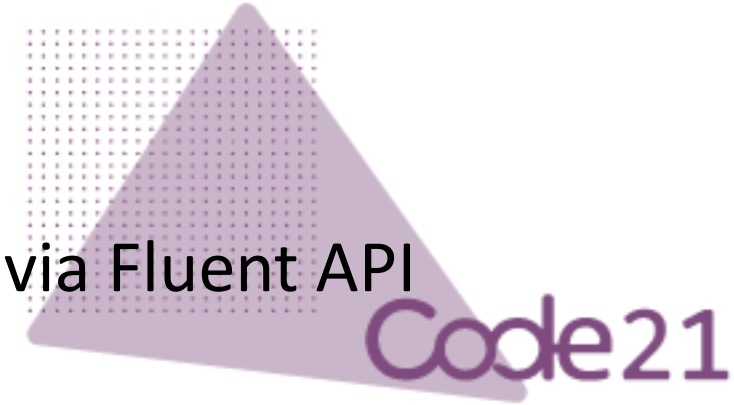
Fluent API

- Mepamento e configuração do modelo possibilitando a alteração das convenções
- Alternativa aos Data Annotations para permitir entidades “limpas”
- A estrutura de configuração pode ficar em outra classe ou projeto
- Organização dos mapeamentos e configuração
- Para algumas configurações só é possível realizar via Fluent API



Fluent API

Permite a configuração de 3 grupos da estrutura do projeto:

- **Model Configuration(*Database*):** Configuração do schema default, funções de banco, padrões de exclusão, etc...
- **Entity Configuration(*Tabelas*):** Configuração das entidades, relaciamentos, chaves primárias, estrangeiras, índices, etc...
- **Property Configuration(*Colunas*):** Configuração dos tipos das propriedades, padrões, validações, limites e tipos nulos

Fluent API

```
public DbSet<Student> Students { get; set; }
public DbSet<Standard> Standards { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    //Configure Column
    modelBuilder.Entity<Student>()
                .Property(p => p.DateOfBirth)
                .HasColumnName("DoB")
                .HasColumnOrder(3)
                .HasColumnType("datetime2");
}
```

Atualizações de Estrutura

- As alterações de estrutura em um ORM precisam ser sincronizadas
- Depende do fluxo escolhido dentro do time para alterações
- Permite a alteração totalmente manual
- Sem interferência ou ação da framework na estrutura
- Porém é possível utilizar o recurso do Migrations
- A plataforma fica então responsável também por controlar a alteração
- Cria mais independência com o banco
- Permite inclusive o recurso de desfazer alterações

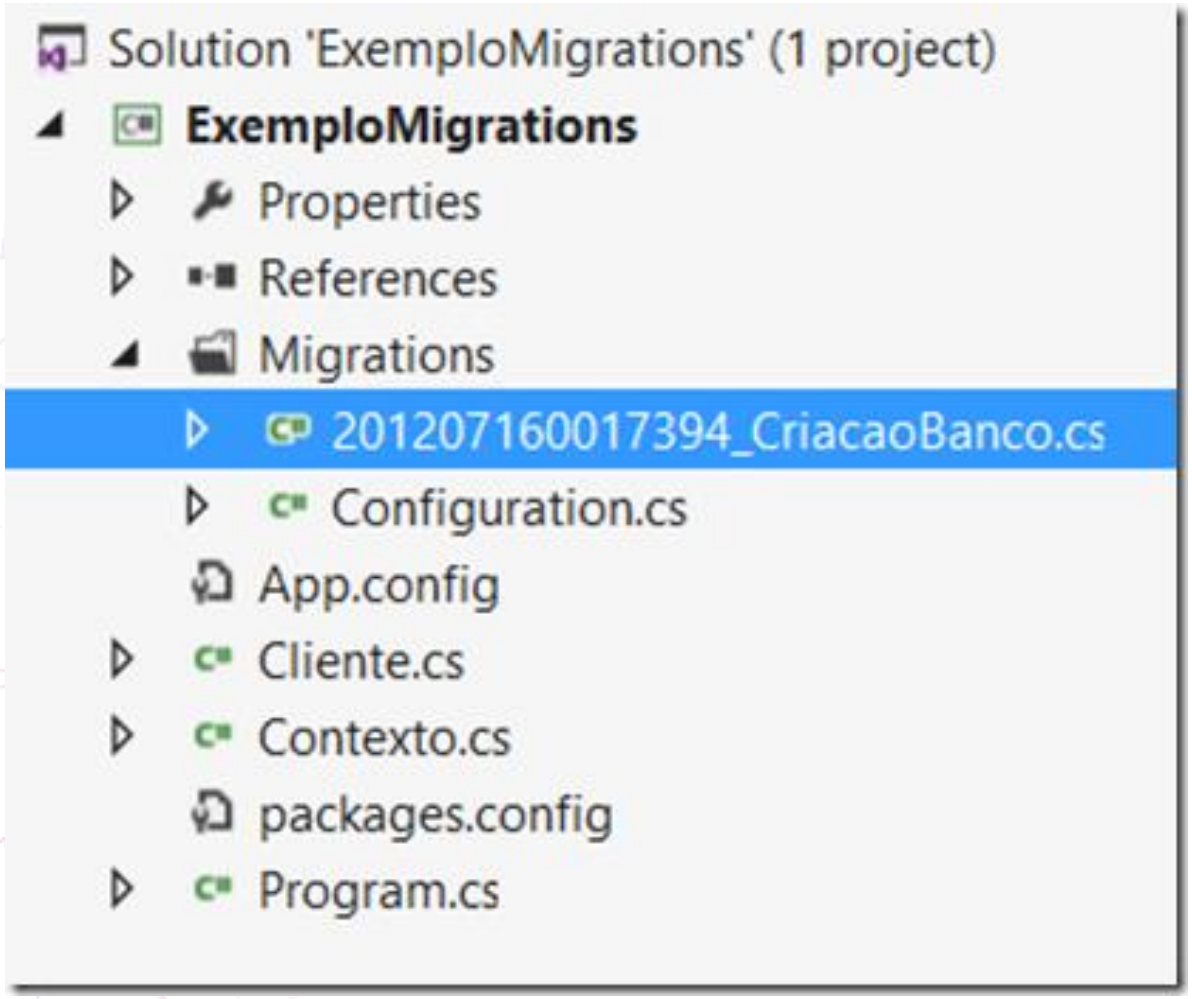
Trabalhando com Migrations

- É uma extenção do Entity Framework ***(Microsoft.EntityFrameworkCore.Tools)***
- Gestão e automatização das atualizações
- O Migrations permite tanto atualizar como reverter alterações
- Utilizando os inicializadores do contexto é possível automatizar totalmente a sincronização de estrutura
- Neste modelo a estrutura de base nunca deve ser alterada
- As alterações serão sempre gerenciadas pelo *Migrations*

Trabalhando com Migrations

- Cada atualização que será sincronizada pode possuir um nome
- Ao gerar uma atualização é possível gerar um script para revisão
- Ou atualizar diretamente a base de destino
- Os comandos que devem ser utilizados são:
  - *Add-Migration "nome"* – gera a alteração e armazena no projeto
  - *Update-Database* – aplica a alteração gerada com o banco
  - *Update-Database script* – gera script e não aplica as alterações
  - *Update-Database "nome"* – reverte a estrutura para o pacote de alterações com o "nome"

Trabalhando com Migrations



Code21

Exercício 17

Trabalhando com Migrations