codehouse
{ACADEMY}

codehouse
{ACADEMY}

Accesos a Datos

codehouse {ACADEMY}

2. ORM .Net

2.1. Dapper

- ✓ Micro ORM que se encarga del manejo de acceso a datos
- ✓ Ejecución de consultas, inserciones, actualizaciones y borrados
- ✓ A destacar:
  - ➢ Consultas parametrizadas
  - ➢ Automapeo
  - ➢ Bulkinsert
  - ➢ Ejecución de múltiples consultas a la vez

codehouse
{ACADEMY}

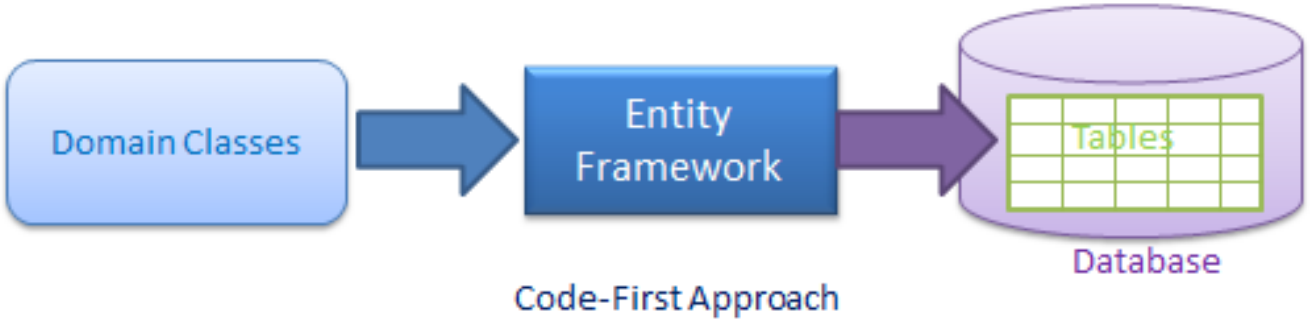
2.2. Entity FrameWork Core

- ✓ ORM creado para el ecosistema .Net
- ✓ Es extensible, ligero, cross-platform y open-source
- ✓ Permite el acceso a datos a través de modelos: Clase que define la entidad y un contexto que representa la conexión a la base de datos

codehouse
{ACADEMY}

2.2.1. Entity FrameWork Core Code First

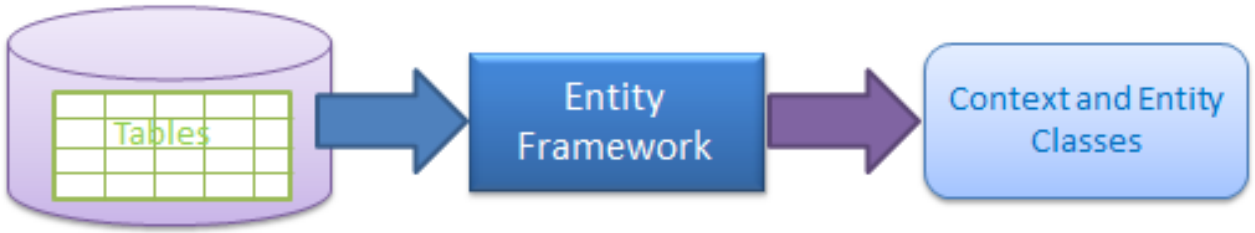
- ✓ El modelo se define en entidades en nuestra aplicación
- ✓ Muy útil para DDD



codehouse
{ACADEMY}

2.2.2. Entity FrameWork Core Database First

- ✓ El modelo se define en la base de datos, por lo que hacemos uso de un modelo ya existente



Database-First Approach

codehouse
{ACADEMY}

2.2.3. Contexto

- ✓ La clase mas importante de Entity Framework
- ✓ Representa la conexión a la base de datos
- ✓ Con ella podemos hacer cualquier operación CRUD
- ✓ En nuestro proyecto debemos crear una clase que herede de DbContext
- ✓ Nos permite modelar nuestras entidades y la relación entre ellas

codehouse {ACADEMY}

2.2.4. Migraciones

- ✓ Es el método para crear o actualizar las bases de datos desde EF
- ✓ Podemos agrupar cambios en el modelo en migraciones
- ✓ Con las migraciones podemos:
  - ➢ Cambiar entidades
  - ➢ Actualizar relaciones
  - ➢ Introducir datos en tablas maestras
  - ➢ Introducir datos en otras tablas ya conectadas
- ✓ Se pueden revertir y por ello tenemos métodos de Up y Down

codehouse
{ACADEMY}

2.2.5. Creando Entidades

- ✓ Uso de DataAnnotation, que permite “marcar” las propiedades con ciertas características
  - [Key]
  - [Required]
  - [ForeignKey(“Nombre de la Propiedad”)]
  - [MaxLength]

codehouse {ACADEMY}

2.2.6. FluentAPI

- ✓ Usando la clave DbModelBuilder podemos configurar nuestras entidades, mas allá de los atributos de DataAnnotation
- ✓ No es excluyente y podemos usar juntos FluentAPI y DataAnnotation
  - ✓ HasKey()
  - ✓ Property()
  - ✓ IsRequired()
  - ✓ HasForeignKey()

codehouse
{ACADEMY}

2.2.7. Buenas Prácticas

- ✓ IQueryable. Montamos las consultas y después hacemos filtros y paginamos y ordenamos, pero antes de obtener los datos.
- ✓ Eager loading. Permite traer los datos relacionados, pero solo cuando es necesario
- ✓ Lazy loading. Método opuesto al anterior. Vamos a traer siempre los datos relacionados. Se puede activar y desactivar.

codehouse
{ACADEMY}