

**Finalidad:**

Prestación del servicio Público de educación superior (art. 1 LOU)

**Responsable:**

Universitat Politècnica de València.

**Derechos de acceso, rectificación, supresión, portabilidad, limitación u oposición al tratamiento conforme a políticas de privacidad:**

<http://www.upv.es/contenidos/DPD/>

**Propiedad intelectual:**

Uso exclusivo en el entorno de aula virtual.

Queda prohibida la difusión, distribución o divulgación de la grabación de las clases y particularmente su compartición en redes sociales o servicios dedicados a compartir apuntes.

La infracción de esta prohibición puede generar responsabilidad disciplinaria, administrativa o civil



# ENTITY FRAMEWORK

---

## Seminario 4 – Desarrollo de Software en Visual Studio 2019

# Objetivos

- Conocer el modelo de persistencia denominado **Entity Framework (EF)**
- Aprender a aplicar el enfoque de desarrollo para EF denominado **Code-First**
- Desarrollar una aplicación de ejemplo basada en el enfoque Code-First en Entity Framework

# Contenidos

1. Introducción
2. DbContext
3. Convenciones Code-First
4. Anotaciones de Datos
5. Inicialización BD
6. Operaciones BD
7. Estrategias de Carga
8. Conclusiones

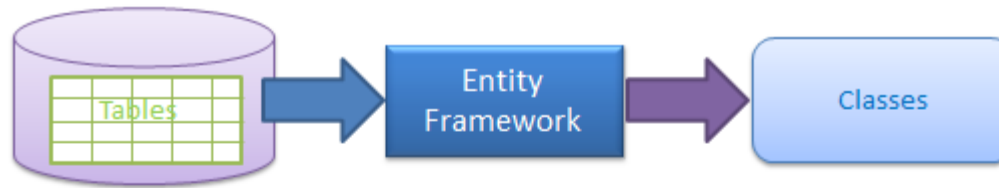
# INTRODUCCIÓN

---

# Introducción

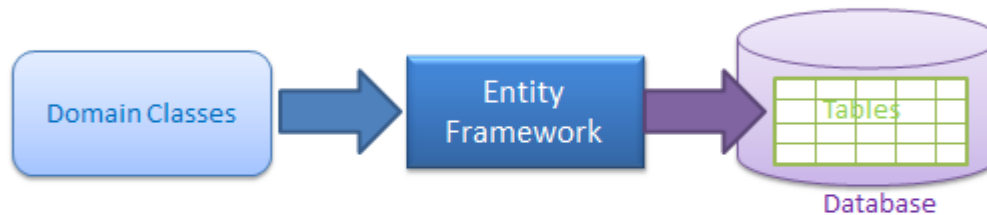
- EF es un framework para Mapeo Objeto-Relacional (Object/Relational Mapping-O/RM)
  - Mantiene el diseño de la base de datos separado del diseño de las clases del dominio.
  - Automatiza operaciones CRUD estándar (Create, Read, Update & Delete) de modo que el desarrollador no necesita escribirlas manualmente.
- EF soporta tres enfoques de desarrollo:
  - **Database-first:** se dispone previamente de la base de datos o se quiere diseñar la base de datos antes que el resto
  - **Code-first:** se crean primero las clases del dominio y después la base de datos adecuada para esas clases.
  - **Model-first:** se quiere diseñar el esquema de base de datos (diseñador visual) y después crear la base de datos y las clases.

# Introducción



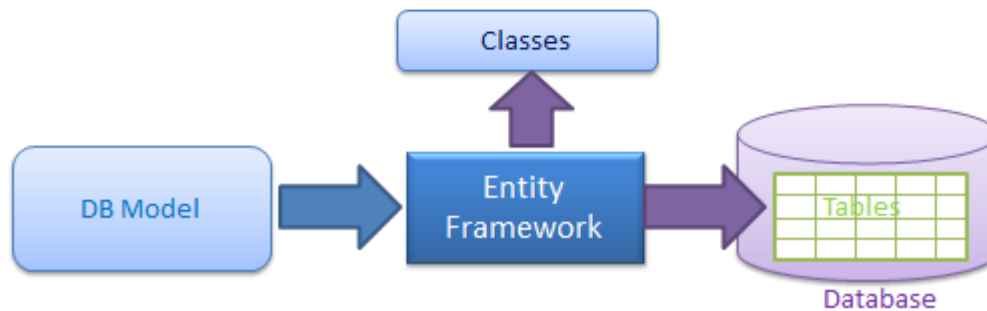
Generate Data Access Classes for Existing Database

**Database-First**



Create Database from the Domain Classes

**Code-First**



Create Database and Classes from the DB Model design

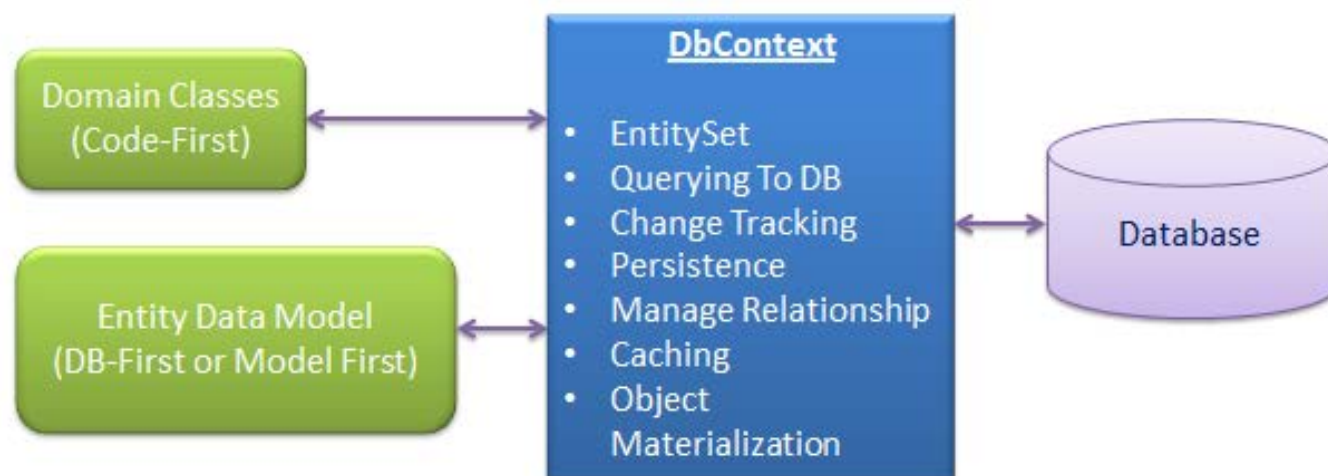
**Model-First**

# DBCONTEXT

---

# La Clase DbContext

- Define el mapeo de objetos del dominio a tablas en la base de datos siguiendo el patrón repositorio.
- Ofrece un mecanismo de acceso en memoria a los objetos persistidos. Cada clase a persistir se representa mediante una colección de tipo **ICollection<TEntity>**

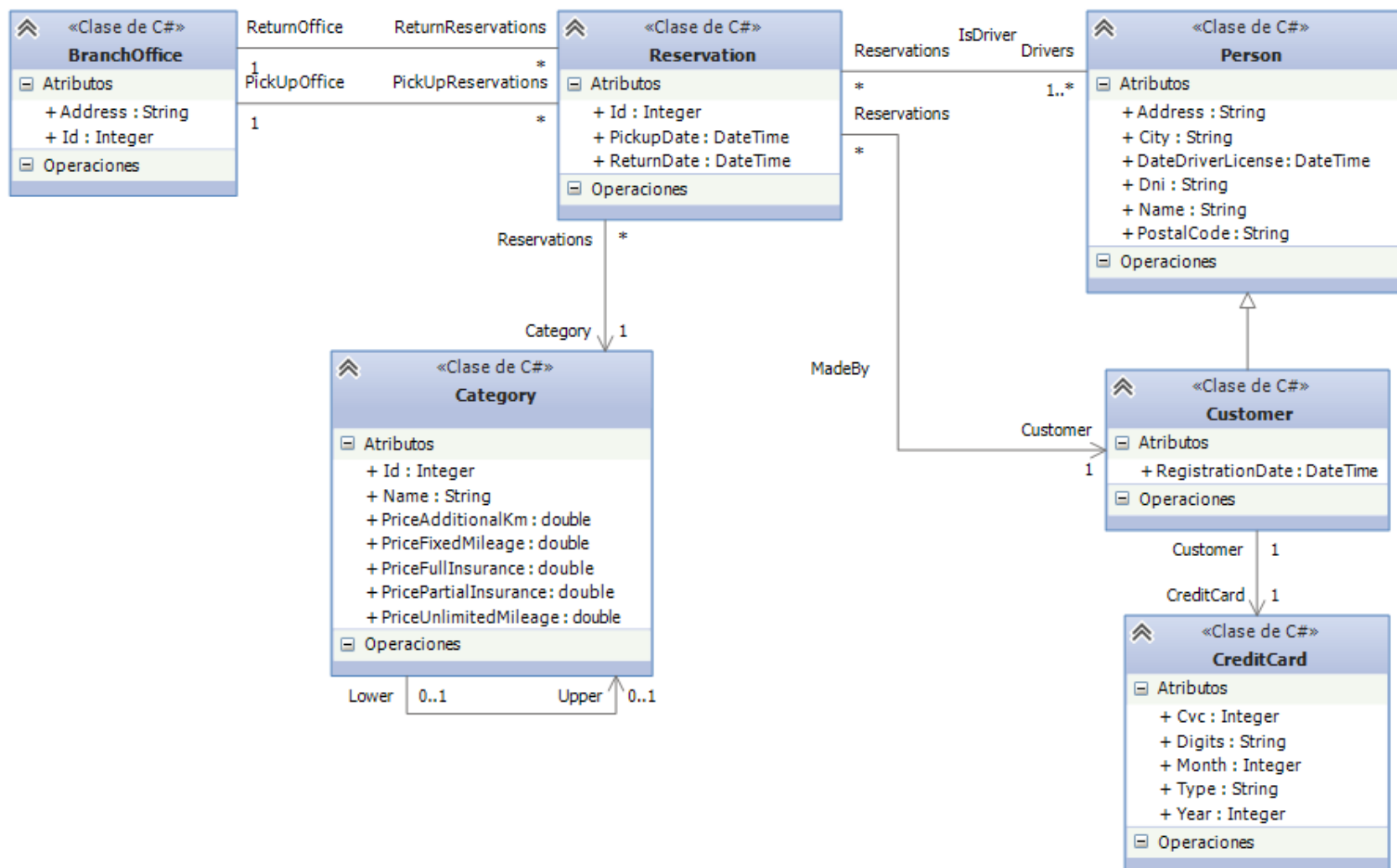




# Funcionalidad de DbContext

- **EntitySet:** contiene conjuntos entidad para todas las entidades que se mapean a tablas en la base de datos (IDbSet<TEntity>).
- **Consultas:** convierte consultas LINQ-to-Entities a consultas SQL y las envía a la base de datos.
- **Control de Cambios:** mantiene constancia de los cambios ocurridos en las entidades después de que hayan sido recuperadas de la base de datos.
- **Persistencia de Datos:** realiza operaciones Insert, Update y Delete sobre la base de datos, en función de lo definido por la entidad.
- **Caching:** Realiza cacheo de primer nivel por defecto. Almacena las entidades que han sido recuperadas durante su ciclo de vida.
- **Manejo de Relaciones:** maneja relaciones usando fluent API en el enfoque Code-First.
- **Materialización de Objetos:** convierte datos crudos de tablas a objetos entidad.


# Ejemplo de Modelo de Dominio



# Ejemplo DbContext

```
internal class VehicleRentalDbContext : DbContext
{
    public VehicleRentalDbContext() : base("Name=VehicleRentalDbConnection")
    {
    }

    public IDbSet<BranchOffice> BranchOffices { get; set; }
    public IDbSet<Reservation> Reservations { get; set; }
    public IDbSet<Category> Categories { get; set; }
    public IDbSet<Person> Persons { get; set; }
    public IDbSet<Customer> Customers { get; set; }
    public IDbSet<CreditCard> CreditCards { get; set; }
}
```



Entity Sets  
(Repositories)

- [IDbSet<TEntity>](#) representa la colección de todas las entidades del contexto, o que se pueden consultar de la base de datos, de un tipo determinado.
- [DbSet<TEntity>](#) es una implementación concreta de IDbSet.

# IDbSet & DbSet

- `IDbSet<TEntity>` representa la colección de todas las entidades del contexto, o que se pueden consultar de la base de datos, de un tipo determinado.
- `DbSet<Tentity>` es una implementación concreta de `IDbSet`.
- Los objetos `DbSet` se crean a partir de un `DbContext` usando el método `DbContext.Set<TEntity>()`

# CONVENCIONES CODE-FIRST

---

# Convenciones Code-First

- Las APIs Code-First crean la base de datos y mapean las clases del dominio con la base de datos usando convenciones Code-First por defecto
  - Convención **Type Discovery**
  - Convención **Primary Key**
  - Convención **Relationship**
  - Convención **Foreign key**
  - Convención **Inheritance**
- Una convención es un conjunto de reglas por defecto para configurar automáticamente un modelo conceptual basado en las definiciones de las clases del dominio

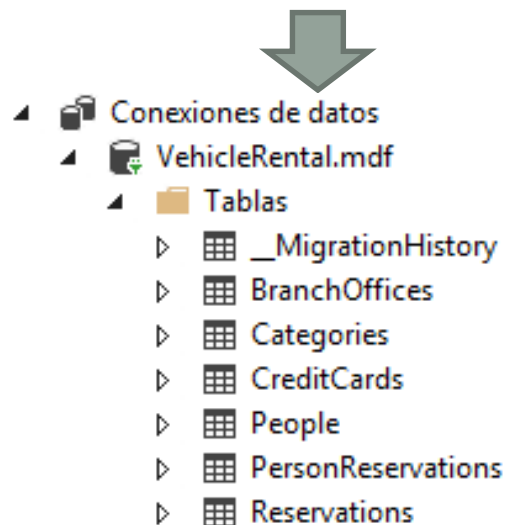
# Convención Type-Discovery

- Code-First creará **tablas** para las clases incluidas como propiedades **DbSet**
- Code-First también incluye cualquier **tipo referenciado** incluido en dichas clases

# Ejemplo Type-Discovery

- EF genera automáticamente una tabla para cada entidad DbSet

```
public IDbSet<BranchOffice> BranchOffices { get; set; }  
public IDbSet<Reservation> Reservations { get; set; }  
public IDbSet<Category> Categories { get; set; }  
public IDbSet<Person> Persons { get; set; }  
public IDbSet<Customer> Customers { get; set; }  
public IDbSet<CreditCard> CreditCards { get; set; }
```



Nombres de las tablas en Plural.  
Ej. People en vez de Person

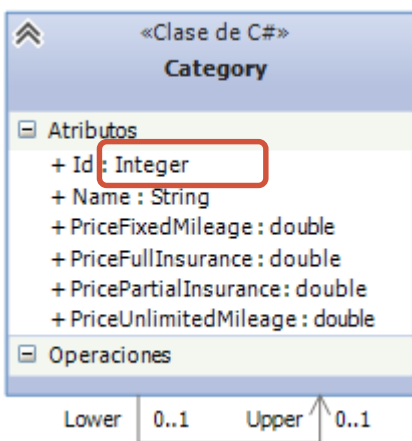
Tablas adicionales para  
relaciones muchos-a-muchos. Ej.  
PersonReservations



# Convención Primary-Key

- Code-First creará una clave primaria para una propiedad si el nombre de la propiedad es **Id** o **<nombre de clase>Id**
- El tipo de una propiedad de clave primaria puede ser cualquiera, pero si el tipo de la propiedad clave primaria es **numérico o GUID** (Identificador Único Global), será configurado como una **columna identidad (autoincrementable)**

# Ejemplo de Convención Primary-Key



```
CREATE TABLE [dbo].[Categories] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NULL,
    [PriceFixedMileage] FLOAT (53) NOT NULL,
    [PriceFullInsurance] FLOAT (53) NOT NULL,
    [PriceUnlimitedMileage] FLOAT (53) NOT NULL,
    [PricePartialInsurance] FLOAT (53) NOT NULL,
    [PriceAdditionalKm] FLOAT (53) NOT NULL,
    [Upper_Id] INT NULL,
    CONSTRAINT [PK_dbo.Categories] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.Categories_dbo.Categories_Upper_Id] FOREIGN KEY ([Upper_Id])
        REFERENCES [dbo].[Categories] ([Id])
);
```

	Nombre	Tipo de datos	Permitir valores NULL
	Id	int	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PriceFixedMileage	float	<input type="checkbox"/>
	PriceFullInsurance	float	<input type="checkbox"/>
	PriceUnlimitedMileage	float	<input type="checkbox"/>
	PricePartialInsurance	float	<input type="checkbox"/>
	PriceAdditionalKm	float	<input type="checkbox"/>
	Upper_Id	int	<input checked="" type="checkbox"/>

▲ **Claves** (1)  
PK\_dbo.Categories (Clave principal, Clustered: Id)

**Restricciones CHECK** (0)

▲ **Índices** (1)  
IX\_Upper\_Id (Upper\_Id)

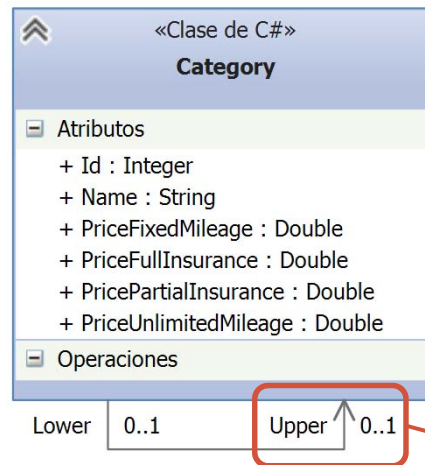
▲ **Claves externas** (1)  
FK\_dbo.Categories\_dbo.Categories\_Upper\_Id (Id)

**Desencadenadores** (0)

# Convenciones Relationship

- Code-First infiere el **tipo de relación** en la base de datos basándose en las **propiedades de navegación**:
  - **referencia** si la multiplicidad máxima es 1, o **colección** si es mayor que 1
- Code-First modelará una **relación uno a uno** si las clases incluyen dos propiedades referencia
- Code-First modelará una relación **uno a muchos** si las clases contienen una referencia y una colección
- Code-First modelará una relación **muchos a muchos** si las clases incluyen dos propiedades de colección

# Ejemplo de Relación Uno a Uno



```
public class Category
{
    public virtual int Id {get;set;}

    public virtual string Name {get;set;}

    public virtual double PriceFixedMileage
    {get;set;}

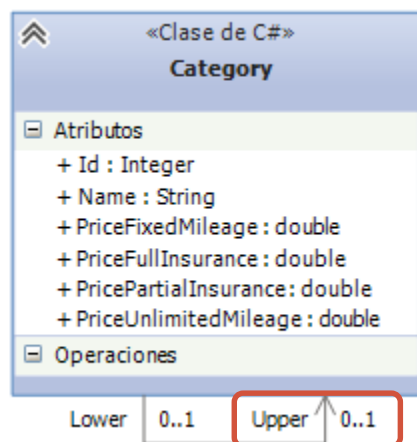
    public virtual double PriceFullInsurance
    {get;set;}

    public virtual double PriceUnlimitedMileage
    {get;set;}

    public virtual double PricePartialInsurance
    {get;set;}
}
```

```
public virtual Category Upper
{
    get;
    set;}
}
```

# Ejemplo de Relación Uno a Uno



```
CREATE TABLE [dbo].[Categories] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Name] NVARCHAR (MAX) NULL,
    [PriceFixedMileage] FLOAT (53) NOT NULL,
    [PriceFullInsurance] FLOAT (53) NOT NULL,
    [PriceUnlimitedMileage] FLOAT (53) NOT NULL,
    [PricePartialInsurance] FLOAT (53) NOT NULL,
    [PriceAdditionalKm] FLOAT (53) NOT NULL,
    [Upper_Id] INT NULL,
    CONSTRAINT [PK_dbo.Categories] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_dbo.Categories_dbo.Categories_Upper_Id] FOREIGN KEY ([Upper_Id])
        REFERENCES [dbo].[Categories] ([Id])
);
```

	Nombre	Tipo de datos	Permitir valores NULL
PK	Id	int	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PriceFixedMileage	float	<input type="checkbox"/>
	PriceFullInsurance	float	<input type="checkbox"/>
	PriceUnlimitedMileage	float	<input type="checkbox"/>
	PricePartialInsurance	float	<input type="checkbox"/>
	PriceAdditionalKm	float	<input type="checkbox"/>
	Upper_Id	int	<input checked="" type="checkbox"/>

▲ **Claves** (1)  
 PK\_dbo.Categories (Clave principal, Clustered: Id)

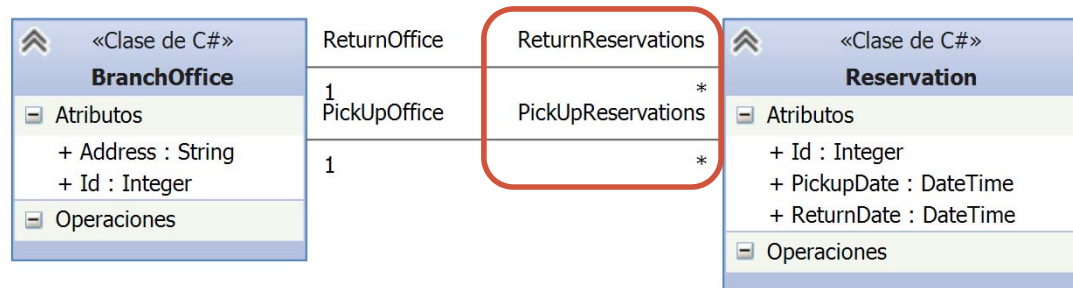
**Restricciones CHECK** (0)

▲ **Índices** (1)  
 IX\_Upper\_Id (Upper\_Id)

▲ **Claves externas** (1)  
 FK\_dbo.Categories\_dbo.Categories\_Upper\_Id (Id)

**Desencadenadores** (0)

# Ejemplo de Relación Uno a Muchos



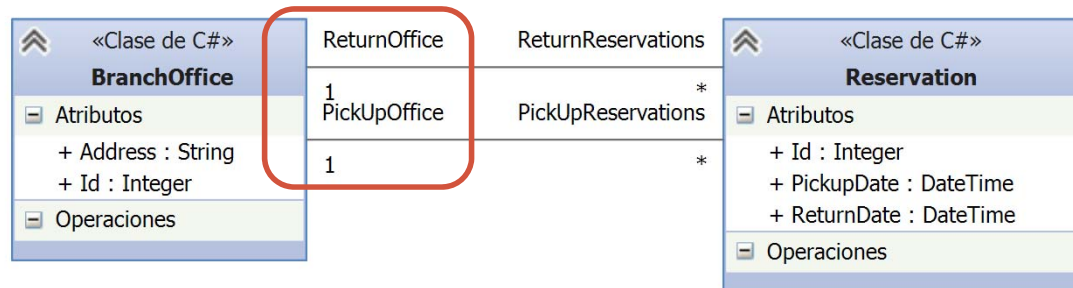
```
public partial class BranchOffice
{
    public string Address {get;set;}

    public int Id {get;set;}
}
```

```
    public virtual ICollection<Reservation> PickUpReservations
    {
        get;
        set;
    }

    public virtual ICollection<Reservation> ReturnReservations
    {
        get;
        set;
    }
}
```

# Ejemplo de Relación Uno a Muchos



```
public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}

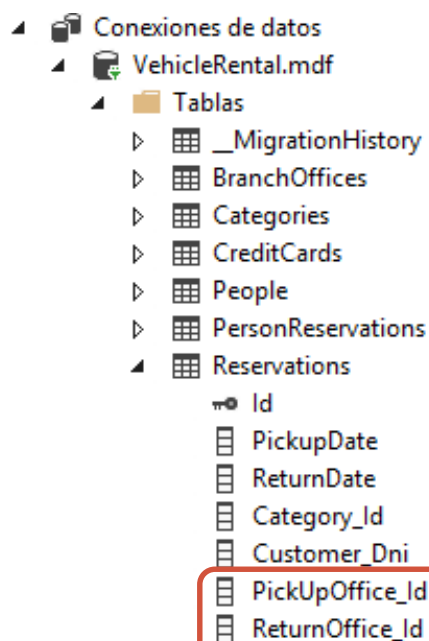
    public virtual BranchOffice PickupOffice {get;set;}
    public virtual BranchOffice ReturnOffice {get;set;}

    public virtual Category Category {get;set;}

    public virtual ICollection<Person> Drivers {get;set;}

    public virtual Customer Customer {get;set;}
}
```

# Ejemplo de Relación Uno a Muchos

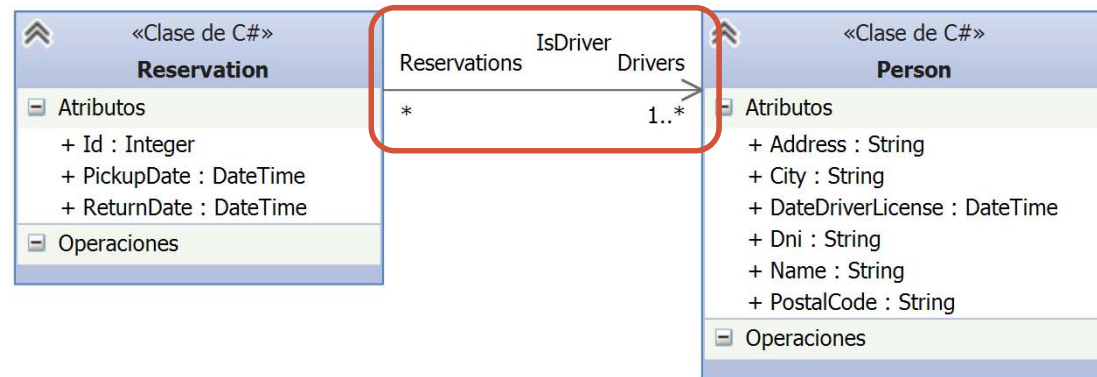


	Nombre	Tipo de datos	Permitir valores NULL
PK	Id	int	<input type="checkbox"/>
	PickupDate	datetime	<input type="checkbox"/>
	ReturnDate	datetime	<input type="checkbox"/>
	Category_Id	int	<input checked="" type="checkbox"/>
	Customer_Dni	nvarchar(128)	<input checked="" type="checkbox"/>
	PickUpOffice_Id	int	<input type="checkbox"/>
	ReturnOffice_Id	int	<input type="checkbox"/>

- Claves (1)**
  - PK\_dbo.Reservations (Clave principal, Clustered: Id)
- Restricciones CHECK (0)**
- Índices (4)**
  - IX\_Category\_Id (Category\_Id)
  - IX\_Customer\_Dni (Customer\_Dni)
  - IX\_PickUpOffice\_Id (PickUpOffice\_Id)
  - IX\_ReturnOffice\_Id (ReturnOffice\_Id)
- Claves externas (4)**
  - FK\_dbo.Reservations\_dbo.Categories\_Category\_Id (Id)
  - FK\_dbo.Reservations\_dbo.People\_Customer\_Dni (Dni)
  - FK\_dbo.Reservations\_dbo.BranchOffices\_PickUpOffice\_Id (Id)
  - FK\_dbo.Reservations\_dbo.BranchOffices\_ReturnOffice\_Id (Id)
- Desencadenadores (0)**



# Ejemplo de Relación Muchos a Muchos



```
public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}

    public virtual BranchOffice PickupOffice {get;set;}

    public virtual BranchOffice ReturnOffice {get;set;}

    public virtual Category Category {get;set;}

    public virtual ICollection<Person> Drivers {get;set;}

    public virtual Customer Customer {get;set;}
}
```

```
public partial class Person
{
    public string Dni {get; set;}

    public string Name {get; set;}

    public string Address {get; set;}

    public string City {get; set;}

    public string PostalCode {get; set;}

    public DateTime DateDriverLicense {get;set;}

    public virtual ICollection<Reservation> Reservations
    {get; set;}
}
```

# Ejemplo de Relación Muchos a Muchos

- ▲ Conexiones de datos
  - ▲ VehicleRental.mdf
    - ▲ Tablas
      - \_\_MigrationHistory
      - BranchOffices
      - Categories
      - CreditCards
      - People
      - PersonReservations
        - ▾ Person\_Dni
        - ▾ Reservation\_Id
      - Reservations

Nueva tabla, con clave primaria compuesta

	Nombre	Tipo de datos	Permitir valores NULL
PK	Person_Dni	nvarchar(128)	<input type="checkbox"/>
PK	Reservation_Id	int	<input type="checkbox"/>

- ▲ Claves (1)
  - PK\_dbo.PersonReservations (Clave principal, Clustered: Person\_Dni, Reservation\_Id)

Restricciones CHECK (0)

- ▲ Índices (2)
  - IX\_Person\_Dni (Person\_Dni)
  - IX\_Reservation\_Id (Reservation\_Id)

- ▲ Claves externas (2)
  - FK\_dbo.PersonReservations\_dbo.People\_Person\_Dni (Dni)
  - FK\_dbo.PersonReservations\_dbo.Reservations\_Reservation\_Id (Id)

Desencadenadores (0)

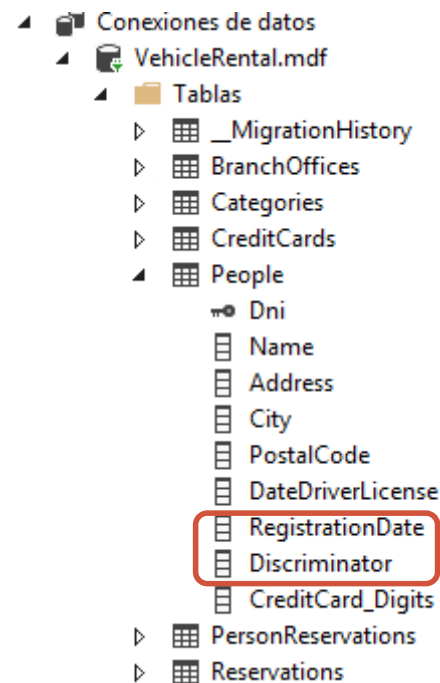
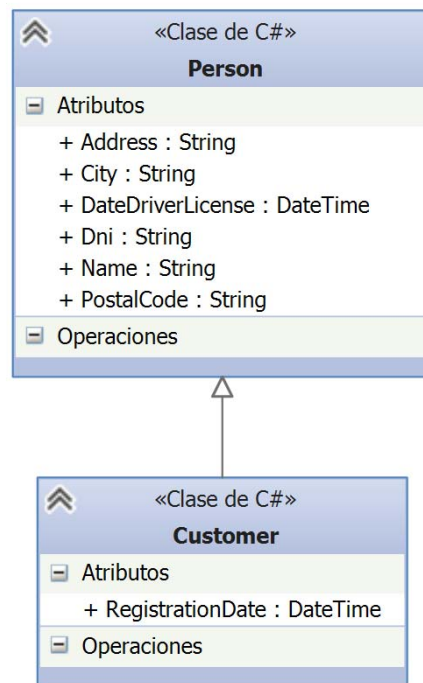
Clave primaria compuesta

Claves foráneas

# Convención de la herencia

- **Tabla por Jerarquía (TPH):** Esta aproximación sugiere una clase para toda la jerarquía de herencia de clases.
  - La tabla incluye una columna discriminadora que distingue entre herencia de clases.
  - Esta es la estrategia predeterminada en EF
- **Tabla por Tipo (TPT):** Esta aproximación sugiere una tabla separada para cada clase de dominio.
- **Tabla por clase concreta (TPC):** Esta aproximación sugiere una tabla para cada clase concreta sin incluir las clases abstractas.
  - Las propiedades de la clase abstracta serán parte de cada tabla de la clase concreta.

# Ejemplo de Tabla por Jerarquía



	Nombre	Tipo de datos	Permitir valores NULL
	Dni	nvarchar(128)	<input type="checkbox"/>
	Name	nvarchar(MAX)	<input checked="" type="checkbox"/>
	Address	nvarchar(MAX)	<input checked="" type="checkbox"/>
	City	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PostalCode	nvarchar(MAX)	<input checked="" type="checkbox"/>
	DateDriverLicense	datetime	<input type="checkbox"/>
	RegistrationDate	datetime	<input checked="" type="checkbox"/>
	Discriminator	nvarchar(128)	<input type="checkbox"/>
	CreditCard_Digits	nvarchar(128)	<input checked="" type="checkbox"/>

## Claves (1)

PK\_dbo.People (Clave principal, Clustered: Dni)

## Restricciones CHECK (0)

## Índices (1)

IX\_CreditCard\_Digits (CreditCard\_Digits)

## Claves externas (1)

FK\_dbo.People\_dbo.CreditCards\_CreditCard\_Digits (Digits)

## Desencadenadores (0)

Dni	Name	Address	City	PostalCode	DateDriverLic...	RegistrationDate	Discriminator	CreditCard_Di...
11111111A	Javier Murillo	Av. de las Tres ...	StartUpCity	11111	12/07/2015 0:00...	04/02/2016 0:00...	Customer	1223344556677...
22222222A	Carlos García	Plaza de los Cas...	HackerCity	99999	23/05/2014 0:00...	NULL	Person	NULL

# Elementos Clave de las Convenciones

Default Convention For	Description
Table Name	<Entity Class Name> + 's' EF will create DB table with entity class name suffixed by 's'
Primary key Name	1) Id 2) <Entity Class Name> + "Id" (case insensitive)  EF will create primary key column for the property named Id or <Entity Class Name> + "Id" (case insensitive)
Foreign key property Name	By default EF will look for foreign key property with the same name as principal entity primary key name. If foreign key property does not exists then EF will create FK column in Db table with <Dependent Navigation Property Name> + "_" + <Principal Entity Primary Key Property Name> e.g. EF will create Standard_StandardId foreign key column into Students table if Student entity does not contain foreignkey property for Standard where Standard contains StandardId
Null column	EF creates null column for all reference type properties and nullable primitive properties.
Not Null Column	EF creates NotNull columns for PrimaryKey properties and non-nullable value type properties.
DB Columns order	EF will create DB columns same as order of properties in an entity class. However, primary key columns would be moved first.
Properties mapping to DB	By default all properties will map to database. Use [NotMapped] attribute to exclude property or class from DB mapping.
Cascade delete	Enabled By default for all types of relationships.

# Configuración del Dominio de Clases

- **Sobrescribir** las convenciones previas configurando tus propias clases de dominio para proveer a EF con la información que necesita.
- Dos formas de configurar las clases de dominio:
  - **DataAnnotations:** configuración basada en atributos, que puede ser aplicada a las clases de dominio y sus propiedades.
  - **Fluent API:** cubre todo lo que las DataAnnotations pueden hacer y, además, permite configuraciones avanzadas que no son posibles con las anotaciones.

# DATA ANNOTATIONS

---

# Data Annotations

- Anotaciones que afectan a la nulidad o el tamaño de la columna en el esquema de la base de datos:
  - Key
  - Timestamp
  - ConcurrencyCheck
  - Required
  - MinLength
  - MaxLength
  - StringLength



# Data Annotations

- Otras anotaciones que determinan el esquema de la base de datos
  - Table
  - Column
  - Index
  - ForeignKey
  - NotMapped
  - InverseProperty

# Ejemplo de Data Annotations

```
public partial class Person
{
    [Key]
    public string Dni {get; set;}

    public string Name {get; set;}

    public string Address {get; set;}

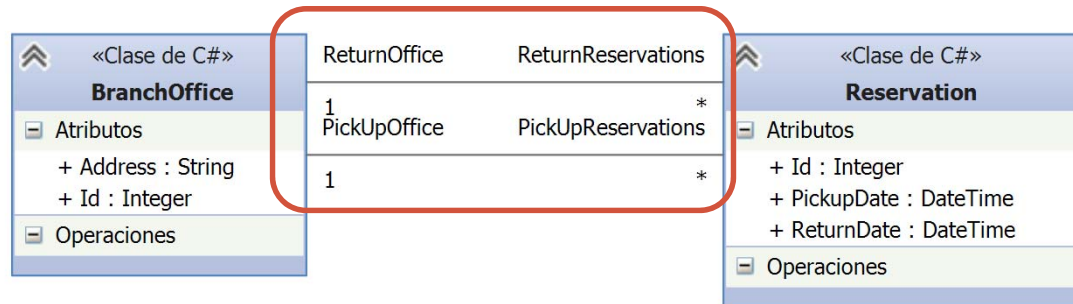
    public string City {get; set;}

    public string PostalCode {get; set;}

    public DateTime DateDriverLicense {get;set;}

    public virtual ICollection<Reservation> Reservations {get; set;}
```

# Ejemplo de Data Annotations



```

public partial class BranchOffice
{
    public string Address {get;set;}

    public int Id {get;set;}

    public virtual ICollection<Reservation>
    PickUpReservations {get; set;}

    public virtual ICollection<Reservation>
    ReturnReservations {get; set;}
}
  
```

```

public partial class Reservation
{
    public DateTime PickupDate {get;set;}

    public DateTime ReturnDate {get; set;}

    public int Id {get; set;}
  
```

```

    [InverseProperty("PickUpReservations")]
    public virtual BranchOffice PickUpOffice {get;set;}

    [InverseProperty("ReturnReservations")]
    public virtual BranchOffice ReturnOffice {get;set;}
  
```

```

    public virtual Category Category {get;set;}

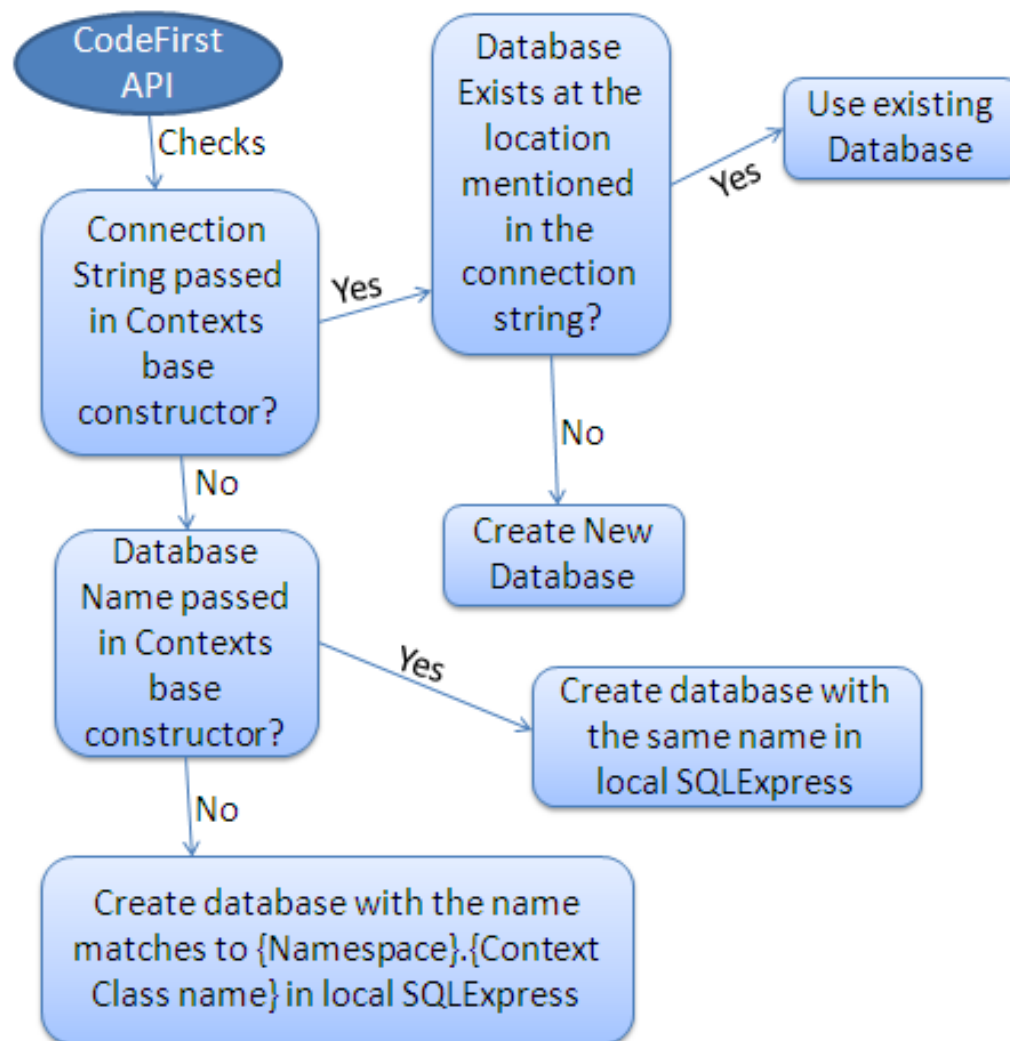
    public virtual ICollection<Person> Drivers {get;set;}

    public virtual Customer Customer {get;set;}
  }
  
```

# INICIALIZACIÓN DE LA BASE DE DATOS

---

# Inicialización de la Base de Datos



# Ejemplo de inicialización de la BD

- Cadena de conexión especificada en el constructor de la clase de contexto (derivada de DbContext)

```
public VehicleRentalDbContext() :  
    base("Name=VehicleRentalDbConnection") {}
```

- Cadena de conexión definida en el archivo de configuración **App.config** (o Web.config)

```
<connectionStrings>  
    <clear />  
    <add name="VehicleRentalDbConnection"  
connectionString="Server=(localdb)\mssqllocaldb;  
Database=VehicleRental;Trusted_Connection=True;MultipleAct  
iveResultSets=true" providerName="System.Data.SqlClient"  
/>  
</connectionStrings>
```

# OPERACIONES DE ACCESO A DATOS CON ENTITY FRAMEWORK

---

# Operaciones de Base de Datos

- IDbSet ofrece métodos para añadir, eliminar y recuperar objetos
- Permite expresar y ejecutar consultas
- Recupera los resultados de la consulta de la base de datos y los transforma en instancias de nuestro modelo de clases
- Puede hacer un seguimiento de los cambios en las entidades, incluyendo adición y borrado; y desencadenar la creación de los comandos de inserción, actualización y borrado que son enviados a la base de datos bajo demanda



# Ejemplo de operaciones EF

```
VehicleRentalDbContext context = new  
VehicleRentalDbContext();
```

```
context.Categories.Add(new Category("luxury",  
23, 12, 2, 32, 14));  
context.People.Add(new Person("222222222A", ...);  
Person p = context.People.Find("12345678A");  
context.People.Remove(p);  
context.SaveChanges();
```

```
context.People.Where(person => person.Id ==  
"123456789A")
```

# Resumen de características clave de EF

- EF es un framework de mapeado objeto-relacional (**Object-Relational Mapping**, ORM)
- EF automatiza las operaciones **CRUD** estándar (Create, Read, Update & Delete) para que el desarrollador no necesite escribirlas de forma manual
- **Code-First** permite centrar la atención en las clases de dominio y crear la base de datos a partir de éstas.
- Code-First crea la base de datos y realiza el mapeo entre las clases del dominio y la base de datos utilizando convenciones y configuración mediante el uso de anotaciones de datos o Fluent API
- **DbContext** permite expresar y ejecutar consultas, mantiene el seguimiento de cambios y materializa objetos

# Tareas

- Explora la aplicación VehicleRental e identifica los diferentes elementos de EF (contexto de la BD, entidades persistidas, inicialización de la BD, cadena de conexión)
- ¿Qué patrón de acceso a datos se utiliza en EF y cómo/dónde se implementa en la aplicación VehicleRental?
- ¿Cuáles son los beneficios de tener una interfaz como la IDAL cuando se trabaja con EntityFramework?

# Tareas

- Entender el significado de las diferentes anotaciones de datos:

[http://www.tutorialspoint.com/entity\\_framework/entity\\_framework\\_data\\_annotations.htm](http://www.tutorialspoint.com/entity_framework/entity_framework_data_annotations.htm)

- Tarea Avanzada. Entender cómo funciona Fluent API

[http://www.tutorialspoint.com/entity\\_framework/entity\\_framework\\_fluent\\_api.htm](http://www.tutorialspoint.com/entity_framework/entity_framework_fluent_api.htm)

# Bibliografía y referencias

- Hirani, Z., et al. Entity Framework 6 Recipes 2013.
- Entity Framework Documentation (MSDN)
  - [Entity Framework Code First Conventions](#)
  - [Entity Framework Code First Data Annotations](#)
  - [Entity Framework Fluent API – Relationships](#)
  - [Entity Framework Fluent API - Configuring and Mapping Properties and Types](#)
  - [Entity Framework Loading Related Entities](#)
- Tutoriales Online
  - <http://www.entityframeworktutorial.net>
  - [http://www.tutorialspoint.com/entity framework/](http://www.tutorialspoint.com/entity_framework/)

# Ejemplo de Tabla por Jerarquía

■