Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratwislava 4

Projektová dokumentácia – The Reconnected

Karol Balko, Michal Dorner, Martin Konôpka, Marek Láni, Martin Lipták,
Andrea Šteňová, Matúš Tomlein

Tím č. 12:	The Reconnected
Vedúci tímu:	Ing. Karol Rástočný
Predmet:	Tímový projekt I
Študijný program:	Softvérové inžinierstvo, Ročník: 1.
Akademický rok:	2012/2013, zimný semester
Kontakt:	timak12@googlegroups.com

Obsah

Úvod **i**

Slovník pojmov **ii**

1 Úlohy pred začatím šprintov **1.1**
1.1 Stretnutie k nasadeniu projektu vo Voi . . . 1.1
1.2 Pripravenie inštalačných súborov a manuálov pre Voi . . . 1.1
1.3 Vytvorenie TCP servera a počúvanie na requesty . . . 1.1
1.4 Sťahovanie stránok z Webu . . . 1.1
1.5 Prístup na statické súbory na disku . . . 1.1
1.6 Dynamické vytváranie obsahu v proxy . . . 1.1
1.7 Inštalácia virtuálneho servera pre potreby projektu . . . 1.2
1.8 Vytvorenie webovej stránky ku projektu . . . 1.2
1.9 Prieskum frameworkov pre Javascript . . . 1.2
1.10 Prieskum podporných nástrojov pre vývoj portálu . . . 1.2
1.11 Analýza doterajšieho projektu . . . 1.2

2 Šprint Francis Kere **2.1**
2.1 Cache stránok . . . 2.1
2.2 Základy prednačítavania . . . 2.3
2.3 Možnosti znovupoužitia pôvodného riešenia . . . 2.6
2.4 Analýza používateľských skupín . . . 2.7
2.5 Qt Zbernica . . . 2.8
2.6 Vloženie Javascript kódu do stránok v proxy . . . 2.9
2.7 Vytvorenie kostry portálu v Backbone.js . . . 2.11
2.8 Analýza a návrh funkcionality portálu . . . 2.13
2.9 Databáza . . . 2.18
2.10 Francis Kere - Pomocné úlohy . . . 2.19

3 Šprint Moi Equator Girls **3.1**
3.1 Vymazávanie cache . . . 3.1
3.2 Prednačítavanie stránok . . . 3.2
3.3 Vytvorenie URL pomocníka na spracovanie HTTP požiadaviek . . . 3.5
3.4 Sledovanie vyťaženia linky . . . 3.7
3.5 Vytvorenie používateľského modulu . . . 3.8
3.6 Vytváranie sedení . . . 3.10
3.7 Registrácia a prihlasovanie používateľov - WEB Gui . . . 3.11
3.8 Analýza možností synchronizácie a čiastočná implementácia . . . 3.12
3.9 Analýza možností komunikácie a voľby servera . . . 3.14
3.10 Zabezpečenie DB . . . 3.15
3.11 Analýza skupín a čiastočný návrh . . . 3.16
3.12 Moi Equator Girls - Pomocné úlohy . . . 3.19

4 Šprint Voi **4.1**
4.1 Ladenie proxy . . . 4.1
4.2 Jednotkové testovanie . . . 4.4
4.3 Integračné testovanie . . . 4.6
4.4 Implementácia komunikácie a voľby servera . . . 4.9
4.5 História prehliadania . . . 4.11
4.6 Databáza pre rozhovory medzi používateľmi . . . 4.13
4.7 Implementácia GUI časti skupín . . . 4.14
4.8 Vytvorenie služby pre používateľské skupiny v Qt . . . 4.15
4.9 Vytvorenie služby pre používateľské skupiny v Qt . . . 4.20
4.10 Vytvorenie služby pre hodnotenia stránok Qt . . . 4.22

5 Šprint Xela **5.1**
5.1 Dokončenie synchronizácie . . . 5.1
5.2 Distribuovaná cache . . . 5.3
5.3 Prechod na Qt5 . . . 5.5
5.4 Router pre moduly . . . 5.7
5.5 Dokončenie testovania . . . 5.8
5.6 Pokračovanie v implementácii GUI časti používateľských skupín . . . 5.9
5.7 Dokončenie manažmentu používateľov . . . 5.10
5.8 Dokončenie služby pre skupiny . . . 5.11
5.9 Pomocné úlohy pre šprint Xela . . . 5.12
5.10 Prednačítavanie odkazov . . . 5.13

6 Šprint Krismasi **6.1**
6.1 Oprava zistených chýb . . . 6.1
6.2 Grafický návrh obrazoviek portálu . . . 6.2

7 Šprint Happy New Year **7.1**
7.1 Výber odkazov na prednačítanie . . . 7.1
7.2 Implementácia GUI časti hodnotenia stránok . . . 7.2
7.3 Implementácia GUI časti odporúčania stránok . . . 7.3
7.4 Implementácia GUI časti profilu používateľa . . . 7.4
7.5 Zlúčenie repozitárov proxy, portálu a testov do jedného spoločného . . . 7.5
7.6 Nový jednoduchší multiplatformový build systém pre portál . . . 7.6
7.7 Filtrovanie zoznamu skupín v Qt . . . 7.7
7.8 Dokončenie skupín portálu . . . 7.8
7.9 Správy medzi používateľmi v skupine . . . 7.10
7.10 Vytvorenie aktivít v Qt . . . 7.11

8 Šprint Hurry up! **8.1**
8.1 Inštalácia aplikácie . . . 8.1
8.2 Vrátenie funkcií prednačitavania a histórie do vkladaného skriptu . . . 8.2
8.3 Zvýrazňovanie odkazov stránok dostupných offline . . . 8.3
8.4 Stránkovanie zoznamov objektov . . . 8.4
8.5 Implementovanie REST centrálnych služieb a komunikácie s nimi . . . 8.5
8.6 Návrh a implementácia . . . 8.5
8.7 Synchronizácia s centrálnymi službami . . . 8.6

8.8 Aktualizácia cache . . . 8.8
8.9 Vylepšenie jadra proxy na spracovanie HTTP dopytov . . . 8.9
8.10 Načítanie stránky z internetu namiesto cache podľa hlavičiek v požiadavke . . . 8.10
8.11 Nahradenie injectu v Proxy cez Extension do prehliadača Chrome . . . 8.11
8.12 Hodnotenie a odporúčanie stránok - IFrame . . . 8.12
8.13 Stránkovanie portálu . . . 8.13
8.14 Štýlovanie správ portálu . . . 8.14

9 Šprint We Are Scientists! 9.1
9.1 Pridanie stiahnutých stránok do profilu . . . 9.1
9.2 Služby na reportovanie pádov a sprístupnenie súborov na aktualizáciu aplikácie . . . 9.2
9.3 Aktualizácia aplikácie . . . 9.3
9.4 Zakázanie a povolenie ukladania stránok . . . 9.5
9.5 Oprava objednávania a zvýrazňovania odkazov . . . 9.6
9.6 Zoznam objednávok na stiahnutie . . . 9.7
9.7 Zmena posielaných dát pri index akcii v službe správ . . . 9.8
9.8 Správa ako aktivita . . . 9.10
9.9 Hlavná stránka s aktivitami používateľov . . . 9.11
9.10 Štýlovanie zvyšných obrazoviek portálu . . . 9.12

10 Šprint Rebirth 10.1
10.1 Aktualizácia aplikácie 2.0 . . . 10.1
10.2 Častejšie mazanie odkazov na prednačítanie . . . 10.3
10.3 Opravenie zvýrazňovania odkazov . . . 10.4
10.4 Obnovovanie aktivít . . . 10.5
10.5 Nekonečné skrolovanie aktivít . . . 10.6
10.6 Backend na zdieľanie súborov . . . 10.6
10.7 Vytvorenie zdieľania súborov v portály . . . 10.8
10.8 Zmena poľa content v aktivitách . . . 10.9
10.9 Nahradenie hodnotení aktivitami . . . 10.10
10.10 Nahradenie odporúčaní aktivitami . . . 10.11

11 Šprint Live Long and Prosper 11.1
11.1 Inštalácia aplikácie . . . 11.1
11.2 Ladenie proxy . . . 11.1
11.3 Offline stránka - GUI . . . 11.3
11.4 Upravenie zdieľania súborov v portály . . . 11.4
11.5 Nahradenie správ aktivitami + komentáre pre aktivity . . . 11.5
11.6 Manuálne prednačítavanie . . . 11.6

12 Technický opis produktu 12.1
12.1 Architektúra prototypu . . . 12.1
12.2 Databázový model . . . 12.2

13 Opis aktuálnej verzie 13.1
13.1 Opis aktuálnej verzie . . . 13.1

Úvod

Tento dokument vznikol počas práce na tímovom projekte tímu číslo 12 v akademickom roku 2012/2013 na Fakulte informatiky a informačných technológií. Jeho obsahom je dokumentácia k vyvíjanému softvéru. Obsahuje zodpovednosti členov tímu, ako aj úlohy za ktoré boli zodpovední.

Motivácia

V dnešnej dobe, keď sa internet stal neoddeliteľnou súčasťou našich životov, sa na svete stále nájdu ľudia, ktorým táto moderná technológia nie je dostupná, a tak sú znevýhodnení oproti ostatnej časti populácie. Tomuto problému čelia hlavne krajiny 3. sveta, pre ktorých rozvoj je získanie pohodlnejšieho prístupu k informáciám prostredníctvom internetu priam nevyhnutný. Web ako nekonečný zdroj informácií a vzdelania môže pomôcť týmto krajinám začať napredovať, zlepšiť rozvoj mladej generácie, a tak pomôcť riešiť aj problémy, ktoré sa zdajú byť informačnými technológiami neriešiteľné. Časovú medzeru čakania týchto miest na kvalitné pripojenie môže vyplniť softvérové riešenie, ktoré zlepší podmienky prehliadania Webu s využitím aj aktuálneho nekvalitného pripojenia.

Zadanie

V rámci súťaže Imagine Cup 2012 navrhol a vytvoril tím z našej fakulty systém OwNet, ktorý pre každého používateľa vytvára jeho vlastnú mini-kópiu webu a umožňuje jeho plnohodnotné prehliadanie offline. Tento systém sa podarilo nasadiť do viacero afrických škôl.

Témou tohto tímového projektu je podpora a ďalší rozvoj (nielen evolučný, ale aj revolučný) OwNetu. Cieľom je analyzovať existujúce riešenie a pokračovať v jeho rozvíjaní napr. nasledovnými smermi:

- vývoj klienta a servera pre platformu GNU/Linux
- vývoj nových, resp. vylepšenie existujúcich algoritmov prednačítavania obsahu
- zrealizovanie plnohodnotného intranetového portálu v rámci systému OwNet
- zavedenie podpory e-Learningu a zdieľania študijných materiálov v rámci OwNetu

Ciele produktu

Keďže vytvorené riešenie je implementované vo frameworku .NET, je ho možné spustiť len v prostredí Windows. Ak však chceme rozšíriť používanosť našej aplikácie do viacerých škôl, musíme zabezpečiť nezávislosť riešenia od používateľovho operačného systému. Ako ciele na tento tímový projekt sme si preto určili:

- Vytvoriť multiplatformové riešenie, ktoré bude rýchlejšie ako to doterajšie.
- Refaktorizáciu existujúcich funkcií a ich zlepšenie.
- Vylepšenie existujúceho portálu pre používateľov.
- Doplnenie funkcie elektronického vzdelávania do portálu.
- Zjednodušenie inštalácie používateľom.

Slovník pojmov

našej aplikácie do viacerých škôl, musíme zabezpečiť nezávislosť riešenia od používateľovho operačného systému. Ako ciele na tento tímový projekt sme si preto určili:

- *Qt projekt* - Hlavný adresár so všetkým kódom, ktorý patrí aplikácii. Obsahuje súbor .pro so zoznamom súborov so zdrojovým kódom a ďalšie nastavenia.
- *Qt podprojekt* - Qt projekt môže byť rozdelený na viac podprojektov alebo podadresárov (subdirectories) v terminológii Qt. Cieľ podprojektu môže byť spustiteľný súbor alebo knižnica.
- *Používateľské testovanie* - Simulácia práce používateľa v aplikácii pomocou automatizovaných nástrojov.
- *DSL* - Domain-specific language. Jazyk obyčajne tvorený kódom v programovacom jazyku, kedy príkazy majú sémantiku v rámci určitej domény.
- *Cache* - vyrovnávacia pamäť webových stránok v proxy aplikácii
- *JSON* - JavaScript Object Notation, štandard pre formát textových správ

1 Úlohy pred začatím šprintov

1.1 Stretnutie k nasadeniu projektu vo Voi

zodpovedné osoby: Martin Konôpka, Marek Láni, Matúš Tomlein

Organizácia Pontis prejavila záujem o náš projekt a navrhla ho nasadiť na 5 školách v oblastiach Voi v Keni. Absolvovali sme stretnutie 9.10.2012 (v zložení zodpovedných osôb spolu s Michalom Barlom) s p. Jakubom Šimekom, predstavili sme mu stav pôvodnej verzie projektu (september 2012). Dohodli sme sa na spôsobe inštalácie v školách s tým, že mu doručíme potrebné súbory, ktoré posunie ďalej IT tímu v Keni.

1.2 Pripravenie inštalačných súborov a manuálov pre Voi

zodpovedné osoby: Martin Konôpka, Matúš Tomlein

Pôvodná verzia projektu si vyžadovala pár úprav pred nasadením. Zároveň, naše posledné inštalačné súbory boli neaktuálne. Matúš Tomlein pripravil inštalačné súbory. Martin Konôpka sa venoval revidovaniu manuálu, ktorý sme prikladali k inštalačným materiálom (od júla 2012 bol neaktualizovaný). Inštalačné súbory s manuálom sme odoslali elektronicky p. Jakubovi Šimekovi dňa 12.10.2012.

1.3 Vytvorenie TCP servera a počúvanie na requesty

#OWNNET-15, zodpovedná osoba: Matúš Tomlein

Vytvorenie skúšobnej verzie TCP servera v C++ a Qt na odchytávanie a spracovanie požiadaviek z webového prehliadača.

Táto úloha zahŕňa aj analýzu možností Qt v porovnaní s predchádzajúcim riešením v .NET.

1.4 Sťahovanie stránok z Webu

#OWNNET-12, zodpovedná osoba: Matúš Tomlein

Sťahovanie dopytov odchytených na TCP serveri na webové stránky a vrátenie stiahnutých dát prehliadaču (na socket). Je to implementácia základnej funkcionality proxy, berie sa ohľad na efektívnu implementáciu.

1.5 Prístup na statické súbory na disku

#OWNNET-13, zodpovedná osoba: Matúš Tomlein

Umožnenie prístupu na súbory uložené v špeciálnom adresári prostredníctvom proxy. Statické súbory sú prístupné na doméne ownet.

1.6 Dynamické vytváranie obsahu v proxy

#OWNNET-16, zodpovedná osoba: Matúš Tomlein

Odpovedanie na dopyty na dynamický obsah, ktorý aplikácia sama vytvára. Tento obsah je prístupný na doméne ownet a väčšinou má formát JSON.

1.7 Inštalácia virtuálneho servera pre potreby projektu

zodpovedná osoba: Michal Dorner

Nainštalovanie servera. Vytvorenie používateľských účtov pre členov týmu a nastavenie prístupu. Inštalácia a konfigurácia webového servera a gitu pre potretreby webovej stránky projektu.

1.8 Vytvorenie webovej stránky ku projektu

zodpovedná osoba: Marek Láni

Vytvorenie webovej stránky v zmysle požiadaviek predmetu tímový projekt.

1.9 Prieskum frameworkov pre Javascript

zodpovedná osoba: Michal Dorner, Martin Lipták

Prieskum Javascript frameworkov, ktoré by sa dali použiť na vytvorenie portálu. Zvolili sme Backbone.js. pre MVC architektúru v kombinácii s require.js pre rozdelenie do modulov a riešenie závislostí.

1.10 Prieskum podporných nástrojov pre vývoj portálu

zodpovedná osoba: Michal Dorner

Prieskum HTML a CSS preprocesorov, možnosti ich automatizácie a integrácie do IDE. Zvolili sme HTML preprocesor JADE a CSS preprocesor SASS. Na automatizáciu prekladania súborov sa použije MAKE. Správnym formátovaním chybových výstupov je dosiahnutá aj integrácia s editorom Sublime Text 2.

1.11 Analýza doterajšieho projektu

zodpovedná osoba: Karol Balko

Prieskum zdrojového kódu doterajšieho projektu OwNet implementovanom v .NET, pre priblíženie fungovania projektu.

2 Šprint Francis Kere

2.1 Cache stránok

#OWNNET-47, zodpovedná osoba: Matúš Tomlein

2.1.1 Úloha

Navrhnúť a implementovať ukladanie navštívených webových stránok do cache a načítavanie webových stránok z cache. Pri návrhu je vhodné vychádzať z implementácie cache z .NET verzie aplikácii OwNet.

Táto úloha vyžaduje aj návrh a implementáciu vhodnej architektúry proxy, ktorá umožňuje efektívne čítanie a zapisovanie do vyrovnácej pamäte.

2.1.2 Analýza

Ukladanie webových stránok do cache musí prebiehať súčasne so sťahovaním stránok a ich zapisovaním na výstup prehliadaču. Preto je potrebný návrh, ktorý umožňuje pre jeden vstupný kanál (web), vytvoriť viacero výstupných kanálov (prehliadač, cache).

V návrhu a implementácie je vhodné zohľadniť aj rozšírenie tejto architektúry na viac ako dva výstupné kanály, keďže môžu pribudnúť aj výstupný kanály pre ďalšie aplikácie OwNet pristupujúce k stránke cez lokálnu sieť.

Je potrebné vyriešiť neskoršie pripojenie nového výstupného kanálu a umožniť mu stiahnutie všetkých dát.

2.1.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu. Architektúru tohto riešenie sme rozšírili a vylepšili na základe skúseností s ňou.

Návrh architektúry pozostáva zo štyroch častí:

1. Vstup
    - Môže to byť Web, výrovnávacia pamäť, statický obsah alebo dynamický obsah
2. Spoj
    - Slúži na prijímanie dát zo vstupu a posielanie ich na výstup
    - Umožňuje registráciu viacerých výstupných kanálov
    - Zabezpečuje, že aj výstupné kanály, ktoré sa pripoja po spustení sťahovania, budú mať prístup k všetkým prijatým dátam
3. Výstup
    - Môže byť viacero, napríklad cache, webový prehliadač (socket) alebo lokálna sieť
4. Smerovač výstupných kanálov
    - Zabezpečuje po vytvorení nového výstupného kanálu, priradenie nového spoju alebo v prípade, že hľadaný obsah je práve sťahovaný, priradenie na existujúci spoj

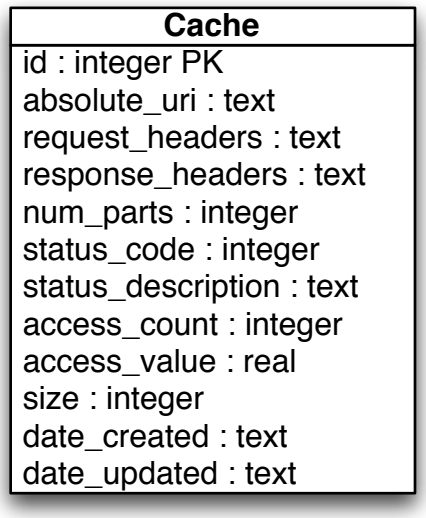
2.1.4 Implementácia

Úloha bola implementovaná v C++ a Qt s použitím databázy SQLite. Objekty v cache sú ukladané ako súbory na disk, kvôli odľahčeniu databázy.

V databáze sú uchované len informácie o cache objekte ako jeho URL adresa, HTTP request a response hlavičky, veľkosť a iné. HTTP hlavičky sú ukladané vo formáte JSON. Keďže jeden objekt môže byť na disku uložený vo viacerých súboroch, v databáze je uchovaný aj počet takýchto častí. Reprezentáciu cache objektu v databáze je možné vidieť na obrázku 1.

Pre vstupný aj výstupný kanál bolo definované rozhranie, ktoré umožňuje jednoduché prídavanie nových typov týchto kanálov.

Spoj si dočasne uchováva dáta v pamäti, pokiaľ má registrované výstupy, kvôli zabezpečeniu registrácie nových výstupov počas sťahovania.



Obr. 1: Reprezentácia cache objektu v databáze

2.1.5 Testovanie

Implementované riešenie sme testovali na vzorke niekoľkých webových stránok. Navštívené web stránky boli uložené do cache a bolo ich možné neskôr načítať aj bez pripojenia na internet. V čase riešenia úlohy nebolo projekt možné otestovať integračne, v neskorších úlohách sa však budeme zaoberať integračným testovaním tejto funkcionality.

2.2 Základy prednačítavania

Úloha #OWNNET-8, podúlohy #OWNNET-59, #OWNNET-60 zodpovedná osoba: Martin Konôpka

2.2.1 Úloha

Implementovať základy prednačítavania, vďaka ktorým bude možné sledovať používateľov pohyb na webových stránkach. Úlohou je zistiť navštívenie stránky používateľom, výber odkazov na prednačítanie.

Úloha je rozdelená do dvoch podúloh:

- **Analýza možností znovupoužitia pôvodného riešenia (OWNET-59)** - Ako je možné využiť existujúce časti kódu pôvodného riešenia tejto funkcionality.
- **Sledovanie navštívenia stránky používateľom (OWNET-60)** - Získanie znalosti o návšteve stránky používateľom, výber prvých pár odkazov stránky na prednačítanie a zaradenie do fronty.

2.2.2 Analýza

Základom prednačítavania je určenie stránky, pre ktoré sa majú prednačítať ich odkazy. Najprv musíme zistiť, ktorú stránku používateľ navštívil. V rámci samotnej proxy aplikácie nie je možné zistiť, ktorá požiadavka z prehliadača pochádza zo stránky vyžiadanej samotným používateľom (môže to byť aj vnorená stránka, napr. reklama).

Vhodným riešením je oznamovanie návštevy z vloženého script súboru na stránke, ktorý proxy aplikácia vkladá do každej stránky. Tento princíp sme uplatnili v pôvodnom riešení a rozhodli sme sa ho uplatniť aj teraz. Samozrejme musíme riešiť problém vnorených stránok, podobne ako v pôvodnom riešení. Script kontaktuje proxy aplikáciu len v prípade, že stránka nie je vnorená.

Potom ako vieme, že používateľ sám vyžiadal stiahnutú stránku, môžeme z nej vybrať odkazy na prednačítanie. Odkazy si proxy aplikácia zaradí do radu odkazov na prednačítanie.

2.2.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu.

Návrh riešenia pozostáva 2 činností: zistenie používateľovej návštevy stránky a výber odkazov na prednačítavanie:

Zistenie používateľovej návštevy stránky

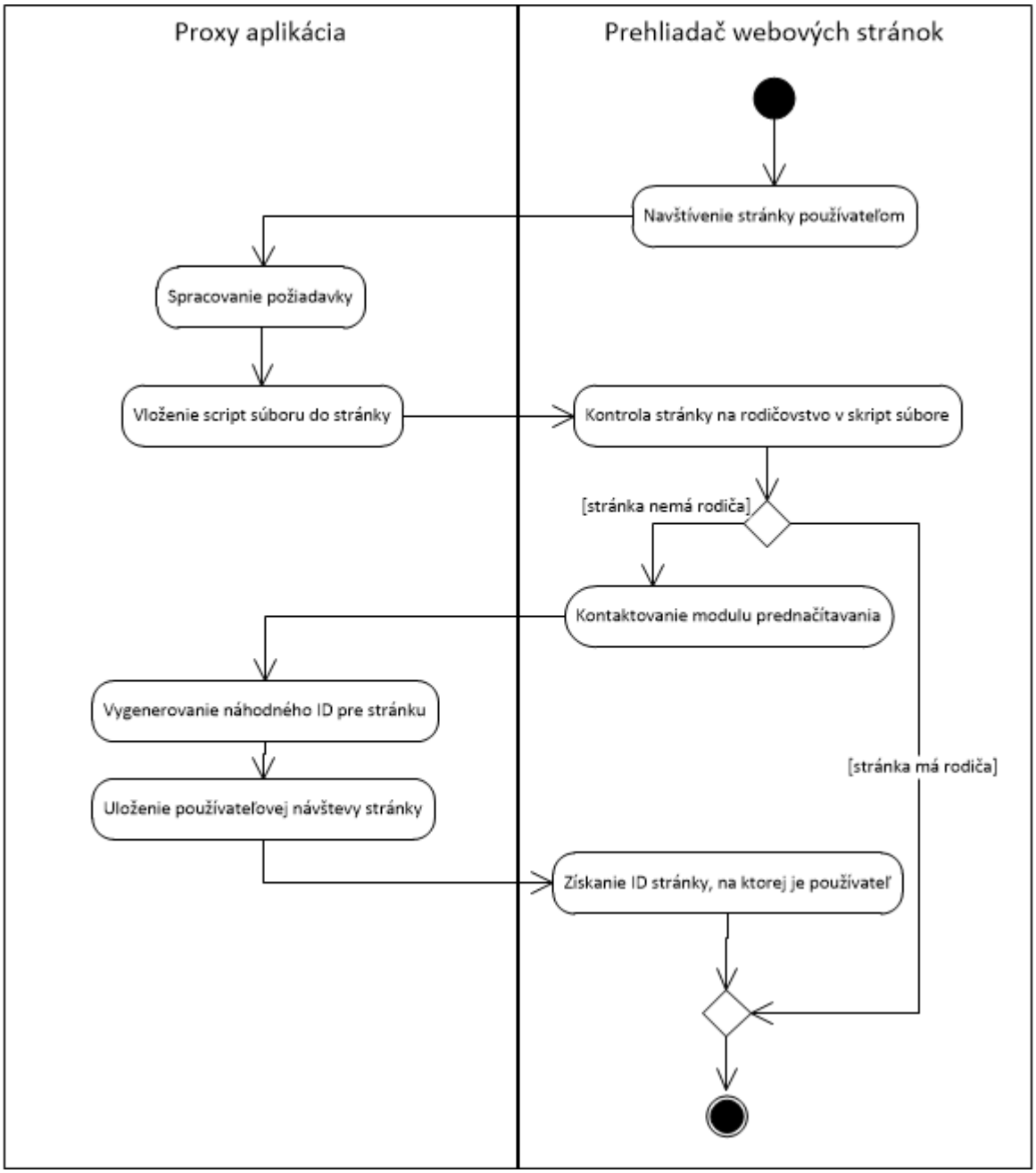
Obrázok 2 znázorňuje priebeh zisťovania používateľovej návštevy stránky. Pozostáva z nasledujúcich krokov.

1. Vstup
    - Navštívenie stránky používateľom, vloženie script súboru proxy aplikáciou.
2. Zistenie, či bola stránka vyžiadaná používateľom
    - Zistenie, či skript je vložený v stránke, ktorá nemá žiadnu rodičovskú stránku.
    - Kontaktovanie modulu prednačítavania proxy aplikácie o používateľovej návšteve.

- Vygenerovanie náhodného ID pre stránku, ktoré bude proxy aplikácia a script používať ako referenciu.

3. Výstup

   - Modul prednačítavania si uchová informáciu o používateľovej návšteve.



Obr. 2: Proces získania informácie o používateľovej návštevy stránky.

Výber odkazov na prednačítanie

1. Vstup
   - Script súbor je vložený na stránke vyžiadanej používateľom, stránka bola oznámená proxy aplikácií.
2. Výber odkazu
   - Script súbor náhodne vyberie 3 odkazy zo stránky.
   - Odošle odkazy spolu s ID aktuálnej stránky modulu prednačítavania v proxy aplikácii.
3. Uchovanie odkazu
   - Modul prednačítavania v proxy aplikácii prijme odkazy.
   - Odkazy sa uložia medzi odkazy stránky podľa ID referencie.
4. Výstup
   - Zoznam odkazov pre prednačítanie pre stránky, ktoré navštívil používateľ.

2.2.4 Implementácia

Úloha bola implementovaná v C++ , Qt a JavaScript. Vkladanie script súboru bolo zabezpečené v úlohe *OWNET-48*. Kód script súboru bol znovupoužitý z pôvodného riešenia. Objekt owNetGLOBAL. ProxyContat kontaktuje proxy aplikáciu vkladaním nových script súborov, ktorých adresy odchytáva proxy aplikácia. Tak vie vložený script komunikovať s proxy aplikáciou. Nie je totiž možné použit AJAX (XmlHttpRequest) z dôvodu XSS (nemožnosť dopytovať URL adresu, ktorá je z inej domény ako aktuálna stránka).

Formát volania proxy aplikácie je nasledujúci:

http://static.ownet/prefetch/visit?page=URL

Proxy aplikácia odpovedá s nasledujúcim reťazcom:

owNetPAGEID = ID

Tak získa script súbor informáciu o priradenom ID stránke. Toto ID používa ako referenciu pri oznamovaní odkazov. Podobným spôsobom sa odosielajú odkazy.

http://static.ownet/prefetch/link?from=ID&to=URL

Tieto dve volania odchytáva modul PrefetchingModule a podľa toho si vytvára záznamy do mapy stránok (kľúčom je ID stránky a hodnotou objekt LoggedPage). Mapa stránok obsahuje navštívené stránky a ich odkazy zvolené na prednačítanie.

2.2.5 Testovanie

Testovanie tejto funkcionality bolo zebezpečené ručným testovaním prehliadania stránok a sledovaním vykonávania kódu aplikácie jej prerušovaním. Podľa úspešnosti operácií zistenia návštevy a výberu odkazov proxy aplikácia vypisovala správy do konzoly.

2.3 Možnosti znovupoužitia pôvodného riešenia

Úloha #OWNNET-8, zodpovedná osoba: Martin Konôpka

2.3.1 Úloha

Analýza služieb v pôvodnom riešení, odhaliť možnosti ich refaktoringu pre účely znovupoužitia v novom projekte, aj keď je založený na inej platforme. Identifikovať ako implementačne náročné bude prepojiť nový projekt s refaktorovanými službami pôvodného projektu, aké prínosy a obmedzenia takýto prístup prinesie.

2.3.2 Analýza

Pôvodný projekt OwNet poskytoval služby vo všetkých troch aplikáciách.

1. Lokálny klient - služby zabezpečovali fungovanie lokálneho portálu a komunikácie vloženého script súboru s proxy aplikáciou. Boli riešené triedami dediacimi od triedy LocalResponder.
2. Lokálny server - služby poskytovali prístup k údajom zdieľanej cache a kolaboratívnych nástrojov.
3. Centrálny server - služby pre komunikáciu viacerých lokálnych serverov cez internet, oznamovanie cache a fungovanie globálnych skupín.

Služby lokálneho klienta a servera umožňovali prístup k takmer všetkým funkcionalitám (uložené stránky, nové načítanie stránky, vykonávanie a ovládanie prednačítavania, kolaboratívne nástroje, manažment používateľov, atď.).

2.3.3 Záver analýzy

Ak by sme uvažovali znovupoužitie kódu, jeho vyčlenenie do modulov a umožnenie komunikovať Qt aplikácii s .Net knižnicou, vykonávanie by bolo spomalené komunikáciou a prekladaním požiadaviek a ich odpovedí cez definované rozhrania. Navyše logika väčšiny služieb lokálneho servera je silno previazaná s dopytovaním do databázy. Tým pádom je volania ťažké odčleniť.

Zhodnotenie znovupoužitia služieb pre jednotlivé funkcionality:

1. Pre uchovávanie stránok (cache) nie je možné použiť, keďže hlavná funkcionalita proxy aplikácie musí byť čo najefektívnejšie.
2. Pre prednačítavanie nie je možné použiť z dôvodu silnej previazanosti s databázou. Väčšina logiky sa prenáša do dopytov v dabatáze z dôvodu efektívnosti.
3. Pre funkcionality lokálneho portálu je služby možné použiť. Ale ak bude portál implementovaný iným spôsobom ako bol v pôvodnom projekte, bude potrebný zásah do pôvodných služieb vo väčšom rozsahu (nielen znovupoužitie, ale aj úprava).

Z vyjadreného záveru sme sa rozhodli nepoužiť priamo zdrojové kódy pôvodného riešenia. Samozrejme však z nich budeme čerpať princípy ako implementovať funkcionality v novom projekte.

2.4 Analýza používateľských skupín

#OWNNET-41, zodpovedná osoba: Marek Láni

2.4.1 Úloha

Analyzujte možnosť znovu-použitia existujúcich služieb skupín v pôvodnom riešení.

2.4.2 Analýza

Analýzou bolo zistené, že znovu-použiteľnosť kódu z pôvodnej verzie v .NET, nie je možné, resp. iba vo veľmi malom rozsahu, nakoľko kód je silno naviazaný na databázu, ktorá sa v novom riešení mení.

2.5 Qt Zbernica

#OWNNET-42, zodpovedná osoba: Marek Láni

2.5.1 Úloha

Vytvorte Qt zbernicu, ktorá bude umožňovať registrovanie modulov, t.j. budú môcť byť využívané aplikáciu. Táto zbernica bude umožňovať komunikáciu proxy s modulmi ale zároveň aj medzi modulmi samotnými. Takýmto spôsobom bude sprístupnený dynamický obsah. Táto úloha sa skladá z nasledujúcich pod-úloh:

- Analýza pôvodnej triedy LocalResponder v .NET projekte, ktorá má za úlohu smerovania komunikácie
- Registrácia modulov
- Odpoveď modulov na požiadavku z proxy
- Komunikácia medzi modulmi

2.5.2 Analýza

Analýzou sme dospeli k záveru, že je nutné vytvoriť rozhranie jednak na strane zbernice ale rovnako i na strane modulov, za pomoci ktorého budú tieto dve stavebné časti architektúry vedieť komunikovať a vymieňať si obojstranne správy/dáta.

2.5.3 Návrh

Návrh rozhrania pre zbernicu pozostáva z funkcie, ktorá registruje moduly, ktoré cez svoje rozhranie zbernici poskytnú funkciu, prostredníctvom ktorej im zbernica môže odovzdať dáta. Moduly dáta spracujú a odpoveď vrátia ako výstup tejto funkcie. V prípade, že moduly potrebujú medzi sebou komunikovať, vyvolajú funkciu zbernice, ktorá sprostredkuje komunikáciu medzi týmito modulmi a je akýmsi prostredníkom.

2.5.4 Implementácia

Úloha bola implementovaná v C++ a Qt. Rozhranie IModule, od ktorého sú odvodené všetky ostatné moduly definuje funkciu processRequest, ktorá slúži na spracovanie požiadavky, resp. rozhoduje, aká akcia modulu je požadovaná. Po spracovaní dát zbernici vráti výsledok.

Na registráciu modulov na zbernici slúži funkcia registerModules. Táto funkcia využíva objekt QMap, v ktorom sa mapujú názvy modulov resp ich url adresa a samotné objekty modulov.

Komunikácia medzi modulmi je umožnená funkciou zbernice callModule, ktorá na cieľovom module vyvolá funkciu processRequest a následne výsledok tejto funkcie vráti modulu, ktorý funkciu callModule vyvolal.

2.5.5 Testovanie

Odtestované boli funkcie registerModules a processRequest a to volaním samotných modulov resp. ich akcií priamo z prehliadača.

2.6 Vloženie Javascript kódu do stránok v proxy

Úloha #OWNNET-48, podúlohy #OWNNET-49, #OWNNET-50 zodpovedná osoba: Karol Balko

2.6.1 Úloha

Navrhnúť a implementovať vloženie JavaScript kódu do webstránky spracovanej cez proxy.

2.6.2 Analýza

Vloženie JavaScript kódu spočíva v zabezpečení správneho umiestnenia odkazu na skript, ktorý sa má vložiť do stránky. V prvom rade je dôležité vhodne spracovať web stránku prostredníctvom proxy servera.

Ako ďalší krok je nutné zabezpečiť vyhľadanie tagu "body", ktorý označuje začiatok tela web stránky a vložiť pred tento tag odkaz na vložený script. Celková analýza vzišla ako správna, avšak bolo treba prepracovať vyhľadávanie body tagu, inak ako podľa pôvodného plánu.

2.6.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu. Architektúru tohto riešenie sme rozšírili a vylepšili na základe skúseností s ňou.

Návrh riešenia pozostáva z 3 častí:

1. Vstup
    - Web stránka, ktorú si vyžiada užívateľ
2. Vloženie javascript kódu
    - Parsovanie HTML kódu web stránky
    - Vyhľadanie tagu "body" v HTML kóde
    - Vloženie odkazu na javascript súbor, ktorý má byť vložený do stránky
3. Výstup
    - Web stránka s vloženým Javascript súborom

2.6.4 Implementácia

Úloha bola implementovaná v C++ a Qt. Pomocou metód tohto frameworku sa spracoval HTML kód a vyhľadával sa v ňom tag "body", vyhľadávanie tagu bolo bez ukončovacej zátvorky >, keďže tento tag, môže obsahovať rôzne parametre, individuálne na každej stránke.

Javascript súbor bol vložený pred tento tag pomocou tagu vkladania skriptu .

Táto funkcionalita bola implementovaná v súbore proxysocketoutputwriter.cpp

2.6.5 Testovanie

Testovanie tejto funkcionality bolo manuálne, zabezpečené vložením skriptu, ktorý obsahoval jednoduché vyskakovacie okno alert, ktoré vypísalo text "JS Inject Working", výber stránok bol čo najširší ako www.google.sk, www.azet.sk, www.sme.sk, www.dsl.sk a ďalších aby sa overila funkcionalita. Na niektorých stránkach ako www.sme.sk, kde bolo vložených veľa skriptov, však spôsoboval problémy, čo bolo zapríčinené chybným vyhľadaním tagu body, avšak tento problém sa odstránil a skript vykazoval korektné správanie.

2.7 Vytvorenie kostry portálu v Backbone.js

#OWNNET-51, zodpovedná osoba: Michal Dorner

2.7.1 Úloha

Vytvoriť zdieľaný modul obsahujúci javascriptové knižnice, Backbone triedy použiteľné vo viacerých moduloch, utility a css framework. Vytvoriť vzorový portálový modul - zadefinovaná adresárová štruktúra, použitie MVC tried a require.js modulov. Vytvoriť Makefile pre buildovanie.

2.7.2 Analýza

Moduly webového portálu budú realizované ako *single page app* ale interne budú pozostávať z vela súborov. Pre udržiavateľnosť, prácu v týme a autoamtické buildovanie je potrebné pevne zadefinovať štruktúru aplikácie.

Require.js vyžaduje konfiguračný subor - nastavenie ciest, import non-AMD modulov a vstupný bod aplikácie.

r.js je optimizér, umožňuje spojenie a minimalizáciu jednotlivých modulov do jedného súboru.

JADE umožnuje zo šablóny vygenerovať aj javascirptové funkcie generujúce html - vhodné na použitie v Backbone Views.

2.7.3 Návrh

Štruktúra portálového modulu:

- *app.build.js* : konfiguračný súbor pre r.js (require.js optimizer)
- *config.js* : konfigiračný súbor pre require.js (konfig modulov)
- *css* : adresár s css štýlmi, css/style.css bude generované
- *img* : obrázky *index.html* : generovaný html súbor - vstupný bod v prehliadači
- *js* : adresár s javascirptovými súbormi
  Makefile : pravidlá pre buildovanie
- *MakeFile* : pravidlá pre buildovanie
- *style* : adresár pre SASS štýly, z style.sass sa generuje style.css
- *templates* : adresár pre JADE šablóny, z index.jade sa generuje index.html
- *templates-client* : adresár pre JADE šablóny na client side použitie, generujú sa do js/tpl

Konfiguračný súbor config.js určuje:

- adresár od ktorého sa budú hľadať vstupné javacsirptiové súbory (moduly)
- skrátené názvy modulov a ich plnú cestu
- štruktúru závislostí a výstupov modulov (externých knižníc) ktoré nie sú v AMD formáte
- vstupný bod aplikácie

Cez konfiguračný súbor app.build.js sa nastavuje proces minimalizácie kódu do produkčnej verzie. Nastaví sa v ňom:

- vstupný konfiguračný súbor pre require.js (config.js)
- výsptuný súbor (app.min.js)
- optimizér (uglify.js - odstranenie bieliych miest, komentárov, premenovanie premenných)

2.7.4 Implementácia

Podla návrhu sa implementovala vzorový modul. Zdrojový kód je v repozitári v branch app-template

2.7.5 Testovanie

Vytvorený vzorový modul sa otestoval ručne spustením v prehliadači. Buildovanie sa otestovalo na linuxe a windows.

2.8 Analýza a návrh funkcionality portálu

Úloha #OWNNET-56, podúlohy #OWNNET-57 a #OWNNET-58, zodpovedná osoba: Andrea Šteňová

2.8.1 Úloha

Analyzovať portál existujúceho riešenia, jeho funkcionalitu a nedostatky. Navrhnúť vylepšenia k existujúcemu riešeniu a popísať ich.

2.8.2 Analýza

Existujúci portál poskytuje funkcie na prácu s obsahom na internete, ako je zdieľanie alebo hodnotenie stránok.Štruktúra existujúceho portálu je nasledovná:

- Nástenka s aktuálnymi udalosťami - obsahuje všetky príspevky používateľov školy - odporúčania, hodnotenia, zdieľania súborov atď. Správy však nie sú nijak rozumne štrukturované.
- Vyhľadávanie - vyhľadávanie medzi stránkami s priradenými tagmi, alebo na sieti. Neobsahuje možnosť vyhľadávať na internete.
- Správy - zobrazované všetkým používateľov v sieti. Zobrazované sú všetky správy za posledný čas, nemajú adresáta, nedá sa na ne odpovedať, sú neštruktúrované, nie je možné v nich vyhľadávať, a nie sú archivované. Správy majú potenciál vylepšovania.
- Hodnotenia - zobrazuje zoznam všetkých hodnotených stránok a priemerných hodnotení týchto stránok. Je to však len neštruktúrovaný zoznam, bez možnosti filtrovania napríklad podľa skupín.
- Odporúčania - podobne ako hodnotenia, odporúčania sú zobrazené len ako zoznam odporúčených stránok, existuje aj filtrovanie po skupinách, ale pri ich väčšom počte môže byť neprehľadné.
- Zdieľané súbory - zdieľanie súborov a ich filtrovanie, podľa toho či ich vytvorili žiaci alebo učitelia.
- História - obsahuje navštívené a uložené stránky, ktoré sa dajú filtrovať podľa času a je ich možné z histórie vymazať. Keďže táto časť portálu je nepotrebná pre bežných používateľov, v novej verzii je potrebné uvažovať o jej presunutí k nastaveniam používateľa.
- Skupiny - stránka obsahuje skupiny v ktorých je používateľ, ako aj všetky skupiny. Ak však bude skupín veľmi veľa, môže byť tento zoznam neprehľadný. V aktuálnej verzii sa nerieši pridávanie ľudí do skupín, každý sa môže pripojiť do ktorej skupiny chce. Pre skupinu si môže zobraziť jej používateľov, ako aj odporúčania v skupine.
- Nastavenia používateľa - ako napríklad zmena hesla a informácií v profile, ale aj nastavenia cachovania používateľovi. Tieto nastavenia je potrebné prerobiť a zjednodušiť, aby boli pochopiteľné aj bežnému používateľovi.
- Zoznam používateľov - zoznam všetkých používateľov na škole, obsah tohto zoznamu sa nedá filtrovať a preto by mal byť prerobený.

- Zoznam stránok na stiahnutie - je schovaný pri užívateľových nastaveniach, obsahuje stránky, o ktoré požiadal používateľ, alebo mu boli prednačítané, v ďalšej verzii programu, by mal byť tento zoznam viditeľnejší.

V aktuálnej verzii sú síce rozlíšené roly študenta a učiteľa, ich funkcionalita sa však veľmi nelíši. Aj keď je to portál pre školy, chýba možnosť elektronického vzdelávania, alebo možnosť zdieľania materiálov aj medzi rôznymi školami.

2.8.3 Návrh

Nasledujú identifikované funkcionality, ktoré by sme chceli v budúcnosti implementovať.

Správa používateľov

Do správy používateľov je potrebné implementovať tieto fukcionality:

Registrácia/prihlásenie

- používateľ má možnosť registrovať sa
- používateľ sa vie prihlásiť/odhlásiť zo systému
- chceme umožniť prihlásenie každého z používateľov zo všetkých počítačov v učebni/škole.

Roly v systéme V doterajšej implementácii systému chýba konkretizácia rolí v systéme a ich možnosti a zodpovednosti. Identifikovali sme tieto roly:

- učiteľ - pracuje s administrátorskou časťou systému, vytvára obsah a hodnotí aktivitu žiakov.
- školský správca - môže a nemusí byť iný od učiteľa. Spravuje vytváranie rôznych skupín po triedach v škole a ich prepojenie (zdieľanie) súborov.
- študent - pracuje s portálom, učí sa v ňom, odporúča články, hodnotí.

Nastavenia používateľov V aktuálnej verzii sú nastavenia používateľov ako zmena kontaktných údajov, hesla alebo nastavenie cachovania implementované dosť komplikovane. Preto treba tieto kroky čo najviac zjednodušiť pre používateľa.

Prerobenie existujúcej funkcionality

Keďže OwNet prerábame aby fungoval multiplatformovo, potrebujeme prerobiť existujúcu funkcionalitu portálu, čo nám dáva možnosť na jej vylepšenie. Vylepšovať budeme nasledujúce časti:

Hodnotenia a odporúčania

- spojiť tieto funkcionality do jednej podstránky
- možnosť hodnotiť/odporúčať stránky
- pridávať k hodnotenia/odporúčania skupinám
- filtrovať tieto hodnotenia/odporúčania

- zobrazovať len relevantné pre používateľa
- zobrazovať odporúčania aj počas pozerania webu pri stránkach s rovnakou doménou alebo pri podobných stránkach

Zdieľanie súborov

- pridávať súbory pomocou Drag&Drop
- vytvoriť hierarchiu súborov do priečinkov
- nastavenie viditeľnosti súborov - všetkým/skupine/konkrétnym osobám/len mne
- tagovať súbory
- filtrovať súbory na základe tagov alebo používateľov, ktorí ich vytvorili
- zobrazovať len priečinky, ktoré sa týkajú konkrétneho používateľa

Skupiny

- vytváranie skupín a nastavenie ich prístupu na pre všetkých/len s heslom/len pre odsúhlasených používateľov
- riešiť pomocou skupín aj školské predmety
- filtrovanie obsahu podľa skupín

Správy

- pridať možnosť poslania súkromnej správy
- pridať možnosť poslania správy skupine, alebo všetkým v systéme
- zobrazovať používateľovi notifikácie a správach, ktoré sa ho týkajú
- zobrazovať používateľovi všetky správy, ktoré sa ho týkajú
- zobrazovať súkromné správy ako prečítané/neprečítané

Pridanie elektronického vzdelávania

Keďže vyvíjame produkt predovšetkým pre školy, je kritické aby sme do neho pridali podporu elektronického vzdelávania. K nej pristupujú učiteľ, ako ten, čo vytvára obsah, a žiak, ktorý s obsahom pracuje.

Predmety a kurzy

- učiteľ
  - vytváranie predmetov a kurzov
  - pridávanie študentov do skupín
  - pridávanie študentov podľa už existujúcej skupiny, alebo podľa zoznamu zo súboru
- žiak
  - prezeranie predmetov a kurzov, do ktorých bol priradený

Testy

- učiteľ
  - vytvorenie testov
  - priradenie testu do predmetu a skupiny
  - nastavenie času zobrazenia a spustenia pre skupiny
  - vytváranie otázok - voľná odpoveď/jedna správna odpoveď/viacero správnych odpovedí...
  - zobrazenie kto z ktorej skupiny už odpovedal
  - uzavretie testov
  - vyhodnotenie testov
  - priradenie testu jednotlivým študentom, nie skupine
- žiak
  - zobrazenie všetkých priradených testov a dátumu, kedy začínajú
  - notifikácia o priradení testu pre študenta, o ohodnotení testu
  - riešenie testu

Učebné texty

- učiteľ
  - vytváranie materiálov
  - priradenie k predmetom
  - kategorizácia, štrukturovanie materiálov
- žiak
  - zobrazenie materiálov podľa predmetov
  - filtrovanie materiálov
  - vyhľadávanie v nich
  - tagovanie materiálov

Bodovanie a hodnotenie študentov

- učiteľ
  - zobrazenie všetkých študentov
  - filtrovanie podľa predmetov/skupín
  - usporiadanie študentov podľa aktivity a bodov
  - pridávanie známok

Správa obsahu

História

- zobrazovať štrukturovanú históriu navštívených a stiahnutých stránok
- pridať možnosť vyhľadávať a mazať tieto stránky
- priamo tu pridať možnosť nastavenia necachovania niektorých zo stránok

Zoznam stránok na stiahnutie

- zobrazovať používateľovi stránky ktoré chcel stiahnuť a ktoré mu systém prednačítal
- filtrovať stránky, ktoré už sú stiahnuté a na ktoré sa ešte čaká
- poslať notifikáciu používateľovi o stiahnutí niektorej zo stránok pre neho

Ďalšia funkcionalita

Identifikovali sme aj ďalšie funkcionality ktoré by bolo dobré implementovať, sú to:

- Vytvorenie nástenky používateľa - na nástenke vidí používateľ svoju fotku, obsah ktorý odporučil a hodnotil, správy ktoré napísal a súbory, ktoré zdieľal.
- Wiki - na efektívnejšie a jednoduchšie vytváranie obsahu, zdieľanie relevantných informácií.
- Fórum - na lepšiu a kategorizovanú diskusiu otázok a problémov používateľov.
- Notifikácie - o každej zmene v systéme, ktorá sa týka konkrétneho používateľa - prišla mu správa, bol mu vytvorený test, bola mu pridelená známka atď. Po kliknutí na notifikáciu sa portál presmeruje na konkrétnu funkcionalitu, ktorá bola zmenená.

2.8.4 Záver

Navrhnuté funkcionality boli pridané do systému na manažovanie úloh a budú vykonané v nasledujúcich šprintoch v priebehu roka podľa ich priority.

2.9 Databáza

Úloha #OWNNET-52, zodpovedná osoba: Martin Lipták

2.9.1 Úloha

Sprístupnenie databázy v aplikácii OwNet.

Analýza

Porovnanie databázových systémov z hľadiska ich vhodnosti pre použitie v proxy a pre portál. Požiadavky na databázu sú nenáročnosť a jednoduchá inštalácia. Toto spĺňa iba databázová knižnica SQLite (MongoDB má tiež takúto možnosť, ale nie je dobre zdokumentocaná a široko používaná). Okrem toho sme zvažovali nasledujúce možnosti.

- MySQL
- PostgreSql
- CouchDB
- MongoDB

Všetky tieto databázy sú náročnejšie na prostriedky ako SQLite a je potrebné ich samostatne pri nasadení inštalovať. CouchDB a MongoDB majú prepracované možnosti replikácie, ktorými sa môžeme pri distribuovanej architektúre inšpirovať.

Implementcia

Pri implementácii SQlite som využil možnosti frameworku Qt, ktorého triedy pre prístup k databáza obsahujú ovládač pre databázu SQLite3. SQlite som implementoval najjednoduchším možným spôsobom vytvorením globálnej databázy pre celú aplikáciu pri inizializácii proxy.

Testovanie

Pripravil som ukážkový kód, ktorý vytvorí databázovú tabuľku, naplní ju záznamami a vyskúša tieto záznamy získať a vypísať. Výsledný kód poslúžil aj ako príklad pre ostatných ako používať databázu.

2.10 Francis Kere - Pomocné úlohy

2.10.1 Pomocné úlohy

Komunikácia s OLPC

zodpovedná osoba: Martin Lipták, Matúš Tomlein

Prieskum možností spolupráce s projektom One Laptop Per Child a mailová komunikácia v mailing liste komunity.

Gymnázium Žiar nad Hronom: Pripojenie na Internet je kvalitné, ale pri hromadnom načítaní videí by mohol Ownet pomôcť. K iným funkciám aplikácie Ownet sa ale nevyjadrili, nevzbudili v nich dostatočný záujem. Cachovanie sa dá ale realizovať aj pomocou centrálneho proxy servera (čo pre slovenskú škole nie je taký problém ako pre školu v Afrike), kedy je inštalácia pre administrátora podstatne jednoduchšia (keďže škola má kvalifikovaného externého administrátora) ako inštalácia aplikácie na všetky počítače.

One Laptop Per Child: Projekt OLPC distribuuje lacné a odolné laptopy v chudobných častiach sveta za účelom zlepšenia vzdelávania. Laptopy obsahujú vlastné prostredie s edukačnými a kolaboratívnymi aplikáciami, ktoré je postavené na GNU/Linux, takže možné budúce nasadenie sa môže týkať len multiplatformovej verzie. Používatelia by kvôli prítomnosti natívneho prostredia s podobným zameraním ako portál pravdepodobne väčšinu buudúcej funkčnosti portálu nevyužili. Kvalitné cachovanie a prednačítavanie by mohlo mať význam. Členovia mailing listu odporučili existujúce riešenia na sprístepnenie offline obsahu (niektoré sme nepoznali). Pýtali sa na rozdiel medzi open-source proxy serverom Squid a našim riešením. Maily boli ladené pozitíne, naznačovali záujem o projekt, ale nikto neprejavil aktuálnu nutnosť podobného riešenia, keďže po výmene niekoľkých e-mailov komunikácia ďalej nepokračovala. Výňatky z komunikácie sú vo výpise **??**.

3 Šprint Moi Equator Girls

3.1 Vymazávanie cache

#OWNNET-82, zodpovedná osoba: Matúš Tomlein

3.1.1 Úloha

Navrhnite a implementujte vymazávanie cache webových stránok po dosiahnutí maximálnej povolenej hranice jej veľkosti. Využite algoritmy z riešenia OwNet v .NET.

3.1.2 Analýza

Pri riešení v .NET bol spomedzi viacerých algoritmov vybraný algoritmus GDSF (Greedy-Dual-Size-Frequency) ako najvhodnejší na cache webových stránok.

Veľkosť cache by sa mala kontrolovať periodicky. Po prekročení stanovenej hranice by sa malo uvoľniť viac miesta, aby nebolo potrebné ďaľšie uvoľňovanie pamäte príliš skoro.

3.1.3 Návrh

Algoritmus GDSF vyžaduje pre každý objekt v cache nasledovné informácie:

- veľkosť
- počet prístupov
- hodnotenie (váha)

Hodnotenie objektu sa počíta pomocou vzorca: $Pr(f) = Clock + Fr(f)x100/ln(Size(f) + 1.1)$

Kde Pr(f) je hodnotenie, Fr(f) je počet prístupov na objekt, Size(f) je veľkosť a Clock je váha naposledy vymazaného objektu z cache. Parameter Clock je preto potrebné ukladať globálne.

3.1.4 Implementácia

Úloha bola implementovaná v C++ a Qt s použitím databázy v SQLite.

Pre každý záznam o cache objekte v databáze bola pridaná informácia o jeho veľkosti, počte prístupov a váhe. Zároveň je v databáze ukladaná globálna informácia o váhe naposledy vymazaného objektu.

Vymazávanie je spúšťané periodicky, predvolený interval je 1 minúta. Maximálna veľkosť cache je nastaviteľná.

3.2 Prednačítavanie stránok

Úloha #OWNNET-65, podúlohy #OWNNET-65 a #OWNNET-106, zodpovedná osoba: Martin Konôpka

3.2.1 Úloha

V úlohe prvého šprintu *Základy prednačítavania (#OWNNET-8)* boli položené základy sledovania používateľovej aktivity, ktoré oznamovali aj odkazy stránok na prednačítanie. V tejto úlohe potrebujeme implementovať mechanizmus ich prednačítavania, ktorý stiahne nielen súbor samotnej stránky, ale aj jej obsah (vložené obrázky, štýly, script súbory).

Úloha je rozdelená na dve podúlohy:

- **Sťahovanie stránok (#OWNNET-106)** - Spustenie stahovania stránok, vyberanie odkazov z radu predpovedí.
- **Mazanie odkazov (#OWNNET-66)** - Keď používateľ opustí stránku, už nie je potrebné pre ňu prednačítavať odkazy. Potrebné je zistiť používateľov odchod zo stránky a vyčistenie radu odkazov.

3.2.2 Analýza

V pôvodnom riešení projektu sme zvolené odkazy na prednačítavanie sťahovali simulovaním navštívenia stránky používateľom v skrytom prehliadači. Tento princíp môžeme uplatniť aj v novom projekte. Proxy aplikácia v pravidelných intervaloch kontroluje zoznam odkazov. Zvolený odkaz spustí v skrytom prehliadači, ktorý je pripojený cez našu proxy aplikáciu. Tým, že sa stránka otvorí v prehliadači a nie len jednoducho stiahne jej súbor, vyvoláme všetky požiadavky na vložené súbory stránky (obrázky, štýly, dynamický obsah).

Toto riešenie je jednoduchšie ako prehľadávanie zdrojového kódu a vhodnejšie, keďže sa stiahne v rovnakej podobe ako ju uvidí používateľ. Po uložení stránky a jej objektov v cache (to je zabezpečené pripojením prehliadaču na proxy) je stránka dostupná aj offline.

Spôsobov mazania odkazov je viacero, spomenieme dva:

1. Prioritizácia odkazov a ich postupná Invalidácia na základe času.
2. Sledovanie používateľovej aktivity na stránky a zachytenie jej opustenia. Keď používateľ opustil stránku, už nie je potrebné stiahnuť jej odkaz (čo nemusí vždy platiť). Oba spôsoby mazania odkazov sme uplatnili v pôvodnej verzii projektu. V rámci tejto úlohy tohto šprintu sa zameriame na druhý spôsob.

3.2.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu. Nasleduje návrh riešenia uvedieme pre obe podúlohy.

Simulácia navštívenia stránky

Nasledujúce činnosti sa vykonávajú v pravidelných intervaloch 2 minút.

1. Vstup

- Rad odkazov na prednačítanie, výber jedného odkazu.

2. Navštívenie stránky v skrytom prehliadači

   - Proxy aplikácia nastaví nastavenia proxy servera skrytému prehliadaču (používateľ s ním neinteraguje) na seba.
   - Proxy aplikácia otvorí zvolený odkaz v skrytom prehliadači.

3. Výstup

   - Stránka je prednačítaná spolu s jej obsahom.

Mazanie odkazov

1. Vstup

   - Používateľ opustí stránku.

2. Zachytenie opustenia stránky

   - Vložený script súbor na stránke zavolá akciu modulu prednačítavania pre oznámenie opustenia stránky (to spôsobí zrušenie všetkých odkazov na prednačítanie z tejto stránky).
   - Ak to nie je možné vykonať (podmienky prehliadača), pri oznámení návštevy novej stránky sa neodošle len jej URL adresa, ale aj URL adresa predchádzajúcej stránky (REFERER pole hlavičky HTTP).

3. Odstránenie odkazov

   - Modul prednačítavania spracuje opustenie stránky odstránením všetkých odkazov z radu stránok na prednačítanie.

4. Výstup

   - Rad stránok na prednačítanie neobsahuje odkazy opustenej stránky.

3.2.4 Implementácia

Úloha bola implementovaná v C++ , Qt a JavaScript. Využili sme modul prednačítavania a script súbor z úlohy OWNNET-8. Kód script súboru bol doplnený o oznamovanie opustenia stránky. Do adresy akcie oznamovania návštevy stránky sa pridal parameter ,,ref“, ktorý obsahuje URL adresu predchádzajúcej stránky (získame z document.referrer v JavaScript):

http://static.ownet/prefetch/visit?page=URL&ref=URLFrom

Zároveň bolo pridané volanie akcie pre zrušenie odkazov aj pri samotnom opustení stránky (napr. bez navigácie na ďalšiu stránku) volaním adresy:

http://static.ownet/prefetch/close?page=ID

Samotná simulácia navštívenia stránky sa vykonáva objektom triedy PrefetchingJob, ktorý je vo vlastníctve PrefetchingModule. Volanie prednačítavania (simulácie) sa vykonáva v pravidelných intervaloch stanovených v návrhu. Skrytý prehliadač zabezpečuje trieda QWebView. Volaním jej funkcie load() sa vykoná otvorenie stránky v prehliadači. Používateľ neovláda prehliadač, ale stránka sa stiahne cez proxy aplikáciu.

3.2.5 Testovanie

Testovanie bolo zabezpečené ručným testovaním prehliadania stránok a sledovaním vykonávania kódu aplikácie jej prerušovaním. Podľa úspešnosti operácií zistenia opustenia stránky proxy aplikácia vypisovala správy do konzoly. Priebeh prednačítavania v skrytom prehliadači sme sledovali podobným spôsobom. Zároveň sme simuláciu sledovali aj cez pripojenie na iný lokálny proxy server (program Fiddler), kde sme overili, či sa okrem stránky stiahne aj jej obsah.

3.3 Vytvorenie URL pomocníka na spracovanie HTTP požiadaviek

Úloha #OWNNET-67, podúlohy #OWNNET-68, #OWNNET-69, #OWNET-107 zodpovedná osoba: Karol Balko

3.3.1 Úloha

Navrhnúť a implementovať triedu URL pomocníka, ktorý spracuje požiadavku, rozloží jej časti a parametre a ukladá ich podľa typu parametrov do vhodných premenných, podľa HTTP metód požiadavky ich správne smerovať na funkcie na ich spracovanie.

3.3.2 Analýza

Pre správne spracovanie požiadavky je najprv nutné zistiť HTTP metódu požiadavky, a podľa nej správne požiadavku spracovať. Ak je HTTP metóda GET, tak sa spracúva URL požiadavky, kde jednotlivé časti URL adresy sa na základe lomítok ukladajú do premenných modul, id (ak existuje), akcia. V prípade POST a PUT požiadaviek sa spracúva telo požiadavky, kde sa na základe hlavičky požiadavky najprv zistí či sa jedná o objekt typu JSON, alebo o kódovaný text s parametrami podobnými GET požiadavke. Z implementácie nám vyšlo, že analýza bola správna.

3.3.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu. Architektúru tohto riešenie sme rozšírili a vylepšili na základe skúseností s ňou.

Návrh riešenia pozostáva z 2 častí:

1. Vstup
    - HTTP požiadavka s metódami, GET, PUT, POST, DELETE
    - Spracovanie požiadaviek na základe použitej HTTP metódy
2. Zistenie HTTP metódy požiadavky
    - Na základe metódy spracovať buď URL, alebo telo požiadavky
    - Vhodne presmerovať požiadavky na funkciu spojenú s metódou HTTP požiadavky

3.3.4 Implementácia

Úloha bola implementovaná v C++ a Qt. Prvý krok spočíva v spracovaní hlavičky požiadavky a zistenie HTTP metódy z tejto hlavičky, ak sa jedná o metódu GET, spracúva sa URL požiadavky podľa lomítiek, kde sa časti medzi týmito lomítkami ukladajú do premenných m_module pre module, m_id pre id (ak obsahuje), a m_action pre akciu. Ak sa jedná o metódu POST a PUT, v hlavičke sa zistí, či sa jedná o objekt JSON, ak áno spracuje sa pomocou vopred implementovanej triedy na spracovanie objektov JSON. Ak sa nejedná o objekt JSON tak sa spracúva telo požiadavky, v ktorom sa najprv nahradia kódované znaky ako napr. %28 za (a po nahradení týchto kódovaných znakov sa ďalej rozdeľuje podľa znaku &, pričom prvá časť sa následne ukladá do premennej typu QStringList ako kľúč a druhá ako hodnota.

Ďalej sa pomocou metódy processRequest() smeruje požiadavka podľa metódy na správnu funkciu.

Táto funkcionalita bola implementovaná v súbore proxyrequest.cpp a imodule.cpp

3.3.5 Testovanie

Testovanie tejto funkcionality bolo zebezpečené pomocou pluginu Poster do prehliadača Mozilla, ktorý umožní simulovať a posielať požiadavky s rôznymi HTTP metódami na proxy server.

3.4 Sledovanie vyťaženia linky

Úloha #OWNNET-70, podúlohy #OWNNET-71 zodpovedná osoba: Karol Balko

3.4.1 Úloha

Navrhnúť a implementovať triedu na sledovanie vyťaženia linky a proxy servera

3.4.2 Analýza

Pre sledovanie vyťaženia linky využijeme počítanie počtu požiadaviek na proxy server za určitý čas. Počet týchto dotazov sa ukladá do poľa o veľkosti 60 prvkov z ktorého sa potom urobí súčet dotazov. Táto analýza vzišla ako dobrá pri finálnej implmentácií počítadla.

3.4.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu. Architektúru tohto riešenie sme rozšírili a vylepšili na základe skúseností s ňou.

Návrh riešenia pozostáva z 2 častí:

1. Počítanie požiadaviek
    - Počet požiadaviek prichádzajúcich na proxy server sa sčítava za stanovený čas a ukladá sa do poľa
2. Sčitanie prvkov poľa - Po naplnení prvkov poľa sa tieto sčítajú a vyjde výsledné číslo

3.4.4 Implementácia

Úloha bola implementovaná v C++ a Qt. Prvý krok spočíva v počítaní počtu požiadaviek, ktoré sú prijímané proxy serverom za určitý definovaný čas. Po uplynutí tohto času, sa súčet počtu prijatých požiadaviek uloží do poľa, kým sa pole nenaplní, pole má veľkosť 60 prvkov. Po naplnení poľa sa sčíta počet požiadaviek uložených v poli a výsledné číslo je počet požiadaviek za definovaný čas x 60.

Táto funkcionalita bola implementovaná v súbore proxytrafficcounter.cpp

3.4.5 Testovanie

Toto počítadlo sa nedalo testovať, keďže bolo len počiatočné, ktoré ešte nemohlo byť implementované vzhľadom na to, že niektoré časti proxy severa ešte neboli hotové. Slúžilo však ako základ na finálnu implementáciu, ktorá prebehla v neskoršej dobe pri implementácií ďalších funkcionalít proxy servera.

3.5 Vytvorenie používateľského modulu

Úloha #OWNNET-72, podúloha #OWNNET-74, zodpovedná osoba: Marek Láni

3.5.1 Úloha

Navrhnite a implementujte príjem a spracovanie požiadaviek z GUI. Overte používateľa v DB, umožnite najmä registráciu nových používateľov.

3.5.2 Analýza

K vyriešeniu tejto úlohy je potrebné vytvoriť model používateľa v databáze vo forme tabuľky, ktorej atribúty budú prebrané z riešenia v .NET. Je nutné generovať náhodné a s čo možno najväčšou pravdepodobnosťou unikátne id, pre každého používateľa. Riešenie je vhodné postaviť ako REST api.

3.5.3 Návrh

Tabuľka v databáze pre používateľov pozostáva z nasledujúcich záznamov:

- id
- first_name
- last_name
- login
- email
- password
- date_created
- date_updated

Pre vytvorenie unikátneho id je vhodné použiť hash funkciu na spojenie aktuálneho času s loginom používateľa.

Rozhodli sme sa, že používateľský modul a aj ostatné vytvárané moduly, vzhľadom na naviazanie na REST API budú poskytovať päť základných funkcií resp. služieb a síce *index, show, create, edit, delete.* Rovnako bude možnosť

3.5.4 Implementácia

Úloha bola implementovaná v C++ a Qt s použitím databázy v SQLite. Generovanie unikátneho id pri registrácií používateľa je uskutočnené za pomoci funkcie QHash, ktorej vstupný parameter je reťazec znakov vytvorený spojením loginu používateľa a aktuálneho času.

V rámci používateľských služieb zatiaľ boli implementované iba akcie create, show a index. Požiadavka na volanie akcie obsahuje dáta vo formáte json. Rozhranie služby pre manažment používateľov je nasledujúce:

- *create:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/users. Zloženie posielaných dát musí byť nasledujúce:

  {

  login:'login',
  first_name:'first_name',
  last_name:'last_name',
  password:'pass',
  email:'email'

  }

  Ako odpoveď je vrátené vygenerované ID používateľa a HTTP status 201 a výsledkom je vytvorený používateľský profil. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *show:*

  Táto akcia je volaná parametrom id, ktorý reprezentuje id používateľa. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/users/:id Ako odpoveď je vrátené Json s údajmi o danom používateľovi a HTTP status 200. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *index:*

  Táto akcia je volaná bez parametrov. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/users/ Ako odpoveď je vrátené Json pole s údajmi o používateľoch a HTTP status 200. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

3.5.5 Testovanie

Testovanie prebiehalo za pomoci nástroja POSTER, doplnok prehliadaču firefox, ktorý dokáže generovať a posielať HTTP požiadavky. Odtestované boli všetky tri akcie, vzhľadom na ich rozhranie.

3.6 Vytváranie sedení

Úloha #OWNNET-72, podúloha #OWNNET-73 zodpovedná osoba: Marek Láni

3.6.1 Úloha

Uchovajte informáciu o aktuálne prihlásenom používateľovi. Aktualizácia údaju podľa prihlásenia a odhlásenia.

3.6.2 Analýza

Informáciu o aktuálnom prihlásenom používateľovi je potrebné pre správny beh celej aplikácie, nakoľko väčšia časť funkcionalita bude sprístupnená až po prihlásení. Po analýze spôsobu ukladania tohto údaju v predošlej verzii OwNet, sme sa rozhodli spôsob ukladania prebrať a uložiť informáciu do premennej typu list resp. mapa.

3.6.3 Návrh

Samotné prihlásenie a vytvorenie sedenia sme zabezpečili samostatným modulom *SessionModule. Tento modul obsahuje list resp. mapu, do ktorej sa ukladá záznam o prihlásení používateľa spolu s jeho identifikátorom. Riešenie prostredníctvom mapy tvorí základ pre rozšírenie a umožnenie ukladania rôznych iných údajov v rámci sedenia.*

3.6.4 Implementácia

Úloha bola implementovaná v C++ a Qt s použitím QVariantMap objektu, do ktorého sú údaje v rámci sedenia ukladané. (Aktuálne iba identifikátor prihláseného používateľa).

SessionModule je odvodený od modulu IModule a služby sprístupňuje prostredníctvom REST API.

3.6.5 Testovanie

Testovanie prebiehalo za pomoci nástroja POSTER, doplnok prehliadaču firefox, ktorý dokáže generovať a posielať http požiadavky.

3.7 Registrácia a prihlasovanie používateľov - WEB Gui

#OWNNET-72, zodpovedná osoba: Michal Dorner

3.7.1 Úloha

Vytvoríť základ modulu my.ownet. Vytvoriť základné štýlovanie stránky a hlavného menu. Umožniť registraciu a prihlasovanie pouzivateľov.

3.7.2 Analýza

Použije sa základ portálu podľa **??**. Na základné štýlovanie stránky sa použije css framework bootstrap, pričom bude vhodné ho upraviť do vizuálnej podoby podobnej s pôvodnou verziou ownetu. Bude potrené vytvoriť modely pre používateľa a sedenie a pohľady pre prihlásenie a registráciu.

3.7.3 Návrh

- *UserModule* : Backbone model predstavujúci používateľa.
- *SessionModule* : Backbone model predstavujúci session. Na rozdiel od klasických Backbone modelov, Session nebude mať ID a url jeho zdroja bude priamo na url http://my.ownet/api/session.
- *LoginView* : Backbone view, zabezpečuje prihlasovaciu obrazovku, vytvára session a načítava používateľa. Pri inicialiácii sa skontroluje či na proxy už sednie nie je vytvorené. Ak áno, tak sa načíta a následne sa podľa toho načíta aj model používateľa a upravý sa rozhtanie.
- *RegistrationView* : Backbone view, zabezpečuje obrazovku na registráciu, vytvára a ukladá UserModel

3.7.4 Implementácia

Bol vytvorený základ portáloveho modulu my.ownet, umožňujúci prihlasovanie a registráciu používateľov. Komunikuje s usermodule a session na strane proxy.

3.7.5 Testovanie

Funkcionalita registrácie a prihlásenia bola overená cez integračné testy. V testoch sa simulujú akcie (kliknutie, vyplnenie formulára) na webovej stránke a následne sa porovnáva či sa v databáze vytvoril používateľ a v proxy sedenie a či sa webové rozhranie následne správne upravilo.

3.8 Analýza možností synchronizácie a čiastočná implementácia

#OWNNET-86, #OWNNET-87, zodpovedná osoba: Matúš Tomlein

3.8.1 Úloha

Aplikácia bude používaná na lokálnej sieti, pričom si budú klienti navzájom zdieľať informácie. Preto je potrebné stanoviť spôsob synchronizácie údajov na sieti. Je potrebné definovať štruktúru synchronizovaného obsahu a spôsob jeho vytvárania.

3.8.2 Analýza

Synchronizáciu bude riadiť server, ktorý sa bude voliť dynamicky spomedzi pripojených klientov na sieti. Klienti mu budú posielať zoznam ich posledných zmien a server im bude vracať zoznam zmien od ostatných klientov na sieti.

Tento zoznam zmien (žurnál) bude pre každú zmenu obsahovať nasledujúce atribúty:

- ID pracovnej skupiny
- Unikátne ID zmeny vo formáte ID_klientacislo_zmeny, kde ID klienta bude unikátne číslo na sieti a číslo zmeny bude inkrementálne zvyšované pre každú novú zmenu vytvorenú na klientovi
- Dátum vytvorenia zmeny
- Dátum prijatia zmeny na server
- Rozsah, teda informáciu či sa jedná o globálnu zmenu alebo len zmenu v jednej skupine používateľov
- Zoznam migrácii databázy vo formáte JSON

Žurnál bude tvorený automaticky pri vytváraní nových záznamov v databáze. Bude uložený v databáze a priebežne synchronizovaný medzi klientmi prostredníctvom severa.

Klienti si budú môcť sťahovať buď všetky zmeny, alebo len zmeny v skupinách v ktorých sú registrovaní ich používateľia. Klienti so všetkými zmenami budú preferovaní pri voľbe servera.

Pre každú skupinu (rozsah) zmien bude mať klient uložený dátum poslednej kontroly zmien na serveri. Pri dopytovaní na nové zmeny, klient zašle serveru dátumy posledných kontrol všetkých jeho registrovaných rozsahov zmien. Server vráti klientovi pre tieto rozsahy len zmeny od dátumu ich poslednej kontroly (berie sa do úvahy dátum prijatia na server).

3.8.3 Implementácia

Migrácie databázy majú nasledovnú štruktúru:

```
{
"columns" :
{
"name" : "Jozef",
"surname" : "Mrkvicka",
"username" : "jozko"
```

```
},
"table" : "users",
"type" : 3,
"where" :
{
"id" : 5
},
"returning_id" :
{
"use_column_as" : "user_id",
"queries" :
[
{
"columns" :
{
"name" : "Ted",
"user_id" : ""
},
"table" : "movies",
"type" : 1
}
]
}
}
```

Každá zmena databázy v aplikácii je vytvorená prostredníctvom triedy DatabaseUpdate, ktorá automaticky zabezpečí jej uloženie do žurnálu zmien vo formáte JSON v definovanej štruktúre. Zo žúrnálu sa následne pri najbližšej synchronizácii pošle na server.

3.9 Analýza možností komunikácie a voľby servera

#OWNNET-79 Manažment klientov na sieti, zodpovedná osoba: Martin Lipták

3.9.1 Úloha

Preskúmať možnosti komunikácie medzi inštanciami OwNet proxy a dynamickej voľby servera, ktorý v danom čase túto synchronizáciu zabezpečí. Táto úloha nebola implementovaná.

3.9.2 Analýza

Prieskum možností sieťových protokolov, broadcastu a multicastu. Inšpirácia možnosťami replikácie v databázach.

3.9.3 Návrh

Hrubý návrh komunikácie klienta sa dá ukázať na príklade. Podrobný návrh ešte nie je v tomto šprinte vytvorený.

1. Zapne sa prvý počítač na sieti. Začne každé 3s (príklad, pri testoch na skutočnej sieti sa určí najvhodnejšia hodnota) posielať broadcast UDP správu v tvare klient:(pracovná skupina):(jedinečný identifikátor):(ip):(skóre). Skóre je číslo zistené na základe hardvérovej konfigurácie, vyjadruje výkon.

2. Keď 10s (príklad) nedostane broadcast od servera, zmení pravidelný broadcast na server:(pracovná skupina):(jedinečný identifikátor):(ip):(skóre).

3. Pripojí sa ďalší počítač a začne posielať client broadcast. Zachytí broadcast od servera. Stiahne si od servera žurnál databázy od posledného timestampu. Porovná skóre servera so svojim. Ak má vyššie skôre, začne posielať server broadcast. Pôvodný server to zachytí, porovná skóre a zmení svoj broadcast na client.

4. Všetci klienti sa pravidelne synchronizujú s aktuálnym serverom (napríklad každých 30s).

5. Pripoja sa ďalší klienti, stiahnu zmeny zo servera, všetci majú nižšie skóre, tak fungujú ako klient.

    - Možnosť 1: Klient pravidelne posiela zmeny svojej databázy vo forme žurnálu serveru a pravidelne ťahá zmeny ostatných zo servera. Žurnál obsahuje zmeny.
    - Možnosť 2: Klient pravidelne posiela zmeny databázy na broadcast. Všetci majú všetky zmeny, server ich iba archivuje v prípade príchodu nového klienta.
    - Po prijatí žurnálu sa žurnál zosynchronizuje s DB. Vyriešia sa prípadné konflikty (úprava záznamu, ktorý bol vymazaný).

6. Vypne sa server. Klienti nedostanú broadcast od servera. Ak niečo potrebujú a na server sa nepripoja, počkajú s požiadavkou. Počítač s najvyšším skóre začne posielať server broadcast.

3.10 Zabezpečenie DB

#OWNNET-52, zodpovedná osoba: Michal Dorner

3.10.1 Úloha

Zabezpečiť bezpečnosť databázy pri SQL reqestoch z webového portálu.

3.10.2 Analýza

Komunikácia z javascriptu v prehliadači s databázou na úrovni SQL príkazov sa ukázala ako nevhodná. Zabezpečiť bezpečnosť databázy by bolo veľmi náročné na implementáciu, pre programátorov nepohodlné na používanie a riešenie by ajtak nebolo úplne spolahlivé.

Pre zachovanie cieľa - ľahká rozšíriteľnosť OwNetu bez potreby rekompilácie, jednoduchšie písanie logiky v javascirpte namiesto C++, bolo potrebné hľadať alternatívne riešenia.

3.10.3 Návrh

Prejsť na Qt 5.0 a použiť vstavaný javascript engine. Logiku pre komunkáciu s databázou bude možné písať bezpečne v javascirpte, ktorý sa bude vykonávať pod proxy. Na strane webového prehliadača sa namiesto SQL príkazov použije bežné REST api.

3.10.4 Implementácia

Na otestovanie uskutočniteľnosti navrhovaného riešenia sa vytvoril prototyp proxy migrovovanej na Qt 5 a testovací javascirptový modul, ktorý vedel odpovedať na requesty a komunikovať z databázou.

3.11 Analýza skupín a čiastočný návrh

Úloha #OWNNET-28, podúlohy #OWNNET-90 a #OWNNET-91, zodpovedná osoba: Andrea Šteňová

Úloha

Analyzujte existujúcu implementáciu skupín a navrhnite jej vylepšenie.

Analýza

Z predchádzajúceho šprintu vznikla analýza používateľského portálu, konkrétne aj analýza skupín. Pri ďalšom riešení chceme poskytovať túto funkcionalitu pre skupiny:

- vytváranie a správa skupín
- autentifikácia do skupiny (pomocou prihlasovacieho hesla, schválením používateľov v skupine, manuálnym pridaním členov skupiny..)
- odporúčania v skupine
- filtrovanie skupín

Návrh

Na umožnenie implementácie vzniknutých funkcionalít skupín sme vytvorili dátový model, ktorý sa nachádza na nasledujúcom obrázku.

Opis tried

Roles - roly v systéme Tabuľka rolí v systéme slúži na uchovávanie dát o používateľských rolách. Jej inštanciami sú napríklad *administrátor*, *učiteľ* alebo *žiak*.

Názov	Typ	Opis
id	INTEGER	Primárny kľúč role
name - názov	TEXT	Názov role

Users - používatelia Tabuľka používateľov v systéme obsahuje informácie o konkrétnych používateľoch, ako je jeho meno, priezvisko, prihlasovacie meno do systému a heslo. Obsahuje cudzí kľúč z tabuľky *roles*, aby bolo možné modelovať rolu konkrétneho používateľa.

Názov	Typ	Opis
id	INTEGER	Primárny kľúč používateľa
first_name	TEXT	Meno používateľa
last_name	TEXT	Priezvisko používateľa
login	TEXT	Login do systému, musí byť unikátny pre sieť
email	TEXT	E-mailová adresa používateľa
password	TEXT	Heslo používateľa, uložené ako Hash funkcia
date_created	DATETIME	Dátum registrácie používateľa
date_updated	DATETIME	Dátum poslednej zmeny používateľa
roles_id	INTEGER	Cudzí kľúč z tabuľky *roles*, reprezentuje akú rolu má používateľ v systéme

Group types - typy skupín Typy skupín reprezentujú rôzne možnosti implementácie skupín - skupina môže reprezentovať predmet, alebo krúžok, ako aj záujmovú skupinu.

Názov	Typ	Opis
id	INTEGER	Primárny kľúč typu skupiny
name	TEXT	Názov typu skupiny
date_created	DATETIME	Dátum vytvorenia typu skupiny
date_updated	DATETIME	Dátum poslednej zmeny typu skupiny

Groups - skupiny Tabuľka skupín v systéme zobrazuje skupiny a ich popisy. Takisto reprezentuje typ autentifikácie v systéme. Aby bolo možné štrukturovať skupiny, pridali sme odkaz na nadradenú skupinu.

Názov	Typ	Opis
id	INTEGER	Primárny kľúč skupiny
name	TEXT	Názov skupiny
description	TEXT	Opis skupiny
password	TEXT	Heslo na prihlásenie do skupiny transformované pomocou hash funkcie, voliteľná položka, len ak je skupina zabezpečená heslom
hasPassword	BOOLEAN	Indikátor, či je skupina zabezpečená heslom
hasApprovement	BOOLEAN	Indikátor, či autentifikácia do skupiny ja pomocou schvaľovania administrátorom skupiny
date_created	DATETIME	Dátum vytvorenia skupiny
date_updated	DATETIME	Dátum poslednej zmeny skupiny
group_types_id	INTEGER	Cudzí kľúč z tabuľky *group_types*, reprezentuje typ skupiny
admin_id	INTEGER	Cudzí kľúč z tabuľky *users*, predstavuje administrátora skupiny
parent_id	INTEGER	Cudzí kľúč z tabuľky *groups*, reprezentuje nadradenú skupinu

Group-Users - priradenie používateľ do skupín Priradenie používateľov do skupín. Keďže však existuje proces autentifikácie používateľa v skupine, musíme si udržiavať aj informáciu o stave členstva- či je schválené, čaká na schválenie, bolo neschválené atď.

Názov	Typ	Opis
users_id	INTEGER	Primárny kľúč z tabuľky *users*
groups_id	INTEGER	Primárny kľúč z tabuľky *groups*
status	INTEGER	Stav členstva - vytvorené/čaká na schválenie/zamietnuté/schválené...
date_created	DATETIME	Dátum vytvorenia členstva
date_updated	DATETIME	Dátum poslednej úpravy členstva v skupine

Recommendations - odporúčania Tabuľka odporúčaní používateľov, každý jej záznam predstavuje konkrétne odporúčanie stránky jedným používateľom. Odporúčanie je priradené do skupiny a presne identifikuje odporúčanú stránku podľa jej absolútnej cesty, nadpisu a vytvorením používateľovho opisu.

Názov	Typ	Opis
id	INTEGER	Primárny kľúč odporúčania
absolute_uri	TEXT	Cesta k odporúčanej stránke
title	TEXT	Nadpis odporúčanej stránky
description	TEXT	Opis odporúčanej stránky
date_created	DATETIME	Dátum vytvorenia odporúčania
users_id	INTEGER	Cudzí kľúč z tabuľky *users*, predstavuje používateľa, ktorý vytvoril odporúčanie
groups_id	INTEGER	Cudzí kľúč z tabuľky *groups*, reprezentuje skupinu, ku ktorej odporúčanie patrí

Implementácia

Skupiny budú implementované v niektorom z nadchádzajúcich šprintov.

3.12 Moi Equator Girls - Pomocné úlohy

3.12.1 Pomocné úlohy

Prieskum použiteľnosti OwNetu na slovenských školách

zodpovedná osoba: Andrea Šteňová, Martin Lipták

Komunikácia s učiteľmi informatiky na svojich stredných školách o možnostiach nasadenia OwNetu na gymnáziách.

Gymnázium Jána Chalupku v Brezne: Na tomto gymnáziu sme sa pokúsili nasadiť aktuálnu .Net verziu programu OwNet, kvôli technickým problémom sa to však nepodarilo. Dostali sme však spätnú väzbu o možnom využití systému a prísľub na nainštalovanie stabilnej multiplatformovej verzie. Na tomto gymnáziu nie je problém s rýchlosťou internetu, ale aj tak učitelia tejto školy videli možnosti použitie riešenia na ich hodinách. Pri spokojnosti s výsledným riešením by určite chceli používať podporu elektronického vzdelávania, preto sme pri následnej analýze skupín využili aj poznatky s rozhovoru s nimi. Keďže však na tejto škole nemajú problém s rýchlosťou pripojenia, boli skeptický pri využití častí portálu ako sú správy medzi používateľmi, hodnotenie a odporúčanie stránok, pretože ich žiaci na to používajú iné informačné kanály.

Inštalácia wiki

zodpovedná osoba: Martin Lipták, Matúš Tomlein

Inštalácia Instiki wiki aplikácie na virtuálny server pre tvorbu dokumentácie.

Buildovanie portálu pod windows

zodpovedná osoba: Michal Dorner

Upravenie Makefile a nájdenie windowsových verzií unix programov potrebných pre buildovanie. Spísanie návodu pre inštalácie potrebných programov.

4 Šprint Voi

4.1 Ladenie proxy

#OWNNET-83, zodpovedná osoba: Matúš Tomlein

4.1.1 Úloha

Analyzujte problémové miesta v návrhu a implementácii proxy a cache funkcionality, ktoré spôsobujú spomalenie, pády alebo úniky pamäte v aplikácii. Pre nájdené problémové miesta navrhnite a implementujte opravné riešenia, ktoré čo najlepšie zamedzia ďaľšiemu výskytu podobných problémov.

4.1.2 Analýza

V aplikácii sme zistili tieto hlavné zdroje neefektívnosti a nestability:

- Chyby spôsobené nepozornosťou
- Neefektívne ukončovanie dopytu, ktoré spôsobuje zbytočné čakanie
- Úniky pamäte pri spracovaní dopytov
- Uchovávanie dát sťahovaných odpovedí na dopyty v pracovnej pamäti
  - Počas sťahovania sa uchováva kvôli umožňeniu čítania viacerými súbežnými výstupnými kanálmi
  - Je to problém hlavne pri sťahovaní veľkých objektov ako sú YouTube videá

4.1.3 Návrh a implementácia

Pre jednotlivé identifikované problémy sme navrhli a implementovali nasledujúce riešenia.

Chyby spôsobené nepozornosťou
Tieto chyby je možné odhaliť len testovaním aplikácie v rôznych prostrediach a na rôznych webových stránkach. Odhalené chyby bolo vo väčšine prípadov možné jednoducho odstrániť.

Neefektívne ukončovanie dopytu, ktoré spôsobuje zbytočné čakanie
Pri ukončení dopytu sa používa viacero naviazaných funkcií *signals* a *slots*, ktoré posúvajú informáciu o ukončení dopytu medzi vrstvami spracovajúcimi dopyt. Toto volanie spôsobuje posielanie na vyššie vrstvy architektúry, ktoré následne volajú pomocou signals a slots funkcie na uvoľnenie dopytu a spracovajúcich objektov.

Navrhli a implementovali sme jednoduchšie a spolahlivejšie posielanie informácie o ukončení spracovania dopytu a zabezpečenie uvoľnenia objektov, ktoré pozostáva z nasledujúcich krokov:

1. Objekt, ktorý sťahuje dopyt (objekt jednej z tried dediacich z ProxyInputObject), vyšle signál o dokončení sťahovania
2. Objekt ProxyDownload, ktorý zabezpečuje komunikáciu medzi vstupom dát a výstupmi, odchytí vyslaný signál a na posledné miesto do zoznamu stiahnutých častí objektu vloží informáciu o konci sťahovania

3. Objekty, ktoré zabezpečujú výstup na niektorý výstupný kanál (objekty tried dediacich z ProxyOutputWriter) po prečítaní všetkých stiahnutých častí v ProxyDownload a získaní poslednej časti vyvolajú svoje ukončenie

   (a) Odhlásia sa zo sťahovania v ProxyDownload

   (b) Uzavrú zapisovanie na svoj výstup

   (c) Upozornia objekt ProxyHandlerSession, že môžu byť vymazaní

4. Po odhlásení všetkých ProxyOutputWriter objektov z ProxyDownload sa tento objekt uzavrie, znemožní sa prihlasovanie nových ProxyOutputWriter objektov a upozorní objekt ProxyHandlerSession, že môže byť vymazaný

5. Keď všetky závislé objekty upozornia ProxyHandlerSession, že môžu byť vymazané, ProxyHandlerSession upozorní cez signal funkciu nadradený objekt ProxyHandler, ktorý ProxyHandlerSession vymaže, čím vymaže aj všetky jemu podradené objekty

6. Následne ProxyHandler zavolá signal funkciu, ktorá zabezpečí ukončenie vlákna používaného na spracovanie dopytu

Úniky pamäte pri spracovaní dopytov
Na zabezpečenie spoľahlivého vymazávania všetkých objektov alokovaných na hromade, sme využili možnosť Qt, ktorá ponúka priradenie rodiča alokovaným objektom a ich automatické vymazanie po vymazaní rodiča. Všetky alokované objekty majú priradeného ako rodiča objekt ProxyHandlerSession, alebo iný objekt, ktorý má rodiča ProxyHandlerSession, alebo bude vymazaný skôr ako ProxyHandlerSession. Objekty ProxyHandlerSession sú vymazávané pri ukončení spracovania dopytu.

Uchovávanie sťahovaných odpovedí na dopyty v pracovnej pamäti
Aby nebolo potrebné uchovávať celý sťahovaný objekt v pracovnej pamäti, sme navrhli a implementovali riešenie, ktoré umožňuje priebežné ukladanie dát po častiach v súboroch na disku. Teda jeden sťahovaný objekt je reprezentovaný viacerými súbormi v cache. Po ukončení zápisu jednej časti do súboru v cache, sa dáta, ktoré obsahuje vymažú z pracovnej pamäti.

Tento prístup je realizovaný nasledovne:

1. Sťahované dáta sa zapisujú do malých blokov dát v pracovnej pamäti, ktoré čítajú jednotlivé výstupné kanále

2. Jedným z výstupných kanálov je zápis do cache, po zapísaní stanovenej hranice veľkosti dát do cache sa uzavrie zapisovaný súbor a zapisuje sa do nového

3. Súbor, pre ktorý bol ukončený zápis, nahradí prvý blok dát v pracovnej pamäti, ktorý je v súbore obsiahnutý

4. Ak ďalšie bloky, ktoré sú obsiahnuté v súbore, nečítajú žiadne výstupné kanále, vymažú sa

5. Ak sú tieto bloky dát čítané jedným alebo viacerými výstupnými kanálmi, do objektu prvého bloku dát v pracovnej pamäti, ktorý zapísaný súbor už neobsahuje, sa zapíšu ID čísla týchto výstupných kanálov

   - Po prístupe posledného z týchto výstupných kanálov na daný blok dát sa vymažú všetky bloky dát v pracovnej pamäti, ktoré súbor nahradil

4.1.4 Testovanie

Testovanie prebiehalo manuálne, návštevou väčšieho množstva webových stránok. Automatizované testovanie je plánované v ďalších šprintoch.

Po aplikovaní spomínaných opatrení bolo postrehnuteľné zlepšenie stability, správy pamäte a aj rýchlosti aplikácie.

4.2 Jednotkové testovanie

#OWNNET-134 Rozdelenie do modulov, zodpovedná osoba: Matúš Tomlein #OWNNET-135 Jednotkové testovanie, zodpovedná osoba: Martin Lipták

4.2.1 Úloha

Analýza, návrh a implementácia možností jednotkového testovania v projekte.

4.2.2 Analýza

Preskúmali sme nástroje umožňujúce unit testovanie v Qt. Je potrebné vytvoriť testovací podprojekt a kód podprojektu prepojiť s kódom hlavného podprojektu. To je zložitejšia časť, pretože aby bolo možné toto prepojenie realizovať, musí byť hlavný podprojekt rozdelený na knižnice (ktoré sa použijú v testovacom podprojekte). Existuje aj ďalšia možnosť - staticky vložiť súbory (pridať do súboru .pro) z hlavného podprojektu do testovacieho podprojektu, ale toto nie je z hľadiska čistoty kódu pekné riešenie.

K jednotkovému testovaniu v Qt neexistovalo veľa dokumentácie a komunikácie na fórach. Toto môže byť spôsobené tým, že pomocou Qt sa väčšinou vyvíjajú desktopové aplikácie, kde sa väčšinou testuje ručne (napriek všeobecne známym prínosom automatizovaného testovania). Jednotkové testovanie takisto vyžadovalo rozsiahly refaktoring kódu. Aplikácia tiež už obsahovala viac kódu, kde bola vysoká možnosť ďalšieho refaktoringu ešte po nevyhnutnom refaktoringu pre potreby testov a tie by bolo treba viackrát prepísať. Všeobecne vývojár pracuje skôr s jednotkovým testom ako integračným, ale k existujúcemu kódu má väčší zmysel písať integračné testy pre umožnenie neskoršieho refaktoringu na vyšších úrovniach ako jednotkové testy. Preto sme jednotkové testovanie na začiatku považovali za menej prioritnú úlohu.

4.2.3 Návrh

Pri návrhu jednotkového testovania sme použili obidva spôsoby zahrnutia kódu aplikácie do testovacieho podprojektu. Na základe týchto možností sme navrhli dva spôsoby jednotkového testovania.

- *Testovanie tried* - samostatné triedy, ktoré sa staticky vložia do testovacieho projektu
- *Testovanie modulov* - moduly, ktoré sa pri spustení testu načítajú a pomocou stubov sa jednotkovo otestuje ich funkčnosť

Bolo by možné využiť nástroje pre jednotkové testovanie aj na vykonanie integračného testovania. V tom prípade by celá aplikácia musela fungovať ako knižnica a triedy vykonávajúce komunikáciu po sieti a prácu s databázou by boli od aplikácie oddelené a pri testoch nastubované. Pre túto možnosť sme sa nerozhodli kvôli potrebe veľkých a neprirodzených zásahov do kódu aplikácie, nemožnosti vykonať používateľské integračné testovanie a problémoch s niektorými triedami aplikácie, ktorých pripojenie na sieť nie je možné nastubovať (QtWebkit jadro webového prehliadača použité pri prednačítavaní odkazov). Výhodou riešenia by bola vyššia rýchlosť testovania a možnosť použitia ladenia pri testovaní.

4.2.4 Implementácia

Pre plnohodnotné využívanie unit testov bolo potrebné rozdelenie aplikácie do modulov a vytvorenie stubov, aby mohli tieto moduly samostatne pracovať. Potom sme pridali testovací podprojekt a

vytvorili sme jeden príklad pre test samostatnej triedy a príklad pre test modulu. Pre modul bolo potrebné vytvoriť viac stubov. Stuby boli vytvorené, ale boli dôležité iba pre kompiláciu projektu, žiadne ďalšie možnosti pre testovanie neobsahovali. Aplikácia ešte stále potrebovala významný refaktoring, aby sa testy dali pohodlne používať s rozumne fungujúcimi stubmi.

Vytvorený jednotkový test modulu slúžil ako ukážka. Vytvorený jednotkový test triedy CommunicationManager bolo použitý na skutočný TDD vývoj logiky protokolu komunikácie po sieti. Ostatní členovia tímu jednotkové testy ešte nevyužili, pretože nemali väčšie skúsenosti s testovaním.

4.3 Integračné testovanie

#OWNNET-132 Nastubovanie siete, databázy a úložiska, zodpovedná osoba: Martin Lipták #OWNNET-133 Integračné testovanie, zodpovedná osoba: Martin Lipták #OWNNET-137 Testovanie Javascriptu, zodpovedná osoba: Martin Lipták

4.3.1 Úloha

Analýza, návrh a implementácia možností integračného testovania v projekte.

4.3.2 Analýza

Dlho sme rozmýšľali nad spôsobom testovania desktopovej aplikácie, ktorá intenzívne pracuje so sieťou. Používateľské rozhranie pozostávajúce z okna v operačnom systéme obsahuje len logo aplikácie a nastavenia, pri testovaní má význam sa sústrediť skôr na správanie sa aplikácie na sieti. Pôvodný plán bol použiť nástroje na jednotkové testovanie a nastubovať všetky časti aplikácie, ktoré komunikujú so sieťou. Toto sme ale z dôvodov opísaných v časti 4.4 nerealizovali. Rozhodli sme sa aplikáciu otestovať radšej zvonka, čo okrem vyriešenia problémov prvého prístupu umožní otestovať celú aplikáciu bez stubovania akýchkoľvek jej častí. Aplikácia musí samozrejme pri testovaní bežať v testovacom prostredí, kde nastavíme iné parametre siete a iné cesty v súborovom systéme ako v produkčnom prostredí, aby sme mohli s aplikáciou komunikovať a zabezpečiť izolovanosť jednotlivých testov. Testovací framework musí teda vedieť aplikáciu pred každým testom spustiť vo vyčistenom (vymazané súbory z predchádzajúceho behu aplikácie) testovacom prostredí, komunikvoať s ňou a po skončení testu ju ukončiť.

Zvažovali sme viac testovacích frameworkov a knižníc. Prvou možnosťou boli nástroje používané v komunite programovacieho jazyka Ruby. RSpec je testovací framework s intuitívnym DSL a bohatými možnosťami vytvárania a refaktorovania testov. Capybara je knižnica, ktorá spolupracuje s RSpec a pridá možnosť používateľského testovania. Capybara-webkit umožní otvárať webovú stránku na pozadí s podporou Javascriptu, čo potrebujeme pre otestovanie portálu, ktorý je postavený na javascriptovom frameworku. Druhou možnosťou testovania by bolo použitie nástrojov v Javascripte ako NodeJS a PhantomJS. Nástroj NodeJS by sme použili na nastavenie testovacieho prostredia, spustenie aplikácie a testov. Pomocou PhantomJS by sme vykonali používateľské testy s podporou Javascriptu. Nástroje v Ruby sme zvolili kvôli skúsenostiam člena tímu, ktorý pripravoval testovací framework s tímto nástrojom, s testovacími nástrojmi v Javascripte nemal skúsenosti nikto v tíme.

Javascript by bolo možné testovať aj samostatne pomocou jednotkových testov alebo používateľských testov v prehliadači s nastubovaním celého servera. Tento prístup má výhodu v tom, že by vývojár pracujúci na frontende mohol pracovať nezávisle na vývojárovi, ktorý implementuje serverové API. Vývojár serverového API by tiež mohol pracovať nezávisle, pretože by použil testy na API a vyvíjal gez frontendu. Avšak týmto by sme nezaručili, že výsledné časti budú navzájom medzi sebou komunikovať, nejde teda o plné integračné testovanie.

4.3.3 Návrh

Navrhli sme viac vrstiev integračného testovania.

- *Používateľské testovanie* - simulácia práce používateľa s aplikáciou s podporou Javascriptu
- *Testovanie požiadaviek* - posielanie HTTP požiadaviek na serverové API služieb

- *Testovanie komunikácie* - posielanie multicastových správ po sieti a testovanie odpovede aplikácie na tieto správy
- *Testovanie databázy* - testovanie pokročilých možností databázy a migrácií
- *Testovanie helperov* - testovanie pomocných funkcií testovacieho frameworku

Používateľské testovanie je najvyššia úroveň testovania a komplexne testuje celú aplikáciu z hľadiska používateľa. Testy musia v rozumnej miere (v prípade veľkého množstva kombinácií je možné otestovať napríklad len jednu alebo dve a ďalšie testovať na nižšej a rýchlejšej úrovni testovania) pokryť všetky scenáre použitia aplikácie.

Testovanie požiadaviek musí pokryť všetky služby serverového API a všetky v rozumnej miere všetky kombinácie práce s týmito službami. Táto úroveň testovania umožní TDD prístup k vývoju serverových služieb a takisto rýchlejšie odhalenie chyby v prípade zlyhávajúceho používateľského testu. Na testovanie serverových službieb sa dajú použiť aj jednotkové testy modulov. Ich výhodou je vyššia rýchlosť, možnosť použitia ladenia a možnosť užšej previazanosti s modulmi (kontrola informácií v session rámci aplikácie). Nevýhodou je ich nepoužiteľnosť pri refaktoringu na vyšších úrovniach kódu aplikácie, neúplné integračné testovanie (bez častí zabezpečujúcich komunikáciu medzi modulmi a jadrom aplikácie) a menej prehľadný kód C++ oproti kódu Ruby.

Testovanie komunikácie zahŕňa často dlho trvajúce testy s čakaním na periodickú sieťovú aktivitu aplikácie. Testovací framework si pomocou testovacej multicast domény vymieňa s aplikáciou správy, overuje ich obsah a fungovanie služieb aplikácie pri rôznych stavoch na sieti.

4.3.4 Implementácia

Framework pre integračné testovanie spustí aplikáciu pred každým testom v testovacom prostredí predaním niekoľkých premenných prostredia a nastavením niekoľkých možností v konfiguračnom súbore aplikácie.

- OWNET_LISTEN_PORT
- OWNET_PROXY_HOST_NAME
- OWNET_PROXY_PORT
- OWNET_DISABLE_GUI
- OWNET_INI_DIR

Premenná OWNET_LISTEN_PORT obsahuje číslo portu, na ktorom počúva proxy server. Premenné OWNET_PROXY_HOST_NAME a OWNET_PROXY_PORT obsahujú nastavenie externého proxy servera pre aplikáciu, ktorý bude spustený testovacím frameworkom a zabezpečí stubovanie siete pri prístupe na externé webové stránky. Možnosť OWNET_DISABLE_GUI zabezpečí nezobrazenie okna aplikácie pri testovaní. Premenná OWNET_INI_DIR obsahuje cestu k adresári s konfiguráciou aplikácie. V konfigurácii sa nastavujú ďalšie parametre testovacieho prostredia.

- data_folder_path
- resources_folder_path
- multicast_group_address

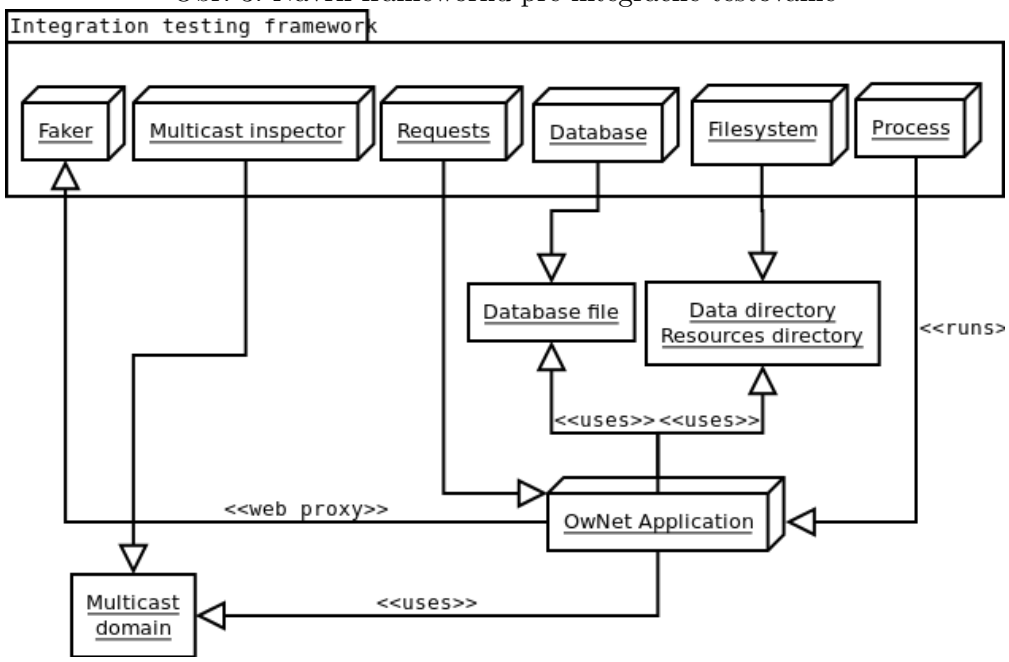
- multicast_port

data_folder_path a resources_folder_path nastavujú cesty k priečinku s používateľskými dátami resp. so zdrojmi aplikácie. Tieto priečinky sú vytvorené pri testovaní a sú do nich nakopírované všetky potrebné súbory. multicast_group_address a multicast_port nastavujú multicastovú doménu pre testovanie. Pri najbližšom refaktoringu sa aj ostatné premenné prostredia presunú do konfiguračného súboru (okrem cesty ku konfigurácii).

Po skončení testu je aplikácia ukončená a testovacie priečinky sú vymazané. Pri testovaní je možné spustiť viac aplikácií OwNet proxy na rôznych portoch a testovať napríklad ako medzi sebou komunikujú.

Testovací framework obsahuje moduly pre komunikáciu s aplikáciou requests, faker a multicast inspector. Requests umožní pohodlné volanie API servera z testov. Faker je proxy server, ktorý testom umožní podstrčiť serveru falošné webové stránky. Aplikácia pristupuje na web iba cez faker. V prípade prístupu na adresu, ktorá nebola podstrčená testom, je vyvolaná výnimka a test skončí. Multicast inspector posiela a prijíma správy v testovacej multicast doméne. Modul database umožní testom prístup k databáze aplikácie. Obrázok č. 3.

Obr. 3: Návrh frameworku pre integračné testovanie



Vytvorili sme viac ukážkových testov (okolo 40). Väčšina z nich bola označená ako pending, čo znamená, že nie sú dokončené, pretože obsahovali zakomentované časti alebo iba komentáre s poznámkami, ako danú funkcionalitu otestovať. Niektoré z týchto testov zlyhali a nebolo jasné, či je chyba na strane aplikácie alebo na strane testovacieho frameworku. Toto sme nechali na vyriešenie členom tímu, ktorí boli zodpovední za danú časť kódu. Okrem ukážkových testov ešte neboli vytvorené ďalšie testy, pretože sa členovia tímu nestihli zoznámiť s integračným frameworkom a so spôsobom vývoja podľa testov.

4.4 Implementácia komunikácie a voľby servera

#OWNNET-147 Implementácia navrhnutého protokolu, zodpovedná osoba: Martin Lipták

4.4.1 Úloha

Implementácia protokolu komunikácie a voľby servera na sieti.

4.4.2 Analýza

Analýza bola vykonaná v minulom šprinte (časť 3.9).

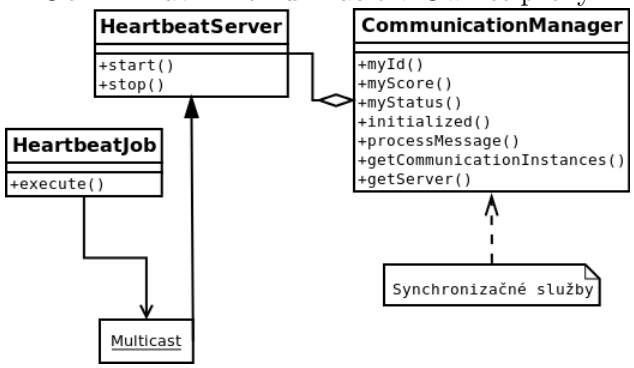
4.4.3 Návrh

Návrh protokolu bol vykonaný v minulom šprinte (časť 3.9).

Protokol sme rozšírili o stav inštancie initializing. Keď sa na sieť pripojí nová inštancia OwNet proxy, ktorá má vyššie skóre ako ktorákoľvek pôvodná inštancia na sieti, namiesto okamžitého prevzatia kontroly nad sieťou, počká niekoľko periód v stave initializing. Ostatné inštancie sa pri prítomnosti stavu initializing dočasne prestanú synchronizovať, najlepšia inštancia v stave initializing sa zosynchronizuje s aktuálnym serverom a zmení svoj stav na server, vtedy sa synchronizácia opäť rozbehne, ale s novým serverom.

Trieda HeartbeatJob periodicky posiela na sieť správy. Trieda HeartbeatServer prijíma správy zo siete a posiela ich triede CommunicationManager. Trieda CommunicationManager obsahuje logiku komunikačného protokolu a poskytuje informácie službám, ktoré vykonávajú synchronizáciu. Najdôležitejšie informácie sú, ktorý počítač je aktuálny server a či je možná synchronizácia (je prítomný server a najlepšia inštancia nie je v stave initializing). Obrázok č. 4.

Obr. 4: Návrh komunikácie v OwNet proxy



4.4.4 Implementácia a testovanie

Pri implementácii sme využili možnosti Qt frameworku. Použili sme triedy QUdpServer (z neho dedí trieda HearbeatServer) a QUdpSocket (v triede HeartbeatJob), ktoré dokážu vytvoriť server resp. posielať UDP správy po sieti a umožňujú nastaviť multicast adresu a port (TCP komunikácia cez multicast nefunguje, pretože jeden počítač nemôže naviazať to isté spojenie na viac počítačov v sieti).

Pri implementácii triedy CommunicationManager sme použili jednotkový test, kde sme dopredu zadefinovali, aké stavy má trieda mať po prijatí správ v rôznom poradí a s rôznymi oneskoreniami.

Pre celkové otestovanie komunikácie sme použili integračný test, v ktorom sa však prejavil problém, ktorý sme jednotkovým testom neodhalili. Problém sme už nestihli do konca šprintu odstrániť, úloha zostala nedokončená a bola prenesená do ďalšieho šprintu.

4.5 História prehliadania

Úloha #OWNNET-133, podúlohy #OWNNET-151, #OWNNET-152 zodpovedná osoba: Martin Konôpka

4.5.1 Úloha

Úlohou je uchovávať prechody používateľa medzi stránkami v databáze. Vytvoriť graf návštev a prechodov medzi stránkami.

Úloha je rozdelená do dvoch podúloh:

- **Vytvoriť databázu (OWNET-51)** - Zadefinovať dátový model a vytvoriť databázu pre uchovávanie histórie.
- **Zachytávať históriu do databázy (OWNET-152)** - Plniť databázu údajmi o používateľových návštevách stránok.

4.5.2 Analýza

Používateľ ,,cestuje“ po Webe, čo potrebujeme vhodne zaznamenať. Používateľ navštívi stránku, klikne na odkaz a pokračuje na ďalšiu stránku. Históriu môžeme uchovať v grafovej štruktúre, ktorej vrcholy budú stránky a hrany prechody medzi stránkami.

O každej stránke môžeme uchovávať jej URL adresu a titulok stránky. O návštevách používateľa na stránkach dátum kedy stránku navštívil a početnosť návštev. O prechodoch medzi stránkami podobne dátum posledného prechodu a počet, koľko krát prechodom prešiel.

Zaznamenanie histórie iniciuje script súbor vložený do stránky. Oznámenie o návšteve a prechode medzi stránkami odchytí modul histórie prehliadania a zapíše tento fakt do databázy.

4.5.3 Návrh

Pri návrhu sme vychádzali z implementácie v .NET riešení OwNetu.

V databáze potrebujeme uchovať informácie o stránkach, ich prepojeniach a používateľových návštevách stránok a prechodmi medzi nimi. Obrázok 4.5.3 znázorňuje údajový model pre túto funkcionalitu.

Pre implementovania oznamovania návštev a prechodov stránok využijeme oznamovanie o navštívení stránky z úlohy Prednačítavanie z predchádzajúcich šprintov.

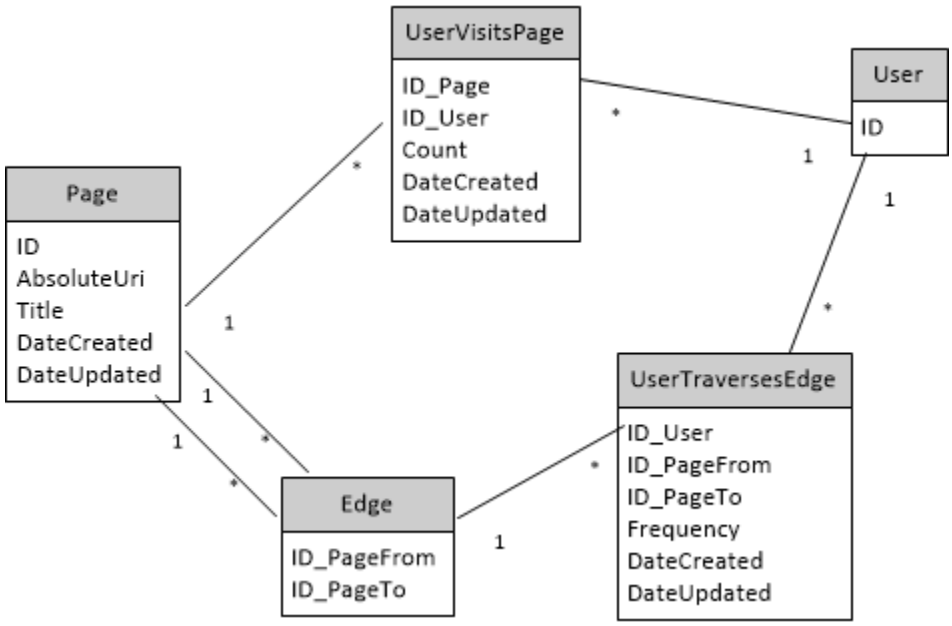
4.5.4 Implementácia

Úloha bola implementovaná v C++ , Qt a JavaScript. Vkladanie script súboru bolo zabezpečené v úlohe *OWNET-48*. Kód script súboru bol znovupoužitý z pôvodného riešenia.

Formát volania proxy aplikácie o návšteve stránky s URL (prípadne prichádzajúci zo stránky URLFrom) je nasledujúci:

http://inject.ownet/api/history/visit/?page=URL&ref=URLFrom

Modul histórie HistoryMdule v proxy aplikácii odchytí volanie a pridá nové záznamy do databázy. Ak stránka neexistuje, vytvorí ju. Naviaže používateľa návštevou (*UserVisitsPage*). Ak je špecifikovaná aj predchádzajúca stránka (parameter *ref*), tak sa vytvorí hrana a prechod používateľa cez hranu (*UserTraversesEdge*).



Obr. 5: Údajový model histórie prehliadania.

4.5.5 Testovanie

Testovanie tejto funkcionality bolo zebezpečené ručným testovaním prehliadania stránok a sledovaním vykonávania kódu aplikácie jej prerušovaním. Podľa úspešnosti operácií zistenia návštevy a výberu odkazov boli vytvárané záznamy do databázy. Vhodný spôsob, ktorým by bolo možné tuto funkcionalitu otestovať sú integračné testy. Tie sa však nepodarilo dostať do behu kvôli problémom s inštaláciou. Možným pokračovaním úlohy je využitie integračného testovania.

4.6 Databáza pre rozhovory medzi používateľmi

#OWNNET-125, zodpovedná osoba: Karol Balko

4.6.1 Úloha

Vytvorenie databázy pre ukladanie rozhovorov medzi používateľmi.

4.6.2 Analýza

Keďže rozhovory medzi používateľmi poskytujú nielen informáciu o samotnej správe, ale aj o osobe, ktorá posiela správu, takisto však existuje aj možnosť komentovať správu, avšak len do prvej úrovne.

4.6.3 Návrh a implementácia

Ako databázu pri našom projekte využívame SQLite v ktorej tvoríme potrebné tabuľky pomocou tzv. migrácií. Implementácia migrácií je zabezpeèená pomocou QT, ktorá vlastne spustí SQL skript uvedený v súbore migrácie.

V súvislosti so všetkými potrebami implementácie rozhovorov je potreba aby naša tabuľka obsahovala nasledujúce hodnoty: ID správy, dátum vytvorenia správy, ID používateľa, ktorý vložil správu, samotný text správy a ID rodiča, ktoré slúži na identifikáciu správy, ktorá je komentovaná

4.6.4 Testovanie

Testovanie tejto databázy nebolo v čase jej vytvorenia možné z dôvodu, že ešte neexistovala implementácia samotného rozhovoru používateľov, avšak musela byť vytvorená ako vodítko, ako sa správy majú ukladať.

4.7 Implementácia GUI časti skupín

Podúlohy #OWNNET-131, zodpovedná osoba: Andrea Šteňová

Úloha

Implementujte základnú funkcionalitu používateľských skupín portálu, ktorá zahŕňa vytváranie, editáciu a mazanie skupín. Návrh a analýza prebehla v predchádzajúcom šprinte.

Implementácia

Základná funkcionalita bola implementovaná v Backbone.js na základe vytvoreného api v QT module. Api komunikuje na adrese ,,my.ownet/api/groups“.

Pre základnú funkcionalitu boli vytvorené nasledujúce súbory:

- GroupsModel - model predstavujúci používateľskú skupinu, je prepojený na api v QT module
- GroupsView - zobrazenie, zabezpečuje prepojenie medzi modelom a zobrazenými obrazovkami, volá funkcie z api v QT module
- šablóny pre interakciu s používateľom

Boli implementované nasledujúce funkcie:

- show - z QT modulu získava zoznam všetkých skupín v databáze, a zobrazuje ich používateľovi s možnosťou zobrazenia skupiny
- showGroup - zobrazenie konkrétnej skupiny, dáva možnosť prejsť na editáciu skupiny
- createGroup - vytvorenie novej skupiny s nastavením možností prihlasovania do skupiny
- saveGroup - uloženie skupiny v QT moduly
- editGroup - editácia skupiny, kontroluje sa aby skupiny mohol editovať len jej administrátor
- deleteGroup - vymazanie skupiny, kontroluje sa aby skupinu mohol zmazať len jej administrátor

Pokročilejšie funkcie so skupinami sa budú implementovať v nasledujúcom šprinte.

4.8 Vytvorenie služby pre používateľské skupiny v Qt

Úloha #OWNNET-160, podúloha #OWNNET-130, zodpovedná osoba: Marek Láni

4.8.1 Úloha

Navrhnite a implementujte príjem a spracovanie požiadaviek z GUI týkajúcich sa používateľských skupín.

4.8.2 Analýza a návrh

Oproti .NET verzii by mali byť používateľské skupiny prepracovanejšie, to znamená ponúkať viac funkcionality. Touto funkcionalitou sú napríklad zaheslené skupiny, schvalovanie členstva v skupine adminom alebo pridávanie administrátorov skupiny. Samozrejme služba pre skupiny má ponúkať aj základné funckie ako vytvorenie, zmazanie, editovanie, zobrazenie detailu skupiny a tiež zobraziť zoznam skupín. Všetká funkcionalita spojená so skupinami by mala byť používateľovi sprístupnená až po prihlásení sa. Rovnako previazanie dát so skupinami by bolo vhodné rozšíriť a takmer všetky dáta generované používateľmi chceme previazať s používateľskými skupinami. Tabuľka v databáze pre skupiny vychádza z návrhu databázového modelu z predchádzajúceho šprintu.

Rovnako ako pre používateľov je aj pre každú skupinu nutné generovať náhodné a s čo možno najväčšou pravdepodobnosťou unikátne id.

Na zabezpečenie správnej funkcionality a bezpečnosti sme identifikovali funkcie, ktoré budú bližšie popísané v časti iplementácia.

4.8.3 Implementácia

Úloha bola implementovaná v C++ a Qt. Generovanie unikátneho id pri vytváraní skupiny je uskutočnené za pomoci funkcie QHash, ktorej vstupný parameter je reťazec znakov vytvorený spojením loginu používateľa a aktuálneho času. Rozhranie a funkcie služby pre manažment skupín sú nasledovné:

- *isAdmin:*

  Parametre tejto funkcie sú id používateľa a id skupiny, na základe ktorých sa v databáze vyhľadá záznam v tabuľke groupAdmins. Ak existuje funkcia vráti hodnotu pravda, čo znači, že daný používateľ je adminom danej skupiny. V opačnom prípade vráti nepravdu.

- *isMember:*

  Parametre tejto funkcie sú id používateľa a id skupiny, na základe ktorých sa v databáze vyhľadá záznam v tabuľke groupMembers. Ak existuje funkcia vráti hodnotu pravda, čo znači, že daný používateľ je členom danej skupiny. V opačnom prípade vráti nepravdu.

- *create:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa.

  Zloženie posielaných dát musí byť nasledovné:

  {

name:'name_of_group',(required)
description:'description',(required)
has_password:'0/1', (required)
has_approvement:'last_name',(required)
group_type:'type_id',(required)
password:'pass' (required if has_password == 0)

}

Odpoveďou je HTTP status "201 Createdä výsledkom je vytvorená skupina v databáze. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *show:*

  Táto akcia je volaná parametrom id, ktorý reprezentuje id skupiny. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/groups/:id Ako odpoveď je vrátené JSON s údajmi o danej skupine a HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

  Na uskutočnenie tejto akcie musí byť používateľ prihlásený a rovnako byť členom danej skupiny.

- *index:*

  Táto akcia je volaná bez parametrov. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/groups Ako odpoveď je vrátené JSON pole s údajmi o skupinách a HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

  Na uskutočnenie tejto akcie musí byť používateľ prihlásený a rovnako byť členom danej skupiny.

- *edit:*

  Akcia sa zavolá PUT HTTP požiadavkov na url adresu my.ownet/api/groups. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa. Používateľ, ktorý sa pokúša pustiť túto akciu musí byť taktiež administrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

  name:'name_of_group',(optional)
  description:'description',(optional)
  has_password:'0/1', (optional)
  has_approvement:'last_name',(optional)
  group_type:'type_id',(optional)
  password:'pass' (required if has_password == 0)

  }

  Odpoveďou je HTTP status "200 OKä výsledkom zmenené údaje o skupine. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *del:*

  Táto akcia je volaná parametrom id, ktorý reprezentuje id skupiny. Akcia sa zavolá DELETE HTTP požiadavkov na url adresu my.ownet/api/groups/:id Ako odpoveď je vrátený HTTP

status "204 No Contentä výsledkom je zmazanie skupiny. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

Na uskutočnenie tejto akcie musí byť používateľ prihlásený a rovnako byť administrátorom danej skupiny.

- *joinGroup:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/joinGroup. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa. Touto akciou sa používatel stene členom skupiny avšak bolo potrebné vyriešiť možnosť, kedy bola skupina zaheslená resp. kedy jej bola nastavené schvalovanie používateľov. V prípade, ze bola zaheslená, kontroluje sa, či sa odoslané heslo zhoduje s heslom zadaným používateľom. V prípade schvalovania, sa používateľovi nastavý status, že je čakateľom na členstvo a tento status môže byť zmenený jedine aministrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

  group_id:'group_id:',(required)
  password:'pass',(required if group has_password)

  }

  Odpoveďou je HTTP status "200 OKä výsledkom vytvorené členstvo v skupine, alebo zaradenie medzi čakateľov na členstvo. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *getApprovements:*

  Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/groups/getApprovements a slúži pre zobrazenie čakateľov na členstvo v danej skupine. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa a používateľ musí byť taktiež administrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

  group_id:'group_id:',(required)

  }

  Odpoveďou je HTTP status "200 OKä JSON pole čakateľov na členstvo v skupine. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *approveUser:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/approveUser a sluzi na schválenie členstva pre čakateľa. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa a používateľ musí byť taktiež administrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

group_id:'group_id:',(required)
user_id:'user_id',(required)

}

Odpoveďou je HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *addAdmin:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/addAdmin a slúži na pridanie Admina pre skupinu. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa a používateľ musí byť taktiež administrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

  group_id:'group_id:',(required)
  user_id:'user_id',(required)

  }

  Odpoveďou je HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *addAdmin:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/addAdmin a slúži na pridanie Admina pre skupinu. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa a používateľ musí byť taktiež administrátorom skupiny.

  Zloženie posielaných dát musí byť nasledovné:

  {

  group_id:'group_id:',(required)
  user_id:'user_id',(required)

  }

  Odpoveďou je HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *getUsersGroups:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/getUsersGroups a slúži na zobrazenie skupín, ktorých je používateľ členom. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa.

  Odpoveďou je HTTP status "200 OKä JSON pole obsahujúce údaje o skupinách, ktorých členom používateľ je. V prípade chybnej požiadavky je návratová hodnota "403 Forbidden".

- *getGroupUsers:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/groups/getGroupUsers a slúži na zobrazenie používateľov danej skupiny. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa.

Zloženie posielaných dát musí byť nasledovné:

{

group_id:'group_id:',(required)

}

Odpoveďou je HTTP status "200 OKä JSON pole obsahujúce údaje o skupinách, ktorých členom používateľ je. V prípade chybnej požiadavky je návratová hodnota "404 Bad Request".

4.8.4 Testovanie

Testovanie prebiehalo za pomoci nástroja POSTER, doplnok prehliadaču firefox, ktorý dokáže generovať a posielať HTTP požiadavky. Vzhľadom na zdĺhavosť testovania boli odtestované a následne odladené iba akcie create, edit, delete. Podrobnejšie testovanie je na pláne v ďalších šprintoch pomocou automatizovaných Unit a integračných testov.

4.9 Vytvorenie služby pre používateľské skupiny v Qt

Úloha #OWNNET-161, podúloha #OWNNET-128, zodpovedná osoba: Marek Láni

4.9.1 Úloha

Navrhnite a implementujte službu pre vytváranie správ v rámci skupín.

4.9.2 Analýza a návrh

Oproti .NET verzii by bolo vhodné správy ukladať v databáze a previazať ich na skupiny. Taktiež chceme použiť viacúrovňové správy, t.j. komentovanie správ (statusov).

Na zabezpečenie správnej funkcionality sme identifikovali funkcie, ktoré budú bližšie popísané v časti iplementácia.

4.9.3 Implementácia

Úloha bola implementovaná v C++ a Qt. Rozhranie a funkcie služby pre správy sú nasledovné:

- *create:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/messages. Predpokladom pre úspešné vykonanie akcie je prihlásenie sa používateľa a takisto členstvo v skupine v ktorej chce správu vytvoriť

  Zloženie posielaných dát musí byť nasledovné:

  {

  text:'text',(required)
  parent_id:'parent_id',(required, 0 if message has no parent)
  group_id:'group_id', (required)

  }

  Odpoveďou je HTTP status "201 Createdä výsledkom je vytvorená správa v databáze. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *index:*

  Táto akcia je volaná bez parametrov. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/messages. Na uskutočnenie akcie musí byť používateľ prihlásený a rovnako byť členom danej skupiny.

  Zloženie posielaných dát musí byť nasledovné: {

  group_id:'group_id', (required)

  }

  Ako odpoveď je vrátené JSON pole so správami patriacimi danej skupine a HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *del:*

  Táto akcia je volaná parametrom id, ktorý reprezentuje id správy. Akcia sa zavolá DELETE HTTP požiadavkov na url adresu my.ownet/api/messages/:id Ako odpoveď je vrátený HTTP status "204 No Contentä výsledkom je zmazanie správy. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

  Na uskutočnenie tejto akcie musí byť používateľ prihlásený a rovnako byť administrátorom danej skupiny alebo autorom správy.

4.9.4 Testovanie

Testovanie je na pláne v ďalších šprintoch pomocou automatizovaných Unit a integračných testov.

4.10 Vytvorenie služby pre hodnotenia stránok Qt

Úloha #OWNNET-164, podúloha #OWNNET-179, zodpovedná osoba: Marek Láni

4.10.1 Úloha

Navrhnite a implementujte službu pre vytváranie hodnotení stránok.

4.10.2 Analýza a návrh

Hodnotenia stránok je vhodné implementovať ako v .NET verzii, kedy sa každé hodnotenie bude viazať na stránku bez väzby na skupinu.

Na zabezpečenie správnej funkcionality sme identifikovali funkcie, ktoré budú bližšie popísané v časti iplementácia.

4.10.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Základným predpokladom pre úspešné vykonanie akcií je prihlásenie sa používateľa.

Rozhranie a funkcie služby pre hodnotenia sú nasledovné:

- *create:*

  Akcia sa zavolá POST HTTP požiadavkov na url adresu my.ownet/api/ratings.

  Zloženie posielaných dát musí byť nasledovné:

  {

  absolute_uri:'absolute_uri',(required)
  value:'value',(required)

  }

  Odpoveďou je HTTP status "201 Createdä výsledkom je vytvorené hodnotenie v databáze. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *index:*

  Táto akcia je volaná bez parametrov. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/ratings.

  Ako odpoveď je vrátené JSON pole s detailami hodnoteniami a HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *del:*

  Táto akcia je volaná parametrom id, ktorý reprezentuje id správy. Akcia sa zavolá DELETE HTTP požiadavkov na url adresu my.ownet/api/ratings/:id Ako odpoveď je vrátený HTTP status "204 No Contentä výsledkom je zmazanie správy. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

  Na uskutočnenie tejto akcie musí byť používateľ prihlásený a rovnako byť administrátorom danej skupiny alebo autorom správy.

- *show:*

  Táto akcia je volaná s parametrom, ktorý reprezentuje URL stránky. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/ratings/:absolute_uri Ako odpoveď je vrátený HTTP status "200 OKä JSON pole s detailami jednotlivých hodnotení danej stránky. V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

4.10.4 Testovanie

Testovanie je na pláne v ďalších šprintoch pomocou automatizovaných Unit a integračných testov.

5 Šprint Xela

5.1 Dokončenie synchronizácie

#OWNNET-170, zodpovedná osoba: Matúš Tomlein

5.1.1 Úloha

Dokončite prácu na synchronizácii, ktorá bola vykonaná v predchádzajúcom šprinte. Navrhnite a implementujte synchronizačné služby na komunikáciu klientov na sieti, získavanie a aplikovanie synchronizačného žurnálu medzi klientami.

5.1.2 Analýza

V predchádzajúcom šprinte bola implementovaná automatická tvorba aktualizačného žurnálu pri vykonávaní UPDATE, INSERT a DELETE dopytov v databáze.

Nie sú ešte definované služby, cez ktoré si budú klienti posielať aktualizácie. Je potrebné vytvoriť funkcie na získavanie len tých zmien zo žurnálu, ktoré dopytujúci klient neobsahuje.

Keďže ešte nebolo implementované rozhranie pre volanie služieb medzi klientami, implementujte aj takéto rozhranie na jednoduché posielanie dopytov na server alebo na iný klient na sieti a vrátenie odpovedí.

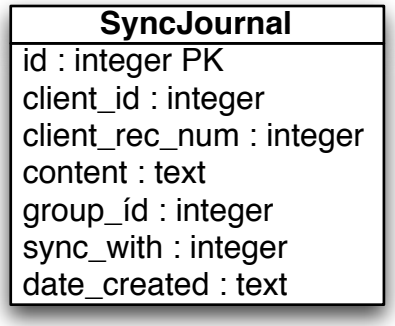
5.1.3 Návrh a implementácia

Volanie služieb na serveri
Pre volanie služieb na serveri bol implementovaný nový modul do aplikácie s názvom *ServerModule*. *ServerModule* ponúka služby, pomocou ktorých môžu moduly v rámci aplikácie volať služby na serveri a na iných klientoch rovnako ako by volali služby na lokálnej proxy. ServerModule ponúka dve služby *server* a *clients*.

Pre zavolanie akcie *sync/get_updates* na serveri, je možné použiť takúto URL aresa cez proxy: *http://ownet/api/server/sync/get_updates* Pre zavolanie tej istej akcie na klientovi s ID 5, je možné použiť takúto adresu: *http://ownet/api/clients/5/sync/get_updates*.

Posielanie synchronizačného žurnálu po sieti
Synchronizácie prebieha posielaním synchronizačného žurnálu medzi klientmi a serverom vo formáte JSON. Synchronizačný žurnál obsahuje akcie, ktoré sa majú vykonať v databáze, typu INSERT, UPDATE a DELETE. Synchronizačný žurnál si klienti ukladajú v databáze v tabuľke *sync_journal*, ktorej definíciu je možné vidieť na obrázku 6.

Keďže nové položky môžu v synchronizačných žurnáloch rôznych klientov pribúdať paralelne a nedá sa predpokladať že každý klient sa bude každú pol minúty synchronizovať so serverom (môže byť napríklad mimo lokálnej siete), nie je možné stanoviť jednotný stav, ktorý by hovoril o aktuálnosti synchronizačného žurnálu pre všetkých klientov v danej pracovnej skupine. Je však možné stanoviť jednotný stav aktuálnosti záznamov, ktoré vytvoril jeden klient, keďže tie majú len jeden zdroj. Celkový stav aktuálnosti synchronizačného žurnálu je preto možné definovať čiastkovými stavmi aktuálnosti záznamov pre jednotlivých klientov. Ďalej je možné tieto stavy deliť nielen podľa klientov ale aj podľa skupiny záznamov do ktorej patria (napríklad šport, hudba). Pre posielanie stavu synchronizačného žurnálu používame formát JSON, ktorý obsahuje hierarchiu ID skupiny >ID klienta >ID posledného záznamu. Touto hierarchiou je možné opísať celkový stav žurnálu klienta pre všetky skupiny a klientov v ňom. Príklad takéhoto opisu stavu:



Obr. 6: Entita uchovávajúca synchronizačný žurnál v databáze. Atribút client_id predstavuje ID klienta, client_rec_num je poradové číslo synchronizovanej položky vytvorenej daným klientom, content obsahuje migráciu databázy vo formáte JSON, ktorá sa má vykonať, group_id je skupina do ktorej záznam patrí a sync_with je nepovinné ID klienta, ktorému sa má synchronizácia poslať.

```
{
NULL: {
12454: 1250,
54532: 1000
},
32: {
12454: 300,
54532: 101
}
}
```

Každý klient volá priebežne, v intervale 30 sekúnd, službu sync/updates na aktuálnom serveri, ktorá je typu POST. Ako telo dopytu jej posiela aktuálny stav aktualizácii v jeho synchronizačnom žurnáli opísaný vyššie. Server mu ako odpoveď posiela záznamy novšie ako tie uvedené v prijatom stave. Odpoveď je vo formáte JSON a je to zoznam migrácii databázy (kapitola 3.8.3) s ID klienta na ktorom migrácia vznikla a číslom migrácie na klientovi.

Aktuálny server volá každých 30 sekúnd tú istú akciu ako bola opísaná vyššie ale na všetkých klientoch práve pripojených na sieti. Ako telo dopytu im posiela stav svojho synchronizačného žurnálu a oni mu ako odpoveď posielajú zoznam nových záznamov.

5.1.4 Testovanie

Synchronizáciu sme testovali pomocou integračných testov, pričom sme vykonávali rôzne akcie v aplikácii cez služby a následne kontrolovali synchronizačné výstupy.

Testovali sme aj v reálnych podmienkach na lokálnej sieti, kedy komunikovali dve aplikácie a navzájom sa synchronizovali.

5.2 Distribuovaná cache

#OWNNET-163, zodpovedná osoba: Matúš Tomlein

5.2.1 Úloha

Navrhnite a implementujte zdieľanie cache webových stránok medzi klientmi na lokálnej sieti. Klienti by mali vedieť pristupovať na stránky uložené v cache iných klientov na lokálnej sieti bez pripojenia na internet.

5.2.2 Analýza

Architektúra proxy bola od začiatku prispôsobená na to aby bolo možné aby viacerí klienti pristupovali na rovnaké dáta sťahované jedným klientom (kapitola 2.1).

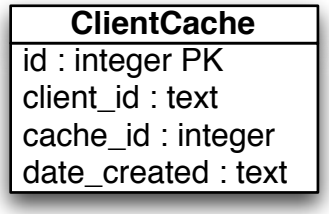
Synchronizácia databázy (kapitola 5.1) zároveň umožňuje informovanie iných klientov na sieti o nových stránkach v cache klienta.

V tejto úlohe je potrebné navrhnúť a implementovať spôsob akým budú klienti pri dopyte na webovú stránku pristupovať na cache iných klientov.

5.2.3 Návrh a implementácia

Sťahovanie z cache iného klienta má menšiu prioritu ako sťahovanie z vlastnej cache, preto pri rozhodovaní odkiaľ sa bude objekt sťahovať sa najprv skontroluje či nie je v lokálnej cache.

Ak nie je v lokálnej cache, proxy sa pozrie do tabuľky *client_caches* (obr. 7), ktorá je synchronizovaná na lokálnej sieti a obsahuje záznamy o tom aké cache sú na akých klientoch. V prípade, že sa nenájdu klienti s daným objektom, začne sa sťahovať z internetu. V opačnom prípade sa vyberú klienti, ktorí sú online a postupne sa proxy od nich snaží objekt stiahnuť. Ak zlyhá jeden klient, skúsi ďalší až kým sa nepodarí objekt od niekoho stiahnuť, alebo kým neostane žiadny klient s danou cache, kedy sa objekt stiahne z internetu.



Obr. 7: Entita uchovávajúca informáciu o tom aký cache objekt je na akom klientovi. Atribút client_id predstavuje ID klienta a cache_id je ID cache objektu, čo je jeho zašifrovaná URL adresa

Sťahovanie objektu od iného klienta prebieha tak, že sa pre daný dopyt nastaví proxy adresa na adresu klienta na ktorom je cache a zavolá sa dopyt s URL na daný webový objekt. Z pohľadu dopytujúcej proxy je až na nastavenie proxy dopytu, sťahovanie rovnaké ako by sťahovala objekt z internetu. Dopytovaná proxy nevie o tom, že dopyt prišiel od inej proxy a vybavuje ho ako keby prišiel z webového prehliadača.

5.2.4 Testovanie

Testovali sme na sieti dvoch počítačov na sade rôznych webových stránok. Neskôr plánujeme túto funkcionalitu otestovať aj integračne.

5.3 Prechod na Qt5

#OWNNET-167, zodpovedná osoba: Michal Dorner

5.3.1 Úloha

Upravit aktualnu verziu proxy na Qt 5. Analyzovať možnosti debugovania pre JavaScirptové moduly pod qt5. Nahradiť QJson knižnicu triedou QJsonDocument. Prepísať deprecated/obsolete z qt 4.8 na ekvivalenty v qt 5.

5.3.2 Analýza

Qt v novej verzii 5.0 prináša nové užitočné funkcie a nakoľko táto verzia bude dalej rozvíjaná, pre nový projekt je výhodnejšie prejsť na túto novú verziu ako ostať na 4.8. Verzia 5.0 nie je ešte vo finálnom stave, ale ako beta2, z čoho vyplýva isté riziko a potreba dôsledného otestovania.

Debugovanie JavaScirptových modulov pod qt5 je možné v plnej miere. Priamo v QtCreator IDE funguje krokovanie, zobrazovanie hodnôt premenných aj logovanie. Namiesto QJsEngine a čistého JavaScirptu je však potrebné použiť jazyk QML a prepojovacie c++ triedy. Z QML je dalej možné importovať aj čisto JavaScriptové súbory. Aj keď je debugovanie možné, vzhľadom na zložitosť celého riešenia v porovnaní alternatívou vo forme vyžitia nových možností Qt5 a c++11 pre smerovnaie požiadaviek, rozhodli sme sa JavaScirptové moduly nateraz neimplementovať.

V súčasnom riešení sa pre prácu s JSON formátom používa externá knižnica QJson. Nakoľko Qt5 má vstavané triedy pre prácu s týmto formátom, bude výhodné ich používať. QJson sa v kóde vyskytuje zbytočne na veľa miestach, výhodnejšie bude nahradiť ho používaním QVarian, sktorým sa ľahšie pracuje a prevod z JSON formátu do reprezentácie v Qt triedach vykonávať na jednom mieste - vstup a výstup zo služieb.

5.3.3 Implementácia

Implementácie prechodu na Qt 5.0 sa vykonala v nasledujúcich krokoch:

1. inštalácia knižníc pre Qt 5.0
2. konfigurácia IDE - pridanie Qt 5.0 do zoznamu verzií qt a vytvorenie kitu používajúceho túto verziu
3. prepisovanie kódu na miestach kde sa zmenilo Qt Api, kým projekt nešlo úspešne kompilovať
4. naštudovanie a prepísanie použitia tried a metód, ktoré sa v Qt 5.0 už neodporúčajú používať alebo majú lepšiu alternatívu (napríklad QRegularExpresion namiesto QRegExp)
5. nahradenie externej knižnice QJson, vstavanými triedami QJsonDocument a QJsonObject

Pri testovaní sa ukázalo že QJsonDocument má chybný export čísiel väčších ako $10\hat{6}$ do JSON formátu. Takéto čísla sa zapisujú v exponenciálnej forme a stráca sa ich presná hodnota, čo je problém pri identifikátoroch. Nakoľko Qt je aj opensource projekt, probém sa vyriešil odvodením triedy od QJsonDocument a prepísani exportnej funkcie. Tento problém sa reportoval do správy úloh Qt projektu a táto chyba by mala byť opravená v Qt verzii 5.0.1. Po opravení chyby už bude možné používať čisto vstavané Qt triedy a túto dočasnú opravu bude možné ostrániť.

5.3.4 Testovanie

Funkčnosť celého sytému musela po prechode na qt5 ostať rovnaká ako predtým. Na otestovanie sa použili existujúce integračné itesty, testujúce fungovanie proxy aj služieb. Fungovanie proxy sa následne otestovalo aj manuálne praktickým používaním proxy pri prehľiadaním webových stránok.

5.4 Router pre moduly

#OWNNET-169, zodpovedná osoba: Michal Dorner

5.4.1 Úloha

Refaktorovať a upraviť systém modulov, služieb a smerovania požiadaviek tak, aby boli nezávislé od proxy. Zmeniť sýstém smerovania požiadaviek aby bolo možné registrovať na danú url a http metódu callback funckiu. Upraviť moduly používateľov a sedení na nový sytém. Upraviť jednotkové testy a súvisiace pomocné triedy na nový systém.

5.4.2 Analýza

Aby moduly a služby neboli závislé od tried tvoriacich proxy server, je potrebné vytvoriť abstrakciu nad požiadavkou a odpoveďou. Bude potrebné stanoviť triedu, ktorá bude tvoriť rozhranie medzi spracovavaním prichádzajúcej HTTP požiadavky v proxy a časťou ktorá zabezpečuje smerovanie požiadaviek v službách. Vďaka oddeleniu týchto častí bude jendocho možné umelo vytvárať požiadavky na služby. Takto sa budú riešiť vzájomné volania medzi službami a simulovanie požidaviek pri jednotkových testoch. Na umožnenie registrácie spätných volaní požiadaviek podľa url bude vhodné využiť nové možnosti c++11 (nový štandard c++) - dátový typ std::function a labmdy.

5.4.3 Návrh

Abstrakciu nad požiadavkou bude predstavoať trieda IRequest, bude informácie od http metóde, url a tele požiadavky. Abstrakciu nad odpoveďou bude predstavoať trieda IResponse, tá bude obsahovať informácie o HTTP status kóde, popise, hlavičkách a tele odpovedi. Smerovaču požiadaviek sa na vstup zadá požiadavka vo forme IRequest a výstupom bude odpoveď vo forme IResposne.

Pre samotné smerovanie sa bude využívať trieda IRoute. V nej bude uložená informácia o vzore url a callback funkciách. Trieda Router bude uchovávať zoznam všetkých IRoute pre jednu službu. Trieda ReqestRouter bude uchovávať Router pre každú službu. Pri spracovaní požiadavky sa najprv podľa požadovanej služby vyberie jej Router a v ňom sa následne otestujú všetky IRoute. Zavolá sa funkcia pridarená k prvej IRoute ktorá vyhovoala zadanej url a metóde. Ak žiadna IRoute nevyhovuje a nie je zadaná funkcia, ktorá sa má v takomto prípade zavolať, tak sa vráti sa chybová odpoveď. Rozhranie medzi proxy a smerovaním požiadaviek bude tovriť trieda ProxyRequestBus.

5.4.4 Implementácia

Úpravy sa implementovali podľa návrhu. Na zjednodušenie zápisu lamda výrazov pre registráciu funkcií služieb sa vytvorilo makro ROUTE.

5.4.5 Testovanie

Nakoľko vonkajšia funkčnosť musela ostať po zmenách rovnaká, na otestovanie sa využili existujúce intergračné testy testujúce moduly a služby. Okrem integračných testov sa služba používateľov testovala aj novým systémom jednotkových testov.

5.5 Dokončenie testovania

#OWNNET-210 Zrýchlenie testov, zodpovedná osoba: Martin Lipták #OWNNET-211 Testy na Windows, zodpovedná osoba: Martin Lipták #OWNNET-213 Ladenie testov, zodpovedná osoba: Martin Lipták #OWNNET-203 Moznost registrovat cesty a smerovat medzi nimi + refactoring modulov/sluzieb, zodpovedná osoba: Michal Dorner

5.5.1 Úloha

Zvýšenie rýchlosti a spoľahlivosti integračných testov. Odladenie integračných testov pre operačný systém Windows. Riešenie problémov pri testovaní a príprava helperov pre ostatných členov tímu.

5.5.2 Analýza a návrh

Objavili sme dve možnosti zrýchlenia testov.

- *Predmigrácia databázy* - Pred všetkými testami testovací framework vykoná migráciu databázy a zmigrovanú databázu uloží. Pred každým testom anmiesto vytvárania prázdnej databázy použiju kópi zmigrovanej.
- *Symbolické odkazy namiesto kopírovania* - Pri príprave súborového systému pre beh aplikácie použiť namiesto kopírovania symbolické odkazy.

5.5.3 Implementácia

Po realizácii možností zrýchlenia, ktoré sú uvedené v predchádzajúcej časti, sa beh testov niekoľkonásobne zrýchlil.

V operačnom systéme Windows nefungovalo presmerovanie výstupu spúšťanej aplikácie cez rúru a testovací framework nevedel, kedy sa aplikácia inizializovala a bola pripravená na testovanie. Tento problém sme vyriešili vytvorením súboru s PID po inicializácii aplikácie. Tiež sme vyriešili niekoľko problémov súvisiacich s inými koncami riadkov v súboroch ako v unixových systémoch.

Odstránili sme niektoré problémy pri testovaní javascriptov (nastavenie capybara retry time). Do testovacieho frameworku sme pridali skript, ktorý dokáže stiahnuť celú webovú stránku aj so všetkými závislosťami (javascripty, štýly, obrázky) a serializovať ju do súboru. V teste je možné stránku z tohto súboru načítať a vyskúšať, či na nej fungujú javascripty, štýly a či má správny obsah. Neskôr bude možné týmto spôsobom testovať, či JS inject nemá problémy s kompatibilitou.

Vďaka refaktoringu modulov a služieb v aplikácii bolo možné odstrániť niektoré zbytočné stuby a urobiť jednotkové testovanie jednoduchším.

Viacerí členovia tímu začali používať jednotkové a integračné testy.

5.6 Pokračovanie v implementácii GUI časti používateľských skupín

Podúlohy #OWNNET-190, zodpovedná osoba: Andrea Šteňová

Úloha

Implementujte ďalšie funkcionality používateľských skupín portálu, ktorá zahŕňa pridávanie sa do skupín a kontrolu používateľských rôl.

Implementácia

Základná funkcionalita bola implementovaná v Backbone.js v minulom šprinte. Jej funkcionalitu bolo potrebné refaktorovať, doplniť kontrolu rôl, a doplniť niektoré pokročilé funkcie.

Boli implementované nasledujúce funkcie:

- joinGroup - pridanie sa používateľa do skupiny
- leaveGroup - odídenie zo skupiny
- funkcie na kontrolu rôl

Ďalšie funkcionality sa budú implementovať v nasledujúcich šprintoch.

5.7 Dokončenie manažmentu používateľov

Úloha #OWNNET-185, podúloha #OWNNET-185 - dokončenie manažmentu používateľov, zodpovedná osoba: Marek Láni

5.7.1 Úloha

Opravte, odladte a otestujte službu pre mažment používateľov.

5.7.2 Analýza a návrh

Vzhľadom na bezpečnosť ukladaných hesiel je potrebné heslá používateľov nie len hashovať, ale rovnako pridať ich saltovanie.

5.7.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Hashovanie hesiel bolo zabezpečené pomocou hashovacej funkcie SHA1 pričom výsledné zahashované heslo uložené v databáze vzniklo po zahashovaní spojenia hesla a saltu. Salt bol vygenerovaný za pomoci aktuálneho času, ktorý bol zahashovaný funkciou SHA1. Pre správnu funkciu autentifikácie so zahashovaným heslom s pridaním saltu, bolo nutné pre tento salt vytvoriť nový stĺpec v tabuľke používateľov, kde sa salt ukladá.

5.7.4 Testovanie

Testovanie prebehlo pomocou portálu ale aj integračných testov, kedy sa registráciou a prihlasovaním používateľov odtestovalo, či funguje autentifikácia používateľov za pomoci hesla, kedy služba na strane využíva hashovanie a saltovanie hesla. Testy boli úspešné.

5.8 Dokončenie služby pre skupiny

Úloha #OWNNET-160, podúloha #OWNNET-189 - dokončenie služby pre skupiny, #OWNNET-191 - Testovanie služby, zodpovedná osoba: Marek Láni

5.8.1 Úloha

Opravte, odladte a otestujte službu pre skupiny. V prípade nutnosti pridajte nové funkcie.

5.8.2 Analýza a návrh

Na základe konzultácie s členmi tímu pracujúcimi na GUI portálu boli identifikované dve nové akcie getIsAdmin a getIsMember, ktorými sa zisťuje, či je daný používateľ administrátor respektíve člen danej skupiny.

Vzhľadom na bezpečnosť ukladaných hesiel je potrebné heslá ku skupinám nie len hashovať, ale rovnako pridať ich saltovanie.

5.8.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Hashovanie hesiel bolo zabezpečené pomocou hashovacej funkcie SHA1 pričom výsledné zahashované heslo uložené v databáze vzniklo po zahashovaní spojenia hesla a saltu. Salt bol vygenerovaný za pomoci aktuálneho času, ktorý bol zahashovaný funkciou SHA1. Pre správnu funkciu autentifikácie so zahashovaným heslom s pridaním saltu, bolo nutné pre tento salt vytvoriť nový stĺpec v tabuľke skupín, kde sa salt ukladá.

Základným predpokladom pre úspešné vykonanie akcií getIsAdmin a getIsMember je prihlásenie sa používateľa. Akcia getIsAdmin/Member je volaná požiadavkou cez url adresu my.ownet/api/groups/isAdmin/Me
ktorej telo obsahuje JSON v zložení {user_id:üser_id",group_id:"group_id"}. V prípade, že daný používateľ je adminom/členom danej skupiny vráti sa JSON odpoveď {isAdmin/Member:"1"}. V opačnom prípade {isAdmin/Member:"0"}

5.8.4 Testovanie

Na testovanie bolo použité integračné testovanie.. Otestovaná bola iba akcia create group. Testované boli všetky identifikované scenáre, ktoré môžu nastať pri vytváraní skupiny. Konkrétne sa kontrolovali odpovede po volaní akcie create v prípade neprihláseného používateľa, s posielaním zlých dát ale rovnako s posielaním dát správnych. Testovanie pomohlo odhaliť menšie problémy pri validovaní dát posielaných v požiadavke a vetvení v kóde.

5.9 Pomocné úlohy pre šprint Xela

5.9.1 Inštalácia Ruby a zoznamovanie sa s rspec frameworkom využívaným na testovanie

V tejto pomocnej úlohe si každý člen musel nainštalovať Ruby a rspec framework, podľa návodu ktorý vytvoril Martin Lipták. Po úspešnej inštalácií si každý člen vyskúšal napísať testy podľa vzorov, ktoré vytvoril Martin Lipták.

5.9.2 Inštalácia a prechod na QT 5

Keďže vyšla skoro finálna verzia QT 5, ktorá ponúka nové možnosti, ktoré nám dokážu pomôcť pri implementácií našich funkcionalít tak sme sa rozhodli, že si ju všetci nainštalujeme. Avšak pri jej inštalácií nastalo veľa komplikácií, najmä na systéme Windows, ktorých vyriešenie nám zabralo mnoho času. Tieto problémy sa nám však podarilo nakoniec vyriešiť a všetci otestovali proxy upravené pre QT 5 úspešne.

5.10 Prednačítavanie odkazov

#OWNNET-162, zodpovedná osoba: Martin Konôpka

5.10.1 Úloha

Vylepšite správu odkazov na prednačítanie, odčleňte ju do databázy a umožnite prioritizáciu odkazov. Odkazy sú momentálne uložené len v pamäti aplikácie počas jej behu. Skrytý prehliadač sa vykonáva v hlavnom vlákne aplikácie, je ho potrebné odčleniť do zvláštneho vlákna a ochrániť aplikáciu pred pádom alebo spomalením, keď sa spustí prednačítavanie.

5.10.2 Analýza

Ukladanie odkazov do databázy je priamočiare, existujúci zoznam odkazov sa nahradí volaniami databázy pre pridávanie a upravovanie odkazov.

Pre načítanie stránky v prehliadači budeme vychádzať z riešenia pôvodnej verzie projektu OwNet, odkazy na prednačítanie nazývame ako objednávky na stiahnutie.

5.10.3 Návrh a implementácia

Tabuľka databázy odkazov bude obsahovať nasledujúce stĺpce:

- ID východzej stránky,
- ID cieľovej stránky,
- adresa cieľovej stránky,
- úspešnosť stiahnutia,
- priorita,
- dátumy vytvorenia a zmeny objednávky na stiahnutie.

Keď sa objednávka na stiahnutie pridá do tabuľky, jej priorita je nastavená o 1 vyššiu hodnotu než je najväčšia priorita objednávky v tabuľke. Základná hodnota priority je 10. Takto zabezpečíme, že objednávky nebudú rovnocenné a berieme do úvahy ich čas vytvorenia.

Plánovanie vykonania prednačítavania je nastavené periodicky na každé 3 minúty. Keď sa takto plánované prednačítavanie spustí, overí sa či OwNet nestiahol za poslednú minútu viac ako 50 webových objektov:

- Ak áno, v tom prípade sa prednačítavanie nespustí a čaká sa ďalšie tri minúty.
- Ak nie, vytvorí sa nové vlákno a v ňom sa vykoná inštancia triedy *BrowserWorker*.

Vytvorený objekt *BrowserWorker* vytvorí inštanciu prehliadača *QWebView*, ktorému sa nastaví proxy server na lokálnu adresu aplikácie OwNet a jej port. Spustí sa otvorenie stránky a *BrowserWorker* čaká na minútu na načítanie stránky.

- Ak sa stránka do časového limitu načíta, označí sa v databáze ako stiahnutá. Tu sa využije vkladaný skript, ktorý oznámi na služby modulu prednačítavania na adrese http://inject.ownet/api/prefetch/ URL adresu stránky, ktorá sa načítala.

- Ak sa stránka nestihla načítať, *BrowserWorker* ukončí načítavanie stránky.

Nakoniec sa vlákno s načítavaním ukončí/zruší.

Pre vylepšenie správy odkazov na prednačítavanie sa každý 50. pokus o spustenie prednačítavania vymažú staré odkazy a v tabuľke ostane len 10 najnovších odkazov.

Sťahovanie z cache iného klienta má menšiu prioritu ako sťahovanie z vlastnej cache, preto pri rozhodovaní odkiaľ sa bude objekt sťahovať sa najprv skontroluje či nie je v lokálnej cache.

5.10.4 Testovanie

Databáza a proces prednačítavania boli testované ručne sledovaním vykonávania kódu a obsahu databázy.

6 Šprint Krismasi

6.1 Oprava zistených chýb

zodpovedná osoba: Marek Láni

6.1.1 Úloha

Úloha pozostávala z opravy chýb, ktoré boli zistené členmi tímu pri vytváraní GUI portálu.

6.1.2 Opravované chyby

- *Oprava editácie hesla pre skupinu aj používateľa* Chyba sa týkala vytvárania nového saltu, ktorým sa zvyšuje bezpečnosť utajenia hesiel. Pri editácií sa zle ukladal novo-vygenerovaný salt. Problém bol odstránený.
- *Oprava ,,vstupovania“ do skupiny* Tento problém sa týkal skupín, ktoré majú nastavené potvrdzovanie vstupu do skupiny adminom. Problém bol pri kontrole, či už používateľ poslal požiadavku, táto kontrola bola opravená.
- *Oprava vymázavania skupín* Pri vymazávaní skupín sa nevymazávali záznamy z iných tabuliek, tento nedostatok bol odstránený
- *Odporučenie jednej stránky viacerým skupinám* Pre zle vytvorenú podmienku, nebolo možné odporučiť stránku viac ako jeden krát. Do podmienky sa preto pridalo okrem kontroly unikátnosti vzhľadom na kombináciu id používateľa a url stránky, aj id skupiny.

6.2 Grafický návrh obrazoviek portálu

Zodpovedná osoba: Andrea Šteňová

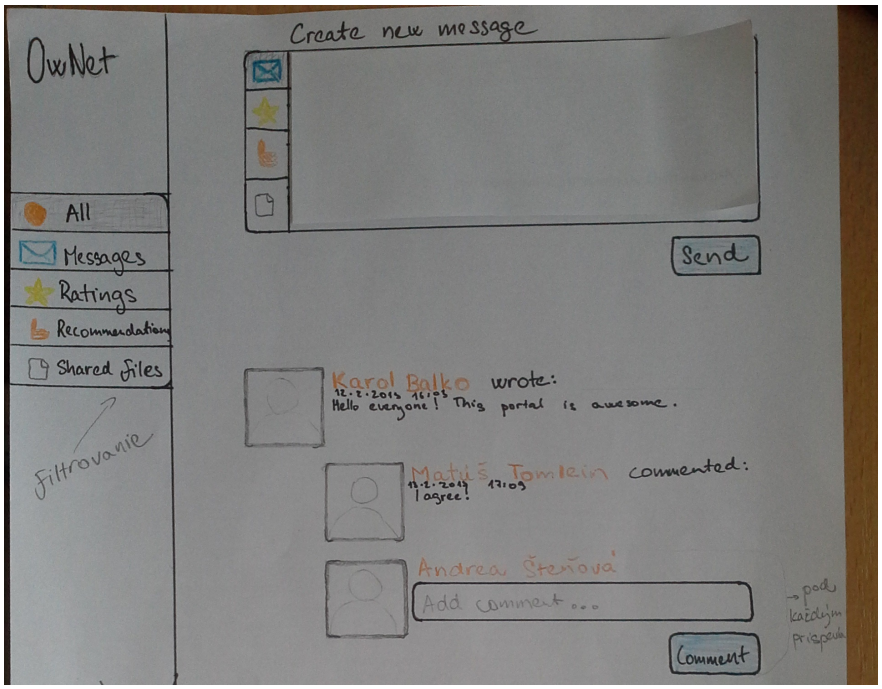
6.2.1 Úloha

Je potrebné navrhnúť rozloženie stránok portálu a ich grafické zobrazenie.

6.2.2 Návrh

Pre každú z implementovaných obrazoviek a obrazoviek, ktoré sa budú implementovať v najbližších šprintoch som vytvorila grafický návrh rozmiestnenia elementov. Návrh som vytvorila na papier a následne odfotila aby bol k dispozícii aj ostatným členom tímu.

Pri návrhu som vychádzala z aktuálnych štýlov, ktoré boli použité z Twitter Bootstrap a slúžili ako základ pre grafické rozhranie portálu. Príklad navrhnutej obrazovky je zobrazený na obrázku 8.



Obr. 8: Návrh obrazovky pre prácu so správami, hodnoteniami a odporúčaniami.

7 Šprint Happy New Year

7.1 Výber odkazov na prednačítanie

zodpovedná osoba: Martin Konôpka

7.1.1 Úloha

Vyberanie odkazov na prednačítanie na základe histórie prehliadania iných používateľov je nepostačujúce, pretože už navštívené stránky nie je nutné sťahovať znovu, už sú uloženie na lokálnej sieti. Je potrebné zvoliť jednoduchý algoritmus, ktorý bude vyberať odkazy, ale zároveň nezaťaží vykonávanie celej aplikácie analýzou a vyhodnocovaním obsahu stránky.

Ako riešenie navrhujeme zvoliť jednoduchú heuristiku, ktorá zo stránky vyberie 3 odkazy na prednačítanie.

7.1.2 Analýza

Výber odkazov na prednačítanie sa vykonáva pre každú stránku, na ktorej používateľ zotrvá viac ako 1 minútu. Preto sa nemôže vykonávať žiadne vyhodnocovanie obsahu stránky, vyberanie kľúčových slov okolo odkazov (ako to navrhujú niektoré prístupy vo vedeckých prácach).

7.1.3 Návrh a implementácia

Návrh riešenia je volanie služby modulu prednačítavania s identifikačným číslom stránky, ktoré sa získa po oznámení návštevy stránky pri zaznamenávaní histórie:

http://inject.ownet/api/prefetch/create/?page=ID

Modul prednačítavania otvorí súbor stránky pomocou knižnice tretej strany *QSgml* a použije ju na získanie všetkých odkazov stránky.

Zo zoznamu odkazov sme navrhli použiť nasledujúcu heuristiku. Z odkazov sa vyberú 3 odkazy nachádzajúce sa na pozícii prvej štvrtiny, polovice a tretej štvrtiny všetkých záznamov. Táto heuristika sa opiera o hypotézu, že stránky obsahujú odkazy v poradí: menu stránky, obsah, pätička. Takto vybrané odkazy sa pridajú do databázy spôsobom opísaným v kapitole 5.10).

Z vybraných odkazov sa ignorujú odkazy s iným protokolom než HTTP.

7.1.4 Testovanie

Proces výberu odkazov sme testovali ručne sledovaním vykonávania kódu a uistením sa, že sa vyberajú odkazy správne. Uvedomujeme si, že heuristika je primitívna a nemusí byť účinná, ale je jednoduchá na vykonanie a nezaťažuje aplikáciu.

7.2 Implementácia GUI časti hodnotenia stránok

Zodpovedná osoba: Karol Balko

7.2.1 Úloha

Implementácia časti hodnotenia portálu

7.2.2 Implementácia

Táto funkcionalita bola implmentovaná v Backbone.js. Api komunikuje na adrese ,,my.ownet/api/activities“, s parametrom type = 1.

Pre základnú funkcionalitu boli vytvorené nasledujúce súbory:

- ActivityModel - model, ktorý reprezentuje aktivitu, ktorá sa neskôr filtruje podľa typu
- RecommendationRatingsView - zobrazenie všetkých, prípadne filtrovaných aktivít
- šablóny pre zobrazenie v prehliadači

Boli implementované nasledujúce funkcie:

- show - získanie kolekcie modelov aktivít a ich zobrazenie
- showRatings - získanie a zobrazenie hodnotení
- deleteRating - zmazanie konkrétneho hodnotení

7.3 Implementácia GUI časti odporúčania stránok

Zodpovedná osoba: Karol Balko

Úloha

Implementácia časti odporúčania portálu

Implementácia

Táto funkcionalita bola implmentovaná v Backbone.js. Api komunikuje na adrese ,,my.ownet/api/activities“, s parametrom type = 0.

Pre základnú funkcionalitu boli vytvorené nasledujúce súbory:

- ActivityModel - model, ktorý reprezentuje aktivitu, ktorá sa neskôr filtruje podľa typu
- RecommendationRatingsView - zobrazenie všetkých, prípadne filtrovaných aktivít
- šablóny pre zobrazenie v prehliadači

Boli implementované nasledujúce funkcie:

- show - získanie kolekcie modelov aktivít a ich zobrazenie
- showRecommendations - získanie a zobrazenie odporúčaní
- deleteRecommendation - zmazanie konkrétneho odporúčania

7.4 Implementácia GUI časti profilu používateľa

Zodpovedná osoba: Karol Balko

Úloha

Implementácia časti profilu používateľa

Implementácia

Táto funkcionalita bola implmentovaná v Backbone.js. Api komunikuje na adrese ,,my.ownet/api/users“ a na adrese ,,my.ownet/api/activities“ s parametrom page.

Pre základnú funkcionalitu boli vytvorené nasledujúce súbory:

- ProfileView - zobrazenie užívateľského profilu, a jeho editácia
- šablóny pre zobrazenie v prehliadači

Boli implementované nasledujúce funkcie:

- show - získanie kolekcie uživateľov a ich zobrazenie
- editProfile - editovanie profilu používateľa
- showProfile - zobrazenie užívateľa na základe user_id
- saveProfile - uloženie editovaného profilu
- showActivities - zobrazenie aktivít v používateľskom profile

7.5 Zlúčenie repozitárov proxy, portálu a testov do jedného spoločného

zodpovedná osoba: Michal Dorner

7.5.1 Úloha

Zlučíť viaceré repozitáre so zdrojovými kódmi do spoločného. Vyriešit konflitky, umiestnenie súborov, upravenie .gitignore súborov. Otestovať či po zlúčení všetko funguje ako predtým.

7.5.2 Analýza a návrh

Momentálne používame tri oddelené Git repozitáre so zdrojovými kódmi. Zvlášť pre proxy, portál a testy. Zistili sme, že toto rozdelenie namiesto lepšej organizácie prináša len zbytočné problémy. Portál a proxy sú natoľko prepojené, že nemá zmysel aby boli v oddelených repozitároch. Tak isto pri súčasnom stave sa nedá pozrieť stav celého systému v minulosti, na koľko nevieme tak ľahko určiť aká verzia proxy bola pri akej verzii portálu.

Jednotlivé repozitáre obsahujú viacero vetví s rozrobenou funkcionalitou. Pred spojením repozitárov bude potrebné v každom repozitári zlučíť vetvy do master vetvy.

Portál bude vhodné začleniť do resources/static priečinka a testy do Tests priečinka. Portál obsahuje Makefile súbory, ktoré sú však v prípade proxy generované automaticky a teda sú v .gitignore aby neboli pod správou verzií. V priečinku s portálom bude potrebné tieto nastavenia prepísať aby sa tieto súbory zachovávali.

7.5.3 Implementácia

Portál bol začlenený do priečinka resources/static a testy do Tests. Nastavenia správy verzií cez .gitignore súbory boli upravené aby sa zachovávali Makefile súbory z portálu.

7.5.4 Testovanie

Funčnosť systému bola po integrácii portálov otestovaná integračnými testami.

7.6 Nový jednoduchší multiplatformový build systém pre portál

zodpovedná osoba: Michal Dorner

7.6.1 Úloha

Nakoľko neustále pretrvávajú problémy s build sytémom portálu a najmä členovia týmu s windows majú problém si bez asistencie nainštalovať potrebné závislosti, bude potrebné tento systém prerobiť. Týmto by sa mal uľahčiť aj prípadný ďalší budúci vývoj iným programátorom.

7.6.2 Analýza a návrh

Súčasná verzia používa na build portálu Make. V Makefile sa používajú aj linuxové utility ako find, sed, cp a rp. Na windows bolo potrebné distribuovať porty týchto utilít a zvlášť ošetrovať niektoré prípady, kôli čomu bol Makefile zbytočne zložitý. Build portálu bol ďalej závisli od ruby a nainštalovaného gemu SASS a ďalej od NODE.js a viacerých NPM balíčkov.

Ukázalo sa že práve NODE.js a jeho NPM balíčky sú dobre multiplatformé, cez Pakcage.json sa dajú pekne definovať zavislosti a tak isto existuje LESS CSS preprocesor, ktorý môže nahradiť SASS a tak bý sme sa zbavili aj závislosti na ruby.

Najviac používaný build systém pod NODE.js sa volá grunt. Je dobre konfigurovateľný a vieme v ňom implementovať logiku z pôvodného Makefile.

7.6.3 Implementácia

Z portálu sa odstránil priecinok .bin, v ktorom sa nachádzali pevne dané NPM balíčky a nahradili sa súborom Package.json v ktorom sú tieto závislosti deklaratívne zapísane. Príkazom "npm install ."sa vykoná inštalácia týchto balíčkov.

Všetky SASS súbory sa následne prepísali do LESS formátu a LESS sa pridal medzi definované závislosti. Build systém Grunt sa pridal medzi závislosti a v Gruntfile sa definoval process buildovania.

7.6.4 Testovanie

Funkčnosť build systému sa otestovala na všetkých platformách. Všetkým členom týmu fungovala správne a nemali probém s inštaláciou.

7.7 Filtrovanie zoznamu skupín v Qt

zodpovedná osoba: Marek Láni

7.7.1 Úloha

Modifikujte službu pre skupiny tak, aby bolo možné filtrovať skupiny na základe členstva v skupine.

7.7.2 Analýza a návrh

Je potrebné zobraziť zoznam skupín, ktorých je daný používateľ členom, ktorých je adminom, ktorých nie je členom a v rámci ktorých čaká na schvalenie. Pre takéto filtrovanie bolo potrebné navrhnúť spôsob akým filtrovanie bude fungovať, čo bolo vyriešené samostatnou akciou pre každý filter.

7.7.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Pre filtrovanie bolo potrebné vytvárať selekty s využitím SQL funkcie JOIN tabuliek groups a group_users resp. group_admins, vďaka čomu sme schopný správnymi podmienkami získať požadované skupiny.

Čo sa týka filtrovania boli vytvorené akcie, ktoré možno volať požiadavkami GET na tieto URL
http://my.ownet/api/messages/+
getUsersGroups
getMyAdminGroups
getNotMyGroups
getAwaitingGroups

Pre volanie týchto akcií musí byť používateľ prihlásený.

7.7.4 Testovanie

Testovanie prebehlo prostredníctvom využitia nástroja Poster, kedy boli vytvorené skupiny tak, aby každý filter pre daného používateľa obsahoval aspoň jednu skupinu a následne boli kontrolované odpovede požiadavok na jednotlivé URL.

7.8 Dokončenie skupín portálu

zodpovedná osoba: Andrea Šteňová

7.8.1 Úloha

Dokončite funkcionalitu skupín, implementujte filtrovanie zoznamu skupín a logiku pridávania do skupín.

7.8.2 Implementácia

V zozname skupín bolo pridané filtrovanie na :

- všetky skupiny
- skupiny v ktorých som administrátor
- skupiny v ktorých som členom
- skupiny v ktorých čakám na akceptáciu prijatia

Toto filtrovanie bolo implementované pomocou QT služieb, kde sa pri každom filtry volala iná zo služieb.

Ďalej som implementovala logiku pridávania používateľov do skupín. Pri vytváraní skupiny totiž môže používateľ zaškrtnúť možnosť, že chce aby schvaľoval používateľov, ktorí sa pridajú do skupiny. Potom iný používateľ sa nemôže priamo pridať do skupiny, ale môže iba požiadať o pridanie.

V zozname skupín potom vidí používateľ tieto možnosti:

- pridanie sa do skupiny
- požiadanie o prijatie do skupiny
- opustenie skupiny
- editácia skupiny, v ktorej je administrátor
- vymazanie skupiny, v ktorej je administrátor

Tiež informujeme používateľa ak už požiadal o prijatie do skupiny.

Administrátor skupiny spravuje používateľov v záložke List members v konkrétnej skupine. Môže:

- schváliť požiadavku na pridanie do skupiny
- odmietnuť požiadavku na pridanie do skupiny
- odstrániť člena skupiny
- spraviť z člena skupiny administrátora

Každý z členov skupiny má možnosť ju opustiť, okrem administrátora, ktorý môže skupinu len vymazať.

Takisto má každý z administrátorov možnosť editácie skupiny.

7.8.3 Testovanie

Testovanie prebiehalo priamo v portály kde som pomocou viacerých používateľov odskúšala rôzne scenáre používania skupín.

7.9 Správy medzi používateľmi v skupine

zodpovedná osoba: Andrea Šteňová

7.9.1 Úloha

Vytvorte stránku na posielanie správ medzi používateľmi, s možnosťou mazania a komentovania príspevkov. Správy bude možné pridávať pre všetkých používateľov, alebo pre používateľov v skupine.

7.9.2 Implementácia

Správy sa vždy posielajú na skupinu. V zozname správ na úvodnej obrazovke sa správy posielajú na predvolenú skupinu General, do ktorej je každý z nich pri registrácii zaradený a z ktorej nemôže odísť. Preto potom odosielanie správ v skupine a na hlavnej obrazovke vyzerá a správa sa pre používateľa rovnako.

Používateľ môže:

- vytvoriť správu
- zmazať svoju správu
- komentovať ľubovoľnú správu

Správy sa zobrazujú chronologicky od najnovšej po najstaršiu, čo zabezpečuje služba na zobrazovanie správ. Komentáre sú priradené k správe a pri vymazaní správy sa vymažú aj všetky jej komentáre.

7.9.3 Testovanie

Testovanie prebiehalo priamo v portály, kde som pomocou viacerých používateľov testovala posielanie, komentovanie a mazanie správ.

7.10 Vytvorenie aktivít v Qt

zodpovedná osoba: Marek Láni

7.10.1 Úloha

Navrhnite a implementujte službu pre vytváranie aktivít. Za aktivitu bude pokladané odporúčania a hodnotenie stránok.

7.10.2 Analýza a návrh

Pre umožnenie zobrazovania tzv. „News Feed“ je potrebné zaviesť nový objekt a služby spojené s ich vytváraním, editovaním, zobrazovaním a vymazávaním. Po analýze sme dospeli k faktu, že hoci všetky údaje, ktoré objekt aktivita obsahuje, sa dajú v databáze vyhľadať, ich zobrazovanie je však oveľa jednoduchšie pri vytvorení samostatného objektu.

Služba a objekt aktivita je opísaný v nasledujúcej časti.

7.10.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Objekt aktivita, ktorý sa ukladá do databázy pozostáva z nasledujúcich polí

- String user_name;
- String content;
- String date_created;
- int activity_type;
- String group_id;
- String user_id;
- String object_id;
- String gender;

Rozhranie a funkcie služby pre aktivity sú nasledovné:

- *create:*

  Akcia sa zavolá priamo pri vytváraní objektov odporúčanie a honotenia. To znamená, že vytvorenie aktivity podlieha podmienkam a validácií pred vytvorením odporúčania a hodnotenia.

- *index:*

  Táto akcia je volaná bez parametrov v prípade, že odpoveďou má byť zoradený zoznam všetkých aktivít na základe času ich vytvorenia. Akcia sa zavolá GET HTTP požiadavkov na url adresu my.ownet/api/activities.

  V prípade, že je potrebné filtrovať zoznam aktivít podľa ich typu pošle sa tento typ ako parameter požiadavky(?type=xy).

Ako odpoveď je vrátené JSON pole s aktivitami, ktoré patria do skupín ktorých je prihlásený používateľ členom a HTTP status "200 OK". V prípade chybnej požiadavky je návratová hodnota "400 Bad Request".

- *del:*

  Akcia sa zavolá priamo pri vymazávaní objektov odporúčanie a honotenia.

- *edit:* Pri editácii hodnotenia a odporúčania je volaná editácia príslušnej aktivity.

7.10.4 Testovanie

Testovanie prebehlo prostredníctvom využitia nástroja Poster, cez ktorý sa simulovali požiadavky, vytvárali a mazali hodnotenia a odporúčania a overovali vytváranie a mazanie aktívit v databáze. Rovnako sme pomocou nástoja Poster testovali index akciu aktivít.

8 Šprint Hurry up!

8.1 Inštalácia aplikácie

Zodpovedná osoba: Martin Lipták

8.1.1 Úloha

Umožnenie jednoduchej a rýchlej inštalácie aplikácie OwNet. Primárne sa zamerať na OS Windows, keďže ten je cieľová platforma pri prvých nasadeniach.

8.1.2 Analýza

Hľadal som nástroje pre vytvorenie inštalačného balíčku pre platformu Windows. Matúš Tomlein mi odporučil InnoSetup, s ktorým má dobré skúsenosti a ponúka všetky štandardné možnosti pre tvorbu inštalátora.

8.1.3 Vytvorenie inštalátora

Vytvoril som virtuálny stroj so systémom Windows XP, na ktorý som nainštaloval všetky potrebné nástroje na vývoj aplikácie OwNet. Na virtuálnom stroji som skompiloval najnovšiu verziu master vetvy aplikácie. Pripravil som inštalačný konfiguračný súbor pre InnoSetup so zoznamom súborov aplikácie, súborov s knižnicami a ďalších vecí ako ikona, spustenie po štarte a možnosť odinštalácie. Nástroj InnoSetup pomocou konfiguračného súboru vytvoril zo skompilovanej aplikácie inštalátor.

8.1.4 Testovanie inštalátora

Na testovanie som vytvoril ďalší virtuálny stroj so systémom Windows XP. Na tento systém som neinštaloval žiadne vývojárske nástroje ani knižnice, ktoré potrebuje aplikácia OwNet. Cieľom bolo otestovať inštaláciu na čistom systéme (na vývojárskom systéme by som si nemusel všimnúť knižnicu chýbajúcu v inštalátore, lebo by sa tento problém kvôli dostupnosti tejto knižnice v systéme neprejavil). Po inštalácii bola aplikácia dostupná v systéme, fungovala správne vrátane monitorovacej služby a po ďalšom štarte systému sa znova spustila.

8.2 Vrátenie funkcií prednačitavania a histórie do vkladaného skriptu

Zodpovedná osoba: Martin Konôpka

8.2.1 Úloha

Po reimplementácii vkladaného skriptu do stránok je potrebné znovu pridať funkcionalitu oznamovania návštevy stránok a ovládania prednačítavania.

8.2.2 Analýza

Reimplementácia vkladaného skriptu do stránok priniesla zmenu logiky vykonávania celého skriptu. Je však možné znovupoužiť existujúci kód, ktorý sa týka oznamovania návštev na stránky a ovládania prednačítavania.

8.2.3 Návrh a implementácia

Do kódu bol späť pridaný objekt HistoryContact, ktorý sa stará o oznamovanie histórie prehliadania (otvorenie stránky) a objekt PrefetchContact, ktorý sa stará oznamovanie o prednačítavaní. Volania metód boli vložené na správne miesta vo vkladanom skripte a upravené podľa zmien v celom skripte. Pomocné funkcie na pridávanie ďalších skriptov do stránky (pre vynútenie volania služieb modulov), pridávanie CSS štýlov, uloženie ID stránky boli do skriptu taktiež pridané.

8.2.4 Testovanie

Nový skript súbor bol testovaný ručne, sledovaním vykonávania kódu a volaním služieb modulov prednačítavania a histórie.

8.3 Zvýrazňovanie odkazov stránok dostupných offline

Zodpovedná osoba: Martin Konôpka

8.3.1 Úloha

Pridajte funkcionalitu zvýrazňovania odkazov na stránky, ktoré sú dostupné offline, t.j. sú uložené v cache aplikácie. Odkazy sa zvýrazňujú cez tlačítko vo vkladanom pruhu na stránke. Odkaz je zvýraznený oranžovým okrajom.

8.3.2 Analýza

Implementácia bude vychádzať z implementácie v pôvodnej verzii projektu. Vo vkladanom pruhu bude tlačítko, ktoré po kliknutí vyžiada príslušný modul o zoznam odkazov, ktoré sú offline. Tento zoznam sa získa odfiltrovaním offline odkazov zo zoznamu všetkých odkazov na stránke. Všetky odkazy sa získajú analýzou súboru stránky.

8.3.3 Návrh a implementácia

Do kódu vkladaného skriptu bol pridaný objekt HighlightSwitch, ktorý odchytáva príkazy používateľa na zvýraznenie a skrytie odkazov dostupných offline. Po kliknutí na tlačítko zvýraznenia sa volá služba modulu prednačítavania (pretože ten je s touto funkcionalitou najviac spätý).

http://inject.ownet/api/prefetch/list/?page=ID

Služba otvorí stránku pomocou knižnice *QSgml* a vyberie všetky odkazy stránky. Z nich vyskladá dopyt do databázy pre overenie uloženie týchto stránok v cache (tabuľka *client_caches*).

Z výsledného zoznamu odkazov, ktoré sú dostupné offline sa vytvorí JavaScript kód v tvare:

owNetHIGHLIGHTLINKS = [http://A, http://B,...]

Tento kód sa pošle ako odpoveď a skript ho načíta. Tým dostaneme zoznam do premennej v skripte na stránke a objekt na zvýrazňovanie môže odkazy zvýrazniť. Zvýraznenie prebieha pridaním novej triedy do odkazov na stránke, ktorá pridá oranžový okraj okolo odkazu. Pri vypínaní zvýraznenia sa iba odstráni nami definovaná trieda z označených odkazov.

Odkazy, ktoré sa majú zvýrazniť sa získavajú pri každej požiadavke používateľa o zvýraznenie odkazov. A to z dôvodu, že počas jeho návštevy na stránke sa mohli pridať nové stránky do cache (napr. otváranie stránok na kartách, alebo aj zmena v cache na lokálnej sieti).

8.3.4 Testovanie

Zvýrazňovanie bolo testované ručne, sledovaním vykonávania kódu a volaním služby modulu prednačítavania.

8.4 Stránkovanie zoznamov objektov

zodpovedná osoba: Marek Láni

8.4.1 Úloha

Navrhnite a implementujte stránkovanie zoznamov objektov, ktoré sú vrátené po zavolaní akcie index v rámci jednotlivých služieb. Konkrétne pre skupiny, odporúčania, správy, hodnotenia a aktivity.

8.4.2 Analýza a návrh

Stránkovanie obsahu by malo fungovať na známom prístupe, kedy je potrebné pre vytvorenie stránkovania v rámci GUI získavať počet stránok do ktorých sa podľa určitého limitu zorazených objektov tento zoznam rozdelí. Následne je možné posielať požiadavky na získanie danej stránky. Práve táto funkcionalita by mala byť službami sprístupnená.

8.4.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Pre služby spomenuté vyššie sme pridali akciu, ktorej návratová hodnota je počet stránok do ktorých sa rodelí zoznam daných objektov. Počet objektov zobrazený na stránku je momentálne fixne obmedzený na 10. V rámci volania index akcie je potom potrebné pridať parameter page, ktorý označuje stránku zoznamu, ktorá sa ma zobraziť.

8.4.4 Testovanie

Testovanie prebehlo prostredníctvom využitia nástroja Poster, cez ktorý sa po vytvorení objektov, ktorých sa týka stránkovanie, testovalo zisťovanie počtu stránok a odosielanie dát objektov, ktoré sa na danej stránke zobrazujú.

8.5 Implementovanie REST centrálnych služieb a komunikácie s nimi

Zodpovedná osoba: Matúš Tomlein

8.5.1 Úloha

Implementovanie a nasadenie webových služieb s ktorými môžu komunikovať OwNet aplikácie na rôzne účely.

8.5.2 Analýza

V predchádzajúcej .NET verzií boli využívané služby vytvorené v .NET WCF. V novej verzií aplikácie sú však všetky lokálne služby implementované ako REST služby vymieňajúce si správy v JSON. Takáto implementácia má výhodu v menšej réžii a veľkosti správ, čo je dôležitý faktor pri komunikácii cez internet prostredníctvom pomalého pripojenia.

8.6 Návrh a implementácia

Kvôli znovupoužitiu niektorých komponentov na centrálnych službách z minulého riešenia boli nové služby tiež implementované v .NET. Nebola však použitá technológia WCF, služby ponúkajú REST rozhranie.

V lokálnej aplikácii bol implementovaný modul cez ktorý je možné jednoducho volať akcie na centrálnych službách ako keby boli volané lokálne služby. Tento modul sprístupňuje službu *central_service*, ktorá po zavolaní dopytu `http://my.ownet/api/central_service/path/to/action?arg=1` zavolá akciu s relatívnou cestou *path/to/action?arg=1* na centrálnych službách. Ako odpoveď vráti odpoveď centrálnych služieb. Akceptuje GET, PUT, POST a DELETE volania.

Centrálne služby boli nasadené a sú prístupné na adrese: `http://yatta.fiit.stuba.sk/OwNetRestService/`

8.7 Synchronizácia s centrálnymi službami

Zodpovedná osoba: Matúš Tomlein

8.7.1 Úloha

Implementovanie synchronizácie databázy s centrálnymi webovými službami. Toto umožní vzájomnú synchronizáciu OwNet klientov aj keď sú niektoré z nich mimo školskej lokálnej siete.

Implementácia by mala umožňovať vkladanie nových záznamov do synchronizácie na strane webových služieb.

8.7.2 Analýza

Na synchronizáciu s centrálnymi službami je najlepšie použiť rovnaký protokol a formát správ ako sa používa na vzájomnú synchronizáciu klientov na lokálnej sieti.

Je potrebné zabezpečiť aby centrálne služby vedeli rozlíšiť klientov z rôznych škôl.

Na jednej lokálnej sieti by sa mal s centrálnymi službami synchronizovať len jeden počítač. Ostatné počítače si nové záznamy z centrálnych služieb automaticky zosynchronizujú z daného počítača.

Aby sa nesťahovalo veľa dát cez internet, z centrálnych služieb by sa nemali sťahovať záznamy o nových cache objektoch na klientoch. Tie ani nemá význam synchronizovať ak nie sú dané počítače na lokálnej sieti, keďže by sa ani nemohli využiť.

8.7.3 Pracovné skupiny

Aby bolo možné rozlíšiť školy, v klientskej aplikácii je implementovaný systém pracovných skupín. Každá pracovná skupina je identifikovaná unikátnym GUID.

Po prvom spustení, vygeneruje každá aplikácia nové GUID pracovnej skupiny a teda má samostatnú pracovnú skupinu. Ak je viac aplikácii s rôznymi pracovnými skupinami na lokálnej sieti (nájdu sa pomocou multicast komunikácie), zobrazí sa v hlavnom okne aplikácie ich zoznam s možnosťou pripojiť sa na ne (obr. **??**).

Pre každú pracovnú skupinu do ktorej sa daný klient pripojil je v priečinku používateľa vytvorená samostatná SQLite databáza. SQLite databáza pracovnej skupiny sa vytvorí po pripojení na pracovnú skupinu a v hlavnom okne aplikácie je možné prepínať medzi vytvorenými databázami.

8.7.4 Implementácia synchronizácie

S centrálnymi službami sa periodicky synchronizuje iba aktuálny server na lokálnej sieti. Synchronizácia prebieha rovnako ako medzi klientami na lokálnej sieti (podrobnejšie bola popísaná v kapitole 5.1).

Posielanie nových záznamov na centrálne služby funguje nasledovne:

1. Klient si vypýta z centrálnych služieb aktuálny stav záznamov na nich
2. Podľa získaného stavu vytvorí zoznam nových zmien, ktoré pošle na centrálne služby

Sťahovanie nových záznamov z centrálnych služieb funguje nasledovne:

1. Na centrálne služby sa pošle aktuálny stav synchronizačného žurnálu
2. Centrálne služby ako odpoveď vrátia nové záznamy, ktoré na klientovi nie sú

Obr. 9: Hlavné okno aplikácie so zoznamom dostupných pracovných skupín.

Na centrálnych službách je možné vkladať nové záznamy do synchronizačného žurnálu. Takéto riešenie je výhodné hlavne v tom, že vložené záznamy sa na lokálnych sieťach automaticky prešíria na všetky počítače bez toho aby si ich každý musel sťahovať.

Pri volaní akcií na centrálnych službách sa vždy posiela GUID aktuálnej pracovnej skupiny.

8.7.5 Testovanie

Na strane centrálnych služieb, boli opísané funkcie testované pomocou jednotkových testov v .NET.

Na strane klientskej aplikácie, boli na testovanie synchronizácie tiež použité jednotkové testy v Qt.

8.8 Aktualizácia cache

Úloha #OWNNET-230, Zodpovedná osoba: Matúš Tomlein

8.8.1 Úloha

Implementovanie aktualizácie webových stránok v cache aplikácie. Aktualizácia by mala priebehať len v prípade, že sa stránka zmenila.

8.8.2 Analýza

Aby nemuseli klientské aplikácie neustále kontrolovať zmeny na webových stránkach, kontrola by mala prebiehať na centrálnych službách.

Je potrebné nezaťažovať internetové pripojenia na aktualizáciu nenavštevovaných web stránok.

Na sledovanie zmien na stránkach je možné využiť službu WebTracker vývijanú vrámci diplomového projektu na FIIT.

8.8.3 Návrh a implementácia

Sledovanie zmien na stránkach vykonáva systém WebTracker, ktorému OwNet centrálne služby posielajú zoznam navštevovaných stránok a pýtajú si od neho zoznam zmenených stránok. Informáciu o zmenenej stránke sa vloží medzi záznamy, ktoré si synchronizáciou s centrálnymi službami stiahne aktuálny server v škole a na lokálnej sieti sa potom synchronizáciou prešíri aj na iné počítače.

Synchronizáciou sa do tabuľky *client_caches* (obr. 7) vloží záznam s ID zmeneného objektu a ID klienta nastaveným na ,,WEB“. Táto tabuľka slúži na rýchle zistenie na akom klientovi sa nachádza cache objekt s daným ID. Keď príde na proxy dopyt na danú cache, skontrolujú sa záznamy v tabuľke *client_caches*. Ak najnovší z nich má ako ID klienta nastavené ,,WEB“ a počítač je pripojený na internet, stiahne sa objekt z internetu a premaže sa cache. Tým sa vytvorí nový záznam v tabuľke *client_caches*, ktorý sa zosynchronizuje na sieti a po ďalšom dopyte na danú cache sa bude preferovať cache na tomto klientovi, keďže má najnovší dátum vytvorenia.

8.8.4 Testovanie

Na strane centrálnych služieb bola funkcionalita otestovaná jednotkovými testami v .NET.

Klientská aplikácia nebola automaticky testovaná, testovali sme ju len manuálne.

8.9 Vylepšenie jadra proxy na spracovanie HTTP dopytov

Zodpovedná osoba: Matúš Tomlein

8.9.1 Úloha

Aktuálna implementácia proxy spôsobuje časté pády a pomalé načítavanie stránok. Je potrebná analýza spôsobov ktorými by sa dalo tomuto správaniu zamedziť a menšie a aj väčšie zásahy v implementácii spracovania webových dopytov v proxy.

Systém je potrebné otestovať na rôznych operačných systémoch.

8.9.2 Analýza

Aktuálna verzia používa QtWebApp HTTP server na odchytenie dopytov z prehliadača, vytvorenie vlákien a na prácu so socketom. Tento HTTP server však nie je optimalizovaný na použitie ako proxy server, je zameraný skôr na poskytnutie funkcionality pre webové aplikácie. Navyše vykonáva veľké množstvo zbytočného spracovania dopytov.

Je potrebná implementácia nového HTTP servera, ktorý bude optimalizovaný na použitie ako proxy server. Nemal by vykonávať zbytočné spracovanie dopytov a odpovedí. Mal by vedieť znovupoužívať otvorené sockety, keďže to vyžadujú moderné prehliadače.

Na sťahovanie webových objektov sa používa QNetworkAccessManager. Ten však prináša rôzne obmedzenia ako maximálny počet súčasne vykonávaných dopytov a tiež vykonáva zbytočné spracovanie odpovedí.

8.9.3 Návrh a implementácia

Ako náhrada HTTP servera QtWebApp, bol implementovaný vlastný TCP server, ktorý využíva triedy QSocket a QTcpServer z knižnice Qt na spracovanie dopytov. Tento server umožňuje znovupoužitie socketov z prehliadača na viaceré dopyty.

Na vykonanie webových dopytov sa namiesto QNetworkAccessManager používa čistý QTcpSocket, ktorý umožňuje väčšiu kontrolu a menej réžie. Do tohto socketu sa zapisujú dáta v podobe v akej boli prijaté socketom z prehliadača. Na rozdiel od predchádzajúcej implementácie sa prijatá odpoveď zo socketu nespracováva, zapíše sa v prijatej podobe aj s HTTP hlavičkami do súboru v cache. V predchádzajúcej verzii boli hlavičky ukladané zvlášť v databáze čo spôsobovalo spracovanie a viac práce s databázou.

Jediné spracovanie, ktoré sa vykonáva je na zistenie dĺžky odpovede. Tá sa zisťuje buď z hlavičky Content-length, alebo v prípade odpovede s hlavičkou ,,Transfer-encoding: chunked“ vyžaduje sledovanie dĺžok jednotlivých prijatých blokov.

Keďže už nie je potrebné vkladanie vlastného obsahu do HTML stránok (injectovanie JavaScript súborov vykonáva rozšírenie v prehliadači), bolo umožnené prijať odpoveď vo podobe gzip, nie len ako čistý text.

8.9.4 Testovanie

Najväčšiu časť implementácie tvorilo manuálne testovanie proxy navštevovaním web stránok a aj pomocou externých skriptov, ktoré dopytovali proxy. Aplikácia bola testovaná pod operačnými systémami Windows, Linux a Mac OS X. Zlepšenie oproti predchádzajúcej verzii bolo viditeľné stabilitou aplikácie a rýchlejším spracovaním dopytov.

Testovanie prostredníctvom jednotkových a integračných testoch nebolo vykonané kvôli náročnosti vytvorenia prostredia, ktoré by zodpovedalo reálnemu.

8.10 Načítanie stránky z internetu namiesto cache podľa hlavičiek v požiadavke

zodpovedná osoba: Michal Dorner

8.10.1 Úloha

V súčasnej verzii sa na znovunačítanie stránky z internetu namiesto cache používa tlačítko v inject bare priamo v stránke. Pre používateľa je to ale nepraktické a neintuitivne, nakoľko na znovunačítanie stránky už obsahuje tlačítko samotný prehliadač. Prehliadač má toto ltačítko umiestnené lepšie pre vyvolanie znovunačítania stránky je možné použiť aj klávesovú skratku.

Je potrebné analyzovať možnosti rozoznania stlačenia tlačítka na znovunačítanie stránky podľa hlavičiek v požiadavke. Ak to bude možné, následne implementovať toto správanie do proxy a zrušiť pôvodné tlačítko.

8.10.2 Analýza a návrh

Prehliadač chrome po stlačení tlačítka na obnovenie stránky posiela nasledovné hlavičky:

Pragma: No-cache

Cache-Control: no-cache

Cache-Control: max-age=0

Prehliadač Firefox posiela dané hlavičky až po stlačení klávesovej skratky na úplné obnovenie stránky. To sa vyvoláva klávesovou skratkou ctrl + shift + R alebo ctrl + shift + F5. Z tohto dôvody by rozoznávanie obyčajného znovunačítania stránky pre prehliadač firefox nebolo možné a muselo by ostať pôvodné tlačítko.

Vzhľadom na plánované zaradenie prehliadača chrome/chromium medzi požiadavky OwNetu to však nebude predstavovať problém a rozhodli sme sa implementovať tento nový spôsob obnovy stránky.

8.10.3 Implementácia

Z JavaScriptového kódu, ktorý sa vkladá používateľom do stránok bola odstránená časť kódu ktorá vytvárala dané pôvodné tlačítko. Z Qt kódu proxy serveru sa do časti v ktorej sa rozhoduje ako obslužiť prichádzajúci požiadavku doplnil test na hlavičky požiadavky. Ak sú splnené podmienky, tak namiesto načítania stránky z cache sa vykoná nové načítanie z internetu.

8.10.4 Testovanie

Vzhľadom na povahu problému by automatické testy v tomto prípade boli zbytočne komplikované a funkčnosť sa otestovala manuálne stlačením tlačítka v prehliadači a kontrolovaním ladiacich výpisov.

8.11 Nahradenie injectu v Proxy cez Extension do prehliadača Chrome

zodpovedná osoba: Michal Dorner

8.11.1 Úloha

Odstrániť vkladanie injectu v Proxy. Vytvoriť rozšírenie pre prehliadač Chrome/Chromium ktorý ho bude vkladať po načítaní stránky. Vytovriť skript na spúšťanie Chrome s nastavenou proxy a predinštalovaným rozšírením.

8.11.2 Analýza a návrh

Vkladanie JavaScript kódu do stránok priamo v kóde nie je vhodné, nakoľko pridáva navyše ďalšie spracovanie streamu dát a hlavne sa kôli tomu zakazovli komprimované stránky. Práve komprimácia stránok výrazne znižuje zaťaženie siete, čo je pre nás kľúčové. Vloženie ¡script¿ tagu do stránky je pritom veľmi jednoducho realizovatelné cez rozšírenie do prehliadača.

Chrome sa dá spúšťať s nastavenou proxy cez argumenty na príkazovom riadku. Pre proxy sa použíje:

–proxy-server=”http=127.0.0.1:8081”

Ak je však už prehliadač otvorený tak sa tieto argumenty ignorujú a vytvorá sa len nové okno v existujúcom session prehliadača. Riešením tohto problému je zadať aj argument:

–user-data-dir=”...”

Keď je zadaný iný domáci adresár, tak sa vytvorí nová session. Zároveň je možné mať v tomto adresáry predinštalované rozšírenie na vklaanie injectu.

8.11.3 Implementácia

Z proxy sa odstránil kód na vkladanie injectu. Do priecinku resouces/chrome sa priadalo vytvorené rozšírenie pre chrome a štartovací skript.

8.11.4 Testovanie

Funkčnosť systému sa overila skontrolovaním, či sa inject vkladá do stránok. Nič iné sa v systéme nemenilo takže ďalšie testy neboli potrebné.

8.12 Hodnotenie a odporúčanie stránok - IFrame

zodpovedná osoba: Michal Dorner

8.12.1 Úloha

Implementovať rozhranie pre ohodnotenie a odporúčenie stránky v IFrame.

8.12.2 Analýza a návrh

V iframe bude bežať webová aplikácia postavená na Backbone.js a Require.js rovnako ako samotný portál. S injectovanym JavaScriptom, ktorý sa nachádza v kontexte samotnej stránky bude komunikovať prostredníctom metódy postMessage(). Na oboch stranách bude pritom implementované events API, cez ktoré sa komunikácia zjednoduchšií.

Inject bude cez udalosti posielať údaje o aktuálnej stránke - url a title. Z iframe kvôli bezpečnostným obmedzeniam by nebolo možné pristupovať k objektom document ani location zo stránky.

8.12.3 Implementácia

Do injectu bol pridaný kód na vloženie iframe tagu do stránky a implementované events API. Iframe si po inicializácii načíta dáta o aktuálne prihlásenom používateľovi. Z iframe je možné sa prihlásiť/odhlásiť, ohodnotiť a odporúčiť stránku. Iframe je štandardne schovaný a zobrazuje sa animáciu po kliknutí na tlačítko v inject bare.

8.13 Stránkovanie portálu

Zodpovedná osoba: Andrea Šteňová

8.13.1 Úloha

Pomocou vytvorených QT služieb upravte portál tak, aby všetky zoznamy v ňom bolo možné stránkovať (zoznam skupín, správ, hodnotení, odporúčaní...).

8.13.2 Návrh

Pri stránkovaní aktivít v portály je potrebné zistiť počet stránok jednotlivých zoznamov a potom volať upravené QT služby so získanou stránkou.

8.13.3 Implementácia

Pri každom zozname bolo potrebné upraviť jeho načítavanie z QT služieb. Vždy sa najprv zistí počet stránok konkrétneho zoznamu, následne sa zavolá služba na zoznam aktivít s prvou stránkou. Používateľovi sa okrem zoznamu aktivít z prvej stánky zobrazí aj možnosť prejdenia na ďalšie z nich. Na získanie zoznamu aktivít sa teda QT službe vždy posiela parameter page, označujúci ktorú stránku má zobraziť.

Počet aktivít na jednu stránku je momentálne nastavený v QT službe na 10 a služby sa nedajú volať s iným stránkovaním. Do budúcna je to možné implementovať tak, že počet aktivít na stránku sa bude získavať z volania služby.

8.14 Štýlovanie správ portálu

Zodpovedná osoba: Andrea Šteňová

8.14.1 Úloha

Aby bol portál použiteľný a nasaditeľný, museli sme upraviť jeho grafické rozhranie. V tejto úlohe som podľa vytvoreného grafického návrhu štýlovala stránku so správami používateľov.

8.14.2 Implementácia

Pri naštýlovaní portálu som vychádzala z návrhu obrazoviek, ktorý bol vytvorený v predchádzajúcich šprintoch. V tejto úlohe som naštýlovala stránku so správami používateľov. Štýly boli vytvorené v jazyku LESS[1], ktorý rozširuje funkcionalitu jazyka CSS. Pred spustením aplikácie sa tieto štýly skompilujú do CSS.

Aby boli aktivity ako mazanie a editácie intuitívnejšie pre používateľov, použila som namiesto slovného opisu obrázkové ikony, ktoré zobrazia opis aktivity pri prejdení myšou ponad obrázok. Obrázky som čerpala zo stránky Icon Archive[2], kde sú na nekomerčné účely dostupné zadarmo.

Aby bolo možné naštýlovať správy podľa návrhu bolo potrebné doplniť a upraviť zobrazujúce informácie. Každá správa potom obsahuje informácie o používateľovi, ktorý ju vytvoril, spolu s možnosťou prejdenia na jeho profil, ako aj informácie o dátume vytvorenia správy, jej obsahu a možnosti jej komentovania a zmazania.

[1] http://lesscss.org/
[2] http://www.iconarchive.com/

9 Šprint We Are Scientists!

9.1 Pridanie stiahnutých stránok do profilu

Zodpovedná osoba: Karol Balko

9.1.1 Úloha

Implementácia stiahnutých stránok do profilu používateľa

9.1.2 Implementácia

Táto funkcionalita bola implmentovaná v Backbone.js. Api komunikuje na adrese ,,my.ownet/api/orders“ s parametrom page.

Pre základnú funkcionalitu použité nasledujúce súbory:

- ProfileView - zobrazenie užívateľského profilu, a jeho editácia
- šablóny pre zobrazenie v prehliadači

Boli implementované nasledujúce funkcie:

- showDownloadOrders - získanie kolekcie stiahnutých stránok a ich zobrazenie

9.2 Služby na reportovanie pádov a sprístupnenie súborov na aktualizáciu aplikácie

Úloha #OWNNET-283, zodpovedná osoba: Matúš Tomlein

9.2.1 Úloha

Implementovanie centrálnych webových služieb na oznámenie pádov klientskej aplikácie. Implementovanie spôsobu na zverejnenie súborov na aktualizáciu aplikácie cez webové služby.

9.2.2 Analýza

Na implementáciu webových služieb na oznámenie pádov klientskej aplikácie by sa mali použiť existujúce REST webové služby v .NET.

Systém na prístup k súborom na aktualizáciu aplikácie by mal umožňovať jednoduchú administráciu súborov.

9.2.3 Návrh a implementácia

Boli implementované webové služby na oznámenie pádov, ktorým sú posielané informácie o páde spolu s ID pracovnej skupiny a ID klienta. Tieto informácie sú potom uložené do databázy.

Prístup k súborom na aktualizáciu aplikácie bol umožnený prostredníctvom virtuálneho priečinka v IIS. Tento virtuálny priečinok je zdieľaný ako Dropbox priečinok ku ktorému majú členovia tímu rýchly prístup zo svojich počítačov.

9.3 Aktualizácia aplikácie

Zodpovedná osoba: Martin Lipták

9.3.1 Úloha

Umožnenie automatickej aktualizácia aplikácie po jej nasadení. Primárne sa zamerať na OS Windows, keďže ten je cieľová platforma pri prvých nasadeniach.

9.3.2 Analýza

Hľadal som existujúce riešenia aktualizácie na Windows.

Mendeley Update Installer je jednoduchý multi-platformový program na aktualizáciu qt aplikácie po stiahnutí súboru s updatom (súbor treba stiahnuť samostatne a potom spustiť update installer, počká na ukončenie aplikácie a prepíše súbory).

https://github.com/mendeley/Update-Installer

Ferfor sa snaží fungovať ako Sparkle na Mac OS X. Je multi-platformový. Vie zobraziť update dialóg pre používateľa v prípade aktualizácie. Používateľ musí ale binárku sám stiahnuť a nainštalovať.

https://github.com/pypt/fervor

Zaujímavá je open-source Google Omaha, ktorú používa Google na všetky svoje projekty. Je iba pre Windows. Okrem automatickej kontroly aktualizácii vyrieši samotnú inštaláciu a crash dumpy. Možno sa bude dať aplikácia po páde znova naštartovať. Asi má aj podporu pre proxy, ale nebude dobré (kvôli prípadom, keď proxy kvôli chybe nebude fungovať) ťahať aktualizačné binárky cez ownet proxy. Aktualizácie sa pre viac počítačov musia stiahnuť viackrát, toto by sme vyriešili asi len vlastnou službou. Omaha funguje v dvoch častiach - klient (one-click installer služby aj s aplikáciou cez browser) a server, ktorý ponúka metadáta a updaty.

http://omaha.googlecode.com/svn/wiki/OmahaOverview.html#_End_User_Installation_

9.3.3 Návrh a implementácia

Existujúce riešenia síce poskytujú veľa funkčnosti a sú dobre otestované, ale taktiež obsahujú veľa funkcií, ktoré nepotrebujeme. Tiež je nutné ich nakonfigurovať, čo napríklad v prípade Google Omaha nie je triviálne. Nakoniec sme sa preto rozhodli pre vlastné riešenie.

Aktualizáciu z veľkej časti vykoná monitorovacia služba OwNetService. Krátko po štarte (po každom štarte aplikácie OwNet, pravdepodobne aj štarte počítača) OwNetu pošle požiadavku na cloudový server.

http://yatta.fiit.stuba.sk/OwNetUpdate/metadata.json

Súbor metadata.json obsahuje potrebné metadáta. Pole last_version je posledná vydaná verzia aplikácie. Pole last_file je názov súboru, ktorý je ZIP archívom so súbormi tejto verzie.

{ "last_version": "1.0.1", "last_file": "qtownet-1.0.1.zip"}

Monitorovacia a aktualizačná služba porovná pole last_version s aktuálnou verziou aplikácie. Ak je dostuplná verzia novšia, stiahne ju, extrahuje súbory do dočasného adresára a spustí aktualizačný skript update.bat. Aktualizačný skript vypne OwNet vrátane monitorovacej služby, prekopíruje súbory z dočasného adresára do adresára aplikácie a znova spustí OwNet spolu s monitorovacou službou.

9.3.4 Testovanie

Funkčnosť bola otestovaná len ručne počas vývoja s testovacími súbormi na aktualizáciu. Je potrebné ďalšie testovanie so skutočnou aktualizáciou a najlepšie na sieti s viacerými inštanciami aplikácie OwNet.

9.4 Zakázanie a povolenie ukladania stránok

Zodpovedná osoba: Martin Konôpka

9.4.1 Úloha

Implementujte ovládanie pre povolenie, resp. zakázanie, ukladania práve prehliadanej stránky vo vkladanom pruhu.

9.4.2 Analýza

Pre navštívenú stránku je možné volať služby, ktoré sa týkajú ukladania stránok pod hlavnou doménou navštívenej stránky:

- *http://inject.ownet/api/cache_exceptions/add* - pridanie výnimky ukladania.
- *http://inject.ownet/api/cache_exceptions/remove* - odstránenie výnimky ukladania.
- *http://inject.ownet/api/cache_exceptions/check* - kontrola, či sa stránka ukladá.

Tieto služby sú volané POST požiadavkami s parametrom URL stránky, ktorej sa týkajú.

9.4.3 Návrh a implementácia

Do vkladaného pruhu do stránky bolo pridané tlačítko, ktorým používateľ pridáva alebo odstraňuje výnimku. Pri navštívení stránky sa kontroluje, či je stránka ukladaná. Volanie služieb sa vykonáva na strane vkladaného rámu, nie pruhu. Preto si pruh a rám vymieňajú správy.

Pruh reaguje na správu *caching:checked* s parametrom *is_exception*, ktorý definuje, či na doménu stránky je nastavená výnimka ukladania.

Rám reaguje na správy *caching:check* a *caching:change* pre kontrolu ukladania a zmenu ukladania. Podľa stavu ukladania sa mení ikona v pruhu.

9.4.4 Testovanie

Implementáciu sme testovali ručne sledovaním vykonávania kódu, ale aj pomocou ovládacieho prvku vo vkladanom pruhu.

9.5 Oprava objednávania a zvýrazňovania odkazov

Zodpovedná osoba: Martin Konôpka

9.5.1 Úloha

Upravte spôsob získavania odkazov zo stránky ako pre prednačítavanie, tak aj zvýrazňovanie odkazov dostupných offline. Proxy už neposkytuje obsah stránok vždy v textovej podobe, obsah už môže byť aj komprimovaný pomocou *gzip*.

9.5.2 Analýza

Odkazy na stránke nie je možné získať z komprimovaného zdrojového kódu stránky. Preto navrhujeme odkazy vyberať vo vkladanom JavaScript skripte do stránky.

9.5.3 Návrh a implementácia

Pri vyvolaní funkcionalít na prednačítanie alebo zvýraznenie odkazov sa najprv získajú všetky odkazy zo stránky priamo v JavaScript súbore. Potom sa odošlú pomocou správ do vloženého rámu na stránke. Rám je na doméne ownet, takže môže priamo dopytovať modul prednačítavania pomocou AJAX volaní. Zdrojový kód rámu po získaní odpovede odošle správu s výsledkom do vloženého skriptu v stránke,

- Skript vkladaný do stránok počúva na správu *highlight*, v ktorej získava zoznam odkazov na zvýraznenie.
- Skript vkladaného rámu počúva na správu *prefetch* so zoznamom maximálne 3 odkazov (objednávok) na prednačítanie.

Komunikácia je pri zvýrazňovaní aj pri objednávaní stránok na prednačítanie jednosmerná. Služby vykonávajúce požiadavky majú rovnaké rozhrania, ale teraz už neprehľadávajú kód stránky. Kód je totiž nedostupný na strane proxy, iba prehliadača.

9.5.4 Testovanie

Implementáciu sme testovali ručne sledovaním vykonávania kódu, ale aj pomocou existujúcich ovládacích prvkov zvýrazňovania kódu.

9.6 Zoznam objednávok na stiahnutie

Zodpovedná osoba: Martin Konôpka

9.6.1 Úloha

Implementujte služby pre získanie a správu objednávok na stiahnutie. Služba má vracať zoznam stránok s ich adresou, ID, informáciou o dokončení. Služba taktiež poskytuje možnosť stránkovania zoznamu a mazania odkazov.

9.6.2 Analýza

Pri volaní služby sa bude dopytovať tabuľka *prefetch_orders* o zoznam záznamov, ich počet a mazanie.

9.6.3 Návrh a implementácia

Modul prednačítavania bol rozšírený o REST službu, ktorá je umiestnená na adrese:

http://_.ownet/api/orders/

Služba poskytuje tieto akcie:

- **INDEX – GET požiadavka vykonaná na adresu služby. Vráti zoznam odkazov, zoznam je možné stránkovať s parametrom *page=PAGE*.**
- **DELETE – DELETE požiadavka vykonaná na adresu služby s parametrom ID objednávky, ktorá sa má zmazať. Akcia vracia odpoveď OK, ak sa objednávka zmazala. Inak vracia viac nešpecifikovanú chybu.**
- **allPagesCount – GET požiadavka vráti počet všetkých stránok zoznamu objednávok.**

9.6.4 Testovanie

Zoznam objednávok a ich mazanie bolo testované ručne volaním služby a sledovaním vykonávania kódu. Po implementácii rozhrania bola služba testovaná aj využitím tohto rozhrania.

9.7 Zmena posielaných dát pri index akcii v službe správ

zodpovedná osoba: Marek Láni

9.7.1 Úloha

Je potrebné zmeniť formu odosielaných dát pri index akcii správ pre ľahšie zobrazenie v GUI, keďže správy sú delené na dve úrovne a síce ,,statusy“ a komentáre na tieto statusy.

9.7.2 Analýza a návrh

Zoradenie vrátených dát v rámci index akcie správ, by malo obsahovať dve úrovne a síce hornú úroveň budú tvoriť statusy a v rámci každého statusu bude zahrnuté pole s komentármi. Samozrejme treba brať do úvahy stránkovanie, ktoré sa bude týkať iba statusov.

9.7.3 Implementácia

Úloha bola implementovaná v C++ a Qt.

Bolo potrebné vytvoriť dva selekty do databázy, prvý na výber statusov a následne druhý na výber komentárov pre tieto statusy.

Výsledná odpoveď vyzerá napríklad nasledovne:


```
{     "date_created": "2013-04-23T22:31:14",
      "first_name": "Karol",
      "gender": "Male",
      ïd": 4,
      "last_name": "Balko",
      "message": "sdfsdf",
      "parent_id": 0,
      "type": "message",
      üid": "C2732831646_564",
      üser_id": 352035774
      },
      {
      "comments": [
      {
     "date_created": "2013-04-24T20:02:36",
     "first_name": "Marek",
     "gender": "Male",
     ïd": 5,
     "last_name": "Lani",
     "message": "Jasan",
     "parent_id": "C2213397453_525",
     "type": "comment",
     üid": "C2213397453_598",
     üser_id": 404094838
    },
    {
     "date_created": "2013-04-23T12:09:04",
     "first_name": "Karol",
```

```
    "gender": "Male",
    ïd": 3,
    "last_name": "Balko",
    "message": "hello",
    "parent_id": "C2213397453_525",
    "type": "comment",
    üid": "C2732831646_518",
    üser_id": 352035774
  }
  ],
}
```


9.7.4 Testovanie

Testovanie prebehlo prostredníctvom využitia nástroja Poster, cez ktorý sa testovala požiadavka a odpoveď akcie index.

9.8 Správa ako aktivita

zodpovedná osoba: Marek Láni

9.8.1 Úloha

Vytvorte nový typ aktivity typ správa.

9.8.2 Analýza a návrh

Keďže správy sa delia na dva typy (status a komentár) bolo potrebné vyriešiť, čo zo správ bude aktivita. Rozhodli sme sa, že aktivitou bude status. Pri vytváraní správy typu status, sa preto vytvorí aj aktivita.

9.8.3 Implementácia

Implementácia nového typu aktivity prebiehala ako v predchádzajúcich prípadoch (odporúčania, hodnotenia), no vyskytla sa požiadavka, ktorá bola pre správy špecifická a síce pri zobrazení aktivity typu správa sme chceli zobraziť aj počet komentárov na správu od ktorej je odvodená daná aktivita. V index akcii aktivít teda bolo potrebné kontrolovať, či je aktivita typu správa. V prípade, že áno bolo potrebné vykonať selekt, ktorým sa zistil počet komentárov danej správy. Tento počet sa potom pripojil k polu content danej aktivity.

9.8.4 Testovanie

Testovanie prebehlo prostredníctvom využitia nástroja Poster. Boli vytvárané a vymazávané správy, pričom sa kontrolovalo správanie čo sa týka vytvárania a vymazávania skupín v databáze.

9.9 Hlavná stránka s aktivitami používateľov

zodpovedná osoba: Michal Dorner

9.9.1 Úloha

Implementovať hlavnú stránku s aktivitami používateľob - správy, hodnotenia a odporúčania. Pri každej aktivite bude uvedený jej autor a bude možné sa prekliknúť na jeho profil. Správy bude možné komentovať, ale komentáre budú štandardne schované. Aktivity budú podporovať stránkovanie.

9.9.2 Analýza a návrh

Najvhodnejšie bude vytvoriť univerzálny ActivityView, ktorý sa následne použije aj pri zobrazení aktivity skupipín a jednotlivých používateľov. NewsFeedView, ktorý bude tvoriť hlavnú stránku tak bude len obalovať AcitivityView a nastavovať parametre filtrovania aktivít.

9.9.3 Implementácia

Do portálu boli pridane dve backbone View: NewsFeedView a ActivityView. Activity view predstavuje univerzálny pohľad na aktivity, umožňuje nastaviť parametre filtrovania podľa typu, používateľa a skupiny. V nastavených intervaloch komunikuje s proxy serverom a aktualizuje sa. Z portálu sa odstránili šamostatné pohľady na hodnotenia a odporúčania, nakoľko bola táto funkcionalita integrovaná do hlavnej stránky. ActivityView bola tak isto pridaná aj do používateľského profilu, kde zobrazuje aktivity daného používateľa.

9.9.4 Testovanie

Zobrazovanie aktivít bolo otestované spolu s hodnoteniami a odporúčaniami. Otvorilo sa nové okno v prehliadači a ohodnotila sa stránka a následne sa poslalo aj odporúčanie. Na hlavnej stránke sa po najbližšej aktualizácii tieto aktivity úspešne zobrazili.

9.10 Štýlovanie zvyšných obrazoviek portálu

Zodpovedná osoba: Andrea Šteňová

9.10.1 Úloha

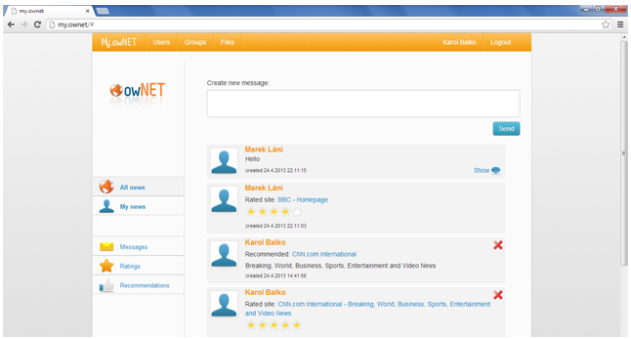
Dokončenie naštýlovania portálu.

9.10.2 Implementácia

Pri obrazovkách som používala grafický náveh vytvorený v predchádzajúcich šprintoch tohto semestra.

Bolo potrebné naštýlovať tieto obrazovky:

- formuláre - registrácia, prihlasovanie, editácia skupín a profilu
- zoznam skupín
- detail skupiny
- zoznam používateľov v skupine
- zoznam aktivít
- profil používateľa
- zoznam požiadaviek na stiahnutie



Obr. 10: Štýlovanie obrazovky so správami

Celý portál je graficky rozdelený na viaceré časti, ako je možné vidieť na obrázku 10, nasledovne:

- horné menu - slúži na prepínanie medzi jednotlivými funkcionalitami portálu
- ľavy blok - informuje o práve zobrazenej funkcionalite portálu (správy, profil používateľa...), ponúka možnosti filtrovania obsahu a obsahuje ďalšie možnosti práce s obsahom (editácia, vytváranie nového obsahu...)

Všetky obrazovky sú naštýlované konzistentne, takže správy, hodnotenia aj odporúčania sú zobrazené rovnako. Všetky obsahujú informáciu o autorovi, dátume vytvorenia a konkrétnej aktivite. Každá informácia o osobe obsahuje linku na jej profil a možnosť mazania aktivity jej autorom.

10 Šprint Rebirth

10.1 Aktualizácia aplikácie 2.0

Zodpovedná osoba: Martin Lipták

10.1.1 Úloha

Zlepšenie automatickej aktualizácia aplikácie po jej nasadení. Cieľom je, aby sa znížili požiadavky na dátové prenosy pri aktualizácii.

10.1.2 Analýza

Zvažoval som niekoľko možností, ako znížiť požiadavky na prenosové pásmo pri aktualizácii aplikácie OwNet.

Prvou bolo nahradenie celej aktualizácie gitom, čím by sa automaticky zabezpečila delta aktualizácia (stiahnutie iba častí, ktoré sa zmenili) a významne by sa zjednodušil celý proces. Nevýhodou je, že tie isté časti by sa opakovane sťahovali na všetkých počítačoch v sieti a tiež úspora pri práci s binárnymi súbormi v gite nemusí byť taká vysoká ako pri textových súboroch, na ktoré sa používa verziovací systém štandardne.

Druhou možnosťou je stiahnutie aktualizácie iba na aktualnom serveri a prešírenie tejto aktualizácie po sieti na aktuálnych klientov v sieti. Pri tejto možnosti sa dá využiť kód vytvorený pre predchádzajúcu verziu aktualizácie. Nevýhodou je, že zabezpečenie delta aktualizácii by vyžadovalo väčšie úsilie. Táto funkčnosť však nemusí byť na začiatku k dispozícii, aktualizácia sa sťahuje len raz.

Nakoniec sme sa rozhodli pre druhú možnosť.

10.1.3 Návrh a implementácia

Aktualizácia na strane aktuálneho servera funguje podobne ako v predchádzajúcej verzii aktualizácie (úloha Aktualizácia aplikácie v šprinte We Are Scientists). Kód, ktorý vykonáva aktualizáciu sa presunul z monitorovacej služby do modulu UpdateModule samotnej aplikácie. Kontrola aktualizácie prebieha po štarte (2 minúty, aby sa zaručene stihla nastaviť rola na sieti a aby bolo jasné, že aktuálna inštancia je server) a periodicky (každé 2 hodiny). Pri dostupnosti aktualizácie server stiahne súbor s tým, že seba nastaví pri sťahovaní ako proxy. Týmto sa stiahnutý súbor uloží do cache.

Pri voľbe servera sa zabezpečí, aby inštancia, ktorá má najvyššiu verziu na sieti, bola serverom. V prípade, že server má vyššiu verziu ako klient, ktorý sa s ním ide synchronizovať, synchronizácia neprebehne (keďže môžu mať rozdielne schémy databázy, ktoré by mohli byť nekompatibilné).

Keď klienti na sieti objavia server, ktorý má vyššiu verziu ako majú oni, spustia aktualizáciu klienta. Vtedy sa vykoná rovnako kontrola aktualizácie bez proxy (či sa náhodou medzičasom neobjavila ešte vyššia verzia) a získa sa URL balíka s aktualizáciou. URL sa stiahne s tým, že sa aktuálna inštancia nastaví ako proxy pre sťahovanie, aby sa stiahnutý balíček dostal do cache klienta v prípade, keď sa server vypne. A aby sa balíček neťahal zo siete znova, aktuálny server sa nastaví ako proxy pre celú aplikáciu OwNet a je stiahnutý cez aktuálny server.

10.1.4 Testovanie

Funkčnosť bola otestovaná len ručne počas vývoja s testovacími súbormi na aktualizáciu. Je potrebné ďalšie testovanie so skutočnou aktualizáciou a najlepšie na sieti s viacerými inštanciami

aplikácie OwNet.

10.2 Častejšie mazanie odkazov na prednačítanie

Zodpovedná osoba: Martin Konôpka

10.2.1 Úloha

Odkazy na prednačítanie sa zbierajú z každej navštívenej stránky (po uplynutí minútového pobytu na stránke). Ich mazanie sa však vykonáva neskoro, čo spôsobuje rýchly nárast zoznamu odkazov a k ich mazaniu sa počas používania aplikácie nedostane.

10.2.2 Analýza

Limit na vykonanie mazania odkazov je nastavený na príliš vysokú hodnotu. Mazanie nastanie až po 50 pokusoch o prednačítanie.

10.2.3 Návrh a implementácia

Limit na vykonanie mazania bol znížený na 2 pokusy. Časovač pre vykonanie jedného pokusu je nastavený na 3 minúty Každé 3 minúty sa vykoná pokus o prednačítanie. Ak je počet objektov stiahnutých proxy serverom menší ako 50, tak sa spustí prednáčítavanie. V rámci vykonávania prednačítania sa však mažú odkazy (spomenuté každé 2 pokusy).

V rámci implementácie bolo zjednodušené otváranie odkazov na prednačítanie. Keďže sa vkladaný rám do stránky nevkladá na strane proxy servera, ale ako rozšírenie prehliadača, tak nie je nutné rozlišovať pôvod otvorenia stránky a pre prednačítanie priamo otvoriť adresu stránky. Tým pádom boli odstránené nasledujúce služby z PrefetchingService:

- load - otvárala stránku odoslanú ako parameter, tým pádom sa v histórii prehliadania (pri prednačítavaní) ako prvý odkaz nachádzala stránka s našou doménou, čo rozlišoval vkladaný skript.
- done - oznámenie o ukončení stiahnutia stránky. Keďže sa do stránky nevkladá skript, nie je možné oznámiť či bola stránka stiahnutá.

10.2.4 Testovanie

Testovanie mazania odkazov sme vykonali sledovaním vykonávania zdrojového kódu počas prehliadania stránok. Pri navštívení stránky sa po uplynutí 1 minúty pridajú nové 3 odkazy do zoznamu pre prednačítanie. Mazanie odkazov bolo kontrolované cez zoznam dostupný v portáli. Mazanie ponechá v zozname 10 odkazov s najvyššou prioritou.

10.3 Opravenie zvýrazňovania odkazov

Zodpovedná osoba: Martin Konôpka

10.3.1 Úloha

Na stránkach s viac ako (približne) 220 odkazov nebolo možné zistiť, ktoré odkazy sú už stiahnuté, a tak ich zvýrazniť. Oprave túto chybu, identifikujte prečo sa vyskytuje.

10.3.2 Analýza

Pre zistenie stiahnutých odkazov stránky sa odosiela zoznam všetkých odkazov (na webové stránky) z danej stránky na službu. Tým pádom je požiadavka veľká podľa počtu a dĺžky odkazov. Modul prednačítavania na strane proxy servera prijme zoznam odkazov a overí, ktoré z nich sú uložené, podľa čoho vracia zoznam odkazov späť. Vrátené odkazy sa na stránke zvýraznia.

Ak je zoznam odosielaných odkazov väčší ako (približne) 220 odkazov, tak požiadavka na proxy server nepríde. Pravdepodobný je teda problém veľkosti požiadavky. Limit veľkosti prichádzajúcej požiadavky na proxy je 16 000 B.

10.3.3 Návrh a implementácia

Limit veľkosti prichádzajúcej požiadavky nie je nastavený s konkrétnym zámerom, je možné ho zvýšiť. Limit bol zvýšený na 160 000 B, čím docielime príjem väčších požiadaviek. Limit je nastavený v triede RequestReader.

10.3.4 Testovanie

Funkcionalita zvýrazňovania odkazov bola testovaná na stránkach so žiadnym odkazom, s desiatkami odkazov i stovkami odkazov. Pre odhalenie problému boli postupne odosielané požiadavky s rastúcim počtom odkazov, až sme narazili na limit 16 000 B. Pokiaľ odkazy stránky nie sú uložené, nezvýraznia sa žiadne. Ak sa identifikuje odkaz, ktorý je stiahnutý, zvýrazní sa oranžovou farbou.

10.4 Obnovovanie aktivít

Zodpovedná osoba: Andrea Šteňová

10.4.1 Úloha

Automatické pravidelné obnovovanie aktivít

10.4.2 Analýza

Upravenie aktuálneho obnovovania aktivít, ktoré vždy načíta nanovo obsah aktivít. To však nie je vhodné, pretože nepodporuje pridávanie komentárov, ktoré ak sa neodošlú, sa pri obnovení zoznamu vymažú.

10.4.3 Implementácia

Obnovenie aktivít sa volá pravidelne každých 10 sekúnd ako aj pri každej zmene (pridanie a zmazanie aktivity, pridanie a zmazanie komentáru alebo zobrazenie ďalšej strany). Toto obnovenie funguje podobne ako prvotné získanie údajov, pomocou metódy fetch, ktorú som potrebovala upraviť.

Upravená metóda získa iba zmenené aktivity a upraví ich v kolekcii - zozname všetkých načítaných aktivít. Nakoniec ich preusporiada podľa dátumu vytvorenia, lebo táto metóda ich pridáva na začiatok zoznamu.

10.4.4 Testovanie

Testovanie prebiehalo používateľským testovaním v nasledujúcom scenári:

1. Vymazanie všetkých aktivít (alebo databázy)
2. Vytvorenie 5 aktivít, očakáva sa obnovenie zoznamu po pridaní každej aktivity
3. Pridanie aktivity iným používateľom, pri automatickom obnovení sa očakáva zobrazenie tejto aktivity ostatným používateľom
4. Pridanie komentáru k 3. aktivite, očakáva sa obnovenie zoznamu s touto aktivitou stále na 3. mieste
5. Zmazanie 4. aktivity, očakáva sa obnovenie zoznamu, ktorý nebude obsahovať túto aktivitu

10.5 Nekonečné skrolovanie aktivít

Zodpovedná osoba: Andrea Šteňová

10.5.1 Úloha

Zrušenie stránkovania aktivít ako sú správy, odporúčania a hodnotenia, nahradenie nekonečným skrolovaním.

10.5.2 Analýza

Nekonečné skrolovanie aktivít je postupné načítavanie stránok aktivít podľa používateľa bez implicitného stlačenia na tlačidlo ďalšej stránky. Hlavne pri aktivitách, ktorých môže byť veľmi veľa, môže byť preklikávanie medzi stránkami časovo náročné.

Nekonečné skrolovanie načíta ďalšiu stránku obsahu keď používateľ naskroluje na spodnú časť stránky. Služby v moduloch ostávajú nezmenené, backbone.js aplikácia si pridáva do zoznamu aktivít tie z ďalšej stránky.

10.5.3 Implementácia

Táto úloha súvisí s implementáciou obnovovania aktivít, tiež som upravila metódu na získavanie aktivít z modulu fetch. Túto metódu som upravila tak, aby nevymazávala aktivity zo zoznamu, ale len pridávala nové získané údaje. Aplikácia si pamätá akú stránku naposledy zobrazovala, aj koľko ich maximálne zobraziť môže.

Aplikácia tiež odchytáva používateľské skrolovanie a keď sa touto aktivitou dostane na spodok stránky, zavolá službu na získanie aktivít z ďalšej stránky a pridá ich do zoznamu.

10.5.4 Testovanie

Testovanie prebiehalo používateľským testovaním so zmenením atribútu v službe na počet aktivít na stránku. Vytvorením väčšieho množstva aktivít a zmenšením tohto atribútu sme zvýšili počet stránok. Vypisovaním do logu počtu už načítaných aktivít sme otestovali správne načítanie stránok.

10.6 Backend na zdieľanie súborov

#OWNNET-304/OWNNET-316, zodpovedná osoba: Matúš Tomlein

10.6.1 Úloha

Navrhnúť a implementovať backend pre zdieľanie súborov medzi používateľmi na lokálnej sieti. Toto riešenie by malo umožňovať replikáciu zdieľaných súborov na lokálnej sieti, prístup na zdieľané súbory na aktuálne pripojených počítačoch. Okrem toho je potrebné vytvoriť služby na pridávanie, odstraňovanie a prístup k týmto súborom.

10.6.2 Analýza a návrh

OwNet klienti na sieti komunikujú bez dlhodobo určeného servera. Volia si aktuálny server, ale ten sa môže pomerne často meniť a nedá sa preto spoliehať na to, že ak sa nejaké súbory uložia na aktuálnom serveri, budú vždy prístupné ako to bolo v predchádzajúcej verzii OwNetu.

Je tu ale možné využiť podobný systém ako sa používa na zdieľanie cache web stránok. Ten funguje tak, že klienti si medzi sebou synchronizujú len meta informácie o súboroch s informáciami

na akých klientoch sa nachádzajú, samotné súbory si preposielajú a duplikujú len pri prístupe na ne. Pri zdieľaných súboroch je potrebné zabezpečiť počiatočnú replikáciu súboru aspoň na dvoch počítačoch aby keď sa jeden zo siete odpojí, mohol byť súbor prístupný z drúheho. Populárne súbory sa potom replikujú po prístupe používateľov na viacerých počítačoch.

Zo systému zdieľania cache je možné využiť aj vymazávanie nepopulárnych súborov. Často sa na niektoré súbory zabudne a na počítačoch len zaberajú priestor, pri využití systému na vymazávanie cache budú nepopulárne súbory automaticky zmazávané keď sa zaplní priestor na disku. Je ale potrebné zabezpečiť aby zdieľané súbory mali dlhšiu životnosť ako obyčajná cache.

10.6.3 Implementácia

Pri implementácii sme využili implementovaný systém na ukladanie cache. Po vložení súboru, sa súboru pridelí URL a pod touto URL sa vloží medzi ostatné objekty v cache. Z cache je prístupný ako aj iné objekty po prístupe na vytvorenú URL.

Metainformácie o súbore ako aj zoznam klientov na ktorých je prístupný sú uložené v databáze a synchronizujú sa na lokálnej sieti. Po prijatí správy o novom súbore na aktuálnom serveri si ho aktuálny server zreplikuje do vlastnej cache. Tým sa zabezpečí replikácia na aspoň dvoch počítačoch na sieti. Súbory sa ďalej replikuje po každom prístupe nejakého klienta na ne.

Súbor má oproti iným objektom v cache väčšiu počiatočnú prioritu aby nebol hneď zmazaný po dosiahnutí maximálnej veľkosti cache.

Boli vytvorené aj služby na vkladanie, zmazávanie a zobrazovanie zoznamu zdieľaných súborov.

10.6.4 Testovanie

Implementácia vo veľkej miere využíva už otestované časti aplikácie ako vkladanie do cache, prístup do cache a zmazávanie cache, tak tieto časti boli testované len manuálne.

10.7 Vytvorenie zdieľania súborov v portály

Zodpovedná osoba: Karol Balko

10.7.1 Úloha

Vytvoriť časť portálu, ktorá umožňuje používateľom zdieľať súbory

10.7.2 Analýza

Pri vytváraní tejto funkcionality portálu bolo potrebné okrem samotnej funkcionality nahrávania súborov vytvoriť aj užívateľské prostredie, takisto ako aj službu. Službu mal na starosti Marek Láni. Podľa požiadaviek užívateľské rozhranie muselo obsahovať filtrovanie súborov na užívateľové a všetky a takisto bolo potrebné umožniť užívateľovi pri nahrávaní súboru vybrať skupinu do ktorej sa súbor nahrá.

10.7.3 Návrh a implementácia

Táto funkcionalita sa implementovala pomocou Javascriptu, konkrétne knižnica Backbone.js a služba bola implementovaná v Qt ako samostatná služba. Boli vytvorené súbory FileView.js, ktorý slúži na zobrazovanie tejto časti portálu a prepája ju so službou.

10.7.4 Testovanie

Funkčnosť bola otestovaná len ručne pridávaním súborov rôznych veľkostí a typov, takisto sa testovala možnosť ich mazania.

10.8 Zmena poľa content v aktivitách

Zodpovedná osoba: Marek Láni

10.8.1 Úloha

Úloha pozostávala zo zmeny serializácie obsahu poľa content v tabuľke aktivity. Doposiaľ bola serializácia vykonávaná iba s použitím znaku ;.

10.8.2 Analýza

Analýzou sa zistil nedostatok spôsobu serializácie cez ;, kedy sa neošetrovalo, či používateľ nezadal do polí tvoriach content tento znak. Analýzou sa dospelo, že najlepším riešením bude pole content serializovať do json formátu.

10.8.3 Návrh a implementácia

Pri implementácií sa využila serializcia používaná aj pri volaní služieb a bolo ju potrebné zmeniť pre správy, odporúčania a hodnotenia. Zmena si vyžadovala aj úpravy v portály, ktorú mali na starosti iní členovia tímu.

10.8.4 Testovanie

Funkčnosť bola otestovaná vytvorením objektov odporúčania, správy a hodnotenia. Následne sa pozorovali návratové hodnoty GET volania nad službami sprístupňujúcimi tieto objekty.

10.9 Nahradenie hodnotení aktivitami

Zodpovedná osoba: Marek Láni

10.9.1 Úloha

Úloha pozostávala zo zmeny manipulácie s hodnoteniami, zrušenia tabuľky ratings a reprezentácie hodnotení ako aktivít v rámci databázy.

10.9.2 Analýza

Analýzou sa zistila istá miera plytvania pamäťou pri duplicite dát uchovávaných aj v tabuľke aktivity aj v tabuľke ratings. Rozhodli sme sa preto tento nedostatok zmeniť a reprezentovať hodnotenia ako typ aktivity, pričom dôležité a pre hodnotenia unikátne dáta ukladáme v poli content.

10.9.3 Návrh a implementácia

Bolo potrebné zabezpečiť, aby adresácia rozhrania z portálu zostala rovnaká bez nutnosti rozsiahlych zmien a tak bolo potrebné zmeniť funkcie na vytváranie, zobrazovanie a vzmazávanie hodnotení v službe ratings, aby sa v rámci nich nevytváral/nemenil/nemazal záznam v tabuľke ratings a aby sa správne parsovalo pole content pri zobrazovaní hodnotení.

10.9.4 Testovanie

Funkčnosť bola otestovaná prostredníctvom portálu, kde sa sledovalo správanie pri manipulácií s hodnoteniami.

10.10 Nahradenie odporúčaní aktivitami

Zodpovedná osoba: Marek Láni

10.10.1 Úloha

Úloha pozostávala zo zmeny manipulácie s odporúčaniami, zrušenia tabuľky recommendations a reprezentácie odporúčaní ako aktivít v rámci databázy.

10.10.2 Analýza

Analýzou sa zistila istá miera plytvania pamäťou pri duplicite dát uchovávaných aj v tabuľke aktivity aj v tabuľke recommendations. Rozhodli sme sa preto tento nedostatok zmeniť a reprezentovať hodnotenia ako typ aktivity, pričom dôležité a pre odporúčania unikátne dáta ukladáme v poli content.

10.10.3 Návrh a implementácia

Bolo potrebné zabezpečiť, aby adresácia rozhrania z portálu zostala rovnaká bez nutnosti rozsiahlych zmien a tak bolo potrebné zmeniť funkcie na vytváranie, zobrazovanie a vzmazávanie odporúčaní v službe recommendations, aby sa v rámci nich nevytváral/nemenil/nemazal záznam v tabuľke recommendations a aby sa správne parsovalo pole content pri zobrazovaní hodnotení.

10.10.4 Testovanie

Funkčnosť bola otestovaná prostredníctvom portálu, kde sa sledovalo správanie pri manipulácií s odporúčaniami.

11 Šprint Live Long and Prosper

11.1 Inštalácia aplikácie

Zodpovedná osoba: Martin Lipták

11.1.1 Úloha

Vytvorenie inštalačného balíka s aktuálnou verziou aplikácie pre operačný systém Windows.

11.1.2 Vytvorenie inštalátora

Vytvoril som inštalačný balík a aktuálnou verziou aplikácie rovnako ako v úlohe Inštalácia aplikácie v šprinte Hurry up.

11.1.3 Testovanie inštalátora

Inštalátor bol pretestovaný na virtuálnom stroji s Windows XP bez akýchkoľvek predchádzajúcich inštalácií OwNetu alebo vývojárskych nástrojov. Inštalátor bol tiež pretestovaný na systémoch vývojárov aplikácie OwNet, ktorý používajú operačné systémy Windows 7 a Windows 8.

11.2 Ladenie proxy

#OWNNET-306, zodpovedná osoba: Matúš Tomlein

11.2.1 Úloha

Refaktorovať implementáciu sťahovania webových dopytov a ich ukladanie do cache. Snažiť sa nájsť jednoduchšie riešenia, ktoré by mohli zrýchliť spracovanie dopytov. Skontrolovať, či sú na niektorých miestach neinicializované premenné alebo či sa nepristupuje na zlé miesta v pamäti.

11.2.2 Analýza

Jedným miestom na refaktorovanie je zabezpečenie súčasného čítania sťahovaného dopytu viacerými výstupmi - jeden alebo viac prehliadačov, zápis do cache. Toto spôsobuje značné skomplikovanie architektúry, keďže je potrebné zabezpečiť aby sa dáta dočasne uchovávali v pamäti ale pritom aby ju nezahlcovali.

V aktuálnej implementácii je braný vysoký ohľad na to aby sa nemusel zápis na výstup prehliadaču vykonávať pre rôznych druhy vstupov na viacerých miestach. Toto je samozrejme dobré na znovupoužitie kódu, ale prináša to aj viaceré obmedzenia a komplikuje architektúru systému.

11.2.3 Návrh a implementácia

Na zjednodušenie zapisovania na viaceré výstupy súčasne sme implementovali nasledujúce riešenie:

1. Prvý dopyt na daný objekt vytvorí sťahovanie
2. Počas sťahovania sa dáta zapisujú priamo do cache na disk
3. Na výstup prehliadaču sa zapisuje čítáním zo súboru v cache

4. Keď na daný objekt pristúpi nový čitateľ (prehliadač), sťahovanie sa nezačne znova ale bude len čítať z cache súboru

Toto riešenie značne zjednodušilo implementáciu a zrýchlilo spracovanie dopytov. Slabým miestom riešenia je, že sťahovanie objektu z pohľadu prehliadača môže byť len také rýchle ako je zapisovanie a čítanie z disku. To však je vo väčšine prípadov rýchlejšie ako pripojenie na internet, tak to nemusí byť problém.

Refaktorované boli aj iné časti spracovania dopytu v proxy čo spôsobili menej zdrojového kódu, menej chýb a rýchlejšie spracovanie dopytov. Bolo upravené ukladanie informácii o prístupoch na cache objekty do databázy aby sa vykonávalo menej dopytov na ňu.

11.2.4 Testovanie

Pri testovaní sme použili niektoré existujúce unit testy a testovali sme aj skúšaním aplikácie manuálne na operačných systémoch Windows, Linux (Ubuntu) a Mac OS X.

11.3 Offline stránka - GUI

Zodpovedná osoba: Andrea Šteňová

11.3.1 Úloha

Vytvorenie a naštýlovanie stránky, ktorá sa zobrazí keď je používateľ offline.

11.3.2 Implementácia

Po požiadavke používateľa, ktorá sa nepodarí splniť (používateľ je offline), aplikácia presmeruje používateľa na špeciálnu URL. Implementovala som stránku, ktorá sa mu pri tom zobrazí a poskytuje jednoduchý formulár na pridanie stránky medzi zoznam stránok na stiahnutie.

Po odoslaní formuláru sa stránka pridá medzi stránky na stiahnutie a používateľ je presmerovaný na zoznam stránok na stiahnutie, kde vidí ich stav.

11.3.3 Testovanie

Testovanie prebehlo používateľským testom, keďže ide o jednoduchý formulár, otestovali sa situácie pri odoslaní vyplneného a prázdneho formulára.

11.4 Upravenie zdieľania súborov v portály

Zodpovedná osoba: Karol Balko

11.4.1 Úloha

Refaktorizácia kódu zdieľania súborov v portály, kvôli zmene služby, ktorá sa spojila s aktivitami. Takisto sa zmenili nejaké časti pri zobrazovaní súborov, ako ikony podľa typu súboru, zjednotil sa dizajn formuláru na nahratie súboru a opravovali sa nejaké chyby.

11.4.2 Analýza

Bolo nutné upraviť súbor FilesView.js, ktorý slúži na zobrazenie zdieľaných súborov. Keďže sa služba zjednotila s aktivitami pôvodná verzia už nebola funkčná. Takisto sa upravovali nejaké vizuálne zmeny v GUI na zdieľanie súborov.

11.4.3 Návrh a implementácia

Táto funkcionalita sa implementovala pomocou Javascriptu, konkrétne knižnica Backbone.js a služba bola implementovaná v Qt, ale už ako súčasť aktivít. Bol upravený súbor FileView.js, ktorý slúži na zobrazovanie tejto časti portálu a prepája ju so službou.

11.4.4 Testovanie

Funkčnosť bola otestovaná len ručne pridávaním súborov rôznych veľkostí a typov, takisto sa testovala možnosť ich mazania.

11.5 Nahradenie správ aktivitami + komentáre pre aktivity

Zodpovedná osoba: Marek Láni

11.5.1 Úloha

Úloha pozostávala zo zmeny manipulácie so správami prvej úrovne (statusmi) a ich reprezentácie ako aktivít v rámci databázy. Tabuľka messages má slúžiť už iba na uchovávanie komentárov k aktivitám.

11.5.2 Analýza

Analýzou sme dospeli k názoru, že správy prvej úrovne nie je nutné reprezentovať samostatne a môžu sa stať aktivitou. Rovnako sme dospeli k názoru, že by mohlo byť zaujímavé pridať možnosť komentovať všetky aktivity.

11.5.3 Návrh a implementácia

Pri implementácií sa najskôr vyriešili zmeny pri manipulovaní so správami prvej úrovne v zmysle zmien pri prechode na aktivity pri odporúčaniach a hodnoteniach. Následne bolo nutné vykonať zmenu mapovania komentárov na tabuľku aktivít, konkrétne na stĺpec uid. Tieto zmeny umožnili vytvárať komentáre pre všetky typy aktivít.

11.5.4 Testovanie

Funkčnosť bola otestovaná vytvorením aktivít (statusov, odporúčani, hodnotení), následným vytvorením komentárov, POST požiadavkov na url http://my.ownet/api/messages s json telom { group_id:xx, user_id:yy, contentent:"random text"}. Rovnako bolo vykonané testovanie zmázávania pri zmazaní komentovanej aktivity a pri zmazaní samotného komentáru.

11.6 Manuálne prednačítavanie

Zodpovedná osoba: Martin Konôpka

11.6.1 Úloha

Umožnite používateľovi zadať požiadavku na stiahnutie stránky v prípade, že pripojenie k internetu nie je dostupné.

11.6.2 Analýza

Pri nedostupnosti pripojenia k internetu a navštívenia webovej stránky sa používateľovi zobrazí webová stránka, ktorá informuje používateľa o výpadku pripojenia. To dokážu samotné prehliadače, ale na tejto stránke môžeme umožniť používateľovi zadať požiadavku na stiahnutie stránky v čase dostupnosti pripojenia. Požiadavku na stiahnutie môžeme považovať za požiadavku podobnú predpovediam prehliadania na prednačítavanie.

11.6.3 Návrh a implementácia

Proxy server priebežne kontroluje dostupnosť pripojenia. Pri pokuse o navštívenie stránky, ktorá nie je dostupná v distribuovanej cache sa používateľovi zobrazí stránka s formulárom, pomocou ktorého môže zadať požiadavku na stiahnutie pri najbližšej dostupnosti pripojenia. Používateľ vyplní popis stránky a uloží požiadavku. Požiadavka je podobná predpovediam pri prednačítavaní, ale má väčšiu prioritu na stiahnutie.

Vytvorenie požiadavky sa vykonáva odoslaním požiadavky CREATE (POST) na adresu služby:

```
http://my.ownet/api/orders/
```

Telo požiadavky vo formáte JSON pozostáva z:

- *title* - popis stránky (zadaný používateľom),
- *url* - adresa stránky.

Služba vracia odpoveď podľa úspechu zapísania stránky do zoznamu na prednačítanie (overuje sa zadaný popis a platnosť URL adresy). Potom je používateľ presmerovaný na stránku so zoznamom všetkých objednávok na stiahnutie (bežne prístupná cez jeho profil).

11.6.4 Testovanie

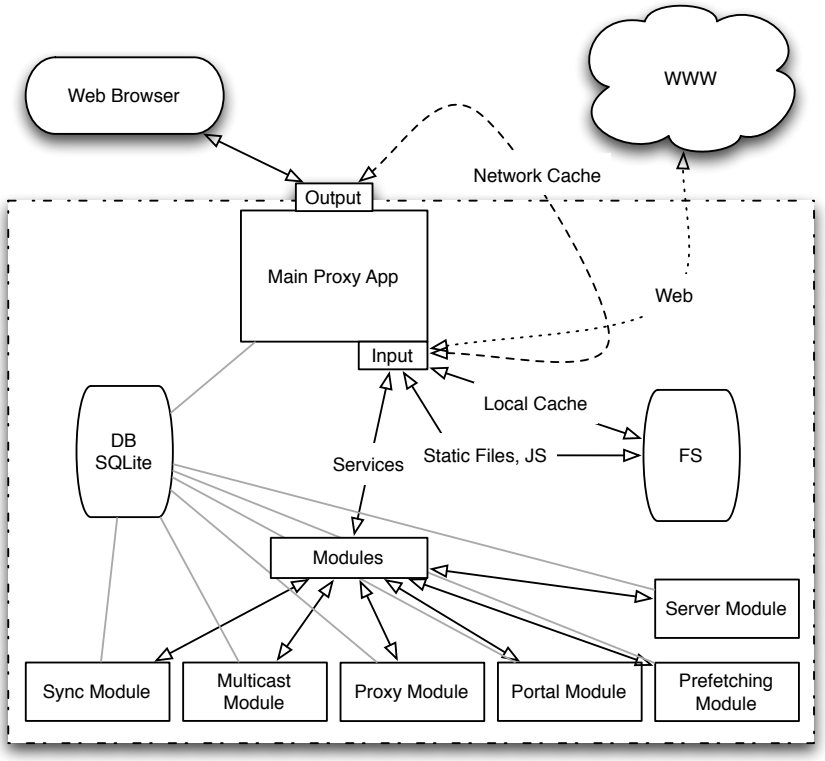
Funkcionalita bola manuálne testovaná sledovaním vykonávania kódu a vykonaním nasledujúcich prípadov:

- Neprázdny popis stránky a platná URL adresa - úspešné pridanie požiadavky.
- Prázdny popis stránky a platná URL adresa - neúspešné pridanie požiadavky.
- Prázdy (neprázdny) popis stránky a neplatná URL adresa - neúspešné pridanie požiadavky.

12 Technický opis produktu

Aplikácia OwNet slúži na skvalitnenie resp. umožnenie prístupu k webovému obsahu aj pri slabom prípadne žiadnom pripojení. Aplikácia ukladá a prednačítava webový obsah a na používateľskom počítači vytvára malú kópiu Webu. Aplikácia OwNet je v aktuálnej verzi zameraná na školské prostredie, a rovnako zahŕňa používateľský portál, prostredníctvom ktorého môžu študenti pracovať a vzdelávať sa s využitím Webu oveľa efektívnejšie. Významnou funkciou aplikácie OwNet je komunikácia počítačov s touto aplikáciou na lokálnej sieti. To znamená, že webový obsah, ktorý sa stiahne je možné distribuovať medzi všetkých prihlásených študentov na danej sieti, čím sa šetrí prenosová linka. Toto distribuovanie je vykonávané za pomoci peer-to-peer komunikácie.

12.1 Architektúra prototypu



Obr. 11: Architektúra systému

Pri tvorbe architektúry sme kládli dôraz na veľkú modulárnosť riešenia, aby bolo v budúcnosti jednoducho rozšíriteľné.

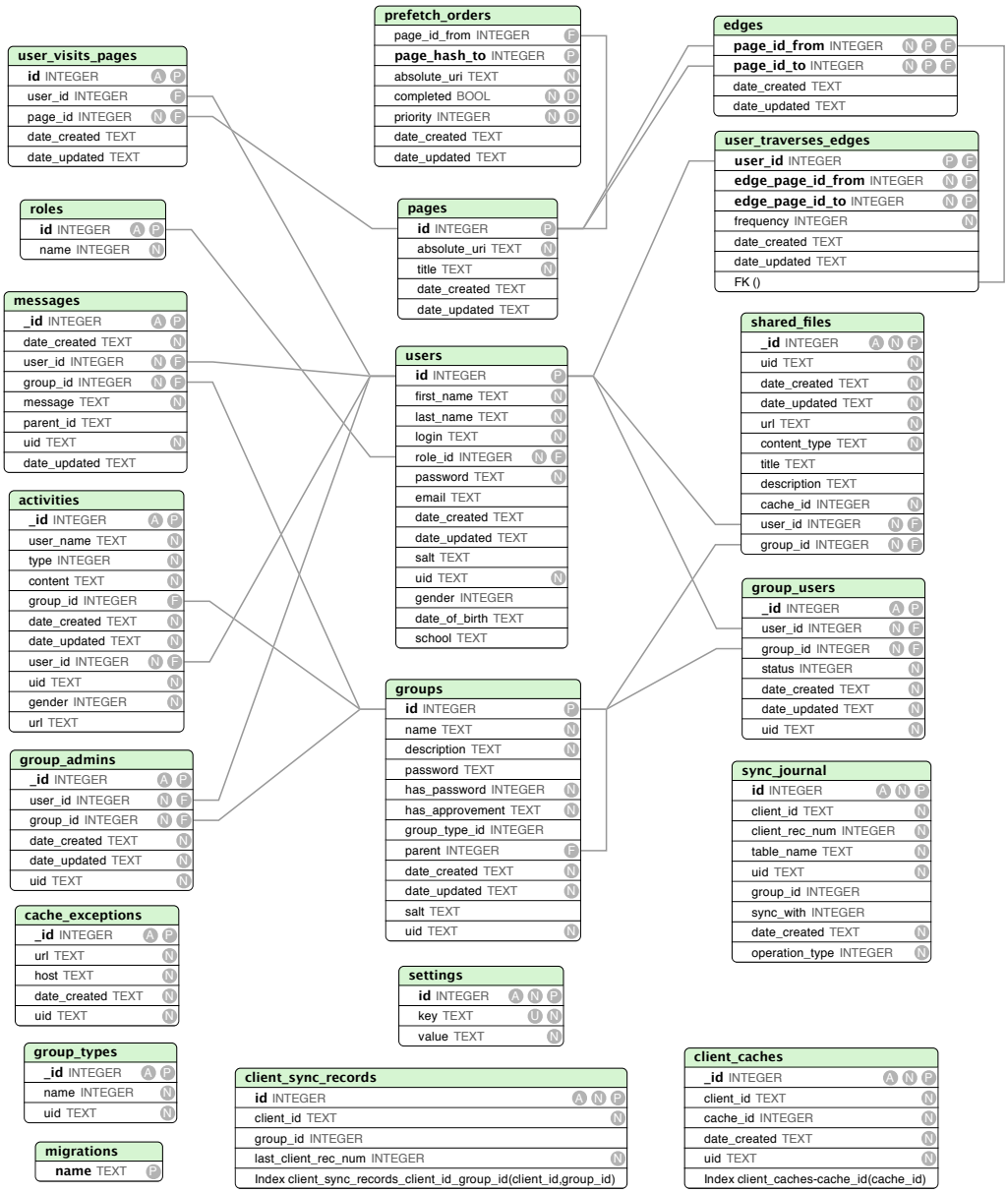
Hlavným komponentom je aplikačná proxy, ktorá riadi hlavnú logiku aplikácie a spracúva dopyty na webové a lokálne objekty a služby. Používateľské a iné dáta sa ukladajú na každom počítači zvlášť v SQLite databáze, ktorá je synchronizovaná naprieč celou sieťou a rovnako cez centrálne služby (cez internet), ktoré majú okrem iného na starosti sledovanie nutnosti obnovovania uložených cache objektov.

Ďalšimi komponentami sú moduly, ktoré sprístupňujú nadstavbovú funkcionalitu a sú to:

- Portal Module - ponúka služby spojené s portálom (odporúčanie a hodnotenie stránok, vytváranie správ, nahrávanie súborov, vytváranie skupín...

- Sync Module - slúži na synchronizáciu dát naprieč sieťou a cez internet, tak aby mal každý študent dostupné aktuálne dáta
- Multicast Module - riadi komunikáciu naprieč sieťou resp. oslovovanie klientov na sieti
- Proxy Module - ponúka doplnkové služby k základnej proxy funkcionalite aplikácie
- Prefetching Module - má na starosti prednačítavanie webového obsahu

12.2 Databázový model



Obr. 12: Databázový model

- *pages* – webové stránky, ktoré navštívil používateľ vo svojej histórii prehliadania. Tabuľka uchováva najmä adresu stránky. Tabuľka je synchronizovaná po sieti.

- *user_visits_pages* – tabuľka uchovávajúca navštívené stránky používateľom ako prepojenie tabuliek používateľov (users) a stránok (pages). Tabuľka je synchronizovaná po sieti.

- *edges* – prepojenia medzi stránkami, ktoré prešiel používateľ. Tabuľka uchováva existujúce orientované prepojenia, nie odkazy, ktoré sú na stránke. Orientáciu prepojenia získame podľa zdroja (page_id_from) a cieľu (page_id_to). Tabuľka je synchronizovaná po sieti.

- *user_traverses_edges* – tabuľka uchovávajúca históriu prechodov používateľa po prepojeniach medzi stránkami (tabuľka edges). Podľa poslednej zmeny záznamu (date_updated) môžeme zistiť posledný prechod. Celkový počet prechodov cez prepojenie je zaznamenaný ako frekvencia (frequency). Tabuľka je synchronizovaná po sieti.

- *prefetch_orders* – tabuľka uchovávajúca objednávky na prednačítanie. Uchováva sa prepojenie na stránku (page_id_from), na ktorej bola požiadavka vykonaná (pri predpovedi na prednačítanie), jej adresa (absolute_uri). Objednávka má svoju prioritu (priority) a indikáciu splnenia (completed). Tabuľka nie je synchronizovaná po sieti.

  item *caches*- Táto tabuľka obsahuje informácie o cache objektoch na danom klientovi. Uchováva počet prístupov na cache objekt, jeho veľkosť a váhu, ktorú potrebuje algoritmus GDSF na vymazávanie cache pri jej zaplnení. Obsah cache je uchovávaný v súboroch na disku, nie v databáze. Táto tabuľka nie je synchronizovaná na sieti.

- *client_caches*- V tejto tabuľke sú informácie o všetkých cache objektoch na lokálnej sieti. Uchováva dvojice - ID klienta na ktorom je cache uložená a ID cache objektu. ID cache objektu je generované ako hash jeho URL adresy. Táto tabuľka je synchronizovaná na sieti aby každý klient vedel o cache na iných klientoch.

- *cache_exceptions*- Sú v nej uložené URL výnimky, ktoré sa nemajú ukladať a načítavať z cache. Tieto výnimky sú spracované na základe host adries v nich a platia pre všetky URL adresy s danou host časťou. Táto tabuľke je synchronizovaná na sieti aby sa cache výnimky zdieľali medzi klientami.

- *sync_journal*- Táto tabuľka slúži na uchovávanie histórie všetkých zmien databázy na sieti. Na základe nej si môžu klienti posielať rozdiely dát v ich databázach vo forme synchronizačného žurnálu. Neobsahuje reálne dáta, ktoré boli v danom kroku pridané, iba referenciu na tabuľky v ktorých boli tieto dáta zmenené.Každý záznam má svoje poradové číslo, ktoré je unikátne v danej skupine a pre daného klienta.

- *client_sync_records*- Obsahuje aktuálny stav záznamov v tabuľke sync_journal - posledné poradové čísla záznamov pre všetkých klientov a skupiny.Tieto informácie by sa dali získať aj zo samotnej tabuľky sync_journal, sú však často používané a preto je efektívnejšie ich uchovávať takýmto spôsobom.

- *shared_files*- Obsahuje zoznam všetkých zdieľaných súborov na sieti. V tejto tabuľke sú uchovávané len meta informácie o zdieľaných súboroch, reálne súbory sú uchovávané ako cache objekty na disku. Táto tabuľka je synchronizovaná na sieti aby sa šírili informácie o zdieľaných súboroch.

- *settings*- Obsahuje dvojice kľúč - hodnota, ktoré sú využívané na uchovávanie nastavení v aplikácii. Táto tabuľka nie je synchronizovaná na sieti.

- *migrations*- Obsahuje zoznam vykonaných migrácii databázy. Pri každom spustení aplikácie sa na základe tejto tabuľky kontroluje či by mala byť ešte nejaká migrácia vykonaná. Táto tabuľka nie je synchronizovaná na sieti.
- *users*- Obsahuje údaje o registrovaných používateľoch. Pri registrácii pre každého používateľa vygeneruje unikátny identifikátor. Tabuľka je synchronizovaná po sieti.
- *group_types* - Záznamy v tabuľke určujú možné roly používateľov v systéme. Tabuľka sa nesynchronizuje po sieti.
- *groups*- Tabuľka slúži na ukladanie skupín vytvorených v rámci portálu.Tabuľka je synchronizovaná po sieti
- *group_types* - Táto tabuľka záhŕňa predom určené typy skupín. Tabuľka sa nesynchronizuje po sieti.
- *group_users*- Táto tabuľka slúži na priraďovanie používateľov do skupín. Za pomoci atribútu status je možné vytvoriť čakanie na členstvo alebo zamietnutie členstva. Tabuľka je synchronizovaná po sieti.
- *group_admins*- Uchováva resp. vytvára administrátorsku rolu používateľov v rámci skupín. Tabuľka je synchronizovaná po sieti.
- *activities*- Tabuľka uchováva vytvorené aktivity typu správa(status), hodnotenie a odporúčanie. Pole content uchováva v serializovanom json formáte špecifické údaje pre každý typ aktivity. Tabuľka je synchronizovaná po sieti.
- *messages*- Tabuľka obsahuje komentáre k aktivitám. Táto tabuľka je synchronizovaná po sieti.

13 Opis aktuálnej verzie

13.1 Opis aktuálnej verzie

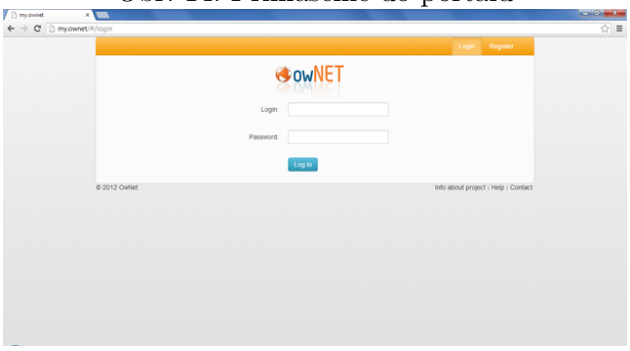
13.1.1 Časť s aplikačnými nastaveniami

Obr. 13: Hlavné okno aplikácie

Hlavné okno aplikácie umožňuje užívateľovi meniť nastavenia proxy servera. Užívateľ má možnosť zvoliť si a pripojiť sa do pracovného prostredia v ktorom sa bude synchronizovať s ostatnými klientmi pripojenými v tomto prostredí. Taktiež má možnosť takéto prostredie vytvoriť.

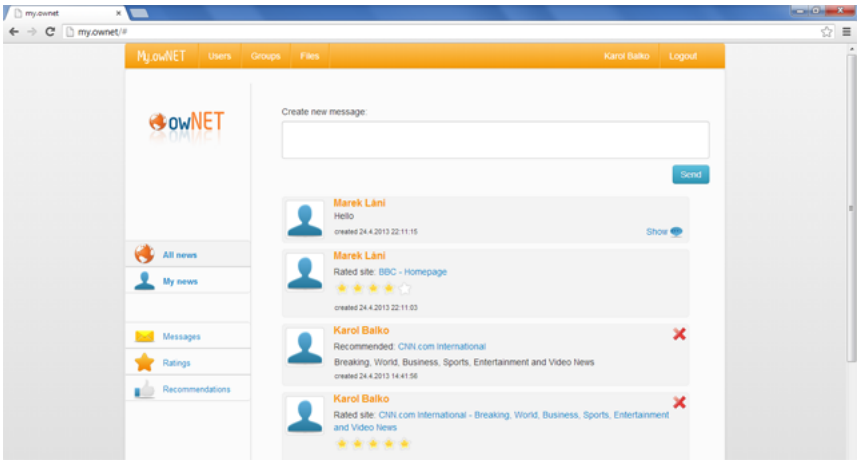
13.1.2 Portálová časť

Obr. 14: Prihlásenie do portálu



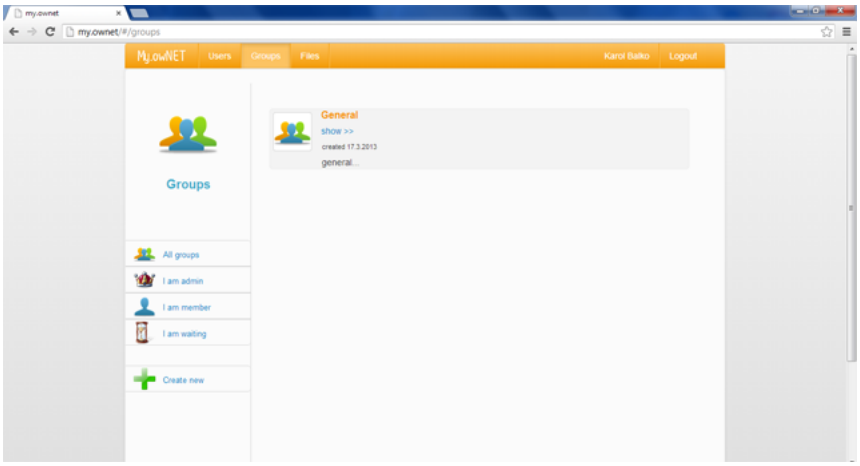
Na zobrazenie portálu sa používa prehliadač Google Chrome. Po prihlásení do portálu sa užívateľovi zobrazí zoznam noviniek, kde sú uvedené stránky ktoré ostatný používatelia hodnotili, alebo odporučili, takisto ako aj správy, ktoré poslali. Priamo zo zoznamu aktivít je možné

správy aj posielať, takisto ako filtrovať novinky podľa typu (odporúčanie, hodnotenie, správy). Ak je užívateľ zároveň tým, čo vytvoril danú aktivitu vie ju aj zmazať.



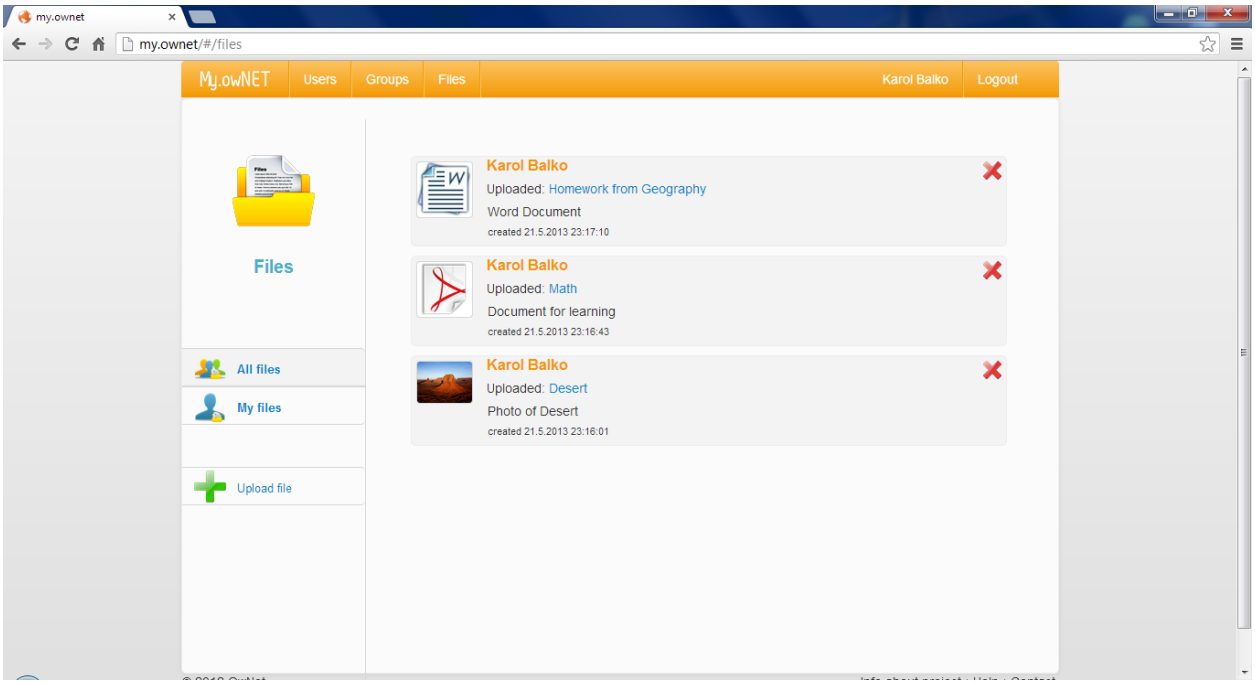
Obr. 15: Zoznam noviniek

Ako ďalšia súčasť portálu je možnosť vytvárať skupiny, do ktorých sa užívatelia môžu pripájať a zdieľať obsah v rámci skupiny.

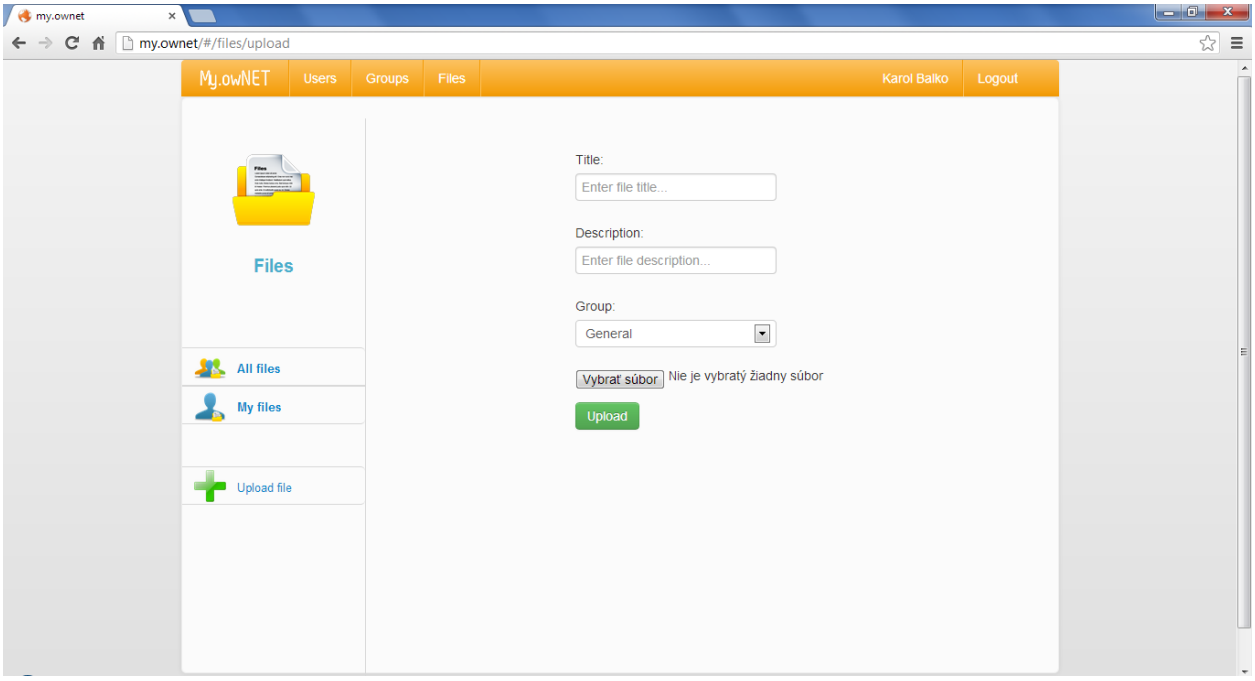


Obr. 16: Sekcia skupín

V sekcií Files je umožnené užívateľom nahrávať súbory, čím ich zdieľajú aj s ostatnými, takisto ako aj ich sťahovať. Pri nahrávaní súboru je takisto možné určiť do ktorej skupiny súbor patrí a tak je možné obmedziť dostupnosť tohto súboru len na určitú skupinu.



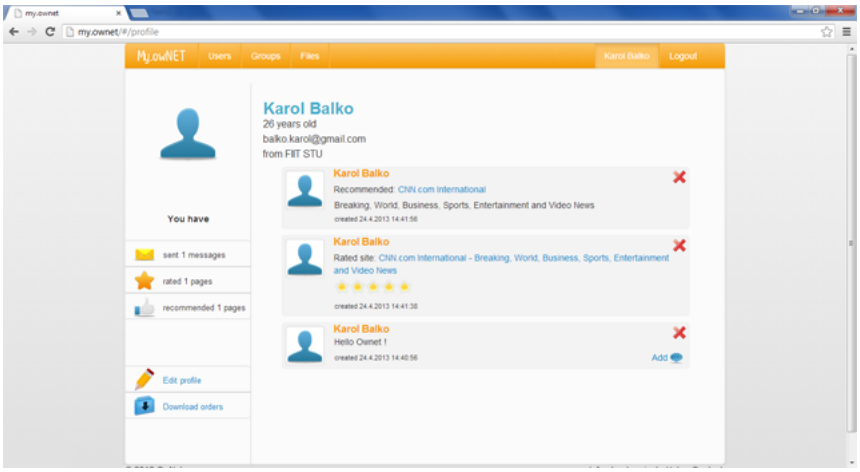
Obr. 17: Sekcia súborov



Obr. 18: Formulár nahrávania súborov

Portál obsahuje aj sekciu profilu, kde je možné okrem zobrazenia vlastného profilu aj jeho upravovanie. Obsahuje štatistiku o počte odporúčaných a hodnotených stránok a aj o počte poslaných

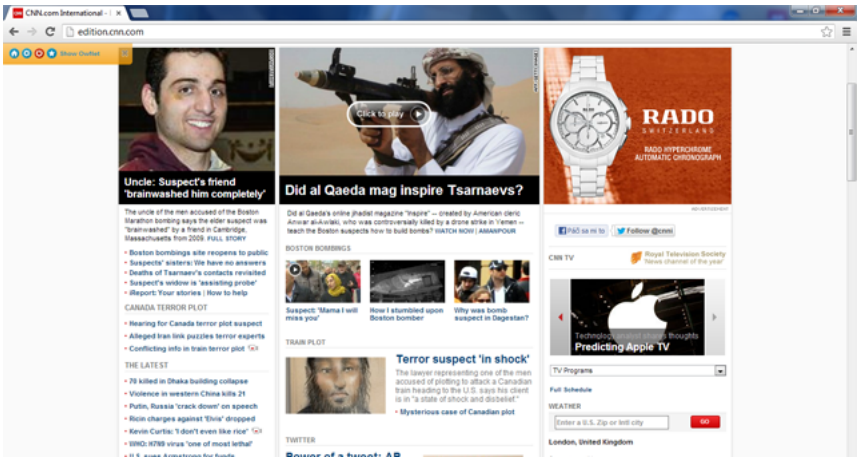
správ. Takisto obsahuje zoznam noviniek, ktoré vytvoril používateľ. Zobrazenie profilu však nie je obmedzené len na práve prihláseného užívateľa, ale aj na ostatných užívateľov.



Obr. 19: Sekcia profil

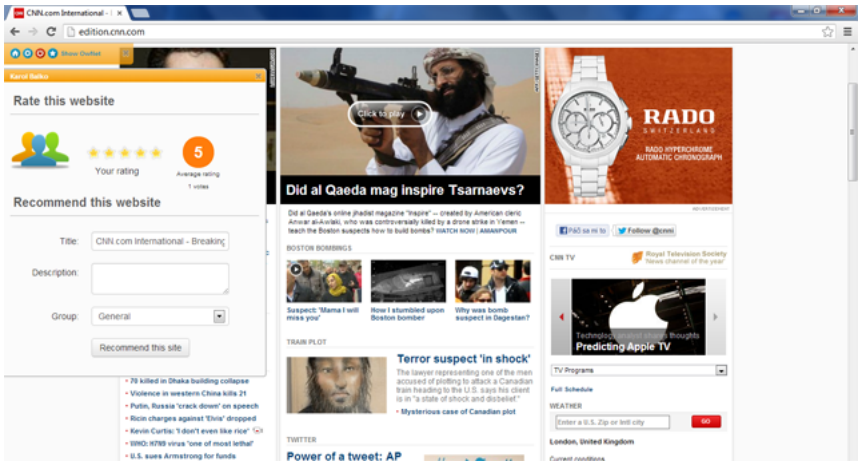
13.1.3 Portálová časť - OwNet Panel

Samotné funkcionality ako pridanie odporúčaní, hodnotení a nastavenie cachovania sú sprístupnené pomocou rozšírenia do prehliadača Google Chrome. Toto rozšírenie pridáva na každú stránku OwNet panel.



Obr. 20: OwNet Panel

Po rozkliknutí je možné odporučiť a ohodnotiť aktuálnu stránku. Tieto hodnotenia sú následne zobrazované v portály, časti zoznam noviniek.



Obr. 21: Otvorený OwNet Panel