# C# asztali alkalmazás

## Grafikus alkalmazás

1. WPF Application (.Net)
2. Csomagok hozzáadása a könyvtárhoz

   Jobb gomb a projektre – Manage NuGet Packages…

   - Microsoft.EntityFrameworkCore
   - Microsoft.EntityFrameworkCore.Abstractions
   - Microsoft.EntityFrameworkCore.SqlServer
   - Microsoft.EntityFrameworkCore.Proxies
   - Microsoft.EntityFrameworkCore.Tools
   - Microsoft.EntityFrameworkCore.Relational

3. Adatbázis lemodellezése
   a. Tools – NuGet Package Manager - Package Manager Console
   b. bemásoljuk

      Scaffold-DbContext
      "Server=.\sqlexpress;Database=Atletika1DB;Trusted_Connection=True;"
      Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -ContextDir Context
      -Context AppDbContext

      *Database= adatbázis neve
   c. models mappában elkészítjük az adatbázisnak megfelelően a classokat

```
[Table("Versenyzo")]
public class Versenyzo
{
    [Key]
    public int VersID { get; set; }
    public string Nev { get; set; }
    public bool Neme { get; set; }

    public Versenyzo() //ctor + tab+tab
    {

    }

    public Versenyzo(int versID, string nev, bool neme)
    {                    //jobb klick Quick Actions and Refactorings…
                         //Generate constructor…

        VersID = versID;
        Nev = nev;
        Neme = neme;
    }
}
```

```
[ForeignKey("Helyszin")] - //adatbázis szerinti idegen kulcs meghatározása
public int HelyID { get; set; }
public virtual Helyszin Helyszin { get; set; } //hogy elérhető legyen a tábla
```

## 4. AppDbContext létrehozása

```
public class AppDbContext : DbContext
    {
        public DbSet<Versenyzo> Versenyzok { get; set; }
        public DbSet<Helyszin> Helyszinek { get; set; }
        public DbSet<Eredmenyek> Eredmenyek { get; set; }
```

SQL szerverhez csatlakozás

```
        protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
        {
            base.OnConfiguring(optionsBuilder);

optionsBuilder.UseSqlServer("Server=.\\SQLEXPRESS;Database=Atletika1DB;Trusted_
Connection=True;");
            optionsBuilder.UseLazyLoadingProxies();
        }
```

Modell készítés, táblák közötti kapcsolatok meghatározása

```
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Eredmenyek>()
                        .HasKey(model => new { model.VersID, model.HelyID});
            modelBuilder.Entity<Eredmenyek>()
                        .HasOne(x => x.Versenyzo);
            modelBuilder.Entity<Eredmenyek>()
                        .HasOne(x => x.Helyszin);
        }
```

## 5. Felület kialakítása MainWindow.xaml

```
    <Window.DataContext>
        <model:Eredmenyek/> //melyik modellt jelenítjük meg az űrlapban
    </Window.DataContext>
```

Sorok, oszlopok definiálása

```
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
     </Grid>
```

Elemek felhelyezése

StackPanel, olyan, mint a div

```
        <StackPanel Grid.Row="1" Grid.Column="0"
                    Orientation="Vertical"
                    Margin="10" Grid.ColumnSpan="1">
            <Label x:Name="lblHelyszin" Content="Helyszín" />
            <ComboBox x:Name="cbHelyszin" //lenyíló menü
                ItemsSource="{
                    Binding Mode=TwoWay,
                    Path=HelyID,
                    RelativeSource={RelativeSource FindAncestor,
                    AncestorType={x:Type Window}}}"
```

```
                    DisplayMemberPath="Varos"
                    SelectedValuePath="HelyID"
                    SelectedItem="{Binding Path=Helyszin}"
                    SelectedValue="{Binding Path=HelyID}"
                    Text="Válassz egy Helyszínt!"
                    />
        </StackPanel>


        <StackPanel Grid.Row="2" Grid.Column="0"
                    Orientation="Vertical"
                    Margin="10">
            <Label x:Name="lblVersenyszam" Content="Verszenyszám" />
            <TextBox x:Name="txtVersenyszam"      //textbox
                     Text="{
                       Binding Mode=TwoWay, Path=Vsenyszam}"
                     />
        </StackPanel>


        <StackPanel Grid.Row="3" Grid.Column="0"
                    Orientation="Vertical"
                    VerticalAlignment="Center"
                    Margin="10">
            <Button x:Name="btnSubmit"      //gomb
                    Content="Mentés"
                    Click="OnClick" />
        </StackPanel>
```

`Grid.ColumnSpan="4"` – négy oszlop összevonása

Lenyíló mező feltöltése adatokkal

```
<Window
        Loaded="OnLoad">
```

.cs fájl

```
   public partial class MainWindow : Window
    {
        private AppDbContext _appDbContext;

      }

        private void OnLoad(object sender, RoutedEventArgs e)
        {
            using (_appDbContext = new AppDbContext())
            {
                cbHelyszin.ItemsSource = _appDbContext.Helyszinek.ToList();
                cbNev.ItemsSource = _appDbContext.Versenyzok.ToList();
            }
              }
```

Gombhoz tartozó függvény megírása

```
private void OnClick(object sender, RoutedEventArgs e)
        {
            Eredmenyek data = (Eredmenyek)DataContext;
            try
            {
                using (_appDbContext = new AppDbContext())
                {
```

```
                    Eredmenyek record = _appDbContext.Eredmenyek.Single(x =>
x.VersID == data.VersID && x.HelyID == data.HelyID);
                    record.Vsenyszam = data.Vsenyszam;

                    _appDbContext.SaveChanges();

                    MessageBox.Show("Sikeres módosítás.", "",
MessageBoxButton.OK);

                    data.Vsenyszam = string.Empty;
                    cbHelyszin.SelectedIndex = -1;
                    cbVersenyzo.SelectedIndex = -1;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Nem volt sikeres a módosítás.", "",
MessageBoxButton.OK);
            }

        }
    }
```

Export button

```
        private void OnExportClick(object sender, RoutedEventArgs e)
        {
            List<Eredmenyek> eredmenyek = null;

            using (_appDbContext = new AppDbContext())
            {
                eredmenyek = _appDbContext.Eredmenyek
                                .Include(x => x.Helyszin)
                                .Include(x => x.Versenyzo).ToList();
            }

            List<string> kimenet = new List<string>();
            string sor = string.Empty;

            foreach (Eredmenyek eredmeny in eredmenyek)
            {
                sor =
$"{eredmeny.Versenyzo.Nev}\t{eredmeny.Helyszin.Ev}\t{eredmeny.Helyszin.Orszag}\
t{eredmeny.Helyszin.Varos}\t{eredmeny.Vsenyszam}\t{(eredmeny.Helyezes.HasValue
? eredmeny.Helyezes : 0)}";
                kimenet.Add(sor);
            }
            File.WriteAllLines("eredmenyek.txt", kimenet);
            MessageBox.Show("Sikeres export.", "", MessageBoxButton.OK);
        }
```

## Consol alkalmazás

1. Console App .NET C#
2. Adat beolvasása txt fájlból
    a. jobbgomb – Properties – Copy always
3. Modell készítéséhez új osztály létrehozása

```
public class Eredmeny
{
    public string Nev { get; set; }
```

```
        public int Ev { get; set; }
        public string Orszag { get; set; }
        public string Varos { get; set; }
        public string Vsenyszam { get; set; }
        public int Helyezes { get; set; }
    }
```

## 4. Adatok beolvasása (Program.cs)

```
List<Eredmeny> eredmenyek = new List<Eredmeny>();

Beolvasas();

Console.ReadKey();

void Beolvasas()
{
    Eredmeny eredmeny = null;
    string[] egySorAdatai = null;

    string[] allLines = File.ReadAllLines("eredmenyek.txt" , Encoding.UTF8);

    foreach (string line in allLines)
    {
        egySorAdatai = line.Split('\t');

        eredmeny = new Eredmeny();
        eredmeny.Nev = egySorAdatai[0];
        eredmeny.Ev = int.Parse(egySorAdatai[1]);
        eredmeny.Orszag = egySorAdatai[2];
        eredmeny.Varos = egySorAdatai[3];
        eredmeny.Vsenyszam = egySorAdatai[4];
        eredmeny.Helyezes = int.Parse(egySorAdatai[5]);

        eredmenyek.Add(eredmeny);
    }
}
```

## 5. Lekérdezések

```
//hány eredmény van a listában?
Console.WriteLine($"1. feladat: A listában {eredmenyek.Count} eredmeny
van.");

//rudugrásban hány első helyezet van?
int rudugrasElsoHelyezetjei = eredmenyek.Where(x => x.Vsenyszam ==
"rúdugrás")
                                .Where(x => x.Helyezes == 1)
                                .Count();

Console.WriteLine($"2. feladat: A rúdugrásban 1. helyezettek száma:
{rudugrasElsoHelyezetjei}.");

//kik versenyeztek már S betűvel kezdődő országban?
List<string> versenyzokSNevuOrszagokban = eredmenyek.Where(x =>
x.Orszag.Contains('S', StringComparison.Ordinal)) //x.Orszag[0] == "S"
                                                  .Select(s =>
(s.Nev)).Distinct()
                                                  .ToList();


Console.WriteLine($"3. feladat: S betűvel kezdődő országokban
résztvettek:");
foreach (string versenyzo in versenyzokSNevuOrszagokban)
{
    Console.WriteLine($" - {versenyzo}");
```

```
    }

    //sportáganként hány eredmény van a listában?
    Dictionary<string, int> szotar = new Dictionary<string, int>();
    int darabszam = 0;

    List<string> versenyszamok = eredmenyek.Select(x => x.Vsenyszam)
                                           .Distinct()
                                           .ToList();

    foreach (string versenyszam in versenyszamok)
    {
        darabszam = eredmenyek.Count(x => x.Vsenyszam == versenyszam);
        szotar.Add(versenyszam, darabszam);
    }

    Console.WriteLine($"4. feladat: Versenyszámok száma:");
    foreach(KeyValuePair<string,int> elem in szotar)
    {
        Console.WriteLine($"- {elem.Key}: {elem.Value}");
    }
```

A 4. feladatra egy másik megoldás

```
Console.WriteLine($"4.2 feladat: Versenyszámok száma:");
int darabszam = 0;

List<string> versenyszamok = eredmenyek.Select(x => x.Vsenyszam)
                                       .Distinct()
                                       .ToList();

foreach (string versenyszam in versenyszamok)
{
    darabszam = eredmenyek.Count(x => x.Vsenyszam == versenyszam);
    Console.WriteLine($"- {versenyszam}: {darabszam}");
}
```

## ITMP Klub minta

### 1. Fájl beolvasása

#### Csv fájlból adat beolvasása

using System.Linq;

mindez az Ad.cs fájlban

```
0 references
public static IEnumerable<Ad> LoadFromCsv(string filename)
{
    foreach (string sor in File.ReadAllLines(filename).Skip(1))
    {
        yield return new Ad(sor);
    }
}
```

constructor készítése

```
class Ad
{
    1 reference
    public Ad(string sor)
    {
        string[] adatok = sor.Split(";");
        this.Id = Convert.ToInt32(adatok[0]);
        this.Rooms = Convert.ToInt32(adatok[1]);
        this.LatLong = adatok[2];
        this.Floors = Convert.ToInt32(adatok[3]);
        this.Area = Convert.ToInt32(adatok[4]);
        this.Description = adatok[5];
        this.FreeOfCharge = adatok[6] == "1";
        this.ImageUrl = adatok[7];
        this.CreateAt = Convert.ToDateTime(adatok[8]);
        this.Seller = new Seller() { Id = Convert.ToInt32(adatok[9]), Name = adatok[10], Phone = adatok[11] };
        this.Category = new Category() { Id = Convert.ToInt32(adatok[12]), Name = adatok[13] };

    }
```

Adatok betöltése a programba

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        List<Ad> hirdetesek = Ad.LoadFromCsv("realestates.csv").ToList();

        Console.ReadLine();
    }
}
```

## Json fájlból adat beolvasása

NugetPackes – Newtonsoft.Json telepítése

using Newtonsoft.Json;

mindez az Ad.cs fájlban

```
0 references
public static IEnumerable<Ad> LoadFromJson(string filename)
{
    return JsonConvert.DeserializeObject<Ad[]>(File.ReadAllText(filename));
}
```

paraméter nélküli konstruktor létrehozása

```
public Ad()
{

}
```

Adatok betöltése a programba

```
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        List<Ad> hirdetesek = Ad.LoadFromJson("realestates.json").ToList();

        Console.ReadLine();
    }
}
```

## 2. Conzol alkalmazás

Fölszinti ingatlanok átlagos alapterületének kiírása (LINQ)

```
Console.WriteLine("1. Földszintes ingatlanok átlagos alapterületét: {0:F2} m2",
    hirdetesek.Where(x => x.Floors == 0).Average(x => x.Area));
```

Metódus készítése pitagorasz

```
0 references
public double DistanceTo(double x, double y)
{
    double dx = Math.Abs(Convert.ToDouble(this.LatLong.Split(",")[0].Replace(".", ",")) - x);
    double dy = Math.Abs(Convert.ToDouble(this.LatLong.Split(",")[1].Replace(".", ",")) - y);
```

pitagorasz számítás

minimum keresés

```
var adat = hirdetesek.Where(x => x.FreeOfCharge)
                .OrderBy(x => x.DistanceTo(47.4164220114023, 19.066342425796986)).First();
Console.WriteLine("2. Medsevár óvodához légvonalban legközelebbi tehermentes ingatlan adatai:");
Console.WriteLine("\tEladó neve: {0}", adat.Seller.Name);
Console.WriteLine("\tEladó telefonja: {0}", adat.Seller.Phone);
Console.WriteLine("\tAlapterület: {0}", adat.Area);
Console.WriteLine("\tSzobák száma: {0}", adat.Rooms);
```

## 3. Grafikus alkalmazás

MainWindow.xaml

1. Title átírása
2. Oszlopok létrehozása

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="2*"/>
    </Grid.ColumnDefinitions>

</Grid>
```

3. Felület feltöltése adatokkal

```
<ListBox Grid.Column="0">

</ListBox>
<StackPanel Grid.Column="1">
    <StackPanel Orientation="Horizontal">
        <Label Content="Eladó neve:" />
        <Label Content="{Binding Name}" />
    </StackPanel>
    <StackPanel Orientation="Horizontal">
        <Label Content="Eladó telefonszáma:" />
        <Label Content="{Binding Phone}" />
    </StackPanel>
    <Button />
    <StackPanel Orientation="Horizontal">
        <Label Content="Hirdetések száma:" />
        <Label Content="0" Name="AdCount"/>
    </StackPanel>
```

```
<ListBox Grid.Column="0" Name="Seller">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Label Content="{Binding Name}" />
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Amit a jobb oldalon kiválasztunk, az a bal oldal adatkörnyezetét megváltoztatja

```
<StackPanel Grid.Column="1" DataContext="{Binding ElementName=Seller, Path=SelectedItem}">
```

4. Adatbázis kapcsolat MainWindow.xaml.cs
    a. függőségek beállítása
    b. MySql connection

```
List<Seller> sellers = new List<Seller>();
MySqlConnection conn = new MySqlConnection("server=localhost; user=root; database=ingatlan; port=3306");
```

    c. sql fájl lefuttatása az adatbázisban

```
0 references
public MainWindow()
{
    InitializeComponent();
    try
    {
        conn.Open();
        string sql = "SELECT id, name, phone FROM sellers";
        MySqlCommand cmd = new MySqlCommand(sql, conn);

        MySqlDataReader rdr = cmd.ExecuteReader();

        while (rdr.Read())
        {
            sellers.Add(new Seller()
            {
                Id = Convert.ToInt32(rdr["Id"].ToString()),
                Name = rdr["Name"].ToString(),
                Phone = rdr["Phone"].ToString()
            });
        }
        rdr.Close();
    } catch (Exception ex)
    {

    }
    conn.Close();
    this.Seller.ItemsSource = sellers;
```

```
<Button Content="Hirdetések betöltése" Click="Button_Click"/>
```

```
1 reference
private void Button_Click(object sender, RoutedEventArgs e)
{
    try
    {
        conn.Open();
        string sql = "SELECT COUNT(id) AS AdCount FROM realestates WHERE sellerid = @sellerid";
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        cmd.Parameters.Add(new MySqlParameter("@sellerid", (this.Seller.SelectedItem as Seller).Id));
        MySqlDataReader rdr = cmd.ExecuteReader();

        rdr.Read();
        this.AdCount.Content = rdr["AdCount"].ToString();
        rdr.Close();
    }
    catch (Exception ex)
    {

    }
    conn.Close();
}
```