Game of Thrones World

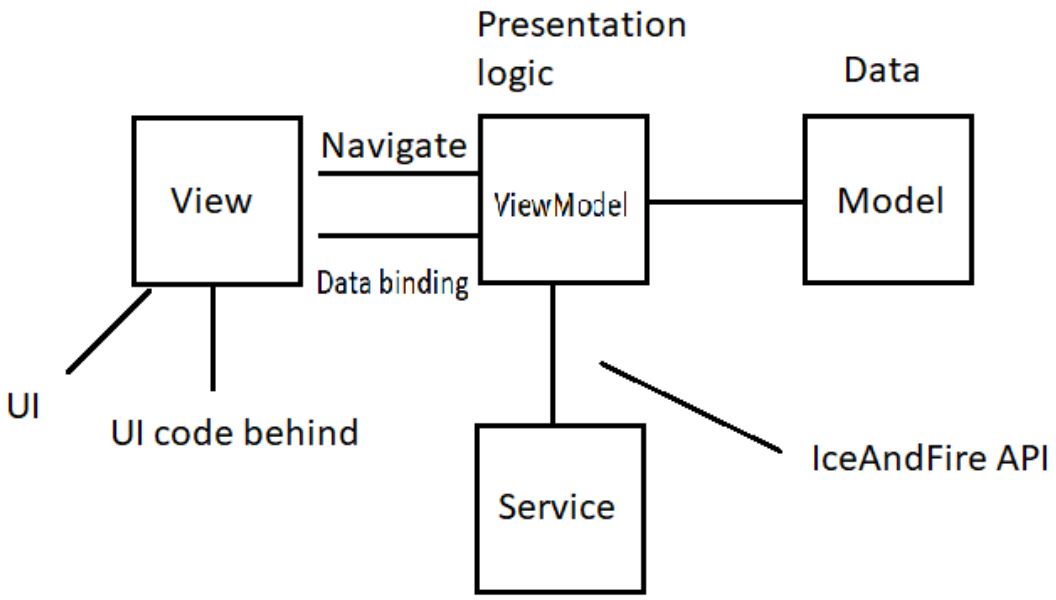
Kliensoldali technológiák házi feladat – Készítette: Varga Tamás E7BN3F

Összefoglaló

A Game of Thrones alkalmazás a népszerű „Trónok harca” könyvsorozat kiadásait, szereplőit és házait mutatja be. Az alkalmazás az Ice And Fire API-ra (https://anapioficeandfire.com/) épül. Az alkalmazás képes a mágikus világ bejárására, könyvek, családok és karakterek listázására és ezek részletes megjelenítésére.

Architektúra

Az alkalmazást a *Modell-View-ViewModell* architektúrális minta alapján készítettem el. Az architektúrális minta segítségéhez a Template10 keretrendszert vettem alapul, ami többek között az MVVM funkciók támogatására is ad előre elkészített megoldást. A rendszer felépítése, és kapcsolódási pontjaik a következő ábrán láthatók.



1. ábra Felépítés és kapcsolódási pontok

Az alkalmazásban használt architekturális elemek:

Views

- **MainPage**: az alkalmazás indítása után látható oldal
- **BooksPage**: a kilistázott könyvek oldala
- **BookDetailsPage**: a kiválasztott könyv részletező oldala
- **CharactersPage**: a kilistázott karakterek oldala
- **CharacterDetailsPage**: a kiválasztott karakter részletező oldala
- **HousesPage**: a kilistázott házak oldala

- **HouseDetailsPage**: a kiválasztott ház részletező oldala

ModelViews

- A fent említett nézetekhez kivétel nélkül tartozik egy ViewModel osztály.

Service

- Ide tartoznak a szolgáltatásosztályok

Fontosabb osztályok, komponensek

Service ősosztály

Az alkalmazás REST-s webszolgáltatást használ. Az https://anapioficeandfire.com/Documentation címen megtalálható a webszolgáltatás interfészének a dokumentációja.

A kérés során már lehetőségünk van az általunk kiválasztott oldalszám és elem/oldalszám kiválasztására. Ez a ?page=X&pageSize=Y paraméterekkel tehetjük meg.

A kapott http válasz fejlécében számunkra fontos információt is kapunk. A válaszban megtalálhatóak a következő, az első, és utolsó oldalra mutató URL linkek. A fenti paraméterekkel ugyanis változtathatjuk az egy oldalon megjelenő elemek számát, így előfordulhat, hogy nem fér ki minden egy teljes oldalra.

- *GET api/books* kérés az összes könyvet listázza
- *GET api/characters* kérés az összes karaktert listázza
- *GET api/houses* kérés az összes házat listázza

Az egyes Modelleknek megfelelő ViewModel osztályok

Az egyes osztályok rendre tartalmaznak legalább egy NotifyTaskCompletion<> property-t. A NotifyTaskCompletion osztály implementálja az INotifyPropertyChanged

interfészt, és aszinkron módon ad segítséget. Az osztály konstruktorában egy Task watch metódusban van megadva, hogy várjon a taszk végeztéig. Amint elkészül, beállítja a saját propertychanged property-t és értesítést ad. Az adatkötés müködéséhez a nézet osztályokban a DataContect tulajdonságnak egy ViewModel van megadva, és a megfelelő részek Binding-olva vannak ennek a NotifyTaskCompletion property Result-jához.

Könyvek listájának lekérése.

Az alábbiakban részletesen áttekintem, milyen lépések történnek, amikor a felhasználó lekéri az összes könyvet.

1. A felhasználó a Books nevű NavigationView MenuItem-en kattint.

2. A View code behind-ban történik a kattintás kezelése. Itt a Frame Navigate() metódusával a BooksPage oldalra navigálunk.

```
private void NavView_ItemInvoked(NavigationView sender, NavigationViewItemInvokedEventArgs args)
{
    var label = args.InvokedItem.ToString();
    var pageType =
        label == "Books" ? typeof(BooksPage) :
        label == "Characters" ? typeof(CharactersPage) :
        label == "Houses" ? typeof(HousesPage) : null;
    if(pageType != null && pageType != ContentFrame.CurrentSourcePageType)
    {
        ContentFrame.Navigate(pageType);
    }

}
```

3. A BooksPage code-behind-jában az `OnNavigatedTo metódus van felülírva. Ez lóg ki egy kicsit az MVVM mintából, de a metódusban csak a ViewModel beállítása történik.`

```
// Just for set the DataContext, the appropriate viewmodel
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    var url = (string)e.Parameter;
    if(url != null)
    DataContext = new BooksPageViewModel(url);
    else
    {
        DataContext = new BooksPageViewModel();
    }
    base.OnNavigatedTo(e);
}
```

4. A ViewModel constructorában a `NotifyTaskCompletion property inicializálása történik. Ez implementálja az INotifyPropertyChanged interfészt, és aszinkron módon történik a http kérések indítása. Amint a kérésre megérkezik a válasz, értesít a változásról, és Page-ben megadott Binding-k alapján megjelenik a képernyőn a megfelelő adat.`

```
// Constructor which set the property
public BooksPageViewModel(string uri)
{

    NotifyBooks = new NotifyTaskCompletion<List<Book>>(Service.GetBooksAsync(uri));
}
```

Forrás

A dokumentációban a következő anyagokat használtam fel:

Kliens oldali technológiák Házi minta dokumentáció

Template10,

3. XAML MVVM gyakorlat.

https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/march/async-programming-patterns-for-asynchronous-mvvm-applications-data-binding

https://gist.github.com/pimbrouwers/8f78e318ccfefff18f518a483997be29