STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Měj přehled
Agregátor zpráv

Michal Bravanský

Moravskoslezský kraj

Bílovec 2020

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Měj přehled
Agregátor zpráv

Měj přehled
News aggregator

Autoři: Michal Bravanský
Škola: Gymnázium Mikuláše Koperníka
17. listopadu 526, 743 11 Bílovec
Kraj: Moravskoslezský
Konzultant: doc. Ing. Jan Platoš, Ph.D.

Bílovec 2020

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Bílovci dne 25.3.2020 ……

Michal Bravanský

Poděkování

Tímto bych chtěl poděkovat doc. Ing. Jan Platoš, Ph.D. za odborné vedení práce.

Anotace

Práce se zabývá vytvářením zpravodajského systému, který by fungoval zcela automaticky a publikoval důležité zprávy. Kvůli zaměření na mladší generaci byly jako platforma pro jejich distribuci zvoleny sociální sítě. Díky automatizaci je celý tento proces rychlý a objektivní.

Klíčová slova

zpravodajství;umělá inteligence;sociální sítě;fasttext;lda

Annotation

The thesis is concerned with development of news system, which would function fully automatically and publish important news. Because of its focus on the younger generation, social networks were chosen as the best platform for its news distribution. Thanks to its automation, this whole process is fast and unbiased.

Keywords

News;artificial intelligence;social networks;fasttext;lda

Obsah

1 Úvod 8
2 Požadavky systému 8
3 Použité technologie 8
4 Zpravodajské portály 10
5 Architektura 10
6 Návrh databáze 11
7 Stahování odkazů a článků 12
7.1 Loaders 12
7.1.1 GetUrls 12
7.1.2 GetArticles 14
7.2 Repository 15
7.3 Konzolová aplikace 18
8 Umělá inteligence 18
8.1 Vytvoření korpusu 18
8.1.1 Stop Words 19
8.1.2 Tokenizace 19
8.1.3 Lemmatizace 19
8.2 Nastavení modelu 20
8.3 Vycvičení modelu 20
9 Porovnávání článků 21
9.1 Připojení k databázi 21
9.2 Porovnání článků 21
9.2.1 Výběr nejdůležitějších témat 23
9.2.2 Získání tématu 23
9.2.3 Cvičení LDA 23
10 Stavba příspěvku 25
10.1 Generování příspěvků 26
11 Sociální sítě 27
11.1 Facebook 27
11.1.1 Registrace aplikace 27
11.1.2 App Review 27
11.1.2 manage_pages 28
11.1.3 publish_pages 28
11.1.4 Proces ověření 28
11.1.5 Publikace příspěvků 29
11.2 Twitter 30

11.2.1 Registrace aplikace 31
11.2.2 Publikace příspěvků 31
11.3 Instagram 32
11.3.1 OneUp 32
11.3.2 Formát RSS Feedu 32
11.3.3 Firebase server 33
11.3.4 Rest API 33
11.3.5 AddPost 33
11.3.6 RSSFeed 34
11.3.7 Publikace příspěvků 34
12 Závěr 36
13 Použitá literatura 37
14 Seznam obrázků 39

1 Úvod

Pojem média bychom mohli označit jako jeden z nejpoužívanějších pojmů poslední doby. Počátek její existence se datuje až do vynálezu knihtisku, který umožnil rychlou a levnou distribuci informací. Ale bude trvat ještě několik století, než budeme moci hovořit o prvních prototypech novin či letáků. (1)

Ve 20. století nastala pro média jedna z největších změn, postupný vynález rádia, televize a nakonec internetu. To zcela změnilo způsob, jakým konzumujeme zprávy. Najednou jsme byli schopni nalézt informace o nejnovějším dění ve světě za 2 kliknutí a možnost si vybrat z desítek zdrojů.

Ve 21. století ale stojí media proti nové výzvě a tou jsou sociální sítě. Tento fenomén zaplavil celou mladou generaci a jednotlivé zpravodajské portály zatím nenalezli způsob, jak správně na těchto nových platformách zprávy distribuovat. Mezi jejich cíle totiž nepatří vydávání zpráv, ale pouze generování kliknutí, které jim přinášejí zisky z reklam. Proto je pro ně výhodnější sdílet pouze odkaz na vlastní článek, než ve svých příspěvcích na sociálních sítích informovat o nejnovějších zprávách své publikum.

Z těchto důvodů už v Česku vzniklo několik studentských iniciativ, které se toto snažili změnit, tedy vytvořit nekomerční zpravodajské kanály na sociálních sítích, například bych uvedl Žbluňk[1] anebo Politika (nejen) pro mladé[2]. Ale tyto projekty jsou postaveny na dobrovolnictví, můžou kdykoliv přestat fungovat.

2 Požadavky systému

Cílem je vytvořit systém, který je schopen ze zpravodajských portálů stahovat články, ty poté mezi s sebou porovnávat a nakonec nejdůležitější zprávy publikovat na sociálních sítích.

1. Program by měl být schopen stahovat články z vybraných medií. To znamená, že dokáže načíst nejnovější články vydané daným zpravodajstvím a poté bude schopen zpracovat HTML kód téhož článku a vytáhnout z něho potřebné údaje a celkový text.
2. Program by měl poté dokázat načtené články porovnávat. Cílem je zjistit, kolikrát se denně nějaké téma opakovalo, protože toto téma poté bude relevantnější k danému dnu.
3. Vybrané témata zpracovat do tvaru příspěvku, který se nejvíce hodí na vybranou sociální síť.
4. Publikace samotných příspěvků.

3 Použité technologie

C#

C# je vysokoúrovňový objektově orientovaný jazyk vyvinutý firmou Microsoft a založený na jazycích C++ a Java. V dnešní době se využívá hlavně ke tvorbě desktopových aplikací pro operační systém Windows, ale lze ho také použít k vývoji webových (ASP.NET) anebo mobilních aplikací (Xamarin). (2)

[1] https://zblunkmedia.com/
[2] https://www.instagram.com/politika_nejen_pro_mlade/?hl=cs

Python

Python je interpretovaný, objektově orientovaný jazyk. Skládá se z open-source modulů a balíčků. (3) Používá se většinou k vědeckým a matematickým výpočtům, kde se právě využívá jeho jednoduchost a modulárnost. Dále ho ale můžeme využít k vývoji webů (Django) či u menších business aplikací. (4)

JavaScript

JavaScript je multiplatformní, objektově orientovaný, událostmi řízený skriptovací jazyk, jehož autorem je Brendan Eich z tehdejší společnosti Netscape. I když se syntaxí řádí mezi jazyky C/Java, zásadně se od nich liší a jedná se zcela odlišný jazyk, slovo Java se vyskytuje v názvu pouze z komerčních důvodů. (5).

Nejčastěji se JavaScript používá na klientské straně webu, kde je interpretovaný samotným prohlížečem. To znamená, že webová stránka nemusí být sestavená pouze ze statického HTML, ale může obsahovat programy, které interagují s uživatelem, komunikují s prohlížečem anebo dynamicky tvoří nový HTML obsah. (6)

Node.js

Jedná se o open-source prostředí designované ke tvorbě webových aplikací a spouští programy napsané v JavaScriptu. K jeho vytvoření došlo při snaze přeměnit JavaScript, jazyk využívaný hlavně na webových stránkách, na tvorbu samostatných aplikací, které by mohly běžet na desktopovém počítači.

Node.js je jednoduchý a efektivní. Jeho balíčkový ekosystém, npm, je také největší archív open-source knihoven na světě. (7)

Firebase

Firebase je platforma pro tvorbu webových a mobilních aplikací, kterou využívá přes 1.5 miliónů aplikací. V současné době ji vlastní Google. Je využívána především startupy a menšími firmami, neboť veškerý hardware se nachází na straně Googlu, takže při programování ho nemusí řešit. V momentální době nabízí přes 15 produktů, například Firebase Database sloužící k úschově dat a Firebase Hosting na hostování vlastní webové stránky. (8)

RSS Feed

RSS (Rich Site Summary) je rodina XML formátů využívaných k distribuci obsahu na webových stránkách. Uživateli umožňuje nastavit odběr obsahu z webových kanálů (např. facebook, twitter,…) v přehledném XML formátu a ty poté dále zpracovávat. (9)

SQL

SQL je standardizovaný programovací jazyk používaný na práci s databázemi a provádění různých operacích na nich. Vyvinuta v 70. letech 20. století, se stal z SQL de facto standardní programovací jazyk pro relační databáze. (10)

Před SQL měly databáze těžké a nepraktické programovací rozhraní. Jedinou jejich výhodou byla jejich relativní vysoká rychlost vykonávání příkazů. Ale s nástupem rychlejších počítačů tohle přestalo být problémem a vývojáři se zaměřili spíše na snížení ceny vývoje a udržování takové databáze v chodu.

A z těchto důvodů vzniklo SQL, které je díky své jednoduchosti využívané po celém světě. Momentálně existuje několik systémů, které implementují SQL, například Microsoft SQL Server, Oracle Database, MySQL a tak dále. (11)

FastText

FastText je knihovna na modelování jazyka a kvalifikaci textu vytvořená AI Research laboratoří řízenou Facebookem. Model umožňuje provádět algoritmy na získání vektorové reprezentace slov. (12)

Slova jsou v takovém případě přeměněny na vektory, které reprezentují projekce slov do nekonečného vektorového prostoru, kde každá dimenze zachycuje jednotlivý význam slova a jeho relaci k ostatním slovům. Hlavní výhodou takové reprezentace je možnost provádět se slovy matematické operace, například sčítání či odečítání vektorů. (13)

REST

Representational state transfer (REST) je architektura definující sadu pravidel, podle kterých se poté vytváření webové service. Tyto service poté poskytují propojení mezi počítači a internetem. Umožňují totiž systémům přistupovat a manipulovat s textovém reprezentací webových resources a to voláním několika různých metod, například GET, POST, PUT anebo DELETE. (14)

Entity Framework

Entity Framework je open source struktura umožňující mapování dat z databáze na objekty daného programovacího jazyka. Dovoluje programátorovi pracovat s daty ve formě objektů a jejich vlastností, bez nutnosti zabývat se databázovými tabulkami a sloupci, kde jsou data uložena. (15)

4 Zpravodajské portály

Jako výchozí portály byly vybrány tyto: IDNES, Aktuálně, Novinky a Deník. Jedná se o nejčastěji navštěvované média (16). Dále se jejich stránky liší přehlednějších a velikostně menším HTML kódem na rozdíl od jiných portálů. Také se pyšní dlouhou historií, tedy velkou databází článků, kterou program může stáhnout a použít jako korpus na trénování umělé inteligence.

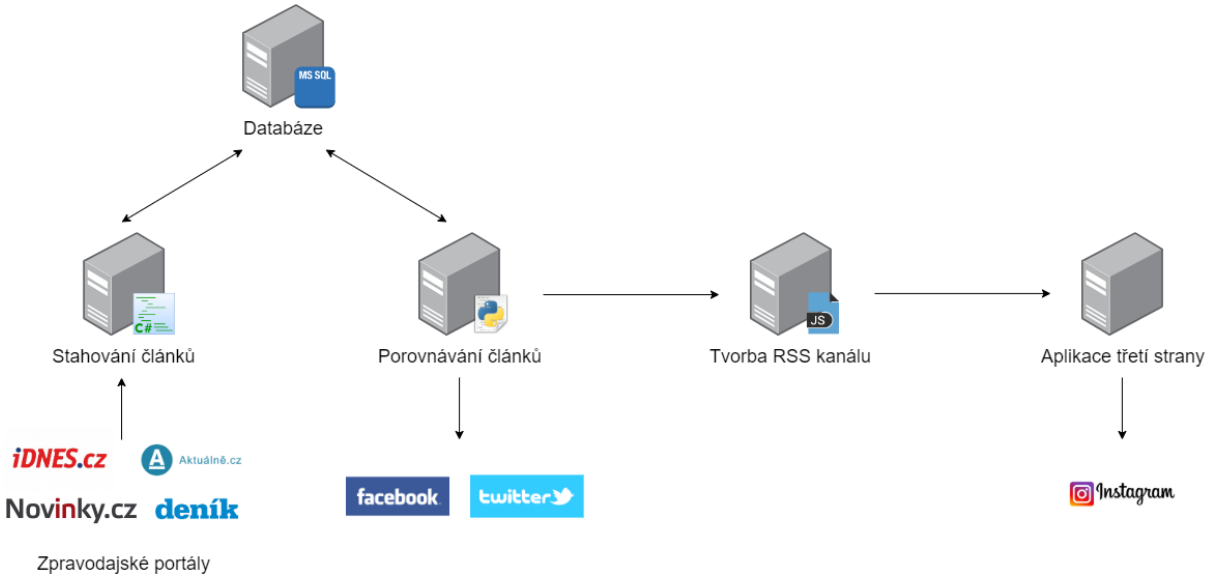
5 Architektura

Systém je rozdělený do několika částí, které spolu pracují paralelně. To nám umožňuje efektivněji pracovat se staženými články a taky zajišťuje lehkou portabilitu, kdy jakoukoliv část lze bez problému využitím v dalších projektech, anebo lze kdykoliv přidat novou funkcionalitu do již stávajícího systému.

Skládá se ze 4 částí. První dvě se nacházejí ve stejném řešení, obě sepsané v C#, ve dvou různých projektech. V řešení se také nachází knihovna s repositáři sestavenými pomocí Entity Frameworku, které propojují programy s databází. Tyto dvě konzolové aplikace mají za úkol načíst všechny nově vydané články a uložit je do databáze. Druhá část tyto odkazy poté otevře a stáhne veškerý obsah.

Třetí částí je projekt sepsaný v Python, který má za úkol spojovat stažené články do asociací. Stáhne si tedy články vydané za poslední dobu a porovná je s ostatními. Po nalezení podobnosti pomocí REST API publikuje příspěvky na sociální sítě.

Protože má ale Instagram uzavřené API, program obsahuje taky čtvrtou část, což je Node.js server. Ten vytváří z přidaných příspěvků RSS Feed, který je předáván aplikaci třetí strany, která má oprávnění na vydávání příspěvků na Instagram a publikuje je.



Obrázek 1 - Diagram systému

6 Návrh databáze

Pro návrh databáze byl použit Microsoft SQL server díky svému jednoduchému propojení s dalšími Microsoft technologiemi, hlavně tedy se C#.

Databáze se skládá z několika tabulek, celý model je vidět v příloze č. 1. První tabulka je Article, která má v sobě uložené veškeré načtené články a informace o nich. Mezi tyto informace patří odkaz, zdroj, datum vydání, titulek a odkaz na úvodní obrázek. Také se v ní nachází sloupec Downloaded, který udává, jestli byl odkaz už otevřen a program už stáhl jeho obsah.

Veškerý text ze článků se tedy přidává do jiné tabulky - Paragraph. V ní se nachází seřazené odstavce, spojené pomocí cizího klíče s jednotlivými články. Články jsou poté spojené do asociací, tabulka Association, která sdružuje články s podobným tématem.

Mezi poslední tabulky patří Connection, která spojuje asociace s články a SourceType, která udává zdroje článků.

7 Stahování odkazů a článků

Nejprve musí program nalézt odkazy k článkům. Protože většina zpravodajských portálů nemá otevřené API na stahování článků, program byl postaven na získávání odkazů přímo z HTML. Stáhne tedy stránku s nejnovějšími články a postupně prochází HTML a ukládá všechny odkazy.

Jedna z dalších možností, co použít místo parsování HTML, je samotný RSS feed jednotlivých portálů. Ale tyto feeds zobrazují pouze nedávno vydané články. To znamená, že program RSS zdroje nemůže využít na tvorbu korpusu a opravdu je jediným řešením parsování HTML.

7.1 Loaders

Pro každý webový portál je v programu vytvořený tzv. Loader, který má za úkol z vybraného portálu získávat data. Protože každá stránka obsahuje jinak strukturované HTML, musí existovat vždy samostatný loader pro každého vydavatele.

Každý loader se skládá ze dvou veřejných metod, GetUrls a GetArticles, kdy jedna vrací odkazy a druhá přijímá odkazy a vrací stažené články. Aby šlo s loaders lépe pracovat, všechny dědí z jednoho interface ILoaders a nachází se v EFUnitOfLoaders, která je poté předávána jako Service do konzolových aplikací.

```
public class EFUnitOfLoaders:IUnitOfLoaders
{
    public AktualneLoader AktualneLoader { get; set; }
    public DnesLoader DnesLoader{ get; set; }
    public NovinkyLoader NovinkyLoader{ get; set; }
    public DenikLoader DenikLoader{ get; set; }

    public EFUnitOfLoaders()
    {
        IUnitOfWork unitOfWork = new EFUnitOfWork();
        AktualneLoader = new AktualneLoader("https://zpravy.aktualne.cz/", "", unitOfWork.SourceTypeRepository.Get(x=>x.Code==Sources.Aktualne.ToString()));
        DnesLoader = new DnesLoader("https://www.idnes.cz/", "zpravy/", unitOfWork.SourceTypeRepository.Get(x => x.Code == Sources.Dnes.ToString()));
        NovinkyLoader = new NovinkyLoader("https://www.novinky.cz/stalo-se", "", unitOfWork.SourceTypeRepository.Get(x => x.Code == Sources.Novinky.ToString()));
        DenikLoader = new DenikLoader("https://www.denik.cz/", "zpravy/", unitOfWork.SourceTypeRepository.Get(x => x.Code == Sources.Denik.ToString()));
    }
```

Obrázek 2 - Ukázka EFUnitOfLoaders

Občas může existovat článek, který načíst neumí, třeba živý přenos nebo grafika. A proto je loader rozdělený na dvě části, neboť v takovém případě se samotný článek nikdy nestáhne, ale první část programu je aspoň schopna tento odkaz uložit do databáze.

7.1.1 GetUrls

Tato metoda jako parametr přijímá datum, do kterého má načíst veškeré články. Tento systém je zavedený v aplikaci z důvodu, že často na stránkách nejsou články seřazené podle data vydání, ale relevantnosti, takže často může být jeden článek zobrazovaný jako první několik hodin, i když portál vydal už několik desítek dalších článků.

Program stáhne HTML do HTML agility pack[3], který umožňuje poté pracovat s HTML v C# podobným způsobem jako v JavaScriptu, tedy například pomocí čtení a psaní DOM a podpory XPATH nebo XSLT. (17)

[3] https://github.com/zzzprojects/html-agility-pack

Odkazy v HTML poté nalezneme buď podle stejného rodiče, nebo třídy. Vybereme je a uložíme do hashovací tabulky. Program pokračuje na další stránky a ukládá další a další odkazy, dokud nenarazí na odkaz, který není v očekáváném intervalu.

Ale z odkazů čas vydání získat nejde a v HTML také není nikde uložený. Zůstávají tedy 2 možnosti, buď otevřít odkaz a načíst čas vydání tam, anebo použít přibližnou hodnotu přiloženou k odkazu v HTML. Opakované otvírání odkazů by zabralo obrovské množství času, takže je lepší používat přibližnou hodnotu u článků. Tento text nejdříve přemění na DateTime a ten se porovná s mezní časovou hodnotou, která byla do programu vložena jako parametr.

```
public List<Article> getUrls(DateTime to, int offset=0)
{
    HtmlWeb client = new HtmlWeb();
    HashSet<string> urls = new HashSet<string>();
    while (true)
    {
        HtmlDocument doc = client.Load(StartUrl + Section + "?offset=" + offset);
        var nodes = doc.DocumentNode.Descendants(0).Where(n => n.Name == "a" && n.ParentNode.HasClass("filter-b-site-aktualne"));

            if (nodes.Count()==0)
                return urls.Select(x=>new Article { Url=x,Downloaded=false,SourceType_ID=Source.SourceType_ID,CreatedDate=DateTime.Now }).ToList();
            if (offset == 0)
            {
                var list = nodes.ToList();
                list.RemoveAt(0);
                nodes = list;
            }
        foreach (var item in nodes)
        {
            var url = item.GetAttributeValue("href", null);
            var fullURL = CheckIfFullAdress(url) ? url : StartUrl + url.Substring(1);
            var datetime = ConvertStringToDatetime(HtmlUtilities.ConvertToPlainText(item.ParentNode.
                    ParentNode.Descendants(0).Where(x => x.HasClass("timeline__label")).FirstOrDefault().InnerText).
                    Replace("\n", string.Empty).Replace("\t", string.Empty));
            if (datetime < to)
                return urls.Select(x => new Article
                {
                    Url = x,
                    Downloaded = false,
                    SourceType_ID = Source.SourceType_ID,
                    CreatedDate = DateTime.Now,
                    LastChange = DateTime.Now
                }).ToList();
            urls.Add(fullURL);

        }


        offset += 20;
    }
```

Obrázek 3 - Stahování odkazů z Aktuálně

```
public DateTime ConvertStringToDateTime(string dateTime)
{
    var split = dateTime.Split(' ');

    if (split[0] == "dnes")
    {
        var time = split[1].Split(':').Select(x => int.Parse(x)).ToArray();
        return new DateTime(DateTime.Now.Year, DateTime.Now.Month,DateTime.Now.Day, time[0], time[1], 0);
    }
    else if (split[0] == "včera")
    {
        var time = split[1].Split(':').Select(x => int.Parse(x)).ToArray();
        return new DateTime(DateTime.Now.AddDays(-1).Year, DateTime.Now.AddDays(-1).Month, DateTime.Now.AddDays(-1).Day, time[0], time[1], 0);
    }
    else
    {
        switch (split[0])
        {
            case "pondělí":
                return DateTime.Now.AddDays(-(int) DateTime.Now.DayOfWeek+1);
            case "úterý":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek+2);
            case "středa":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek + 3);
            case "čtvrtek":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek + 4);
            case "pátek":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek + 5);
            case "sobota":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek + 6);
            case "neděle":
                return DateTime.Now.AddDays(-(int)DateTime.Now.DayOfWeek + 7);
            default:
                return DateTime.ParseExact(dateTime, "d. M. yyyy", CultureInfo);
        }
    }
}
```

Obrázek 4 - Určování přibližného času vydání článku z textu u Novinky

7.1.2 GetArticles

Do metody se vkládají odkazy a vrací už stažené články společně s odstavci. Program jako v předchozí metodě stáhne HTML, v tomto případě článku, a vloží ho do HTML agility pack. Z něho poté vytáhne veškeré potřebné údaje.

Nejprve načte všechny paragrafy, které obsahují text článku a ten poté po odstavcích uloží. Samotný text musí být z HTML formátu převeden na normální string. Ještě také najde titulek, Url obrázku a nakonec i datum vydání. Webovému uživateli se ale většinou nezobrazuje přesné datum vydání, takže program nalezne script, ve kterém je datum vydání obsaženo, a pomocí Regex technologie ho vytáhne.

Když program nedokáže článek načíst, přeskočí ho. Občas se stane, že se nepodaří programu navázat SSL tunel či vytvořit bezpečné spojení se serverem. V takovém případě program přejde na 1 minutu do spánku a poté se znovu spustí.

```
public List<Article> GetArticles(List<Article> articles)
{
    HtmlWeb client = new HtmlWeb();
    Regex timeRegex = new Regex("'publishedDateTime': \"(.+)\"");
    for(int count=0;count< articles.Count;count++)
    {
        try
        {
            var article = articles[count];
            HtmlDocument doc = client.Load(article.Url);
            var articleNode = doc.GetElementbyId("copylink-content");

            //precise time selector
            var script = doc.DocumentNode.SelectSingleNode("//script[contains(text(), 'publishedDateTime')]").InnerHtml;
            var match = timeRegex.Match(script).Groups[1].Value.Split('+')[0].Replace("T", " ");
            article.ReleasedDate = DateTime.ParseExact(match, "yyyy-MM-dd HH:mm:ss", CultureInfo);

            article.Title = HtmlUtilities.ConvertToPlainText(articleNode.ChildNodes.Where(n => n.HasClass("article-title")).FirstOrDefault().InnerHtml);
            var image = doc.DocumentNode.Descendants(0).Where(x => x.HasClass("article-multimedium")).FirstOrDefault();
            article.ImageUrl = image != null ? image.Descendants(0).Where(x => x.Name == "img").FirstOrDefault().GetAttributeValue("src", null) : null;
            var allParagraphs = articleNode.Descendants(0).Where(n => n.Name == "p" || n.Name == "h2").ToArray();
            var texts = allParagraphs.Select(n => n.InnerHtml).Select(n => HtmlUtilities.ConvertToPlainText(n)).ToList();
            texts.RemoveAt(texts.Count - 1);
            article.Paragraphs = texts.Select((x,i)=>new Paragraph { Text=x,Position=i, Article_ID = article.Article_ID }).ToArray();
            article.Downloaded = true;
        }
        catch
        {
            //skip article
            articles.RemoveAt(count);
            count--;
        }
    }

    return articles;
}
```

Obrázek 5 - Stahování článků z Deníku

7.2 Repository

Pro práci s Entity Framework se většinou využívají Repositories, které omezují programátorovi přístup k databázi a tím dělají kód přehlednější a bezpečnější.

V projektu je vytvořena RepositoryBase, která obsahuje metody, kterými program komunikuje přes DbContext s databází. Mezi ně patří například Add, Get, Update a GetMany. Pro jednotlivé tabulky jsou vytvořeny repositáře, které dědí právě z RepositoryBase.

```
public abstract class RepositoryBase<TEntity> : IRepository<TEntity> where TEntity : class
{
    public DbContext DbContext { get; set; }

    public RepositoryBase(DbContext DbContext)
    {
        this.DbContext = DbContext;
    }

    public virtual void Add(TEntity entity)
    {
        DbContext.Set<TEntity>().Add(entity);
    }

    public void Delete(TEntity entity)
    {
        DbContext.Set<TEntity>().Remove(entity);
    }

    public virtual TEntity Get(Expression<Func<TEntity, bool>> predicate = null, params string[] includeProps)
    {
        return GetManyQueryable(predicate).FirstOrDefault();
    }

    public async Task<TEntity> GetAsync(Expression<Func<TEntity, bool>> predicate = null, params string[] includeProps)
    {
        return await Task.FromResult(Get(predicate));
    }

    public virtual List<TEntity> GetMany(Expression<Func<TEntity, bool>> predicate = null, params string[] includeProps)
    {
        return GetManyQueryable(predicate).ToList();
    }
```

Obrázek 6 - Ukázka RepositoryBase

V programu lze nalézt tyto Repository:

1. ArticleRepository
2. AssociationRepository
3. ConnectionRepository
4. ParagraphRepository
5. SourceTypeRepository

```
public class ArticleRepository : RepositoryBase<Article>
{
    public ArticleRepository(DbContext DbContext) : base(DbContext)
    {
    }

    public override Article Get(Expression<Func<Article, bool>> predicate, params string[] includeProps)
    {
        return GetManyQueryable(predicate).Include(x => x.SourceType).FirstOrDefault();
    }

    public override List<Article> GetMany(Expression<Func<Article, bool>> predicate, params string[] includeProps)
    {
        return GetManyQueryable(predicate).Include(x => x.SourceType).ToList();
    }
}
```

Obrázek 7 - Ukázka Repository pro články

Ty jsou zapouzdřeny v EFUnitOfWork, který je jako Service posílán do konzolových aplikací. V něm program přistupuje k jednotlivým repositářům. Při zlikvidování EFUnitOfWork se změny uloží do databáze.

```
public class EFUnitOfWork : IUnitOfWork
{
    private bool disposed;

    public DbContext DbContext { get; set; }

    public IRepository<Article> ArticleRepository { get; set; }

    public IRepository<Paragraph> ParagraphRepository { get; set; }

    public IRepository<SourceType> SourceTypeRepository { get; set; }

    public IRepository<Association> AssociationRepository { get; set; }

    public IRepository<Connection> ConnectionRepository { get; set; }

    public EFUnitOfWork()
    {
        DbContext = new DbEntities();
        DbContext.Database.CommandTimeout = 1000;
        ArticleRepository = new ArticleRepository(DbContext);
        ParagraphRepository = new ParagraphRepository(DbContext);
        SourceTypeRepository = new SourceTypeRepository(DbContext);
        AssociationRepository = new AssociationRepository(DbContext);
        ConnectionRepository = new ConnectionRepository(DbContext);
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    protected void Dispose(bool disposing)
    {
        if (disposing)
        {
            if (!disposed)
            {
                if (DbContext.ChangeTracker.Entries().Any(e => e.State == EntityState.Added
                || e.State == EntityState.Modified || e.State == EntityState.Deleted))
                {
                    DbContext.SaveChanges();
                }

                DbContext.Dispose();
            }
        }

        disposed = true;
    }

    public void Save()
    {
            DbContext.SaveChanges();
    }
}
```

Obrázek 8 - Ukázka EFUnitOfWork

7.3 Konzolová aplikace

V projektu se nacházejí dvě konzolové aplikace, jedna na načítání odkazů a druhá ke stahování článků.

První má za úkol načíst odkazy. Z Services dostane UnitOfLoaders, kde načte všechny články vydané za posledních 6 hodin. Ty poté vloží do databáze, přičemž kontroluje, jestli už článek se stejným odkazem nebyl vložen do databáze a tím pádem v tabulce nebyla duplicitní data.

Druhá konzolová aplikace vybere zatím nestažené články přidané za poslední den a pomocí UnitOfLoaders je stáhne.

8 Umělá inteligence

Dalším krokem bylo vytvořit algoritmus na porovnávání textu. Podle četnosti daného tématu bychom mohli určit jeho relevantnost k danému období. Existuje několik způsobů, jak texty porovnávat, ale v tomto případě byl zvolen FastText.

8.1 Vytvoření korpusu

Pro jakoukoliv umělou inteligenci používanou na zpracování jazyka je potřeba mít dostatečně velký korpus, tedy text, kde se daný jazyk používá. Protože program porovnává články, musí se učit z textů, kde se používá podobná slovní zásoba, tedy z dalších článků.

Z těchto důvodů taky program využívá stahování článků přímo přes HTML, díky čemuž může získat veškeré články vydané daným zpravodajským kanálem.

Po 20 dnech dokázal program stáhnout půl miliónů článků, což je přibližně 1 miliarda slov, čímž se řadí mezi jedny z největších českých publicistických korpusů kdy vytvořených. V porovnání největší český korpus obsahuje 4,5 miliard slov. (18)

Výhodou vytvořeného korpusu je, že pochází přesně z portálů, z kterých bude program stahovat články. Díky tomu bude v korpusu obsažena stejná terminologie a slovní zásoba jako ve vstupních textech a bude díky tomu lépe fungovat, než kdyby se program učil na jiném korpusu.

Veškerý obsah databáze je exportován do textových souborů, ze kterých se poté model učí. Tato data jsou rozdělená do 50 souborů, zbavena všech tzv. stop words a celý text je před uložením ztokenizován a lemmizován.

```
EFUnitOfWork unitOfWork = new EFUnitOfWork();
var allArticles = unitOfWork.ArticleRepository.GetMany(x=>x.Downloaded);

for (int count = 0; count < allArticles.Count; count += allArticles.Count / 50)
{
    using (StreamWriter STW = new StreamWriter("data/file" + count/ (allArticles.Count / 50)))
    {
        var item = allArticles[i];
        if (item.Downloaded)
        {
            STW.Write(item.Title + ".");
            foreach (var paragraph in item.Paragraphs.OrderBy(x => x.Position))
            {
                STW.Write(" " + paragraph.Text);
            }
            STW.WriteLine();
        }
    }
}
```

Obrázek 9 - Projekt pro exportování článků

8.1.1 Stop Words

Stop Words je soubor často používaných slov, v češtině se třeba jedná o slova „nebo“ a „on“. Nevytvářejí žádnou přidanou hodnotu a jejich vektory by se navzájem vynulovaly, takže není žádný důvod, proč by je měl text obsahovat. (19)

8.1.2 Tokenizace

Tokenizace je proces, při kterém je text rozdělený to tzv. tokenů, což je nejmenší jednotka textu. (20) K provedení této akce program využívá open-source python knihovnu nltk[4].

8.1.3 Lemmatizace

Jedná se o proces, při kterém se heslo převede na jeho reprezentativní slovní podobu, tzv. lemmu. (21) V programu je použita knihovna Majka vyvinutá na MUNI[5].

[4] https://www.nltk.org/
[5] https://github.com/petrpulc/python-majka

```
def preprocess_text(document):
        # Remove all the special characters
        document = re.sub(r'\W', ' ', str(document))

        # Converting to Lowercase
        document = document.lower()

        # Tokenization
        tokens = document.split()
        tokens = [word for word in tokens if word not in get_stop_words('czech')]

        # Lemmatization
        tokens=[morph.find(x) for x in tokens]

        preprocessed_text = ' '.join(tokens)

        return preprocessed_text
```

Obrázek 10 - Ukázka tokenizace a lemmatizace

8.2 Nastavení modelu

Model cvičíme na korpusu o velikost 500 tisíc článků vydaných za posledních 20 let. Program postupně načítá 50 souborů, ve kterých je korpus rozložený a postupně ukládá model při každém pátém. Několikrát se stalo, že kvůli vysoké paměťové náročnosti program vyhodil chybu, takže bylo nutné postupně ukládat modely. Kromě toho nám to také dovoluje porovnávat modely z různých fází učení.

Model má následující parametry, velikost jednotlivých vektorů je 200 dimenzí. Počet dimenzí udává, kolik informací vektory dokáží držet, proto se k větším korpusům hodí větší počet dimenzí a naopak. Z několika pokusů bylo zjištěno, že nejlépe v našem případě pracují 200 dimenzionální vektory. To je způsobeno hlavně velkou velikostí korpusu a velkými tematickými rozdíly mezi články, kdy je slovní zásoba v nich opravdu velká.

Dále má vlastnost min_words, která udává, kolik nejméně slov musí mít věta, aby se z ní umělá inteligence mohla učit významy slov. Bylo vyzkoušeno zase několik možností a nejlépe osvědčilo používat všechny slova. Používání všech vět, i například dvouslovných pořád rozšiřuje slovní zásobu modelu a i kratší věty se také objevují ve zpravodajských článcích.

Windows_size je parametr, který udává, kolik okolních tokenů je relevantních k jednomu tokenu. Pokud nemáme ucelený text o jednom tématu, tato hodnota musí být menší, protože okolní slova nejsou až tak relevantní. Ale pokud máme články, které většinou píší o jednom tématu, okolní slova jsou hodně důležitá pro pochopení významu jednoho tokenu. Proto byla zvolena hodnota 40, protože umožňuje při zpracování tokenu tvořit vztahy s více okolními tokeny.

8.3 Vycvičení modelu

Model se cvičil v Python programu a za pomocí knihovny Gensim[6].

Model se učil přibližně 10 dní, za které prošel 1 miliardu slov. Následně byl tento model uložen ve formátu Word2Vec pro použití v dalších částech aplikace.

[6] https://radimrehurek.com/gensim/about.html

```
for i in range(0,50):
        with open('results/tokenized'+str(i),encoding="utf8") as f:
                content = f.readlines()
        word_tokenized_corpus = [x.strip().split() for x in content]
        embedding_size = 200
        window_size = 40
        min_word = 1
        down_sampling = 1e-2

        if(i==0):
                ft_model = FastText(word_tokenized_corpus,
                      size=embedding_size,
                      window=window_size,
                      min_count=min_word,
                      sample=down_sampling,
                      sg=1,
                      iter=100)
        else:
                ft_model.build_vocab(word_tokenized_corpus, update=True)
                ft_model.train(word_tokenized_corpus, total_examples=len(word_tokenized_corpus), epochs=ft_model.epochs)

        if i%20==0:
                ft_model.save("models/model"+str(i))
```

Obrázek 11 - Cvičení modelu

9 Porovnávání článků

Systém načte veškeré články vydané za posledních 6 hodin a převede je do jejich vektorových formátů. Poté tyto články mezi sebou porovná, a pokud nalezne podobné téma, vytvoří příspěvek, který sdílí na sociální sítě.

Kvůli opakovanému použití FastText modelu byl zvolen jako programovací jazyk Python, který pro operace s modelem poskytuje největší zázemí.

9.1 Připojení k databázi

Nejprve se program musí připojit k databázi a vytáhnout z ní články. K tomu využívá Python knihovnu pypyodbc[7], která umožňuje komunikovat s SQL serverem na základě SQL příkazů. Získaná data se vkládají do knihovny pandas [8], která data přehledně zpracuje do tabulky, složené z polí a slovníků, která programu umožňuje s daty jednoduše pracovat.

9.2 Porovnání článků

Pomocí knihovny spacy[9] se načte model vytvořený v předchozí části systému.

Texty ze článků jsou ztokenizovány a lemmatizovány, a posléze vloženy do spacy, která je pomocí modelu převede na jejich vektorové ekvivalenty. Program je poté ukládá společně s informacemi o článcích v poli objektů.

Články se mezi sebou porovnávají, přičemž systémem každý s každým. I když to znamená, že se pro **n** článků musí provést $\mathbf{n^2}$ porovnání, u tak malého počtu článků, přibližně 200, to není problém.

[7] https://pypi.org/project/pypyodbc/
[8] https://pandas.pydata.org/
[9] https://spacy.io/

Při porovnání spacy vrací hodnotu od 0 do 1, která udává, jak jsou vektory kosinově vzdálené. Čím blíže jedničce, tím jsou si články podobnější.

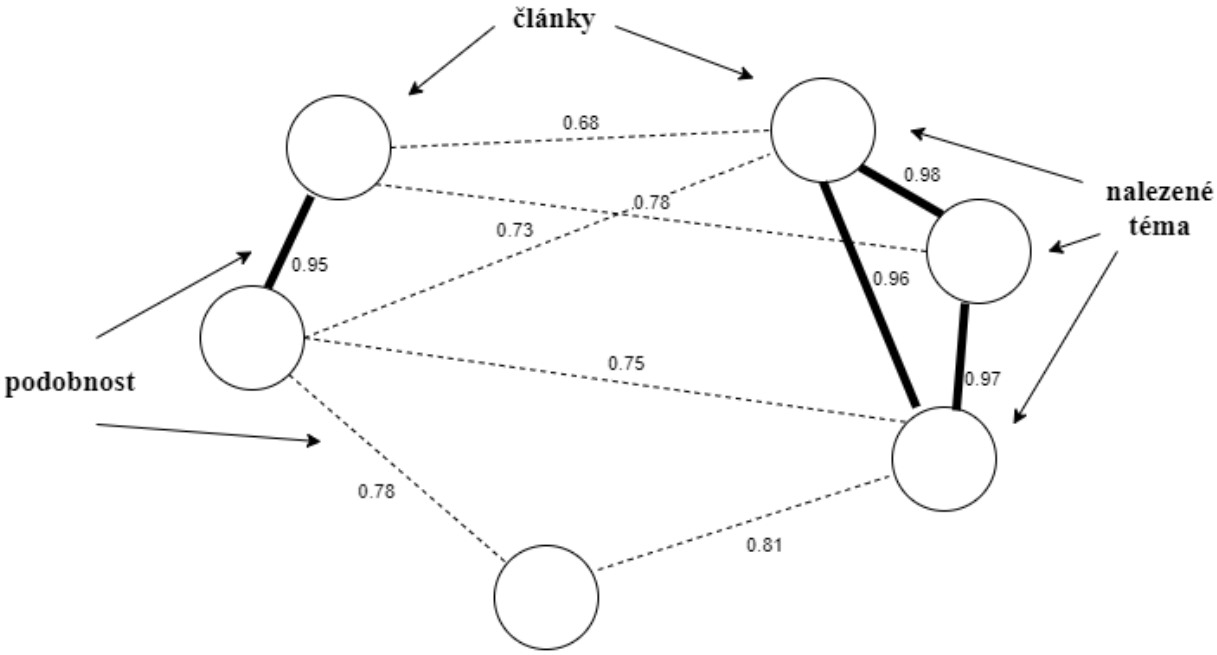
demonstrace	protest	0.419
mír	válka	0.027
vláda	premiér	0.357
unie	evropa	0.282

Obrázek 12 - Ukázka kosinové vzdálenosti vybraných slov podle vlastního FastText modelu

Protože při spojování vektorů a jejich následném porovnaní zaleží, aby byl počet vektorů na obou stranách stejný, program z každého článku vybere prvních 100 tokenů, které potom porovnává.

Kromě toho je u porovnávání ještě implementována další logika, protože kromě samotného textu můžou podobnost článků značit i další faktory, například čas vydání a délka. I toho se snaží program využít, a to tím, že k podobnosti přičítá dodatečné hodnoty, pokud články byly vydané ve stejném časovém období nebo jsou podobně dlouhé.

Tato data si lze představit jako graf, kde články jsou vrcholy a velikosti hran udávají, jak si jsou 2 články podobné. Úkolem programu je najít v tomto grafu podgrafy, které jsou spolu nějak spojené, tedy vzájemná vzdálenost vrcholů je co největší a přitom podgraf obsahuje co nejvíce vrcholů.



Obrázek 13 - Ukázka grafu článků

Existuje několik algoritmů na lezení těchto podgrafů, ale nejvíc se osvědčilo jednoduché prohledávání. Zvolíme si číslo K mezi 0 a 1. Teď z grafu odstraníme všechny hrany, které jsou menší než K. Vznikne nám soubor nespojeným podgrafů. Ty teď program může pomocí prohledávání do hloubky nalézt a uložit. Tímto algoritmem lze v O(n) nalézt všechny podgrafy podobných článků, přičemž celý algoritmus má složitost $o(n^2)$.

```
def getSubgraphs(id):
    if(visited[id]==True):
        return []
    result=[id]
    saved=topic[id].copy()
    visited[id]=True
    for i in saved:
        result=result + list(set(getSubgraphs(i)) - set(result))
    return result
```

Obrázek 14 - Ukázka získávání podgrafů pomocí rekurze

Tyto podobnosti se zanesou do databáze. Při dalším spuštění tohoto algoritmu jsou už zařazené články ignorovány, aby se program vyvaroval publikování příspěvků o stejných tématech.

Takových souborů článků vznikne v průběhu dne přibližně 20, takže je potřeba určit ty nejdůležitější, které se nakonec publikují na sociálních sítích.

9.2.1 Výběr nejdůležitějších témat

Program postupuje podle několik kritérií při valuaci daného tématu. První znakem relevance daného tématu je počet článků, ve kterých je zmíněn. Čím více článků zmiňuje téma, tím je téma důležitější.

Další charakteristikou relevantnějších témat je to, že je zmiňují různé zpravodajské portály. Program je tedy sestaven takovým způsobem, aby taková témata více preferoval.

Systém určí hodnotu tématu následovně:

1. Jeden bod za každý článek
2. Dva body za každý jedinečný zpravodajský server

Pokud hodnota tématu je aspoň 8, zveřejní se příspěvek s daným tématem. To znamená, že musí vyjít aspoň 3 články se stejným tématem na různých mediích, aby byl publikován. Tento způsob velice zvýhodňuje témata zmiňovaných na různých serverech, ale podle výsledků z testování se při tomto způsobu nejlépe vytvořil pravdivé přehledy dne.

9.2.2 Získání tématu

I když program dokáže zjistit, jak moc si jsou články podobné, neví, co je spojuje. A k tomu je využit LDA z knihovny Gensim[10].

9.2.3 Cvičení LDA

Nejprve se LDA musí naučit, jaká témata může vůbec používat. Vycvičí se tedy na korpusu půl miliónu článků a počet témat je nastaven na 25 tisíc. To znamená, že na jedno téma připadá průměrně přibližně 20 článků, což je v případě porovnávání zpravodajských článků optimální. U každého souboru témat poté program sečte procentní zastoupení všech témat a třemi největšími identifikuje dané téma.

[10] https://radimrehurek.com/gensim/models/ldamodel.html

```
for article in articles:
    texts_2 = [[text for text in doc.split()] for doc in article]
    dictionary.add_documents(article)
```

Obrázek 15 – Postupné tvoření korpusu

```
lda_model = LdaMulticore(corpus=articles,
                         id2word=dictionary,
                         num_topics=25000)
```

Obrázek 16 - Ukázka trénování LDA modelu

10 Stavba příspěvků

Nejprve si program zvolí, z jakého článku bude čerpat data. Porovná všechny články v souboru článků a ten, který bude ke všem ostatním průměrně nejblíže, je zvolen jako ten, který dané téma charakterizuje nejlépe.

Samotný příspěvek se skládá z výrazného obrázku, který má za úkol upoutat uživatele. Tvoří se pomocí jednotné šablony, takže je uživatelé lépe rozpoznají a asociují s tímto projektem. Na fotce se nachází fotka z vybraného článku a jeho titulek. To také zajistí jistou jedinečnost příspěvku a získá pozornost uživatele.

Pod příspěvkem lze nalézt text, jehož účel je informovat uživatele podrobněji. Jeho obsah by měl přiblížit téma, ale na druhou stranu nebýt moc dlouhý, aby čtenáře neodradil. K tomu program volí vždy první odstavec článku, který je informativní a většinou nepřekročí délku několika vět.

Poté následují hashtagy, jejíž cílem je rozšířit příspěvky mezi nové uživatele. Kromě základních a generických hashtagů, které mají za cíl dostat příspěvek k co největšímu počtu čtenářů, lze u příspěvků nalézt i 3 hashtagy, které jsou vygenerované počítačem. Jedná se o identifikaci daného tématu pomocí LDA, ale převedené do formy hashtagů, aby program cílil i na specifické uživatele sociálních sítí.

Nakonec je vložen ještě zdroj ve formě odkazu, aby měl uživatel možnost si o tématu přečíst více.

Obrázek 17- Příklad příspěvku na Instagramu

10.1 Generování příspěvků

Příspěvky jsou tvořeny pomocí Python knihovny PILLOW[11], která umožňuje jednoduché a intuitivní práci s obrázky.

Jako první si program stáhne bytovou interpretaci úvodních obrázků a to pomocí knihovny requests[12]. Ty se konvertují na normální Image. Následně program vytvoří obrázek, kam vloží úvodní fotku z vybraného článku do prvních tří čtvrtin obrázků a oranžový obdélník do zbytku.

Program zvolí správnou velikost písma, aby byl text dostatečně čitelný, ale nepřesahoval oranžový obdélník a vloží ho do něj. Protože u PILLOW neexistuje žádný způsob psaní textu na více řádků, program obsahuje vlastní algoritmus, kde zkouší postupně přidávat slova a počítá jejich délku s daným fontem a při přesáhnutí velikosti oranžového obdélníku začne psát na nový řádek.

Na levé straně se přidá ještě logo a název Měj přehled a to jednak z estetického hlediska, ale také, aby šli příspěvky publikované programem lépe rozpoznat mezi ostatními zpravodajskými servery. Obrázek je poté exportován do formátu png.

```
lastLength=0
writing=""
lastYPos=0
allLines=[]
heightSizes=[]
for item in splitted:
    writing+=item
    if(font.getsize(writing)[0]<=newimg.size[0]*9/10):
        lastLength=len(writing)
        writing+=" "
    else:
        allLines.append(writing[:lastLength])
        heightSizes.append(lastYPos)
        lastYPos+=font.getsize("writing")[1]
        lastLength=0
        writing=item+" "
```

Obrázek 18 - Algoritmus pro řádkování textu

Následuje tvorba textu, program vybere první odstavec daného článku a zkombinuje ho s hashtagy. Poté přidá pouze zdroj a příspěvek je připraven k publikování.

[11] https://pillow.readthedocs.io/en/stable/
[12] https://requests.readthedocs.io/en/master/

11 Sociální sítě

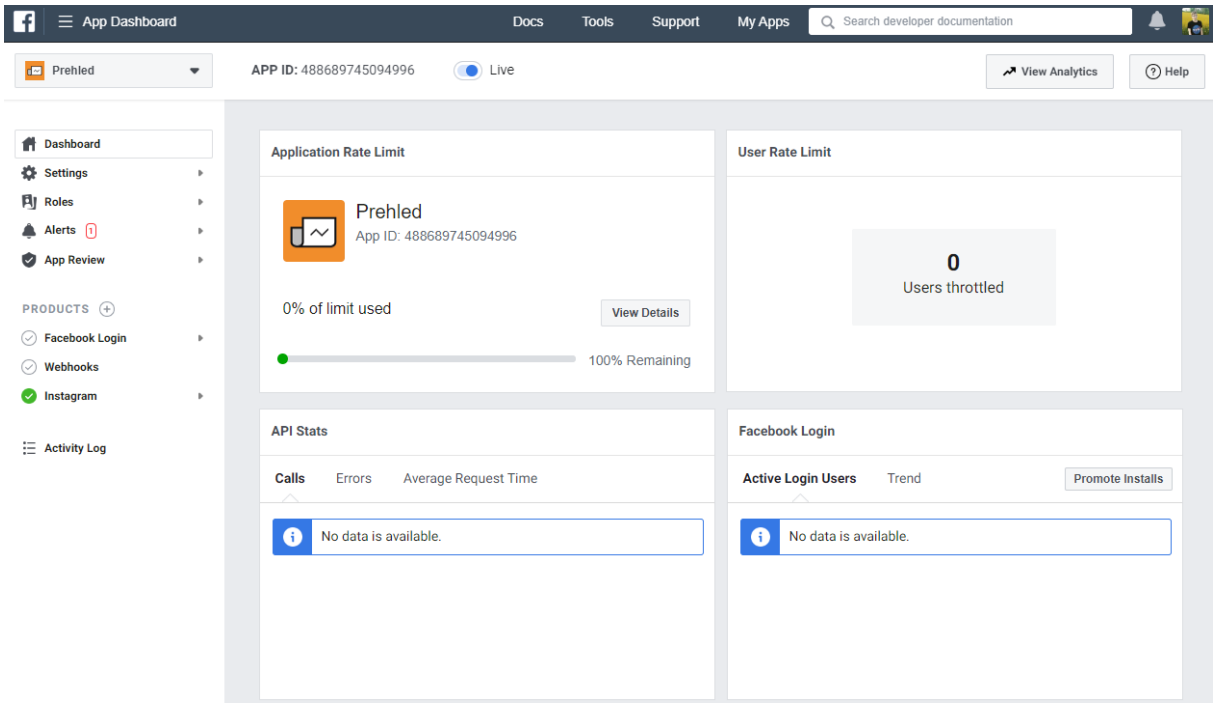
Na trhu se vyskytují desítky sociálních sítí, ale jenom některé se dají využít při distribuci našich příspěvků. Cílené platformy totiž musí mít co největší uživatelskou komunitu a musí být uzpůsobené na publikaci zpravodajských příspěvků. Z těchto důvodů autor použil Twitter, Facebook a Instagram.

11.1 Facebook

Se svými 2,2 miliardy uživateli se jedná o největší sociální síť na světě. (22) Také umožňuje vytvářet příspěvky s obrázky a neomezenými texty a pomocí sdílení se tyto posty můžou šířit mezi další uživatele. To z ní dělá jednu z nejideálnějších platforem na publikaci příspěvků. Ale kvůli fiasku s Cambridge Analytics a s nelegálním sběrem dat musel Facebook uzavřít své veřejné API, protože právě přes něj byla data sbíraná. (23)

11.1.1 Registrace aplikace

Aby mohl program používat Facebook Graph API, tedy jediné API pro Facebook, musí být registrovaný jako prověřená aplikace, aby se předešlo nekalým činnostem. Tento proces začíná ověřením osoby, kdy je nutné poskytnout svůj doklad s fotografií, aby se ověřila identita autora programu. Poté musí být aplikace vytvořená na Facebook Developers a doplněna o logo a základní popis. Tyto údaje nejsou povinné, ale doporučuje se jim věnovat aspoň trochu času, protože drasticky zvyšují šanci přijmutí aplikace Facebookem.



Obrázek 19 - Facebook aplikace

11.1.2 App Review

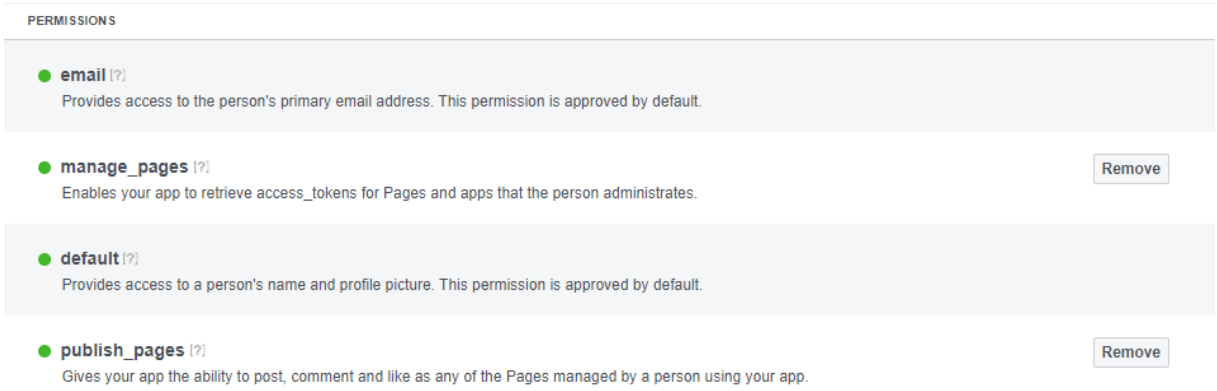
V App Review si autor aplikace vyžádá oprávnění, která mu poté umožňují pracovat s Facebook přes API. Existují různé typy oprávnění, ale program potřebuje jenom 2, manage_pages a publish_pages.

11.1.2 manage_pages

Toto oprávnění umožňuje aplikaci získat Page Access Token, který se potom používá při volání REST metod. Také pomáhá pracovat s příspěvky, komentáři a likes na stránce. (24)

11.1.3 publish_pages

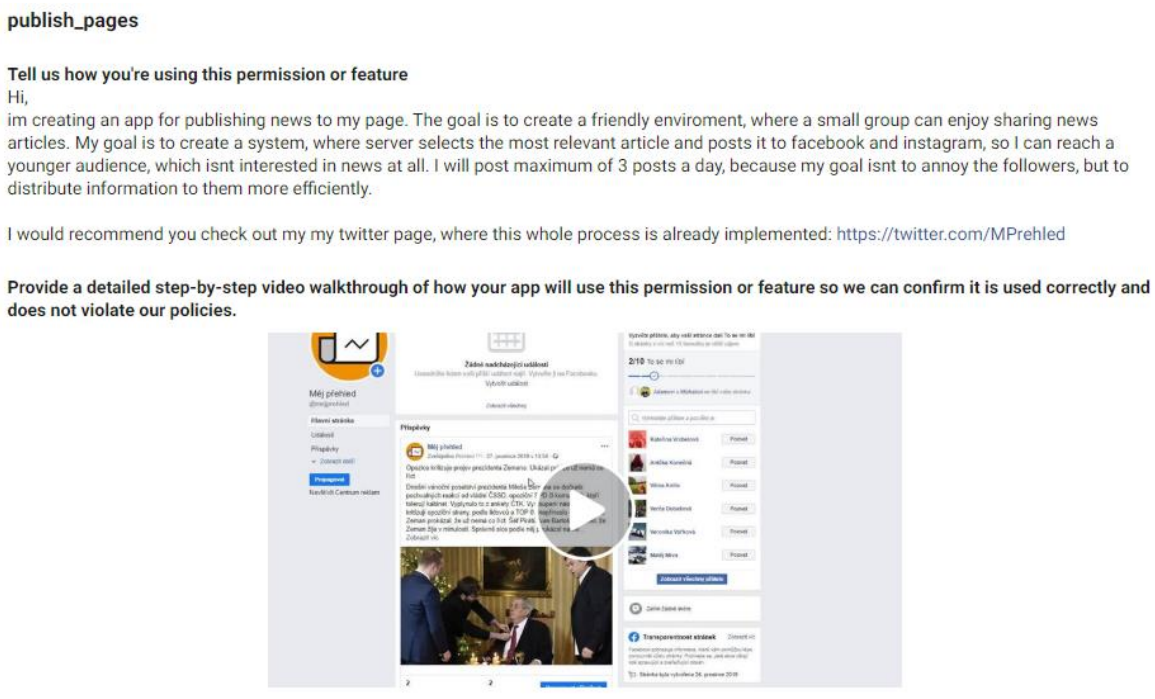
Dovoluje aplikaci publikovat příspěvky a komentáře, přidávat likes na stránky, sledovat, publikovat a odstraňovat videa. K fungování potřebuje mít také oprávnění manage_pages. (25)



Obrázek 20 – Ukázka získaných oprávnění

11.1.4 Proces ověření

Nejprve je nutné popsat, co bude aplikace s oprávněními dělat. Bylo tedy sepsáno pár odstavců o projektu a k tomu připojeno ještě video ukazující hlavní funkcionalitu programu, tedy publikování příspěvků.



Obrázek 21 - Ukázka popisu funkcionality programu

Po přibližně 4 dnech byla aplikace přijata a získala veškerá oprávnění, o které bylo žádáno. Díky tomu může program pomocí svého Access Token vytvářet příspěvky na jakýkoliv profil na Facebook, ale v případě tohoto systému se jedná pouze o jeden.

11.1.5 Publikace příspěvků

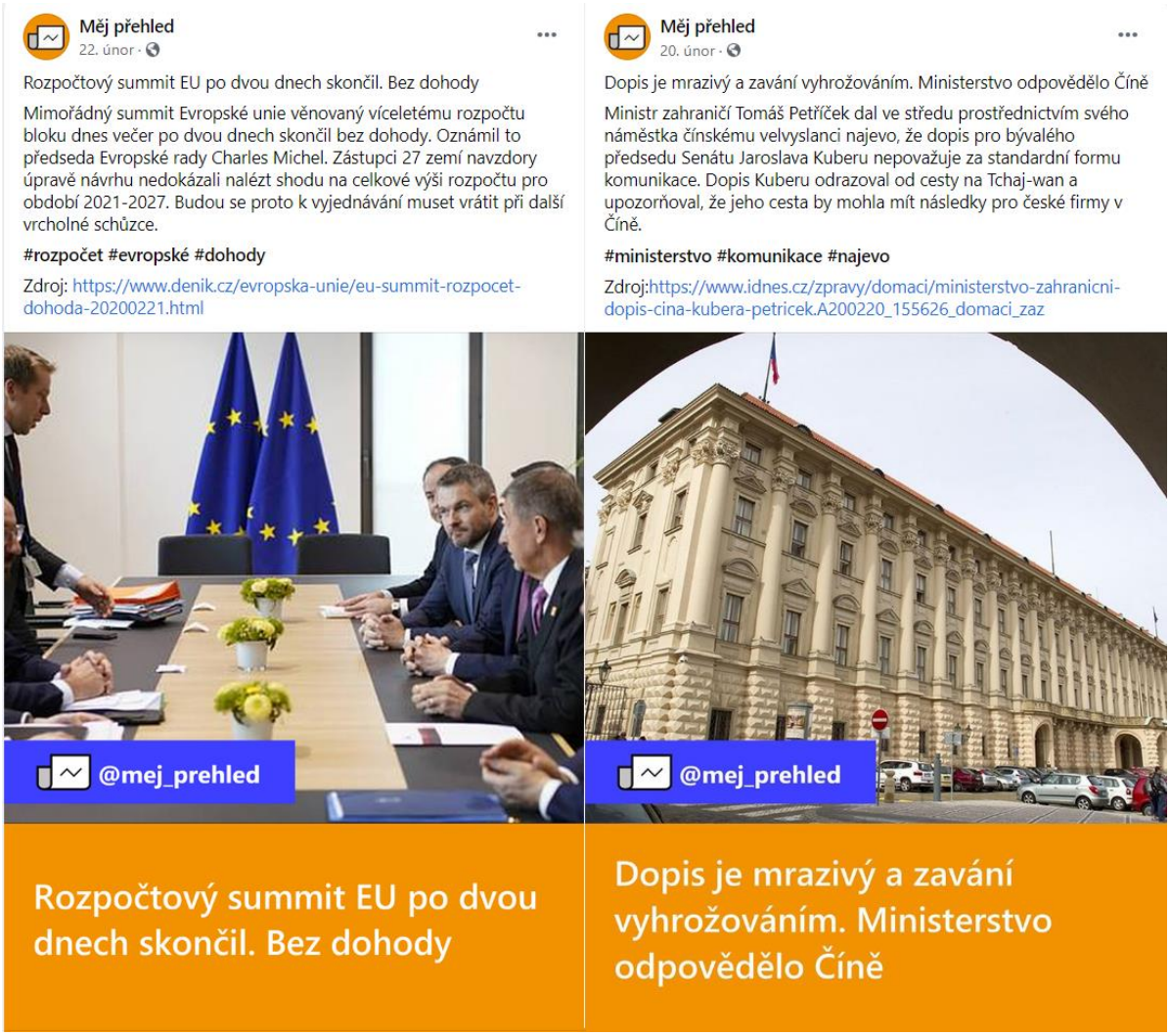
Příspěvek je složený z úvodní fotky, titulku, hashtagů, prvního odstavce a zdroje. I když o tom mnoho lidí neví, i hashtagy na Facebooku můžou pomoci příspěvku dostat se k novým uživatelům a rozšířit tak jejich dosah. (26)

Publikace je provedena POST akcí na https://graph.facebook.com/page_id/photos.

Page_Id v tomto případě udává ID aplikace, tedy Facebook stránky. Toto id je jedinečné pro každého uživatele a stránku. Kdybychom ho zaměnili za ID nějaké jiné stránky, mohl by program publikovat příspěvky i na ni.

K vyvolané POST akci se ještě přikládá samotný obrázek. Mezi parametry patří caption, což je popisek příspěvku. Další parametrem je access_token. Ten se používá k autorizaci uživatele, kdy zkontroluje, jestli má daný uživatel právo přidávat příspěvky na danou stránku.

Obrázek 22 - Profil na Facebooku



Obrázek 23 - Ukázka příspěvků na Facebooku

11.2 Twitter

Twitter je poskytovatel sociální sítě, který umožňuje uživatelům posílat a číst příspěvky zaslané jinými uživateli, známé jako tweety. Počet uživatelů k roku 2019 byl roven přibližně 330 miliónů uživatelů (27), z toho přibližně 10% je tvořeno počítačem řízenými účty, takzvanými boty. U twitteru je omezený počet charakterů (znaků) u příspěvků na 280, takže neumožňuje vytváření publikování delších popisků. (28) Tyto příspěvky jsou tak kratší a většinou obsahově chudší, jejich časová relevance je velice nízká. To znamená, že se příspěvek přestane zobrazovat uživatelům po kratší době, než je třeba typické na jiných platformách.

Kvůli omezené velikosti popisku se taky změní formát příspěvků. V tomto případě příspěvek nebude tvořit žádná fotka, ale pouze odkaz. Odkazy totiž na twitteru zvyšují zapojení uživatelů oproti pouhým obrázkům. (29) Vyplatí se tak více vytvářet tweety pouze s odkazem, protože budou mít větší zapojení a tím pádem i dosah. V popisku bude poté umístěn titulek článku a hashtagy vytvořené pomocí LDA. U twitteru jsou hashtagy velice důležité, spojují jednotlivá témata dohromady a můžeme pomocí nich dosáhnout většího počtu uživatelů.

11.2.1 Registrace aplikace

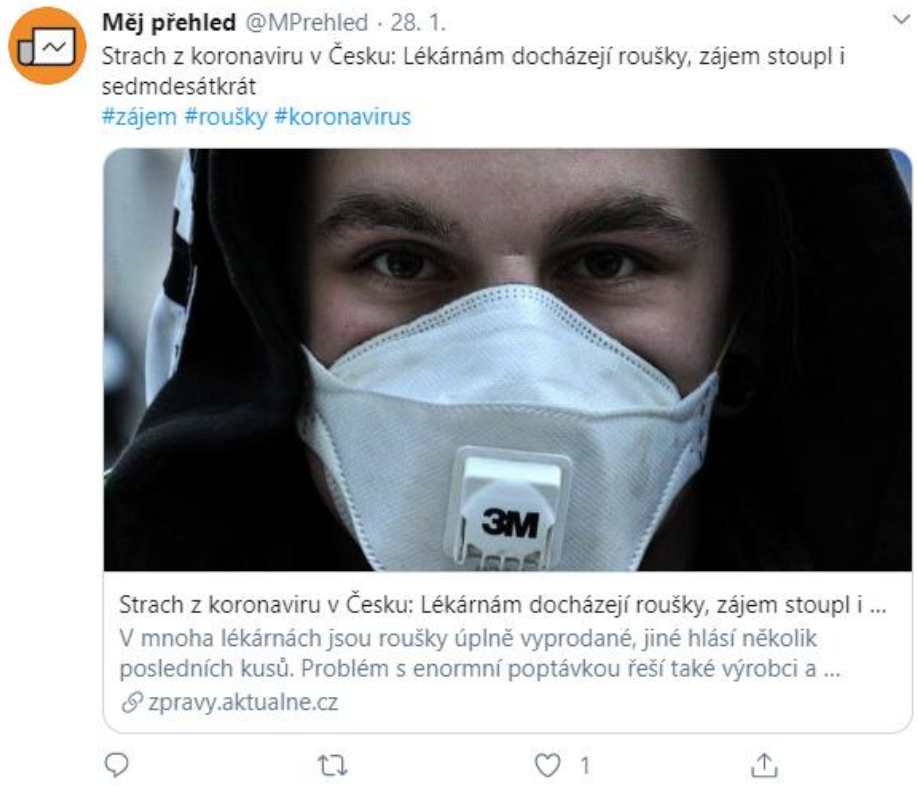
Neboť se na twitteru vyskytuje velké množství botů, na kterých je z části založený i celý systém twitteru, registrace aplikace je zdánlivě jednoduchá. Aby uživatel mohl využívat API, musí si nejprve zažádat o developer status, kdy vyplní dotazník ohledně způsobů, jak chce API využívat. Poté, co ho twitter potvrdí, muže začít registrovat svoje vlastní aplikace.

Při registraci se vyplňuje dotazník o využití API, ale v porovnání o hodně kratší než už u zmiňovaného Facebooku. V případě tohoto programu stačí pouze právo na zpracování a publikování příspěvků, takže byla popsána funkcionalita programu a odeslán daný dotazník. Po přibližně 5 minutách byla registrace potvrzena a program mohl začít využívat Twitter API.

11.2.2 Publikace příspěvků

Oproti Facebooku a Instagramu existují pro Twitter funkční Python knihovny. V programu je využita knihovna twitter[13], která umožňuje jednoduchou správu twitter účtů. Nejprve se inicializuje klient, přes kterého aplikace posléze komunikuje se servery twitteru.

Publikace příspěvku poté probíhá pomocí volání jediné metody, která bere jako parametr popisek příspěvku. Tento tweet je poté přidán k danému účtu a automaticky se mu vytvoří i karta s odkazem.



Obrázek 24 – Ukázka příspěvku na Twitteru

[13] https://python-twitter.readthedocs.io/en/latest/

11.3 Instagram

Instagram je sociální síť ve formě volně dostupné aplikace pro různé mobilní operační systémy. Svým uživatelům umožňuje sdílení fotografií, videí a chatování s přáteli. (34) V roce 2018 měl více než miliardu aktivní uživatelů měsíčně. V USA má účet na Instagramu 75% populace ve věkové kategorii od 18 do 24 let, tedy cílová skupina tohoto projektu. (30)

Díky tomu se Instagram velice hodí na publikování příspěvků. A také se na této platformě nenachází tak velká konkurence, není totiž možné přidávat odkazy k příspěvkům, takže nejdou dobře monetizovat. Z těchto důvodů ostatní zpravodajské portály nejsou na Instagramu až tak aktivní. (31)

Kvůli již zmiňované kauze s Cambridge Analytics s nelegálním sběrem dat o uživatelích bylo uzavřeno i Instagram API. A protože Facebook se snaží vytvořit na Instagramu prostředí, na kterém se dají vytvářet příspěvky pouze přes mobilní zařízení, úplně zablokoval API pro publikování příspěvků. K němu mají od té doby přístup pouze ověřené společnosti a jejich výběr je už uzavřen. I když existuje šance, že po nějaké době Facebook znovu API otevře, není jisté kdy. A proto bude program muset využívat aplikaci třetí strany na publikování příspěvků na Instagram. (32)

11.3.1 OneUp

OneUp je internetová aplikace umožňující naplánování a automatickou recyklaci příspěvků na sociálních sítích. Jeho hlavní výhodou je velice levné předplatné v porovnání s podobnými službami. Využívají ho hlavně firmy pro správu účtů, neboť pro jejich vytváření a plánování zobrazuje také statistiky. Náš program využije ale jinou jeho funkci, a to publikování příspěvků přes RSS Feed. (33)

Poté, co autor prošel většinu aplikací, které dokáží publikovat příspěvky na Instagram, zjistil, že neexistuje žádná platforma, která by je umožňovala publikovat přes volání jejich API. Jedinou zbývající možností bylo tedy vytváření příspěvků přes RSS zdroje.

Program bude mít ale různé formáty příspěvků na rozdílných sociálních sítích, nemůže proto pouze využít RSS kanál generovaný z jedné sociální sítě a poté ho publikovat na Instagram. Z tohoto důvodu si musí RSS Feed vytvářet sám.

11.3.2 Formát RSS Feedu

RSS Feed má přesně daný formát, který musí splňovat, aby ho dokázaly aplikace zpracovat. První element je rss, který říká, o jakou verzi RSS feedu se jedná, a také se v něm odkazuje na různé XML Namespaces, které se tam využívají.

Následují jednotlivé kanály, v programu se využívá pouze jeden. V něm lze nalézt informace o něm, jeho popis, odkaz, název a jazyk. Poté následují items, což jsou jednotlivé příspěvky.

V každém item je nadpis, z něho se tvoří popisek příspěvků, takže se v něm nachází veškerý text. Poté následuje datum vydání příspěvku. OneUp si pamatuje, kdy byl vydaný poslední příspěvek, aby při příštím načtení RSS Feed věděl, které příspěvky přibyly a musí je ještě publikovat.

```
40.       <item>
41.         <title>Prezident Miloš Zeman podporuje opatření zavedená v boji proti šíření nového typu koronaviru. Pokud
se někomu zdají drastická, má si uvědomit, že při jejich nezavedení by mohly nastat smrtelné případy onemocnění. Zeman
to řekl televizi Nova. Prezident si nemyslí, že by se situace vyhrotila jako v Itálii, která zavedla karanténu v celé
zemi. Varoval před nákupní horečkou.
42.
43.     #školy #opatření #koronaviru
44.
45.     Zdroj: https://zpravy.aktualne.cz/domaci/zeman-je-pripraveny-na-karantenu-po-navratu-ze-ciny-kroky-
vl/r~69b18d12630511eab0f60cc47ab5f122/</title>
46.         <pubDate>Wed, 11 Mar 2020 14:52:19 GMT</pubDate>
47.         <media:content medium="image" url="https://i.ibb.co/kmvcZWj/eb94b06f4604.png"/>
48.         <link>https://www.facebook.com/mejprehled/photos/a.112346846939711/119460812894981/</link>
49.         <enclosure url="https://i.ibb.co/kmvcZWj/eb94b06f4604.png" length="0" type="image/jpeg"/>
50.         <description>![CDATA[&lt;div&gt; &lt;img src="https://i.ibb.co/kmvcZWj/eb94b06f4604.png"style="width:
100%;"&gt; &lt;div&gt;Prezident Miloš Zeman podporuje opatření zavedená v boji proti šíření nového typu koronaviru.
Pokud se někomu zdají drastická, má si uvědomit, že při jejich nezavedení by mohly nastat smrtelné případy onemocnění.
Zeman to řekl televizi Nova. Prezident si nemyslí, že by se situace vyhrotila jako v Itálii, která zavedla karanténu v
celé zemi. Varoval před nákupní horečkou.
51.
52.     #školy #opatření #koronaviru
53.
54.     Zdroj: https://zpravy.aktualne.cz/domaci/zeman-je-pripraveny-na-karantenu-po-navratu-ze-ciny-kroky-
vl/r~69b18d12630511eab0f60cc47ab5f122/&lt;/div&gt; &lt;/div&gt;]]&gt;</description>
55.       </item>
```

Obrázek 25 - Ukázka item uvnitř RSS zdroje

11.3.3 Firebase server

Firebase server bude mít za úkol přistupovat do databáze, vybrat příspěvky a tvořit z nich RSS zdroj, který bude poté vložen do OneUp. Firebase byl zvolen převážně kvůli doméně a hostingu zdarma. Ten projektu přiřadí jeho jedinečnou URL a umožňuje na ní spustit server aplikace. Také na něm lze ukládat data v databázi, ve které program může ukládat příspěvky a poté z nich generovat RSS Feed. Program tedy využije tuto službu a přitom na ní spustí Node.js server a kód sepsaný v JavaScriptu.

Pokud program využije neplacenou verzi Firebase, musí počítat s jistými omezeními. První je omezený počet invokací metod na Rest API. Ale protože k REST API přistupuje pouze OneApp a to ne tak často, tohle není problém. Ale také je omezená velikost databáze, což znamená, že v ní nemůžeme uložit všechny články. V databázi bude proto ukládat pouze příspěvky, které se mají publikovat a informace o nich.

11.3.4 Rest API

API se bude skládat ze dvou metod, RSSFeed a AddPost. RSSFeed bude vracet RSS kanál a AddPost přidá nový příspěvek do RSS zdroje.

11.3.5 AddPost

Tato akce přidá příspěvek do databáze. Její parametry jsou imageUrl a title. Všechny musí být vloženy do RRS zdroje ve formátu URL odkazu, neboť OneUp neakceptuje textový formát obrázků. Title udává popisek příspěvku. Server poté k příspěvku přidá ještě přesný čas vyvolání této akce a tyto hodnoty uloží do databáze.

11.3.6 RSSFeed

K vytvoření RSS kanálu využije Node.js knihovnu xmlbuilder[14], která umožňuje jednoduše tvořit XML a poté ho exportovat do textového formátu. Nejprve program sestaví informace o kanálu a potom interuje přes poslední deset přidaných příspěvků, které přidá do XML. A nakonec toto XML v textovém formátu vrací.

```
exports.rssfeed = functions.https.onRequest(async (req, res) => {
  const db = admin.firestore()
  var root = builder.create('rss');
  root.att("version","2.0")
  root.att("xmlns:content","http://purl.org/rss/1.0/modules/content/")
  root.att("xmlns:dc","http://purl.org/dc/elements/1.1/")
  root.att("xmlns:atom","http://www.w3.org/2005/Atom")
  root.att("xmlns:media","http://search.yahoo.com/mrss/")
  const channel=root.ele("channel");
  channel.ele("description",'',"![CDATA[Měj přehled. Zpravodajský portál řízený umělou inteligencí Aktuální zpravy a denní přehledy]]")
  channel.ele("link","","https://www.facebook.com/mejprehled/")
  channel.ele("title",'',"![CDATA[ Měj přehled]]")
  channel.ele("language",'',"![CDATA[ en ]]");
  let citiesRef = db.collection("posts");
  let allCities = citiesRef.orderBy("PublishDate", "desc").get().then(snapshot => {
    snapshot.forEach(doc => {
      let item=channel.ele('item')
      item.ele("title",'',doc.data().Title)
      item.ele("pubDate",'',doc.data().PublishDate.toDate().toUTCString());
      item.ele("media:content",{'medium': 'image','url':doc.data().ImageUrl})
      item.ele("link",'',"https://www.facebook.com/mejprehled/photos/a.112346846939711/119460812894981/")
      item.ele("enclosure",{'url':doc.data().ImageUrl,'length':'0','type':'image/jpeg'})
      item.ele("description",'',"![CDATA[<div> <img src=\""+doc.data().ImageUrl+"\"style=\"width: 100%;\"> <div>"+doc.data().Title+"</div> </div>]]>")
    });

    const xml = root.end({ pretty: true});

    res.send(xml);
  })
  .catch(err => {
    console.log('Error getting documents', err);
  });
});
```

Obrázek 26 - Generování RSS Feedu

11.3.7 Publikace příspěvků

Připojení RSS zdroje k OneApp je jednoduché, stačí provést pár kliků. Do dialogového okna se vloží odkaz na RSS Feed, vybere daný profil a všechno už funguje bez jakékoliv lidské pomoci. OneApp každých 6 hodin načte RSSFeed znovu a zjistí, jestli byly vydány nějaké nové příspěvky. Pokud ano, tak je s odstupem půl hodiny všechny publikuje na Instagram.

Protože se do RSSFeedu musí obrázky ukládat v podobě URL odkazů, musí program také svoje vygenerované obrázky uložit online. A k tomu využije ImgBB[15], což je stránka, která zdarma umožňuje hosting obrázků. Jeho hlavní výhodou je otevřené API, která programu dovoluje ukládat obrázky na jejich servery a poté okamžitě získat jejich URL. Přidání obrázku se provede zavoláním POST akce upload, přičemž mezi jejími parametry je uživatelský klíč a obrázek v base64. V odpovědi server vrací URL odkaz.

Program poté zavolá metodu na Firebase serveru, kde přidá daný příspěvek s jeho popiskem a odkazem na obrázek.

[14] https://www.npmjs.com/package/xmlbuilder
[15] https://cs.imgbb.com/

Obrázek 27 - Ukázka příspěvků na Instagramu

12 Závěr

V této práci je popsán systém, který dokáže vybrat důležité zprávy, distribuovat a informovat o nich mladou generaci přes sociální sítě. Díky vytvořenému virtuálnímu reportérovi funguje tento systém nezávisle na člověku a může běžet 24 hodin denně. Zprávy zpracovává objektivně a nezaujatě z nich vybírá ty nejdůležitější. Zvládá zpracovat tisíce článků denně a z několika rozdílných zdrojů.

Dalším důležitým aspektem celého projektu jsou samotné jazykové modely, jak fastText, tak LDA. K jejich vytvoření bylo potřeba stáhnout půl miliónů zpravodajských článků, přibližně 300 miliónů slov a samotné trénování modelů trvalo skoro měsíc. Složitým procesem bylo i samotné publikování výsledků na sociálních sítích.

Výsledky této práce jdou vidět na těchto účtech:

https://www.facebook.com/mejprehled
https://www.instagram.com/mej_prehled/?hl=cs
https://twitter.com/MPrehled

Tento systém byl nabídnut firmě Seznam, která je vedoucí českou firmou na poli technologií a nově se starají i o vlastní zpravodajství. Do termínu odevzdání práce se nestačily vyjednat přesnější podmínky spolupráce.

13 Použitá literatura

1. -. *Mediagram*[online]. 2020 [Cit. 10. 2. 2020]. Dostupné z URL: https://mediagram.cz/dejepis/vyvoj-medii-od-knihtisku-po-internet
2. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://cs.wikipedia.org/wiki/C_Sharp
3. -. *Python*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.python.org/doc/essays/blurb/
4. -. *Python Institute*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://pythoninstitute.org/what-is-python/
5. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://cs.wikipedia.org/wiki/JavaScript
6. -. *tutorialspoint*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.tutorialspoint.com/javascript/javascript_overview.htm
7. Priyesh Patel. *freeCodeCamp*[online]. 18.4.2018 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/
8. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://en.wikipedia.org/wiki/Firebase
9. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2019]. Dostupné z URL: https://cs.wikipedia.org/wiki/RSS
10. Margaret Rouse: *SearchSQLServer*[online]. 7.9.2016 [Cit. 14. 2. 2020]. Dostupné z URL: https://searchsqlserver.techtarget.com/definition/SQL
11. Martin Heller: *InfoWorld*[online]. 1.11.2019 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.infoworld.com/article/3219795/what-is-sql-the-first-language-of-data-analysis.html
12. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://en.wikipedia.org/wiki/FastText
13. Jayesh Bapu Ahire. *Medium*[online]. 12.3.2018 [Cit. 14. 2. 2020]. Dostupné z URL: https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf
14. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://en.wikipedia.org/wiki/Representational_state_transfer
15. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://en.wikipedia.org/wiki/Entity_Framework
16. MediaGuru: *MediaGuru*[online]. 23.9.2018 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.mediaguru.cz/clanky/2018/09/zpravodajske-weby-v-prvni-desitce-i-echo-v-blizkosti-take-forum-24/
17. -. *Html Agility Pack*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://html-agility-pack.net/
18. Václav Cvrček. *Český národní korpus*[online]. 20.2.2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://wiki.korpus.cz/doku.php/cnk:uvod
19. Kavita Ganesan. *Kavita Genesan*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://kavita-ganesan.com/what-are-stop-words/
20. Václav Cvrček. *Český národní korpus*[online]. 24.11.2014 [Cit. 14. 2. 2020]. Dostupné z URL: https://wiki.korpus.cz/doku.php/pojmy:token
21. Václav Cvrček. *Český národní korpus*[online]. 4.3.2019 [Cit. 14. 2. 2020]. Dostupné z URL: https://wiki.korpus.cz/doku.php/pojmy:lemma
22. Esteban Ortiz-Ospina. *Our World in Data*[online]. 18.9.2019 [Cit. 14. 2. 2020]. Dostupné z URL: https://ourworldindata.org/rise-of-social-media
23. Josh Constine. *TechCrunch*[online]. 4.4.2018 [Cit. 14. 2. 2020]. Dostupné z URL: https://techcrunch.com/2018/04/04/facebook-instagram-api-shut-down

24. -. *Facebook for developers*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://developers.facebook.com/docs/facebook-login/permissions/#reference-manage_pages
25. -. *Facebook for developers*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://developers.facebook.com/docs/facebook-login/permissions/#reference-publish_pages
26. Brent Barnhart. *sproutsocial*[online]. 22.1.2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://sproutsocial.com/insights/hashtags-on-facebook/
27. Ying Lin. *Oberlo*[online]. 30.11.2019 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.oberlo.com/blog/twitter-statistics
28. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://cs.wikipedia.org/wiki/Twitter
29. Scott Ayres. *Social Media Lab*[online]. 11.8.2017 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.agorapulse.com/social-media-lab/link-tweet-with-an-image-vs-tweet-with-twitter-cards-does-it-really-matter/
30. -. *Wikipedia*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://cs.wikipedia.org/wiki/Instagram
31. Tony Tran. *Hootsuite*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://blog.hootsuite.com/instagram-demographics/
32. Francesca Molteni. *Medium*[online]. 5.6.2019 [Cit. 14. 2. 2020]. Dostupné z URL: https://medium.com/@fmolteni/the-news-industry-is-in-decline-and-why-instagram-is-here-to-help-ffd74fcd825d
33. -. *OneUp*[online]. 2020 [Cit. 14. 2. 2020]. Dostupné z URL: https://www.oneupapp.io/

14 Seznam obrázků

Obrázek 1 - Diagram systému 11
Obrázek 2 - Ukázka EFUnitOfLoaders 12
Obrázek 3 - Stahování odkazů z Aktuálně 13
Obrázek 4 - Určování přibližného času vydání článku z textu u Novinky 14
Obrázek 5 - Stahování článků z Deníku 15
Obrázek 6 - Ukázka RepositoryBase 16
Obrázek 7 - Ukázka Repository pro články 16
Obrázek 8 - Ukázka EFUnitOfWork 17
Obrázek 9 - Projekt pro exportování článků 19
Obrázek 10 - Ukázka tokenizace a lemmatizace 20
Obrázek 11 - Cvičení modelu 21
Obrázek 12 - Ukázka kosinové vzdálenosti vybraných slov podle vlastního FastText modelu 22
Obrázek 13 - Ukázka grafu článků 22
Obrázek 14 - Ukázka získávání podgrafů pomocí rekurze 23
Obrázek 15 – Postupné tvoření korpusu 24
Obrázek 16 - Ukázka trénování LDA modelu 24
Obrázek 17- Příklad příspěvku na Instagramu 25
Obrázek 18 - Algoritmus pro řádkování textu 26
Obrázek 19 - Facebook aplikace 27
Obrázek 20 – Ukázka získaných oprávnění 28
Obrázek 21 - Ukázka popisu funkcionality programu 28
Obrázek 22 - Profil na Facebooku 29
Obrázek 23 - Ukázka příspěvků na Facebooku 30
Obrázek 24 – Ukázka příspěvku na Twitteru 31
Obrázek 25 - Ukázka item uvnitř RSS zdroje 33
Obrázek 26 - Generování RSS Feedu 34
Obrázek 27 - Ukázka příspěvků na Instagramu 35

Příloha 1: Datový model databáze