

Sezione 17

Lavorare con i dati


ADO.NET: risultati tabellari

```
SELECT Id, Title, Author FROM Courses;
```


Id	Title	Author

ADO.NET: risultati tabellari

SELECT Id, Title, Author FROM Courses WHERE Id={id}; SELECT Id, Title
FROM Lessons WHERE CourseId={id};



Id	Title	Author



Id	Title

ADO.NET: risultati scalari

```
INSERT INTO Courses (Id, Title) VALUES (1, 'Titolo'); SELECT  
last_insert_rowid();
```

last_insert_rowid()
1715

ADO.NET: risultati scalari

```
SELECT COUNT(*) FROM Courses;
```

COUNT(*)
1530

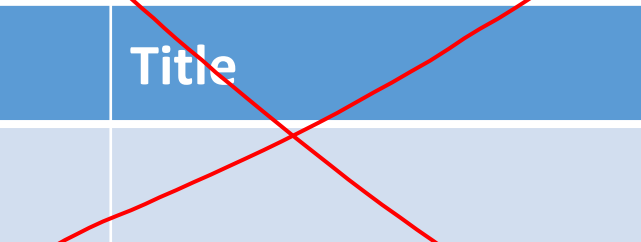
ADO.NET: risultati scalari

```
SELECT AVG(Rating) FROM Courses;
```

AVG(Rating)
3.73125

ADO.NET: comandi

UPDATE Courses SET Title='Nuovo titolo' WHERE Id=34; UPDATE Courses SET Title='Altro titolo' WHERE Id=56;



Id	Title

2

*Fornisce invece il
Numero di righe
interessate dal
comando
(cumulativo)*

*Un comando SQL
non restituisce
risultati tabellari*

ADO.NET: leggere un risultato tabellare

```
SELECT Id, Title FROM Courses;
```

Id	Title

ADO.NET: leggere un risultato tabellare

Si usa il metodo `ExecuteReaderAsync` del `SqlCommand`

- Restituisce un `SqlDataReader`

```
using var cmd = new SqlCommand("SELECT Id, Title FROM Courses", conn);
using SqlDataReader reader = await cmd.ExecuteReaderAsync();
var dataSet = new DataSet();
do
{
    var dataTable = new DataTable();
    dataSet.Tables.Add(dataTable);
    dataTable.Load(reader);
} while (!reader.IsClosed);
```

ADO.NET: inviare un comando

Si usa il metodo `ExecuteNonQueryAsync` del `SqlCommand`

- Restituisce un `int` che rappresenta il numero di righe interessate

```
using var cmd = new SqlCommand("UPDATE Courses ... ", conn);  
int affectedRows = await cmd.ExecuteNonQuery();
```

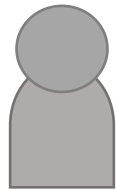
ADO.NET: leggere un risultato scalare

Si usa il metodo `ExecuteScalarAsync` del `SqlCommand`

- Prende il valore della prima riga, prima colonna e lo restituisce come object

```
using var cmd = new SqlCommand("SELECT COUNT(*) FROM Courses", conn);  
object result = await cmd.ExecuteScalarAsync();
```

Concorrenza pessimistica



Mario Rossi
DOCENTE

Modifica corso

Titolo

Come fare gli origami

Descrizione

B *I*    

Lorem ipsum dolor sit amet, **consectetur adipiscing elit**, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat



Luigi Verdi
ASSISTENTE

Spiacenti, questa pagina al momento
non è accessibile perché in uso
da un altro utente.

Ti preghiamo di tornare più tardi.

Concorrenza ottimistica



Mario Rossi

DOCENTE

I dati sono stati salvati con successo

Come fare gli origami [Modifica](#)

Lorem ipsum dolor sit amet, **consectetur adipiscing elit**, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



Luigi Verdi

ASSISTENTE

Modifica corso

Spiacente, non è stato possibile salvare perché nel frattempo il corso è stato già aggiornato da qualcun altro

Titolo

Come fare gli origami

Descrizione

B *I*

Lorem ipsum dolor sit amet, **consectetur adipiscing elit**, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

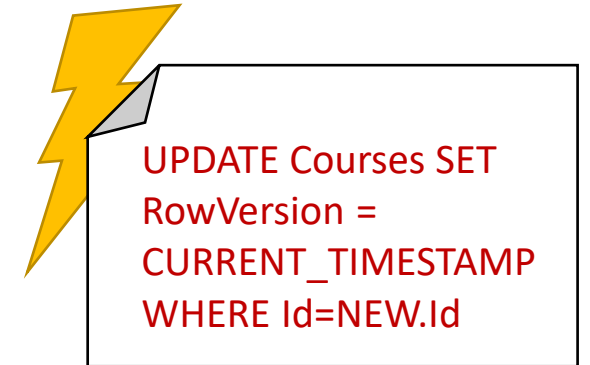
Aggiungere la colonna RowVersion

Tecnologia Database	Aggiunta di una colonna per la concorrenza ottimistica a una tabella esistente
MySQL / MariaDB	ALTER TABLE Courses ADD RowVersion TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;
Sql Server	ALTER TABLE Courses ADD RowVersion rowversion;
Oracle	ALTER TABLE Courses ADD RowVersion timestamp; <i>+trigger (https://stackoverflow.com/questions/1614233#answer-1614340)</i>
PostgreSql	ALTER TABLE Courses ADD RowVersion timestamp default current_timestamp; <i>+ trigger (https://x-team.com/blog/automatic-timestamps-with-postgresql/)</i>
Sqlite	ALTER TABLE Courses ADD RowVersion DATETIME; <i>+ trigger (https://stackoverflow.com/questions/6578439#answer-29095144)</i>

Trigger

È un oggetto del database (proprio come lo sono una tabella o un indice) e si crea con il comando **CREATE TRIGGER**

Esegue istruzioni SQL in reazione a un evento che ha riguardato una tabella



Id	Title	Author	RowVersion
1	Modellare la ceramica	Mario Rossi	2020-03-05 15:33:15
2	Chitarra per principianti	Mario Rossi	2020-03-05 15:33:00

Implementare la concorrenza ottimistica

Modifica corso 2020-05-02 14:00:00

Titolo

Come fare gli origami

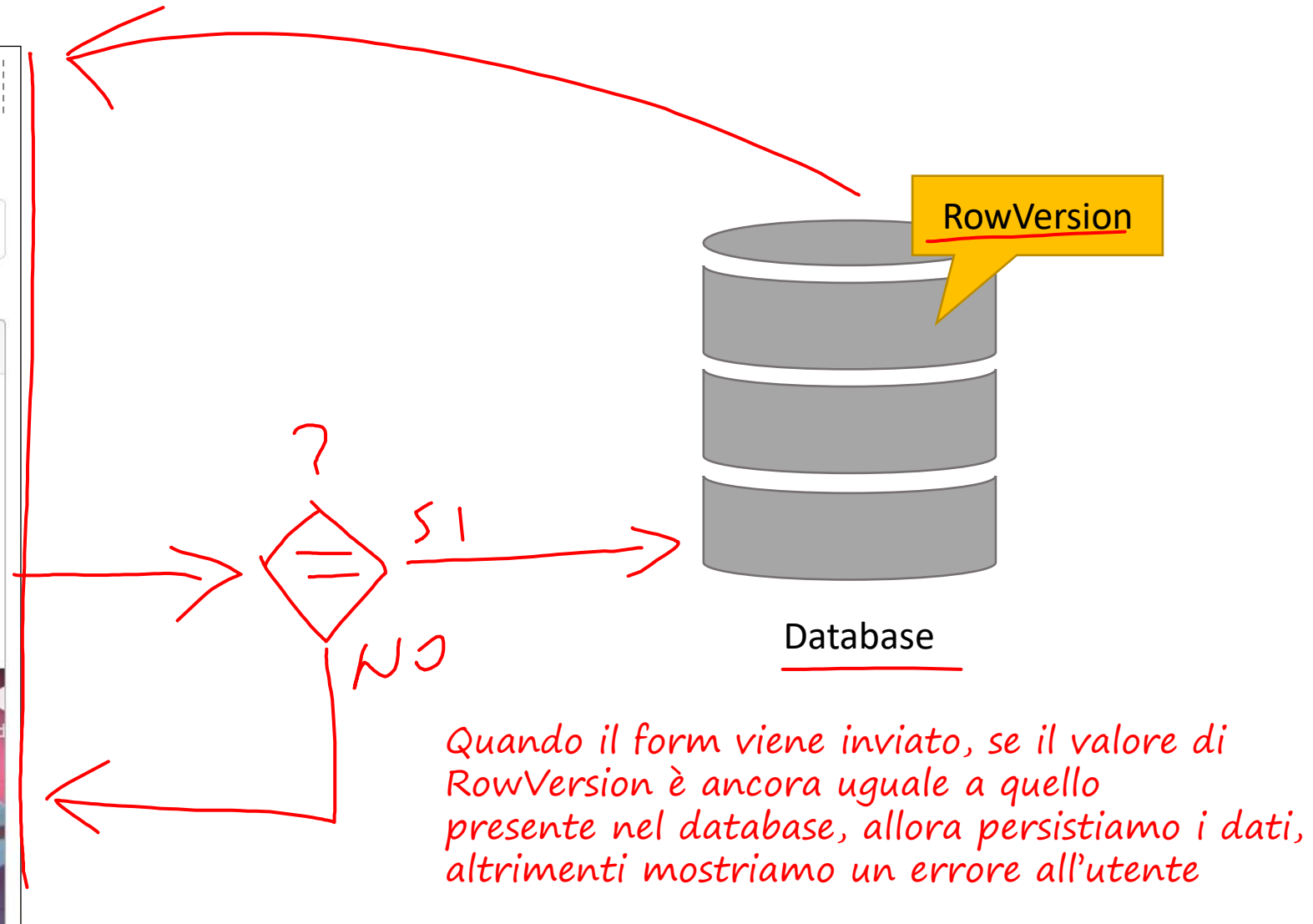

Descrizione

B *I* [List Icons] [Link Icon] [Code Icon]

Lorem ipsum dolor sit amet, **consectetur adipiscing elit**, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

- Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
- Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

♥ ORIGAMI VOLPE DA 1 FOGLIO ♥ LAVORETTI ...



Concorrenza ottimistica con ADO.NET

Quando dobbiamo aggiornare un corso, inviamo il comando **UPDATE** e nella clausola **WHERE** indichiamo anche la colonna RowVersion

```
int affectedRows = await db.CommandAsync(  
    $"UPDATE Courses SET Title={inputModel.Title}  
    WHERE Id={inputModel.Id} AND RowVersion={inputModel.RowVersion}"  
);
```

...e poi controlliamo il valore di affectedRows.

Se affectedRows == 0, solleviamo un'eccezione.

Concorrenza ottimistica con EFCore

Creiamo la proprietà `RowVersion` nella classe di entità `Course` e mappiamola così nel metodo `OnModelCreating` del `DbContext`.

```
entity.Property(course => course.RowVersion).IsRowVersion();
```

Al momento di aggiornare l'entità `Course`, valorizziamo così la sua proprietà `RowVersion`.


```
dbContext.Entry(course).Property(course => course.RowVersion)  
    .OriginalValue = inputModel.RowVersion;
```


Catturiamo l'eventuale `DbUpdateConcurrencyException`


Prerequisiti per usare le migration

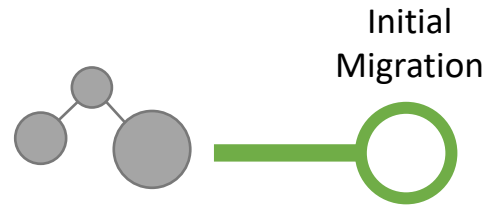
- Entity Framework Core
- Installare il global tool **dotnet-ef**
`dotnet tool install --global dotnet-ef`
- Installare il pacchetto NuGet `Microsoft.EntityFrameworkCore.Design`
`dotnet add package Microsoft.EntityFrameworkCore.Design`

Approccio code-first

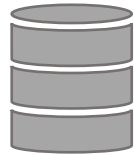
 Models/Entities

 Course.cs

 Lesson.cs




Modello





Database

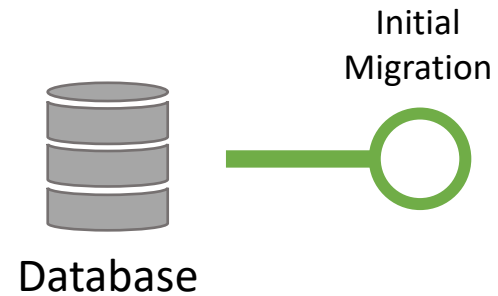
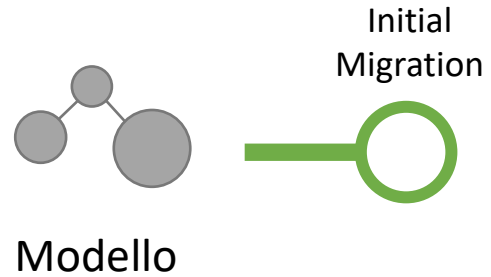
```
> dotnet ef migrations add InitialMigration
```

Far evolvere la struttura del database

 Models/Entities

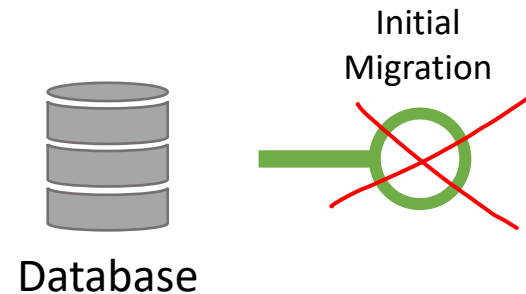
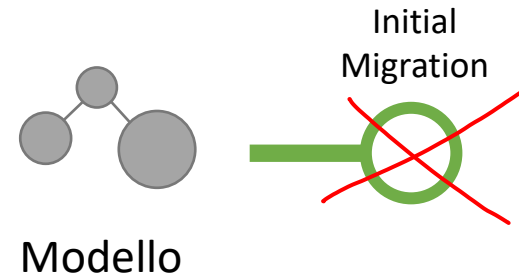
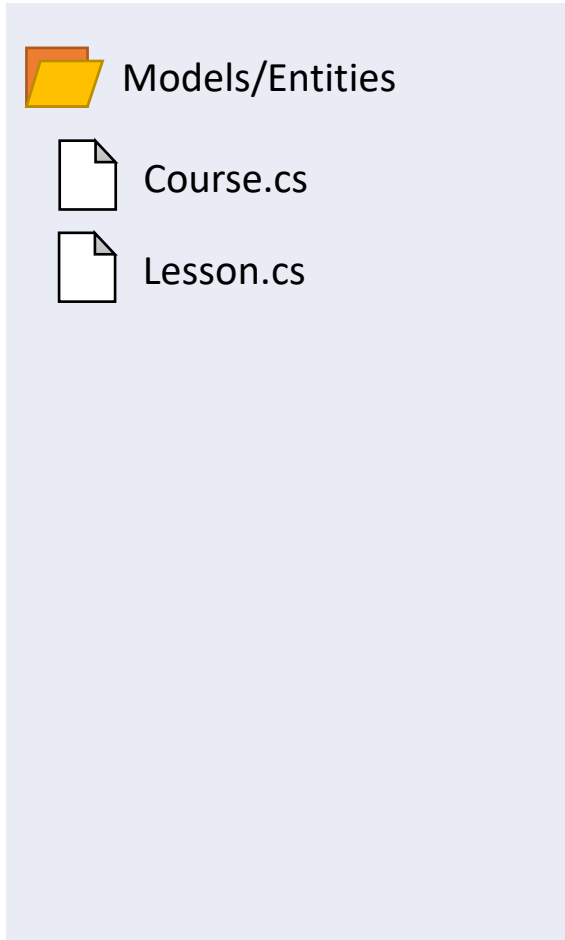
 Course.cs

 Lesson.cs



```
> dotnet ef database update InitialMigration
```

Annullare una migration già applicata



*Nome della migration a cui
si vuole tornare, oppure 0
per tornare all'inizio*

```
> dotnet ef database update 0  
> dotnet ef migrations remove
```

Per eliminare l'ultima migration creata nel progetto

Creare un vincolo di univocità con EFCore

Nel DbContext, al mapping dell'entità Course aggiungiamo:

```
entity.HasIndex(course => course.Title).IsUnique();
```

Nel generare una migration, EFCore considera non solo le classi di entità ma anche il loro mapping.

Aggiungere dei trigger alla migration

Entity Framework Core non supporta i trigger...

...ma ci lascia eseguire delle istruzioni SQL arbitrarie.

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.Sql("CREATE TRIGGER ...");
}
```

```
protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.Sql("DROP TRIGGER ...");
}
```


Seeding del database

Inserire delle righe in una o più tabelle;

Utile per precaricare elenchi che non cambiano mai (o quasi) come i nomi delle province italiane, delle lingue supportate, delle valute;

Nel mapping dell'entità nel DbContext aggiungiamo:

```
entity.HasData(  
    new Course("Titolo del corso", "Nome autore"),  
    new Course("Altro titolo", "Altro autore")  
);
```

Generare script SQL con le migration

In ambiente di produzione non abbiamo la .NET Core SDK

~~dotnet ef database update~~

```
dotnet ef migrations script InitialMigration > file.sql
```

```
dotnet ef migrations script InitialMigration UniqueCourseTitle  
> file.sql
```

Modificare entità correlate alla principale

Tutto in una pagina: singolo form

Titolo	
<input type="text"/>	

Descrizione

<u>Titolo lezione</u>	<u>Durata</u>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>

Indicato quando abbiamo poche entità correlate e poche caselle per ciascuna di esse. Altrimenti la pagina risulterà caotica.

Modificare entità correlate alla principale

Tutto in una pagina: singolo form

The diagram illustrates a single-page form for modifying related entities. The form is enclosed in a dashed red border. It contains the following fields and actions:

- Titolo**: A text input field with a green save icon and a red delete icon to its right. A red arrow points from this field to the word **UPDATE**.
- Descrizione**: A large text area. A red arrow points from this field to the word **UPDATE**.
- Table**: A table with two columns, Titolo lezione and Durata. There are four rows of input fields. Red arrows point from each row to the word **UPDATE**.

On the left side of the form, there are three red arrows pointing right, indicating a sequence of updates.

Modificare entità correlate alla principale

Tutto in una pagina: molteplici form

The image shows a user interface for managing data. It consists of two main parts, both enclosed in red dashed borders. The top part is a form with two fields: 'Titolo' (Title) and 'Descrizione' (Description). To the right of the 'Titolo' field are two icons: a green square with a white floppy disk icon (representing 'Update') and a red square with a white trash can icon (representing 'Delete'). The bottom part is a table with two columns: 'Titolo lezione' (Lesson Title) and 'Durata' (Duration). There are four rows in the table. Each row has input fields for both columns. To the right of each row are the same 'Update' and 'Delete' icons as in the top form.



Titolo	Descrizione		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		

Titolo lezione	Durata		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		
<input type="text"/>	<input type="text"/>		

→ UPDATE o DELETE

Modificare entità correlate alla principale

Pagine multiple: form modifica corso





Titolo

Descrizione

Prezzo intero

Prezzo corrente

Pagine multiple: form modifica lezione



Titolo

Descrizione

Durata

Nuovi componenti



Spostare una classe da un namespace all'altro

The screenshot shows the Visual Studio IDE with the following components:

- EXPLORER:** The file explorer on the left shows a project structure with folders like `Services`, `Application`, and `Courses`. The file `AdoNetCourseService.cs` is highlighted under the `Courses` folder.
- Code Editor:** The main editor shows the content of `AdoNetCourseService.cs`. The namespace is being changed from `MyCourse.Models.Services.Application` to `MyCourse.Models.Services.Application.Courses` on line 17. The code includes several `using` statements and a class definition with properties and a constructor.
- Context Menu:** A right-click context menu is open over the namespace line, offering two options:
 - Sposta il file in 'Models\Services\Application'
 - Modifica lo spazio dei nomi in 'MyCourse.Models.Services.Application.Courses'** (highlighted in blue)
- Terminal:** The bottom panel shows the `TERMINAL` tab with the message: "Terminal will be reused by tasks, press any key to close it."
- Status Bar:** The bottom status bar shows the current file is `AdoNetCourseService.cs` at line 17, column 47, with a UTF-8 encoding and CRLF line endings.

Associare entità con Entity Framework Core

Opzione #1: valorizzare la proprietà CourseId dell'entità Lesson

```
var lesson = new Lesson();  
lesson.CourseId = inputModel.CourseId;  
dbContext.Add(lesson);  
await dbContext.SaveChangesAsync();
```

Associare entità con Entity Framework Core

Opzione #2: valorizzare la proprietà di navigazione Course

```
Course course = await dbContext.Courses.FindAsync(inputModel.CourseId);  
var lesson = new Lesson();  
lesson.Course = course;  
dbContext.Add(lesson);  
await dbContext.SaveChangesAsync();
```

Associare entità con Entity Framework Core

Opzione #3: aggiungere l'entità Lesson alla collezione Lessons

```
Course course = await dbContext.Courses.FindAsync(inputModel.CourseId);  
var lesson = new Lesson();  
course.Lessons.Add(lesson);  
dbContext.Add(lesson);  
await dbContext.SaveChangesAsync();
```

Modifica lezione

Salva



Titolo

Lezione 1

Descrizione

B *I*

At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga. Et harum quidem rerum facilis est et expedita distinctio.

Durata stimata

00:06:09



Ordine

1000

Punto n° 17

Consentire la gestione delle lezioni del corso, che include inserimento, modifica ed eliminazione.

A woman with short blonde hair and dark-rimmed glasses is sitting at a desk in an office. She is wearing a white short-sleeved top and has her arms crossed. The background shows a large window with horizontal blinds, through which a city skyline is visible at night. The office interior is dimly lit, with some lights visible through the blinds.

Falle sparire dal database,
non mi interessa mantenerle

Modifica corso

Salva



Titolo

Web marketing facile

Descrizione

B *I*

Lorem ipsum dolor sit amet consectetur adipisicing elit. Et minima sunt quia nulla voluptate, illum eum incidunt repudiandae beatae, vero accusantium minus hic eveniet omnis laborum architecto inventore dolores molestias? Placeat sequi sapiente hic culpa optio quisquam est fugiat dolorem itaque non, quasi cum voluptates quidem repudiandae doloribus? Autem mollitia esse odio nihil atque non ea quisquam consequuntur exercitationem? Amet! Non ut itaque qui tempore illum! Amet, accusamus minima. Ut rerum praesentium obcaecati sint, accusantium maxime odio voluptatibus quaerat repudiandae corrupti magnam, non perferendis. Officia recusandae delectus dolor quidem reprehenderit! Dolores eos eveniet quod molestiae praesentium earum fugit similique fugiat? Molestias veniam eos enim! Ad, id. Rem similique explicabo deleniti possimus facilis rerum deserunt minus aperiam suscipit! Ipsa, id laudantium. Rem distinctio ex magni unde doloremque a, quae nesciunt, obcaecati animi perspiciatis earum, vel consectetur pariatur tempora dicta. Quos architecto delectus, quis nostrum repudiandae molestiae quas distinctio atque cupiditate temporibus? Deserunt optio molestias alias aspernatur. Ducimus veniam quibusdam, sit saepe illum officiis obcaecati dolore atque totam consequatur

Email di contatto

tutor@example.com

Prezzo intero

EUR 19,99

Prezzo corrente

EUR 17,99

Immagine rappresentativa



Punto n° 19

Un corso non può essere eliminato fisicamente dal database. Può essere portato dal docente sullo stato "Deleted" che lo renderà di fatto invisibile e imm modificabile da chiunque, compreso il docente.

Implementare l'eliminazione

Views/Lessons/Edit.cshtml

Salva



```
<form method="post">
  <button type="submit">Salva</button>
  <button type="submit" asp-action="Delete" asp-route-id="@Model.Id">🗑️</button>
  <!-- Qui altri elementi -->
</form>
```

*Più prioritario rispetto all'attributo
asp-action eventualmente presente
sul form*

Eliminare un'entità con EFCore

Recuperiamo l'entità e poi la eliminiamo

```
Lesson lesson = await dbContext.Lessons.FindAsync(inputModel.Id);  
dbContext.Remove(lesson);  
await dbContext.SaveChangesAsync();
```


Eliminare un'entità con EFCore (trucco)

Creiamo un'entità fittizia e poi inganniamo il Change Tracker

```
var lesson = new Lesson {  
    Id = inputModel.Id  
};  
dbContext.Entry(lesson).State = EntityState.Deleted;  
await dbContext.SaveChangesAsync();
```

Eliminare un'entità con Entity Framework Plus

Nuget: <https://www.nuget.org/packages/Z.EntityFramework.Plus.EFCore/>

```
await dbContext.Lessons.Where(lesson => lesson.Id == inputModel.Id).DeleteAsync();
```

Soft-delete

Per prima cosa aggiungiamo una nuova colonna alla tabella Courses

Id	Title	Author	RowVersion	Status
1	Corso di ceramica	Mario Rossi	2020-02-05 14:00:00	Deleted

La colonna Status controlla la visibilità di un corso



*Draft
Published
Deleted*

Soft-delete con Entity Framework Core

1. Quando vogliamo eliminare il corso impostiamo lo Status su Deleted

```
course.ChangeStatus(CourseStatus.Deleted);
```

2. Aggiungiamo un "Global Query Filter" al mapping nel DbContext

```
entity.HasQueryFilter(c => c.Status != CourseStatus.Deleted);
```

Global Query Filter

Servono ad applicare dei criteri di filtro "invarianti", cioè che devono essere sempre applicati, a prescindere dal caso d'uso.

- Per implementare la soft-delete;
- Per applicazioni multi-tenant.

Soft-delete con ADO.NET

1. Quando vogliamo eliminare il corso impostiamo Status su Deleted

```
int affected = await db.CommandAsync(  
    $"UPDATE Courses SET Status='Deleted' WHERE Id={inputModel.Id}");
```

2. Modifichiamo ogni query SQL per includere il filtro

```
DataSet dataSet = await db.QueryAsync(  
    $"SELECT Id, Title, Author FROM Courses WHERE  
    WHERE Id={inputModel.Id} AND Status<>'Deleted'");
```

Soft-delete (considerazioni)

- Con EFCore è facile, mentre se usate ADO.NET dovete ricordarvi di aggiornare tutte le vostre query;
- Le righe restano nelle tabelle ma questo NON sostituisce il backup periodico del database che deve essere fatto SEMPRE;
- Può essere usato in combinazione con l'eliminazione fisica di una riga per realizzare una sorta di "Cestino".

Progresso nella specifica

- Punto 6: visualizzazione delle lezioni;
- Punto 17: gestione delle lezioni;
- Punto 19: eliminazione (soft-delete).

Requisiti funzionali

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23		

Requisiti non funzionali

a	b	c	d
---	---	---	---