# Esercitazione di Fine Settimana – Week 6

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Isabella |
|  |  | Cognome | Centofanti |
|  |  | Data | 25/06/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Descrivere le modalità di definizione del modello dati in Entity Framework?

Entity Framework è una tecnologia di accesso al dato. Definisce un modello di astrazione dei dati e quindi è indipendente dall’origine dei dati. Abbiamo tre modalità di creazione del modello:

DATABASE-FIRST: il modello viene importato da un DB esistente, che mi creerà quindi le classi in C#. Se modifico il DB posso quasi sempre farlo;

MODEL-FIRST: è il modello che definisce il codice: crea uno schema dal modello; se faccio delle modifiche perso tutto! il codice non viene riutilizzato.

CODE-FIRSt: quello che usiamo noi. E’ un implementazione fisica basata sul nostro codice, si crea il modello dal codice! Mi concentro sul design del dominio o meglio,sull’app che voglio ottenere. Ci basta conoscere C# per farlo. Per poterlo utilizzare basta installare dei pacchetti tramite il Nuget Package Manager (Microsoft.EntityFrameworkCore…).

1. Scrivere con un esempio pratico come definire una chiave primaria ed una chiave esterna utilizzando le data annotation e fluent api.

Le fluent api e le data annotation vengono utilizzate con Entity Framework per l’accesso al dato. Ci aiutano a definire le caratteristiche che avrà la tabella nel nostro Db di SQL.

**Nelle data annotation**:

[Key]

Public int ID {get;set;}

È l’indicazione per la primary key

[ForeignKey(nameof(IDPersona))] scrivo l’annotazione fk e quale sarà la mia variabile

Public int IDPersona {get;set;} mia FK

Public Persona Persona {get;set;} mia Navigation Property che mi collega alla tab persona

**Nelle Fluent api:**

nella classe Configuration della mia entità(supponiamo Veicolo),dopo aver esteso IEntityTypeConfiguration<Veicolo>

nel metodo Configure(implementato dall’interfaccia),scrivo:

builder.HasKey(k => k.ID); -> indico quale variabile è la mia PK

poi definisco la FK e il collegamento con la tabella

builder.HasOne(e => e.Persona) -> navigation property della mia classe veicolo

.WithMany(r => r.Veicoli) -> collection che avrò nella mia classe Persona

.HasForeignKey(t => t.IDPersona);

1. Descrivere l’utilizzo delle Migration e i vantaggi che ne derivano

Le Migration servono in caso di modifiche al modello da riflettere nel DB. Ho la possibilità di aggiungere un campo alla mia tabella senza dover eseguire direttamente il drop.

Ci sono diversi tipi di migration:

1)le migration automatiche: avvio un comando dalla package manager console e se il comando ha accesso creo il file:

-enable-Migration

2)Code-based migration: sono utili se ho il DB già pronto e devo fare delle modifiche. Specifico infatti i comandi:

Add-Migration <migration name>

Questo comando crea una nuova classe con tutte le modifiche

Update-Database

Aggiorna il DB

Sono utili sia perché mi consentono di creare il db,che di modificarlo in corso d’opera(avendolo già pronto)senza dover cancellare completamente una tabella e inoltre mi consentono di fare anche il rollback di una modifica(sempre con un comando).

**Esercitazione Pratica**

Sulla base del modello definito in allegato realizzare i requisiti richiesti.

L'applicazione deve consentire di gestire i clienti dell'assicurazione. In particolare deve essere possibile:

- Inserire nuovi clienti

- Inserire una polizza (rcauto, furto, vita) per un cliente

- Stampare i dati delle polizze presenti a db (comprese le info sul cliente che l'ha stipulata\*).

\*In fase di stampa dei dati del cliente deve essere visualizzabile anche la spesa totale mensile che il cliente sostiene per tutte le sue polizze.

Requisiti tecnici:

Utilizzare le Migration di EF per la modifica del DB

Utilizzare le Fluent-Api per la gestione dei requisiti del database

Realizzare un menù per l’interazione con l’utente