C#
Data Access and Entity Framework Core

Rasmus Lystrøm
Associate Professor
ITU

Agenda

Databases (SQL Server)

Old school SQL

SQL Injection

Secrets

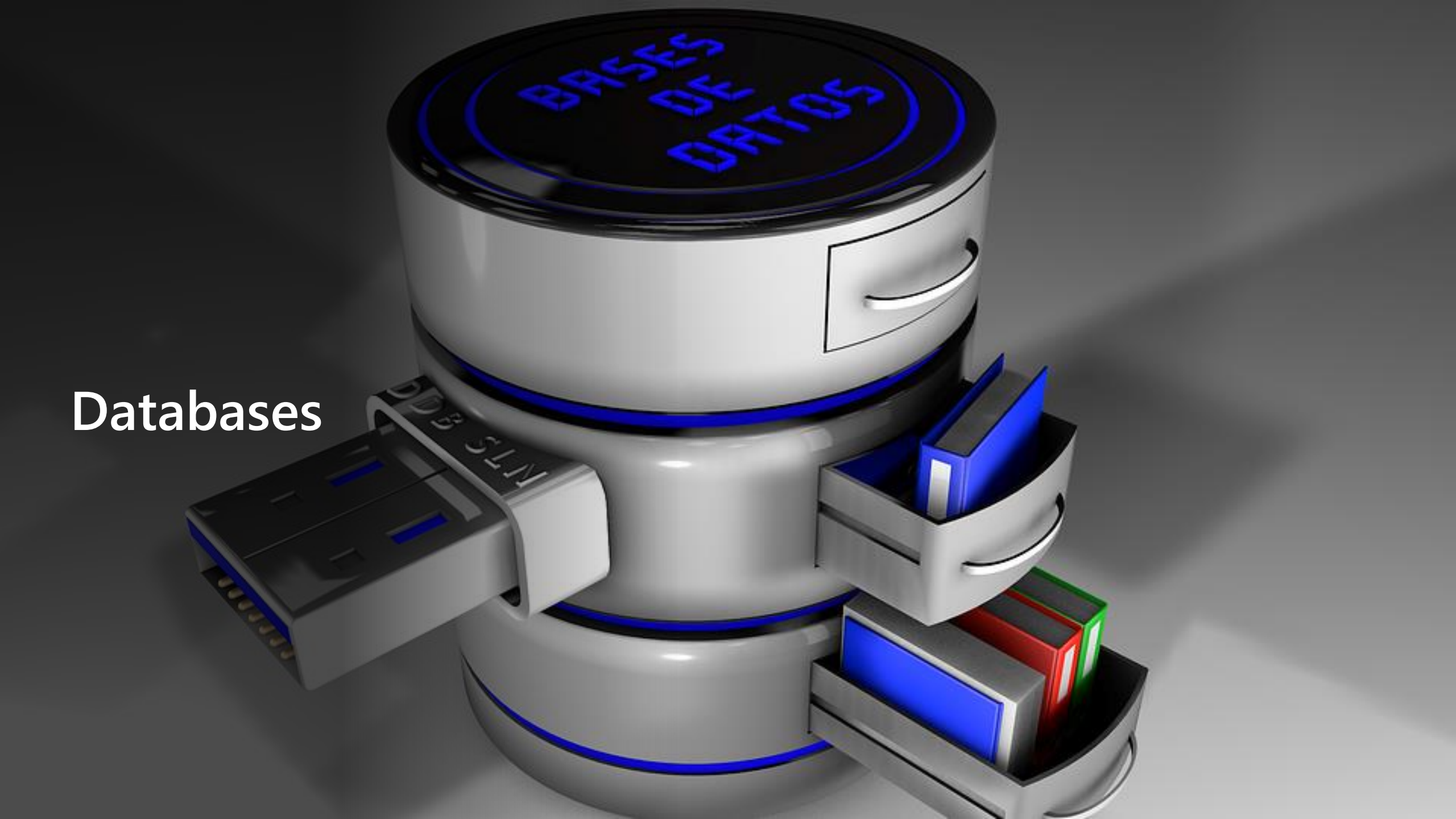
The IDisposable interface

Entity Framework Core

Clean Onions

Lazy vs. Eager Loading

Databases



Databases

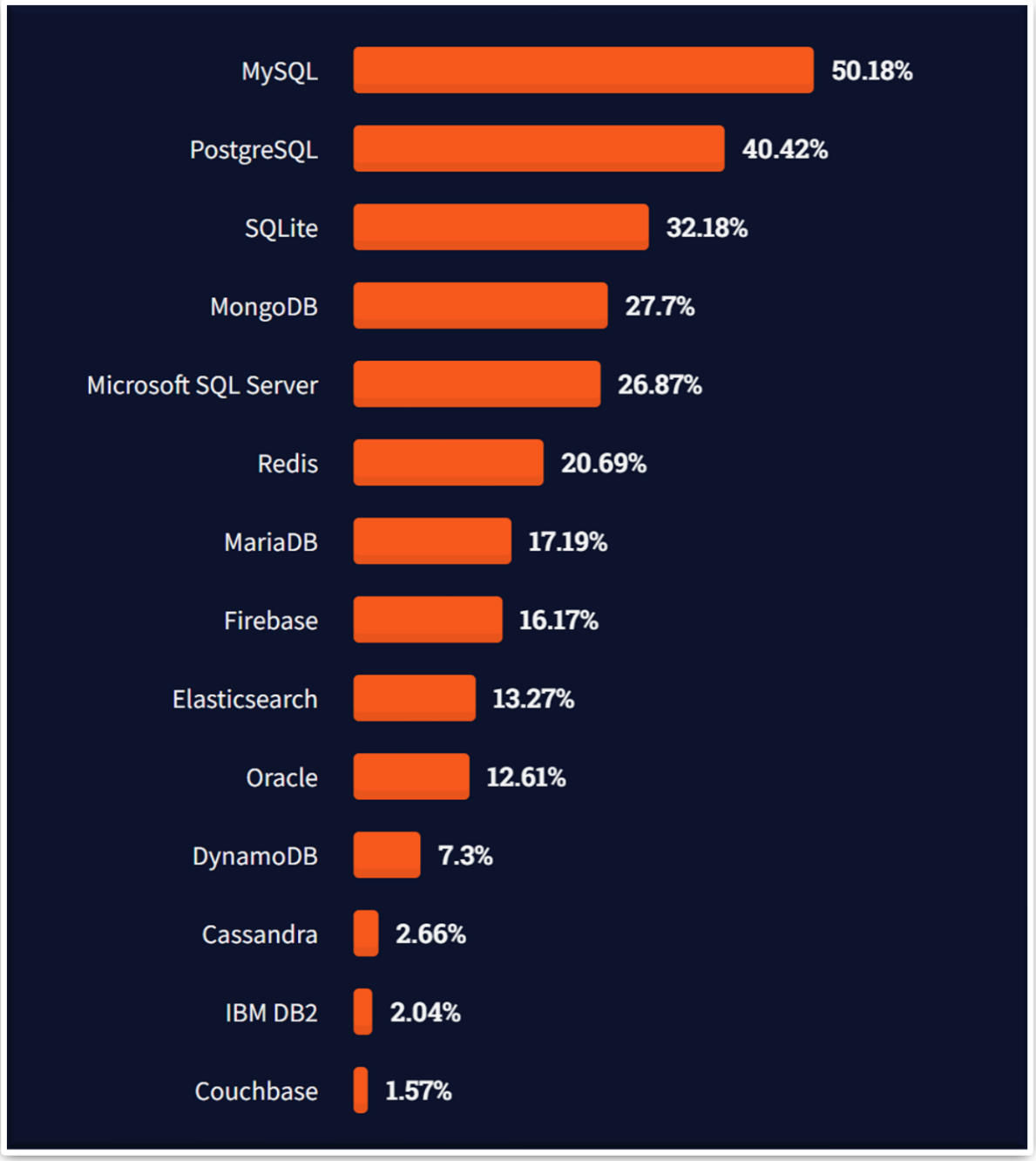
Microsoft SQL Server
Oracle Database
IBM Db2
MySQL
MariaDB
PostgreSQL
SQLite

Azure Cosmos DB
Amazon DynamoDB
MongoDB
Couchbase
Redis
Elasticsearch
Neo4j

Microsoft

Most popular databases

https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies



SQL Server

```
docker pull mcr.microsoft.com/mssql/server:2019-latest

$password = New-Guid

docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=$password" `
    -p 1433:1433 `
    -d mcr.microsoft.com/mssql/server:2019-latest
```

SQL Server Demo

SQL Server Docker Container

Old School SQL

```
dotnet add package System.Data.SqlClient
```

```
var cmdText = "SELECT * FROM Animals";
using var connection = new SqlConnection(connectionString);
connection.Open();
using var command = new SqlCommand(cmdText, connection);
using var reader = command.ExecuteReader();
while (reader.Read())
{
    ...
}
```

https://twitter.com/overflow_meme/status/1223835574848630784

IDisposable

IDisposable

```
var resource = new Resource();
try
{
    ...
}
finally
{
    if (resource != null)
    {
        resource.Dispose();
    }
}
```

IDisposable II

```
var resource = new Resource();
try
{
    ...
}
finally
{
    resource?.Dispose();
}
```

IDisposable III

```
using (var resource = new Resource())
{
    ...
}
```

IDisposable IV

```
using var resource = new Resource();

...
```

SQL Injection

SQL Injection

A **SQL injection** attack consists of insertion or “injection” of a SQL query via the input data from the client to the application.

A successful SQL injection exploit can:

- read sensitive data from the database,
- modify database data (Insert/Update/Delete),
- execute administration operations on the database (such as shutdown the DBMS),
- or worse

https://owasp.org/www-community/attacks/SQL_Injection

Secrets

Secrets

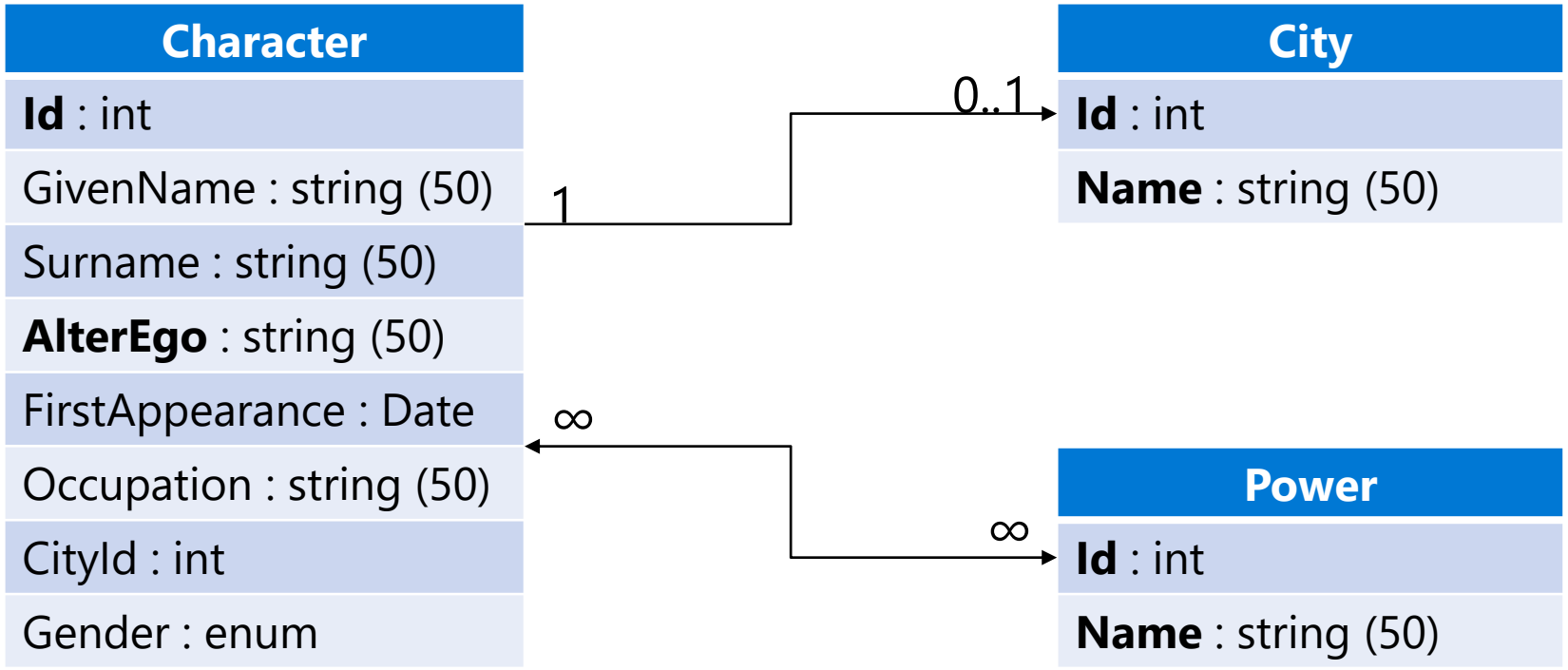
```
dotnet user-secrets init
dotnet user-secrets set "ConnectionStrings:ConnectionString" "..."
dotnet add package Microsoft.Extensions.Configuration.UserSecrets
```

Secrets

```
using Microsoft.Extensions.Configuration;

...

var configuration = new ConfigurationBuilder()
    .AddUserSecrets<Program>()
    .Build();

var connectionString = configuration.GetConnectionString("ConnectionString");
```

Entity Framework Core

Model

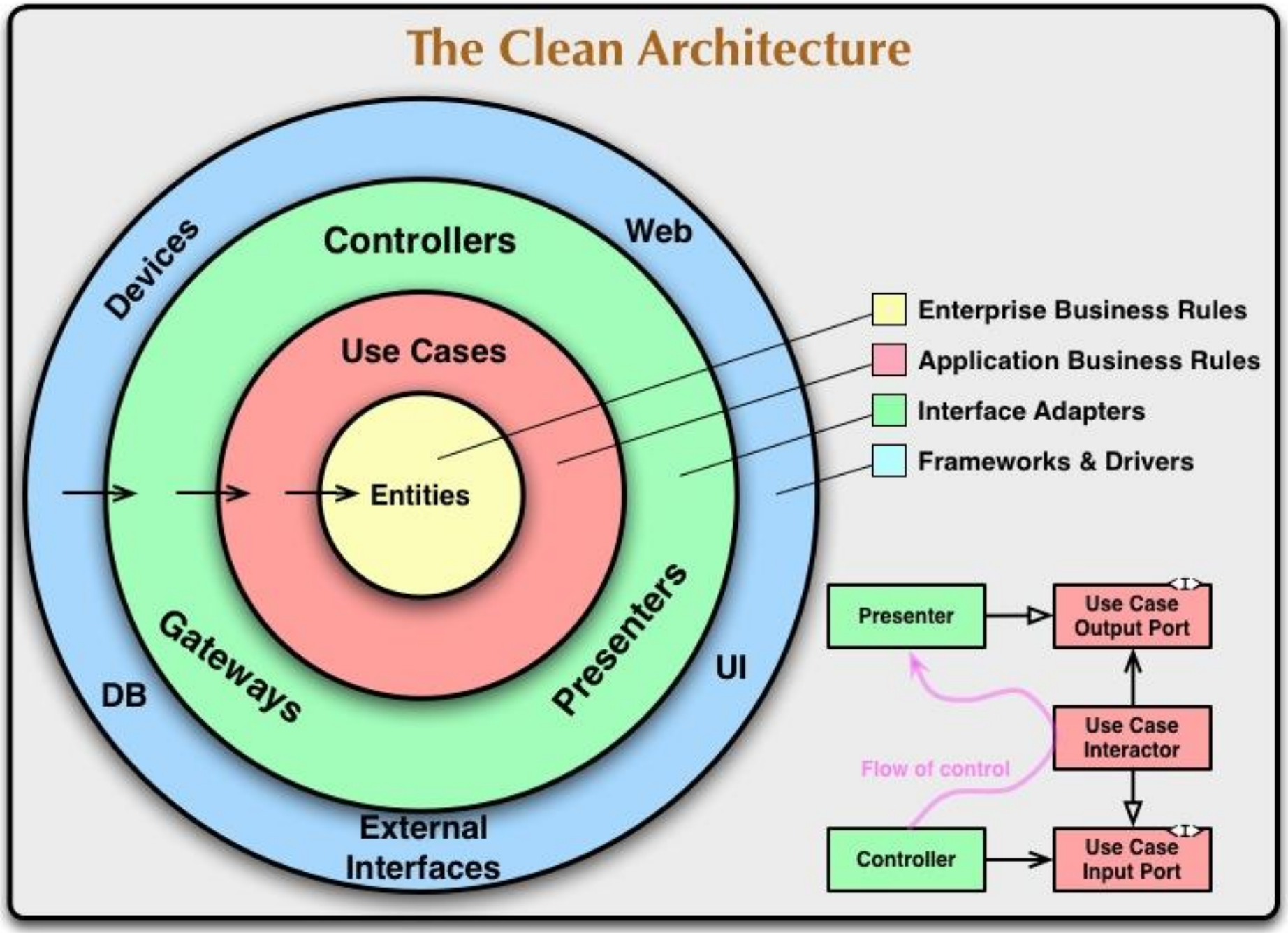

Character
Id : int
GivenName : string (50)
Surname : string (50)
AlterEgo : string (50)
FirstAppearance : Date
Occupation : string (50)
CityId : int
Gender : enum
City
Id : int
Name : string (50)
Power
Id : int
Name : string (50)
1
0..1
∞
∞

Entity Framework Core

```
dotnet tool install --global dotnet-ef
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet ef migrations add InitialCreate
dotnet ef database update
```

https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/dotnet
https://docs.microsoft.com/en-us/ef/core/modeling/

Onion Architecture


The Clean Architecture
Devices
Web
Controllers
Use Cases
Entities
Gateways
Presenters
DB
UI
External
Interfaces
Enterprise Business Rules
Application Business Rules
Interface Adapters
Frameworks & Drivers
Presenter
Use Case
Output Port
Use Case
Interactor
Flow of control
Controller
Use Case
Input Port

Lazy Loading

https://docs.microsoft.com/en-us/ef/core/querying/related-data/lazy

Lazy Loading

```
dotnet add package Microsoft.EntityFrameworkCore.Proxies

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    => optionsBuilder.UseLazyLoadingProxies()
                     .UseSqlServer(...);
```

https://docs.microsoft.com/en-us/ef/core/querying/related-data/

Thank you