České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Filip Dyrčík**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Klientská část aplikace SerialFreak**

Pokyny pro vypracování:

Navrhněte a implementujte klientskou část webové aplikace SerialFreak agregující zdroje informací o seriálech. Aplikace bude integrována se sociální sítí Facebook pro realizaci systému doporučování v rámci sociálních vazeb na této síti. Aplikace nabídne možnost evidence zhlédnutých epizod a jejich hodnocení. Proveďte kvalitativní uživatelský průzkum na jehož základě navrhněte řešení splňující principy User Centered Design. Analyzujte technologie pro implementaci klientské aplikace dostupné i na všech hlavních mobilních platformách. Aplikaci průběžně testujte a integrujte.

Seznam odborné literatury:

Abras, Chadia, Diane Maloney-Krichmar, and Jenny Preece. "User-centered design." Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications 37.4 (2004): 445-456.

Vedoucí: Ing. Pavel Strnad, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

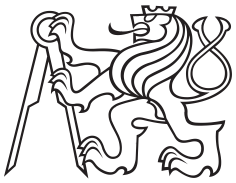
prof. Dr. Michal Pěchouček, MSC.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 2. 2016

Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Klientská část aplikace SerialFreak

Filip Dyrčík

Školitel: Ing. Pavel Strnad, Ph.D.
Leden 2017

Poděkování

Rád bych poděkoval Ing. Pavlu Strnadovi, Ph.D., že podpořil projekt SerialFreak v jeho začátcích a zastřešil ho svým odborným vedením. Děkuji za jeho pomoc a podporu při práci na této diplomové práci. Dále bych rád poděkoval Bc. Jakubu Pýchovi za skvělou spolupráci, která provázela celý projekt až do jeho úspěšného konce. A v neposlední řadě děkuji mé rodině a přátelům, kteří mi byli oporou po celou dobu mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 9.1.2017 .....

Abstrakt

Práce se zabývá návrhem webové aplikace určené pro uživatele sledující televizní seriály. Návrh aplikace je podroben principům designu zaměřeného na uživatele. Dále práce pojednává o analýze webových technologií pro realizaci klientské části aplikace a popisuje jejich využití při samotné implementaci.

Klíčováslova: UCD, Průzkum, Prototypování, SPA, React, Flux, Responzivnost

Školitel: Ing. Pavel Strnad, Ph.D.

Abstract

This paper deals with draft of web application intended for users watching TV series. Design of application is subjected to principals of user centered design. It also deals with analysis of web technologies for the implementation of the client side of the application and describes it´s usages during the actual implementation.

Keywords: UCD, Exploration, Prototyping, SPA, React, Flux, Responsiveness

Title translation: Client side of the application SerialFreak

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V dnešní uspěchané době se seriály těší velké oblibě a stojí za nimi velká fanouškovská základna. Její zástupci mají k dispozici mnoho způsobů, kde a jakým způsobem tento typ multimediálního obsahu konzumovat, avšak o uživatelské přívětivosti těchto řešení by se již dalo debatovat. Právě tento fakt vnuknul mně a mému kolegovi Jakubu Pýchovi myšlenku tvorby aplikace, která by seriálovým nadšencům, jako jsme my, usnadnila život. Rozhodli jsme se společně tomuto tématu věnovat v našich diplomových pracích a společný projekt jsme nazvali SerialFreak.

1.1 Motivace

Hlavní motivací pro naši práci je přinést seriálové komunitě i příležitostným divákům nové inovativní řešení pro jejich potřeby. Takto bychom rádi oslovili co možná nejširší okruh uživatelů a dále toto řešení rozvíjeli a provozovali i po dokončení diplomové práce. Toho nelze docílit bez dostatečné znalosti budoucích uživatelů našeho řešení a všech jejich potřeb. Z tohoto důvodu je část této práce věnována uživatelskému průzkumu, ze kterého vyjdeme při návrhu funkcionality naší budoucí aplikace.

1.2 Cíle

Hlavním cílem této práce je sestavit klient-server aplikaci, která poskytne novou funkcionalitu uživatelům sledujícím seriály na základě jejich potřeb. V procesu vývoje budou zohledněny principy tvorby designu zaměřeného na uživatele. Dále bude kladen důraz na rozšiřitelnost aplikace a její reálný provoz v produkčním prostředí. Práce je rozdělena na dvě dílčí části. První část, které je věnován tento text, se věnuje sběru a analýze dat o uživatelích, sestavení funkčních požadavků, analýze technologií pro realizaci frontendové části aplikace a její samotné implementaci. Druhá část, která je rozebrána v diplomové práci Bc. Jakuba Týchy, se zaobírá backendovou částí aplikace, tedy technologiemi pro vývoj serverové části aplikace, analýzou možností jejího nasazení a samotnou implementací aplikační logiky serverové strany.

Kapitola 2
Uživatelský průzkum

Uživatelský průzkum primárně slouží k získávání, udržování a následnému prezentování informací o uživatelích. Pod informacemi si lze představit jejich potřeby, zvyky, zkušenosti či dovednosti. Toto vše lze zjišťovat nejen o uživatelích existujícího produktu, ale i o potencionálních uživatelích produktu budoucího. Právě tohoto faktu využijeme v této práci a zaměříme se na cílovou skupinu lidí, kterou chceme oslovit naší aplikací. Tento přístup je nazýván v originálním znění User Centered Design, v češtině pak Design zaměřený na uživatele.

2.1 Design zaměřený na uživatele

Samotný termín je definován v publikaci Design zaměřený na člověka: soubor nástrojů [1] následovně:

> Design zaměřený na člověka je proces a soubor technik, které lze využít při vytváření nových řešení pro svět. Tato řešení zahrnují produkty, služby, prostředí, organizace a způsoby interakce.

Pojem zaměřený na člověka znamená, že design vychází z potřeb a chování lidí, které chceme naším řešením oslovit. Prvním krokem je naslouchání a porozumění potřebám potenciálního uživatele. Tyto získané informace následně uplatníme při návrhu našeho řešení. K získání informací o uživatelích existují dva přístupy, které mohou být použity komplementárně, avšak každý z nich poskytuje jiný druh informací. Prvním přístupem je přístup kvantitativní, jehož síla tkví ve velkém vzorku lidí, na kterých provádíme výzkum. Opakem je přístup kvalitativní, který pracuje s malým vzorkem populace, avšak umožňuje zkoumat problémy více do hloubky. Oba přístupy jsou popsány v kapitolách 2.2 a 2.3 na základě přednášek z předmětu Psychologie v HCI [2].

2.2 Kvantitativní přístup

Kvantitativní přístup staví na možnosti oslovit velkou masu lidí, a získat tak velkou sadu dat. Data by měla být kvantifikovatelná nebo statisticky

zpracovatelná, proto samotný způsob, jakým jsou získávána, musí být dobře strukturovaný. Primárně se tento způsob využívá pro testování hypotéz.

2.2.1 Výhody

- Umožňuje testování hypotéz
- Poskytuje exaktní numerická data
- Rychlý sběr dat

2.2.2 Nevýhody

- Opomíjí jevy, které nejsou přímými předměty výzkumu
- Získané informace mohou být příliš abstraktní na jejich aplikaci v reálném prostředí

2.3 Kvalitativní přístup

Kvalitativní přístup využívá menšího vzorku lidí nejen ke zkoumání jevů, ale především k hledání jejich podstaty. Hlavní otázky nejsou kolik nebo jak často, ale proč a jak. Tento přístup tak poskytuje ucelený obrázek o zkoumané problematice. Díky méně strukturovanému vedení výzkumu vzniká veliké množství různorodých dat, která jsou hůře analyzovatelná, ale obsahují mnohem více informací než data kvantitativní. Kvalitativní přístup slouží primárně k vytváření hypotéz.

2.3.1 Výhody

- Poskytuje vhled a detailní popis jevů
- Pomáhá v začátku výzkumu, rozšiřuje pohled na zkoumaný problém
- Zkoumá jevy v jejich přirozeném prostředí
- Umožňuje formulace hypotéz

2.3.2 Nevýhody

- Ne vždy je možné získané informace zobecňovat
- Časově náročné
- Možnost ovlivnění výsledků výzkumu předpojatostí výzkumníka

2.4 Metodika výzkumu

Jelikož hledáme nové inovativní řešení pro fanoušky seriálů, je jasné, že nejprve musíme zjistit, co potřebují a proč. Již otázka „Proč?“ směřuje k tomu, že kvalitativní výzkum je pro náš záměr více než vhodný. Hlavním cílem naší práce je popsat, vysvětlit a interpretovat jednání, které vykazují potenciální uživatelé naší aplikace při sledování seriálů. Dále to mohou být jejich postoje, prožívání a způsoby, jakými seriály sledují. Tyto činnosti jsou uvedeny jako klíčové charakteristiky terénního výzkumu v knize Kvalitativní přístup a metody v psychologickém výzkumu [3], z které budeme čerpat i v dalších částech textu. Terénní výzkum je jedním ze základních typů kvalitativního výzkumu a dle Miovského je jeho „zlatým základem“.

2.4.1 Výběr výzkumného vzorku

Pro realizaci kvalitativního výzkumu je třeba zvolit vhodné participanty (výzkumný vzorek), kteří budou dobře reprezentovat celou cílovou skupinu lidí, které chceme oslovit. To znamená, že v onom vzorku budou zastoupeny všechny typy uživatelů, které můžeme mít, ideálně i ve stejném poměru, v jakém se vyskytují ve společnosti. Pro tuto činnost je v angličtině používán termín sampling (vzorkování). Pokud se toto podaří, je velká šance, že nasbíraná data budou zobecnitelná. Existuje několik metod, které slouží tomuto účelu.

Náhodné vzorkování

- Jednoduché - každý z cílové skupiny lidí může být vybrán se stejnou pravděpodobností
- Systematické - stejné jako jednoduché s tím rozdílem, že jsou lidé z cílové skupiny vybráni dle určitého klíče (např. je sestaven seznam jmen a z něj je vybrán každý desátý člověk)

Nenáhodné vzorkování

- Kvótní výběr - definovány kvóty, které musí výzkumný vzorek naplnit
- Sněhová koule - v první fázi je osloveno několik jedinců, kteří splňují kritéria pro účast ve výzkumu, a ti jsou poté požádáni o kontakty na další potenciální respondenty, kteří splňují také ona kritéria
- Sebevýběr - respondent se sám přihlásí do výzkumu například na základě inzerátu

Metody lze libovolně kombinovat a konkrétní způsob, jakým je získán finální vzorek, je ponechán na úsudku výzkumníka. Pravidlem, které by však mělo být vždy naplněno, je to, že je výzkumník schopný všechna svá rozhodnutí zdůvodnit a obhájit vůči cílům výzkumu.

V našem konkrétním případě by nejvhodnější metodou bylo náhodné vzorkování. Reálně však její použití není možné, protože nemáme možnost vymezit celou cílovou skupinu (nemáme informaci o každém, zda sleduje, či nesleduje seriály). Musíme se tedy zaměřit na metody nenáhodné.

Sněhová koule by jistě mohla fungovat, ale sociální propojení respondentů by mohlo způsobit nedostatečnou rozmanitost získaných dat. Jinými slovy respondenti by mohli být vzájemně ovlivněni svými zvyky a sociálním prostředím, z kterého se znají.

Samotný sebevýběr by opět nemusel poskytnout reprezentativní vzorek, tentokrát z důvodu motivace, kterou respondenti musí mít, aby se dobrovolně do výzkumu přihlásili. Tato motivace by mohla všechny spojovat v určitém ohledu, který nejsme schopni odhalit a který by mohl celý výzkum ovlivnit. Je tedy třeba určit jistá kritéria (kvóty), dle kterých potenciální respondenty vybereme. V případě nesplnění některé z kvót je poté nutno přistoupit k jiné metodě výběru respondentů nebo oslovit cílovou skupinu s možností participování ve výzkumu jiným způsobem. Pro sestavení kvót je třeba definovat cílovou skupinu, na kterou se zaměřujeme.

Cílová skupina

Naše aplikace bude cílit na dospívající a dospělé lidi (16-60) v aktivním věku života (studenti, výdělečně činní), kteří sledují alespoň jeden nebo více seriálových titulů a jsou technicky zdatní ve smyslu užívaní zařízení s přístupem na internet. Velká část cílové skupiny by zároveň měla být součástí nějaké sociální sítě nebo mít alespoň zkušenosti s jejím využíváním.

Screener

Pro výběr respondentů využijeme krátký dotazník, který pokrývá všechny naše požadavky na participanty našeho výzkumu. Tento dotazník se nazývá screener a je součástí přílohy B.1.

Oslovení respondentů

Oslovení respondentů proběhlo na sociální síti Facebook prostřednictvím výše popsaného dotazníku poskytnutého v elektronické podobě jako Google formulář. Konečný výběr participantů výzkumu byl podřízen všem určeným kritériím a zároveň mému úsudku tak, aby se participanti výzkumu neznali.

2.4.2 Metody získávání dat

Pro získání kvalitativních dat lze použít více metod. Mezi ty základní řadí Miovský [3] následující:

- Pozorování
- Metoda moderovaného rozhovoru (Interview)

- Skupinová interview a ohniskové skupiny
- Kvalifikovaný odhad

Výběr metody nebo více metod je stěžejní pro úspěch výzkumu a je podřízen jeho cílům. Na první pohled je jasné, že některé z metod jsou pro případ získání vhledu do potřeb fanoušků seriálů nevhodné.

Pozorování by bylo jen těžko proveditelné (těžko by někdo výzkumníka pustil do svého soukromí, aby byl sledován po celý den/noc po delší dobu) a nemuselo by ani poskytnout dostatečné informace, pokud by nebyla možnost dotázat se na jevy, které výzkumník vypozoroval. Dále časová náročnost této metody by byla nad rámec možností této práce.

Kvalifikovaný odhad založený na expertní znalosti jedince, který se pohybuje v prostředí cílové skupiny, také nelze použít pro náš výzkum, jelikož člověk, který by znal celou naši cílovou skupinu, nejspíše neexistuje. I kdybychom takového člověka našli, pak není doporučeno použít tuto metodu jako primární, jelikož při ní vznikají mnohá zkreslení. Tato zkreslení jsou dána tím, že jsou nám informace expertem pouze zprostředkovány, a velice záleží nejen na jeho expertních znalostech, ale i na jeho schopnostech pozorování, sociální citlivosti a schopnosti nám informace správně předat.

Skupinová interview již představují metodu, která by byla proveditelná, avšak zde hrozí riziko ovlivnění získaných dat skupinovými fenomény, které mohou v průběhu interview vzniknout. Tomu lze částečně zabránit dobrými pozorovacími schopnostmi výzkumníka a jeho dovednostmi v práci se skupinovou dynamikou. Jelikož jsem tuto metodu dosud nepoužil, tak ji vzhledem k mým zkušenostem hodnotím jako příliš náročnou.

Poslední, zatím nezmíněnou metodou, je rozhovor s jedním participantem, tedy interview. Jedná se o jednu z nejpoužívanějších metod kvalitativního výzkumu. Poskytuje možnost rychlé odezvy obou zúčastněných stran, především umožňuje výzkumníkovi klást dodatečné otázky, a zkoumat tak věci do hloubky. Nevýhodou jest fakt, že může být rozhovor velice ovlivněn výzkumníkem. Zde je důležité nejen zvládnutí potřebných sociálních dovedností a citlivosti, ale také kultivace pozorovacích schopností. Pro naši práci se jeví tato metoda jako nejvýhodnější, proto se budeme věnovat primárně jí. Existují tři varianty rozhovoru.

- Nestrukturovaný
- Semi-strukturovaný
- Strukturovaný

Nestrukturovaný rozhovor je metoda připomínající běžný rozhovor o jistém tématu. Dbá především na přirozenost běhu rozhovoru a na rozvíjení témat, ke kterým rozhovor spěje. Není předem dána žádná posloupnost témat ani žádné podokruhy. Je dáno pouze hlavní téma vztahující se k cíli výzkumu. Vedení rozhovoru lze upravovat v závislosti na situaci, nemusíme se všech participantů ptát na stejné otázky ve stejném pořadí ani vést rozhovor stejným způsobem (míra zasahování do rozhovoru výzkumníkem). Opakem jest

rozhovor strukturovaný, kde je vše předem dané a definované. Je předem daný formát rozhovoru, kterého se musí výzkumník držet. Tematické okruhy a otázky jsou v určitém pořadí (může být určený i čas na jejich odpověď). Metoda stojí na hranici mezi dotazníkovými metodami a interview.

Semi-strukturovaný rozhovor je kombinací obou předchozích typů rozhovoru, a dokáže tak řešit mnoho nevýhod jak strukturovaného, tak i nestrukturovaného rozhovoru. Pro jeho realizaci vytváříme určité schéma, které je pro výzkumníka závazné. Většinou se jedná o tematické okruhy a jejich podotázky, které výzkumníka zajímají. V jakém pořadí a jakým způsobem se ovšem výzkumník k odpovědím dostane, je ponecháno na něm a na jeho sociálním cítění. Časový limit na jednotlivé okruhy také není stanoven a nemusí být pro jednotlivé participanty výzkumu stejný. Závisí na rozhodnutí výzkumníka, do jaké míry rozpracovává téma do hloubky, do jakého okamžiku je to užitečné vzhledem k cílům výzkumu.

Vzhledem k cílům výzkumu této práce je nejvhodnější metodou semi-strukturovaný rozhovor poskytující dostatečnou volnost při zkoumání nově objevených jevů v průběhu výzkumu. Zároveň zajistí zodpovězení nejdůležitějších otázek, které se k výzkumu vážou, tedy jakým způsobem lidé sledují seriály a proč. Tyto otázky jsou zapracovány v šabloně popisující průběh rozhovoru (Session Guide). Samotná šablona je přiložena v této práci jako příloha B.2.

Rozhovory budou nahrávány formou audio záznamu pro následnou analýzu. Jako doplňující pomůcka poslouží blok pro zápis doplňující poznámek (fakta, která nezachytí audio záznam: například jak se participant tvářil, smál apod. u určitých témat nebo otázek).

2.4.3 Zpracování a analýza dat

Po absolvování všech rozhovorů vznikne několik hodin jejich audio záznamů. S takovými daty se velmi těžko pracuje a je třeba je nějakým způsobem zpracovat tak, aby je bylo možné efektivně analyzovat.

Prvním a časově nejnáročnějším krokem je plná transkripce audio záznamů do textu. To ale nelze chápat jako pouze mechanickou úlohu. Jak zmiňuje Miovský [3], již zde dochází v závislosti na transkriptorovi ke ztrátám určitého typu informací (síla hlasu, intonace, apod.), kterým lze dobrým transkripčním systémem předejít, nebo je alespoň omezit. Zároveň však dodává, že časová, technická a personální náročnost transkripce musí být adekvátní rozsahu a cílům studie. Jinými slovy, zda nám například podrobný a vysoce sofistikovaný transkripční systém přinese údaje relevantní našemu úkolu a zda jej budeme vůbec schopni technicky realizovat. Na základě tohoto tvrzení a konzultace s Mgr. Jakubem Francem, Ph.D, který se věnuje profesně UX (user experience) a vyučuje na ČVUT FEL předmět Psychologie v HCI [2], jsem se rozhodl transkripci textu nedělat, ale přistoupit rovnou ke druhému kroku, kterým je kódování dat.

Kódování dat lze chápat jako zachycení nejelementárnějších informací z rozhovorů pomocí klíčových slov či slovních spojení a je již součástí samotné

analýzy dat. Takové kódy lze poté sjednocovat do větších významových celků a pracovat s nimi efektivněji. Proces kódování Miovský [3] shrnuje takto: „Proces kódování je de facto procesem identifikace a systematického označování významových celků dle vytvořených kritérií.“. Prakticky se tedy bude jednat o poslech všech audiozáznamů a zachycení všech důležitých či zajímavých informací do kódů, se kterými bude následně provedena analýza. Jelikož nemusíme vždy v daný moment porozumět všem různým významům kódované informace, její pojmenování a zařazení nemusí být konečné a lze se k tomuto procesu v průběhu analýzy opakovaně vracet.

Po úspěšném zakódování všech informací následují další kroky analýzy vedoucí k budování teorie, na jejímž základě bude stanovena hypotéza. Miovský [3] tyto kroky popisuje následovně:

Archivace kódovaných dat - propojení kódů s původní informací (vytvoření databáze kódů s označením jejich zdrojového záznamu a času)

Propojování dat - hledání spojitostí v datech, propojení ve větší celky, vytváření kategorií

Komentování a doplňování dat - rozšíření dat či jejich uvedení do kontextu (např. poznámky)

Vyvozování závěrů a verifikace - interpretace údajů a ověřování její platnosti, identifikace podmínek, za nichž je vytvořená interpretace údajů validní, a naopak podmínek, za nichž přestává být (nebo její části) platná.

Budování teorie - vytvoření a vysvětlení interpretačního rámce, který popisuje a vysvětluje naše nálezy

Grafické mapování - znázornění teorie či samotných nálezů v grafické podobě (diagramy, schéma, modely, atp.)

Pro fázi propojování dat, komentování a doplňování pak popisuje několik dílčích postupů (metod), které tyto fáze mohou doprovázet. Metody, které považuji za přínosné pro můj výzkum, jsou následující.

Metoda vytváření trsů slouží k seskupení a konceptualizaci informací/-kódů do skupin dle např. určitých jevů, časových údajů, údajů o poloze apod. Tímto vznikají určité celky/kategorie, které poskytují obecnější informace, a vzniká tak určitá hierarchie mezi základními kódy a kategoriemi.

Metoda prostého výčtu umožňuje kvantifikovat některá kvalitativní data. Jde například o četnost výskytu určitého jevu či poměru jeho výskytu k jevu jinému. Může se také jednat o vyjádření intenzity prožitku (jak moc) apod., který nám dává také platnou informaci o kvalitě.

Metoda zachycení vzorů seskupuje opakující se jevy do vzorců, které tento jev popisují. Vznikají tak obecnější principy, které tento jev vystihují. Ztráta bohatosti a jedinečnosti pozorovaných jevů je nahrazována obecnější kategorií

či vzorem.

Metoda vyhledávání a vyznačování vztahů slouží buď k vyznačení vztahů, na které jsme upozorněni přímo účastníky výzkumu explicitně a kterým se následně snažíme přiřadit přijatelné vysvětlení, anebo k vyhledání implicitních vztahů na základě vnitřních, nebo vnějších souvislostí. Vnějšími souvislostmi můžeme například chápat náš fyzický vzhled a reakce kolemjdoucích na nás. Vnitřními pak například naše prožitky v závislosti na okolí, hudbě apod.

2.5 Analýza dat

Po důkladném výběru metod pro analýzu dat je možné přistoupit k její samotné realizaci, kterou popíši v následujících podkapitolách.

2.5.1 Kódování dat

Proces kódování dat jsem realizoval takovým způsobem, aby při něm zároveň vznikl i archiv kódovaných dat, viz Archivace kódovaných dat v předchozí sekci 2.4.3. Tedy při poslechu rozhovorů jsem zaznamenával důležité informace a zároveň k nim přiřazoval čas, kdy byly zmíněny. Po takto hrubém zpracování záznamů jsem přistoupil k samotnému zakódování dat a přiřazení dimenzí jevům, u kterých to bylo vhodné. Výsledek této práce přikládám v příloze C.

2.5.2 Metoda vytváření trsů

Po zakódování dat jsem vyznačil hlavní kategorie, kterých se jevy týkaly a to:

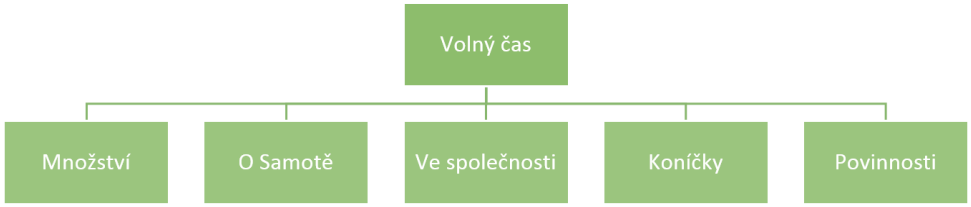
- Volný čas (Zeleně)
- Využívání internetu (Modře)
- Sledování seriálů (Žlutě)

Tyto kategorie nám poskytnou lepší orientaci v kódovaných datech a zároveň umožňují zkrácení kódovaných slovních spojení, jelikož samy o sobě vyjadřují, čeho se kódy týkají. Například kód "Sledování seriálů"sám o sobě nese málo informací o svém významu, ale kategorie mu přiřadí správný význam.

- Sledování seriálů (Zeleně) = Participant sleduje seriály ve svém volném čase
- Sledování seriálů (Modře) = Participant využívá internet ke sledování seriálů

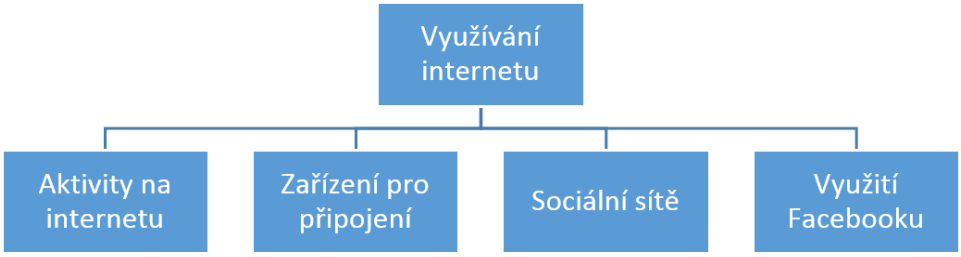
Po základní kategorizaci kódů jsem přistoupil k vytváření dalších podkategorií, které shlukují data sobě blízká do menších bloků, jenž jsou více srozumitelné a dávají čtenáři možnost rychlejší orientace v nasbíraných datech. Celé zpracování metody vytváření trsů je také součástí přílohy C.

Volný čas



Obrázek 2.1: Kategorizace volného času.

Využívání internetu



Obrázek 2.2: Kategorizace využívání internetu.

Sledování seriálů

Obrázek 2.3: Kategorizace sledování seriálů

2.5.3 Metoda prostého výčtu

Při zařazování kódů jednotlivých participantů ke kategoriím vznikalo poměrně velké množství duplikovaných dat. Jinými slovy některé jevy se objevily u více participantů. Následující tabulka vyzdvihuje nejzajímavější jevy a jejich četnosti v nasbíraných datech.

Využívání internetu		Sledování seriálů	
Jev	Četnost	Jev	Četnost
Facebookování	6	Sledování seriálu večer/před spaním	6
Využívání pouze facebooku	6	Sledování epizod seriálu popořadě	6
Organizování přes Facebook	5	Sledování seriálu pro oddech	5
Komunikování přes Facebook	5	Sledování různých žánrů	5
Emailování	3	Sledování nepříběhových seriálů nahodile	5
Sledování/stahování seriálů	3	Sledování seriálů online	5
Poslouchání hudby	3	Sledování epizod seriálu nahodile	5
Čtení zpráv	3	Sledování seriálů o samotě	5
		Sledování na základě prvního dojmu	5
		Sledování na základě doporučení	5
		Sledování příběhových seriálů postupně	4
		Sledování seriálů přes den ve volných chvílích	4
		Sledování seriálů v televizi	4
		Zvolení epizody ke sledování dle názvu epizody	4
		Uchovávání oblíbených seriálů	4
		Hledání informace o datu vydání nové epizody	4
		Sledování seriálů pro vyplnění volného času	3
		Sledování seriálů u jídla	3
		Preferování sledování ihned na úkor kvality	3
		Vybírání epizody ke sledování dle oblíbenosti	3
		Sledování příběhových seriálů postupně	3
		Sledování online s problémy se sekáním	3
		Vyhledávání informace o obsahu seriálu	3
		Vyhledávání zajímavostí o seriálu	3
		Stahování seriálů do PC, když není online	2
		Sledování online, když nenalezne epizodu ke stažení	2
		Mazání seriálů po zhlédnutí	2
		Vybírání seriálu ke sledování dle hodnocení	2
		Sledování seriálů pro zamyšlení	1
		Sledování seriálů v páru	1
		Otravování reklamou	1

Obrázek 2.4: Nejčetnější výskyty stejných kódů

2.5.4 Metoda zachycení vzorů

Při aplikaci metody vyváření trsů a metody prostého výčtu byly objeveny jevy, které se objevují společně, a tvoří tak určité vzory chování participantů.

Využívání internetu

Všichni participanti zmínili sociální síť Facebook jako jedno z využití internetu a označili ji jako jedinou sociální síť, kterou používají. Facebook používají denně především pro komunikaci a organizaci událostí s přáteli.

Sledování seriálů

Participanti si chtějí u seriálu především odpočinout, a vyplnit tak krátké chvíle volného času, které mají k dispozici. Nejčastěji se jedná o čas během jídla nebo před spaním.

Participanti nemají vyhraněný vkus na žánr seriálu, avšak častěji vyhledávají krátké komediální seriály, které sledují v počítači nebo v televizi.

Participanti rozlišují mezi seriály s dějovou linkou přesahující jednu epizodu a mezi seriály, jejichž epizody neobsahují návaznost na epizody minulé. Toto rozlišení způsobuje různé způsoby sledování dějových a nedějových seriálů ve smyslu pořadí sledování epizod. Konkrétně dějové seriály participanti sledují od první epizody popořadě a nedějové sledují nahodile dle oblíbenosti epizod.

Participanti, kteří preferují stažené epizody do počítače, si více potrpí na kvalitu videa. Naproti tomu participanti, kteří kvalitu videa nevyžadují, preferují sledování seriálů online.

Participanti stahující seriály do počítače označují stahování titulků za problematické. Samotné stažení titulků pro ně představuje práci navíc.

Participanti, kteří preferují stahování před sledováním online, zmiňují jako hlavní problémy špatnou kvalitu videa a zasekávání streamu.

Participanti si vyhledávají nejčastěji popis seriálu na CSFD nebo na seriálových portálech edna.cz a serialzone.cz. Jde jim především o ujištění, zda se jim seriál bude líbit.

Pokud participanti nevědí, kde hledat seriály nebo informace o nich, tak sahají po vyhledávači Google a čerpají informace ze zdrojů, které jim vyhledavač poskytne.

Vysílací časy zahraničních seriálů hledají participanti na seriálových serverech, ale vysílací časy českých seriálů hledají v televizním programu, nejčastěji na

portálu seznam.cz.

Participanty nejčastěji přivedlo ke sledování určitého titulu seriálu doporučení od kamaráda/známého. Dále kladou veliký důraz na první dojem ze seriálu, když s ním přijdou do styku.

2.5.5 Metoda vyhledávání a vyznačování vztahů

Explicitní

Lidé využívají Facebook především z důvodu jeho rozšíření a uživatelské základny bez ohledu na jeho funkce, tím pádem nemají potřebu vyhledávat jiné alternativy.

Míra sledování seriálů je závislá na množství volného času, pro který není jiné lepší využití.

Výběr žánru seriálu je podřízen aktuální náladě diváka. Při nedostatku energie volí spíše sitkomy a komedie pro odpočinek. Naopak pokud chtějí větší prožitek a zapojení mysli, pak volí seriály s komplexnější zápletkou (krimi, drama, apod.).

Implicitní

Pro vyznačení vztahů vyhledáváme v kódovaných datech související data, která následně interpretujeme v jejich souvislostech.

Sledování krátkých seriálů - Potřeba sledovat okamžitě - Sledování 1-2 epizod - Sledování seriálů pro vyplnění kratších úseků volného času

Z výše uvedených kódů vyplývá, že lidé inklinují ke krátkým seriálům především proto, aby jimi vyplnili krátké úseky svého volného času. Kratší délka epizody jim umožňuje časovou variabilitu, proto mohou zhlédnout buď jednu, nebo i více epizod, a vyplnit tak přesný časový rámec, který mají k dispozici. Jelikož je časový úsek krátký, tak se nechtějí zdržovat stahováním epizod do počítače a chtějí začít sledovat ihned.

Sledování seriálů pro vyplnění kratších úseků volného času - Potřeba sledovat okamžitě - Preferování sledování online - Sledování seriálů online nahodile - Preferování sledování ihned na úkor kvality

Jelikož mají lidé málo volného času, který chtějí využít pro sledování seriálů, tak hledají možnost započetí sledování ihned poté, co se rozhodnout, že si chtějí něco pustit. Tuto možnost spatřují ve sledování seriálů online, kde pro ně není důležité ani jaký konkrétní díl onoho seriálu si pustí, tak ani kvalita,

ve které ho sledují. Nejdůležitějším aspektem je, že sledují okamžitě.

Sledování seriálů pro vyplnění volného času - Sledování seriálů přes den ve volných chvílích - Sledování seriálů u jídla - Sledování seriálů večer - Sledování seriálů před spaním

Tyto kódy naznačují, že volný čas lidé nacházejí jak přes den, tak i ve večerních hodinách. Sledování přes den je doplněno informací o sledování u jídla, čili čas jídla je vnímán jako forma volného času, který lze využít nejen za účelem najedení. Podobně sledování seriálů večer je syceno sledováním před spaním, takže i zde je krátký časový úsek před spánkem vnímán jako volný čas vhodný pro sledování seriálů.

Sledování epizod seriálu nahodile - Sledování nepříběhových seriálů nahodile - Sledování již viděných seriálů nahodile - Vybírání epizody ke sledování dle oblíbenosti - Zvolení epizody ke sledování dle názvu epizody - Vyhledání epizody dle prvních pár sekund videa

Pokud se divák rozhodne k výběru náhodné epizody, pak ho k tomu především vede fakt, že seriál již viděl a dějovou linku zná nebo se jedná o seriál bez příběhové linie, kde má každá epizoda svůj vlastní děj. Konkrétní epizodu poté volí na základě oblíbenosti z minula a vyhledává ji dle jejího názvu nebo dle prvních pár sekund videa.

Sledování zahraničních seriálů v počítači - Sledování českých seriálů v televizi - Vyhledání informace o vysílacím čase v programu - Sledování na základě reklamy - Sledování na základě doporučení - Sledování epizody na základě rozšíření ve společnosti

Diváci sledují v PC převážně zahraniční seriály, a české sledují v televizi v jejich vysílacích časech, které si zjišťují na internetu v televizním programu. Sledování seriálu započínají nejen na základě doporučení či rozšíření ve společnosti, ale i na základě reklamy.

2.6 Formulace hypotéz

Z výzkumu jasně plyne neotřesitelné postavení facebooku na poli sociálních sítí. Lidé ho vnímají jako prostředek komunikace a představuje pro ně důležitý zdroj kontaktů. Připojují se k němu převážně ze svých osobních počítačů, jelikož velká obrazovka pro ně znamená komfort práce. Mobilní zařízení používají, pouze pokud jsou mimo dosah PC.

Velkým aspektem, který ovlivňuje chování a rozhodování lidí, se ukazuje být volný čas. Lidé pociťují nedostatek volného času, který by mohli věnovat čistě osobním aktivitám a odpočinku. Proto vyhledávají i velice krátké úseky času, které se snaží vyplnit relaxační aktivitou, kterou se jeví být sledování

seriálů. Tyto úseky času jsou velice často chvíle při konzumaci jídla nebo večerní ukládání ke spánku.

Seriály nejsou rozlišovány dle svých konkrétních žánrů, ale jsou vnímány v obecnějších kategoriích, které udávají kritéria potřeby diváka.

Dle žánru

- Odpočinkové (sitcomy, komedie, ..)
- K zamyšlení (krimi, drama, ..)

Dle obsahu

- Dějové (s příběhem probíhajícím celou sérii/seriál)
- Epizodní (epizody svým dějem nenavazují na předchozí)

Dle délky

- Krátké (do 30 minut)
- Dlouhé (nad 30 minut)

Aktuální potřeba diváka pak směřuje k selekci seriálu dle daných kritérií. Obsahová kategorie pak ovlivňuje i strategii výběru epizody. Dějové seriály jsou sledovány popořadě od první epizody po poslední. Naproti tomu u epizodních seriálů volí divák libovolnou epizodu, nejčastěji dle osobní oblíbenosti, pokud již seriál zná. Mezi epizodami se nejčastěji orientuje dle názvu epizody nebo dle prvních pár vteřin videa.

Nejčastěji diváci sledují seriály na svých osobních počítačích nebo v televizi. Osobní počítače jim především zpřístupňují zahraniční seriály, které se v České republice nevysílají. V televizi pak sledují převážně české seriály.

Již zmíněný nedostatek času také způsobuje potřebu sledovat seriál ihned po rozhodnutí se ke sledování. Tento fakt směřuje diváky ke sledování seriálů online, kde jsou k dispozici, nebo ke stažení seriálu do PC předem. Zde se rozdělují diváci na dvě skupiny. První z nich ignorují neduhy online sledování (špatná kvalita videa, zasekávání videa, reklamy) a konzumují seriály online bez jejich uchovávání. Druhá skupina přikládá kvalitě videa větší význam a stahuje si seriály do PC. Ke sledování online přistupují, jen pokud není určitá epizoda ke stažení dostupná.

Před započetím sledování nového seriálu diváci pociťují potřebu zjistit si obecný obsah seriálu a ujistit se, že je jeho náplň opravdu zajímá a že je bude bavit. Obsah nejčastěji vyhledávají na serverech CSFD.cz, edna.cz, serialzone.cz. Kromě samotného obsahu vyhledávají i recenze a hodnocení. Pokud nenaleznou dostatek informací zde, uchylují se k hledání přes vyhledávač Google.

Sledování zahraničních seriálů v originálním znění doprovází nutnost řešení českých titulků. To platí především v případě videí stažených do PC, které nemají titulky vložené. Stahování titulků a výběr správné stopy je tak další práce navíc, která je hodnocena jako nepříjemná.

Další vyhledávanou informací o seriálu je datum vysílání/vydání nové epizody u aktuálně vysílaného seriálu. V případě zahraničních seriálů se diváci obrací na seriálové servery (edna.cz, serialzone.cz). Pokud se jedná o české seriály, pak tuto informaci hledají v televizním programu, nejčastěji na seznam.cz.

Nejčastějším důvodem pro započetí sledování nového seriálu je doporučení od přátel či známých. O přijetí, nebo nepřijetí seriálu ke sledování pak rozhoduje první dojem z jedné nebo několika prvních epizod.

2.6.1 Shrnutí

Navzdory nedostatku času lidé pro sledování seriálů vynakládají nemalou část svého časového fondu i na jiné aktivity než jen na samotné sledování. Mezi tyto aktivity patří především:

- Výběr nového seriálu ke sledování (doporučení -> hledání obsahu / hodnocení -> hledání epizody ke sledování / stažení)
- Výběr seriálu a epizody pro sledování v určitou chvíli (vybírání seriálu pro uspokojení aktuálních potřeb -> vybírání epizody dle názvu / oblíbenosti / chronologie -> hledání epizody ke sledování / stažení)
- Čekání nové epizody sledovaného seriálu (hledání data vysílání/vydání nové epizody -> hledání epizody ke sledování / stažení)
- Stažení epizody do PC (hledání epizody ke stažení -> hledání správné stopy titulků)

Pro dokončení podaktivit často používají různé prostředky a servery a celý proces je tím ještě složitější a zdlouhavější. Hlavní potřebou lidí je úspora času při vykonávání nutných akcí vedoucích ke sledování seriálů.

2.6.2 Návrh řešení

Nabídneme uživateli službu v podobě webové aplikace, která usnadní a především zrychlí aktivity prováděné při sledování seriálů. S využitím sociální sítě Facebook lze realizovat systém pro doporučování seriálů ke sledování. Systém bude fungovat jak na bázi osobního doporučení v rámci přátel, tak nabídne i možnost doporučení systémem na základě osobních preferencí a hodnocení.

Webová aplikace nabídne agregovaná data o seriálu a epizodách, aby uživatel nemusel vyhledávat dodatečné informace na jiných serverech. Aplikace bude evidovat aktuálně sledované seriály uživatelem a nabídne jednoduchý a efektivní způsob výběru epizody ke sledování na základě aktuálních preferencí uživatele (viz kategorie žánr, obsah a délka v sekci 2.6).

Pro dlouhodobý komfort při sledování seriálů aplikace nabídne osobní kalendář s vysílacími časy aktuálně vysílaných seriálů, které uživatel sleduje.

Pro posílení komunity a vytvoření velké uživatelské základny využijeme sociální síť Facebook následovně:

- Doporučení seriálu ke sledování na Facebooku
- Sledování aktivit přátel v aplikaci (započetí sledování seriálu, sledování epizody, hodnocení seriálů a epizod apod.)
- Možnost sdílení vlastních aktivit na zeď Facebooku

Dále bude možnost nastavení facebookových upozornění na vybrané události. Například na vydání nové epizody sledovaného seriálu nebo na doručení doporučení seriálu ke sledování.

2.7 Sestavení funkčních požadavků na aplikaci

Z návrhu řešení 2.6.2 již vyplývají funkční požadavky na aplikaci. Každému funkčnímu požadavku bude přiřazeno pořadové číslo (ID) a priorita. Na základě priority budou funkce později implementovány.

2.7.1 Funkční požadavky

Priorita požadavků:

1. Nízká
2. Vysoká
3. Kritická

1. Přihlášení přes Facebook (priorita: 3)
K aplikaci se uživatelé budou hlásit striktně pomocí Facebook rozhraní. Po úspěšném přihlášení k Facebooku bude uživatel autorizován k přístupu do aplikace.

2. Agregace informací o seriálu (priorita: 2)
Aplikace bude sjednocovat data o seriálu z více zdrojů, a nabídne tak uživateli ucelené informace a hodnocení seriálu z více portálů.

3. Přidání seriálu ke sledování (priorita: 3)
Uživatel bude schopen zařadit seriál mezi sledované, čímž se mu zpřístupní rozšířené funkce aplikace nad tímto seriálem popsané dále.

4. Kategorizace sledovaného seriálu (priorita: 3)
Při zařazení seriálu do sledování bude uživatel schopen zvolit kategorie seriálu (dějový x epizodní, odpočinkový x k zamyšlení) pro pozdější filtrování dle typu seriálu.

5. **Doporučení seriálu na základě aktivity přátel (priorita: 2)**
   Uživateli aplikace zprostředkuje doporučení seriálů, které mají jeho přátelé ve sledování, a zároveň zohlední i další aspekty pro co nejlepší shodu se zájmem uživatele.

6. **Osobní doporučení seriálu od přítele (priorita: 2)**
   Aplikace poskytne možnost uživateli osobně doporučit jím sledovaný seriál svému příteli. Ten bude na tuto událost v aplikaci upozorněn a bude mu nabídnuta možnost seriál sledovat.

7. **Evidence zhlédnutých epizod sledovaných seriálů (priorita: 3)**
   Uživatel bude schopný v aplikaci spravovat stav ke každé epizodě sledovaného seriálu ve smyslu, zda ji viděl, či nikoliv.

8. **Hodnocení zhlédnutých epizod (priorita: 2)**
   Uživatel bude moci zhodnotit každou epizodu sledovaného seriálu, zda se mu líbila, či nikoliv. Akce bude proveditelná již při zhlédnutí epizody. Hodnocení bude možné měnit a mazat.

9. **Doporučení epizody ke zhlédnutí (priorita: 3)**
   Aplikace doporučí uživateli pro každý sledovaný seriál jednu epizodu ke zhlédnutí. Epizoda bude zvolena na základě kategorie seriálu (epizodní x dějový). V případě dějového seriálu bude uživateli nabídnuta první nezhlédnutá epizoda. V případě epizodního seriálu aplikace doporučí epizodu s ohledem na hodnocení uživatele, hodnocení přátel uživatele, tak i hodnocení z externích seriálových portálů.

10. **Manuální výběr epizody ke zhlédnutí (priorita: 3)**
    Uživatel bude schopný vyhledat jakoukoli epizodu seriálu. Aplikace nabídne kompletní seznam sezón a epizod, v kterém se bude moci uživatel pohybovat.

11. **Osobní kalendář s daty vysílání sledovaných seriálů (priorita: 1)**
    Aplikace zprostředkuje uživateli data o vysílacích časech nových epizod seriálů, které má ve sledování.

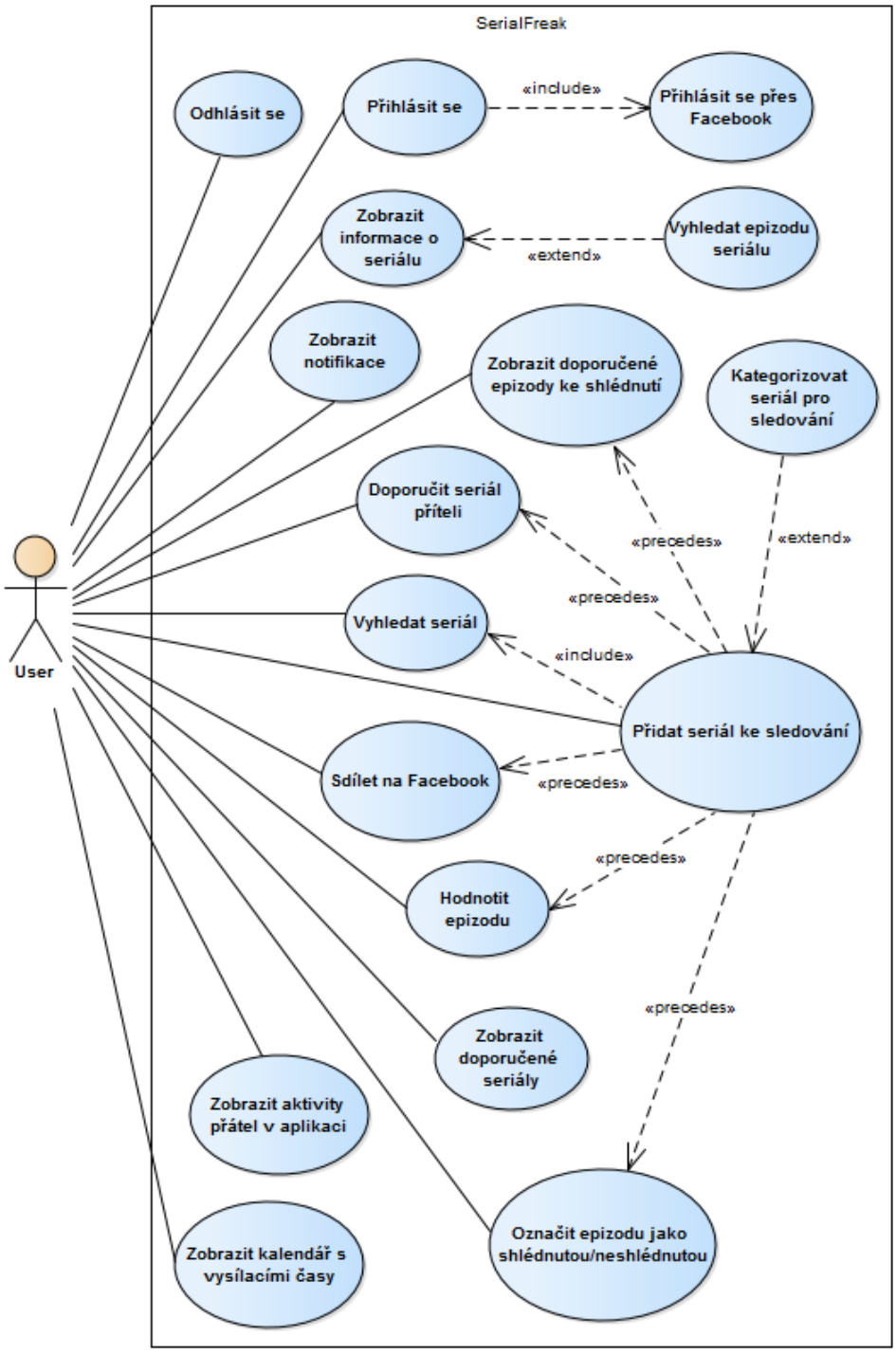
12. **Přehled aktivit přátel v aplikaci (priorita: 2)**
    Uživatel bude schopen sledovat aktivity svých přátel v aplikaci. Základní zaznamenávané aktivity budou: přidání seriálu ke sledování, zhlédnutí epizody seriálu, hodnocení epizody seriálu

13. **Sdílení vlastních aktivit na zeď Facebooku (priorita: 2)**
    Uživatel bude schopen sdílet své aktivity na svém facebookovém profilu. Aktivitami pro sdílení budou: zhlédnutí epizody, přidání seriálu k odběru. Dále bude moci sdílet odkaz na svůj oblíbený seriál jako celek.

14. **Notifikace na zvolené události na Facebooku (priorita: 3)**
    Uživatel bude notifikován o důležitých událostech v aplikaci v notifikačním centru Facebooku. Takovou událostí může být například vydání nové epizody sledovaného seriálu.

2.7.2 Případy užití

Pro větší přehlednost, jaké možnosti aplikace uživateli nabídne, jsem sestavil následující diagram případů užití.



Obrázek 2.5: Diagram případů užití

2.7.3 Storyboards

Z případů užití je jasné, jaké funkce může uživatel v aplikaci použít. Avšak nemusí být zřejmé, v jakých situacích by jich měl využít. Abych nastínil, jakým způsobem by aplikace měla být pro uživatele přínosná, vytvořil jsem dva takzvané storyboardy. Jedná se o sekvenci kreseb ilustrujících sekvenci událostí, které doprovází použití aplikace. Mnou vybrané případy zdaleka nepokrývají všechny případy užití aplikace, ale popisují dva pro aplikaci stěžejní.

Obrázek 2.6: Rychlé sledování

První ze storyboardů vykresluje situaci, kdy se uživatel rozhodne pro sledování nějakého seriálu. Uživatel má tušení, jaký typ seriálu by si chtěl pustit, ale neví, jaký konkrétně, natož jakou jeho epizodu. Díky aplikaci SerialFreak snadno a rychle vyfiltruje mezi svými seriály ty, které jsou pro onu chvíli vhodné, a z nabídnutých již není problém zvolit jeden konkrétní. Navíc se nejedná pouze o seriály, ale přímo o epizody, takže po výběru seriálu má uživatel ihned jasno, na jakou epizodu se má podívat, a může začít sledovat.

Obrázek 2.7: Sledování aktivit přátel

Druhá ilustrace ukazuje, jakým způsobem by uživatelé mohli objevovat pro ně nové a neznámé seriály, které by jinak unikly jejich pozornosti. Z uživatelského průzkumu vyplynulo, že nové seriály uživatelé objevují náhodně právě díky svým přátelům.

Kapitola 3

Návrh uživatelského prostředí

Po stanovení funkčních požadavků aplikace lze přistoupit k návrhu uživatelského rozhraní. To však nelze chápat jen jako návrh grafického vzhledu a rozmístění ovládacích prvků na webové stránce. Aby byl uživatelský zážitek z používání aplikace co nejlepší, je nutné dbát na základní vlastnosti dobrého UXD (User eXperience Design), zejména na použitelnost. Norma ISO/DIS 9241-11.2 [4] použitelnost definuje jako míru, do které je aplikace použitelná pro konkrétní uživatele tak, aby efektivně, účinně a uspokojivě dosáhli svých cílů v daném kontextu užití. Dílčí vlastnosti takové aplikace jsou:

Naučitelnost
: Zvládnutí základních úkonů při první interakci s aplikací.

Efektivita
: Rychlost práce s aplikací při vykonávání běžných úkolů.

Spokojenost
: Jak příjemné je uživateli s aplikací pracovat.

Chybovost
: Kolik chyb uživatelé při práci s aplikací dělají a jak náročné je jejich řešení.

Zapamatovatelnost
: Do jaké míry je pro uživatele těžké vrátit se k efektivní práci s aplikací po delší době jejího nepoužívání.

Splnění výše uvedených bodů vyžaduje značné úsilí. Pro dosažení co nejlepšího výsledku s ohledem na časové možnosti a rozsah této práce jsem zvolil následující postup.

1. Skicování/Mockupování
2. Tvorba prototypu
3. Testování prototypu s uživateli
4. Úprava rozhraní na základě nálezů při testování

3.1 Metodika návrhu

Pro utvoření základní představy, jak by měla aplikace vypadat a jaké by mělo být základní workflow [5] v aplikaci, jsem využil metody Design studio, prezentované v rámci předmětu Návrh uživatelského rozhraní [6]. Metoda spočívá ve skicování, prezentování a následné vzájemné kritice návrhů mezi členy týmu. Celý proces probíhá v několika iteracích, kdy jsou postupně návrhy konkretizovány a sjednocovány do finálního návrhu.

Iterace.

1. Indviduální návrh každého člena týmu a jeho prezentace
2. Připomínkování předložených návrhů
3. Individuální úprava vlastních návrhů (konkretizace detailů) na základě připomínek a jejich prezentace
4. Připomínkování upravených návrhů
5. Společný návrh konečného řešení

Pozn.: Iterace 3 a 4 může být několikrát opakována.

Pracovní skupinu design studia by měli tvořit alespoň 4 lidé, ale jelikož SerialFreak vyvíjíme s kolegou Jakubem Pýchou pouze ve dvou, sáhli jsme po zjednodušené variantě o dvou lidech. V první fázi jsme na základě funkčních požadavků stanovili základní čtyři obrazovky aplikace, které je nutno navrhnout. A to domovskou stránku, stránku se seznamem odebíraných seriálů, stránku vyhledávání seriálů a stránku detailu seriálu. Na tyto definované stránky jsme následně aplikovali jednotlivé iterace Design studia. Výstupem procesu byl soubor náčrtků rozložení jednotlivých stránek, které jsem využil pro tvorbu prototypu.

Prototypy se dělí do dvou kategorií. Low fidelity (Lo-Fi) a High Fidelity (Hi-Fi). Jak již název napovídá, rozdíl mezi prototypy tvoří míra věrnosti prototypu vůči konečnému produktu. Lo-Fi prototypování se vyznačuje rychlým a jednoduchým přechodem mezi designovým konceptem k hmotnému a testovatelnému prototypu. Cílem je ukázat základní charakteristiky navrženého designu a odhalit jeho případné chyby již na začátku vývoje. Hi-Fi prototyp naopak slouží k prezentaci vzhledu finálního produktu, je již postaven na technologiích finálního produktu, i když nemusí být implementována všechna jeho aplikační logika.

Rozhodl jsem se pro tvorbu Lo-Fi prototypu. Díky jeho nízké náročnosti na tvorbu s ním lze v krátké době oslovit potenciální uživatele naší aplikace a otestovat s nimi, zda naše návrhy neobsahují kritické chyby, které by

uživatelům znesnadnily práci s budoucí aplikací. Získaná data z testování lze následně využít pro úpravu a zlepšení konečného realizovaného designu.

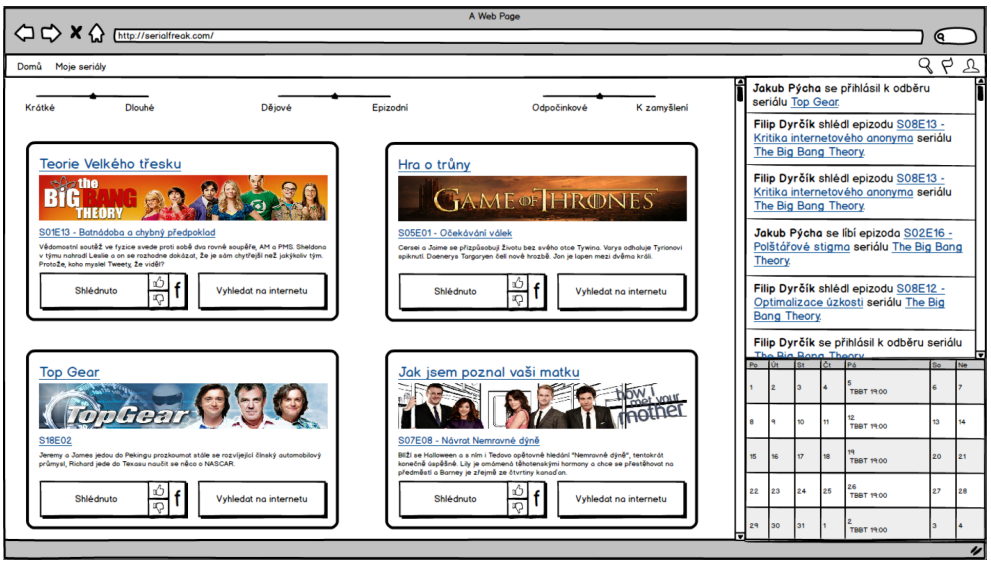
3.2 Low Fidelity Prototyp

Pro realizaci Lo-Fi prototypu jsem zvolil program Balsamiq [7]. Jedná se o jednoduchý editor, v kterém lze vytvářet návrhy uživatelského prostředí a exportovat je ve formátu PDF. Poskytuje možnost vytvoření aktivních prvků rozhraní, díky kterým lze přecházet mezi jednotlivými obrazovkami (stránkami) aplikace. S touto možností lze sestavit testovací scénáře a převést je na sekvenci po sobě jdoucích akcí nad uživatelským rozhraním. Prvním krokem však bylo vytvoření statických verzí všech hlavních obrazovek aplikace.

3.2.1 Domovská obrazovka

Jedná se o úvodní obrazovku zobrazenou uživateli po jeho přihlášení. Jejím hlavním obsahem jsou tři sekce, a to Epizody ke shlédnutí, Události a Kalendář. Sekce Události a Kalendář jsou společné pro všechny hlavní obrazovky. Při přechodech mezi obrazovkami bude měněna pouze levá část stránky.

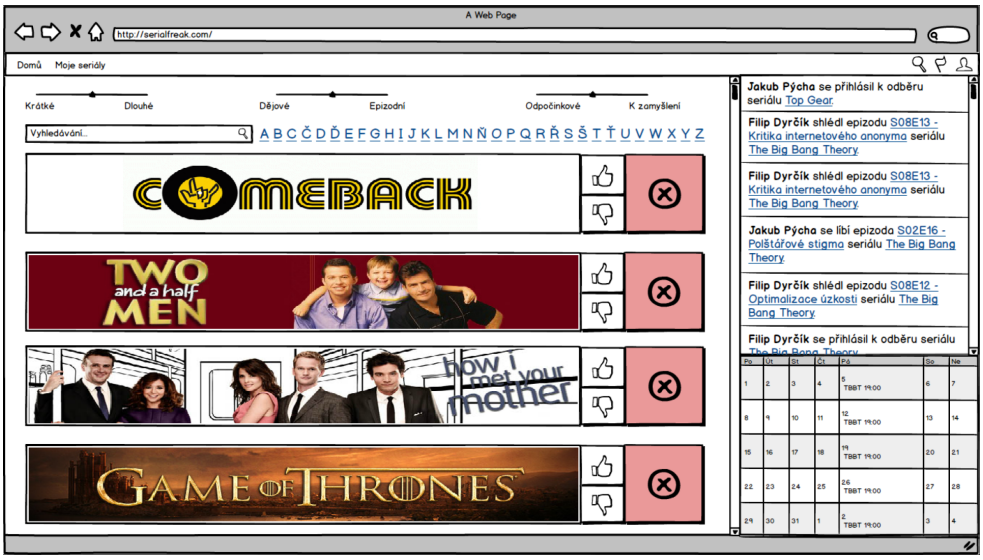
Sekce epizody ke zhlédnutí obsahuje dláždice s epizodami seriálů. Každý seriál, který má uživatel v odběru, reprezentuje jedna dláždice s epizodou, která je uživateli doporučena ke zhlédnutí. Doporučení vzniká na základě žánrové kategorizace seriálu. V případě dějového seriálu je uživateli doporučena první nezhlédnutá epizoda v chronologickém pořadí. V případě epizodního je nabídnuta epizoda na základě hodnocení uživatele, hodnocení přátel a hodnocení na jiných seriálových portálech. Každou epizodu lze vyhledat na internetu a následně označit za zhlédnutou. Současně se zhlédnutím epizody je možné ji i ohodnotit (líbí se X nelíbí se) a sdílet tento fakt na sociální síti Facebook.



Obrázek 3.1: Domovská obrazovka

3.2.2 Moje seriály

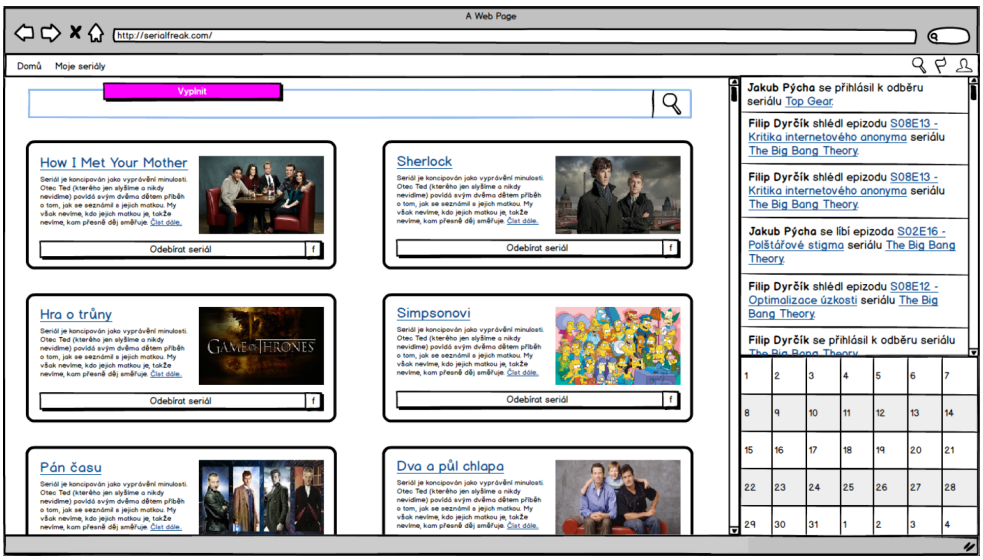
Seznam sledovaných seriálů uživatel nalezne v sekci Moje seriály. Obrazovku tvoří bannery seriálů, které lze filtrovat pomocí kategorií či počátečního písmena. V seriálech lze i vyhledávat pomocí textového pole.



Obrázek 3.2: Moje seriály

3.2.3 Vyhledávání

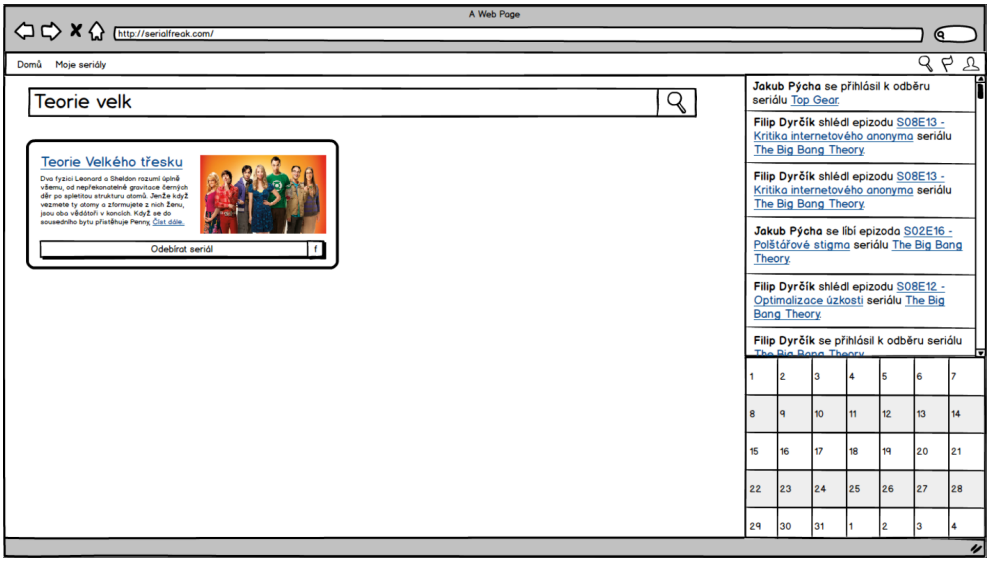
Zde může uživatel vyhledávat seriály a přidávat si je k odběru. Před prvním vyhledávacím dotazem jsou uživateli nabídnuty seriály doporučené. Každý seriál je vykreslen v podobě dláždice.



Obrázek 3.3: Vyhledávání

Vyhledaný seriál

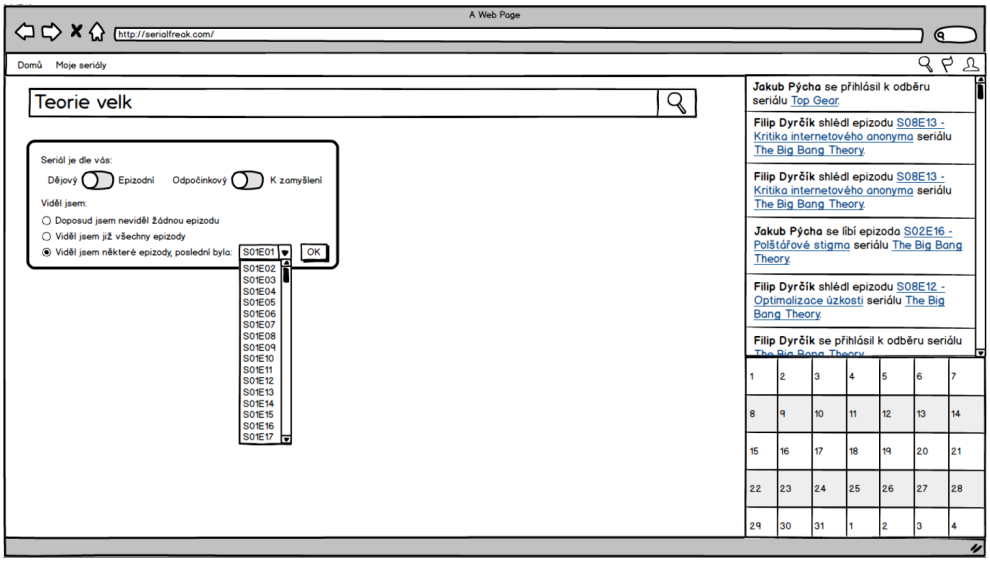
Po zadání vyhledávacího řetězce jsou uživateli vylistovány pouze seriály, které ho obsahují ve svém názvu.



Obrázek 3.4: Po vyhledání seriálu

Odběr seriálu

Po kliknutí na tlačítko odebírat je dláždice překreslena na formulář, kde uživatel určí osobní kategorizaci seriálu a může předem zadat informaci, které epizody seriálu již viděl.

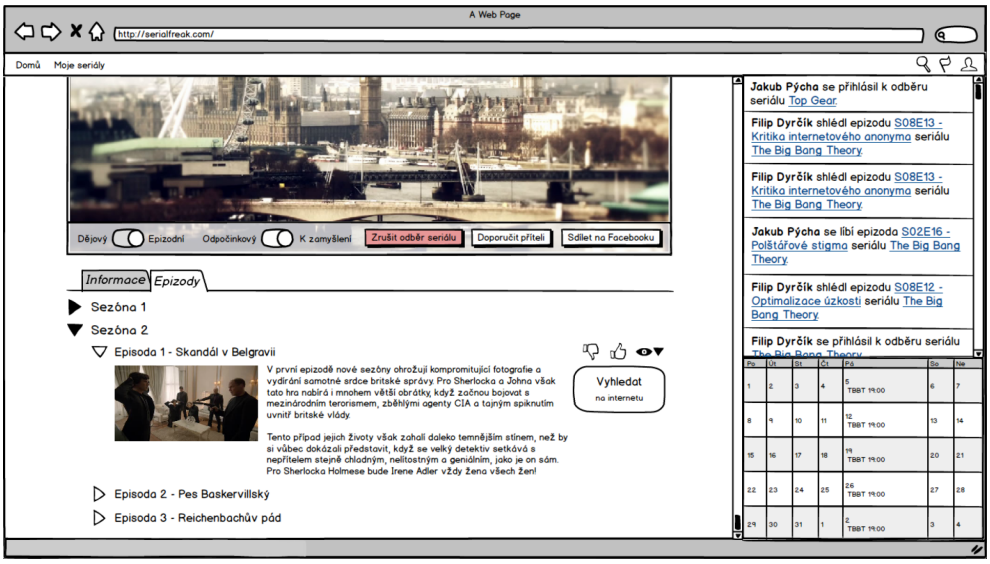


Obrázek 3.5: Po kliknutí na odběr seriálu

3.2.4 Seriál

Stránka seriálu obsahuje základní informace o daném seriálu a lze listovat v jeho sezónách a epizodách. Stránka má dva stavy vůči uživateli. Buď je seriál odebíraný uživatelem, nebo není. V případě, že je odebírán, lze na stránce editovat kategorizaci seriálu, doporučit seriál kamarádovi, sdílet na Facebook či zrušit jeho odběr. Pokud seriál není uživatelem odebíraný, pak je přístupné pouze tlačítko odebírat vyvolávající formulář stejný jako v případě odběru na stránce vyhledávání.

Obrázek 3.6: Stránka seriálu



Obrázek 3.7: Seznam sezón a epizod

3.3 Testování prototypu

Abychom byli schopni otestovat navrženou funkcionalitu, je třeba sestavit testovací scénáře, které pokryjí stěžejní funkce celé aplikace.

Pro aplikaci je velice důležité, aby se nový uživatel po prvním přihlášení v aplikaci rychle zorientoval a byl shopný ji začít používat. První scénář se proto věnuje této problematice a reflektuje přihlášení do aplikace, postup přidání prvního seriálu ke sledování a následně testuje interakci s ovládacími prvky na domovské obrazovce. Scénář pro testování s uživateli by zároveň neměl obsahovat instrukce pro konkrétní interakce s uživatelským rozhraním (kliknout na tlačítko, zadat text do pole vyhledávání atd), nýbrž obecné úkoly, které uživatel musí převést na konečné akce s aplikací. Kompletní seznam úkolů prvního scénáře je obsahem přílohy D.1.

Druhý scénář naopak testuje orientaci uživatele v aplikaci za stavu, kdy již odebírá větší množství seriálů. Těmito akcemi jsou především filtrování a vyhledávání odebíraných seriálů, listování v seznamu epizod a akce nad nimi, notifikace aplikace a orientace na stránce seriálu. Seznam úkolů druhého testovacího scénáře je obsahem přílohy D.2.

Na základě scénářů byly vytvořeny prototypy, které umožňují průchod aplikací v jimi určené posloupnosti. Prototypy přikládám v příloze D.

3.3.1 Testování s uživateli

Z důvodu časové náročnosti zpracování výsledků testování jsem zvolil malý testovací vzorek participantů, který nemůže odhalit všechny problémy v navržené aplikaci, ale poodhalí ty nejzávažnější. Pro výběr participantů jsem použil stejné techniky jako při výběru participantů uživatelského průzkumu v sekci 2.4.1. Cílová skupina byla taktéž totožná.

Participant 1

Popis uživatele.

Pohlaví: Muž

Věk: 22

Status: Student

Poznámky z testování.

1. *Scénář.*

- Přihlášení do aplikace a vyvolání formuláře odběru prvního seriálu bez náznaku problému
- Není si jistý, jak lze ovlivnit pořadí v jakém bude seriál sledovat, předpokládá, že pořadí bude chronologické

- Kategorie seriálu dějový a epizodní jsou pro uživatele pochopitelné, ale není mu jasné, jak souvisí s pořadím sledování
- Výběr zhlédnutých epizod a přidání seriálu v pořádku
- Označení akcí nad epizodou (like a sdílení na FB) s následným označením epizody jako zhlédnuté bez problému

2. Scénář.

- Filtrování epizod ke zhlédnutí dle kategorie seriálu v pořádku
- Otevření stránky seriálu a změna obrázku bez zaváhání
- Nalezení epizody bez problému
- Uživatel tápe v tom, zda epizodu již viděl, nebo ne, a následně chvíli hledá, jak označit epizodu za zhlédnutou
- Změna nastavení kategorizace seriálu v pořádku
- Zobrazení notifikačního centra a otevření nové notifikace bez problému
- Odběr seriálu po vysvětlení kategorizace v předchozím scénáři již v pořádku samostatně
- Uživatel je zmaten, že mu bylo zachováno filtrování na domovské obrazovce
- Po pokynu k odebrání seriálu z odběru uživatel chvíli tápe, kde seriál hledat, ale nakonec zvolí správně sekci moje seriály, vyhledá daný seriál a dá odebrat seriál z odběru
- Filtrování v sekci moje seriály ok a označení seriálu jako "to se mi líbí"bez problému
- Vyhledat nový seriál se uživatel snaží v sekci moje seriály pomocí filtrování, poté textovým polem, následně chce přejít na domovskou obrazovku. Vyhledávání na liště menu nachází až posléze, načež seriál v pořádku vyhledal.
- Odhlášení po pár vteřinách rozmyšlení v pořádku

Shrnutí. Názvy kategorií ovlivňující pořadí epizod při sledování seriálu jsou nesrozumitelné a těžko spojitelné s jejich účelem. Bude více než vhodné zvážit jejich přejmenování, ze kterého bude jasněji vyplývat jejich účel.

Označení stavu epizody (ve smyslu zda byla viděna) není v prototypu zřetelné. Částečně tento fakt je způsoben vzhledem ikony oka, která je v programu Balsamiq dostupná. Druhým faktorem je rozklikávací nabídka cílená pouze na šipku nacházející se vedle něj. Uživatel má více tendenci klikat přímo na oko.

Aplikace obsahuje více vyhledávacích polí pro různé sekce. Pro uživatele je těžko rozpoznatelné, které k jakému účelu slouží, a vykazuje snahu využít první, které spatří.

Participant 2

Popis uživatele.

Pohlaví: Žena

Věk: 25

Status: Zaměstnaný

Poznámky z testování.

1. Scénář.

- Přihlášení v pořádku a vyhledání seriálu v pořádku
- Nejprve uživatel vybírá poslední zhlédnutou epizodu a kategorizaci ignoruje, nenapadá ho, že by měl vybrat dějovou kategorii, aby sledoval seriál popořadě
- Po vysvětlení uživatel navrhuje přímočařejší pojmenování kategorie
- Neví, co ovlivní kategorizace odpočinkový/k zamyšlení
- Označení akcí nad epizodou (like a FB sdílení) a následné zhlédnutí epizody bez problému

2. Scénář.

- Filtrování epizod v pořádku
- Chvilka zaváhání nad změnou obrázku, ale následuje správný klik na ovládací tlačítko jeho změny
- Vyhledání epizody OK
- Stejně jako u prvního participanta špatně rozpoznává, zda viděl, nebo neviděl danou epizodu
- Notifikace zpozorována ihned a správně nad provedena akce otevření
- Uživatel je zaskočen uloženými filtry
- Vyhledání seriálu v sekci moje seriály provedeno správně
- Akce nad seriálem provedeny bez problému
- Neodebíraný seriál se snaží vyhledat taktéž v sekci moje seriály, po objevení globálního vyhledávání uživatel seriál vyhledal v pořádku
- Odhlášení by uživatel řešil zavřením prohlížeče, ale po výzvě objevuje ikonu profilu a úspěšně se odhlašuje

Shrnutí. Uživatel měl problémy s porozuměním kategorizaci seriálu při jeho odběru. Názvy kategorií jsou zavádějící a jejich účel je dle uživatele málo srozumitelný. Rozpoznání stavu zhlédnutí epizody není zcela jasné. Dále uložené filtry na domovské obrazovce matou uživatele a očekávání spíše směřuje k defaultnímu stavu zobrazení epizod všech seriálů. Uživatel nerozlišuje mezi vyhledávacími poli v různých částech stránky. Ty mu způsobují problémy v rozhodnutí, které pole k čemu slouží. Uživatel očekává od aplikace, že ho po zavření okna automaticky odhlásí.

Návrh řešení nálezů z testování

Kategorizace seriálu. Z kategorizace seriálu musí být uživateli ihned jasné, jakým způsobem její volba ovlivní další chování aplikace k onomu seriálu. Zároveň názvy kategorií musí být přímočaré a jasně vyjadřovat jejich význam.

Kategorizace na Odpočinkové seriály a seriály K zamyšlení splňuje přímočarost názvu, avšak nový uživatel aplikace nemá možnost zjistit, k čemu tato volba slouží. Pro informování uživatele bude implementována ikona nápovědy, která bude obsahovat popis funkce této kategorie.

Kategorizace na Dějové a Epizodní seriály nesplňuje ani jedno výše uvedené kritérium. Pro informování uživatele o její funkci opět poslouží ikona nápovědy. Aby názvy kategorizace odpovídaly její funkci, nově budou Dějové seriály označeny jako seriály pro sledování POPOŘADĚ a Epizodní jako seriály pro sledování NÁHODNĚ.

Stav epizody (zhlédnutá x nezhlédnutá). Ikona použitá v prototypu neplní dobře svou funkci. Implementovaná ikona ve finální aplikaci musí být zřetelně symbol oka, pro intuitivní přiřazení její funkce uživatelem. Zároveň její stav musí reflektovat stav zhlédnutí epizody. Toho lze snadno docílit barevným odlišením těchto dvou stavů (červená = nezhlédnutá, zelená = zhlédnutá).

Filtrování epizod na domovské obrazovce. Aby uživatel nebyl maten zachovaným stavem z posledního filtrování epizod, domovská obrazovka vždy zobrazí svůj defaultní stav a vypíší se na ní všechny epizody seriálů, které má uživatel v odběru.

Vyhledávací pole. Aplikace obsahuje příliš velké množství vyhledávacích textových polí, které plní funkci hledání v ruzných sekcích aplikace. Uživatelé nerozeznávají vyhledávání seriálů, které již mají v odběru, a které nikoli. Vyhledávání proto bude sjednoceno do jedné obrazovky, která bude plnit tuto funkci a vyhledá všechny seriály na základě vyhledávacího řetězce nehledně na stav odběru uživatele k němu.

Kapitola 4

Analýza technologií pro klientskou část aplikace

Pro realizaci navrženého řešení je třeba implementovat facebookovou aplikaci. Existují čtyři základní druhy facebookových aplikací:

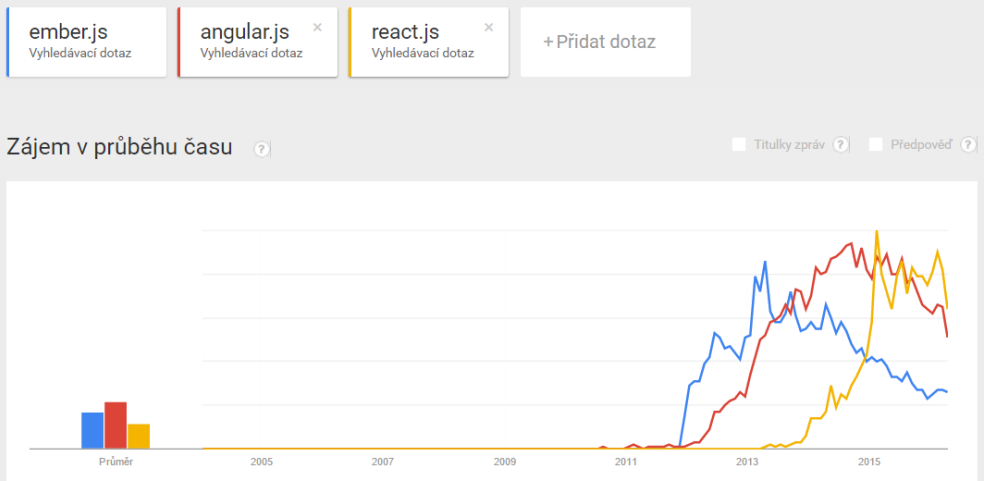
- Aplikace běžící v prostředí Facebooku (dostupná na apps.facebook.com)
- Aplikace běžící v rámci facebookové stránky (dostupná pod záložkou na stránce)
- Webová aplikace komunikující s Facebookem
- Mobilní aplikace

Z výše zmíněných variant je pro potřeby SerialFreaku nejvíce vhodná webová aplikace komunikující s Facebookem. S využitím responzivního designu lze aplikaci snadno provozovat i na mobilních zařízeních, a tím tak ušetřit náklady na vývoj nativní aplikace pro hlavní mobilní platformy.

Pro aplikaci mířící na uživatelský komfort a zažitek je nesmírně důležitá odezva systému na akce uživatele. Tento aspekt zohledňuje trend Single Page Aplikací (SPA). SPA je webová aplikace, která má umístěnu všechnu svou funkcionalitu na jedné stránce a komunikací se serverem zajišťuje dynamické překreslování pouze měnících se fragmentů stránky namísto stahování celého HTML kódu po každé akci uživatele. Překreslování stránky je realizováno javascriptovými funkcemi. Hlavní výhodou tohoto přístupu je tedy datová úspora při přenosu dat, která způsobuje rychlou odezvu systému. Nevýhodou jest nefunkčnost stránky v prohlížeči bez javascriptové podpory.

Existuje nemalé množství javascriptových frameworků, které usnadňují vývoj SPA aplikací. Mezi ty nejrozšířenější patří:

- Ember.js
- Angular.js
- React.js



Obrázek 4.1: Zájem o frameworky v průběhu času. Zdroj: [8]

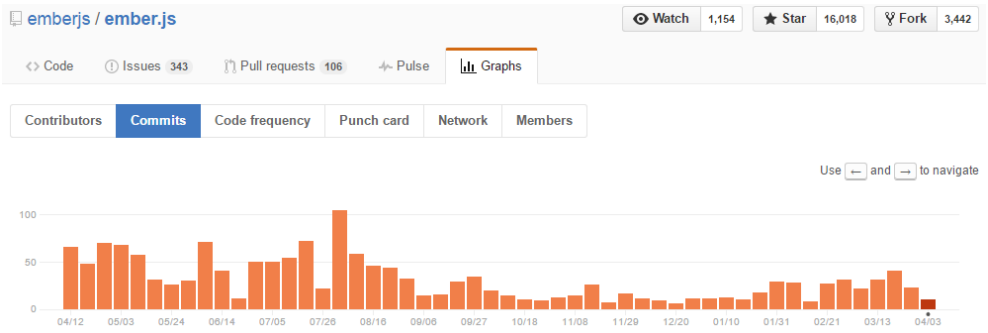
Graf reflektuje množství vyhledávacích dotazů na jednotlivé frameworky. Lze pozorovat obrovskou převahu Angular.js, za kterým stojí velká uživatelská základna. Angular.js byl prvním vydaným frameworkem z výše zmíněných. Za zmínku stojí zvýšení zájmu o nejmladší z frameworků React.js v roce 2015, jehož komunita roste každým dnem.

Každý z frameworků přistupuje k realizaci SPA jiným způsobem a nelze jednoznačně určit, který framework je obecně nejlepší. Výběr frameworku je závislý na potřebách konkrétního projektu. Projekt SerialFreak staví především na uživatelském zážitku, tedy rychlost přechodů v UI je stěžejní. V tomto aspektu je nejlepší React.js využívající virtuální DOM (document object model), díky kterému jsou změny UI rychlejší především při aktualizaci velkého množství položek ve stránce.

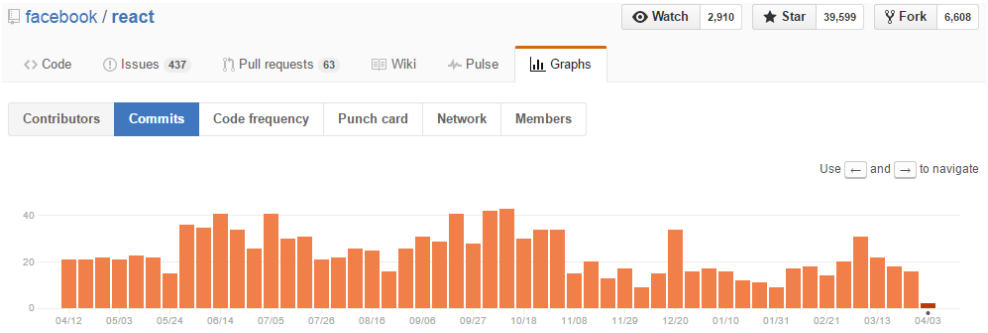
Dalším důležitým faktorem je budoucí podpora frameworku pro další vývoj a rozšíření SerialFreaku v delším časovém horizontu. Aktuálně jest ve vývoji Angular.js 2.0, který nenavazuje na první Angular.js, nýbrž reimplementuje celý framework novým způsobem a nebude podporovat staré API. Ukončení podpory Angular.js je tedy velkým rizikem pro celý projekt a jeho využití nebudu dále uvažovat.

V neposlední řadě je třeba uvažovat rychlost vývoje aplikace. Ember.js představuje velice robustní řešení, které však s sebou přináší velké úsilí pro započetí práce a má velice strmou učící křivku. S ohledem na velikost aplikace, která by nevyužila potenciál Ember.js v celém jeho rozsahu by se pak jednalo o úsilí zbytečné.

Z výše uvedeného plyne jako nejvhodnější využití frameworku React.js, který jest v současné době zároveň nejperspektivnější variantou. Důkazem budiž grafy reflektující počty commitových příspěvků v obou repozitářích projektů, kde lze pozorovat dlouhodobější pokles příspěvků Ember.js projektu.



Obrázek 4.2: Počet commitů v projektu Ember.js v průběhu času. Zdroj: [9]



Obrázek 4.3: Počet commitů v projektu React.js v průběhu času. Zdroj: [10]

4.1 React.js

Framework vytvořil Facebook s cílem odstranit problémy, se kterými se potýkal při vývoji UI své aplikace. Veřejnosti byl představen v roce 2013. Po jeho zveřejnění svými přednostmi přesvědčil spoustu velkých projektů. Mezi ty nejznámější, které jsou na React.js postaveny, patří Instagram, Netflix, Yahoo mail client, Sberbank, BBC a seznam by mohl pokračovat dále.

React.js je open-source javascriptovou knihovnou umožňující dynamické zobrazení dat (view) modelu. Views jsou renderovány jako HTML tagy a jsou vkládány do stránky. Samotné view jsou vytvářeny jako hierarchie komponent, ve které nemohou podkomponenty ovlivnit nadřazené komponenty. Tok dat je striktně nastaven shora dolů. Změna dat způsobuje efektivní aktualizaci HTML prvků ve stránce, které tato data ovlivňují.

4.1.1 Hlavní vlastnosti

Hlavní vlastnosti frameworku React.js jsou následující.

Jednostranný datový tok

Grafické rozhraní je definováno funkcí stavu modelu. Nadřazený prvek UI nemůže být přímo ovlivněn svým potomkem, nýbrž potomek vyvolává funkci,

která může zasáhnout do modelu aplikace, a změnit tak její stav. Změnu stavu registruje nadřazený prvek, na jehož základě aktualizuje části view, které jsou změnou dat ovlivněny.

Virtuální DOM

React vytváří v paměti svou vlastní datovou strukturu, nad kterou programátor pracuje. S touto strukturou tedy může být zacházeno, jako kdyby byla po každé změně celá znovu vyrenderována. Namísto toho však React v reálném DOM mění pouze ty části, kterých se změna opravdu dotýká.

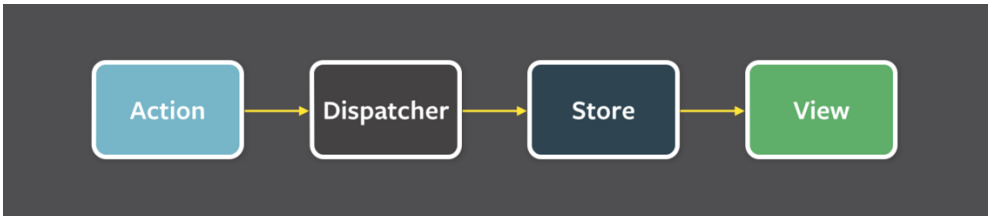
JSX (JavaScript XML)

Rozšíření javascriptové syntaxe pro vytváření javascriptových objektů pomocí HTML.

4.1.2 Flux

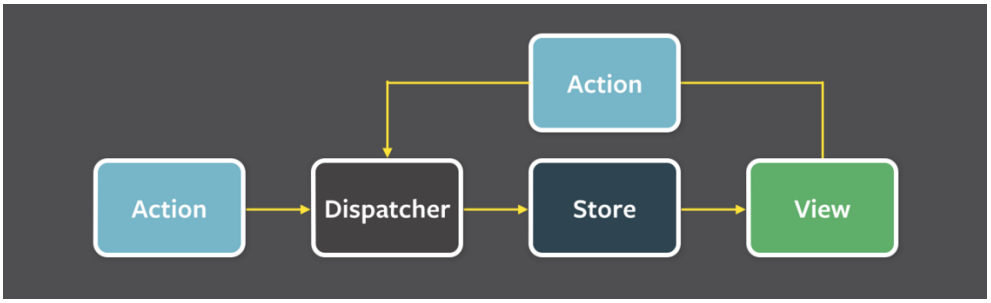
Flux je aplikační architektura, která doplňuje hierarchické sestavení komponent React.js frameworku o definici jednostranného datového toku v aplikaci. Základními pilíři architektury jsou tyto tři části:

- Dispather
- Stores
- Views (komponenty Reactu)



Obrázek 4.4: Datový tok v architektuře flux. Zdroj: [15]

Actions jsou jednoduché objekty nesoucí nová data. Všechna data prochází dispatcherem, který jest centrálním uzlem. Tento uzel vyvolává callback funkce, které registrují jednotlivá Stores, díky čemuž jsou data doručena všem relevantním adresátům. Stores vyvolávají události informující Views o změně dat. Views aktualizují data a změní svůj stav, čím způsobí aktualizaci dat ve všech svých potomcích. Změna dat ve Views může vyvolat další Actions, která jsou opět předána Dispatcherovi, a celý cyklus se opakuje.



Obrázek 4.5: Actions v architektuře Flux. Zdroj: [15]

Flux není nezbytně nutné implementovat pro použití React.js, ale je dobrou konvencí i investicí ve vývoji SPA aplikací pomocí React.js. Architektura odstraňuje problémy s cyklícími se akcemi klasické MVC architektury díky využití Dispatcheru a zajištění dokončení všech změn na Views způsobené jednou akcí před předáním akce další na Stores. Díky tomuto faktu jsou všechny akce nad daty synchronizované a jsou prováděny ve správném pořadí.

Kapitola 5
Implementace

V současné době se již klientské webové aplikace nestarají pouze o hloupé zobrazování dat, nýbrž ve většině případů obstarávají i nemalou část aplikační logiky. Ve spojení s realizací SPA a responzivity se jedná o netriviální část softwaru, která pro svůj vývoj vyžaduje vhodné vývojové prostředí. Je více než vhodné přistoupit k použití dostupných knihoven, které řeší velké množství základních problémů, na které může vývojář při implementaci takové aplikace narazit.

5.1 Vývojové prostředí

Základním kamenem vývojového prostředí je IDE. Pro vývoj javascriptové webové aplikace jsem zvolil JetBrains Webstorm [12], který jest leaderem na poli vývojových prostředí pro tvorbu klienstkých webových aplikací. Poskytuje možnost instalace podpůrných pluginů pro vývoj nad frameworkem React.js, díky čemuž usnadní ladění aplikace.

5.2 Použité technologie

Pro vývoj aplikace bylo použito mnoho technologií. Jejich základní výčet je obsahem následujících podkapitol.

5.2.1 Balíčkovací systém

Pro správu knihoven a závislostí v aplikaci jsem zvolil balíčkovací systém NPM [13]. Původně byl balíčkovacím systémem technologie Node.js [14]. Nyní se již jedná o obecný javascriptový balíčkovací systém. Pro svůj běh však stále potřebuje prosředí Node.js, proto je instalace Node.js nezbytná.

Nejlepším způsobem jak spravovat lokálně instalované balíčky je použití package.json souboru. Jeho základními výhodami jsou:

1. Poskytuje dokumentaci, na jakých balíčcích je projekt závislý
2. Umožňuje specifikovat verzi balíčku, na kterém projekt závisí

3. Dělá projekt reprodukovatelným, čímž usnadňuje jeho sdílení mezi vývojáři

5.2.2 Framework

Z analýzy 4 vzešel jako nejvhodnější framework pro realizaci SPA React.js. Jeho základní vlastnosti již byly také pospány v kapitole 4.1. Konkrétní postup vývoje React.js aplikace bude popsán v následující kapitole 5.4.

5.2.3 Knihovny

V průběhu vývoje počet použitých knihoven vzrostl až na číslo 23. Nebudu zde popisovat všechny z nich, avšak pouze ty stěžejní, na kterých je aplikace postavena. Kompletní výpis knihoven a jejich verzí lze najít ve zdrojových kódech v souboru package.json.

Aplikační závislosti

Flux [15]. Jak již bylo zmíněno, Flux je především styl architektury aplikace nežli knihovna. Avšak je k dispozici balíček, který poskytuje implementaci jeho stěžejní součásti, a tím je Dispatcher. Pro základní předávání událostí (actions) mezi uživatelským rozhraním (Views) a úložišti (Stores) je tato implementace dostatečná.

React-Bootstrap [16]. Jedná se o knihovnu frontend ovládacích prvků webových stránek jako jsou tlačítka, formulářové prvky, menu apod. založená na Twitter Bootstrap, ale implementovaná nad frameworkem React.js. Díky tomuto faktu je jejich použití v React.js aplikaci velice snadné při zachování čistého a čitelného kódu.

React-Router [17]. React Router udržuje uživatelské prostředí synchronizované s URL internetového prohlížeče a poskytuje API, díky kterému je možné pomocí URL směrovat na různé části aplikace či rozpoznávat dynamická URL pro zobrazení konkrétní stránky (například stránka seriálu, kdy je v URL uvedeno id seriálu, který má být zobrazen).

jQuery [18]. Velice bohatá javascriptová knihovna umožňující především dynamickou manipulaci s HTML prvky stránky, dále pak zpracování událostí nad nimi a v neposlední řadě Ajax, který je mnohem snáze použitelný díky API, které pro něj jQuery poskytuje.

Async [19]. Async je sada užitečných funkcí usnadňující práci s asynchronním javascriptem. Obsahuje okolo 70 funkcí, které se hrubě dělí na funkce pro běžnou práci s kolekcemi a na funkce, které řídí asynchronní tok zpracování dat.

Vývojové závislosti

browserify [20]. Zprostředovává možnost načítání nainstalováných npm závislostí pomocí funkce require(). Browserify rekursivně analyzuje veškerá užití funkce require() a sestaví jeden kompletní funkční javascriptový balíček, který může být předložen prohlížeči.

watchify [21]. Jedná se o knihovnu umožňující spuštění broserify v režimu sledování, kdy na každou změnu v javascript kódu je vyvoláno znovusestavení kompletního javascriptového balíčku pomocí browserify. Díky tomuto faktu je hlavní javascriptový balíček inkrementálně rebuildován za běhu a vývojář se o jeho sestavení v průběhu vývoje nemusí starat.

gulp [22]. Gulp je nástroj pro automatizaci procesů, které webový vývojář musí opakovaně vykonávat ručně. Jeho hlavním cílem je především zvýšení produktivity vývojáře. Pomocí Gulp lze snadno spouštět a vykonávat například následující úlohy:

- Sestavování a minimalizování knihoven
- Obnovení prohlížeče při změně zdrojového kódu
- Spouštění unit testů
- Spouštění analýzy kódu
- Kompilace Less/Sass do CSS souborů
- Přesouvání změněných souborů do cílového či distribučního adresáře

gulp-babel [23]. Plugin pro Gulp zpřístupňující kompilátor Babel [24], za jehož pomoci lze psát javascript pomocí nové syntaxe bez ohledu na podporu u internetových prohlížečů. Dále umožňuje konverzi JSX 4.1.1 syntaxe frameworku React.js.

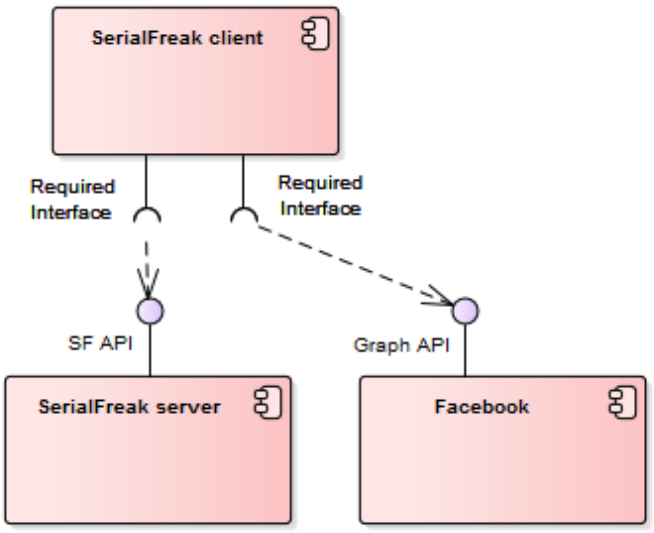
gulp-browserify [25]. Plugin pro Gulp zpřístupňující funkce knihovny Browserify 5.2.3.

gulp-uglify [26]. Plugin pro Gulp zpřístupňující funkce knihovny UglifyJS [27]. Tato knihovna umožňuje především komprimovat javascriptový kód, a vytvořit tak minifikovanou verzi javascriptového balíčku vhodnou pro distribuci na webový server.

gulp-react [28]. Plugin pro Gulp řešící překlad React JSX do prostého javascriptu.

5.3 Architektura aplikace

Aplikace bude komunikovat se dvěma rozhraními. Hlavní komunikační kanál se serverovou stranou aplikace bude realizován pomocí REST rozhraní komunikujícím přes https protokol. Jeho návrhem se zabývá ve své diplomové práci kolega Jakub Pýcha. Druhou implementovanou komunikací bude spojení s API Facebooku pro realizaci přihlášení přes Facebok a provedení akcí uživatele, které nevyžadují zásah serveru (sdílení seriálů, či epizod na Facebook).



Obrázek 5.1: Komponentový model aplikace

Samotná vnitřní implementace klientské části aplikace bude ctít zásady architektury Flux popsané v sekci 4.1.2.

5.4 Implementace aplikace nad React.js

Vývoj aplikace nad frameworkem React.js vyžaduje důkladnou přípravu před započetím práce. Jednosměrný datový tok 4.1.1 zaručuje nemožnost komunikace podkomponent s nadřazenými komponentami. Je třeba proto důkladně navrhnout jejich hierarchii, analyzovat jejich závislosti a zodpovědnosti a na tomto základě sestavit strom komponent, ze kterých vznikne konečné UI aplikace.

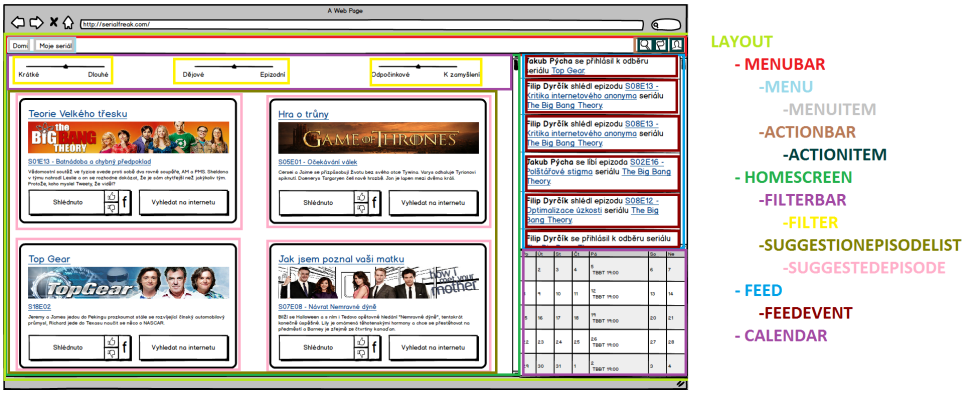
Pro začínající vývojáře Facebook publikuje článek v dokumentaci frameworku React, který popisuje, jak začít s vývojem React aplikace a jakým způsobem nad ní přemýšlet. Článek je v originálním znění nazván Thinking in React [29] a byl mi velkou inspirací při započetí práce. Popsaný postup reflektují následující kapitoly textu.

5.4.1 Sestavení komponentové hierarchie

Prvním krokem jest návrh rozložení komponent na jednotlivých obrazovkách. Jelikož UI obrazovek již máme ze sekce 3.2 navržené, rozdělíme na nich prvky do hierarchie komponent. Pomocí barevných obdélníků vyznačíme komponenty. Vnořený obdélník je vždy podkomponenta komponenty, která ji ohraničuje. Po tomto vyznačení sestavíme strom komponent. Jedna komponenta může být zároveň součástí více obrazovek, proto komponenty na různých obrazovkách nemusí být jedinečné. Rozvržení komponent na obrazovkách jsem navrhl následovně.

Domovská obrazovka.
Nejvýše nadřazenou komponentou je komponenta Layout pokrývající celou plochu obrazovky. Jejími přímými podkomponentami jsou Menubar (menu aplikace), Feed (výpis událostí přátel přihlášeného uživatele), Calendar (kalendář s daty vysílání nových epizod odebíraných seriálů) a Homescreen (komponenta zobrazující uživateli doporučené epizody ke zhlédnutí). První 3 jmenované jsou společné pro všechny obrazovky a není třeba je uvádět znovu.

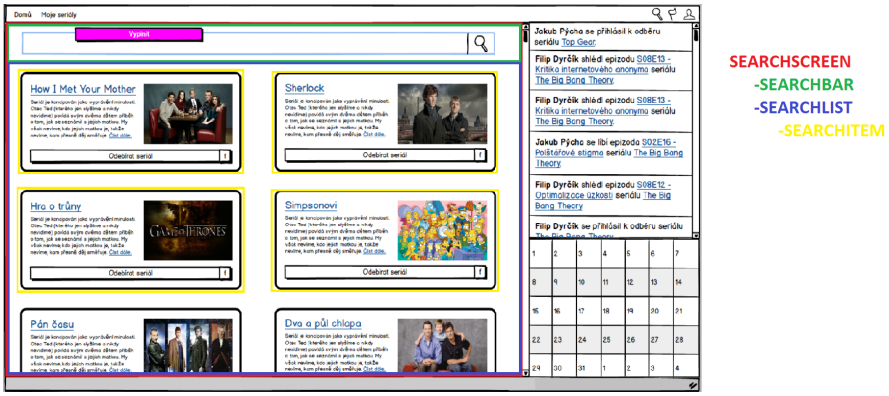
Komponenta Homescreen dále obsahuje dvě komponenty a to Filterbar, jejíž potomci jsou filtry sloužící k upřesnění výběru epizod, a Suggestedepisodelist, která obsahuje již konkrétní epizody ke zhlédnutí.



Obrázek 5.2: Rozvržení komponent Domovské obrazovky

Vyhledávání.

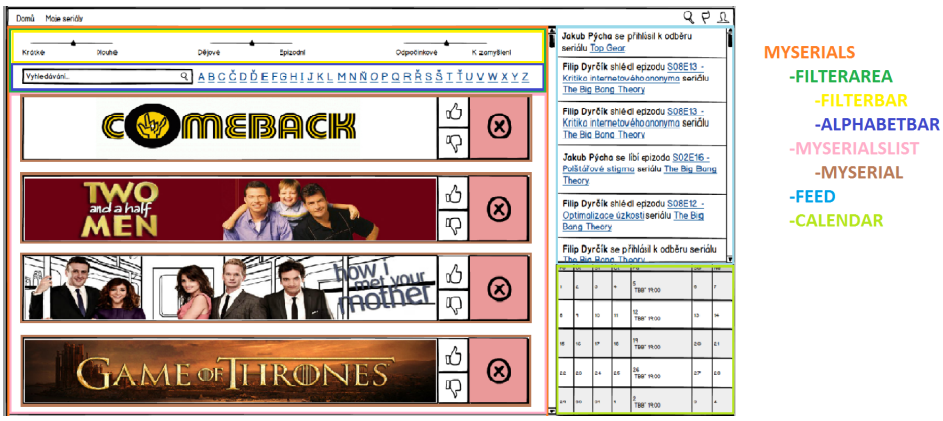
Na obrazovce vyhledávání je komponenta Homescreen nahrazena komponentou Searchscreen. Tato komponenta obsahuje komponentu Searchbar, která obsluhuje vstupy uživatele, a komponentu Searchlist zobrazující seznam vyhledaných epizod, kde každý vyhledaný seriál reprezentuje jedna komponenta SearchItem.



Obrázek 5.3: Rozvržení komponent na stránce Vyhledávání

Moje seriály.

Sekce Moje seriály namísto komponenty Homescreen obsahuje komponentu Myserials. Jejími potomky pak jsou komponenty Filterarea a Myseriallist. Filterarea obsahuje Filterbar stejně tak jako Homescreen, ale přidává k ní ještě komponentu Alphabetbar umožňující filtrování seriálu dle počátečního písmena (vyhledávací pole bylo z návrhu odstraněno po zhodnocení nálezu 3.3.1 z testování s uživateli).



Obrázek 5.4: Rozvržení komponent v sekci Moje seriály

Seriál.

Rozložení UI na obrazovce seriálu se v průběhu vývoje změnilo natolik, že zde nebudu demostrovat na prototypu původně navrženou hierarchii komponent. Konečnou podobu obrazovky seriálu popíši později v sekci 5.9.

5.4.2 Určení minimální reprezentace stavu UI

Pro každou obrazovku UI je nyní třeba určit, jaká data jsou dostatečná pro její zobrazení. Tento minimální soubor dat, který definuje zobrazené uživatelské prostředí, se nazývá stav (state). Tento stav může být rozdělen mezi více komponent dané obrazovky. Dalším krokem tedy je určení, jakou část stavu má která komponenta udržovat.

Základní postup, jak rozdělit stav mezi komponenty, je následující. Pro každou část stavu v aplikaci:

1. Identifikovat všechny komponenty, které vykreslují něco na základě tohoto stavu.
2. Nalézt společného vlastníka (komponentu, která leží nade všemi komponentami, které potřebují tento stav).
3. Společný vlastník nebo jakákoli jemu nadřazená komponenta má vlastnit tento stav.
4. Pokud nelze nalézt komponentu, která by dávala smysl pro držení tohoto stavu, pak lze založit komponentu novou pouze pro držení stavu a umístit ji kamkoli nad společného vlastníka komponent, které potřebují tento stav.

Výše uvedený postup popíši na konkrétním případu domovské obrazovky. Data (stav), která aplikace potřebuje k zobrazení domovské obrazovky, jsou následující:

1. Stav přihlášení uživatele
2. Počet nepřečtených notifikací
3. Seznam doporučených epizod
4. Stavy filtrů epizod
5. Stav ovládacích prvků (like, dislike, FB sdílení) nad epizodou
6. Seznam aktivit přátel uživatele

Všechna ostatní vykreslená data se odvíjejí od výše zmíněných a dají se dopočítat. Na stavu přihlášení závisí všechny komponenty domovské obrazovky, proto informaci o tomto stavu musí nést nejvýše položená komponenta, tedy Layout. Počet nepřečtených notifikací je zobrazen pouze na liště Menu, proto tento stav může náležet právě jí. Seznam doporučených epizod vykresluje Suggestedepisodelist, čili by i on měl držet tento stav. Jelikož v dotazu na server na doporučené epizody je třeba zadat stavy všech filtrů, nemůžou svůj stav držet individuálně. Proto stav každého filtru drží nadřazená komponenta Filterbar. Doporučené epizody jsou samostatné jednotky, tudíž si stav svých ovládacích prvků mohou držet samy. Aktivity uživatele vykresluje pouze komponenta Feed, proto jejich seznam bude držet ve svém stavu právě ona.

Totožným způsobem je třeba pokrýt všechny obrazovky celé aplikace. Každý stav lze traverzovat hierarchií komponent dolů pomocí vlastností (props) jednotlivých komponent. Opakem je situace, kdy potřebujeme notifikovat nadřazenou komponentu o změně stavu. Příkladem budiž filtrovací lišta. Pokud se změní hodnota jednoho z filtrů, je třeba notifikovat Filterbar, která způsobí znovunačtení doporučených epizod dle aktuálních hodnot filtrů. React tuto situaci neřeší implicitně, ale nechává řešení na vývojáři. Je proto naprosto transparentní, jak datový tok v aplikaci probíhá. Implementačním řešením této situace je předání callback funkce každému filtru pomocí props, která volá nad komponentou Filterbar funkci setState(), a tím aktualizuje stav komponenty.

5.4.3 Implementace React.js komponenty

Po definování komponent a jejich stavů nastává čas na jejich implementaci. Postup a principy jejich tvorby popíši na konkétním zjednodušeném případu domovské obrazovky. Pro vytvoření komponenty použijeme metodu createClass() React knihovny. Jedinou metodou, kterou nezbytně musí komponenta implementovat, je metoda render, která navrací HTML kód, který má být vykreslen na stránku.

```
var React = require('react');

var HomeScreen = React.createClass({

    render: function () {
        return (
            <div id="homeScreen">
                <h1>Epizody ke shlédnutí</h1>
            </div>
        );
    }
});

module.exports = HomeScreen;
```

Komponentě Domovské obrazovky v sekci 5.4.2 nebyl přiřazen žádný stav. Z důvodu zjednodušení implementace se však ukázalo jako vhodné nedržet seznam doporučených epizod v komponentě suggestedEpisodeList, nýbrž právě v komponentě Domovské obrazovky. Stav je nutné inicializovat a k tomu slouží metoda getInitialState(). Prvotní stav komponenty je prázdné pole doporučených epizod. V metodě render() již nyní můžeme zobrazit seznam epizod právě na základě definovaného stavu.

```
var HomeScreen = React.createClass({
    getInitialState: function () {
        return {
            suggestedEpisodes: []
```

```
        };
    },
    render: function () {
        return (
            <div id="homeScreen">
                <h1>Epizody ke shlédnutí</h1>
                {this.state.suggestedEpisodes}
            </div>
        );
    }
});
```

Doporučené epizody ke zhlédnutí je třeba získat ze serveru. Komponenta tedy musí v určité chvíli získat pomocí API potřebná data a uložit je do svého stavu. Stav je aktualizován pomocí metody setState(). Aktualizace stavu následně způsobuje opětovné volání metody render, tím pádem překreslení uživatelského rozhraní s novým stavem.

Moment, ve kterém se má komponenta dotázat serveru na data, lze zvolit z několika metod. Metody jsou volány na základě životního cyklu komponenty a jsou následující.

Usazení komponenty (Mounting). Metody volané, když je vytvořena instance komponenty a vložená do DOM

- getInitialState()
- componentWillMount()
- render()
- componentDidMount()

Aktualizace komponenty (Updating). Aktualizace komponenty je způsobena změnou stavu (state) nebo vlastností (props). Metody jsou volány, pokud bude komponenta překreslena.

- componentWillReceiveProps()
- shoudComponentUpdate()
- componentWillUpdate()
- render()
- componentDidUpdate()

Odebrání komponenty (Mounting). Metoda je volána, pokud bude komponenta odebrána z DOM

- componentWillUnmount()

V našem příkladu, kdy chceme získat seznam epizod ke zhlédnutí, je nejvhodnější volání API v metodě componentDidMount(), která je volána po usazení a vykreslení komponenty v DOM. Po získání nových dat zavoláme v callback funkci metodu setState(), čímž aktualizujeme stav komponenty, a vyvoláme tak sled aktualizačních funkcí 5.4.3, mezi nimiž je i metoda render(), díky čemuž se vykreslí námi získané epizody. Celá implementace komponenty vypadá následovně.

```
var React = require('react');
var HomeScreenApi = require('../api/homescreenApi');

var HomeScreen = React.createClass({
    getInitialState: function () {
        return {
            suggestedEpisodes: []
        };
    },
    componentDidMount: function () {
        HomeScreenApi.getSuggestedEpisodes(function(episodes){
            this.setState({
                suggestedEpisodes: episodes
            });
        });
    },
    render: function () {
        return (
            <div id="homeScreen">
                <h1>Epizody ke shlédnutí</h1>
                {this.state.suggestedEpisodes}
            </div>
        );
    }
});

module.exports = HomeScreen;
```

Tato triviální implementace postačuje k prostému výpisu epizod (za předpokladu, že pole epizod navrácené serverem obsahuje jednoduché informace, například pouze pole názvů epizod). Skutečná implementace komponenty vyžaduje více asynchronních volání API pro získání potřebných dat k zobrazení všech informací o doporučené epizodě a implementuje především architekturu Flux namísto přímočaré aktualizace stavu na základě callback funkce. Implementaci Flux bude věnována další sekce 5.5.

5.5 Implementace architektury Flux

Pro implementaci jednosměrného datového toku v aplikaci jsem se rozhodl použít architekturu Flux [15]. Její základní myšlenku jsem již zmínil v sekci 4.1.2, proto se nyní již budu zabývat pouze její konkrétní realizací v aplikaci SerialFreak. Naváži na předchozí sekci popisující základní implementaci komponenty Domovské obrazovky a ukáži dále, jakým způsobem je nutné aplikaci upravit, aby ctila zásady architektury Flux.

Pro realizaci Flux je třeba vytvořit čtyři základní funkční jednotky, kterými jsou Dispatcher, Store, View a ActionCreators. View již máme připravené z předchozí sekce 5.4.3 a Dispatcher lze převzít z knihovny Flux 5.2.3. Tím nám zbývají pouze Stores pro uchovávání dat a notifikování View o jejich změně a ActionCreators, pomocí nichž budou vytvářené Actions nesoucí nová data, která Dispatcher doručí cílovým Store.

Nejprve vytvoříme Store pro uchování doporučených epizod. Implementace vypadá následovně.

```
var SerialFreakDispatcher = require('flux').Dispatcher
var EventEmitter = require('events').EventEmitter;
var assign = require('object-assign');

var CHANGE_EVENT = 'change';
var _suggestedEpisodes = [];

var HomeScreenStore = assign({}, EventEmitter.prototype, {
    emitChange: function () {
        this.emit(CHANGE_EVENT);
    },
    addChangeListener: function(callback) {
        this.on(CHANGE_EVENT, callback);
    },
    removeChangeListener: function(callback) {
        this.removeListener(CHANGE_EVENT, callback);
    },
    getSuggestedEpisodes: function (callback) {
        callback(_suggestedEpisodes);
    }
});
HomeScreenStore.dispatchToken=SerialFreakDispatcher.register(
  function (action) {
    switch (action.type){
        case 'getEpisodes':
            _suggestedEpisodes = {
                suggestedEpisodes: action.suggestedEpisodes
            };
            HomeScreenStore.emitChange();
            break;
```

```
        default:
    }
  }
);

module.exports = HomeScreenStore;
```

Základ Store tvoří EventEmitter knihovny Events [30], který umožňuje vyvolávání pojmenovaných událostí (v implementaci HomeScreenStore události 'change'), na základě kterých volá registrované funkce (listeners). Pro vyvolání události slouží funkce emit(), k zaregistrování listeneru pak funkce on(). Pro větší transparentnost funkcí vystavuje Store navenek funkce emitChange() pro vyvolání události změny dat, addChangeListener() pro registraci callback funkce, removeChangeListener() pro odebrání callback funkce a getSuggestedEpisodes() pro získání aktuálně uložených doporučených epizod pro Domovskou obrazovku.

Store se následně registruje u Dispatcheru, aby mu byly delegovány v aplikaci vzniklé Actions, které popíši společně s ActionCreators níže. Pokud je typ akce 'getEpisodes', pak akce obsahuje i doporučené epizody, které jsou uloženy do proměnné _suggestedEpisodes, a je zavolána funkce emitChange(), která způsobí provolání všech registrovaných funkcí.

O vytváření Actions se starají ActionCreators. Actions jsou malé objekty nesoucí informaci o typu akce a zároveň v sobě uchovávají nová data určená k uložení do Store a následnému vykreslení do View. Implementace ActionCreators pro Domovskou obrazovku vypadá následovně.

```
var SerialFreakDispatcher = require(
    '../dispatcher/SerialFreakDispatcher'
);

module.exports = {
    receiveSuggestedEpisodes: function (episodes) {
        SerialFreakDispatcher.dispatch({
            type: 'getEpisodes',
            suggestedEpisodes: episodes
        });
    }
};
```

Modul HomeScreenActionCreators obsahuje jedinou funkci, a to receiveSuggestedEpisodes(). Jak již název napovídá, jedná se o funkci volanou při přijetí doporučených epizod ze serverového API. Funkce deleguje na Dispatcher nový objekt (akci), do kterého uloží typ akce a samotné přijaté epizody. Dispatcher následně rozešle tuto akci všem Stores, kde je, nebo není zpracována na základě typu akce, jak bylo popsáno výše.

Posledním krokem je původně navržené View (komponenta Domovské obrazovky) upravit pro využití nově vytvořených struktur architektury Flux.

```
var React = require('react');
var HomeScreenApi = require('../api/homescreenApi');
var HomeScreenStore = require('../../stores/HomeScreenStore');
var HomeScreenActionCreators = require(
    '../../actions/HomeScreenActionCreators'
);
var HomeScreen = React.createClass({
    getInitialState: function () {
        return {
            suggestedEpisodes: []
        };
    },
    componentDidMount: function () {
        HomeScreenStore.addChangeListener(
            this._onHomeScreenChange
        );

        HomeScreenApi.getSuggestedEpisodes(function(episodes){
            HomeScreenActionCreators
                .receiveSuggestedEpisodes(episodes);
        });
    },
    componentWillUnmount: function () {
        HomeScreenStore
            .removeChangeListener(this._onHomeScreenChange);
    },
    _onHomeScreenChange: function () {
        HomeScreenStore
            .getSuggestedEpisodes(this.setSuggestedEpisodes);
    },
    setSuggestedEpisodes: function (episodes) {
        this.setState({
            suggestedEpisodes: episodes.suggestedEpisodes
        });
    },
    render: function () {
        return (
            <div id="homeScreen">
                <h1>Epizody ke shlédnutí</h1>
                {this.state.suggestedEpisodes}
            </div>
        );
    }
});
module.exports = HomeScreen;
```

Po usazení komponenty je u HomeScreenStore zaregistrována jako listener metoda _onHomeScreenChange(), která získává nová data ze store a ukládá je do stavu komponenty. Následuje volání API, kde je v callback funkci pomocí HomeScreenActionCretors vytvořena akce přijetí epizod. Tu zpracuje následně Store, uloží nově získané epizody a volá dříve registrovanou metodu _onHomeScreenChange(), čímž se aktualizuje stav komponenty a epizody jsou vykresleny do DOM. Pokud bude komponenta odstraněna z DOM, pak je volána metoda componentWillUnmount() a z HomeScreenStore je odstraněn i listener této komponenty.

Nyní je minimalistická názorná ukázka realizace komponenty pomocí architektury Flux kompletní. Stejným postupem jsou implementovány i všechny ostatní komponenty aplikace, které asynchronně získávají data ze serveru či facebookového Graph API.

5.6 Navigace v React.js aplikaci

Jak již bylo zmíněno v sekci 5.2.3, k navigaci v aplikaci jsem použil knihovnu React Router. Jejími základními kameny jsou komponenty Router, Route, IndexRoute a Link. S využitím JSX 4.1.1 je lze použít stějně jako každou jinou React komponentu.

Router, Route a IndexRoute slouží k základní konfiguraci hierarchie komponent. Komponenta Router je nejvýše postavenou komponentou celé hierarchie komponent celé aplikace. Jeho potomky jsou pak komponenty Route a IndexRoute, kde IndexRoute udává výchozí komponentu k zobrazení, pokud URL neudává konkrétnější cestu k více vnořené komponentě. Komponenty Route dále definují, na jaká URL se mají zobrazit jaké komponenty. Základní konfigurace routeru aplikace SerialFreak vypadá následovně.

```
<Router history={History}>
    <Route path="/" component={Screen}>
        <IndexRoute component={Layout}/>
        <Route path="login" component={LoginPage}/>
        <Route path="homescreen" component={Layout}>
            <IndexRoute component={Homescreen}/>
        </Route>
        <Route component={Layout}>
            <Route path="myserials" component={MySerials}/>
            <Route path="search" component={Search}/>
            <Route path="series/:id" component={Serial}/>
        </Route>
    </Route>
</Router>
```

K rozpoznání dynamických URL používá React Router parametry, které jsou anotované pomocí dvojtečky. Tento případ lze pozorovat výše v případě komponenty Serial. Každá URL směřující na series/??? (kde otazníky můžou představovat jakýkoli řetězec) je mapována na komponentu Serial, které je

pomocí objektu params řetězec předán. V našem případě je například url series/12586 přeložena do následujícího kódu.

```
<Screen>
  <Layout>
    <Serial params={{ id: '12586' }}/>
  </Layout>
</Screen>
```

5.7 Implementace API

Jak již bylo zmíněno v sekci 5.3, návrh rozhraní mezi klientskou a serverovou stranou aplikace vypracoval kolega Jakub Pýcha. Dokumentaci REST API [31] vystavil na službě Apiary.io [32]. Tato služba také umožňuje vytvoření mockového serveru, díky kterému jsem mohl započít implementační práci na klientu nezávisle na stavu implementace serverové strany. Mockový server navrací předem vytvořená mocková data, která jsou však pro základní implementaci komponent aplikace dostatečná.

Pro samotnou implementaci HTTP požadavků jsem použil ajax [33] z knihovny jQuery 5.2.3. Jeho základní použití budu demostrovat na konkrétním příkladu získání doporučených epizod pro Domovskou obrazovku.

```
var $ = require('jquery');
var serverAdrress = require(
    '../constants/SerialFreakConstants').serverAdrress;

module.exports = {
    getHomescreen: function (callback, error) {
        $.ajax({
            url: serverAdrress + '/users/me/home',
            headers: {
                'Content-Type': 'application/json',
                'x-access-token': sessionStorage
                                        .getItem('token')
            },
            method: 'GET',
            dataType: 'json',
            success: callback,
            error: error
        });
    }
};
```

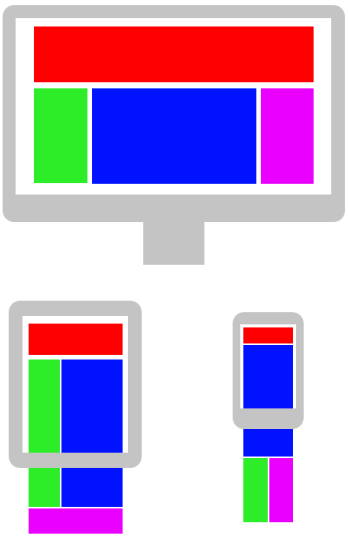
Metoda ajax() přijímá jako parametr objekt definující celý HTTP požadavek. Po jeho vykonání je volána metoda na základě jeho výsledku. Například pokud požadavek metody GET vrátí status kód 200, tak je vykonána metoda

definovaná atributem success a jsou jí předána i navrácená data z těla odpovědi. Pokud je vrácen kód jiný, pak je volána metoda definovaná atributem error.

Vytvoření ostatních HTTP metod (POST,PUT,DELETE) je analogické k výše popsanému požadavku GET. API bylo kompletně implementováno vůči mockovému serveru a až následně integrováno se serverovou aplikací v posledních fázích vývoje.

5.8 Responzivita a vzhled aplikace

Pro podporu mobilních zařízení jsem se rozhodl aplikaci postavit na responzivním designu [34]. Tento termín značí, že bude zobrazení stránky vždy optimalizováno pro všechny druhy zařízení (PC, notebooky, mobily, tablety, atp.). Příklad takového designu je znázorněn na obrázku 5.5.



Obrázek 5.5: Zobrazení totožné stránky na různých zařízeních. Zdroj: [35]

Pro urychlení vývoje aplikace a snadnou implementaci responzivního chování jsem využil knihovny ReactBootsrap 5.2.3. Hlavní výhodou této knihovny je možnost využití komponenty Grid, která obsahuje jednoduchý souřadnicový systém pro pozicování prvků na stránce v závislosti na velikosti plochy dostupné pro zobrazení stránky.

Celý vzhled aplikace je definován pomocí kaskádových stylů. Aplikace využívá moderních CSS3 vlastností, které například umožňují rozšířené možnosti stylování pozadí a ohraničení, 2D/3D transformací, animací apod. Celý stylopis je obsahem souboru main.css ve zdrojových kódech aplikace. Snímky obrazovky finální verze aplikace, jak desktopové, tak mobilní, přikládám v příloze E.

Aplikaci jsem vyvíjel, testoval a ladil v internetovém prohlížeči Google Chrome. Využil jsem jeho široké podpory moderních webových technologií a kvalitních vývojářských nástrojů, které poskytuje. Po dokončení práce byla aplikace odladěna i pro prohlížeč Mozilla Firefox a mobilní Google Chrome. Ostatní prohlížeče prozatím aplikace nepodporuje.

5.9 Změny aplikace v průběhu implementace

V průběhu implementace nastaly situace, kdy bylo třeba zvážit změnu původního návrhu a provést změny v aplikaci. Výčet provedených změn je následující.

5.9.1 Stránka seriálu

Po základní implementaci rozvržení stránky zobrazující detail seriálu 3.2.4 se ukázalo, že úvodní obrázek seriálu na širokoúhlém monitoru zabírá téměř veškerý prostor obrazovky. Důležité informace o seriálu tak byly skryty a uživatel musel provést scrolování, aby vůbec zjistil, že se na stránce nacházejí.

Úvodní obrázek byl proto zaměněn za plakát seriálu, který svými rozměry umožňuje umístění informací o seriálu vedle něj. Odpadá tak nutnost použití tabového menu, pod obrázek a pod informace o seriálu může být umístěn seznam sezón a epizod. Nové rozložení obrazovky seriálu lze pozorovat na snímku obrazovky E.4 v příloze E.

5.9.2 Kalendář

Původně navržený kalendář s daty vysílání nových epizod nebyl z časových důvodů ve finální aplikaci implementován. Jelikož se jednalo o funkční požadavek s nízkou prioritou, není jeho realizace nezbytně nutná. Prostor na stránce, který pro něj byl vyhrazen, nyní zabírá panel událostí přátel uživatele. Jeho doplnění bude součástí budoucího vývoje aplikace.

5.10 Integrace se serverovou částí aplikace

V závěru implementace nastal čas pro integraci klientské a serverové části aplikace. Jelikož byla klientská strana aplikace implementována vůči navrženému mockovému API, které server realizuje, samotná integrace byla v praxi provedena pouhým přepsáním konstanty v kódu, která uchovává URL adresu serveru. Po nasměrování klientské aplikace vůči serverové nenastaly vážné problémy se spuštěním aplikace.

Aplikováním testů 6 byly odhaleny drobné problémy v chování uživatelského rozhraní. Problémy byly způsobeny chybami v kódu jak klientské, tak serverové aplikace. Debugování aplikací jsme s kolegou Jakubem Pýchou prováděli současně, což nám umožnilo problémy řešit velice agilně a rychle.

Kapitola 6

Testování

Aplikace byla průběžně testována manuálními testy uživatelského rozhraní. Pro jejich návrh jsem využil znalostí z předmětu Základy testování software [36]. Testování bylo provedeno vždy vůči mockovému serveru vytvořenému ve službě Apiary [32]. Pro každý funkční požadavek, který je vizualizován v UI aplikace, vznikl alespoň jeden testovací scénář, který ho pokrývá. Každý testovací scénář pak obsahuje jeden či více testovacích případů, které scénář realizují.

Scénáře testují pouze zobrazení jednotlivých komponent uživatelského rozhraní a možné interakce s nimi. Jejich cílem není pokrytí aplikační logiky. Lze je považovat za určitý typ smoke testů [37], které je dobré aplikovat při větším zásahu do rozložení uživatelského rozhraní aplikace.

Na ukázku, jak takový testovací scénář vypadá, uvedu scénář pokrývající funkční požadavek č. 1 - Přihlášení přes Facebook. Kompletní sada testovacích scénářů je obsahem přílohy F.

6.1 Testovací scénář: Přihlášení do aplikace

ID pokrytých funkčních požadavků: 1

ID testovacího scénáře: TS001

Popis: Test přihlášení uživatele do aplikace. Scénář pokrývá i různé stavy přihlášení uživatele do služby Facebook v prohlížeči.

Počet testovacích případů: 3

6.1.1 Testovací případ: Uživatel nepřihlášen na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC001

Předpoklady: Validní přihlašovací údaje ke službě Facebook. Uživatel není v prohlížeči přihlášen ke službě Facebook.

Popis: Test přihlášení uživatele do aplikace za stavu, kdy není v prohlížeči přihlášen do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook
2	Klik na tlačítko Přihlásit se přes Facebook	Zobrazen formulář pro zadání přihlašovacích údajů do služby Facebook
3	Zadat validní přihlašovací údaje a kliknout na tlačítko login	Zobrazena Domovská obrazovka aplikace

6.1.2 Testovací případ: Uživatel je přihlášen na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC002

Předpoklady: Uživatel je v prohlížeči přihlášen ke službě Facebook.

Popis: Test přihlášení uživatele do aplikace za stavu, kdy je v prohlížeči přihlášen do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Domovská obrazovka aplikace

6.1.3 Testovací případ: Změna stavu přihlášení na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC003

Předpoklady: Validní přihlašovací údaje ke službě Facebook. Uživatel není v prohlížeči přihlášen ke službě Facebook.

Popis: Test změny stavu přihlášení v aplikaci při změně stavu přihlášení do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook
2	Přihlásit se v nové záložce prohlížeče do služby Facebook	Uživatel přihlášen do služby Facebook
3	V původní záložce s aplikací SerialFreak provést znovunačtení stránky (F5)	Zobrazena Domovská obrazovka
4	V druhé záložce se stránkou Facebook provést odhlášení	Uživatel odhlášen ze služby Facebook
5	V záložce s aplikací SerialFreak provést znovunačtení stránky (F5)	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook

Kapitola 7

Nasazení

Zdrojové kódy aplikace nelze distribuovat v nezměněné podobě přímo na webový server. Postup nasazení aplikace popíši v následujících dvou kapitolách.

7.1 Build a minifikace zdrojových kódů

Aby bylo možné aplikaci zobrazit v intenetovém prohlížeči, je nutné ji přeložit do běžného javascriptu. Zároveň pro reálné nasazení v praxi je nezbytně nutné zaručit, aby velikost dat, kterou uživatel musí stáhnout do svého prohlížeče, byla co nejmenší. K tomuto účelu jsem sestavil pomocí nástroje Gulp 5.2.3 skript, který řeší oba výše zmíněné problémy a jednoduchý minifikovaný javascriptový soubor přesune do distribučního adresáře.

```
var path = {
    ROOT_JS: 'src/js/app.js',
    DEST_BUILD: 'dist/build',
};
gulp.task('build', function () {
    gulp.src(path.ROOT_JS)
        .pipe(browserify({
            insertGlobals: true,
            debug: !gulp.env.production
        }))
        .pipe(react())
        .pipe(babel({
            presets: ['es2015']
        }))
        .pipe(uglify())
        .pipe(gulp.dest(path.DEST_BUILD));
});
```

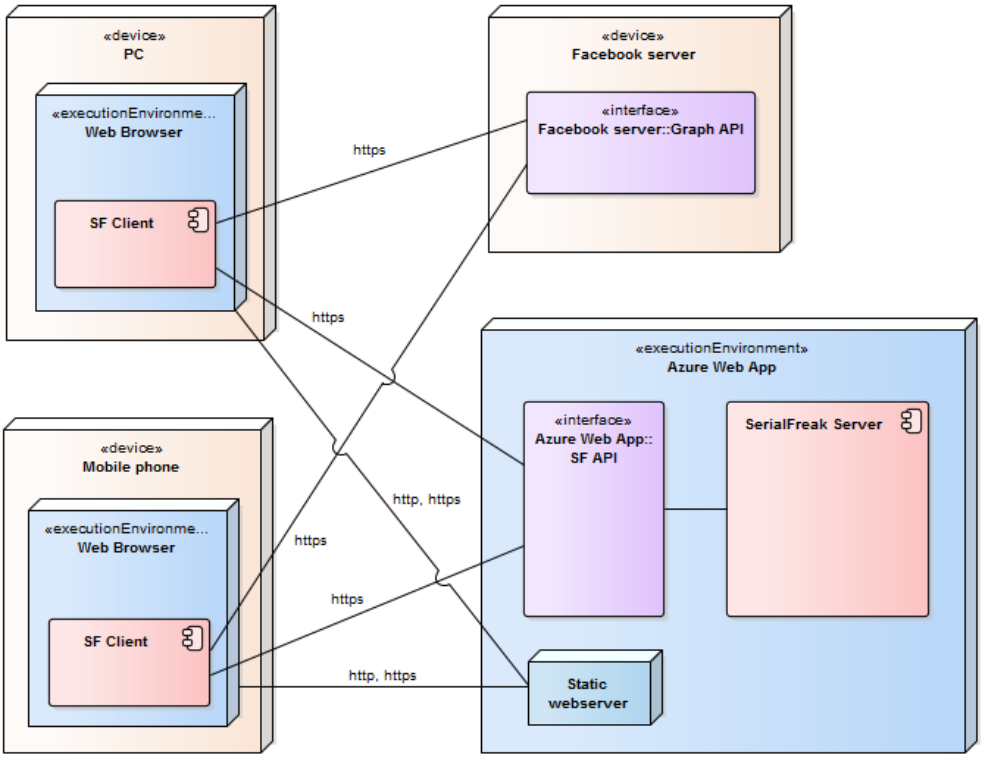
1. Připojení všech závislostí aplikace do jednoho souboru pomocí knihovny gulp-browserify 5.2.3.
2. Překlad React JSX do prostého javascriptu pomocí gulp-react 5.2.3.

3. Překlad syntaxe ES2015 pomocí babel 5.2.3
4. Minifikace javascriptového souboru pomocí uglify 5.2.3
5. Přesun nově vytvořeného souboru app.js do distribučního adresáře

7.2 Webový server

Posledním krokem jest samotné nasazení aplikace na veřejný webový server. Serverová aplikace je nasazena jako Microsoft Azure Web App [38] jejíž, součástí je i Microsoft IIS webový server [39]. Jeho defaultní konfigurace poskytuje možnost vystavit navenek jak klientskou webovou aplikaci, tak i serverové REST API. Rozhodl jsem se využít této možnosti a aplikaci umístit do cloudového prostředí Microsoft Azure Webb App.

Pro nahrání mé klientské části aplikace mi kolega Jakub Pýcha poskytl přístupové údaje ke Git repozitáři sloužícímu k nasazení aplikace do jím vytvořené Azure Web App. V něm se nachází složka public, kam lze klientskou aplikaci umístit. Všechny HTTP požadavky jsou přednostně směrovány právě na ni. HTTP požadavek GET na kořenovou URL proto v odpovědi vrátí soubor index.html, čímž je aplikace zavedena do prohlížeče uživatele. HTTP požadavky, pro které neexistuje v public složce totožně nazvaný zdroj, jsou poté směrovány na serverové API. Výše popsané reflektuje následující diagram nasazení.



Obrázek 7.1: Diagram nasazení

Kapitola 8

Závěr

Aplikace SerialFreak byla na začátku projektu pouhým nápadem. Díky provedení uživatelského průzkumu, který poskytl cenná data o potencionálních uživatelích aplikace, byl nápad transformován na konkrétní vizi aplikace usnadňující správu sledování seriálů a umožňující uživatelům rozšiřovat svůj seriálový svět díky sociálním konexím ze služby Facebook.

Po úspěšném konkretizování požadavků na aplikaci byly vytvořeny základní prototypy aplikace, které byly podrobeny testování s uživateli. Testování neodhalilo zásadní chyby v konceptu aplikace. Drobné nálezy z testování však byly podnětem pro úpravu návrhů před započetím implementace, čímž ušetřili mnoho úsilí, které by bylo nutné vynaložit na opravy, pokud by chyby v návrhu byly nalezeny až v pozdějších fázích vývoje.

Výběr frameworku React.js pro implementaci SPA aplikace byl správnou volbou. Při implementaci jsem nenarazil na žádný problém frameworku, který by mě při vývoji aplikace jakkoli omezil či zdržel. Ve spojení s architekturou Flux se jedná o velice efektivní a snadný způsob realizace SPA aplikace.

Vývoj při striktním rozdělení aplikace na dvě části, kdy jsem neměl možnost zasáhnout do serverové části aplikace, byl pro mě novou zkušeností. Avšak díky jasné definici API a možnosti využití mockového serveru se jednalo o velice efektivní způsob vývoje ve dvoučlenném týmu. Následná integrace klienta a serveru proběhla bez větších problémů pouhým přesměrováním požadavků na reálně nasazený server a odladěním drobných chyb v kódu.

Aplikace je nasazena v cloudovém prostředí Microsoft Azure. Aplikace je v testovacím provozu, kdy je přihlášení do aplikace přístupné pouze definovaným uživatelům služby Facebook. Zrušení testovacího provozu z pohledu klientské části aplikace podléhá především vyřešení podpory zbylých nejrozšířenějších internetových prohlížečů. Výčet kroků dalšího vývoje aplikace je obsahem následující podkapitoly 8.1.

8.1 Budoucí vývoj

Aplikace v současné podobě není připravena na reálný provoz. Postup vedoucí k úspěšnému nasazení aplikace do praxe bez omezení přístupu obsahuje tyto kroky.

8.1.1 Podpora internetových prohlížečů

Aplikace v současné verzi plně podporuje pouze prohlížeče Google Chome a Mozilla Firefox. Rozšíření podpory aplikace pro prohlížeče Internet explorer (verze 11 a vyšší), Microsoft Edge a Safari je pro nasazení aplikace do reálného provozu stěžejní. Stejně tak je nutné rozšířit i podporu mobilních prohlížečů.

8.1.2 Lokalizace

Klientská aplikace je lokalizována pouze v českém jazyce. Rozšíření jazykové podpory alespoň pro angličtinu je nezbytné.

8.1.3 Implementace kalendáře

Je třeba doplnit chybějící kalendář s daty vysílání sledovaných seriálů, který nebyl v rámci práce z časových důvodů implementovaný.

8.1.4 Optimalizace rychlosti

API aplikace bylo navrženo dle zásad RESTfull rozhraní, jejichž striktní dodržení má v určitých případech dopad na rychlost načítání dat ze serveru. Rozhraní lze upravit pro efektivnější přenos dat.

Literatura

[1] IDEO. *Design zaměřený na člověka: soubor nástrojů.* Brno: Flow, 2011.

[2] Franc, Jakub *Psychologie v HCI* České Vysoké Učení Technické v Praze, 2015.

[3] Miovský, Michal *Kvalitativní přístup a metody v psychologickém výzkumu.* Praha: Grada, 2006.

[4] ISO/DIS 9241-11.2, `http://www.iso.org/iso/catalogue_detail.htm?csnumber=63500` [Online; navštíveno 16. 12. 2016]

[5] Workflow, `https://cs.wikipedia.org/wiki/Workflow` [Online; navštíveno 7. 1. 2017]

[6] Míkovec, Zdeněk *Návrh uživatelského rozhraní* České Vysoké Učení Technické v Praze, 2015. `http://nur.felk.cvut.cz/lecture-design-studio-sketching/` [Online; navštíveno 7. 1. 2017]

[7] Balsamiq `https://balsamiq.com` [Online; navštíveno 17. 12. 2016]

[8] Google Trends, `https://www.google.cz/trends/explore#q=ember.js%2C%20angular.js%2C%20react.js&cmpt=q&tz=Etc%2FGMT-2` [Online; navštíveno 15. 1. 2016]

[9] Github of Ember.js project, `https://github.com/emberjs/ember.js/graphs/commit-activity` [Online; navštíveno 15. 1. 2016]

[10] Github of React.js project, `https://github.com/facebook/react/graphs/commit-activity` [Online; navštíveno 15. 1. 2016]

[11] Facebook Flux Overview, `https://facebook.github.io/fslux/docs/overview.html` [Online; navštíveno 2. 1. 2017]

[12] JetBrains Webstorm `https://www.jetbrains.com/webstorm/` [Online; navštíveno 27. 12. 2016]

[13] NPM `https://www.npmjs.com/` [Online; navštíveno 3. 1. 2017]

[14] Node.js `https://nodejs.org/en/` [Online; navštíveno 27. 12. 2016]

[15] Flux https://www.npmjs.com/package/flux/ [Online; navštíveno 27. 12. 2016]

[16] React-Bootstrap https://react-bootstrap.github.io/ [Online; navštíveno 3. 1. 2016]

[17] React Router https://github.com/ReactTraining/react-router/ [Online; navštíveno 2. 1. 2017]

[18] jQuery https://jquery.com/ [Online; navštíveno 2. 1. 2016]

[19] Async http://caolan.github.io/async/ [Online; navštíveno 27. 12. 2016]

[20] Browserify https://github.com/substack/node-browserify/ [Online; navštíveno 27. 12. 2016]

[21] Watchify https://github.com/substack/watchify/ [Online; navštíveno 27. 12. 2016]

[22] Gulp http://gulpjs.com/ [Online; navštíveno 27. 12. 2016]

[23] gulp-babel https://github.com/babel/gulp-babel/ [Online; navštíveno 27. 12. 2016]

[24] Babel https://babeljs.io/ [Online; navštíveno 27. 12. 2016]

[25] gulp-browserify https://www.npmjs.com/package/gulp-browserify/ [Online; navštíveno 27. 12. 2016]

[26] gulp-uglify https://www.npmjs.com/package/gulp-uglify/ [Online; navštíveno 27. 12. 2016]

[27] UglifyJS https://github.com/mishoo/UglifyJS/ [Online; navštíveno 27. 12. 2016]

[28] gulp-react https://www.npmjs.com/package/gulp-react [Online; navštíveno 27. 12. 2016]

[29] Thinking in React https://facebook.github.io/react/docs/thinking-in-react.html [Online; navštíveno 2. 1. 2017]

[30] Events https://nodejs.org/api/events.html [Online; navštíveno 2. 1. 2017]

[31] Apiary API dokumentace http://docs.serialfreak1.apiary.io/# [Online; navštíveno 8. 1. 2017]

[32] Apiary.io https://apiary.io/ [Online; navštíveno 8. 1. 2017]

[33] Ajax http://api.jquery.com/jquery.ajax/ [Online; navštíveno 2. 1. 2017]

[34] Responsive design `http://www.w3schools.com/html/html_responsive.asp` [Online; navštíveno 7. 1. 2017]

[35] Responsive design image `https://cs.wikipedia.org/wiki/Responzivn%C3%AD_web_design` [Online; navštíveno 3. 1. 2017]

[36] Bureš, Miroslav *Základy testování software* České Vysoké Učení Technické v Praze, 2015.

[37] Smoke testy `http://testovanisoftwaru.cz/tag/smoke-testy/` [Online; navštíveno 6. 1. 2017]

[38] Microsoft Azure Web App `https://azure.microsoft.com/cs-cz/services/app-service/web/` [Online; navštíveno 7. 1. 2017]

[39] Microsoft IIS Webserver `https://www.iis.net/` [Online; navštíveno 7. 1. 2017]

Příloha A

Zkratky

UCD - User Centered Design

SPA - Single Page Application

HCI - Human Computer Interaction

UX - User eXperience

UXD - User eXperience Design

Lo-Fi - Low Fidelity

Hi-Fi - High Fidelity

UI - User Interface

DOM - Document Object Model

API - Application Programming Interface

JSX - Javascript Syntax eXtension

XML - Extensible Markup Language

MVC - Model View Controller

IDE - Integrated Development Environment

NPM - Node Package Manager

URL - Uniform Resource Locator

HTML - HyperText Markup Language

CSS - Cascading Style Sheets

REST - Representational State Transfer

IIS - Internet Information Services

Příloha B

Uživatelský průzkum

B.1 Screener

1. Jaký je váš věk? (kritérium: 16-60 let)
   - 0-15
   - 16-25
   - 25-40
   - 40-60
   - 60 a více
2. Jaké je vaše pohlaví? (kritérium: 50% Mužů, 50% žen)
   - Muž
   - Žena
3. Jaký je váš statut? (kritérium: Student, Zaměstnaný, OSVČ)
   - Student
   - Zaměstnaný
   - OSVČ
   - Nezaměstnaný
   - Důchodce
4. Trpíte nějakým postižením, které vás omezuje v práci s počítačem? (kritérium: NE)
   - ANO
   - NE
5. Máte ve vaší domácnosti připojení k internetu? (kritérium: ANO)
   - ANO

- NE

6. Jak často používáte internet (doma, v zaměstnání, kdekoliv)? (kritérium: Alespoň 1x týdně)
   - Vůbec
   - Méně než 1x týdně
   - 1-3x týdně
   - Denně
7. Jste uživatelem nějaké sociální sítě? (kritérium: ANO)
   - ANO
   - NE
8. Sledujete seriály? (kritérium: ANO)
   - ANO
   - NE
9. Sledujete nějaký seriálový titul pravidelně? (kritérium: ANO)
   - ANO
   - NE
10. Jak často sledujete seriály? (kritérium: Alespoň 1x měsíčně)
    - Vůbec
    - Méně než 1x měsíčně
    - 1-2x měsíčně
    - 1x týdně
    - 2-4x týdně
    - Denně
11. Kolik seriálových titulů jste sledovali/sledujete v posledních 3 letech? (kritérium: Alespoň jeden)
    - Žádný
    - 1-3
    - 4-8
    - Více než 8
12. Navštěvujete některý ze seriálových portálů jako jsou například serial-zone.cz či edna.cz ? (kritérium: 50% ANO, 50% NE+NEVÍM)
    - ANO
    - NE

B.2 Session Guide

1. Příprava před samotným rozhovorem
   a. Představit se
   b. Podat participantovi základní informace o rozhovoru
      (i) Anonymita
      (ii) Formát rozhovoru
      (iii) Délka rozhovoru
   c. Dohoda pro pořízení záznamu
   d. Prostor pro otázky participanta
2. Rozhovor
   a. Základní informace o participantovi
      (i) Obor studia, zaměstnání či podnikání
      (ii) Bydlení
      (iii) Rodinný stav
      (iv) Koníčky
   b. Časové možnosti participanta
      (i) Jak vypadá váš běžný den?
      (ii) Kolik volného času máte v pracovním týdnu k dispozici?
      (iii) Jakým způsobem ho využíváte?
      (iv) Kolik volného času máte běžně o víkendu k dispozici?
      (v) Jakým způsobem ho využíváte?
   c. Práce s internetem
      (i) K jakým aktivitám využíváte internet nejčastěji?
      (ii) Jaká zařízení k připojení k internetu používáte?
      (iii) Jaké vnímáte rozdíly v použití různých zařízení?
      (iv) Které sociální sítě používáte?
      (v) Proč je používáte?
      (vi) Jsou pro vás důležité?
      (vii) Jak často se k nim připojujete?
   d. Sledování seriálů
      (i) Co pro vás znamenají seriály?
      (ii) Jaké sledujete seriály?
      (iii) Kde je sledujete?
      (iv) Jakým způsobem (nahodile, organizovaně)?
      (v) Kdy je sledujete?
      (vi) Sledujete seriály sami nebo s někým?

e. Informace o seriálech
   (i) Hledali jste někdy nějaké informace o vámi sledovaném seriálu?
   (ii) Jaký typ informací to byl?
   (iii) Jak jste ve vašem hledání postupovali?
   (iv) Jaký byl výsledek hledání (úspěch x neúspěch + pocity)?
   (v) Proč jste začal/a sledovat vámi aktuálně sledovaný seriál?
   (vi) Jakým způsobem se dozvídáte o nových seriálech, které by vás mohli bavit?

3. Ukončení rozhovoru
   a. Dodatečné postřehy či dotazy participanta
   b. Ujištění participanta, že byl rozhovor pro výzkum přínosný
   c. Předání odměny a rozloučení

Příloha C

Analýza rozhovorů

Poznámky z rozhovorů a jejich následnou analýzu jsem provedl v aplikaci Mirosoft Excel. Rozsah sešitu není vhodný pro tisk, proto ho přikládám jako soubor interview-analysis.xlsx na přiloženém CD v adresáři Analýza rozhovorů.

Příloha D

Testovací prototypy

Testovací prototypy jsou mnohastránková PDF s aktivními prvky, díky kterým lze přecházet mezi jednotlivými stránkami. Celé je přikládám na přiloženém CD v adresáři Prototypy. Soubor Scenario1.pdf pokrývá scénář D.1 a soubor Scenario2.pdf scénář D.2.

D.1 Testovací scénář 1

1. Přihlaste se do aplikace
2. Potvrďte souhlas zpřístupnění informací aplikaci
3. V současné době jste začali sledovat seriál The Big Bang Theory. Přidejte si jej ke sledování a sdílejte tento fakt na sociální síť facebook.
4. Seriál chcete sledovat popořadě v časovém sledu, jak byl vysílaný v televizi. Poslední vámi viděná epizoda je dvanáctá z první série.
5. Z popisu epizody zjišťujete, že jste se spletli a viděli jste i 13. epizodu. Epizoda se vám líbila a chcete tento fakt sdílet na svou zeď na Facebooku.
6. Čtrnáctou epizodu jste již neviděli a rádi byste si ji nyní pustili.

D.2 Testovací scénář 2

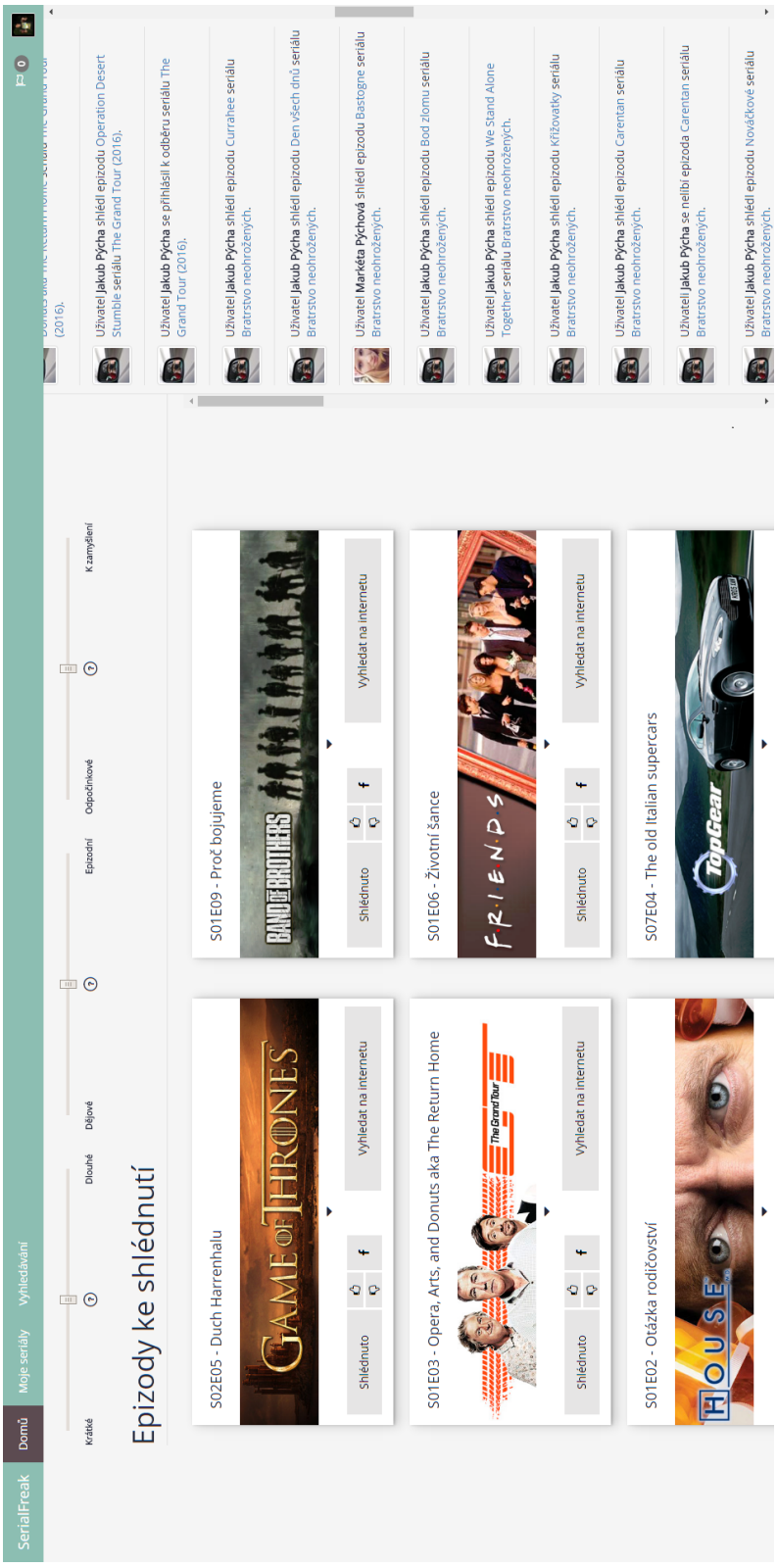
1. Jste přihlášeni v aplikaci a máte v odběru větší množství seriálů. Jedním z nich je Sherlock, který jste označili za epizodní seriál k zamyšlení. Zadejte oba parametry do filtrování pro jeho nalezení.
2. Otevřete stránku tohoto seriálu a změňte úvodní obrázek na jiný.
3. Vyhledejte epizodu S02E01 - Skandál v Belgrávii a označte ji jako neshlédnutou.
4. Obdrželi jste notifikaci na událost v aplikaci. Zkontrolujte, o co se jedná a proveďte akce z toho plynoucí.

5. Seriál Dva a půl chlapa jste již viděli a máte shlédnuté všechny epizody. Přidejte si seriál k odběru.
6. Vraťte se na domovskou stránku a zkontrolujte, zda je vám nyní seriál ke sledování.
7. V současné době máte přihlášen odběr seriálu Hra o trůny. Seriál již nechcete dále sledovat. Zrušte jeho odběr.
8. Ohodnoťte pozitivně seriál Simpsonovi. (Použijte abecední filtrování)
9. Vyhledejte informace o seriálu IT Crowd. Seriál nyní nesledujete.
10. Odhlašte se z aplikace.

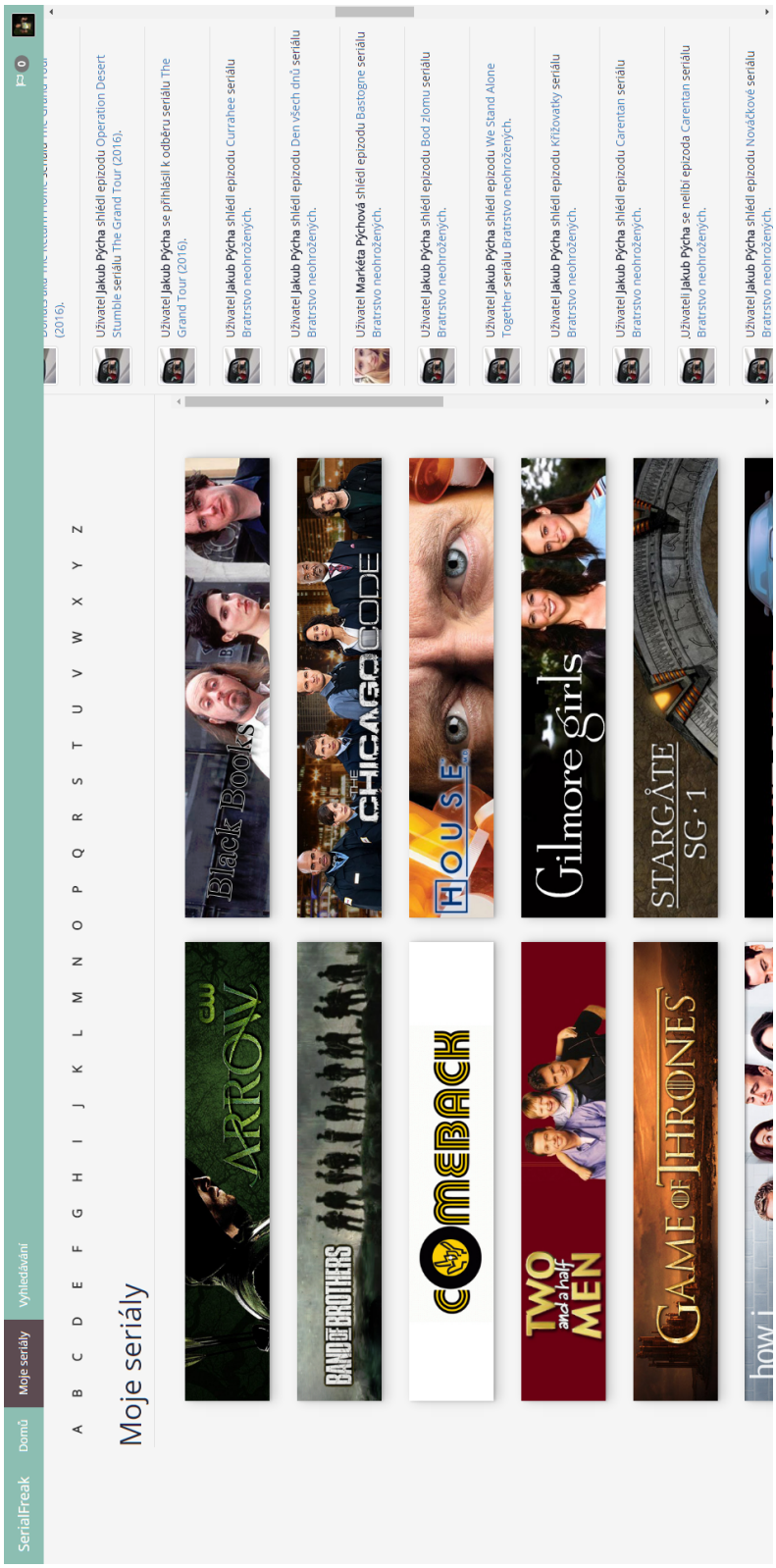
Příloha E

Snímky obrazovky

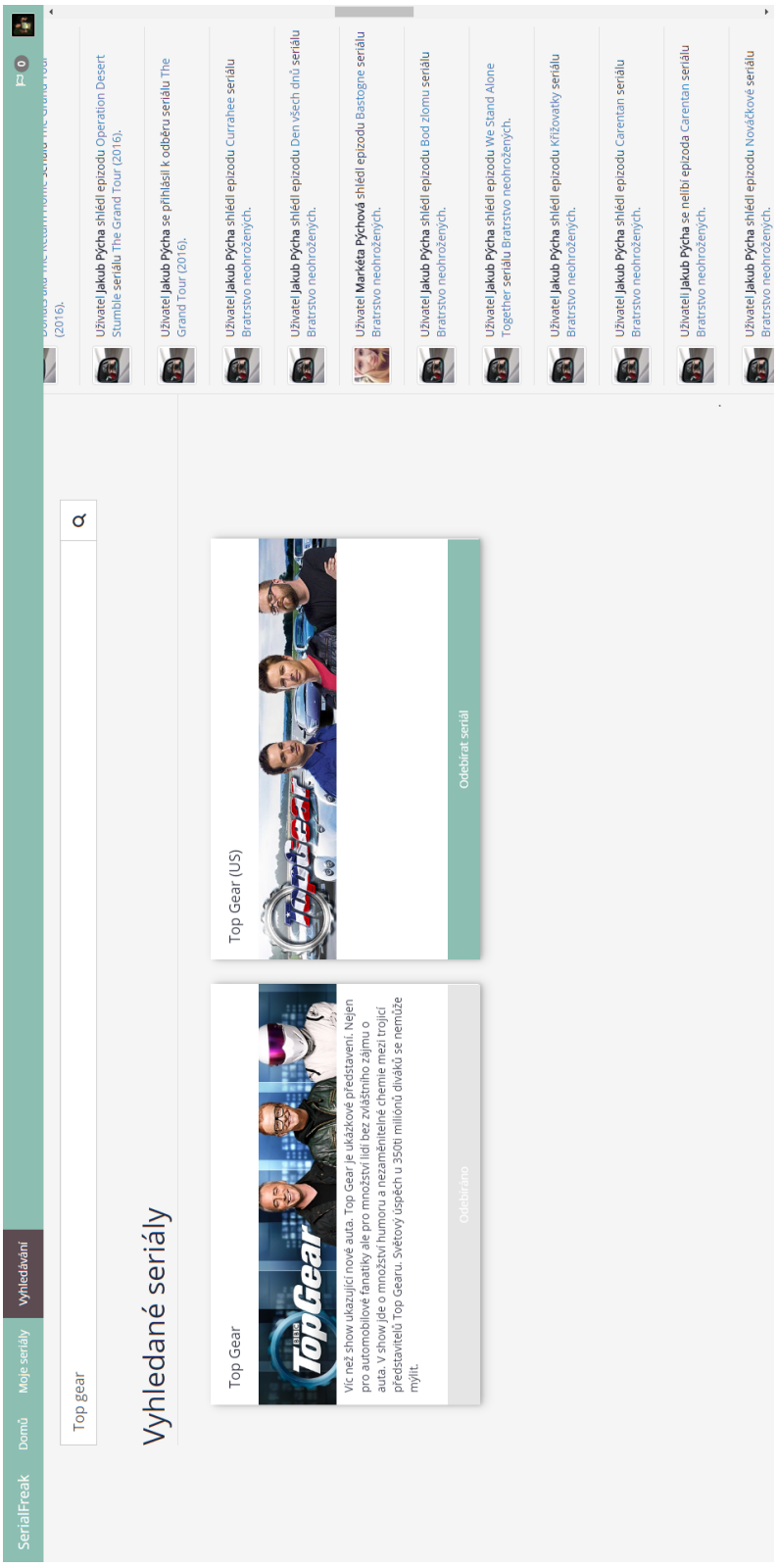
Následující obrázky zobrazují vzhled uživatelského rozhraní na hlavních čtyřech obrazovkách aplikace. Ukazují pouze základní vzhled aplikace bez ukázky funkcí interaktivních prvků jako jsou skryté rozklikávací panely s textem, rolovací seznamy sezón, epizod a jejich detailů na stránce seriálu apod. V první sekci je znázorněn vzhled aplikace na velké obrazovce, v druhé pak na mobilním zařízení.



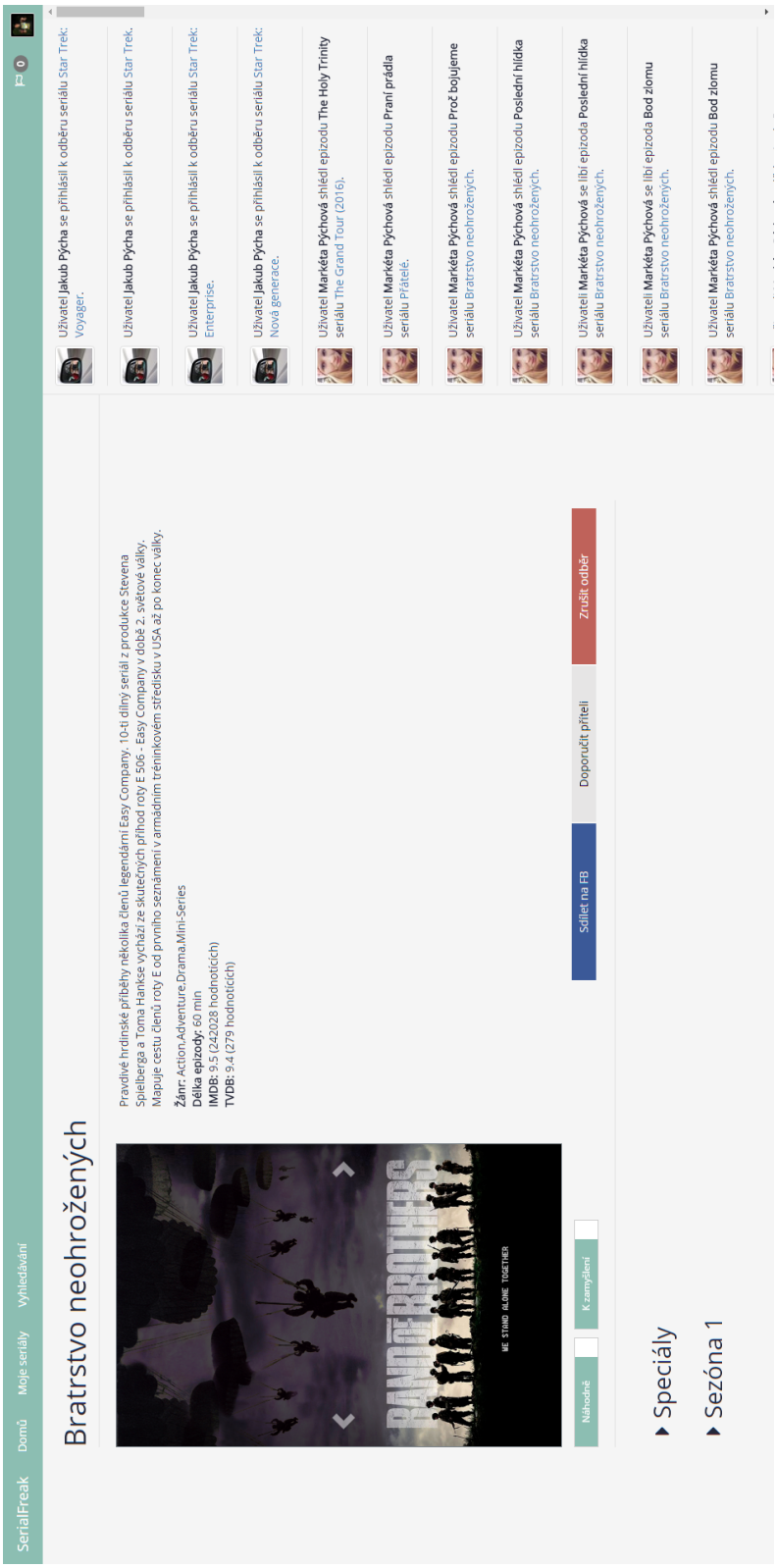
Obrázek E.1: Domovská obrazovka.



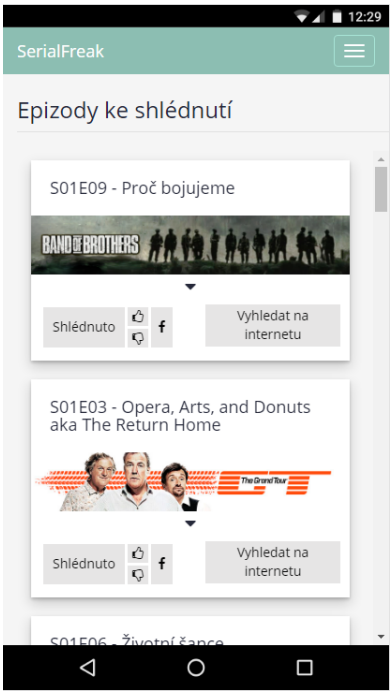
Obrázek E.2: Moje Seriály.



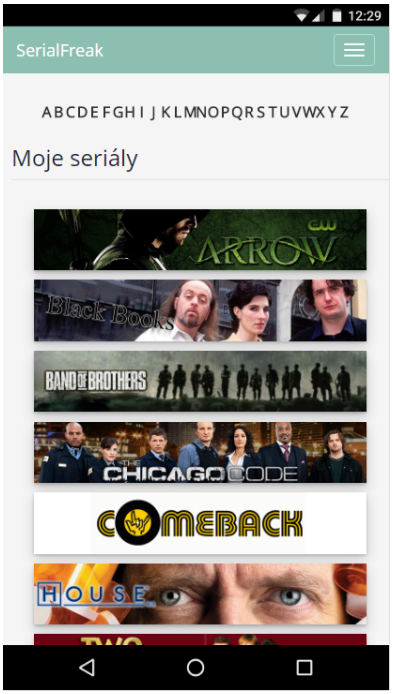
Obrázek E.3: Vyhledávání.



Obrázek E.4: Stránka seriálu.



(a) : Domovská obrazovka.



(b) : Moje seriály.

(c) : Vyhledávání.

(d) : Stránka seriálu.

Obrázek E.5: Mobilní verze aplikace.

Příloha F

Testovací scénáře aplikace

F.1 Testovací scénář: Přihlášení do aplikace

ID pokrytých funkčních požadavků: 1

ID testovacího scénáře: TS001

Popis: Test přihlášení uživatele do aplikace. Scénář pokrývá i různé stavy přihlášení uživatele do služby Facebook v prohlížeči.

Počet testovacích případů: 3

F.1.1 Testovací případ: Uživatel nepřihlášen na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC001

Předpoklady: Validní přihlašovací údaje ke službě Facebook. Uživatel není v prohlížeči přihlášen ke službě Facebook.

Popis: Test přihlášení uživatele do aplikace za stavu, kdy není v prohlížeči přihlášen do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook
2	Klik na tlačítko Přihlásit se přes Facebook	Zobrazen formulář pro zadání přihlašovacích údajů do služby Facebook
3	Zadat validní přihlašovací údaje a kliknout na tlačítko login	Zobrazena Domovská obrazovka aplikace

F.1.2 Testovací případ: Uživatel je přihlášen na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC002

Předpoklady: Uživatel je v prohlížeči přihlášen ke službě Facebook.

Popis: Test přihlášení uživatele do aplikace za stavu, kdy je v prohlížeči přihlášen do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Domovská obrazovka aplikace

F.1.3 Testovací případ: Změna stavu přihlášení na Facebook

ID testovacího scénáře: TS001

ID testovacího případu: TC003

Předpoklady: Validní přihlašovací údaje ke službě Facebook. Uživatel není v prohlížeči přihlášen ke službě Facebook.

Popis: Test změny stavu přihlášení v aplikaci při změně stavu přihlášení do služby Facebook.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Zadat URL aplikace do adresy prohlížeče	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook
2	Přihlásit se v nové záložce prohlížeče do služby Facebook	Uživatel přihlášen do služby Facebook
3	V původní záložce s aplikací SerialFreak provést znovunačtení stránky (F5)	Zobrazena Domovská obrazovka
4	V druhé záložce se stránkou Facebook provést odhlášení	Uživatel odhlášen ze služby Facebook
5	V záložce s aplikací SerialFreak provést znovunačtení stránky (F5)	Zobrazena Přihlašovací obrazovka s tlačítkem Přihlásit se přes Facebook

F.2 Testovací scénář: Zařazení seriálu mezi sledované

ID pokrytých funkčních požadavků: 2,3,4,5,9

ID testovacího scénáře: TS002

Popis: Test vyhledání a přidání seriálu mezi sledované v jehož průběhu je ověřeno i zobrazení doporučených seriálů ke sledování.

Počet testovacích případů: 2

F.2.1 Testovací případ: Přidání seriálu ke sledování z obrazovky Vyhledávání

ID testovacího scénáře: TS002

ID testovacího případu: TC004

Předpoklady: Uživatel přihlášený do aplikace. Zobrazena Domovkská obrazovka aplikace.

Popis: Test přidání seriálu ke sledování provedeném pomocí dláždice vyhledaného seriálu na stránce Vyhledávání.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Klik na tlačítko Vyhledávání v liště Menu	Zobrazeno textové pole se zástupným textem: Zadejte název seriálu pod nímž se nachází sekce Doporučené seriály. Sekce doporučené seriály obsahuje dláždice se seriály Arrow, The Grand Tour a Top Gear
2	Zadat do textového pole: Jak jsem poznal a klik na ikonu lupy	Zobrazena dláždice se seriálem Jak jsem poznal vaši matku. Dláždice obsahuje název seriálu, obrázek, popis a tlačítko Odebírat.
3	Klik na tlačítko Odebírat	Dláždice je otočena a na její zadní straně se nachází formulář s kategorizací seriálu, výběrem zhlédnutých epizod a tlačítky Odebírat seriál a Zrušit
4	Vyplnit formulář (Sledování: Pořadě, Typ seriálu: Odpočinkový, Viděl jsem: Dopodud jsem neviděl žádnou epizodu)	Formulář vyplněn dle zadání
5	Klik na tlačítko Odebírat seriál	Dláždice otočena zpět, tlačítko odběru je neaktivní, šedé s textem Odebíráno
6	Klik na tlačítko Domů v liště Menu	Zobrazena Domovská obrazovka, na které je dláždice s epizodou S01E01 - Pilot seriálu Jak jsem poznal vaši matku

F.2.2 Testovací případ: Přidání seriálu ke sledování z obrazovky Detailu seriálu

ID testovacího scénáře: TS002

ID testovacího případu: TC005

Předpoklady: Uživatel přihlášený do aplikace. Zobrazena Domovkská obrazovka aplikace.

Popis: Test přidání seriálu ke sledování provedeném na stránce Detail seriálu. Test zároveň ověřuje existenci agregovaných dat u seriálu.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Klik na tlačítko Vyhledávání v liště Menu	Zobrazeno textové pole se zástupným textem: Zadejte název seriálu pod nímž se nachází sekce Doporučené seriály. Sekce doporučené seriály obsahuje dláždice se seriály Arrow, The Grand Tour a Top Gear
2	Zadat do textového pole: Jak jsem poznal a klik na ikonu lupy	Zobrazena dláždice se seriálem Jak jsem poznal vaši matku. Dláždice obsahuje název seriálu, obrázek, popis a tlačítko Odebírat.
3	Klik na obrázek seriálu	Zobrazena obrazovka Detailu seriálu Jak jsem poznal vaši matku. Obrazovka obsahuje obrázek seriálu, popis, Žánr, Délku epizody, hodnocení IMDB a TVDB, tlačítko Odebírat seriál a seznam sezón seriálu
4	Klik na odebírat seriál	Zobrazen formulář s kategorizací seriálu, výběrem zhlédnutých epizod a tlačítky Odebírat seriál a Zrušit
5	Vyplnit formulář (Sledování: Popořadě, Typ seriálu: Odpočinkový, Viděl jsem: Dopodud jsem neviděl žádnou epizodu)	Formulář vyplněn dle zadání
6	Klik na tlačítko Odebírat seriál	Formulář je uzavřen a na stránce seriálu se nyní nachází ovládací prvky kategorizace seriálu a tlačítka Sdílet na FB, Doporučit příteli a Zrušit odběr
7	Klik na tlačítko Domů v liště Menu	Zobrazena Domovská obrazovka, na které je dláždice s epizodou S01E01 - Pilot seriálu Jak jsem poznal vaši matku

F.3 Testovací scénář: Akce uživatele na stránce sledovaného seriálu

ID pokrytých funkčních požadavků: 6,7,8,10,13

ID testovacího scénáře: TS003

Popis: Test ovládacích prvků na stránce sledovaného seriálu.

Počet testovacích případů: 2

F.3.1 Testovací případ: Doporučení seriálu příteli a sdílení seriálu na FB

ID testovacího scénáře: TS003

ID testovacího případu: TC006

Předpoklady: Uživatel přihlášený do aplikace. Zobrazen detail seriálu Jak jsem poznal vaši matku, který má uživatel ve sledování.

Popis: Test ověřující možnost uživatele doporučit seriál příteli a sdílet seriál na svůj facebookový profil.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Klik na tlačítko Doporučit příteli	Zobrazen formulář s jedním vylistovaným uživatelem (Testovací uživatel)
2	Klik na Testovacího uživatele	Formulář je uzavřen
3	Klik na tlačítko Sdílet na FB	Zobrazen facebookový formulář obsahující název seriálu Jak jsem poznal vaši matku s jeho popisem a zdrojem SERIALFREAK umožňující připsání vlastního textu a zveřejnění na Facebook
4	Připsat text libovolného znění a zvolit ve volbě soukromí Jenom já	Formulář je vyplněn
5	Klik na tlačítko Zveřejnit na Facebooku	Formulář je uzavřen
6	Navštívit facebookový profil přihlášeného uživatele	Příspěvek je přítomen na profilu přihlášeného uživatele

F.3.2 Testovací případ: Práce s epizodami sledovaného seriálu

ID testovacího scénáře: TS003

ID testovacího případu: TC007

Předpoklady: Uživatel přihlášený do aplikace. Zobrazen detail seriálu Jak jsem poznal vaši matku, který má uživatel ve sledování.

Popis: Test ověřující možnost změny stavu zhlédnutí epizod a jejich hodnocení v seznamu sezón a epizod.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Klik na šipku u Sezóny 1	Vylistovány epizody 1. sezóny seriálu (1. Pilot, 2. Fialová žirafa, atd.) a u každé je zobrazena ikona palce nahoru, palce dolů a oka
2	Klik na šipku u Sezóny 1	List epizod je skryt
3	Klik na text Sezóna 2	Vylistovány epizody 2. sezóny seriálu (1. Kde jsme to byli?, 2. Škorpion a žába, atd.) a u každé je zobrazena ikona palce nahoru, palce dolů a oka
4	Klik na ikonu palce nahoru u 1. epizody 2. sezóny	Ikona je zelená, Ikona palce dolů je šedá
5	Klik na ikonu palce dolů u 1. epizody 2. sezóny	Ikona je červená, Ikona palce dolů je šedá
6	Klik na ikonu palce nahoru u 1. epizody 2. sezóny	Ikona je zelená, Ikona palce dolů je šedá
7	Klik na ikonu oka u 1. epizody 2. sezóny	Ikona je zelená
6	Klik na ikonu oka u 1. epizody 2. sezóny	Ikona je červená
8	Klik na 1. epizodu 2. sezóny	Zobrazen detail epizody, kde se nachází obrázek epizody, popis a tlačítko Vyhledat na Internetu
9	Klik na tlačítko Vyhledat na internetu	Otevřena nová záložka s vyhledávacím dotazem Jak jsem poznal vaši matku S02E01

F.4 Testovací scénář: Navigace v aplikaci

ID pokrytých funkčních požadavků: 1,2,5,9,12

ID testovacího scénáře: TS004

Popis: Test zobrazení hlavních obrazovek a komponent aplikace.

Počet testovacích případů: 1

F.4.1 Testovací případ: Průchod aplikací

ID testovacího scénáře: TS004

ID testovacího případu: TC008

Předpoklady: Validní prihlašovací údaje ke službe Facebook. Uživatel je v prohlížeci přihlášen ke službe Facebook. Zobrazena Přihlašovací obrazovka.

Popis: Test přihlášení do aplikace a zobrazení všech hlavních částí UI s ověřením jeho obsahu. Test odhlášení.

Krok	Popis kroku testu	Očekávaný výsledek kroku testu
1	Klik na tlačítko Přihlásit přes Facebook	Vlevo zobrazena Domovská obrazovka s dláždicemi doporučených epizod. Vpravo zobrazen panel událostí přátel. URL: /homescreen
2	Klik na tlačítko Moje Seriály v menu	Levá strana obrazovky překrelena seznamem sledovaných seriálů. URL: /myserials
3	Klik na tlačítko Vyhledávání v menu	Levá strana obrazovky překrelena na stránku s vyhledávacím polem a sekcí Doporučené seriály. URL: /search
4	Klik na tlačítko Domů v menu	Levá strana obrazovky překrelena na Domovskou obrazovku s dláždicemi doporučených epizod. URL: /homescreen
5	Klik na obrázek některé z dláždic	Zobrazena obrazovka Detailu seriálu URL: /serial/:id, kde :id je číselné ID seriálu.
6	Klik na tlačítko zpět v prohlížeči	Zobrazena Domovská obrazovka
7	Klik na profilový obrázek v pravé části menu	Zobrazen detail profilu s tlačítkem Odhlásit
8	Klik na tlačítko Odhlásit	Zobrazena přihlašovací obrazovka. URL: /login
9	Znovunačíst stránku v prohlížeči (F5)	Zobrazena přihlašovací obrazovka

Příloha G

Obsah přiloženého CD

```
/
├── Analýza rozhovorů
│   └── interview-analysis.xlsx
├── Distribuční verze aplikace
├── DP-LATEX ..... Text diplomové práce ve formátu LATEX
├── Prototypy
│   ├── Soubory aplikace Balsamiq ..... Zdrojové soubory prototypů
│   ├── Scenario1.pdf
│   └── Scenario2.pdf
├── UML modely
│   └── serialfreak.eap
├── Zdrojové kódy aplikace
└── DP-dyrcifil-2017.pdf ..... Text diplomové práce ve formátu PDF
```