissão de JWT
- (Opcional) refresh token

Conteúdo API

- Cadastro e edição de cursos
- Cadastro de aulas
- Consulta de conteúdo programático

Alunos API

- Matrícula em cursos
- Progresso de aulas
- Finalização de cursos e geração de certificados
- Histórico do aluno

Pagamentos API

- Processamento de pagamento
- Consulta de status
- Emissão de eventos de pagamento confirmado ou rejeitado

BFF (Backend for Frontend)

- Centraliza as chamadas para o front-end
- Orquestra os fluxos complexos entre serviços evitando que o front-end seja obrigado a orquestrar a chamada de N APIs.

Modelo de Autenticação e Usuários

- Apenas a **Auth API** possui acesso ao banco de dados de usuários.
- JWT é assinado e validado nas demais APIs.
- Claims no JWT definem o tipo de usuário (Aluno/Admin) e seu ID.
- O front-end deve autenticar uma vez e utilizar o token para as demais requisições.
- Tipos de usuário:
    - **Administrador:** Controle total para cadastrar cursos, aulas, gerir assinaturas, pagamentos, alunos.
    - **Aluno:** Acesso restrito para matrícula em cursos, visualização de aulas/conteúdos e gerenciamento das suas próprias informações e pagamentos.

Quem Executa as Ações

- **Aluno autenticado:** Realiza suas próprias matrículas, consulta cursos/aulas disponíveis, realiza pagamentos e acessa materiais das aulas.
- **Administrador autenticado:** Realiza operações administrativas (criação e manutenção dos cursos, gestão financeira e monitoramento dos alunos).

Casos de Uso Detalhados

Login e Emissão de JWT

- Ator: Aluno ou Administrador
- Fluxo:
    1. Informa e-mail e senha
    2. API Auth valida credenciais e retorna JWT

Cadastro de Curso (Conteúdo API)

- Ator: Administrador
- Fluxo:
    1. Cadastra curso com nome e conteúdo
    2. Curso salvo com ID e disponível para alunos

Matrícula e Pagamento (Fluxo entre APIs)

- Ator: Aluno
- Fluxo:
    1. Aluno seleciona curso (Alunos API)
    2. Matrícula criada com status "pendente"
    3. Aluno realiza pagamento (Pagamentos API)
    4. Evento `PagamentoConfirmado` publicado
    5. Alunos API consome evento e ativa matrícula

Realização de Aula

- Ator: Aluno
- Fluxo:
    1. Aluno acessa aula (Conteúdo API)
    2. Progresso registrado na Alunos API via chamada HTTP

Finalização de Curso

- Ator: Aluno
- Fluxo:
    1. Aluno solicita finalização (Alunos API)
    2. API valida progresso
    3. Certificado gerado e disponibilizado

Requisitos Técnicos

- Linguagem: C# com .NET 8 ou superior
- Arquitetura: Microsserviços REST
- Autenticação: JWT via ASP.NET Core Identity na Auth API
- Banco: SQL Server por API + SQLite com Seed
- Mensageria: RabbitMQ (preferencial)
- Resiliência: Retry, Circuit Breaker (Polly)
- Documentação: Swagger em cada API

Critérios de Sucesso

- Cada API funciona de forma independente, com seu próprio BD e sem acessar outras APIs
- JWT validado corretamente entre serviços
- Fluxos de matrícula, pagamento e certificado funcionam com eventos
- Prova de resiliência: falha no pagamento não compromete sistema
- Readme completo com instruções para execução local
- Configuração (**obrigatório**):
    - O projeto deve rodar com a menor configuração de infra possível, para isso utilize a prática ensinada no vídeo a seguir: https://desenvolvedor.io/plus/criando-e-populando-automaticamente-qualquer-banco-de-dados

Prazos e Tipo de Entrega

- **Tipo de Entrega:** Projeto para desenvolvimento **em GRUPO**
- Início: 28/01/2026
- Definição dos Grupos: 02/02/2026
- Primeira Entrega: 02/03/2026
- Entrega Final: 06/04/2026

Instruções Importantes

Este documento descreve os principais requisitos e funcionalidades esperadas no projeto, oferecendo uma visão geral do domínio e dos objetivos a serem atingidos.

Nem todos os detalhes técnicos e funcionais foram explicitamente descritos.

Espera-se que você compreenda profundamente o problema apresentado e utilize o conhecimento adquirido ao longo dos cursos para implementar as soluções adequadas, tomando decisões técnicas alinhadas com boas práticas.

Os casos de uso especificados devem ser implementados e funcionar conforme descrito, mas você tem autonomia para adotar abordagens técnicas e estratégias adicionais, desde que mantenha o projeto alinhado aos requisitos e objetivos apresentados neste escopo.

Entrega

Repositório no GitHub:

- O código deve ser versionado e entregue através de um repositório público no Github e dentro dos padrões especificados em: https://github.com/desenvolvedor-io/template-repositorio-mba

Documentação:

- O README.md deve seguir as diretivas e padrões informados na documentação do projeto referência.

- Incluir um arquivo FEEDBACK.md no repositório onde os feedbacks serão consolidados, o instrutor fará um PR no repositório atualizando este arquivo.

Matriz de avaliação

Os projetos serão avaliados e receberão uma nota de 0 até 10 considerando os critérios a seguir:

Critério	Peso	Comentários
Funcionalidade	30%	Avalie se o projeto atende a todos os requisitos funcionais definidos.
Qualidade do Código	30%	Considere clareza, organização, uso de padrões de codificação.
Eficiência e Desempenho	10%	Avalie o desempenho e a eficiência das soluções implementadas.
Inovação e Diferenciais	10%	Considere a criatividade e inovação na solução proposta.
Documentação	10%	Verifique a qualidade e completude da documentação, incluindo README.md.
Resolução de Feedbacks	10%	Considere como o aluno ou grupo abordou os feedbacks da revisão de código.

PS – A nota final será atribuída ao grupo como um todo. Entretanto, a participação individual de cada aluno será observada e considerada. Alunos que não demonstrarem participação evidente poderão sofrer penalização proporcional na nota aplicada.