VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

RSS ČTEČKA PRO IOS

RSS READER FOR IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **IAROSLAV BOGDANCHIKOV**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN HRUBÝ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student:	**Bogdanchikov Iaroslav**
Program:	Informační technologie
Název:	**RSS čtečka pro iOS**
	RSS Reader for iOS
Kategorie:	Uživatelská rozhraní

Zadání:

1. Prostudujte programování aplikací pro iOS. Prostudujte systém RSS. Prostudujte současný stav dostupných aplikací pro čtení dokumentů přes RSS.
2. Navrhněte aplikaci pro čtení dokumentů přes RSS.
3. Aplikaci implementujte pro iOS.
4. Testujte aplikaci na různých typech zdrojů. Aplikaci srovnejte s jinými nástroji tohoto typu.

Literatura:

- Keur, Ch., Hillegass, A.: iOS Programming: The Big Nerd Ranch Guide, Big Nerd Ranch Guides; 6 edition (January 6, 2017), ISBN-13: 978-0134682334

Pro udělení zápočtu za první semestr je požadováno:

- První dva body.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce:	**Hrubý Martin, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1. listopadu 2018
Datum odevzdání:	15. května 2019
Datum schválení:	1. listopadu 2018

Abstrakt

Cílem této práce je vytvořit mobilní aplikaci pro operační systém iOS pro snadné sledování, prohlížení a uložení/archivaci zpráv ze zpravodajských serveru a blogů, čerpaných ze zdrojů ve formátu RSS. Aplikace je implementována v programovacím jazyce Swift. Pro perzistenci dat byl použit aplikační rámec Core Data firmy Apple. Práce se skládá z teoretické a praktické částí. V teoretické částí je popsán formát RSS a zásady programování aplikací pro operační systém iOS. Praktická část obsahuje popis návrhu, implementace a testování aplikace. Výsledná aplikace umožňuje uživateli jednoduše pracovat s RSS kanály a příspěvky a díky dodržování Apple Human Interface Guidelines vypadá přirozeným prvkém systému iOS.

Abstract

The aim of this work is to create a mobile application for iOS operating system for easy following, reading and storing/archiving of stories from news sites and blogs accessed through web feeds in RSS format. The application is implemented in Swift programming language. Apple's Core Data application framework was used for data persistence. The work consists of theoretical and practical parts. The theoretical part describes the RSS format and principles of programming an application for iOS operating system. The practical part contains description of design, implementation and testing of the application. The resulting application provides user with a simple instrument to work with RSS channels and stories, and by following the Apple Human Interface Guidelines the application fits naturally in iOS ecosystem.

Klíčová slova

Mobílní aplikace, Apple, iOS, Swift, Core Data, RSS, webový zdroj.

Keywords

Mobile application, Apple, iOS, Swift, Core Data, RSS, web feed.

Citace

BOGDANCHIKOV, Iaroslav. *RSS čtečka pro iOS*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Hrubý, Ph.D.

RSS čtečka pro iOS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Iaroslav Bogdanchikov
15. května 2019

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce panu Ing. Martinu Hrubému, Ph. D. za pomoc a rady, které mi byly poskytnuté při návrhu a implementaci aplikace.

Obsah

1 Úvod **3**

2 Formáty RSS a Atom **4**
2.1 Vznik a vývoj formátů pro syndikaci web obsahu 4
2.2 Formát RSS 5
2.3 Formát Atom 6
2.4 Princip práce webových kanálů 7
2.5 Výhody a nevýhody RSS 8
2.6 Analýza existujících řešení 9
2.6.1 Newsify 9
2.6.2 Reeder 9
2.6.3 Unread 9
2.6.4 Shrnutí 10

3 Programování aplikací pro operační systém iOS **11**
3.1 Vývojové prostředí Xcode 11
3.2 Programovací jazyky 12
3.3 Zásady programování aplikací pro iOS 12
3.3.1 Návrhové vzory používané v iOS aplikacích 12
3.3.2 Auto Layout 14
3.3.3 Core Data 14

4 Návrh aplikace **16**
4.1 Požadovaná funkcionalita 16
4.2 Uživatelské rozhrání 16
4.3 Datový model 18

5 Implementace **20**
5.1 Struktura aplikace 20
5.2 Implementace jádra 20
5.2.1 Třída XMLFeedParser 21
5.2.2 Třída GetFeedOperation 21
5.3 Implementace vrstvy Model 22
5.3.1 Datové typy pro práci s kanály a články 22
5.3.2 Perzistence dat 22
5.3.3 Třída PersistentContainer 22
5.4 Implementace vrstvy View 23
5.5 Implementace vrstvy Controller 23

5.5.1 Přidávání nového kanálu 23
5.5.2 Aktualizace odbíraných kanálů 24

6 Testování **26**
6.1 Testování na různých typech webových zdrojů 26
6.2 Porovnání aplikace RSS Reader s existujícími aplikacemi 27
6.2.1 Porovnání s Newsify 27
6.2.2 Porovnání s Reeder 27
6.2.3 Porovnání s Unread 27

7 Závěr **30**

Literatura **31**

A Webové zdroje použité pro testování **35**

Kapitola 1

Úvod

Technologie se neustále rozvíjí a značně zjednodušují mnoho věcí. Jedna z věcí, která byla hodně zjednodušena je sdílení informací se světem. Dnes doslova každý člověk, který má přístup k internetu může v řádech minut založit webové stránky libovolného typu od jednoduchého blogu do komplikovaného zpravodajského webu. Kvůli tomu značně roste počet zdrojů informace. Vzniká tak problém spotřeby informací z těchto zdrojů. Sledovat každý zdroj ručně a hledat, jestli se změnil jeho obsah a jestli se objevilo něco nového je velmi pracné a zabírá obrovské množství cenného času. Naštěstí většina webových stránek podporuje technologie, které by měly pomoct s řešením tohoto problému. Těmito technologiemi jsou datové formáty RSS a Atom. S těmito formáty pracuji nástroje, které se nazývají RSS čtečky. RSS čtečka je program, který umí stahovat aktualizovaný obsah sledovaných webových stránek za použitím výše zmíněných formátů a ukazovat ho uživateli v čitelné formě.

Jako dlouhodobý aktivní uživatel RSS čteček a mobilních zařízení Apple postupem času jsem měl možnost vyzkoušet různé aplikace tohoto typu, které existuji pro mobilní operační systém iOS. Bohužel ani s jedním z nich jsem nebyl úplně spokojen. Tato skutečnost v kombinaci s touhou se naučit programovat aplikace pro operační systém iOS a vytvořit pro něj smysluplný a užitečný program, který bych mohl používat v každodenním životě mě motivovala k provedení této bakalářské práce.

Cílem této práce je nastudovat, jak funguje šíření web obsahu pomocí formátů RSS a Atom. Nastudovat programování aplikací pro operační systém iOS. Dále na základě získaných znalostí navrhnout a implementovat RSS čtečku s jednoduchým uživatelským rozhraním, která by umožnila uživateli snadný odběr a prohlížení aktualizací z webových stránek. Implementovaná aplikace se bude nazývat RSS Reader.

Práce je členěna do sedmi kapitol. Ve druhé kapitole je uvedena krátká historie vzniku formátů RSS a Atom, jejich popis a provedena analýza existujících řešení. Následuje kapitola, ve které se popisuji zásady programování pro operační systém iOS a dostupné nástroje. Čtvrtá kapitola se věnuje návrhu aplikace, konkrétně uživatelskému rozhrání a vnitřní architektuře. Další kapitola obsahuje informaci o implementaci a použitých API. Kapitola číslo šest se zabývá testováním výsledné aplikace. V závěru jsou zhodnoceny výsledky práce a uvedeny možností budoucího vývoje.

Kapitola 2

Formáty RSS a Atom

Táto kapitola se zabývá popisem formátů RSS a Atom, principem jejich práce, výhod, případných nedostatků a analýzou existujících RSS čteček pro operační systém iOS.

RSS a Atom jsou typy webových zdrojů neboli kanálů (v angličtině web feed nebo news feed). Webový kanál je dokument, který obsahuje odkazy na nový a aktualizovaný obsah konkrétní webové stránky. Tento dokument je zveřejňován autory webových stránek ve formátu, který může být zpracován softwarem určeným k zobrazení takových dokumentů.

Proces zveřejnění aktualizovaného obsahu přes webový kanál se nazývá *syndikace*. Proces pravidelného získávání aktualizací z webového kanálu se nazývá *odběr*.

Nástroje, které umí pracovat s webovými kanály se nazývají RSS čtečky. RSS čtečky také splňuji roli agregátorů, to jest umožňují zobrazení aktualizací z mnoha kanálů najednou.

2.1 Vznik a vývoj formátů pro syndikaci web obsahu

Jedním z prvních předchůdců syndikace web byl formát MCF – Meta Content Framework vyvinutý ve společnosti Apple Ramanathanem V. Guhou [31] a dalšími. Dále Ramanathanem Guhou s pomocí Tima Braye, jedním z autorů originální specifikace XML, MCF byl přizpůsoben formátu XML a tak se stál základem pro RDF [31] – Resource Description Framework (systém popisu zdrojů).

V březnu roku 1999 byla vydána první verze formátu RSS, která měla číslo verze 0.9 a byla založena na formátu RDF, a proto zkratka RSS znamenala RDF Site Summary [29]. Na jejím vývoji spolupracovali Ramanathan V. Guha a Dan Libby [31].

Již v červenci roku 1999 byla vytvořena verze 0.91 [30], ve které z původní specifikace byly odstraněny prvky RDF a formát byl přejmenován na Rich Site Summary.

V následujících letech na vývoji formátu se podílely různé vývojové skupiny, které měly odlišnou představu o tom, jak se má formát dále rozvíjet a kvůli tomu vzniklo dvě nekompatibilní větvě.

V prosinci roku 2000 vývojář Dave Winer se svojí skupinou UserLand Software vytvořil verzi RSS 0.92 [35], která umožnila přenos audio souboru a tím pomohla rozvinutí podcastingu[1].

Nejrozšířenější dnes verze RSS, 2.0, byla vydána UserLand Software v září roku 2002 [36]. Ve verzi 2.0 se naposledy změnil význam zkratky na Really Simple Syndication. V roce 2003 autorská práva na formát RSS 2.0 byla předána výzkumnému centru Harvardové univerzity Software Berkman Klein Center for Internet & Society [37].

[1] History of podcasting https://en.wikipedia.org/wiki/History_of_podcasting#The_RSS_connection

Významnou událostí v popularizaci online syndikace obsahu a zejména formátu RSS se stalo nabízení k odběru RSS kanálu newyorským deníkem The New York Times v roce 2002 [38].

Po vydání verze 2.0 formát RSS stále obsahoval nevyřešené problémy, které ale nešlo vyřešit kvůli tomu, že jakékoliv změny byly zakázány pro zachování stability formátu. Tuto situaci rozhodl vyřešit Sam Ruby, který rozhodl vytvořit nový formát. Tento formát později obdržel jméno Atom. Tato myšlenka byla podpořena mnohými významnými vývojáři včetně tvůrce RSS 2.0 Dava Winera[2]. [32]

V roce 2005 formát byl standardizován[3] organizací IETF – Internet Engineering Task Force (Komise pro technickou stránku internetu).

2.2 Formát RSS

Jak již bylo zmíněno výš dnes zkratka RSS znamená **Really Simple Syndication**. Poslední a nejpoužívanější verze RSS je 2.0. Proto tato podkapitola bude věnována popisu konkrétně této verze.

RSS je dialektem značkovacího jazyka XML – eXtensible Markup Language (rozšiřitelný značkovací jazyk) a musí odpovídat specifikaci XML verze 1.0.

Kořenovým elementem RSS dokumentu je element `<rss>`. Kvůli tomu, že existuje mnoho verzí formátu, které se stále mohou používat, v atributu `version` elementu `rss` musí být uvedena verze.

Jediným synovským elementem elementu `<rss>` je element `<channel>`. Tento element musí povinně obsahovat alespoň tři následující elementy, které nesou informace o kanálu (metadata):

- `<title>` – název kanálu;
- `<link>` – odkaz na webové stránky kanálu;
- `<description>` – popis kanálu.

Kanál také může zahrnovat i nepovinné elementy, jako například `<image>` (obrázek kanálu, například logo), `<category>` (jedna nebo více kategorií do kterých kanál tematicky patří), `<language>` (jazyk obsahu kanálu) a další.

Obsahem kanálu jsou články. Počet článku v kanálu je libovolný. Článek se označuje elementem `<item>`. Žádný z podelementů elementu `<item>` je povinný, ale každý článek musí obsahovat buď element `<title>`, nebo `<description>`.

Nejčastěji článek obsahuje následující elementy:

- `<title>` – nadpis článku,
- `<description>` – obsah článku,
- `<link>` – odkaz, který vede na konkrétní stránku odpovídající článku,
- `<pubDate>` – datum a čas publikace článku ve formátu RFC 822[4].

[2] Tentative endorsement of Echo http://backend.userland.com/2003/06/26
[3] The Atom Syndication Format https://tools.ietf.org/html/rfc4287
[4] RFC 822 https://www.rfc-editor.org/rfc/rfc822.txt

Článek také může obsahovat elementy `<autor>` (e-mail autora článku), `<category>` (kategorie), `<comments>` (odkaz na stránku s komentáři k článku) a další.

Jeden článek může obsahovat jak celý text článku z webových stránek, tak i stručný popis s podmínkou, že v elementu `<link>` bude odkaz na stránku s celým textem. Element `<description>` může obsahovat i text s HTML entitami[5].

Příklad RSS dokumentu:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title>Channel's Title</title>
        <description>Short description of this RSS channel.</description>
        <link>https://example.com</link>
        <item>
            <title>Item's Title</title>
            <link>https://example.com/post/1</link>
            <pubDate>Mon, 15 Apr 2019 15:53:30 +0000</pubDate>
            <comments>https://example.com/post/1</comments>
            <description>Example description of this item.</description>
        </item>
    </channel>
</rss>
```

2.3 Formát Atom

Stejně jako formát RSS Atom je založen na značkovacím jazyce XML verze 1.0.

Pro datum a čas se používá formát RFC 3399[6]. Je profilem formátu ISO 8601 a slouží k použití v protokolech síťové vrstvy.

Kořenovým elementem je element `<feed>`. Je obdobou elementu `<channel>` v RSS. Obdobu obalovacímu elementu `<rss>` Atom nemá.

Kanál ve formátu Atom musí obsahovat tyto metadatové elementy:

- `<id>` – je trvalý IRI – Internationalized Resource Identifier (mezinárodní identifikátor zdroje) řetězec, který jednoznačně identifikuje kanál.
- `<title>` – název kanálu.
- `<updated>` – datum a čas kdy kanál byl naposledy úpraven.

Po metadatech může následovat libovolný počet článku, které jsou označovány elementem `entry`.

Do seznamu povinných metadatových elementů elementu `<entry>` patří stejné elementy jako u elementu `<feed>`. Navíc `<entry>` musí povinně obsahovat jeden či více elementů `<author>`. V případě, že element `<feed>` již obsahuje element `<author>`, může být u článku vynechán.

Nepovinným elementem, který se vyskytuje nejvíc je element `<link>`. Nejčastěji odkazuje na vnější zdroje jako například obrázky.

[5] HTML entita https://cs.wikipedia.org/w/index.php?title=HTML_entita&oldid=14027263

[6] RFC 3399 https://tools.ietf.org/html/rfc3339

Velký rozdíl mezi RSS a Atomem se týká textu článku. RSS má pro text článku vyhrazen element `<description>`, který se má použit bez ohledu na to, jestli je to jen krátký souhrn nebo celý text. V Atomu se naopak dělá rozdíl mezi souhrnem a celým textem. Souhrn musí být vložen do elementu `<summary>`, celý text – do `<content>`. Každý článek smí obsahovat maximálně jeden element `<summary>` a `<content>`.

Příklad Atom dokumentu:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Example Feed Title</title>
    <link href="https://example.com/"/>
    <updated>2019-01-01T00:00:00Z</updated>
    <author>
        <name>John Doe</name>
    </author>
    <id>02e549b46f5a0cc3df34752beb5e5a6e</id>

    <entry>
        <title>Example Entry Title</title>
        <link href="https://example.org/blog/1/"/>
        <id>8735a92509ef17ba6e888a56acc7ef2c</id>
        <updated>2019-01-01T00:00:00Z</updated>
        <summary>Example entry summary.</summary>
    </entry>
</feed>
```

2.4 Princip práce webových kanálů

Před tím, než někdo by měl možnost odbírat aktualizace z RSS kanálu musí se vytvořit obsah RSS dokumentu a ten dokument se musí zveřejnit.

Vytvoření a zveřejnění RSS dokumentu musí zajistit správce webu. Existuje tři možnosti, jak vytvořit RSS dokument:

- Ručně. Nejpracnější způsob. Může se používat, pokud jde o jednodušší webové stránky vytvořené na zakázku.
- Použití knihoven. Potřebuje trochu programování, ale povoluje automatizovat generování RSS dokumentu. Používá se u více pokročilejších webu vytvořených na zakázku.
- Automaticky. Tato možnost existuje, pokud je využíván CMS - Content Management System (systém pro správu obsahu), která při zveřejnění článku automaticky přidává ho do RSS dokumentu a odmazává starší články. Tato možnost je také podporována mnoha blogovacími služby jako například Medium, WordPress.org, Blogger, Squarespace a dalšími.

Po vytvoření obsah RSS dokumentu se ukládá do souboru na server. Tento soubor nebo jeho obsah musí být zveřejněn. Soubor může být zveřejněn přímo:

`https://example.com/rss.xml`

nebo může být zveřejněn jeho obsah přes speciální koncový bod[7]:

[7] Koncový bod https://cs.wikipedia.org/wiki/Koncový_bod

```
https://example.com/rss
```

Po zveřejnění RSS dokumentu specializovaný program, který se nazývá RSS čtečka může přes tento odkaz odbírat aktuální data o obsahu webu.

RSS čtečka je klientský program, který komunikuje se serverem prostřednictvím protokolu HTTP – Hypertext Transfer Protocol a rozhrání REST – Representational State Transfer. Pro získaní RSS dokumentu program odesílá HTTP požadavek typu GET s uvedením URL – Uniform Resource Locator (jednotná adresa zdroje) požadovaného dokumentu:

```
GET rss
Host: example.com
```

Program dále může zpracovat získaný obsah podle svého uvážení a zobrazit výsledek zpracování uživateli.

2.5 Výhody a nevýhody RSS

RSS a online syndikace obecně mají své výhody a nevýhody. Podíváme se ně z jak z pohledu uživatelů, tak i z pohledu autorů obsahu.

Výhody pro uživatele:

- Všechny aktualizace v jednom místě. Není potřeba ručně hledat aktuální informace každého zdroje zvlášť.
- Není potřeba poskytovat žádné osobní údaje, jako například adresa elektronické pošty.
- Jednoduchá správa kanálů a článků.
- Některé čtečky umožňují upravovat vzhled a způsob ražení článků. Uživatel tak může přizpůsobit proces čtení svým potřebám.
- Některé čtečky umožňují vyhledávání kanálu na základě klíčových slov.
- Pro zastavení odběru stačí jednoduše smazat kanál ze čtečky. Některé čtečky umožňují zastavení odběru i bez smazání.

Nevýhody pro uživatele:

- Novinářské webové stránky často poskytují jen značně zkrácené verze svých článků.
- Většina čteček nepodporuje přehrávání videí a obrázků ve formátu GIF.
- Většina webových stránek, které lze číst jen na základě předplatného neposkytuje kanály k odběru.

Výhody pro autory:

- Zvyšuje návštěvnost webových stránek [32].
- Při odběru kanálu se mohou generovat zpětné odkazy na webové stránky autora. Větší počet zpětných odkazů zlepšuje pozici webu ve výsledcích internetových vyhledávačů [33].

Nevýhody pro autory:

- Problematický sběr statistik. Například, nelze přesně určit čtenost článků a počet odběratelů kanálu.
- Do obsahu kanálu nelze vložit interaktivní elementy, což omezuje autory, které chtějí sdílet články se zajímavějším obsahem.

2.6 Analýza existujících řešení

Kvůli tomu, že protokoly RSS a Atom existují už více než 16 let a jsou stále populární, na trhu mobilních aplikací existuje mnoho různých řešení. Výsledek vyhledávání dotazu „rss reader“ v App Store obsahuje desítky aplikací. Tyto aplikace jsou dostupné buď zdarma, nebo v placené nebo předplacené formě.

Je potřeba se zmínit, že všechny analyzované aplikace mají určitou základní funkcionalitu. Všechny čtečky podporují formáty RSS a Atom, umí přidávat kanály na základě URL, zobrazovat články a označovat je za oblíbené.

V této analýze jsou představeny aplikace, jejichž uživatelem jsem byl s cílem najít čtečku, která by odpovídala mým potřebám. Kvůli tomu, že většina aplikací mají velmi podobnou funkcionalitu, která se skoro neliší od základní pro analýzu jsem zvolil aplikace s nejzajímavější funkcionalitou. K těmto aplikacím patří **Newsify**[8], **Reeder**[9] a **Unread**[10].

2.6.1 Newsify

Má standardní rozhrání ve formě tabulek. Výhodou je, že nepotřebuje účet pro použití. Obrovské množství nastavení, kterých jsou podle mě až příliš mnoho a jejich názvy jsou často docela matoucí. Při přidávání kanálu se nabízí seznam některých předvyplněných kanálu. Také existuje hledání podle klíčových slov, ale není propracované, a proto výsledky často nejsou příliš užitečné. Kanály lze zařazovat do složek. Články lze číst i z jednotlivých kanálů (to jest nejen ze všech najednou), ale míchají se tam nové články se starými a již přečtenými, což podle mě snižuje použitelnost. Oblíbené články zařazovat do složek nejde. Pro některé elementy rozhrání funguje gesto dlouhého stisku, které je velmi nepatrné a jehož vyvolání nezpůsobuje žádnou zpětnou vazbu, což vede k tomu, že uživatel ve většině případů ani nevšimne, že se něco stalo. Verze zdarma obsahuje reklamu, její vypnutí stojí 79 Kč.

2.6.2 Reeder

Je jedna z prvních RSS čteček, která se objevila na iOS. Má dostatečný počet rozumných nastavení pro úpravu vzhledu. Obsahuje na výběr 4 barevné motivy stejné jako v aplikaci Apple Books. Umožňuje zařazovat kanály do složek. Má integraci s obrovským množstvím služeb třetích stran. Navigace skrz aplikaci je občas složitá, protože je zčásti založená gestech a ikonách, které ne vždy mají zřejmý vyznám. Aplikace je placená a stojí 129 Kč.

2.6.3 Unread

Tato čtečka má v App Store velké množství pozitivních hodnocení a je chválena recenzenty, ale vše její funkce, kvůli kterým byla dobře hodnocená pro mě naopak její použitelnost buď

[8] https://itunes.apple.com/us/app/newsify-your-news-blog-rss/id510153374
[9] https://itunes.apple.com/us/app/reeder-4/id1449412357
[10] https://itunes.apple.com/us/app/unread-rss-reader/id1252376153

neovlivnily, nebo snížily. Mezi tyto funkce patří velké počet barevných motivu a navigace skoro zcela založená na gestech. Polovina motivů se mi vůbec nepřišla jako použitelná pro čtení kvůli zvoleným barvám. Druhá polovina je pro čtení dobrá, ale liší se jen akcentovou barvou. Navigace založená na gestech se mi přišla jako skoro vůbec nepoužitelná. Bylo potřeba zapamatovat ze které strany obrazovky je potřeba vyvolat gesto a co se stane po jeho vyvolání. Navíc výsledek vyvolání gesta se mění na každé obrazovce. Dalším nedostatkem aplikace je to, že ji nelze používat bez účtu agregační služby třetí strany a neumožňuje spravovat kanály. Posledním hlavním nedostatek je to, že ve verzi zdarma lze přečíst jen 50 článků, potom jen 3 články za den. Placená verze stojí 249 Kč.

2.6.4 Shrnutí

Po analýze lze usoudit, že i přesto, že existuje mnoho řešení je stále prostor na vylepšení. Hlavně ze strany navigace, práce s články, použití čtečky bez nutnosti mít účet služby třetí strany.

Kapitola 3

Programování aplikací pro operační systém iOS

iOS (dříve iPhoneOS) je proprietární mobilní operační systém vytvořený a vyvíjený společností Apple, Inc. Byl poprvé představen v roce 2007 jako operační systém pro iPhone. Dnes na iOS pracují jak zařízení iPhone, tak i iPad a iPod. Je to druhý nejrozšířenější na světě mobilní operační systém pro mobilní zařízení [34]. Poslední verze, iOS 12, byla vydána v září 2018. Významným krokem ve vývoji iOS se stalo vydání iOS 11 v roce 2017, která ukončila podporu 32 bitových aplikací a přístrojů [23].

Pro vytvoření aplikací pro své zařízení Apple vytvořila několik nástrojů. K tímto nástrojům patří vývojové prostředí Xcode a programovací jazyky Objective-C a Swift.

3.1 Vývojové prostředí Xcode

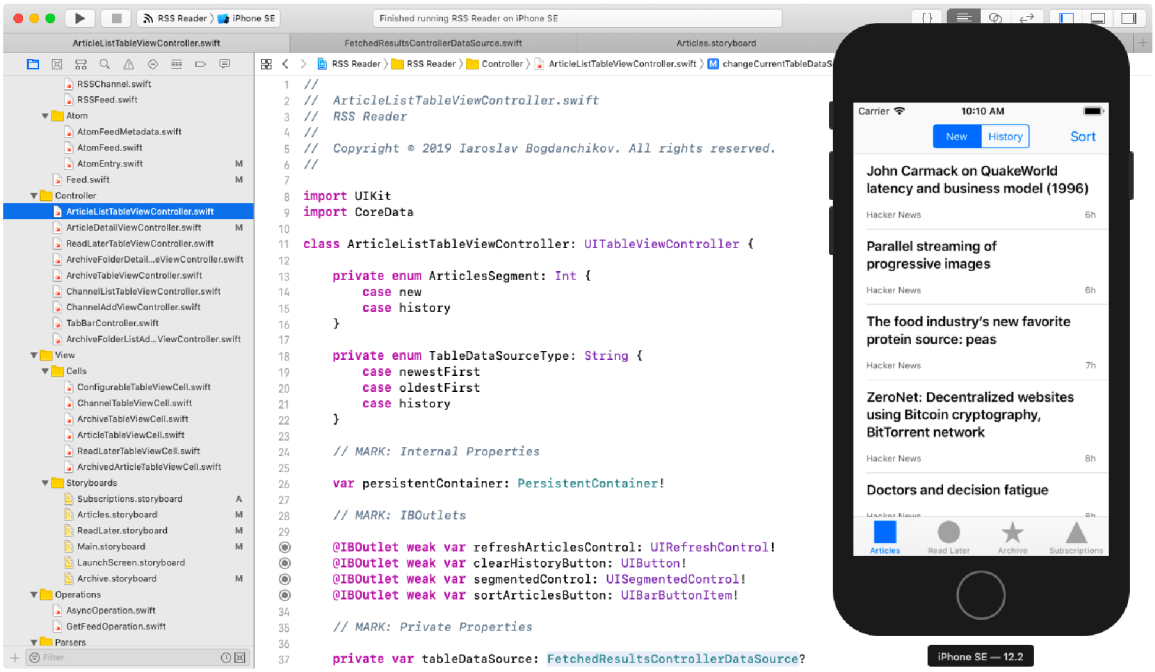
Xcode (viz obrázek 3.1) je vývojové prostředí společnosti Apple. Je dostupné pouze pro operační systém macOS. Poslední dostupná verze je 10.2. Umožňuje vytvářet aplikace pro všechny zařízení a operační systémy vyráběné a vyvíjené Apple.

Zahrnuje v sebe následující prvky:

- Textový editor.
- Interface Builder – program pro vytvoření grafického uživatelského rozhrání aplikací.
- Simulator – program pro testování aplikací na simulátorech odpovídajících skutečným zařízením Apple.
- Debugger – program pro testování a ladění aplikace.
- Instruments – program pro analýzu výkonnosti aplikace.
- Swift Playgrounds – vývojové prostředí pro experimentování se Swift kódem bez nutnosti vytvářet nějakou aplikaci.

Xcode také podporuje verzování zdrojového kódu pomocí systémů Git a Subversion [26]. Umí pracovat i s aktivy (anglický – assset) aplikace, tj. obrázky, ikony atd.

Je založen na souboru překladových nástrojů LLVM (Low Level Virtual Machine), který zahrnuje překladače Clang a GCC a díky tomu má plnou podporu jazyků Swift, C, C++ a Objective-C [25].



Obrázek 3.1: Ukázka vývojového prostředí Xcode se spuštěným Simulatorem

3.2 Programovací jazyky

Objective-C je objektově orientovaný programovací jazyk [3], který je založen na programovacím jazyku C s přidaným zasíláním zpráv objektům jako v jazyce Smalltalk. Do roku 2014 byl hlavním programovacím jazykem pro programování aplikací pro macOS a iOS. V roce 2014 Apple představila nový programovací jazyk – Swift. Od té doby Swift se stává de facto hlavním jazykem pro nové aplikace.

Swift [2] je univerzální multi-paradigmatický kompilovaný programovací jazyk s otevřeným zdrojovým kódem. Swift zachovává mnozí koncepty, které existuji v Objective-C, ale jsou navrženy s ohledem na větší bezpečnost výsledného kódu. Poslední verze jazyka je 5.0.

3.3 Zásady programování aplikací pro iOS

V této podkapitole budou popsány zásady programování aplikací pro iOS. Především návrhové vzory, které by se měly použít při návrhu a implementaci aplikací. Také bude popsán framework Core Data, který se používá pro perzistenci dat a Auto Layout – systém omezení pro vytváření dynamických uživatelských rozhrání.

3.3.1 Návrhové vzory používané v iOS aplikacích

V programovacích prostředích Cocoa a Cocoa Touch, které obsahují systémové knihovny a aplikační rámce pro vytvoření aplikací jsou důsledně používány určité návrhové vzory [1]. Tyto návrhové jsou silně doporučeny společností Apple i pro použití v aplikacích. Nejdůležitější z nich budou popsány v tomto oddílu.

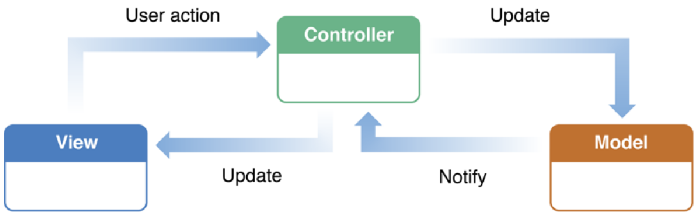
Model - View - Controller

Nejdůležitějším návrhovým vzorem je Model - View - Controller (MVC) [8]. MVC je architektonický návrhový vzor, který se tradičně používá pro aplikace s grafickým uživatelským rozhraním. Je klíčovou komponentou pro dobře navrženou aplikaci. Je rozdělen na tři komponenty: model, pohled a řadič (viz obrázek 3.2). Jeho výhodou je to, že dovoluje nejen rozdělit objekty v aplikaci do třech samostatných vrstev, ale také definovat to jakým způsobem se tyto objekty mají komunikovat. Díky těmto vlastnostem závislost mezi objekty různých vrstev je nízká a roste znovupoužitelnost objektů.

Pod Modelem se rozumí data, která jsou specifická pro danou aplikaci. V modelu je také definována business logika nad těmito daty.

Vrstva View definuje objekty, které jsou viditelné uživateli. Obvykle vhodným způsobem zobrazuje uživateli data z modelu a objekty, které umožňují tato data nějakým způsobem měnit. Objekty této vrstvy umí odpovídat na akce uživatele.

Objekty vrstvy Controller zprostředkuji komunikaci mezi objekty vrstvy Model a objekty vrstvy View. Pokud uživatel chce provést nějakou akci, kontrolér dostává o tom zprávu z view, interpretuje ji a předává ji do modelu, aby se v něm provedli odpovídající změny. Stejně tak, pokud došlo k nějakým změnám v modelu, model o tom informuje kontrolér, předává mu nová data, kontrolér pak tuto změnu předává objektům vrstvy View, aby se nová data mohla zobrazit uživateli.



Obrázek 3.2: Diagram návrhového vzoru Model - View - Controller (převzato z [8]).

Delegation

Delegation [7] je návrhový vzor, ve kterém se zúčastňuji dva objekty – delegující objekt a delegovaný objekt. Delegovaný objekt může provádět akce v odpověď na zprávy delegujícího objektu. V průběhu vykonávání programu, delegující objekt může zaslat zprávu delegovanému objektu, která říká, že buď již nastala nebo nastane nějaká událost a delegovaný objekt má možnost na tuto zprávu určitým způsobem zareagovat. Výhodou tohoto návrhového vzoru je možnost implementace specifického chování bez použití dědičnosti. Je často používán v aplikačních rámcích operačních systémů macOS a iOS.

Target - Action

Target - Action [9] je návrhový vzor, který je nejvíc užitečný při implementaci grafických uživatelských rozhraní. Používá se především u prvků uživatelského rozhraní, jako například tlačítko, slider a podobně, které umí odpovídat na akce uživatele (zmáčknutí tlačítka, posun slideru). Tímto prvkům se říká cíl (target). Každému takovému objektu lze přidělit nějakou zprávu (action), která mu pak bude zaslána při provedení nad ním nějaké akce. Cílový objekt odpovídá na zaslanou zprávu vykonáním metody odpovídající dané zprávě.

3.3.2 Auto Layout

Původně zařízení iPhone měli stejnou velikost obrazovky a rozlišení, ale s představením iPhone 4S a dalších modelů se tyto parametry začaly hodně lišit. Pro vývojáře to znamenalo, že museli přizpůsobovat jejich aplikace pro větší počet zařízení, co znamenalo značné zkomplikování vývoje aplikací. Kvůli tomu v iOS 6 a Xcode 4.5 Apple zavedla [10] nový systém, který se nazývá Auto Layout [4]. Je to systém omezení, které se dá nastavit jak s použitím grafického uživatelského rozhraní v Interface Builder, tak i programově. Omezení se nastavuji pro každý prvek uživatelského rozhraní pomocí pravidel, které říkají, kterou vzdálenost daný prvek má mít od jiných prvků, jakou má mít velikost a podobně. Na základě těchto omezení uživatelské rozhraní se může dynamicky přizpůsobovat obrazovkám různé velikosti a různým rozlišením bez toho, aby se dělalo nějaké další ruční vypočítávání správných rozměrů.

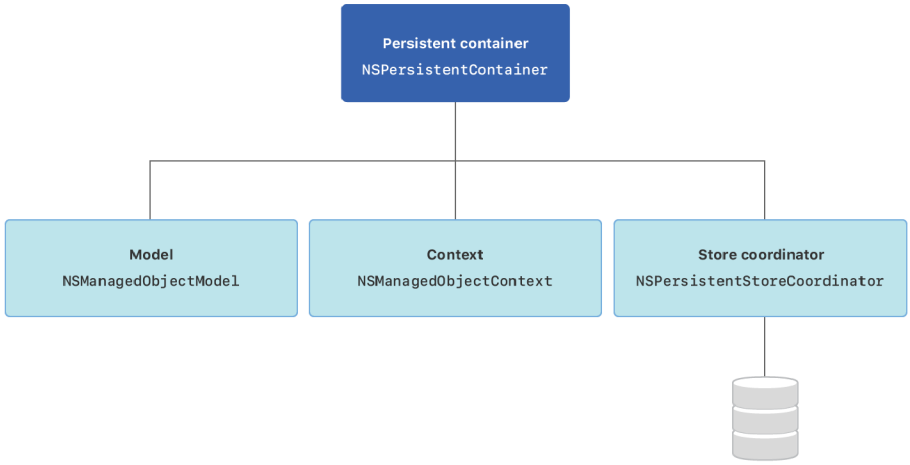
3.3.3 Core Data

Většina aplikací potřebuje uložit nějaká data, například pro off - line použití. Apple k tomuto účelu nabízí aplikační rámec Core Data. Core Data reprezentuje vrstvu Model z návrhového vzoru MVC. Core Data model se obvykle vytváří [12] tím, že se vytvoří soubor s příponou .xcdatamodel. Uvnitř Core Data pracuje primárně se souborem relačního databázového systému SQLite, který se často používá pro uložení dat přímo do aplikace. Není to jediná možnost [6], lze také nastavit, aby Core Data pracoval se soubory ve formátu XML nebo binárními soubory. Core Data nepracuje s těmito soubory přímo, ale zapouzdřuje je a nabízí práci s daty na vyšší úrovni pomocí entit, jejich atributů a vztahu mezi entitami.

Jedné entitě musí odpovídat třída, která je podtřídou třídy NSManagedObject [5]. Jedna instance takové třídy odpovídá jednomu řádku v databázové tabulce. Vztahy popisuji, jak jedna entita ovlivňuje jinou. Mezi nejdůležitější vlastnosti vztahu patří název, cílová entita a kardinalita. Entity a vztahy mezi nimi lze vytvořit pomoci programu Core Data model editor, který je součásti vývojového prostředí Xcode.

Pro práci s Core Data je potřeba v aplikaci vytvořit takzvaný Core Data Stack [18] (viz obrázek 3.3). Typicky se to děla při startu aplikace. Vytváří se instance třídy NSPersistentController, která zahrnuje v sobě objekty tříd NSManagedObjectModel, NSManagedObjectContext a NSPersistentStoreCoordinator a provádí jejich inicializaci:

- Instance NSManagedObjectModel odpovídá souboru .xcdatamodel a obsahuje informaci o entitách, jejich atributech a vztazích.
- Instance NSManagedObjectContext sleduje změny instancí entit.
- NSPersistentStoreCoordinator ukládá a načítá instance entit do/z databáze.



Obrázek 3.3: Diagram Core Data Stack (převzato z [18]).

Kapitola 4

Návrh aplikace

V této části je popsán návrh aplikace RSS Reader z pohledu uživatelského rozhrání a datového modelu.

4.1 Požadovaná funkcionalita

Na základě analýzy existujících řešení byl sestaven seznam požadovaných funkcí, které by měl RSS Reader umět:

- Podpora formátů RSS a Atom.
- Jednoduchá navigace mezi části programů.
- Odběr kanálu na základě URL.
- Vypnutí/zapnutí odběru kanálu bez toho, aby byl smazán.
- Zobrazení všech nepřečtených článků.
- Umožnit rychlým a přirozeným způsobem přidat článek do seznamů pro čtení.
- Uložení článků do archivu s možností zařazení článků do složek.
- Zachování historie přečtených článků s možností smazání.

4.2 Uživatelské rozhrání

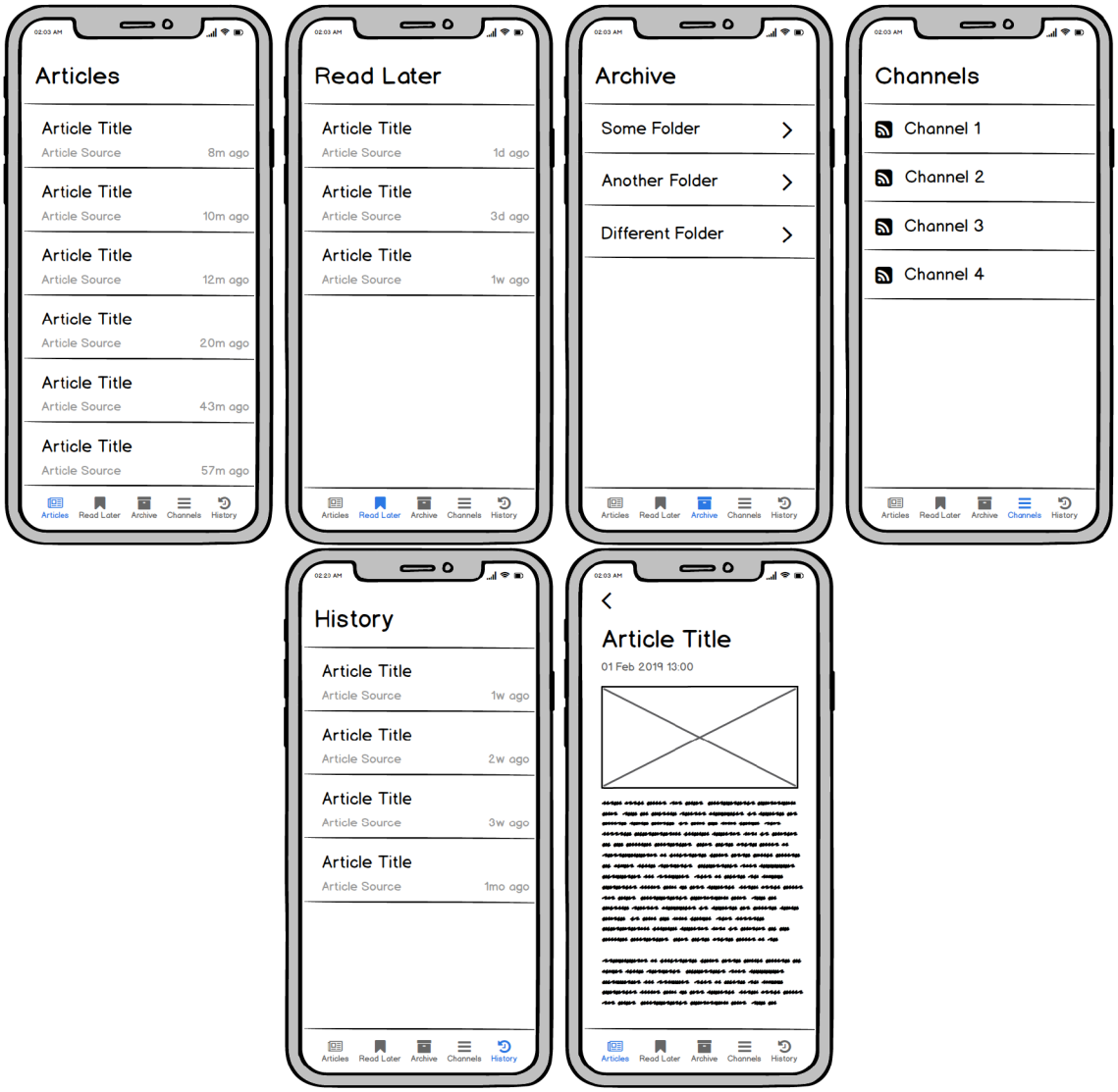
Jedním z hlavních cílů uživatelského rozhrání aplikace RSS Reader je zjednodušit navigaci, která v existujících řešeních je zbytečně složitá. Proto na začátku rozhrání bylo rozděleno do pěti základních částí:

- Articles – pouze nové články.
- Read Later – seznam všech článků, které byly uloženy pro pozdější přečtení.
- Archive – archiv, ve kterém články jsou zařazeny do složek vytvořených uživatelem.
- Channels – seznam kanálů, které odebírá uživatel.
- History – seznam všech přečtených článků.

Přístup k jednotlivým částem je umožněn přes prvek uživatelského rozhraní, který se nazývá **Tab Bar** [19]. Tab bar je prvek, který se používá striktně pro navigaci a jeho účelem je organizovat informaci na úrovni aplikace a zploštit navigační hierarchii. Použití tab baru tak vede k zjednodušení navigace. Tab bar se zobrazuje ve spodní části obrazovky.

Pro zobrazení seznamů článků a odbíraných kanálů byly použity tabulky [20]. Tabulky umožňují zobrazit jak malé, tak velké množství dat čistě a efektivně. Tabulky jsou ideálním prvkem pro zobrazení textového obsahu. Navíc řádky tabulky umožňují provádět nad nimi rychlé akce, jako například smazání. Existuje ještě alternativa v podobě kolekcí [11]. Kolekce podporují větší úpravu vzhledu položek, ale Apple doporučuje používat kolekce především pro vizuální obsah, například obrázky.

Prvotní verze modelu uživatelského rozhraní lze vidět na obrázku 4.1.



Obrázek 4.1: Model uživatelského rozhraní

Časem se prvotní model uživatelského rozhraní trochu změnil. Místo samostatného tabu pro historii se tab Articles se rozdělil na dva segmenty pomocí prvku **Segmented Control** [17]. Výchozím segmentem je segment **New**, který zobrazuje nové články. Druhým

segmentem je segment **History**, kde se zobrazují všechny staré články. Změnu lze vidět na obrázcích 4.2 a 4.3.

Obrázek 4.2: Obrazovka s novými články

Obrázek 4.3: Obrazovka se starými článků

4.3 Datový model

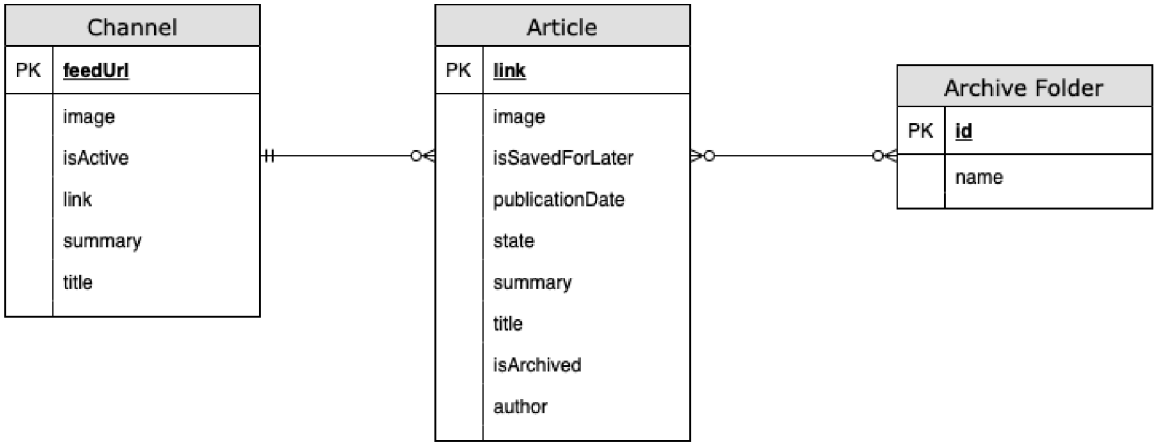
Pro správnou práci aplikace RSS Reader je potřeba ukládat některá data. Za prvé bude potřeba uložit data kanálu. Tak vznikne entita **Channel**. Aby bylo možné pravidelně stahovat aktualizovaný obsah z kanálu je potřeba uložit jeho adresu. Adresa kanálu bude uložena v atributu **feedUrl**, který zároveň bude i primárním klíčem entity. Dále podle požadované funkcionality potřebujeme rozeznat, jestli je kanál aktivní či není. Tato informace bude uložena v atributu **isActive**. Aby šlo kanál jednoduše identifikovat v atributu **title** bude uložen název kanálu a v atributu **image** obrázek kanálu. Popis kanálu je ukládán do atributu **summary**.

Za druhé je potřeba ukládat data o každém staženém článku. Tyto data budou ukládána do entity **Article**. U článku je potřeba minimálně uložit nadpis – atribut **title**, obsah – atribut **summary** a datum zveřejnění – **publicationDate**. Součásti některých článků může být i obrázek. Obrázky budou uloženy do atributu **image**. Podle atributu **state** se rozlišuje v jakém stavu je článek. Rozlišují se tři stavy: nový, přečtený a starý. Atribut **isSavedForLater** určuje, jestli článek byl uložen do seznamu pro pozdější přečtení. Atribut **isArchived** určuje, jestli článek byl uložen do archivu. Primárním klíčem entity **Article** je atribut **link**, který obsahuje odkaz na webovou stránku článku. Pokud je k dispozici jméno autora článku bude uloženo do atributu **author**.

Aby šlo články ukládat do složek v archivu byla vytvořena entita **Archive Folder**. Názvu složky odpovídá atribut **name**. Primárním klíčem entity je atribut **id**.

Kanál může obsahovat libovolný počet článků, proto relace mezi entitou **Channel** a entitou **Article** je 1:N. Článek může být uložen do libovolného počtu složek a složka může obsahovat libovolný počet článků, proto relace mezi entitami **Article** a **Archive Folder** je N:M.

ER diagram výsledného datového modelu lze vidět na obrázku 4.4.



Obrázek 4.4: Model použitý v aplikací RSS Reader

Kapitola 5

Implementace

Na základě znalostí získaných v kapitole 2 a na základě návrhu v kapitole 4 byla aplikace RSS Reader implementována. Pro implementaci byl použit programovací jazyk Swift verze 5.0. Aplikace RSS Reader byla implementována pro operační systém iOS verze 11.4 – 12.2. Při implementaci nebyly použité žádné knihovny.

5.1 Struktura aplikace

Zdrojové soubory aplikace RSS Reader zařazeny do několika složek podle významu, který v dané aplikace mají.

Aplikace se skládá z následujících složek:

- **Core** – jádro projektu. V této složce se nachází soubory nezbytné pro základní práci aplikace.
- **Model** – obsahuje vše co se týká datového modelu aplikace. Dále obsahuje následující podsložky:
  - **Core Data** – obsahuje soubor s Core Data modelem a vygenerované třídy entit.
  - **RSS** – obsahuje třídy pro formát RSS.
  - **Atom** – obsahuje třídy pro formát Atom.
- **Controller** – obsahuje všechny kontroléry, které svazuj datový model s grafickým rozhraním.
- **View** – obsahuje soubory, které definují grafické uživatelské rozhrání. Obsahuje podsložky:
  - **Cells** – zdrojové soubory, které definuje specializované chování buněk tabulek.
  - **Storyboards** – soubory, které byly vytvořené v Interface Builder.

5.2 Implementace jádra

Do teto kategorie patří třídy, bez kterých RSS Reader nemůže pracovat. Nejzajímavějšími z nich jsou třídy `XMLFeedParser`, která se používá pro zpracování XML dokumentů a třída `GetFeedOperation`, která se zabývá stahováním a uložením dat webových kanálů.

5.2.1 Třída XMLFeedParser

Třída XMLFeedParser umožňuje zpracovat stažený soubor webového kanálu.

`XMLFeedParser` je delegátem [27] instance třídy `XMLParser` [28]. `XMLParser` je syntaktický analyzátor XML dokumentu řízený událostmi, který pomocí zpráv svému delegátu informuje ho o tom, že byl zpracován nějaký prvek (například element, atribut atd.) XML dokumentu.

Všechny zpracované prvky XML dokumentu a jejich hodnoty se ukládají do dvou instančních proměnných: `metadata` a `content`. Do proměnné `metadata` se ukládají metadata kanálu a do proměnné `content` jeho obsah (články). Obě proměnné jsou typu asociativní pole. Jejími klíči jsou řetězce v tečkové notaci, které odpovídají cestě k hodnotě v XML dokumentu. Například, cesta pro hodnotu názvu kanálu „Channel's Title“ z dokumentu 2.2 bude `rss.channel.title`. Pro hodnotu atributu XML elementu se používá stejný princip, například cesta pro hodnotu URL webových stránek kanálu z dokumentu 2.3 bude `feed.link.href`.

Formát je určen podle prvního elementu dokumentu.

Po zpracování celého dokumentu analyzátorem se použije metoda *constructFeed(for:)*, která vytvoří objekt správného typu podle poskytnutého formátu kanálu. Pokud byl zpracován dokument ve formátu RSS vytvoří se instance třídy RSSFeed (viz 5.3.1), pokud byl dokument ve formátu Atom, pak se vytvoří instance třídy AtomFeed (viz 5.3.1). Metodou se vrací objekt typu `Feed`.

5.2.2 Třída GetFeedOperation

Třídě `GetFeedOperation` přes konstruktor se předávají URL XML souboru a objekt, pomocí kterého se budou data ukládat do databáze. Cílem funkce `fetchFeed()` je stáhnout obsah XML souboru a předat ho dalším komponentám ke zpracování.

XML soubor se stahuje pomoci instance třídy `URLSession` [24]. URLSession poskytuje API pro práci s úlohy, které se týkají přenosu dat po síti. API URLSession je asynchronní, a proto výsledek se předává buď přes uzávěr nebo pomocí delegování. Delegování se používá spíše pro složitější případy, kde je vyžadována vyšší úroveň kontroly nad procesem. Zde bude stačit zpracovat výsledek v uzávěru. Proto byla použitá metoda `dataTask(with: completionHandler:)` [13]. Tato metoda stahuje data z předaného koncového bodu a drží je v paměti. Uzávěru obsahuje kód, který zpracuje výsledek. Pokud při stahování dojde k chybě úloha se skončí. Pokud se data úspěšné stáhnou ze serveru bude vytvořena instance třídy `XMLFeedParser`, která zpracuje stažená XML data. Pokud se data podařilo úspěšné zpracovat a vytvořit z nich objekt kanálu, který je typu `Feed` 5.3.1, zavolá se metoda `downloadImages(for:)`, která stáhne všechny obrázky, pro které existuji odkazy v objektu kanálu a uloží je do něj. Obrázky se stahuji stejným způsobem jako XML soubor. Kvůli tomu, že kanály často obsahují články s obrázky a je potřeba je všechny stáhnout operace musí počkat až se dokončí stahování všech obrázku. To lze zaručit použitím třídy `DispatchGroup` [14]. DispatchGroup umožňuje synchronizovat skupinu úloh v rámci jedne jednotky. Je potřeba přidat do operace `GetFeedOperation` instanční proměnnou, která bude obsahovat instanci třídy `DispatchGroup`. Před vytvořením úlohy se stahováním obrázků je potřeba poslat instancí `DispatchGroup` zprávu `enter`, a po dokončení zpracování úlohy zprávu `leave()`. Po vytvoření všech úloh je nutné zavolat metodu `wait()` instancí `DispatchGroup`, aby operace počkala, až se dokončí všechny stahovací úlohy. Potom se objekt typu `Feed` uloží do databáze pomoci objektu typu `PersistentContainer` (viz 5.3.3).

5.3 Implementace vrstvy Model

Implementace modelu se skládala ze dvou částí: vytvoření datových typů pro práci s daty kanálů a článků a perzistence dat.

5.3.1 Datové typy pro práci s kanály a články

Pro práci s daty kanálu a článku byly vytvořeny nové datové typy. Kvůli tomu, že kanály jak ve formátu RSS, tak ve formátu Atom obsahuji hodně podobných údajů a mají podobnou strukturu jsem rozhodl tyto údaje sjednotit do protokolů. Základem pro oba formáty se stal typ `Feed`. `Feed` je protokol, který definuje jeden kanál celkově a požaduje, aby třída nebo struktura implementující protokol měla tři atributy:

- `format` – definuje typ kanálu. Je typu `SupportedFeedType`. `SupportedFeedType` je číselník, který obsahuje dvě hodnoty `rss` a `atom`.
- `metadata` – definuje objekt s údaji o kanálu. Je typu `FeedMetadata`.
- `content` – definuje pole objektů článků. Objekty jsou typu `FeedItem`.

`FeedMetadata` je protokol, který obsahuje údaje o kanálu (název, popis, odkaz na webové stránky atd.). `FeedItem` je protokol, který označuje jeden článek kanálu.

Tyto protokoly implementuji následující struktury:

- Pro formát RSS:
  - `RSSChannel` implementuje protokol `FeedMetadata`.
  - `RSSItem` implementuje protokol `FeedItem`.
  - `RSSFeed` implementuje protokol `Feed`.
- Pro formát Atom:
  - `AtomFeedMetadata` implementuje protokol `FeedMetadata`.
  - `AtomEntry` implementuje protokol `FeedItem`.
  - `AtomFeed` implementuje protokol `Feed`.

5.3.2 Perzistence dat

Pro perzistenci dat byl zvolen aplikační rámec Core Data, který byl popsán v části 3.3.3.

5.3.3 Třída PersistentContainer

Pro práci s Core Data byla vytvořena třída `PersistentContainer`, která je podtřídou třídy `NSPersistentContainer`. Třída `PersistentContainer` byla vytvořena podle doporučení z dokumentace [18]. Tato třída zahruje Core Data Stack a může obsahovat libovolné pomocné metody pro práci s datovým úložištěm. Její jediná instance se vytváří jednou ve třídě `AppDelegate` [22] při startu aplikace. Dále se tato instance předává kořenovému view kontroléru – `TabBarController`. `TabBarController` pak přidává tuto instanci dál všem view kontrolérům, které ji potřebují.

Protokol CDPersistable

Protocol `CDPersistable` obsahuje jednu metodu `convertToManagedObject`. Třídy a struktury, které implementují tento protokol mohou být uloženy do vnitřního úložiště pomocí Core Data. V RSS Reader tento protokol implementuji struktury `RSSChannel`, `RSSItem`, `AtomFeedMetadata` a `AtomEntry`.

5.4 Implementace vrstvy View

Vrstvu view lze implementovat třemi způsoby:

- Výhradně v prostředí Interface Builder.
- Výhradně použitím programovacího jazyka.
- Kombinovaným způsobem.

Pokud je uživatelské rozhraní aplikace je dostatečně jednoduché doporučuje se použít první způsob. Je rychlejší a má výhodu, že lze hned vidět, jak bude vypadat aplikace za běhu. Nevýhodou je menší flexibilita a omezený počet typů vzhledů prvků, které lze vytvořit nebo použít.

Kvůli tomu, že uživatelské rozhraní RSS Reader je dostatečně jednoduché, tak většina uživatelského rozhraní je vytvořena v prostředí Interface Builder. S drobnými výjimky, kde jsem musel použít nastavení použitím programovacího jazyka.

Uživatelské rozhraní je implementováno ve souborech s příponou `.storyboard`: Main, Articles, ReadLater, Archive a Subscriptions.

Obrazovky, které odpovídají jednotlivým tabům tab baru jsou implementovány v souborech Articles, ReadLater, Archive a Subscriptions pro lepší udržitelnost a snazší práci s nimi.

V souboru Main je implementován tab bar, který je vstupním bodem aplikace. Tab bar dále obsahuje odkazy na ostatní obrazovky.

Všechny obrazovky byly implementovány s použitím Auto Layout. To znamená, že se aplikace bude dobře zobrazovat na všech modelech iPhone. Pro text je použita technologie Dynamic Type [21]. Použití této technologie znamená, že pokud uživatel nastaví v systému jinou velikost textu než standardní, projeví se to i v RSS Reader.

Pro správné zobrazení řádků tabulek bylo potřeba vytvořit samostatné třídy pro každý typ řádku. Tyto třídy mají být poděděny z třídy UITableViewCell.

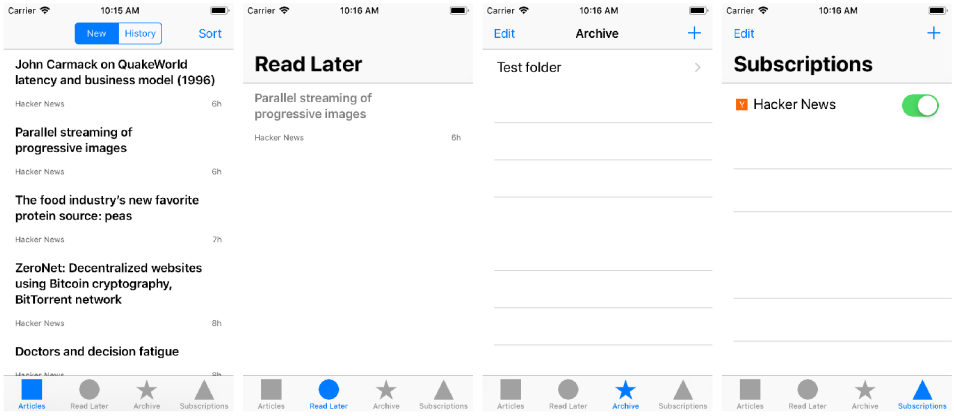
Příklady implementovaných obrazovek lze vidět na obrázku 5.1.

5.5 Implementace vrstvy Controller

Controller je vrstva, která reaguje na události z vrstvy View a na základě těchto události aktualizuje data z vrstvy Model. To znamená, že pro každý prvek uživatelského rozhraní musí existovat odpovídající kontrolér.

5.5.1 Přidávání nového kanálu

Jakmile uživatel přidá kanál do čtečky bude potřeba stáhnout obsah souboru ze zadané adresy, zpracovat tento soubor a uložit zpracovaný obsah do databáze. Kvůli tomu, že tento



Obrázek 5.1: Implementováné obrazovky aplikace RSS Reader

proces není triviální a může trvat poměrně dlouhou dobu není vhodné tuto činnost provádět na hlavním vláknu aplikace. Není to vhodné z toho důvodu, že na hlavním vlákně se provádí zpracování všech událostí, které se týkají uživatelského rozhrání a na hlavním vlákně se také provádí aktualizace všech prvku uživatelského rozhrání. Pokud na hlavním vlákně se bude provádí činnost, která trvá dlouhou dobu, způsobuje to „zmrznutí“ uživatelského rozhrání a vede to k nepříjemným uživatelským zkušenostem. Proto se musí využít technologie, která se nazývá GCD – `Grand Central Dispatch`. GCD povoluje souběžně zpravovat více úloh. Apple vždy doporučuje používat nejvyšší úroveň dostupné abstrakce a jen v případě potřeby snižovat se na nižší úroveň. Nejvyšší úroveň abstrakce při práci s GCD je abstraktní třída `Operation` [15]. Kvůli tomu, že Operation je abstraktní třída musíme ji zdědit ve vlastní třídě. Operace jsou dvojího druhu: synchronní a asynchronní. Rozdíl mezi synchronní a asynchronní operací je v tom, že synchronní operace nevytváří nové vlákno, ve kterém pak bude spuštěna její úloha a provádí ji v aktuálním vláknu. V mém případě bylo potřeba použít asynchronní operaci, aby operace nezdržovala vykonání části programu, ze kterého byla spuštěna. Asynchronní operace je implementována ve třídě `AsyncOperation`. Pro implementaci asynchronní operace bylo potřeba předefinovat metodu `start()`, která spouští vykonání úlohy a vlastnosti `isAsynchronous`, `isExecuting` a `isFinished`, udávající stav zpracovávané úlohy. Specializovaná operace potom má zdědit třídu `AsyncOperation` a předefinovat metodu `main()`, ve který bude obsahovat spuštění specializované úlohy. Specializovaná operace, která provádí stahování a zpracování XML souboru a stahování obrázku z XML souboru je implementována ve třídě `GetFeedOperation`.

5.5.2 Aktualizace odbíraných kanálů

Aktualizace odebíraných článků se uskutečňuje na záložce **Articles** při provedení gesta pull-to-refresh[1]. Jako reakce na toto gesto se volá metoda `performUpdateSession` v kontroléru `ArticleListTableViewController`. Za prvé se kontroluje, jestli instanční atribut `updateSession` již existuje, jestli ano, pak to znamená, že aktualizace kanálů již probíhá, jinak se vytvoří nový objekt třídy `UpdateSession`. Třída `UpdateSession` je implementována podle návrhového vzoru delegate a spravuje aktualizaci kanálů. Delegátem této metody je kontrolér `ArticleListTableViewController`, který dostává informace o tom, jestli aktualizace byla dokončena. Dále metoda načítá z databáze všechny aktivní kanály. Pokud

[1]Pull-to-refresh https://en.wikipedia.org/wiki/Pull-to-refresh

nějaké jsou vytváří pro každý z nich operaci `GetFeedOperation` a přidává je do objektu třídy `OperationQueue`. `OperationQueue` [16] je fronta, která spravuje vykonání operací. Pro každou vytvořenou operaci se také definuje uzávěr, podle kterého operace se má po skončení provedení své úlohy nahlásit objektu `updateSession` to, že byla dokončena.

Kapitola 6

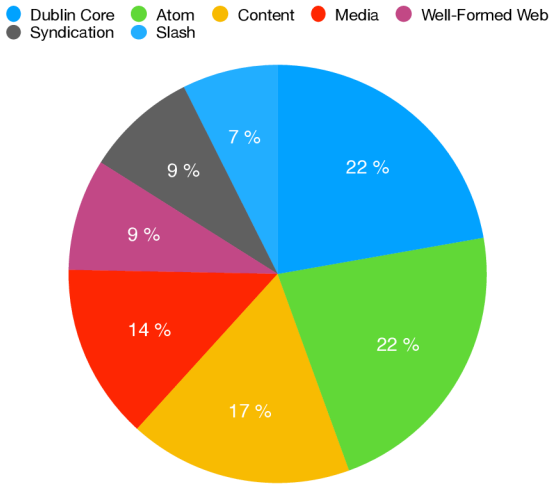
Testování

Při vývoji RSS Reader byl průběžně testován na různých typech webových zdrojů. Výsledky tohoto testování jsou shrnuty v první části této kapitole. Ve druhé části je provedeno porovnání vytvořené aplikace RSS Reader s existujícími RSS čtečkami.

RSS Reader byl testován jak v simulátorech různých iOS zařízení, tak i na reálném zařízení iPhone SE. Při testování se používala iOS verze 12.2, ale vzhledem k tomu, že cílovou verzi aplikace je verze 11.4 chování aplikace by mělo by stejné pro všechny verze iOS počínaje 11.4.

6.1 Testování na různých typech webových zdrojů

Hlavní funkcí RSS čtečky je správné zobrazení aktualizovaného obsahu webových zdrojů – článků. Pro ověření této funkcionality aplikace byla testována na vzorku třiceti pěti webových zdrojů (viz příloha A). Čtyři ze třiceti pěti zdrojů jsou ve formátu Atom, ostatní jsou ve formátu RSS 2.0. Malý počet zdrojů ve formátu Atom je způsoben jak tím, že formát Atom není tak rozšířený jako formát RSS a proto, najít webový zdroj ve formátu Atom je celkem netriviální úkol, tak i tím, že hlavním zaměřením této bakalářské práce je formát RSS. Většina webových zdrojů ze vzorku ve formátu RSS také používá různá rozšíření (viz obrázek 6.1).



Obrázek 6.1: Počet použití rozšíření formátu RSS z testovacího vzorku

I když prvotní cíle této práce nezahrnovaly implementaci jakýchkoli rozšíření v průběhu testování bylo zjištěno, že ignorování některých elementů některých rozšíření mělo vliv na kvalitu výsledného zobrazení článků, protože některé důležité části obsahu byly předávány přes elementy rozšíření místo standardních elementu formátu RSS. Příkladem takových případů může být předávání obrázku článku v elementu `<media:thumbnail>` rozšíření Media místo standardního elementu `<image>` nebo textové části obsahu v elementu `<content:encoded>` rozšíření Content místo standardního elementu `<description>`. Toto chování není zakázané a není chybou. Dokonce patří do seznamu doporučení a nejlepších praxí pro vytvoření lepších RSS dokumentů[1]. Proto RSS Reader byl o tyto elementy rozšířen.

Po úpravě aplikace RSS Reader a po provedení dalšího testování obsah všech zdrojů kromě dvou se zobrazoval správně a v plném rozsahu. První výjimkou je zdroj č. 11, u kterého se u každého článku se zobrazoval stejný obrázek jako v textu článku. Tento problém se mi podařilo vyřešit tak, že pokud text článku obsahoval stejný URL jako URL obrázku článku, pak jsem obrázek článku nezobrazoval. Druhou výjimkou je zdroj č. 34, jehož články často obsahují mnoho obrázků. Problém je v tom, že tyto obrázky nejsou součástí obsahu elementu `<content>` a jsou uvedeny jako odkazy na vnější zdroj elementu `<entry>`. Potom nelze poznat kam do textového obsahu patří tyto obrázky. Považuji tuto výjimku za nedostatek ze strany autora tohoto zdroje. Kromě toho ani jedna jiná RSS čtečka také nebyla schopná tento problém vyřešit.

6.2 Porovnání aplikace RSS Reader s existujícími aplikacemi

RSS Reader byl porovnán s aplikacemi z podkapitoly 2.6. V RSS Reader ve srovnání s ostatními aplikacemi není funkcionalita pro práci s články konkrétního kanálu. Také čtečka neumí pracovat se službami třetích stran a není možnost měnit vzhled nebo chování aplikace. RSS Reader nabízí jednodušší uživatelské rozhraní, možnost vypnout odběr kanálu bez toho, aby byl smazán a uložení článků do složek v archivu.

6.2.1 Porovnání s Newsify

Ve srovnání s Newsify ve výsledné aplikace není reklama.

Příklad zobrazení složitého článku je na obrázcích 6.2 a 6.3.

6.2.2 Porovnání s Reeder

Reeder umí načítat webovou stránku článku a extrahovat z ní text a zobrazovat tak celý text článku v případě, že obsah článku získaný přes webový kanál není úplný.

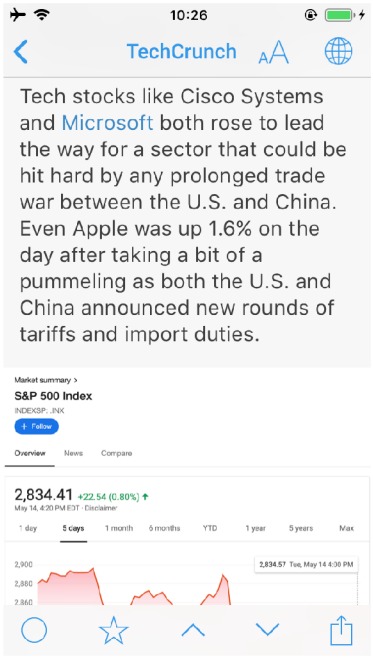
Příklad zobrazení složitého článku je na obrázcích 6.4 a 6.5.

6.2.3 Porovnání s Unread

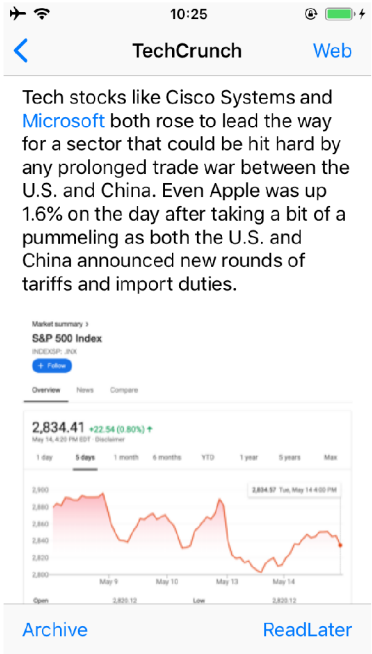
V RSS Reader na rozdíl od aplikace Unread není nutné mít účet služby třetí strany pro použití aplikace a není omezen počet článků, který lze přečíst za den.

Příklad zobrazení složitého článku je na obrázcích 6.6 a 6.7.

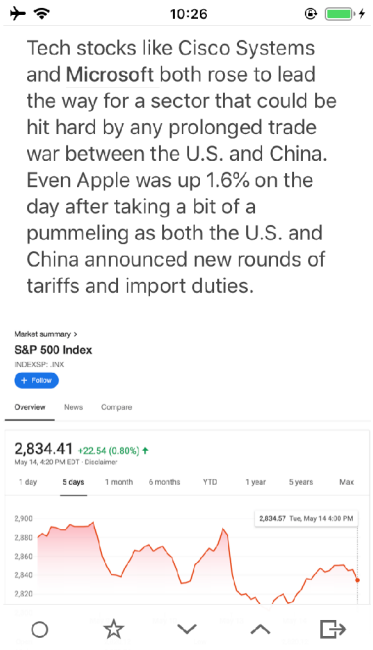
[1] RSS Best Practices Profile http://www.rssboard.org/rss-profile



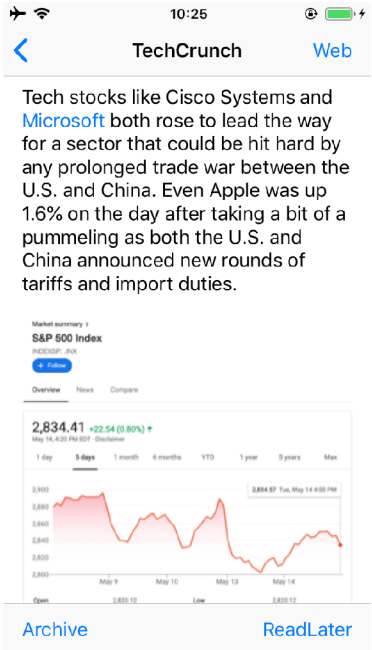
Obrázek 6.2: Zobrazení článku v Newsify



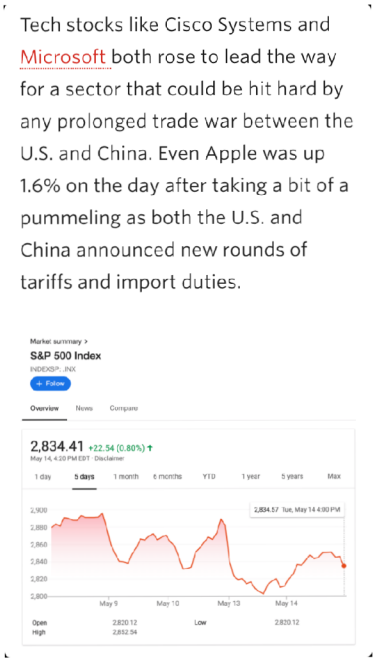
Obrázek 6.3: Zobrazení článku v RSS Reader



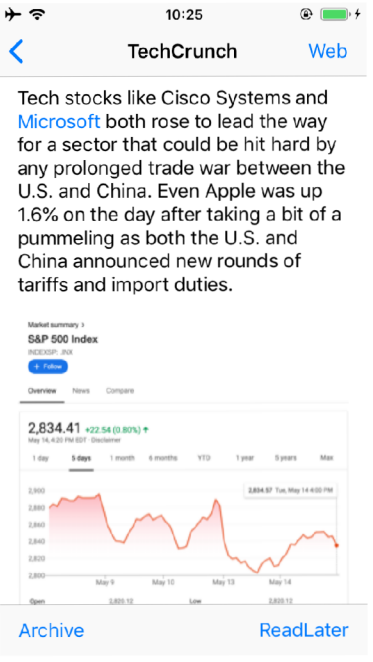
Obrázek 6.4: Zobrazení článku v Reeder



Obrázek 6.5: Zobrazení článku v RSS Reader



Obrázek 6.6: Zobrazení článku v Unread



Obrázek 6.7: Zobrazení článku v RSS Reader

Kapitola 7

Závěr

Cílem bakalářské práce bylo vytvořit RSS čtečku se základní funckionalitou typickou pro aplikace tohoto typu, ale s některými vylepšeními. Nejdřív bylo potřeba nastudovat formáty RSS a Atom (viz 2) a provést analýzu existujících řešení (viz 2.6). Dále naučit se programovací jazyk Swift, vývojové prostředí Xcode a principy programování pro operační systém iOS (viz 3).

Po získaní základních znalostí pro vytvoření RSS čtečky v kapitole 4 byl sestaven seznam požadovaných funkcí, který aplikace musí umět. V téže kapitole byl vytvořen návrh uživatelského rozhraní, datového modelu a funkcí. V kapitole 5 je popsána implementace podle vytvořeného návrhu.

Po implementaci aplikace RSS Reader byla otestována na vzorku webových zdrojů (viz 6.1) a porovnána s existujícími aplikacemi stejného typu (viz 6.2).

Tato práce mě přinesla velké zkušenosti jak v oblasti programování aplikací pro operační systém iOS, tak i při použití programovacího jazyku Swift.

Myslím, že cíl bakalářské práce byl dosáhnout. Výsledná aplikace plně splňuje mé očekávání a již jsem ji začal používat v každodenním životě.

Jako budoucí rozšíření v aplikace lze dokončit implementaci RSS rozšíření, umožnit pracovat s články konkrétního kanálu, doplnit vyhledávání webových zdrojů, vyhledávání mezi články a také možnost extrakci textu z obsahu webové stránky článku pro případ kdy nebude k dispozici celý text článků.

Literatura

[1] Apple Inc.: *Cocoa Design Patterns*. 2013, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html

[2] Apple Inc.: *Swift - Apple Developer*. 2013, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/swift/

[3] Apple Inc.: *The Objective-C Programming Language*. 2013, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html

[4] Apple Inc.: *Auto Layout Guide: Understanding Auto Layout*. 2016, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html

[5] Apple Inc.: *Data Programming Guide: Creating a Managed Object Model*. 2017, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/KeyConcepts.html#//apple_ref/doc/uid/TP40001075-CH30-SW1

[6] Apple Inc.: *Data Programming Guide: Persistent Store Types and Behaviors*. 2017, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CoreData/PersistentStoreFeatures.html#//apple_ref/doc/uid/TP40001075-CH23-SW1

[7] Apple Inc.: *Delegation*. 2018, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/Delegation.html

[8] Apple Inc.: *Model-View-Controller*. 2018, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[9] Apple Inc.: *Target Action*. 2018, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/library/archive/documentation/General/Conceptual/Devpedia-CocoaApp/TargetAction.html

[10] Apple Inc.: *Xcode Release Notes*. 2018, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/library/archive/releasenotes/DeveloperTools/RN-Xcode/Chapters/Introduction.html#//apple_ref/doc/uid/TP40001051-CH1-SW643

[11] Apple Inc.: *Collections - Views - iOS - Human Interface Guidelines - Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/design/human-interface-guidelines/ios/views/collections/

[12] Apple Inc.: *Creating a Core Data Model | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/coredata/creating_a_core_data_model

[13] Apple Inc.: *dataTask(with:completionHandler:) - URLSession | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/foundation/urlsession/1410330-datatask

[14] Apple Inc.: *DispatchGroup - Dispatch | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/dispatch/dispatchgroup

[15] Apple Inc.: *Operation - Foundation - Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/foundation/operation

[16] Apple Inc.: *OperationQueue - Foundation | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/foundation/operationqueue

[17] Apple Inc.: *Segmented Controls - Controls - iOS - Human Interface Guidelines - Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/design/human-interface-guidelines/ios/controls/segmented-controls/

[18] Apple Inc.: *Setting Up a Core Data Stack | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/coredata/setting_up_a_core_data_stack

[19] Apple Inc.: *Tab Bars - Bars - iOS - Human Interface Guidelines - Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/design/human-interface-guidelines/ios/bars/tab-bars/

[20] Apple Inc.: *Tables - Views - iOS - Human Interface Guidelines - Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/design/human-interface-guidelines/ios/views/tables/

[21] Apple Inc.: *Typography - Visual Design - iOS - Human Interface Guidelines - Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/design/human-interface-guidelines/ios/visual-design/typography/

[22] Apple Inc.: *UIApplicationDelegate - UIKit | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL
https://developer.apple.com/documentation/uikit/uiapplicationdelegate

[23] Apple Inc.: *Updating Your App from 32-Bit to 64-Bit Architecture | Apple Developer Documentation*. 2019, [Online; navštíveno 14.05.2019].
URL https://developer.apple.com/documentation/uikit/core_app/updating_your_app_from_32-bit_to_64-bit_architecture

[24] Apple Inc.: *URLSession - Foundation - Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/foundation/urlsession

[25] Apple Inc.: *Xcode — Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/xcode/

[26] Apple Inc.: *Xcode Features — Apple Developer*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/xcode/features/

[27] Apple Inc.: *XMLParserDelegate - Foundation | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL
https://developer.apple.com/documentation/foundation/xmlparserdelegate

[28] Apple Inc.: *XMLParser - Foundation | Apple Developer Documentation*. 2019, [Online; navštíveno 26.04.2019].
URL https://developer.apple.com/documentation/foundation/xmlparser

[29] Guta, R. V.; Libby, D.: *RSS 0.90 Specification*. [Online; navštíveno 13.05.2019].
URL http://www.rssboard.org/rss-0-9-0

[30] Guta, R. V.; Libby, D.: *RSS 0.91 Specification*. [Online; navštíveno 13.05.2019].
URL http://www.rssboard.org/rss-0-9-1-netscape

[31] Hammersley, B.: *Content Syndication with RSS*. O'Reilly Media, 2003, ISBN 0596003838.

[32] Hammersley, B.: *Developing Feeds with Rss and Atom*. O'Reilly Media, 2005, ISBN 0596008813.

[33] Olsen, M.: *Maximizing PageRank with New Backlinks*. In *Algorithms and Complexity*, editace T. Calamoneri; J. Diaz, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ISBN 978-3-642-13073-1, s. 37–48.

[34] Statista: *Mobile OS market share 2019*. [Online; navštíveno 14.05.2019].
URL https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[35] UserLand Software: *RSS 0.92 Specification*. [Online; navštíveno 13.05.2019].
URL http://www.rssboard.org/rss-0-9-2

[36] UserLand Software: *RSS 2.0 Specification*. [Online; navštíveno 13.05.2019].
URL http://www.rssboard.org/rss-2-0

[37] Winer, D.: *Scripting News: 7/18/2003*. [Online; navštíveno 13.05.2019].
URL http://scripting.com/2003/07/18.html#rss20News

[38] Winer, D.: *RSS came from the publishing industry | Scripting News Annex*. 2006, [Online; navštíveno 14.05.2019].
URL https://scripting.wordpress.com/2006/01/20/rss-came-from-the-publishing-industry-2/

Příloha A

Webové zdroje použité pro testování

1. https://www.cbsnews.com/latest/rss/main
2. http://feeds.bbci.co.uk/news/rss.xml
3. https://abcnews.go.com/abcnews/topstories
4. https://www.economist.com/europe/rss.xml
5. http://rss.cnn.com/rss/edition.rss
6. http://www.nytimes.com/services/xml/rss/nyt/HomePage.xml
7. http://feeds.washingtonpost.com/rss/world
8. https://www.theguardian.com/world/rss
9. http://feeds.reuters.com/reuters/topNews
10. http://tonsky.me/blog/atom.xml
11. https://vc.ru/rss/all
12. https://meduza.io/rss/all
13. https://habr.com/ru/rss/best/daily/?fl=ru
14. https://alexmak.net/feed
15. https://news.ycombinator.com/rss
16. https://www.newyorker.com/feed/everything
17. http://feeds.foxnews.com/foxnews/latest
18. https://www.engadget.com/rss.xml
19. https://www.theverge.com/rss/index.xml
20. https://www.polygon.com/rss/index.xml

21. https://www.nationalreview.com/feed/
22. http://feeds.arstechnica.com/arstechnica/index
23. https://techcrunch.com/feed/
24. http://feeds.macrumors.com/MacRumors-All
25. https://servis.idnes.cz/rss.aspx?c=zpravodaj
26. https://www.aktualne.cz/rss/
27. https://servis.lidovky.cz/rss.aspx?r=ln_domov
28. https://feeds.feedburner.com/penize?format=xml
29. https://www.zive.cz/rss/sc-47/
30. https://www.novinky.cz/rss2/
31. https://www.parlamentnilisty.cz/export/rss.aspx
32. https://www.e15.cz/rss
33. https://www.irozhlas.cz/rss/irozhlas
34. https://www.apple.com/newsroom/rss-feed.rss
35. https://developer.apple.com/news/rss/news.rss