Parte 1 - Arquitetura: App, Data, Domínio

Status	Lendo - In progress

Vamos entender o papel de cada camada e como proceder para estruturar seu projeto corretamente:

Estrutura Geral:

- **App**: A aplicação principal (neste caso, um console app).
- **Data**: Responsável pela interação com a base de dados (biblioteca de classes).
- **Domínio**: Contém a lógica central do negócio, as regras e entidades (biblioteca de classes).

Parte 1: Entendimento geral

1. Camada "Domínio" (Regras de Negócio)

A **camada de domínio** é o coração da sua aplicação. Aqui você define as entidades e as regras de negócio, e ela não deve depender das outras camadas (como infraestrutura ou apresentação). Esta camada deve ser focada em conceitos e comportamentos que são próprios do seu negócio.

O que incluir na camada de Domínio:

Entidades: Modelos que representam os principais objetos do sistema. Por exemplo, em um sistema de e-commerce, você poderia ter entidades como `Produto`, `Pedido`, `Cliente`.

Objetos de valor: Pequenos tipos que têm significado, como `Dinheiro` (um valor monetário) ou `CPF` (um documento).

Serviços de domínio: Para operações que envolvem mais de uma entidade ou regras complexas. Por exemplo, um serviço que calcula o preço final de um pedido com base em várias regras.

Repositórios (Interfaces): Interfaces que definem métodos para acessar os dados (como buscar, adicionar, atualizar uma entidade), sem implementar como isso é feito. A implementação real será feita na camada Data.

---

2. Camada "Data" (Acesso a Dados)

A **camada de dados** (Data) é onde você implementa o acesso ao banco de dados ou qualquer outro serviço de armazenamento. Esta camada vai conter as implementações dos **repositórios** definidos na camada de domínio, mas agora com a lógica de como os dados são persistidos ou consultados (ex.: usando Entity Framework).

O que incluir na camada de Data:

Repositórios (implementações): Aqui você implementa os repositórios que definiram as operações de acesso a dados no domínio.

Contexto de dados: Se você estiver usando ORM (Object Relational Mapper) como o Entity Framework, o contexto de dados (`DbContext`) ficará aqui.

---

3. Camada "App" (Apresentação)

Essa camada é responsável pela interface com o usuário, e no seu caso, como você está criando uma aplicação console, ela vai lidar com a entrada e saída de dados para o console. Ela também vai **orquestrar** as chamadas entre as outras camadas, como acessar o repositório para buscar ou salvar dados e executar regras de negócio.

O que incluir na camada de App:

- **Ponto de entrada da aplicação**: O método `Main` que inicia a execução.

- **Serviços de aplicação**: Controla o fluxo de dados entre a camada de apresentação e a camada de domínio.
- **Injeção de dependências**: Para conectar as interfaces do domínio com suas implementações na camada de dados, usando um contêiner de injeção de dependência (como `Microsoft.Extensions.DependencyInjection`).

Como conectar tudo?

Referências de projeto: No Visual Studio, você vai precisar adicionar referências para que a **camada App** conheça as camadas **Data** e **Domínio**. A camada **Domínio** deve ser independente da camada **Data**, e apenas a **App** e **Data** devem referenciá-la.

- A camada **App** vai depender das camadas **Domínio** e **Data**.
- A camada **Data** depende apenas da **Domínio**.

Boas práticas

- **Injeção de Dependência**: Evite criar as instâncias diretamente. Em vez disso, use injeção de dependências para resolver as dependências da aplicação.
- **Separação de responsabilidades**: Cada camada tem sua responsabilidade específica, o que facilita a manutenção e testes.
- **Testes**: Como a camada de domínio está desacoplada das outras camadas, é fácil criar testes unitários para verificar a lógica de negócio.

Recapitulando

O que é cada camada?

1. **Aplicação**: É a parte do projeto que interage com o usuário ou sistema externo. No seu caso, é o programa principal, a "cara" do seu sistema. Aqui, você faz coisas como ler entradas e mostrar mensagens de saída.

2. **Domínio**: É o **coração** da sua aplicação, onde ficam as regras do negócio. Toda a lógica que faz seu sistema funcionar corretamente, como validações e cálculos, está aqui. É onde você define os comportamentos do seu sistema.
3. **Dados**: Lida com o armazenamento e recuperação de dados, como salvar e buscar informações de um banco de dados. No seu exemplo, podemos deixar essa camada de lado por enquanto, já que você está mais focado nas outras duas.
