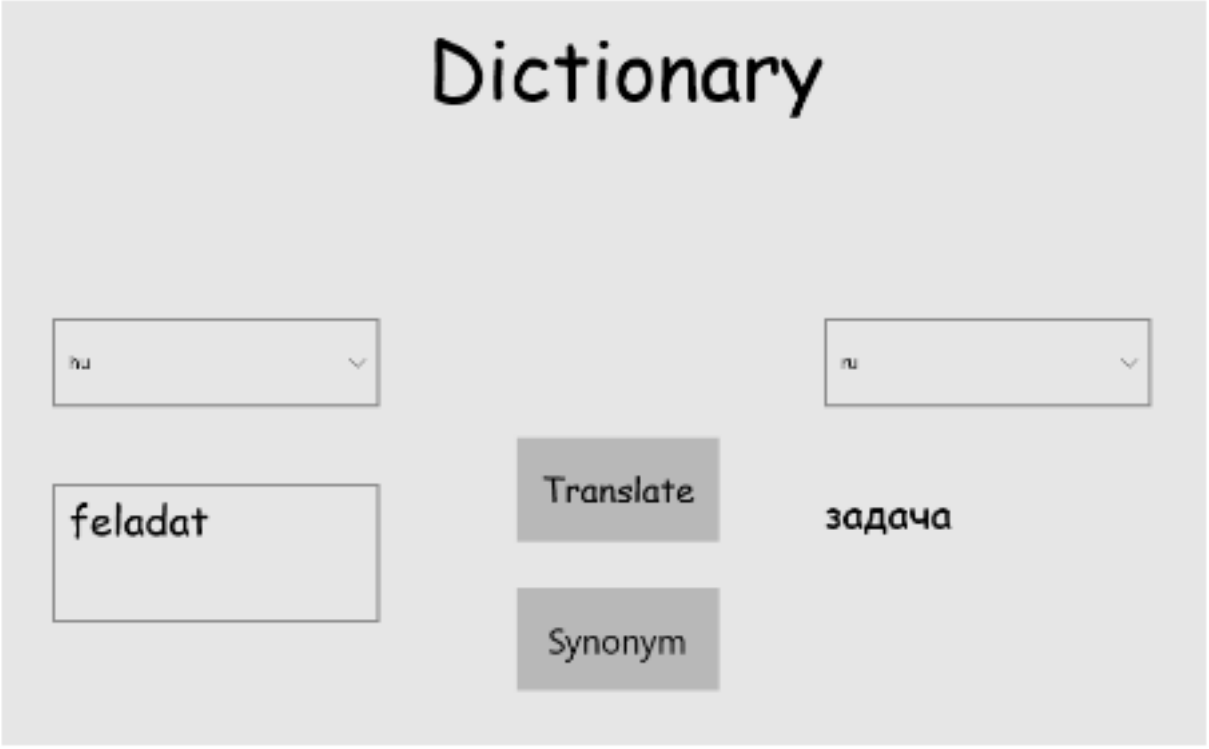
Kliensoldali technológiák
Szótár - Készítette: Heveli Richard (NODP2T)

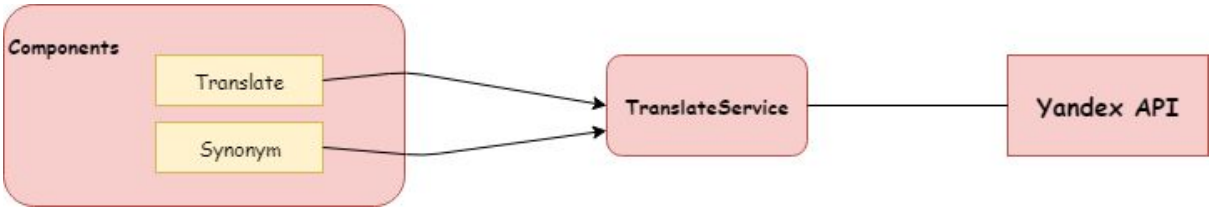
Összefoglaló

Az alkalmazás egy egyszerű szótárat valósít meg. Egy ingyenes API-ra épít, melynek neve Yandex dictionary API. A szótárban lehetőség van a forrás és a célnyelv kiválasztására. Miután kiválasztottuk a forrásnyelvet, a célnyelvek közül csak olyanokat választhatunk, amelyikhez tényleges szótár is létezik. A fordítás mellett az alkalmazás képes szinonímák keresésére is. A felhasználó a "Translate" gombbal fordíthat szót, a "Synonym" szó segítségével pedig szinonímát kereshet.



Architektúra

Az alkalmazást Universal Windows Platform segítségével valósítottam meg, az alkalmazás architektúrája a következőképpen néz ki:



Főbb egységek

Az MVVM mintát követtem, így a projektemben ez alapján osztottam fel az osztályokat.

Models: Itt található a Languages osztály. Ennek a szerepe az, hogy az APIn hívott getLangs hívás eredményét tárolja. Ez nyelv szópárokat tartalmaz, mint például : "en-ru" "hu-ru". Mivel ez egy string lista, ezért fel kell dolgoznunk ezt az adatot. Ezért hoztam létre az osztályban egy makeList függvényt. Ez a függvény a lista stringből egy olyan Dictionary típusú eredménnyel tér vissza, amelynek kulcsa egy string(egy adott nyelv), ehhez a kulcshoz pedig tartozik egy Value érték, ami List<string>. Ezt használom majd a cél és forrásnyelv kiválasztásánál.

A TranslateData osztály egy szó fordításához szükséges adatokat tartalmazza, ami nem más mint maga a szó, a forrásnyelv és a célnyelv.

A WordResult osztály egy fordított szó eredményét fogja tárolni, ezt a Visual Studio beépített funkciójának segítségével hoztam létre, lekértem egy konkrét szó eredményét JSON-ben, majd ez alapján a Paste JSON As Classes opcióval legeneráltam az osztályt. Ez tartalmazza majd a lefordított szó szövegét, és ezek mellett sok más adatot, amiknek ebben a feladatban nincs szerepe.

ViewModel: Itt azok az adatok vannak tárolva, amik segítségével a Viewban megjeleníthetjük a kívánt szavakat, nyelveket stb. A nyelvlistákat ObservableCollection segítségével tárolom, míg a lefordított szót egy stringben. Az osztály implementálja az INotifyPropertyChanged interfészt. Ez amiatt szükséges, hogy amikor megváltozik translatedWord attribútum, akkor a Viewban frissüljön a textblock értéke.

View: A View xaml nyelven van írva. A MainPage, amit az alkalmazáskor használok, buttonokból, textboxból, comboboxokból áll. A felhasználó interakciói hatására a MainPage.xaml.cs fájlból meghívódnak a viewmodel megfelelő függvényei, és ezáltal frissül a felhasználó által látott textboxok adatai is. Itt van implementálva a Translate button click listenerje, ami a Viewmodel translate függvényének segítségével API hívást hajt végre és lekéri a textboxban szereplő szó fordítását. A nyelv lista választásakor a ComboBox SelectionChanged függvénye fut le.

Yandex API

A feladatomban a Yandex Dicionary API-ját használom. Ez az API lehetőséget teremt a szavak megkeresése és lefordítása mellett szinonímák keresésére is. Az API használata a programomban:

Az API két fontosabb függvényét használom ahhoz, hogy adatokat kérjek le. Az egyik a getLangs, ami a nyelvpárokat tartalmazza. A másik pedig a lookup, amivel egy konkrét szót tudok lekérdezni. Az API hívás során itt szükséges megadni a forrásnyelvet, célnyelvet, és magát a szót. Ha ugyanazt a célnyelvet választjuk

ki mint forrásnyelvet, akkor szinonímákat kereshetünk. Az API ezáltal egy szinoníma szótárként is használható, nem csak fordításra.
Az APIról részletes leírás olvasható a https://tech.yandex.com/dictionary/doc/dg/concepts/About-docpage/ oldalon.

Fordítás

A felhasználó az alkalmazás elindítása után lefordíthat szavakat. Ehhez a textboxba kell szót beírnia, és a Translate gombra kattintani.
A nyelvek kiválasztására két ComboBox van létrehozva. Az első combobox a forrásnyelv, a második a célnyelv kiválasztását segíti. Amikor a felhasználó kiválasztja a forrásnyelvet, meghívódik a ComboBox SelectionChanged függvénye. Itt a viewModel pushpairs metódusa feltölti a célnyelv listát olyan nyelvekkel, amik elérhetőek az adott forrásnyelvhez. Ez biztosítja azt, hogy a felhasználó nem választ olyan célnyelvet, amihez nem tartozik szótár.
A translate gomb Click függvénye a ViewModel translate függvényét hívja tovább. A translate függvény lényege, hogy egy TranslateService objektumot hoz létre. A szolgáltatás elvégzi az API hívást. A szolgáltatása Yandex APIból kiolvassa a JSONben leírt adatot, ezt konvertálom át egy WordResult objektummá, majd átállítom az adatokat tároló property értékét. Ez a property bindingolva van a Viewon, ezért a felhasználónak meg is jelenik a fordított szó. A szolgáltatás hibakezeléssel végzi el az API hívást, így ha a megadott URL cím helytelen, egy MessageDialog jelenik meg ami értesíti a felhasználót a hibás kérésről, valamint a valós adat helyett egy üres adatot továbbít. Az HttpClient hívás és a hibakezelés a szolgáltatás osztályban az alábbi képen látható:

```
private async Task<T> GetAsync<T>(Uri uri)
{
    using (var client = new HttpClient())
    {
        try
        {
            var response = await client.GetAsync(uri);

            var json = await response.Content.ReadAsStringAsync();
            T result = JsonConvert.DeserializeObject<T>(json);
            return result;

        }
        catch {
            var messageDialog = new MessageDialog("Http failure response for: " + uri.ToString());
            string json = "";
            await messageDialog.ShowAsync();
            T result = JsonConvert.DeserializeObject<T>(json);
            return result;
        }

    }
}
```