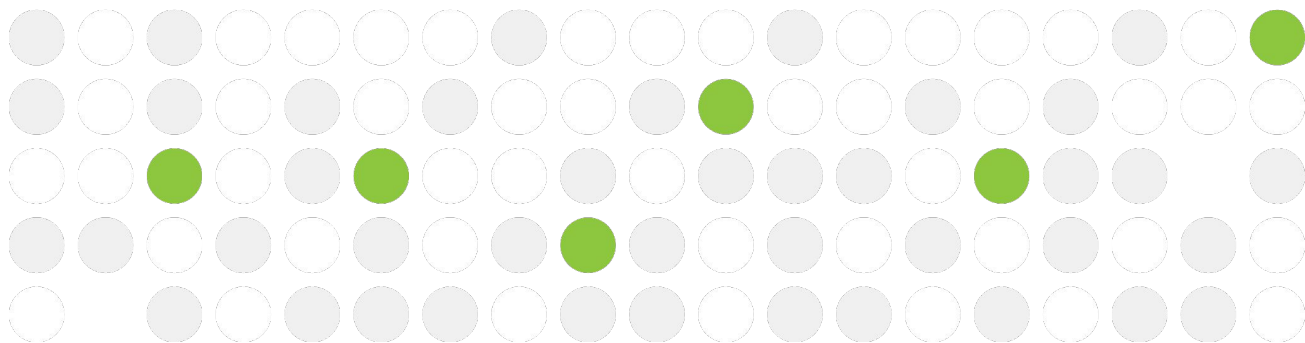




EXPERT NETWORK

# Web Development with **ASP.NET Core 7**



## Who we are?

- 18 years old and still counting
- 100 colleagues working on several disciplines: NET, PHP, Front-End, Mobile, QA, PO, UI/UX Design
- 120+ delivered projects
- 10+ active project teams
- We are here to stay. We are expanding our unique campus.



# Great News

Our **Full Stack .NET Internship** will start soon.

Wanna join us?

- Send your CV at [hr@expertnetwork.ro](mailto:hr@expertnetwork.ro).
- Application Deadline: March 19th

For more details access our website: [expertnetwork.ro](http://expertnetwork.ro)



EXPERT NETWORK

# The Challenge

How does building an application based on the knowledge you gain along the way sound?

During the last training session(s) you will be able to sign up for this and here's what the best 3 apps will translate into:

- A special prize!
- Internship applicants will have an advantage in the selection process!



EXPERT NETWORK

# Trainers

## Web

### Web Development With ASP.NET Core 7



Remus  
Bălăucă



Gabriel  
Gorgan



Alexandru-Cristian  
Stan

# The Sessions

1. **Data Access**
2. **Concepts and Techniques**
3. **ASP.NET Core Introduction**
4. **ASP.NET Core Advanced**
5. **Deploy in the Cloud**

\*Note that each session builds upon the previous one.



# For this session

- The mission!
- SQL
- Entity Framework Core
- Database Context
- LINQ

\*Don't forget about the prerequisites!



# The mission!

In order to better understand the technologies, and also because the program is called “FII Practic”, we need to learn by challenging ourselves to implement a tool of some sort.

The project: “**FII Practic Cars**”

Build a web application that manages the FII Practic Car Fleet.





# First: What needs to be stored?

- User details
- Vehicle details
- Information on who drives the vehicles



# Storage solution?

We will use a relational database because:

- Reliability
- Ease of use
- Easy to get started
- Inexpensive (\*not always)
- Can handle large amounts of data (\*to a certain degree)



# Relational Database

Term dates back to 1970 (E. F. Codd). Hand numerous implementations over the decades. The current most popular are:

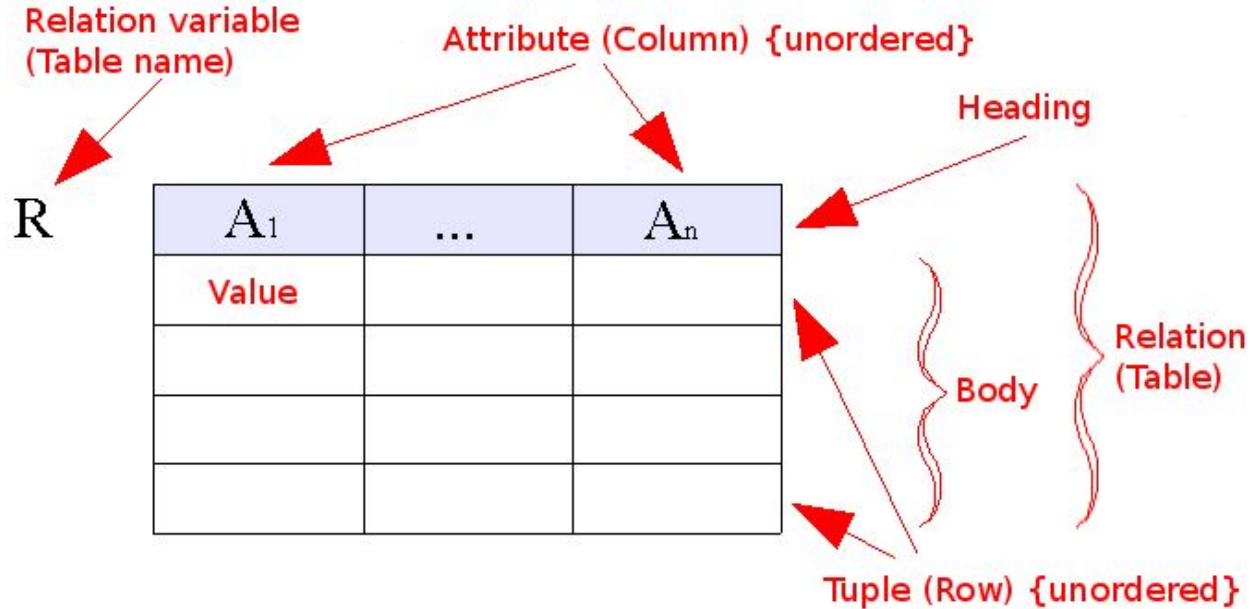
- Oracle
- MySql
- **Microsoft SQL Server** <- *we will use this one*
- PostgreSQL

\*The cloud-hosted solutions are quickly gaining popularity.

<https://db-engines.com/en/ranking/relational+dbms>



# Relational Model



[https://en.wikipedia.org/wiki/Relational\\_model](https://en.wikipedia.org/wiki/Relational_model)



# Modeling Our Data

Vehicles	Brands	Models	Users
VIN	Name	Name	First Name
Registration Date	Logo	Model Year	Last Name
Registration Number	<b>Driver License</b>	Engine Displacement	Email
Inspection Validity		Fuel Type	Password Hash
	Number		Birth Date
	Validity		Avatar



# SQL Server Data Types

Exact Numeric	Approx. Numeric	Date and Time	Strings
int	float	date	char
bigint	real	datetime	nchar
decimal		datetime2	varchar
money		time	nvarchar
bit			text

# Primary Keys

A **primary key** is a special relational database table column (or combination of columns) designated to uniquely identify each table record.

The Database Engine enforces data uniqueness by automatically creating a unique index for the primary key columns.

If clustered or nonclustered is not specified for a primary key constraint, clustered is used if there is no clustered index on the table.

# Foreign Keys

In a **foreign key** reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table. This column becomes a foreign key in the second table.

Unlike primary key constraints, creating a foreign key constraint does not automatically create a corresponding index. However, manually creating an index on a foreign key is often useful.

<https://docs.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver15>





# Relationship Types

- 1 - 1 (one-to-one)
- 1 - n (one-to-many)
- m - n (many-to-many)



# Exercise 1: Database Creation

```
CREATE DATABASE FilPracticCars;
```

```
CREATE TABLE Vehicles
```

```
(  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    VIN CHAR(17) NOT NULL,  
    RegistrationNumber NVARCHAR(10) NOT NULL,  
    RegistrationDate DATE NOT NULL,  
    InspectionValidUntil DATE NOT NULL,  
    ExteriorColor NVARCHAR(50) NULL,  
    InteriorColor NVARCHAR(50) NULL  
)
```

```
ALTER TABLE Vehicles ADD IsDeleted BIT NOT NULL DEFAULT 0
```

```
CREATE TABLE Users
```

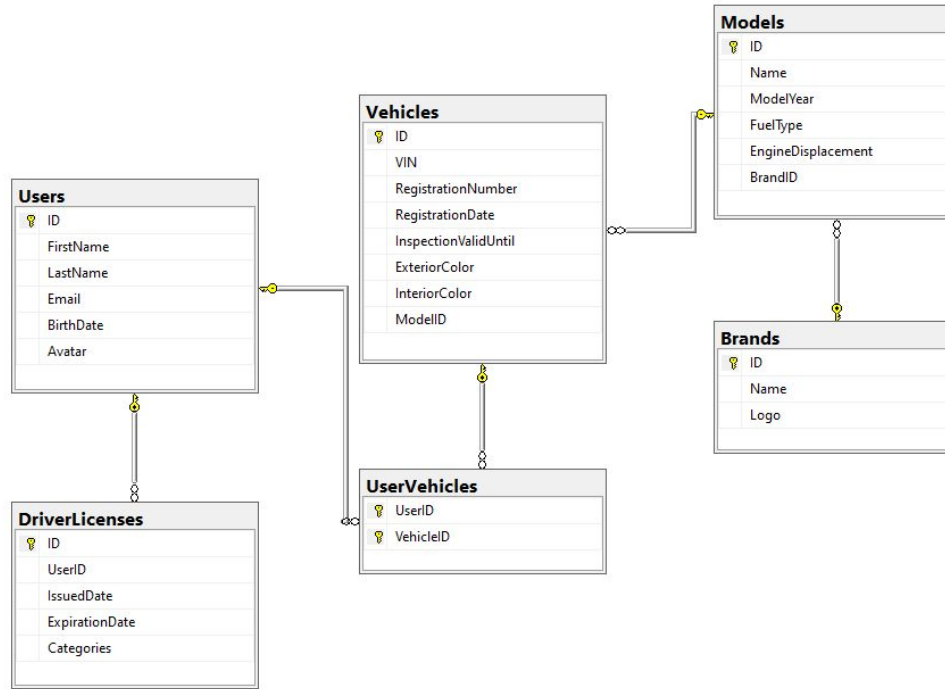
```
(  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    FirstName NVARCHAR(400) NOT NULL,  
    LastName NVARCHAR(400) NOT NULL,  
    Email NVARCHAR(320) NOT NULL,  
    PasswordHash NVARCHAR(32) NOT NULL,  
    BirthDate DATE NOT NULL,  
    Avatar NVARCHAR(1024) NULL  
)
```

```
CREATE TABLE Brands
```

```
(  
    ID INT IDENTITY(1,1) PRIMARY KEY,  
    Name NVARCHAR(400) NOT NULL,  
    Logo NVARCHAR(1024) NULL  
)
```



# Result



# ORM

Object–Relational Mapping (tools) for converting data between incompatible type systems using object-oriented programming languages.

This creates, in effect, a "virtual object database" that can be used from within the programming language.

\*We will use **Entity Framework Core**.

[https://en.wikipedia.org/wiki/Object%E2%80%93relational\\_mapping](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping)



# Entity Framework Core

EF Core can serve as an object-relational mapper (O/RM), which:

- Enables .NET developers to work with a database using .NET objects.
- Eliminates the need for most of the data-access code that typically needs to be written.

Data access is performed using a **model**. A model is made up of **entity classes** and a **context object** that represents a session with the database. The context object allows querying and saving data.

<https://docs.microsoft.com/en-us/ef/core/>



EXPERT NETWORK

# Context & Entities

In our project, the Context should be a class that represents a session with the FIIPracticCars database. => CarsContext

The context should contain Entity classes, one for each table in our situation. We will need to add sets of each type as members of the FIIPracticCarsContext.



# Exercise 2: EF Core Setup

1. Install package **Microsoft.EntityFrameworkCore.SqlServer**
2. Create Entities classes
3. Create DbContext class
4. Configure DbContext to use your database
5. Test by doing some queries



# Conventions

There are some out-of-the-box conventions that make setup easier:

- ID member of a class will be considered a Primary Key.
- Property names should match the table column.
- The Table name should be pluralized (Users table ↔ User class).
- FK and Navigation Properties names

<https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key#conventions>





# Model Builder

In case the Conventions do not apply, we will need to use the `OnModelCreating` method on the `DbContext` to configure the relationships.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
```

<https://docs.microsoft.com/en-us/ef/core/modeling/relationships?tabs=fluent-api%2Cfluent-api-simple-key%2Csimple-key#manual-configuration>



# Model Builder

For our database we will need the following:

```
modelBuilder.Entity<UserVehicles>()  
    .HasKey(uv => new { uv.UserID, uv.VehicleID });
```

```
modelBuilder.Entity<User>()  
    .HasMany(u => u.Vehicles)  
    .WithMany(v => v.Users)  
    .UsingEntity<UserVehicles>();
```

```
modelBuilder.Entity<User>()  
    .HasOne(u => u.DriverLicense)  
    .WithOne(dl => dl.User)  
    .HasForeignKey<DriverLicense>(dl => dl.UserID);
```



# LINQ

Language-Integrated Query (LINQ) is the name for a set of technologies based on the integration of query capabilities directly into the C# language.

With LINQ, a query is a first-class language construct, just like classes, methods, events. You write queries against strongly typed collections of objects by using language keywords and familiar operators.

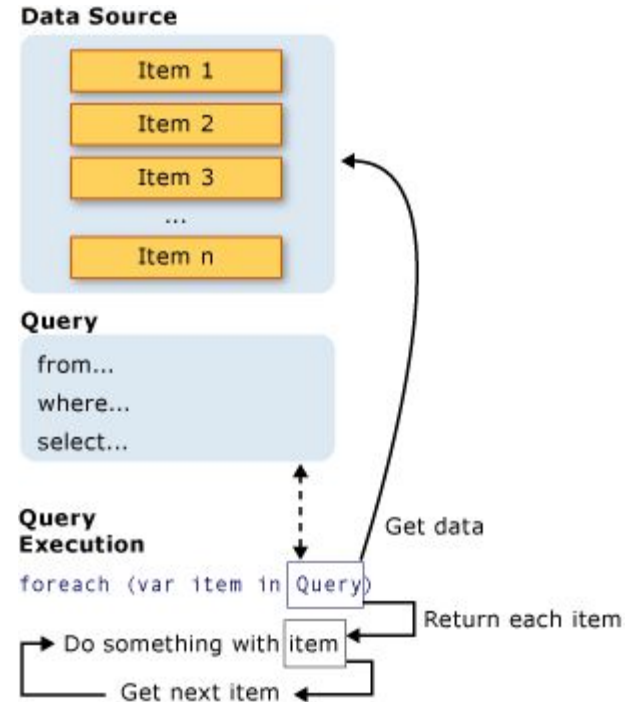
<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries>



# LINQ

All LINQ query operations consist of three distinct actions:

1. Obtain the data source.
2. Create the query.
3. Execute the query.



# LINQ Syntaxes

## Method Syntax

```
var users = context.Users
    .Where(u => u.Email.EndsWith("@expertnetwork.ro"))
    .ToList();
```

## Query Syntax

```
var users = (from user in context.Users
    where user.Email.EndsWith("expertnetwork.ro")
    select user).ToList();
```

\*Both syntaxes accomplish the same thing, but they look different.

\*\* You and your team should pick one and use it in the whole project.



# LINQ Operators

**Restriction:** Where

**Projection:** Select, SelectMany

**Partitioning:** Take, Skip, TakeWhile, SkipWhile

**Grouping:** GroupBy

**Ordering:** OrderBy, OrderByDescending, ThenBy, ThenByDescending

**Conversion:** ToList, ToArray, ToDiscionary

**Aggregate:** Count, Sum, Min, Max, Average

**Quantifiers:** Any, All



# Next Week

## Concepts and Techniques

- Repository
- Inversion of Control
- Dependency Injection
- Starting the Web Project

\*Don't forget the feedback form!

