ADO.NET

NuGet: system.data.sqlclient v4.5.1 system.data.Common v4.3.0

```
using System.Data;
using System.Data.SqlClient;
```

**********CODICE TABELE**********

```
CREATE TABLE [dbo].[Giocatori] (
    [IdGiocatore] INT           IDENTITY (1, 1) NOT NULL,
    [Nome]        NVARCHAR (50) NOT NULL,
    [Cognome]     NVARCHAR (50) NOT NULL,
    [DataNascita] DATE          NOT NULL,
    [Nickname]    NVARCHAR (50) NOT NULL,
    [Livello]     INT           NULL,
    PRIMARY KEY CLUSTERED ([IdGiocatore] ASC),
    CHECK ([Livello]<=(6)),
    CHECK ([Livello]>=(0))
);


CREATE TABLE [dbo].[Partite] (
    [IdPartita] INT           IDENTITY (1, 1) NOT NULL,
    [Tipo]      NVARCHAR (50) NOT NULL,
    [Campo]     INT           NOT NULL,
    [OraInizio] DATETIME      NOT NULL,
    [OraFine]   DATETIME      NOT NULL,
    [Risultato] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([IdPartita] ASC)
);


CREATE TABLE [dbo].[Prenotazioni] (
    [IdPartita]   INT NOT NULL,
    [IdGiocatore] INT NOT NULL,
    CONSTRAINT [PK_Prenotazioni] PRIMARY KEY CLUSTERED ([IdPartita] ASC, [IdGiocatore] ASC),
    CONSTRAINT [FK_Prenotazioni_Giocatori] FOREIGN KEY ([IdGiocatore]) REFERENCES [dbo].[Giocatori]
([IdGiocatore]),
    CONSTRAINT [FK_Prenotazioni_Partite] FOREIGN KEY ([IdPartita]) REFERENCES [dbo].[Partite]
([IdPartita])
);
```

**********PROGRAM.CS**********

```
Console.WriteLine("              **************  BUONGIORNO!  ***************");
            Console.WriteLine("       Benvenuto nel nostro gestionale del campo da basket di Pegli 2");
            Console.WriteLine("\n\nPremi invio per iniziare\n");
            Console.ReadLine();
            TorreControllo Controllo = new TorreControllo();
            string digit = null;
            do
            {
                Console.WriteLine("\nChe azione vuoi eseguire?\n" +
                    " p -> Aggiungi nuova Partita\n" +
                    " g -> Aggiungi nuovo Giocatore\n" +
                    " c -> Cerca Partita per data\n" +
                    " m -> Cerca giocatori per età\n" +
                    " lg -> Lista di giocatori nel database\n" +
                    " pp -> Partite in programma\n" +
                    " del -> Cancella un giocatore\n" +
                    " exit -> (TERMINA)\n");
                digit = Console.ReadLine();
```

```
                switch (digit)
                {
                    case "p":
                    case "P":
                        {
                            Controllo.IscrizionePartita();
                            break;
                        }
                    case "g":
                    case "G":
                        {
                            Controllo.IscrizioneGiocatore();
                            break;
                        }
                }
            }
            while (digit != "exit");
        }--> chiusura metodo static void Main(string[] args)
```

**********GIOCATORE.CS**********

```
class Giocatore
    {
        public int idGiocatore { get; }
        public string nome { get; set; }
        public string cognome { get; set; }
        public DateTime dataNascita { get; set; }
        public string nickname { get; set; }
        public int livello { get; set; }

        public Giocatore(string nome, string cognome, DateTime dataNascita, string nickname, int
livello)
        {
            this.nome = nome;
            this.cognome = cognome;
            this.dataNascita = dataNascita;
            this.nickname = nickname;
            this.livello = livello;
        }
    }
```

**********TORRECONTROLLO.CS**********

```
namespace Ado_Basket
{

    static class Stringa // in una classe statica avulsa dal resto, così la puoi richiamare quando
                         e dove vuoi
    {
        public static string CONN_STRING = @"Data Source=(localdb)\MSSQLLocalDB;
                                    Initial Catalog=BasketStation;Integrated
                         Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;
                                    ApplicationIntent=ReadWrite;MultiSubnetFailover=False";
    }

    class TorreControllo
    {
        //  Q U E R Y  //
        //Stampami le partite di quel giorno
        private static string STAMPA_PARTITE_IN_DATA = @"
            SELECT *
```

```
            FROM dbo.Partite as P
            WHERE  P.OraInizio BETWEEN @Data AND DATEADD(day,1,@Data)
         ";
        //Aggiungi una partita nel database
         //PER INSERIRE I VALORI ANCHE NELLA TABELLA DI CONGIUNZIONE PROBAB SERVE UN'ALTRA QUERY CHE USI
LO SCOPE IDENTITY O ALTRO COSI DA PIGLIARSI LE ULTIME PK
        private static string AGGIUNGI_PARTITA = @"
            INSERT INTO dbo.Partite (Tipo,Campo,OraInizio,OraFine,Risultato)
            VALUES (@Tipo,@Campo,@DataIn,@DataF,@Ris)
            SELECT SCOPE_IDENTITY()
         ";
        // INSERT INTO dbo.Prenotazioni (IdPartita,IdGiocatore) VALUES (@Tipo,@Campo,@DataIn,@DataF,
@Ris);  //capire come inserire valori in più tabelle
        //Aggiungi un giocatore al database
        private static string ISCRIVI_GIOCATORI = @"
            INSERT INTO Giocatori(Nome,Cognome,DataNascita,Nickname,Livello)
            values (@Nome,@Cognome,@DataNascita,@Nickname,@Livello)
         ";

        //Guardami se quel giorno a quell'ora il campo è libero
        private static string TROVA_PARTITA_DATA_ORA = @"
            SELECT *
            FROM dbo.Partite as P
            WHERE  P.OraInizio BETWEEN @Data AND @Data
         ";

        //Aggiungimi nella tabella di congiunzione id partita e giocatori
        public static string RIEMPI_TABELLA_UNIONE = @"
            INSERT INTO dbo.Prenotazioni (IdGiocatore,IdPartita)
            values (@idGiocatore,@idPartita)
         ";

        //Calcolami il livello medio dei giocatori di età inferiore a..
        private static string TROVA_LIVELLO_MEDIO = @"
            SELECT AVG(Livello) as media
            FROM dbo.Giocatori
            where DATEDIFF(year, DataNascita, SYSDATETIME()) < @Anni
         ";

        //Dammi la lista di giocatori
        private static string LISTA_GIOCATORI = @"
            SELECT * FROM dbo.Giocatori ORDER BY Livello desc
         ";

        //Dammi la lista delle partite in programma
        private static string LISTA_PARTITE = @"
            SELECT *
            FROM dbo.Partite
            WHERE OraInizio > GETDATE()
         ";

//  M E T O D I //
        // 1 //
        public void PartiteInData()
        {
            Console.WriteLine("Inserisci la data della partita:");
            DateTime data;
            data = DateTime.Parse(Console.ReadLine());
            using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING))
            {
                conn.Open();
                using (SqlCommand cmd = new SqlCommand(STAMPA_PARTITE_IN_DATA, conn))
                {
                    cmd.Parameters.AddWithValue("@Data", data);
```

```
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        Console.WriteLine("\nID: " + reader["IdPartita"]);
                        Console.WriteLine("\nTipo: " + reader["Tipo"]);
                        Console.WriteLine("\nNumero Campo: " + reader["Campo"]);
                        Console.WriteLine("\nOra Inizio: " + reader["OraInizio"]);
                        Console.WriteLine("\nOra Fine: " + reader["OraFine"]);
                        Console.WriteLine("\nRisultato partita: " + reader["Risultato"]);
                    }
                }
            }
        }

// 2 // non ritorno un oggetto partita(della classe) ma infilo i dati direttamente nel database
        public void IscrizionePartita()
        {
            try // ECCO COME PIAZZARE UNA ECCEZIONE
            {
                string tipo, oraInizio, oraFine, dataInizio, dataFine;
                int campo;
                int cont = 4;
                int[] giocatori = new int[4];
                Console.WriteLine("Tipo di partita: 1vs1/2vs2");
                tipo = Console.ReadLine();
                Console.WriteLine("\nChe campo Desiderate? (1-2-3-4)");
                campo = int.Parse(Console.ReadLine()); //Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("\n che giorno?");
                dataInizio = Console.ReadLine();
                Console.WriteLine("\nA che ora volete iniziare?");
                oraInizio = Console.ReadLine();
                Console.WriteLine("\nA che ora volete finire?");
                oraFine = Console.ReadLine();
                Console.WriteLine("Inserisci il risultato:");
                String risultato = Console.ReadLine();
                dataFine = dataInizio + " " + oraFine;
                dataInizio = dataInizio + " " + oraInizio;
                DateTime data1 = DateTime.Parse(dataInizio);
                DateTime data2 = DateTime.Parse(dataFine);
                if (tipo == "1vs1")
                {
                    cont = 2;
                    for (int i = 0; i < cont; i++)
                    {
                        Console.WriteLine("\nID giocatore" + i);
                        giocatori[i] = int.Parse(Console.ReadLine());
                        //int player = giocatori[i];
                    }
                }
                else
                {
                    for (int i = 0; i < cont; i++)
                    {
                        Console.WriteLine("\nID giocatore" + (i + 1));
                        giocatori[i] = int.Parse(Console.ReadLine());
                    }
                }
                int id = AddPartita(tipo, campo, data1, data2, risultato); // gli passi i parametri di
sopra
                //string agg = aggiungi();
                Console.WriteLine("partita inserita con id:" + id);
                Iscrizione(id, giocatori, cont);
```

```
            }
            catch(FormatException ex)
            {
                Console.WriteLine("ECCEZIONE RILEVATA: " + ex);
                Console.WriteLine("\nINSERISCI LA DATA IN FORMATO GG-MM-AA E L'ORA IN FORMATO HH:MM
\n");
                IscrizionePartita();
            }

        }
        // 3 //
        public int AddPartita(string tipo, int campo, DateTime dataInizio, DateTime dataFine, string
risultato)
        {
            using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING))
            {
                conn.Open();

                using (SqlCommand cmd = new SqlCommand(AGGIUNGI_PARTITA, conn))
                {

                    cmd.Parameters.AddWithValue("@Tipo", tipo);
                    cmd.Parameters.AddWithValue("@Campo", campo);
                    cmd.Parameters.AddWithValue("@DataIn", dataInizio);
                    cmd.Parameters.AddWithValue("@DataF", dataFine);
                    cmd.Parameters.AddWithValue("@Ris", risultato);
                    decimal result = /*(decimal)*/Convert.ToDecimal(cmd.ExecuteScalar()); // puoi fare
sia il cast che usare il Covert.ToDecimal
                    return (int)result;
                }
            }
        }

        // 3 plus //
        public void Iscrizione(int id, int[] idGiocatore, int cont)
        {
            using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING))
            {
                conn.Open();
                for (int i = 0; i < cont; i++)
                {
                    using (SqlCommand cmd = new SqlCommand(RIEMPI_TABELLA_UNIONE, conn))
                    {
                        cmd.Parameters.AddWithValue("@idPartita", id);
                        cmd.Parameters.AddWithValue("@idGiocatore", idGiocatore[i]);
                        Console.WriteLine("Numero righe inserite: " + cmd.ExecuteNonQuery());
                    }
                }
            }
        }
// 5 //
        public decimal ValoreMedG()
        {
            Console.WriteLine("Inserisci l'età massima");
            int eta;
            eta = Convert.ToInt32(Console.ReadLine());
            decimal media = 0;
            using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING))
            {
                conn.Open();
                using (SqlCommand cmd = new SqlCommand(TROVA_LIVELLO_MEDIO, conn))
                {
                    cmd.Parameters.AddWithValue("@Anni", eta);
                    using (SqlDataReader reader = cmd.ExecuteReader())
```

```
                {
                    while (reader.Read())
                    {
                        media = Convert.ToDecimal( reader["media"]); // Convert.ToInt32
                    }
                }
            }
        }
        Console.WriteLine("\nLa media dei giocatori sotto i " + eta + " è: " + media);
        return media;
    }
    // 6 //
    public void ListaGiocatori()
    {
        using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING)) //ti connetti al
database
        {
            conn.Open(); //apri la connessione
            using (SqlCommand cmd = new SqlCommand(LISTA_GIOCATORI, conn)) //attivi la query
            {
                //cmd.Parameters.AddWithValue("@Data", data);
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        Console.WriteLine(reader["Nome"].ToString()
                        + " " +
                        reader["Cognome"].ToString()
                        + " " + " ha attualmente un livello pari a " +
                        reader["Livello"].ToString());
                    }
                }

            }

        }
    }
    // 7 //
    public void PartiteInProgramma()
    {
        using (SqlConnection conn = new SqlConnection(Stringa.CONN_STRING)) //ti connetti al
database
        {
            conn.Open(); //apri la connessione
            using (SqlCommand cmd = new SqlCommand(LISTA_PARTITE, conn)) //attivi la query
            {
                //cmd.Parameters.AddWithValue("@Data", data);
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                    Console.WriteLine("\nPartite in programma:\n");
                    while (reader.Read())
                    {
                        Console.WriteLine(reader["IdPartita"].ToString()
                        + " " +
                        reader["Tipo"].ToString()
                        + " " +
                        reader["Campo"].ToString()
                        + " " +
                        reader["OraInizio"].ToString());
                        //+ " " +
                        //reader["OraFine\n"].ToString());
                    }
                    Console.WriteLine("\nPress any key to continue");
                    Console.ReadLine();
```

```
                    }
                }
            }
        }
```

DELETE QUERY : `DELETE FROM dbo.Giocatori WHERE IdGiocatore = 4004`

MVC

NuGet: Microsoft.VisualStudio.Web.CodeGeneration.Design v2.1.1
Microsoft.EntityFrameworkCore.Tools v2.1.4

```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MvcProvaesame.Model;
using MvcProvaesame.Model.ViewModel;
```

**********CODICE TABELE**********

```
CREATE TABLE [dbo].[Autori] (
    [Id]       INT            IDENTITY (1, 1) NOT NULL,
    [Email]    NVARCHAR (MAX) NOT NULL,
    [Nome]     NVARCHAR (MAX) NOT NULL,
    [Skill]    NVARCHAR (MAX) NOT NULL,
    [Telefono] NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_Autori] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Presentazioni] (
    [Id]      INT            IDENTITY (1, 1) NOT NULL,
    [Fine]    DATETIME2 (7)  NOT NULL,
    [Inizio]  DATETIME2 (7)  NOT NULL,
    [Livello] NVARCHAR (MAX) NOT NULL,
    [Titolo]  NVARCHAR (MAX) NOT NULL,
    CONSTRAINT [PK_Presentazioni] PRIMARY KEY CLUSTERED ([Id] ASC)
);

CREATE TABLE [dbo].[Registrazioni] (
    [Autore]        INT NOT NULL,
    [Presentazione] INT NOT NULL,
    CONSTRAINT [PK_Registrazioni] PRIMARY KEY CLUSTERED ([Autore] ASC, [Presentazione] ASC),
    CONSTRAINT [FK_Registrazioni_Autori_Autore] FOREIGN KEY ([Autore]) REFERENCES [dbo].[Autori] ([Id])
ON DELETE CASCADE,
    CONSTRAINT [FK_Registrazioni_Presentazioni_Presentazione] FOREIGN KEY ([Presentazione]) REFERENCES
[dbo].[Presentazioni] ([Id]) ON DELETE CASCADE
);


GO
CREATE NONCLUSTERED INDEX [IX_Registrazioni_Presentazione]
    ON [dbo].[Registrazioni]([Presentazione] ASC);
```

**********SVOLGIMENTO**********

- ApplicazioneWeb ASP.NET Core ---> asp.net core 2.1 VUOTO

- Tasto dx sul progetto, Apri in esplora risorse , tasto dx + SHIFT, Apri finestra di comando qui, incolli il codice qui sotto:

dotnet ef dbcontext scaffold "Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Presentazioni;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False;" Microsoft.EntityFrameworkCore.SqlServer -d -o Model -c MvcContext --force

- Se ti crea la cartella bene, se non te la crea installare il nuget EFTools.

- Program.cs da non toccare, Startup da configurare con il seguente codice:

```
 public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit
https://go.microsoft.com/fwlink/?LinkID=398940
        public IConfiguration Configuration { get; }

        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddDbContext<AutoriContext>(opts => opts.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
            services.AddMvc();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseStatusCodePages();
            app.UseStaticFiles();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
```

- Creare la cartella "Controllers" e aggiungere un nuovo controller vuoto chiamato "Home". Aprirlo, tasto destro sul metodo Index e "crea visualizzazione", il nome delle view sempre Index e crei. Verrà generata una cartella che da adesso conterrà le tue view.

- dopo aver fatto questo creare il file appsettings.json in cui sarà contenuta la stringa di connessione del progetto, togli tutto quello che c'è dentro e inserisci il seguente codice:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=Presentazioni;Integrated
Security=True;Connect Timeout=
30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"
```

```
  }
}
```

- Tasto dx sulla cartella View --> aggiungi nuovo elemento --> pagina razor che chiaerai _ViewImports

```
@using MvcPersonello.Controllers;
@using MvcPersonello.Model;
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

- Tasto dx sulla cartella View --> aggiungi nuovo elemento --> pagina razor che chiamerai _Layout
Nota bene che deve essere inserito in una cartella dentro le view che si chiama Shared.
Copiare ed incollare il codice che trovi al suo interno da un altro progetto mvc

Eliminare il seguente codice all'interno del file copiato

```
  <partial name="_CookieConsentPartial" />
```

INCOLLARE IL SEGUENTE CODICE PER INSTALLARE BOOTSTRAP NEL PROGETTO

<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>

INCOLLARE IL SEGUENTE CODICE PER INSTALLARE JQUERY NEL PROGETTO

Sito jquery --> download -->il primo, versione 3.1.1 --> copi tutto e lo incolli in un file txt --> cambi l'estensione del file appena creato in .js --> incolli questo file in wwwroot in esplora soluzioni e lo trascini nell'head del _Layout.

- Ora creiamo il _ViewStart, che anch'essa sarà una pagina razor.

```
@{
    Layout = "_Layout";
}
```

- Creiamo i controller autogenerati, se fallisce l'autogenerazione installare il nuget codegeneration.
Tasto dx sulla cartella controllers, aggiungi, controller, mvc con ef, selezioni modello e contesto e via. Ti crea automaticamente anche le CRUD view.

- completare l'home page inserendo:

```
link rel="stylesheet" href="~/Style.css" /> nell'head
```

Poi nell'index dell'homecontroller posizionare un'immagine di benvenuto:

```
@{
    ViewData["Title"] = "Home Page";
}
```

```
<div style="text-align:center; padding:30px">
    <img src="~/HomePage.png" style="margin-top:30px"/>
</div>
```

- Aggiungere il PresentazioniViewModel

```
    public class PresentazioneViewModel
    {
        public int Id { get; set; }
        public DateTime Fine { get; set; }
        public DateTime Inizio { get; set; }
        public string Livello { get; set; }
        public string Titolo { get; set; }
        public int IdAutore { get; set; }

        public List<Autori> autori = new List<Autori>();
    }
```

- Implementiamo il codice nel PresentazionisController:

```
        // GET: Presentazionis/Create
        public IActionResult Create()
        {
            PresentazioniViewModel pres = new PresentazioniViewModel();
            var auto = _context.Autori.ToList();
            foreach(var i in auto)
            {
                pres.autori.Add(i);
            }
            return View(pres);
        }

        // POST: Presentazionis/Create
        // To protect from overposting attacks, please enable the specific properties you want to bind
to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create(PresentazioniViewModel presentazioni)
        {
            if (ModelState.IsValid)
            {
                Presentazioni presentazione = new Presentazioni()
                {
                    Inizio = presentazioni.Inizio,
                    Fine = presentazioni.Fine,
                    Livello = presentazioni.Livello,
                    Titolo = presentazioni.Titolo
                };
                _context.Add(presentazione);
                await _context.SaveChangesAsync();
                int idPresentazione = presentazione.Id;
                Registrazioni registra = new Registrazioni()
                {
                    Autore = presentazioni.IdAutore, // recuperato dalla drop down list
                    Presentazione = idPresentazione
                };
                _context.Registrazioni.Add(registra);
                await _context.SaveChangesAsync();
                return RedirectToAction(nameof(Index));
            }
            return View(presentazioni);
        }
```

- Adesso andiamo nella view Create di Presentazionis, cambiare il modello in cima alla pagina mettendo

```
@model MvcProvaesame.Model.ViewModel.PresentazioniViewModel
```

e modificando il codice successivo con:

```
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Fine" class="control-label"></label>
                <input asp-for="Fine" class="form-control" />
                <span asp-validation-for="Fine" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Inizio" class="control-label"></label>
                <input asp-for="Inizio" class="form-control" />
                <span asp-validation-for="Inizio" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Livello" class="control-label"></label>
                <input asp-for="Livello" class="form-control" />
                <span asp-validation-for="Livello" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Titolo" class="control-label"></label>
                <input asp-for="Titolo" class="form-control" />
                <span asp-validation-for="Titolo" class="text-danger"></span>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
            <div>
                <label asp-for="autori" class="control-label"></label>
                <select asp-for="IdAutore" asp-items="@(new SelectList
                                                        (Model.autori,"Id","Nome"))">
                    <option>Please select one</option>
                </select>
                <input type="submit" />
            </div>
        </form>
    </div>
</div>
```

PARTE MOLTO IMPORTANTE E FONDAMENTALE UNIRE LE VIEW APPENA CREATO CON L'HOME CONTROLLER GRAZIE AI TAG HELPERS.

- Validazione email e telefono:

```
[EmailAddress(ErrorMessage = "Invalid Email Address")]
public string EmailAddress { get; set; }
[RegularExpression(@"^([0]|\+91[\-\s]?)?[789]\d{9}$", ErrorMessage = "Invalid ")]
public string PhoneNumber { get; set; }
```

API+ANGULAR

NuGet: Microsoft.VisualStudio.Web.CodeGeneration.Design v2.1.1

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
```

```
using Microsoft.EntityFrameworkCore;
using ApiOspedale.Model;
using Microsoft.AspNetCore.Cors;
```

**********CODICE TABELE**********

```
CREATE TABLE [dbo].[Medici] (
    [IdMedico]       INT           IDENTITY (1, 1) NOT NULL,
    [Nome]           NVARCHAR (50) NOT NULL,
    [Cognome]        NVARCHAR (50) NOT NULL,
    [DataNascita]    DATE          NOT NULL,
    [DataAssunzione] DATE          NOT NULL,
    [DeptId]         INT           NULL,
    PRIMARY KEY CLUSTERED ([IdMedico] ASC),
    CONSTRAINT [FK_Medici_Dipartimenti] FOREIGN KEY ([DeptId]) REFERENCES [dbo].[Dipartimenti]
([IdDipartimento])
);

CREATE TABLE [dbo].[Dipartimenti] (
    [IdDipartimento] INT           IDENTITY (1, 1) NOT NULL,
    [Nome]           NVARCHAR (50) NOT NULL,
    [Area]           NVARCHAR (50) NOT NULL,
    [PrimarioId]     INT           NOT NULL,
    CONSTRAINT [PK_Dipartimenti] PRIMARY KEY CLUSTERED ([IdDipartimento] ASC),
    CONSTRAINT [FK_Dipartimenti_Medici] FOREIGN KEY ([PrimarioId]) REFERENCES [dbo].[Medici]
([IdMedico])
);
```

**********SVOLGIMENTO**********

- ApplicazioneWeb ASP.NET Core ---> asp.net core 2.1 API

- Tasto dx sul progetto, Apri in esplora risorse , tasto dx + SHIFT, Apri finestra di comando qui, incolli il codice qui sotto:

dotnet ef dbcontext scaffold "Data Source=(localdb)\MSSQLLocalDB;Initial Catalog=Presentazioni;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False;" Microsoft.EntityFrameworkCore.SqlServer -d -o Model -c MvcContext --force

- Se ti crea la cartella bene, se non te la crea installare il nuget EFTools.

- Modificare l'appsettings.json immettendo:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=Hospital;Integrated
Security=True;Connect Timeout=
30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"
  }
}
```

- Modificare la Startup inserendo il contesto, i servizi e la policy cors:

```
public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
```

```
        {
            services.AddDbContext<HospitalContext>(opts => opts.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));

            services.AddTransient<Repository, Repository>();

            services.AddCors(o => o.AddPolicy("MyPolicyCORS", builder =>
            {
                builder.AllowAnyOrigin()
                       .AllowAnyMethod()
                       .AllowAnyHeader();
            }));

            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseHsts();
            }
            app.UseCors("MyPolicyCORS");

            app.UseHttpsRedirection();
            app.UseMvc();
        }
    }
```

- Aggiungere il viewmodel:

```
public class MediciVModel
    {
        public int IdMedico { get; set; }
        public string Nome { get; set; }
        public string Cognome { get; set; }
        public DateTime DataNascita { get; set; }
        public DateTime DataAssunzione { get; set; }
        public string Dipartimento { get; set; }

        public MediciVModel()
        {

        }

        //tutta sta merda per questo, assegnare al deptid del medico il nome del dipartimento
        //così da poterlo usare per risolvere il primo punto dell'esame
        public MediciVModel(Medici medico)
        {
            IdMedico = medico.IdMedico;
            Nome = medico.Nome;
            Cognome = medico.Cognome;
            DataNascita = medico.DataNascita;
            DataAssunzione = medico.DataAssunzione;
            Dipartimento = medico.Dept.Nome;
        }
    }
```

- Scrivere i metodi dentro la repository nei model:

```
    public class Repository
    {
        private HospitalContext ctx;

        public Repository(HospitalContext context)
        {
            this.ctx = context;
        }

        //ritorna la lista di tutti i dipartimenti
        public List<Dipartimenti> GetAllDipartimenti()
        {
            return ctx.Dipartimenti.ToList();
        }

        //ritorna la lista di tutti i medici
        public List<Medici> GetAllMedici()
        {
            return ctx.Medici.ToList();
        }

        //ritorna la lista di medici + si porta dietro tutte le proprietà della classe Dipartimento
grazie a "Dept"
        public List<Medici> GetMediciPlus()
        {
            return ctx.Medici.Include(d => d.Dept).ToList();
        }

        //ritorna una lista di medici in base al numero dato e all'ID del dipartimento.
        //QUERY: deptId = all'id del dipartimento che si inserice,medici ordinati per datadiassunzione,
prende gli n dati utili indicati,
        // e include nel risultato l'oggetto Dipartimento Dept, che si porta dietro tutte le proprietà
utili.
        public List<Medici> MediciPiuAnziani (int n, int dipartimento)
        {
            return ctx.Medici.Where(d => d.DeptId == dipartimento).OrderByDescending(x =>
x.DataAssunzione).Take(n).Include(x => x.Dept).ToList();
        }
```

- Autogenerare i controllers con webapi+EF. Se non li fa scaricare il nuget codegenerator

```
    [EnableCors("MyPolicyCORS")]
    [Route("api/dipartimenti")]
    [ApiController]
    public class DipartimentisController : ControllerBase
    {
        private Repository repo;

        public DipartimentisController(Repository r)
        {
            this.repo = r;
        }

        // localhost/api/dipartimenti
        [HttpGet]
        public IEnumerable<Dipartimenti> GetDipartimenti()
        {
            return repo.GetAllDipartimenti();
        }
```

```
    [EnableCors("MyPolicyCORS")]
    [Route("api/medici")]
    [ApiController]
    public class MedicisController : Controller
    {
        private Repository repo;

        public MedicisController(Repository r)
        {
            this.repo = r;
        }

        ////localhost/api/medici
        //[HttpGet]
        //public IActionResult GetMedici()
        //{
        //    return repo.GetAllMedici().Select(m=>m.ToDTO());
        //}

        [HttpGet]
        public List<MediciVModel> GetMedici (int numero, int dipartimento)
        {
            if(numero == 0)
            {
                //se n = 0 allora te ne crea uno nuovo
                return repo.GetMediciPlus().Select(m => new MediciVModel(m)).ToList()/*.Select(m =>
m.ToDTO())*/;
            }
            else
            {
                //li seleziona e la menata del select è solo per far comparire il nome del dipartimento
di fianco al medico
                return repo.MediciPiuAnziani(numero, dipartimento).Select(m => new MediciVModel
(m)).ToList();
            }
        }
```

- PARTE ANGULAR

Aprire la cartella vuota, tasto dx+SHIFT, aprire finestra di comando e ng new nomeapp.
Aggiungere subito le immagini dentro la cartella "img".
Dentro il file padre Style.css incollare:

```
body {
  background-image: url(/img/sf.jpg);
  background-size: cover;
  border-bottom-left-radius: 20px;
}

th{
  font-size: 18px;
  color: black;
  background-color: snow
}
```

- Cambiare l'Index.html con questo

```
<!doctype html>
<html lang="en">
<head>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-
```

```
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">

  <meta charset="utf-8">
  <title>Ospedale website</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

- Cambiare l'appcomponent.html con il LAYOUT del sito

```
<div style="text-align:center">
  <img src="/img/WELCOME.png" style="widows: 400px;height:190px;margin-top: 20px;border-style: outset;border-color: crimson">
</div>
<div style="text-align: center">
   <span class="badge badge-secondary" style="font-size:35px;margin-top:30px;margin-left:30px;">Premi un bottone per visualizzare le
informazioni richieste</span>
  </div>
<br>
    <div style="text-align:center">
      <a class="zio" routerLinkActive='active' [routerLink]="['/mioprofilo']">
        <button class="button button-clear" style="
          cursor: pointer;
          text-align: center;
          font: bold 15px Arial, Helvetica, sans-serif;
          color: rgb(0, 0, 0);
          padding: 10px 20px;
          border: outset 3px #0076a3;
          background: #77d1f1;
          margin-top: 30px">HOME</button></a>
                      
      <a class="zio" routerLinkActive='active' [routerLink]="['/medici']">
        <button class="button button-clear" style="
          cursor: pointer;
          text-align: center;
          font: bold 15px Arial, Helvetica, sans-serif;
          color: rgb(0, 0, 0);
          padding: 10px 20px;
          border: outset 3px #0076a3;
          background: #77d1f1;
          margin-top: 30px">Mostra i medici</button></a>
                      
      <a class="zio" routerLinkActive='active' [routerLink]="['/medicianziani']">
        <button class="button button-clear" style="
          cursor: pointer;
          text-align: center;
          font: bold 15px Arial, Helvetica, sans-serif;
          color: rgb(0, 0, 0);
          padding: 10px 20px;
          border: outset 3px #0076a3;
          background: #77d1f1;
          margin-top: 30px">Mostra anzianità</button></a>
    </div>

<router-outlet></router-outlet>
```

- Nell'appcomponent.ts eliminare la variabile del titolo

- Creare il welcomecomponent che al suo interno avrà solamente qualche scritta base per l'home page

```
<b>
<div style="font-size:21px;font-style:italic;color:firebrick;margin: 70px">
  <ul>
    In questa pagina potrai:
    <li>
      visualizzare la lista dei medici con relativo dipartimento associato
    </li>
    <li>
      cercare gli n medici con più anzianità per un dato dipartimento
    </li>
  </ul>
</div>
</b>
```

- Passaggio fondamentale creare i path per i collegamenti tra le pagine

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';
import { WelcomeComponent } from './welcome/welcome.component';
import { HttpClientModule } from '@angular/common/http';
import { MediciComponent } from './medici/medici.component';
import { MediciModule } from './medici/medici.module';

@NgModule({
  declarations: [
    AppComponent,
    WelcomeComponent,
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    RouterModule.forRoot([
      {path: 'home', component: WelcomeComponent},
      {path: '', redirectTo:'home',pathMatch:'full'},
      {path:'**', redirectTo:'home', pathMatch:'full'}
    ]),
    MediciModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

- Creare il componente su cui si fa l'ngfor per creare una tabella:

MEDICISERVICE.TS

```
@Injectable({
  providedIn: 'root',
})
export class MediciService {

  private mediciUrl="https://localhost:44365/api/medici";
  private dipartimentiURL="https://localhost:44365/api/dipartimenti"
  constructor(private http: HttpClient) { }

  getMedici(): Observable<Medico[]> {
```

```
   return this.http.get<Medico[]>(this.mediciUrl)
     .pipe(
       tap(data => console.log(JSON.stringify(data)))
     );
  }

  getDipartimenti():Observable<Dipartimento[]>
  {
    return this.http.get<Dipartimento[]>(this.dipartimentiURL)
    .pipe(
      tap(data => console.log(JSON.stringify(data)))
    );
  }

  getImpiegatiAnziani(num:number,dipartimento:number):Observable<Medico[]>
  {
    return this.http.get<Medico[]>(this.mediciUrl+"?numero="+num+"&dipartimento="+dipartimento)
    .pipe(
      tap(data => console.log(JSON.stringify(data)))
    );
  }
}
```

MEDICIMODULE.TS

```
@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    RouterModule.forChild([
      { path: 'medici', component: MediciComponent },
      { path: 'medicianziani', component: MediciAnziani },
       ])
  ],
  declarations: [MediciComponent,MediciAnziani]
})
export class MediciModule { }
```

MEDICICOMPONENT.TS

```
import { Component, OnInit } from '@angular/core';
import { Medico } from './medico';
import { MediciService } from './medici.service';

@Component({
  selector: 'app-medici',
  templateUrl: './medici.component.html',
  styleUrls: ['./medici.component.css']
})
export class MediciComponent implements OnInit {

  private medici:Medico[];
  errormessage:string;

  constructor(private service:MediciService) { }

  ngOnInit() {
    this.service.getMedici().subscribe(
      medici => {
        this.medici = medici;
      },
      error => this.errormessage = <any>error
```

```
    );
  }

}
```

MEDICICOMPONENT.HTML

```
<table class="table" style="margin-left:35px;width: 900px;position: absolute;left: 520px;margin-top: 80px;border-style: double;border-color: black;">
    <thead style="border-style: double;border-color:black;">
      <tr>
        <th>
          ID
        </th>
        <th>
          NOME
        </th>
        <th>
          COGNOME
        </th>
        <th>
          DATA DI NASCITA
        </th>
        <th>
          ASSUNZIONE
        </th>
        <th>
          ID DIPARTIMENTO
        </th>
      </tr>
    </thead>
    <tbody>
      <tr class="card" *ngFor='let medico of medici'>
        <td >{{medico.idMedico}}</td>
        <td >{{medico.nome}}</td>
        <td >{{medico.cognome}}</td>
        <td >{{medico.dataNascita}}</td>
        <td >{{medico.dataAssunzione}}</td>
        <td >{{medico.dipartimento}}</td>
      </tr>
    </tbody>
```

MEDICO.TS

```
export class Medico{
    idMedico:number;
    nome:string;
    cognome:string;
    dataNascita:Date;
    dataAssunzione:Date;
    dipartimento:string;
}
```

-A QUESTO PUNTO SI AGGIUNGE IL COMPONENTE PER GLI ANZIANI

ANZIANI.TS

```
import { Component, OnInit } from '@angular/core';
import { Dipartimento } from './dipartimento';
import { Medico } from './medico';
import { MediciService } from './medici.service';
```

```
@Component({
  templateUrl: './medicianziani.html',
  styleUrls: ['./medicianziani.css']
})
export class MediciAnziani  implements OnInit {

    private medici:Medico[];
    private dipartimenti:Dipartimento[];
    private _numero:number;
    private _dipartimento:number;
    errorMessage:string;

    constructor(private  service:MediciService) {
       this.numero=2;
       this.dipartimento=null;
    }

    set numero(x:number)
    {
       if(x>0)
       {
          this._numero=x;
          this.onChange(); //richiama qui onchange
       }
    }

    get numero()
    {
       return this._numero;
    }

    set dipartimento(x:number)
    {
      this._dipartimento=x;
      this.onChange();//richiama qui onchange
    }

    get dipartimento()
    {
       return this._dipartimento;
    }

    ngOnInit(): void {
      this.service.getDipartimenti().subscribe(
          dipartimenti => {
            this.dipartimenti = dipartimenti;
          },
          error => this.errorMessage = <any>error
       );
    }

    onChange()
    {
      this.service.getImpiegatiAnziani(this.numero,this.dipartimento).subscribe(
          medici => {
            this.medici = medici;
          },
          error => this.errorMessage = <any>error
       );
    }
}
```

ANZIANI.HTML

```
<div class="table" style="margin-left:35px;width: 900px;position: absolute;left: 520px;margin-top: 80px;border-style: double;border-color: black;">
    <div style="margin-left:35px;margin-top: 20px">
      <span>Numero risultati</span><br/>
      <input type="number" [(ngModel)]=numero  ><br/>
      <select class="form-control" [(ngModel)]=dipartimento>
      <option value=""
      disabled
      selected
      hidden>Seleziona un dipartimento</option>
      <option *ngFor= 'let dipartimento of dipartimenti' [value] = 'dipartimento.idDipartimento' style="margin-right:100px">{{dipartimento.nome}} </option>
      </select>
    </div>

    <b><div class="card" *ngFor='let medico of medici' style="margin-left:35px;margin-top: 20px; margin-bottom: 20px">
      <span >Id: {{medico.idMedico}}</span><br/>
      <span >Nome: {{medico.nome}}</span><br/>
      <span >Nome: {{medico.cognome}}</span><br/>
      <span >Data Nascita: {{medico.dataNascita}}</span><br/>
      <span >Data Assunzione: {{medico.dataAssunzione}}</span><br/>
      <span >Dipartimento: {{medico.dipartimento}}</span><br/>
    </div></b>
```