Esercitazione di Fine Settimana – Week 6

Nome valentina

Cognome comerci

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 ci permette di eseguire operazioni su database senza passare attraverso sql, questo è possibile sfruttando una tecnologia definita come ORM (object relational mapping) che traduce il nostro codice in query comprensibili dal DBMS, questo comporta un completo disaccoppiamento tra dati e applicazione e la possibilità di mantenere la stessa rappresentazione anche se cambia il modello fisico.

POSSIAMO USARE DIVERSI APPROCCI PER DEFINIRE IL MODELLO DATI

DATABASE-FIRST
- il modello viene importato da un DB esistente

MODEL-FIRST
- modello database viene creato dal designer di visual studio
- non favorisce il riutilizzo del codice

CODE-FIRST
- modello creato dal codice
- implementazione fisica viene basata sul nostro codice
- ci si concentra molto di più sul domain design
- possiamo utilizzare un unico linguaggio (csharp)
- possiamo mettere facilmente sotto source control il nostro database
- evitiamo la mole di codice autogenerato da edmx

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

DATA ANNOTATION

//chiave primaria

[Key]

```
public string CODICE { get; set; }
```

//Chiave esterna verso veicolo

public String EntityCODICE { get; set; }

//Navigation Property

public Entity Entity{ get; set; }

FLUENT API

//chiave primaria

Definiamo la classe, poi in un folder context creiamo un context specifico dove definiamo:

```
builder.HasKey(k => k.Codice);
```

//foreign key

```
builder.HasOne(x => x.Entity2).WithMany(y => y.Entities).HasForeignKey(f => f.Entity2Code);
```

3. Descrivere l'utilizzo delle Migration e i vantaggi che ne derivano

Il modello dati cambia continuamente mano a mano che nuove funzioni vengono implementate: nuove entità o proprietà vengono aggiunte e rimosse e gli schemi di database devono essere modificati di conseguenza per essere mantenuti sincronizzati con l'applicazione.
Le migrazioni permettono di aggiornare in modo continuo lo schema del database per mantenerlo sincronizzato con il modello di dati dell'applicazione, preservando al contempo i dati esistenti nel database.

Quando viene fatta una modifica del modello di dati, viene aggiunta una migrazione da lato codice, EF Core confronta il modello corrente con quello vecchio per determinare le differenze e genera un file di origine della migrazione.
Tutte le migrazioni vengono registrate in una tabella di cronologia speciale, consentendoci di sapere quali migrazioni sono state applicate e quali no.