**Solution IEscola**

Solutions Folder

1 – API

Ponto de entrada e saída da nossa aplicação.

2- Application

Serviços -> Chamadas a outros domínios, validação, Orquestrar o funcionamento da App

3- Domain

Domínio – Core, Classes de negócio, Entidades, Repository, independente das outras aplicações

4- Infra

Data: Fonte de dados

5 – CrossCutting

Uma dll de referência para os outros projetos

**Orientação a objetos e tipos de dados**

Classe -> Elemento que possui atributos/propriedades, métodos

Nível de acesso: internal, private, protected, public

Encapsulamento:

Classe abstrata: Classe modelo que não pode ser instanciada(new). Apenas herdada.

Classe estática: Classe que não pode ser instanciada e pode ser utilizada diretamente(apenas escrevendo o nome da classe).

Interface: É um contrato que define as ações que uma classe que a implementa deve fazer(É obrigatório fazer o que está definido na interface).

Conceitos – **SOLID**

**S – Single Responsability Principle (Princípio de Responsabilidade Única)**

R: Princípio que define uma única responsabilidade para um objeto(shiita)/método(cara relax)

**O – Open/Closed Principle(Aberto ou fechado)**

R: Um objeto/classe deve estar fechado para alteração e aberto a extensão.

**L – Liskov**

R: Programar utilizando subtipos(com herança)

**I – Interfaces**

R: Você deve programar orientado a Interfaces e não a instancias concretas da classe(derivação, contrato, implementação)

**D – Dependency Injection(IoC – Inversion of Control)**

R: Aplicar a injeção de dependência via contratos. Sua classe que depende no fluxo de uma outra classe deve depender de sua abstração/contrato/interface.

**LifeTime dos objetos na memória**

// services.AddSingleton -> Instancia única na memória()

// services.AddScoped -> Instancia única na memória() durante a requisição

// services.AddTransient -> Uma instancia nova por chamada(E não request)

// Alerta Singleton não pode ter dependencia para Scoped