**Plan: ASP.NET and Databases**

**Objective**

In this lecture, the students should be introduced to an overview of Entity Framework Core and the Code First Approach, EF Components and EF Core Configuration. An introduction to CRUD operations should be made and database migration should be explained.

**Motivation**

The Entity Framework Core is the standard ORM framework for .NET and .NET Core.

**Content**

### Entity Framework Core: Overview (~55 min)

* **Entity Framework Core: Overview**
  + The standard **ORM framework** for **.NET**   
    and **.NET Core**
  + Provides LINQ-based data queries and **CRUD** operations
  + Automatic **change tracking** of in-memory objects
  + Works with many relational databases (with different providers)
  + Open source with independent release cycle
* **ASP.NET Core MVC + Entity Framework**
* **What is the Code First Approach?**
  + **Code First** means to write the .NET classes and let EF Core create the **database** from the **mappings**
* **Why Use Code First?**
  + Write code **without** having to define **mappings** in XML or **create** database **tables**
  + Define objects in **C# format**
  + Enables database persistence with no configuration
  + Changes to code can be **reflected** (migrated) in the schema
  + **Data Annotations** or **Fluent API** describe properties
    - **Key**, **Required**, **MinLength**, etc.
* **Code First Basic Workflow** 
  + Define the data model (**Code First** or **Scaffold from DB**)
  + Write & execute query over **IQueryable**
  + EF generates & executes an **SQL query** in the **DB**
  + EF transforms the query results into .NET objects
  + Modify data with C# code and call "**Save Changes()**"
  + Entity Framework generates & executes SQL command to modify the DB

### Q&A [Sli.do] (~5 min)

### BREAK: 15 min

### EF Core Components (~20 min)

* **Domain Classes (Models)**
  + Bunch of normal C# classes (POCO)
    - May contain **navigation** **properties** for **table** **relationships**
  + Recommended to be in a **separate class library**
  + Another example of a domain class (model)
* **The DbContext Class**
  + Usually named after the databasee.g. **ShoppingListAppDbContext**
  + Manages model classes using **DbSet<T>** type
  + Easily navigate through **table relations**
  + Managing database **creation**/**deletion**/**migration**
  + Executing **LINQ queries** as native **SQL queries**
  + **DbContext** properties
    - **Database** - **EnsureCreated**/**Deleted** methods, DB Connection
    - **ChangeTracker** - holds info about the **automatic change tracker**
* **Defining DbContext Class**

### EF Core Configuration (~20 min)

* **EF Core Setup**
  + To add EF Core support to a project in Visual Studio:
    - Install it from **Package Manager Console**
    - Or using **.NET Core CLI**
  + EF Core is modular – any **data providers** must be installed too
  + To use the Entity Framework Core **tools**, install also
* **How to connect to SQL Server?**
  + In ASP.NET Core **connection** **string** is in the **appsettings.json** file and has the following **properties**
  + Use the**DbContext** and tell it to useSQL with the connection string in in the **Program** class
* **Database.EnsureCreated()**
  + When you create the DB context, you can call **Database.EnsureCreated()**
  + This will **create the DB + schema**, when the DB is missing
  + EnsureCreated() does not use migrations 🡪 you should drop the enrite DB when you change the DB schema
* **Database.EnsureCreated() – Example**

### Q&A [Sli.do] (~10 min)

### BREAK: 15 min

### CRUD Operations (~40 min)

* **Live Demo of the ShoppingList App**
* **Reading Data**
  + ProductsControllerData
  + Controller+Model
  + View
* **Creating Data**
  + Controller+Model
  + View
* **Updating existing data**
  + Controller
  + View
* **Deleting existing data**
  + Controller

### Q&A [Sli.do] (~10 min)

### BREAK: 10 min

### Database Migrations (~20 min)

* **What are database migrations?**
  + Updating database schema **without losing data**
    - Adding/dropping tables, columns, etc.
  + Migrations in EF Core keep their **history**
    - Entity Classes, DB Context versions are all **preserved**
  + **Automatically** generated by certain EF tools
* **Migrations in EF Core**
  + Add a migration
  + Undo a migration
  + Commit changes
* **Migrate the ShoppingListApp**
* **Auto Run Migration Scripts at Startup**
  + This will apply the migration scripts (if not yet applied)
  + Simple, but can cause problems 🡪 not recommended in production
  + Recommended approach: migrate the database by hand

### Q&A [Sli.do] (~10 min)

**Exercise**

Give tasks and examples that include CRUD operations in ASP.NET Core MVC with EF.

**Evaluation & Exam**

The lecture will be included in the exam.