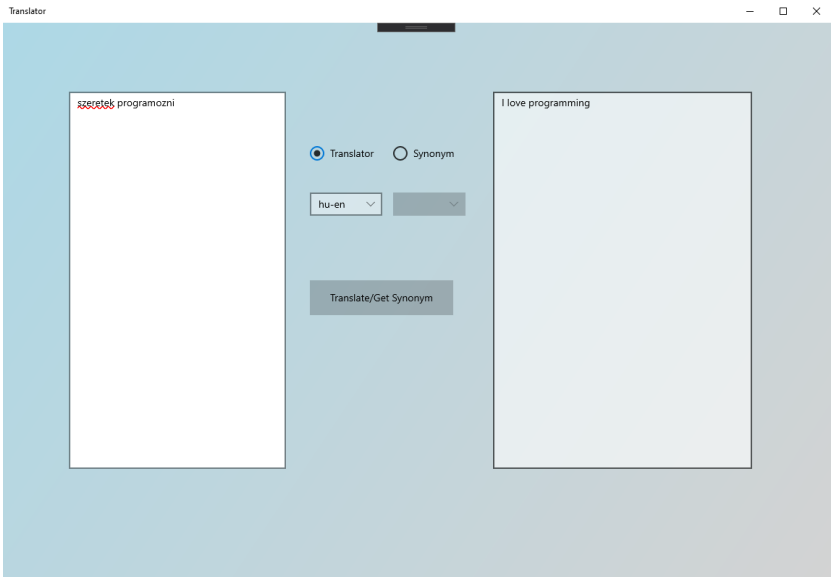
Szótár/szinonima kereső

Kliensoldali technológiák házi feladat –
Készítette: Villám Roland (T7L4OS)

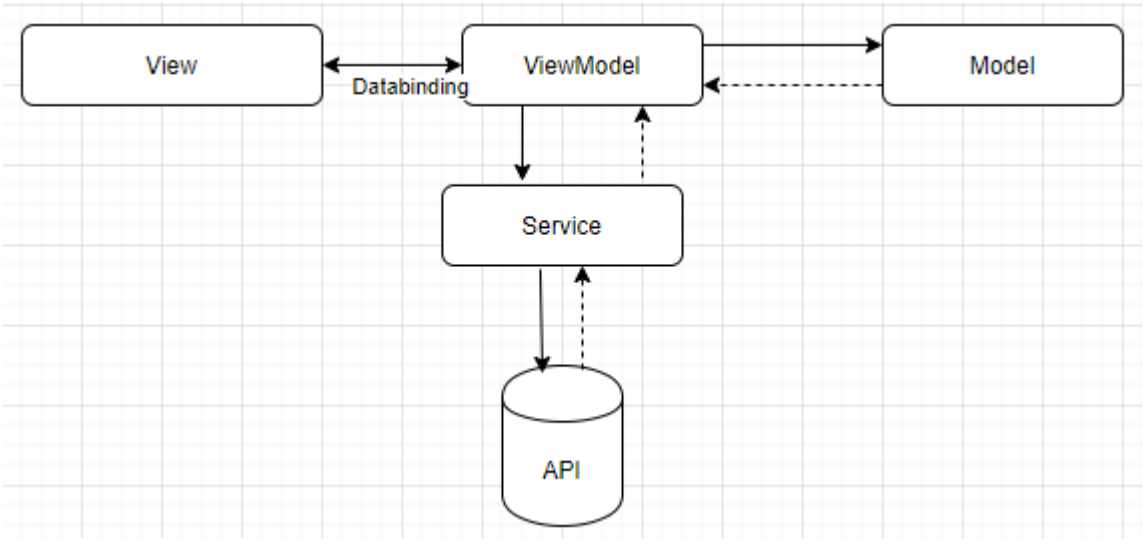


Összefoglaló

Az alkalmazás lehetővé teszi, hogy a felhasználó egy internetes szótárat, illetve egy szinonima szótárat egy alkalmazás keretein belül használjon. Ha fordítani akar, egy hosszú listából kiválaszthatja, hogy milyen nyelvről szeretne melyikre fordítani. Megadhat egy lefordítandó szót, de akár egy vagy több mondatot is. Ha valaminek a szinonimáját keresi, csak egy radio button-t kell, hogy átállítson.

Architektúra

Az alkalmazás a klasszikus MVVM modellre épül. Az én alkalmazásomban van egy plusz komponens, a Service is.



Fontosabb komponensek

Az ábrán látható komponensekről bővebben.

View

Ez az egység felelős a felületért. Ebben az alkalmazásban, egy felület intéz mindent. Egy Textbox-ból kéri be a lefordítandó szót/szavakat, vagy a szót, aminek a szinonimáját keresi a felhasználó. Miután a felhasználó kiválasztja két radio button közül, hogy a fordító vagy a szinonima fonkciót szeretné használni, az ahhoz tartozó combo boksz elérhetővé, lenyithatóvá válik. Ha mégis a másik funkcióra vált a felhasználó akkor az eddig aktív combo boksz inaktív lesz, a másik lesz lenyitható. Ha a felhasználó választott nyelvet vagy nyelv párt, a gomb megnyomásával meghívjuk a ViewModel megfelelő függvényét.

ViewModel

A View absztrakciója. Ez a komponens közvetíti az adatot a View és a Model között. Ez a View DataContext-je. A View-ban a gombnyomás hatására egy ViewModel-beli függvény hívódik meg, amely tovább hív egy Service osztály függvényt.

Service

Ez a komponens felelős a hálózati kommunikációért. Itt tevődik össze több részből az URL, ahonnan az API válaszát várjuk. Ennek mindkét funkciónál az első része egy egyéni API kulcs(key). Ezután hozzátevődik a keresendő szó, illetve a nyelv vagy nyelv pár. A szinonima hívásnál az URL legvégén jelzem, hogy a választ nem XML, hanem JSON formátumban várom. A View inicializálásánál szintén API hívásban kérem le a fordítónál lehetséges nyelv párokat. Mindegyik API hívásra a válasz egy string, amit deszerializásva JSON objektumot kapok, azt adom vissza a ViewModelnek.

Model

A modell definiálja az adatot, amit az alkalmazás használ. Ez a réteg nem tartalmaz WPF secifikus kódot, pl. ObservableCollection. A modell vázát egy próba hívás JSON vázából hoztam létre.

Használt API-k

Yandex Translate API

A Yandex nagyon egyszerű és könnyen használható lehetőség fordító API-nak. A fejlesztő igényelhet ingyenesen API kulcsot, egyetlen megkötés, hogy az ingyenek API kulccsal csak napi 2500 kérés hívható. Ezen projektnél ez teljes mértékben elég. Lehetőséget az egy olyan API hívásra, ami visszaadja az aktuálisan elérhető nyel párokat, hogy miről fordíthatunk mire., így később nem kell kézzel módosítani a listát, ha az változik.

Thesaurus API

Ez az API kevesebb lehetőséget ad, mint a Yandex, de az ingyen elérhető szinonima API-k közül ez volt a leghasználhatóbb. Nem ad lehetőséget az elérhető nyelvek lekérésére API hívással, így azt csak kézzel lehet karbantartani. Más szinonima API-kal szemben viszont igen részletes választ ad vissza. Megadja a szófajt, és egy szóra általában (más szinonima API-val ellentétben) számos szinonimát ad vissza.

Fordítás

Az alábbiakban részletesen áttekintjük, milyen lépések történnek, amikor egy szöveget fordításra elküld a felhasználó:

1. Kiválasztja a translation radio buttont, majd választ nyelvet a combo boxból. Ezek értéke globális változóban tárolódik. A gomb megnyomására, ha a beviteli mező nem üres meghívja a ViewModel megfelelő függvényét.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    if (is_translate_active)
    {
        if (string.IsNullOrEmpty(to_trans.Text))
        {
            mainPageViewModel.Result = "Type something to translate!";
        }else {
            string trans = to_trans.Text;
            mainPageViewModel.GetTranslation(trans, lan_pair);
        }
```

2. A ViewModel tovább hívja a megfelelő Service osztály függvényét.
3. A Service osztályban összerakásra kerül a megfelelő API hívás URL, majd ezt átadja egy aszinkron függvénynek, ami a HTTP kérést végzi.
4. Az aszinkron függvény a válasz stringet deszerializálja JSON objektummá.
5. A függvények visszatérnek, a visszakapot objektum megfelelő részét hozzáadom a megfelelő tárolóba.
6. Databinding-gal megjelenik a fordítás eredménye a Text block-on.

```
<TextBlock Text="{Binding Result, UpdateSourceTrigger=PropertyChanged}"
```

Források

A képek a https://www.diagrams.net/ segítségével készültek.