

Eseményvezérelt alkalmazások: 4. gyakorlat

A munkafüzet bevezet minket a modell-nézet alapú grafikus alkalmazás fejlesztésbe. A modell komponens felel az üzleti logikáért, független a nézettől és a grafikus könyvtáraktól (*Windows Forms*). Állapotváltozásait a nézet felé eseményeken keresztül jelzi. A nézet komponens a megjelenítésért és a megjelenítéshez kapcsolódó logikáért felel elsődlegesen.

Dokumentum statisztikák (grafikus felülettel)

Készítsünk grafikus felületet a 2. gyakorlaton elkészített, szöveges állományokban a szavak előfordulását számláló alkalmazáshoz, melyen megjelenítjük a betöltött fájl szövegét, a szavak előfordulásainak számát, valamint még néhány hasznos információt!

Az új projekt létrehozása során válasszuk a **Windows Forms App** sablont! A projekt neve legyen DocuStatView, A *solution* neve pedig DocuStat!

Ne a *Windows Forms App (.NET Framework)* sablont válasszuk, ugyanis az .NET 8 helyett a korábbi .NET Frameworkre épülő projektet fog létrehozni!

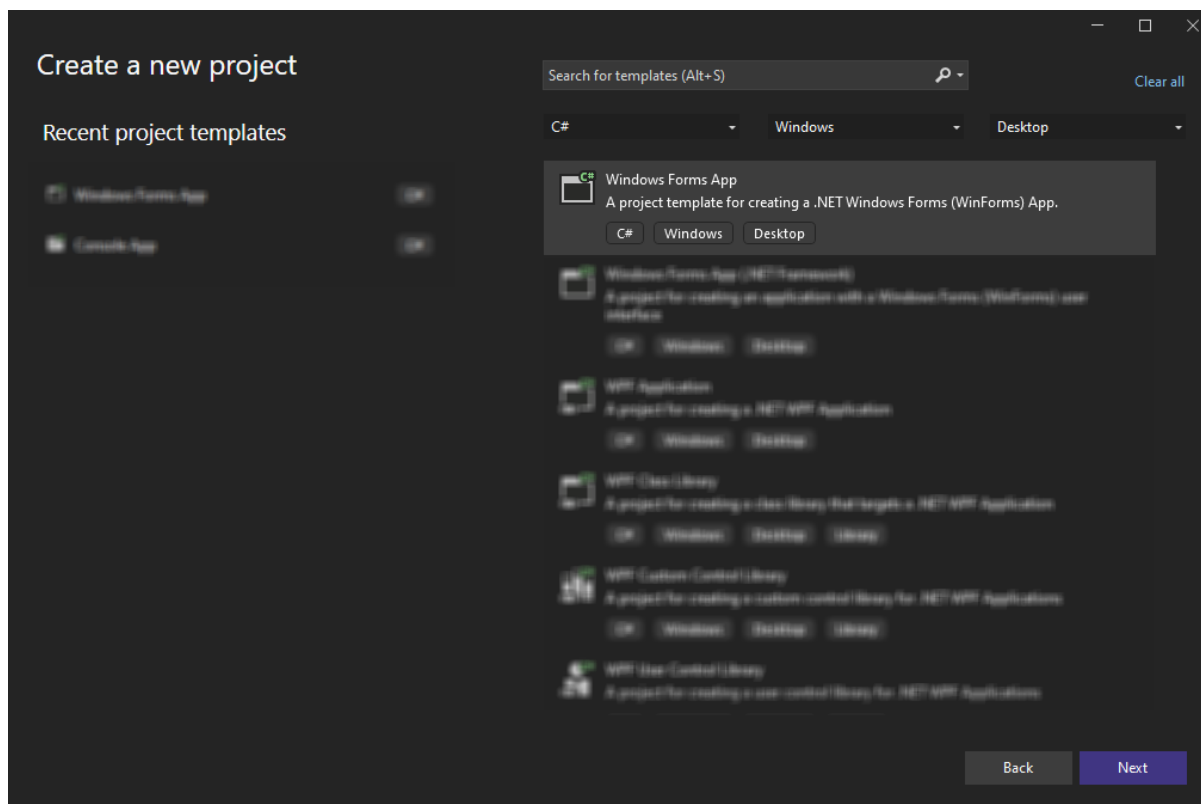


Figure 1: Windows Forms projekt létrehozása

Nevezzük át a `Form1.cs` fájlt `DocuStatDialog.cs`-re!

A *VS2022*-ben fenn áll az a bug, hogy a Windows Forms tervezőfelületén egy *form* vagy egy *user control* átnevezését követően, ha nem kerül annak *Designer* nézete bezárásra és újrainvitásra, akkor a tervezőfelületen végzett módosítások láthatóan megtörténnek, de kódgenerálás nem történik a háttérfájlbba (*Designer.cs*), így valójában elvesznek a módosítások. Előreláthatólag a következő verzióban lesz benne a javítás: [Bug report](#).

A Designerben helyezzük fel a következő vezérlőket az ablakra:

- **MenuStrip:** az ablak tetején, a címsor alatt helyezkedjen el, ez lesz az alkalmazás menüsávja. A vezérlőre jobb klikkelve válasszuk ki az **Edit Items...** menüpontot. Az **Add** gombbal adjunk új menüelemet (**MenuItem**) a menüsávhoz.
 - A jobb oldalon látható részben be tudjuk állítani a menüelem nevét ((**Name**)), ez legyen `fileMenu`, és szövegét (**Text**), ez legyen `File`. A `fileMenu`-t kijelölve keressük meg a tulajdonságok között a **DropDownItems** elemet, és a (**Collection**)-re kattintva válasszuk ki a mellette megjelenő három pontot. Ezzel egy hasonló menüt kapunk, ahol a `fileMenu`-höz adhatunk új menüelemeket az **Add** gombbal.
 - * Adjunk a `fileMenu`-höz egy új elemet, ennek neve ((**Name**)) legyen `openFileDialogMenuItem`, felirata (**Text**) pedig legyen `Open file dialog`.
 - * Egy második menüelem neve legyen `countWordsMenuItem`, a felirata pedig legyen `Count words`.

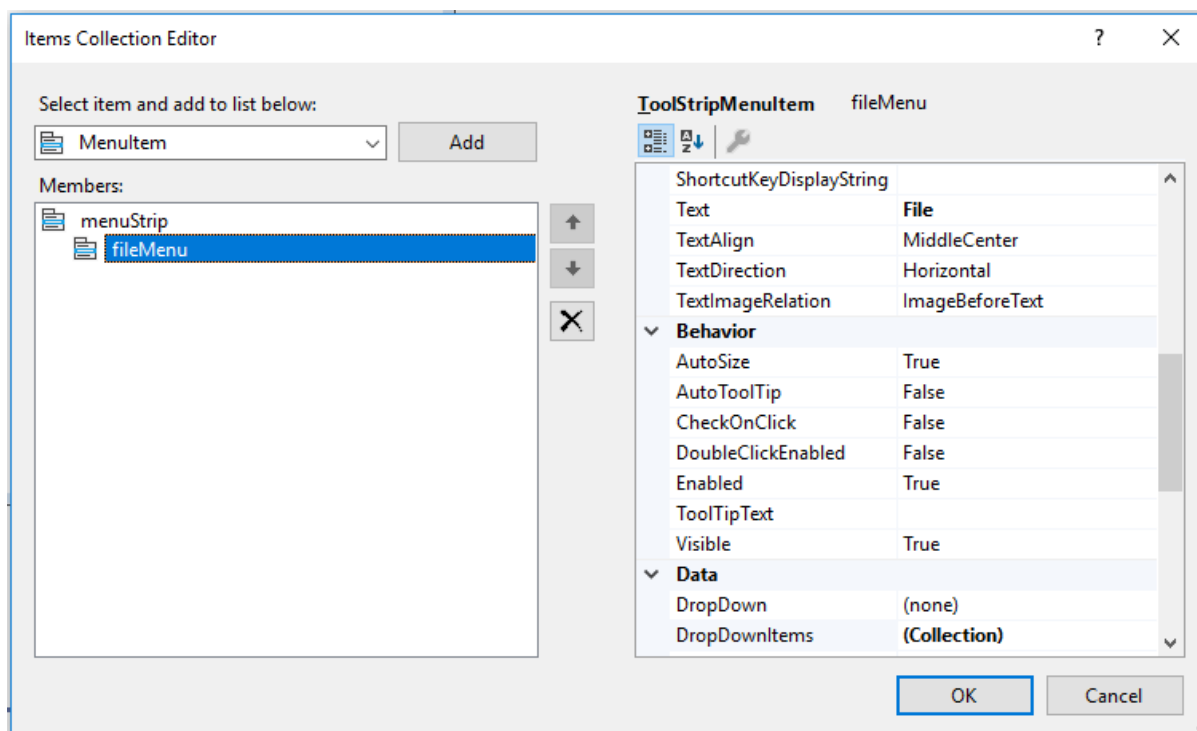


Figure 2: Menüpont hozzáadása a menüsávhoz.

- **TextBox:** ebben fogjuk megjeleníteni a szöveges fájl tartalmát. Helyezzük az ablak bal oldalára, és méretezzük szélességét úgy, hogy kb. az ablak felét foglalja el. A *Solution Explorer* alatt megjelenik a Designerben aktuálisan kijelölt elemhez tartozó tulajdonságok listája (**Properties**), amelyben módosítani tudjuk a vezérlő viselkedését, eseménykezelőket adhatunk az eseményeihez stb. A `TextBox` (**Name**) tulajdonságát írjuk át `textBox`-ra, majd a `ScrollBars` tulajdonságot állítsuk `Vertical`-ra, hogy görgethető legyen a mező. Állítsuk igazra a `MultiLine` tulajdonságot, hogy a magassága több soros is lehessen, ezután növeljük meg a magasságát. A `ReadOnly` tulajdonságot is állítsuk igazra, hogy a felületen keresztül ne legyen szerkeszthető a szöveg.

- **ListBox:** ebben jelenítjük meg a szavakat és előfordulásaik számát. Helyezzük az ablak jobb oldalára, és méretezzük át úgy, hogy mérete a bal oldali **TextBox** méretével megegyezzen. A **Properties** ablakban adjuk neki a **listBoxCounter** nevet.
- **Label:** Adjunk a felülethez az ábrán látható módon 3 címkét ezekkel a feliratokkal: **Minimum word length:**, **Minimum word occurrence:**, **Ignored words:**. Majd adjunk még további 6 címkét az alábbi név-felirat tulajdonságokkal:
 - **labelCharacters** - Character count:,
 - **labelNonWhitespaceCharacters** - Non-whitespace characters:,
 - **labelSentences** - Sentence count:,
 - **labelProperNouns** - Proper noun count:,
 - **labelColemanLieuIndex** - Coleman Lieu Index:,
 - **labelFleschReadingEase** - Flesch Reading Ease:.
- **NumericUpDown:** ezekben fogjuk beállítani a minimális szóhosszot és minimális előfordulások számát. Mindegyik **Minimum** tulajdonságát állítsuk 1-re. Név tulajdonságuk legyen **spinBoxMinLength** és **spinBoxMinOccurrence**. Az előfordulások **Maximum** tulajdonsága legyen 500!
- **TextBox:** A szavak számlálásából kizárandó szavakat ide lehet majd beírni vesszővel elválasztva, szóköz nélkül. Neve legyen **textBoxIgnoredWords**.

A teljes ablak kijelölésével láthatjuk a **Properties** ablakban az ablak tulajdonságait. Legyen a fejléc szövege (**Text**) **Document statistics**! A **FormBorderStyle** tulajdonságot állítsuk **FixedSingle**-re, a **MaximizeBox** tulajdonságot pedig hamisra, így az ablak fix méretű lesz.

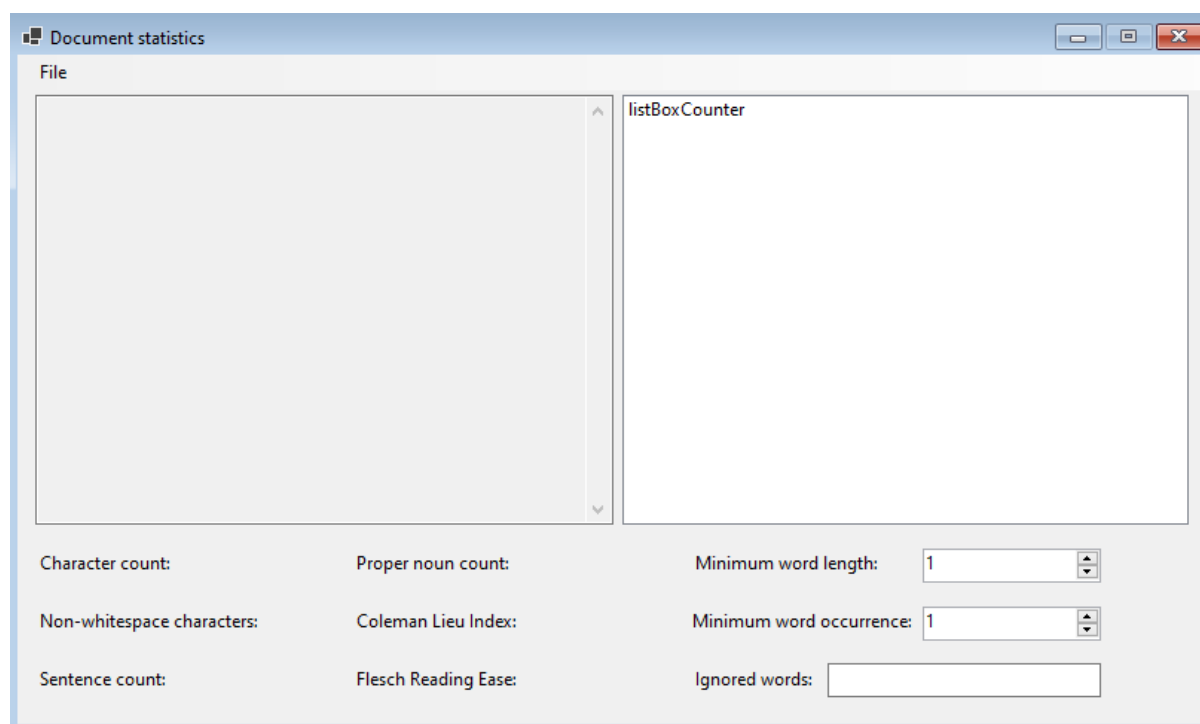


Figure 3: Az alkalmazáshoz tartozó ablak nézete a Designerben.

Modell aggregálása a nézetben ^{KM}

Mivel fel fogjuk használni a 2. gyakorlat **DocumentStatistics** osztályát, ezért először hozzá kell férnünk az azt tartalmazó projekthez is. Jobb klikkeljünk a **Solution** nevére, **Add -> Existing Project...**. Válasszuk ki a 2. gyakorlat **DocuStat.csproj** nevű projekt fájlját. Ezt követően a **DocuStatView** projekt nevére jobb klikkeljünk, **Add -> Project Reference...**. Pipáljuk be a **DocuStat** projektet, majd okézzuk le az ablakot.

Az eseményvezérelt paradigmának megfelelően a modell 2 eseménnyel jelzi az állapotváltozását. A DocumentStatistics osztályhoz adjunk hozzá két egyedi eseményt: FileContentReady és TextStatisticsReady:

```
public event EventHandler? FileContentReady;
public event EventHandler? TextStatisticsReady;
```

Ezeket akkor fogjuk kiváltani, amikor a szöveges fájl tartalma betöltésre került (FileContentReady), valamint a metrikák kiszámítása megtörtént (TextStatisticsReady). Szükségünk lesz két privát metódusra, amelyekkel ezt el tudjuk érni:

```
private void OnFileContentReady()
{
    FileContentReady?.Invoke(this, EventArgs.Empty);
}

private void OnTextStatisticsReady()
{
    TextStatisticsReady?.Invoke(this, EventArgs.Empty);
}
```

Hívjuk meg ezeket a Load() metóduson belül a megfelelő helyeken: - OnFileContentReady(): a szöveges fájl tartalmának beolvasását követően. - OnTextStatisticsReady(): a metrikák kiszámítását követően.

Nyissuk meg a DocuStatDialog.cs fájlt (ami semmiképpen nem a DocuStatDialog.Designer.cs)! Ebben a fájlban írjuk meg a menüelemekhez tartozó eseménykezelőket, amelyek a fájlbeolvasást és a statisztika elkészítést fogják végezni, és kitöltik a textBox-ot és a listBoxCounter-t. A usingok közé vegyük fel az ELTE.DocuStat.Model névteret! Vegyünk fel egy privát DocumentStatistics? típusú adattagot (_documentStatistics).

Fájl betöltése ^{EM}

Definiáljuk az openFileDialogMenuItem-hez tartozó eseménykezelőt! Az eseménykezelő egy privát, void visszatérési típusú metódus, amely a küldő objektumot és egy eseményargumentumot vár paraméterül. A neve legyen OpenFileDialog.

```
private void OpenFileDialog(object? sender, EventArgs e)
```

A fájlt most egy fájlállító dialóguson keresztül fogjuk megnyitni. Használjuk ehhez az OpenFileDialog osztályt. A dialógus kezdeti könyvtárát az InitialDirectory, a megengedett fájl típusokat a Filter tulajdonságon keresztül állíthatjuk be. Ha azt szeretnénk, hogy minden megnyitáskor a kezdeti könyvtár jelenjen meg, állítsuk a dialógus RestoreDirectory tulajdonságát igazra.

A dialógus ShowDialog() metódusát meghívva megjelenítjük a fájlállítót. Ha itt sikeresen kiválasztottunk egy fájlt, akkor példányosítsunk egy új DocumentStatistics objektumot, melynek konstruktor paramétere legyen az openFileDialog FileName propertyje. Iratkozzunk fel a modell két eseményére a UpdateFileContent és UpdateTextStatistics eseménykezelőkkel. Ezután hívjuk meg a _documentStatistics Load() metódusát. Ne feledjük, hogy itt el kell kapnunk az esetleges kivételt. A catch ágban jelenítsünk meg egy MessageBox-ot a kivétel szövegével (Message), majd térjünk vissza.

```
private void OpenFileDialog(object? sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory = "C:\\\\";
        openFileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog.RestoreDirectory = true;

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            try
            {

```

```

        _documentStatistics = new DocumentStatistics(openFileDialog.FileName);
        _documentStatistics.FileContentReady += UpdateFileContent;
        _documentStatistics.TextStatisticsReady += UpdateTextStatistics;
        _documentStatistics.Load();
    }
    catch (System.IO.IOException ex)
    {
        MessageBox.Show("File reading is unsuccessful!\n" + ex.Message,
            "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
}
}
}

```

A `UpdateFileContent` eseménykezelőben megspórolhatjuk a szöveg ismételt betöltését abban az esetben, ha ugyanazt a fájlt ugyanazzal a tartalommal próbáljuk betölteni, mint ami már a `textBox`-ban van. Vizsgáljuk meg, hogy a `_documentStatistics.FileContent` egyezik-e a `textBox` tartalmával (`Text`), feltéve, hogy előbbi nem üres (amit a `String.IsNullOrEmpty()` metódussal tudunk ellenőrizni)! Ha a betöltendő szöveg egyezik a korábbival, térjünk vissza.

A `textBox` `Text` tulajdonságának adjuk értékül a `FileContent` tartalmát, így megjelenik a `textBox`-ban a fájlból beolvasott szöveg. A `listBoxCounter` `Items` kollekciójának tartalmát töröljük a `Clear()` metódussal.

```

private void UpdateFileContent(object? sender, EventArgs e)
{
    if (_documentStatistics?.FileContent == textBox.Text)
        return;

    textBox.Text = _documentStatistics?.FileContent;
    listBoxCounter.Items.Clear();
}

```

A `textBox` alatt elhelyezkedő címkéket is frissíteni kell, hogy már a kiszámított metrikák értékei is megjelenjenek rajtuk. Ezt a `UpdateTextStatistics` eseménykezelővel fogjuk elérni. A `_documentStatistics` megfelelő tulajdonságait felhasználva állítsuk át a címkék `Text` tulajdonságát úgy, hogy az eredeti szövegek mellett az eredmények is látszódjanak.

Az `OpenDialog`-ot kössük össze a konstruktorban az `openFileDialogMenuItem` `Click` eseményével.

```
openFileDialogMenuItem.Click += OpenDialog;
```

Eseménykezelőt a `==` operátorral vehetünk le egy eseményről.

Szó statisztikák megjelenítése és szűrése ^{EM}

Definiáljunk egy új eseménykezelőt `CountWords` néven! Láthatósága, visszatérési típusa, paraméterei ugyanazok legyenek, mint az eddigi eseménykezelőknek.

Ha még nem olvastunk be fájlt, azt szeretnénk, ha ez a menüpont feldobna egy üzenetet. Ha a `_documentStatistics.FileContent` üres (amit a `String.IsNullOrEmpty()` metódussal tudunk megvizsgálni), dobjunk fel egy `MessageBox`-ot, amely tájékoztat, hogy még nincs beolvasott szöveg, majd térjünk vissza.

Változókbán tároljuk el a felhasználó által beállított paramétereket (`minLength`, `minOccurrence`, `ignoredWords`), ezeket alakítsuk megfelelő formátumúra! A `Convert.ToInt32()` metódussal lehet egész számmá alakítani értékeket. Az ignorált szavak megfelelő átalakítására célszerű a `Split()`, `Select()` és `ToList()` metódusokat használni (ld. 2. gyakorlat).

Majd a 2. gyakorlathoz hasonlóan rendezzük a `_documentStatistics.DistinctWordCount` elemeit.

A `listBoxCounter` vezérlő `BeginUpdate()` metódusát meghívva kezdetjük a párok kiírását. A rendezett párokon végig iterálva adjuk hozzá az `Items`-hez szöveggént a párok kulcsát és értékét, majd az `EndUpdate()` metódussal zárjuk le a `listBoxCounter` szerkesztését.

```
listBoxCounter.Items.Clear();
listBoxCounter.BeginUpdate();
foreach (var pair in pairs)
{
    listBoxCounter.Items.Add(pair.Key + ": " + pair.Value);
}
listBoxCounter.EndUpdate();
```

A lista frissítését a `BeginUpdate()` és az `EndUpdate()` keretébe zárása nélkül is elvégezzük, ez esetben azonban minden elem hozzáadása a megjelenítés frissítését váltja ki, amely így egy villogó hatást okozhat a felhasználói felületen.

A `CountWords` eseménykezelőt az előzőhöz hasonlóan kössük rá a `countWordsMenuItem` vezérlő `Click` eseményére.

Az alkalmazás osztálydiagramja:

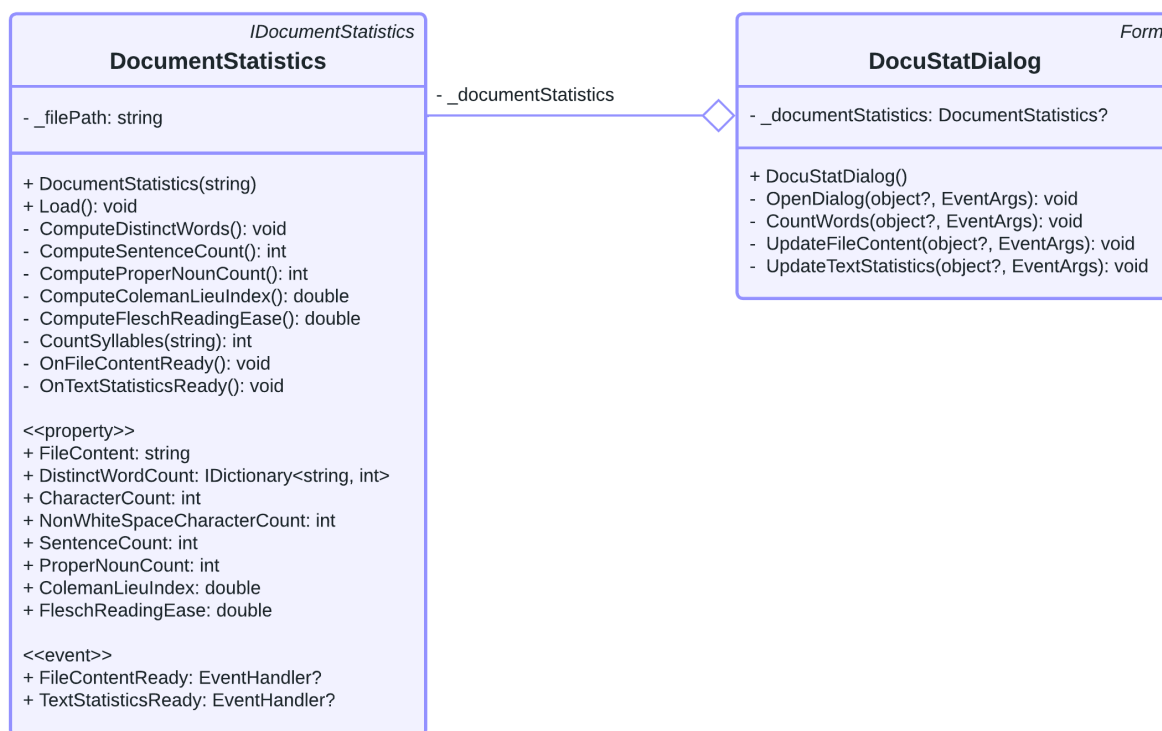


Figure 4: Az alkalmazáshoz tartozó osztálydiagram.