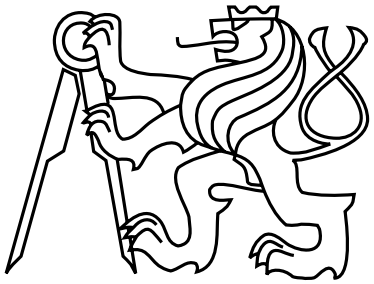
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačové grafiky a interakce



Diplomová práce

Nástroj pro získávání a správu informací

Bc. Václav Hoštička

Vedoucí práce: Ing. Tomáš Novotný

Studijní program: Otevřená informatika, magisterský

Obor: Softwarové inženýrství

11. května 2015

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu Ing. Tomáši Novotnému za odbornou pomoc, cenné rady a trpělivost při vedení této práce. Dále bych chtěl poděkovat svým nejbližším za podporu a důvěru poskytovanou v celé době studia.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 11. 5. 2015 .....

Abstract

This diploma thesis deals with the topic of task managment and sharing. Part of this work is research on the existing solutions in this area used also for requirements specification. The result of the thesis is a tool which enables user to extract data from web pages and save them as tasks or create these tasks manually. User can set various attributes to the tasks, categorize them and share them with other users. Tool can be run in web browser or as an Android application which is also working without connection to server. Solution concerns also with the data synchronization between the devices.

Used technologies include a compiler *Saltarelle* which can compile `C#` language to JavaScript, *Node.js*, a platform for building network applications using JavaScript and a framework *Cordova* for building mobile applications using standard web technologies. Further, the tool utilizes number of libraries, particularly *Socket.IO*, a library enabling real-time bidirectional client-server communication.

Abstrakt

Tato diplomová práce se zabývá oblastí organizace, správy a sdílení úkolů. V rámci práce byla zpracována rešerše existujících řešení, na jejímž základě byly podrobněji definovány požadavky na vyvíjený nástroj. Výsledkem je nástroj, který umožňuje uživatelům extrahovat data z webových stránek a pracovat s nimi jako s úkoly nebo tyto úkoly vytvářet manuálně. Úkolům je možné nastavovat různé atributy, kategorizovat je a sdílet s ostatními uživateli. Aplikace je spustitelná ve webovém prohlížeči i jako aplikace pro mobilní platformu Android, která funguje i bez připojení k serveru. Součástí řešení je synchronizace dat mezi zařízeními.

Mezi použité technologie patří překladač z jazyka `C#` do JavaScriptu *Saltarelle*, platforma pro vyvíjení JavaScriptových síťových aplikací *Node.js* a framework pro vytváření mobilních aplikací za použití webových technologií *Cordova*. Nástroj dále využívá množství knihoven, zejména *Socket.IO* pro obousměrnou komunikaci mezi klientem a serverem v reálném čase.

Obsah

1 Úvod **1**

2 Popis problému, specifikace cíle **2**
2.1 Popis problému . . . 2
2.2 Cíle práce . . . 3
2.2.1 Správa úkolů . . . 3
2.2.1.1 Synchronizace . . . 3
2.2.1.2 Sdílení a spolupráce . . . 3
2.2.2 Extrakce informací . . . 4
2.3 Specifikace požadavků . . . 4
2.3.1 Funkční požadavky . . . 4
2.3.2 Nefunkční požadavky . . . 5

3 Rešerše nástrojů pro sdílení úkolů **6**
3.1 Účel rešerše . . . 6
3.2 Obecné charakteristiky . . . 6
3.3 Přehled nástrojů . . . 7
3.3.1 TickTick . . . 7
3.3.2 Todoist . . . 9
3.3.3 Producteev . . . 11
3.3.4 Taskshare . . . 13
3.3.5 Google Tasks . . . 15
3.3.6 GmailSharedTasks . . . 16
3.3.7 hiTask . . . 17
3.3.8 DropTask . . . 19
3.3.9 Basecamp . . . 21
3.3.10 Remember The Milk . . . 22
3.4 Shrnutí rešerše nástrojů . . . 25

4 Použité technologie **27**
4.1 Saltarelle . . . 27
4.1.1 Použití JavaScriptových knihoven . . . 27
4.2 ExtBrain Framework . . . 28
4.3 Cordova . . . 29
4.3.1 File API . . . 29

4.3.2 SQLite plugin . . . 30
4.3.3 Camera API . . . 30
4.3.4 BarcodeScanner plugin . . . 30
4.3.5 WebIntent plugin . . . 30
4.3.6 Toast plugin . . . 31
4.4 Node.js . . . 31
4.4.1 Express . . . 32
4.4.2 Socket.IO . . . 32
4.4.3 Sqlite3 . . . 32
4.5 SQLite databáze . . . 32
4.6 jQuery . . . 33
4.7 Selectize . . . 33
4.8 LESS . . . 34
4.9 Bootstrap . . . 34

5 Analýza a realizace **35**
5.1 Použité programovací jazyky . . . 35
5.2 Architektura . . . 37
5.2.1 Node.js server . . . 37
5.2.1.1 Server API . . . 38
5.2.2 Databáze . . . 38
5.2.3 Webový klient . . . 39
5.2.4 Mobilní klient . . . 39
5.2.5 Extraktor . . . 39
5.3 Databázové schéma . . . 41
5.3.1 Tabulka Node . . . 41
5.3.2 Tabulka User . . . 42
5.3.3 Tabulka Attachment . . . 42
5.3.4 Tabulka AccessRights . . . 42
5.3.5 Tabulka Settings . . . 43
5.4 Funkcionalita, realizace a uživatelské rozhraní . . . 43
5.4.1 Registrace a přihlášení . . . 43
5.4.2 Správa úkolů . . . 44
5.4.2.1 Práce s daty . . . 45
5.4.2.2 Aktualizace klientů v reálném čase . . . 45
5.4.3 Synchronizace . . . 46
5.4.3.1 Mobilní klient - server . . . 47
5.4.3.2 Server - server . . . 48
5.4.4 Sdílení a spolupráce . . . 48
5.4.5 Extrakce . . . 49
5.4.6 Mobilní získávání dat . . . 50
5.5 Licence . . . 51

6 Testování **52**
6.1 Účel testu . . . 52
6.2 Cílová skupina . . . 52
6.2.1 Screener . . . 52
6.2.2 Pre-test dotazník . . . 53
6.3 Průběh testování . . . 53
6.3.1 Úkoly . . . 54
6.3.2 Post-test dotazník . . . 55
6.4 Participanti . . . 55
6.4.1 Participant 1 . . . 55
6.4.2 Participant 2 . . . 55
6.4.3 Participant 3 . . . 56
6.5 Nálezy . . . 56
6.5.1 Kritické . . . 56
6.5.2 Střední . . . 57
6.5.3 Mírné . . . 58
6.5.4 Návrhy . . . 59

7 Závěr **60**

Literatura **61**

A Seznam použitých zkratek **63**

B Screenshoty uživatelského rozhraní **65**

C Popis instalace a spuštění **68**

D Obsah přiloženého CD **70**

Seznam obrázků

3.1 *TickTick* - přehled webového rozhraní. 7
3.2 *TickTick* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu obsahující podúkoly. 8
3.3 *Todoist* - přehled webového rozhraní. Vpravo zobrazené menu pro úpravu atributů úkolu. 10
3.4 *Todoist* - dvě vybrané obrazovky mobilní aplikace, menu a přehled úkolů. . . 11
3.5 *Producteev* - přehled webového rozhraní. 12
3.6 *Producteev* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu. 13
3.7 *Tasksharе* - přehled webového rozhraní. 14
3.8 *Google Tasks* - rozhraní integrované do Gmailu nacházející se v pravé dolní části obrazovky. 15
3.9 *GmailSharedTasks* - přehled webového rozhraní. 16
3.10 *hiTask* - přehled webového rozhraní. 17
3.11 *hiTask* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu. 18
3.12 *DropTask* - přehled webového rozhraní. Vpravo je zobrazené menu pro editaci atributů úkolu. 19
3.13 *DropTask* - dvě vybrané obrazovky mobilní aplikace, grafické přidávání úkolů a přehled aktivit. 20
3.14 *Basecamp* - přehled webového rozhraní. 21
3.15 *Basecamp* - dvě vybrané obrazovky mobilní aplikace, přehled *todo* listů a nedávných aktivit. 22
3.16 *Remember The Milk* - přehled webového rozhraní. 23
3.17 *Remember The Milk* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a menu. 24

4.1 Ukázka detailu *Toast message*. 31
4.2 Ukázka detailu použití knihovny *Selectize*. 34

5.1 Ukázka možné komunikace jednotlivých částí systému. 38
5.2 Diagram procesu implementace mobilního klienta. 39
5.3 Ukázka instalace bookmarkletu přetažením do lišty záložek. 41
5.4 Schéma databáze, shodné pro server i mobilní aplikaci. První sloupec obsahuje SQLite typ, druhý popis uložených dat a třetí název sloupce v tabulce. 42
5.5 *IAM Toolkit* - přehled webového rozhraní. 43

5.6 *IAM Toolkit* - přihlašování. Aplikace značí, že není připojena přes Socket.IO kanál. 44
5.7 *IAM Toolkit* - *detail přehledu úkolů*.. 45
5.8 *IAM Toolkit* - ukázka rozhraní pro malé obrazovky, vlevo prohlížeč, vpravo mobilní aplikace. Obě mají zobrazené dolní menu pro akce s úkolem. 46
5.9 Sekvenční diagram průběhu synchronizace. 47
5.10 *IAM Toolkit* - logy serverů při synchronizaci, nahoře klient, dole server. 48
5.11 *IAM Toolkit* - nastavování sdílení z webového rozhraní. 49
5.12 *IAM Toolkit* - ovládací prvek extraktoru vložený do stránky na wikipedii. . . 49

B.1 *IAM Toolkit* - přehled webového rozhraní. 65
B.2 *IAM Toolkit* - přidání nového úkolu. 66
B.3 *IAM Toolkit* - detail úkolu. 66
B.4 *IAM Toolkit* - mobilní aplikace. Menu, přidávání úkolu, detail úkolu. 67
B.5 *IAM Toolkit* - mobilní aplikace. Dokončená synchronizace, obrazovka připojení, přihlašování. 67

Seznam tabulek

3.1 Přehled podpory základních funkcí v testovaných nástrojích. 25
3.2 Přehled podpory vybraných pokročilých funkcí v testovaných nástrojích. . . . 26
3.3 Podporované platformy u testovaných služeb. 26

Kapitola 1

Úvod

Tato práce vznikla v rámci výzkumného projektu *ExtBrain* [7], který se zaměřuje na zjednodušování každodenních úkolů vývojářů a dalších, zejména pokročilých, uživatelů, kteří potřebují extrahovat, spravovat a sdílet informace. V rámci své bakalářské práce [1] jsem se věnoval vytvoření grafického rozhraní pro *ExtBrain Extractor*[1] za použití webových technologií a testoval nasazení extraktoru na různých platformách. Práce volně navazovala na bakalářskou práci Jiřího Maška [2], který implementoval GUI pro *ExtBrain Extractor* jako rozšíření pro prohlížeč Mozilla Firefox pomocí Mozilla Application Frameworku. V této diplomové práci ustupuje téma extrakce do pozadí a větší důraz je kladen na správu a sdílení dat.

Zadáním práce je vytvořit nástroj, pomocí kterého bude možné ručně vytvářet či extrahovat data z webu, pracovat s nimi jako s úkoly a tyto úkoly dále kategorizovat, synchronizovat mezi zařízeními a umožnit spolupráci a sdílení s ostatními uživateli. Výsledný program musí být spustitelný ve webovém prohlížeči i jako mobilní aplikace pro operační systém Android. Jedním z požadavků je i závěrečné testování s uživateli.

V následující kapitole *Popis problému, specifikace cíle* (2) je nejprve popsána motivace pro vznik programu a cíle práce. Poté jsou specifikovány požadavky na systém. Třetí kapitola *Rešerše nástrojů pro sdílení úkolů* (3) obsahuje rozsáhlý průzkum existujících otestovaných služeb. Popisovány jsou zejména netradiční funkce a aspekty, u kterých se nabízí přímé srovnání s nástrojem vytvářeným v rámci této práce. Ve čtvrté kapitole s názvem *Použité technologie* (4) jsou představeny stěžejní technologie využité při implementaci. Pátá kapitola *Analýza a realizace* (5) popisuje architekturu systému, jednotlivé komponenty a grafické uživatelské rozhraní. Obsah šesté kapitolu *Testování* (6) je věnován popisu testování s uživateli a kategorizaci nálezů. V závěrečné kapitole (7) jsou shrnuty výsledky práce, její hlavní přínosy a možnosti dalšího vývoje.

Výsledkem této práce je aplikace spustitelná v prohlížeči, aplikace pro mobilní platformu Android a extraktor dat integrovatelný do prohlížeče.

[1] **ExtBrain Extractor** je nástroj napsaný v jazyce JavaScript, který je určený k automatickému získávání dat z webových stránek. Jako vstup přijímá JSON v požadovaném formátu definující pravidla extrakce.

Kapitola 2

Popis problému, specifikace cíle

V této kapitole je nejprve stručně představen problém, na který má vyvíjený program reagovat. Následně jsou vytyčeny cíle práce. Na konci kapitoly jsou shrnuty funkční a nefunkční požadavky na systém.

2.1 Popis problému

První organizéry byly lidmi využívány již před stovkami let[1] a není tedy překvapením, že s vývojem moderních počítačů začaly vznikat jejich elektronické obdoby. Velký rozmach programů pro organizaci úkolů byl zaznamenán zejména s nástupem přenosných zařízení[2]. Obzvláště na takových mobilních přístrojích mají organizéry smysl, protože je uživatel nosí velmi často při sobě.

Přestože vzniká množství těchto aplikací, mnohé spojují shodné nedostatky. Obvyklým problémem je, že jsou zaměřeny pouze na jedinou platformu (např. Apple iOS, nebo Android) a neposkytují webové rozhraní. Dostupné jsou však i velmi propracované systémy, které běží na více platformách a jsou schopny integrovat data z celé řady externích zdrojů.

Představitelem takového vyspělého systému je kupříkladu *Todoist* (3.3.2). Aplikace umožňuje pokročilou kategorizaci do projektů a vnořených úkolů, obsahuje nástroje pro vizualizaci práce na úkolech, upomínání, filtry, sdílení a spolupráci na úkolech pro více uživatelů a to vše dokáže synchronizovat na množství platforem. K tomu nabízí integraci s různými kalendářními, emailovými a jinými externími službami. K dispozici je dokonce i programátorské API.

[1] Pro vytváření seznamů úkolů pro lepší organizování denního programu je známý například Benjamin Franklin, jeden z politiků, kteří se zasloužili o založení Spojených států amerických. Více na http://lifehacker.com/benjamin-franklins-best-productivity-tricks-637033563, nebo https://blog.bufferapp.com/the-origin-of-the-to-do-list-and-how-to-design-one-that-works.

[2] O tom se lze snadno přesvědčit pohledem do online galerií aplikací pro různé mobilní platformy. Např. výsledky vyhledávání dotazu „todo“ v Google play (obchod s aplikacemi pro operační systém Android) na adrese https://play.google.com/store/search?q=todo&c=apps.

2.2 Cíle práce

Z výše uvedeného popisu problému vyplývá, že se jedná o velmi konkurenční prostředí. Některé současné nástroje jsou natolik komplexní, že vytvořit produkt se srovnatelnou funkcionalitou v rámci diplomové práce není realizovatelné. Je tedy třeba nalézt chybějící funkcionalitu.

Jednou takovou oblastí je manuální extrakce z webových stránek, která nebyla nalezena v žádném nástroji pro organizaci úkolů. Další výhodu by mohlo poskytovat serverové řešení aplikace. Uživatel by měl mít možnost pracovat offline nejen z mobilní aplikace, ale po spuštění lokálního Node.js serveru (5.2.1) i z počítače v prohlížeči, například při cestování se špatně dostupným signálem nebo ze zahraničí. Po připojení by mělo být možné data synchronizovat s libovolným dalším serverem, všechny servery by měly být z pohledu systému rovnocenné.

Celá práce má za cíl položit stabilní základy pro budoucí rozšíření. Důraz na architekturu je podporován i zvoleným jazykem pro vývoj `C#`, přestože výsledná aplikace běží v JavaScriptu[3]. Tím je projekt lépe udržovatelný, protože při implementaci lze využívat výhod vysokoúrovňového jazyka a pokročilé vývojářské nástroje (více v sekci 5.1).

2.2.1 Správa úkolů

Jeden ze dvou hlavních požadavků na vyvíjený nástroj je vytváření a správa informací, které jsou v aplikaci reprezentovány úkoly. Konkrétní požadavky na práci s úkoly nebyly specifikovány explicitně, ale vyplynuly přirozeně z rešerše existujících nástrojů (kapitola 3).

Předpokládaná základní sada funkcí obsahuje vytváření a modifikaci úkolů, shlukování do kategorií, označování tagy (štítky), přidávání příloh a další. Aplikace by měla být schopna pracovat s plnohodnotnými podúkoly.

2.2.1.1 Synchronizace

Základním aspektem při práci s úkoly by měla být také jejich snadná synchronizace. K obousměrné synchronizaci by mělo docházet na dvou úrovních. Jednak mezi lokální databází v mobilním zařízení a Node.js serverem, jednak mezi libovolnými dvěma Node.js servery. Každý server by měl být schopný operovat s více klienty, ať už mobilními nebo klienty v prohlížeči.

2.2.1.2 Sdílení a spolupráce

Aplikace by měla umožnit efektivní sdílení úkolů mezi více uživateli a jejich vzájemnou spolupráci. Důraz by měl být kladen zejména na jednoduchost takové operace pro uživatele, ale zároveň by měli mít uživatelé poměrně volné možnosti v tom, jaké úkoly nebo skupiny úkolů sdílet s jakými uživateli.

[3] Překlad z `C#` do JavaScriptu je zajištěn kompilátorem *Saltarelle* (4.1).

2.2.2 Extrakce informací

Druhým hlavním požadavkem na systém je existence nástroje, který umožní uživateli extrahovat části webových stránek. Ten by měl být integrovatelný do webového prohlížeče a podporovat co nejvíce platforem. Bylo rozhodnuto, že původní extraktor, jehož grafické rozhraní bylo vyvinuto v rámci bakalářské práce [1], nebude využit. Při testování s uživateli, zejména s těmi se žádnými nebo pouze povrchními znalostmi HTML a CSS, se ukázalo, že ovládání je stále příliš složité. Vytvořený nástroj by měl mít zejména intuitivní a jednoduché ovládání, kterému může ustoupit pokročilá funkcionalita. Získané části HTML stránek by měli být automaticky uloženy do databáze serverové části aplikace.

Extrakcí informací u mobilních zařízení se rozumí pořizování a nahrávání fotek a také získávání a zpracování dat z Android Intentů[4].

2.3 Specifikace požadavků

Následuje seznam funkčních a nefunkčních požadavků na vyvíjený systém. Požadavky se týkají serverové i klientské části aplikace, ve druhém případě potom obou typů klientů - webového prohlížeče a mobilní aplikace.

2.3.1 Funkční požadavky

1. Vytváření, editace a mazání úkolů.
2. Kategorizace úkolů.
3. Označování úkolů tagy.
4. Vytváření hierarchie úkolů, tzn. práce s podúkoly.
5. Sdílení úkolů mezi více uživateli.
6. Možnost použití mobilní aplikace offline.
7. Synchronizace mezi mobilním zařízením a serverem.
8. Synchronizace mezi libovolnými dvěma servery.
9. Jednoduchá extrakce HTML stránek ve webovém klientovi.
10. Získávání dat z Android Intentů v mobilním klientovi.

[4] **Andorid Intenty** představují konstrukt v operačním systému Android pro sdílení dat mezi aplikacemi. Více v sekci 4.3.5.

2.3.2 Nefunkční požadavky

1. Serverová část bude při běhu využívat JavaScript na platformě Node.js.
2. Oba typy klientů budou realizovány pomocí standardních webových technologií HTML, CSS a JavaScriptu.
3. Mobilní klient pro operační systém Android bude vytvořen pomocí frameworku *Cordova* (4.3).
4. Zdrojový kód pro server i klienty bude vyvíjený v jazyce `C#` a překládán do JavaScriptu pomocí kompilátoru *Saltarelle* (4.1).
5. Zdrojový kód pro vytváření stylů bude využívat CSS preprocesor LESS (4.8)
6. Jednotlivé části systému budou sdílet co největší množství zdrojového kódu.
7. Architektura bude umožňovat snadné budoucí rozšíření programu.
8. Zvolený databázový systém musí podporovat fulltextové vyhledávání. Server i mobilní klient by měly používat pokud možno stejný databázový systém.
9. Grafické rozhraní by mělo být uživatelsky přívětivé a mělo by využívat dostupné knihovny pro budování designu a vytváření prvků pro interakci s uživatelem.
10. Layout bude responzivní, bude se tedy přizpůsobovat různým velikostem zobrazovací plochy.

Kapitola 3

Rešerše nástrojů pro sdílení úkolů

V rámci diplomové práce byla zpracována rešerše již existujících nástrojů pro správu a sdílení úkolů. Kapitola obsahuje stručné představení vybraných služeb s velkým počtem uživatelů.

3.1 Účel rešerše

Zmapované nástroje poskytly přehled současného uživatelského rozhraní a nabídky funkcionalit v této oblasti. Účelem rešerše je tedy zejména podpora konkurenceschopnosti vyvíjeného nástroje. Může však sloužit i samostatně jako poměrně ucelený pohled na aktuální (první polovina roku 2015) možnosti nástrojů zabývajících se organizací a sdílením úkolů.

3.2 Obecné charakteristiky

Všechny testované nástroje umožňují základní správu úkolů a jejich sdílení. Jedinou výjimkou je nástroj *Gmail Shared Tasks* (sekce 3.3.6), který pouze zprostředkovává sdílení *Google úkolů* (sekce 3.3.5) mezi více uživateli.

Většina nástrojů dále obsahuje funkcionalitu pro kategorizaci úkolů, přiřazení tagů nebo přiřazení priorit. Některé nástroje jsou schopny vkládat podúkoly, označovat splněné a prohlížet smazané položky. Kromě standardního řazení a vyhledávání jsou příležitostně k dispozici i filtry pro složitější selekci existujících úkolů. Aplikace také řeší vizualizaci úkolů v čase podle jejich data trvání či dokončení. Objevují se běžné tabulkové seznamy či zobrazení v grafickém kalendářním přehledu. U některých nástrojů lze nastavit opakování událostí společně s upozorněním.

Webový prohlížeč, jako základní běhové prostředí, podporují všechny testované služby[1]. Téměř všechny služby poskytují aplikace pro mobilní platformy Android a Apple iOS. Menší podpora se objevuje pro Windows Phone[2], ostatní systémy pro mobilní zařízení jsou vývojáři

[1] Existence verze nástroje pro webový prohlížeč byla základní podmínkou pro zahrnutí do rešerše.

[2] Platformu Windows Phone z testovaných nástrojů podporuje pouze *Basecamp*, z ostatních např. *Wunderlist* (https://www.wunderlist.com/) nebo *Evernote* (https://evernote.com). Vzhledem k zaměření práce zejména na platformu Android nebyly do rešerše zahrnuty, svým pojetím webového rozhraní připomínají např. nástroj *Todoist* (3.3.2). Přednost proto dostaly služby, které se více odlišovaly.

často opomíjeny[3]. Vzácně se ale objevují desktopové aplikace pro Windows a Mac. Některé aplikace jsou vyvíjeny i jako doplňky pro prohlížeče (Google Chrome, Mozilla Firefox) nebo zcela ojediněle i pro poštovní klienty (Outlook, Thunderbird, Postbox). S podporou více prostředí úzce souvisí i téma synchronizace, ta probíhá většinou automaticky s různou kvalitou. Ne všechny mobilní aplikace umožňují práci s úkoly offline.

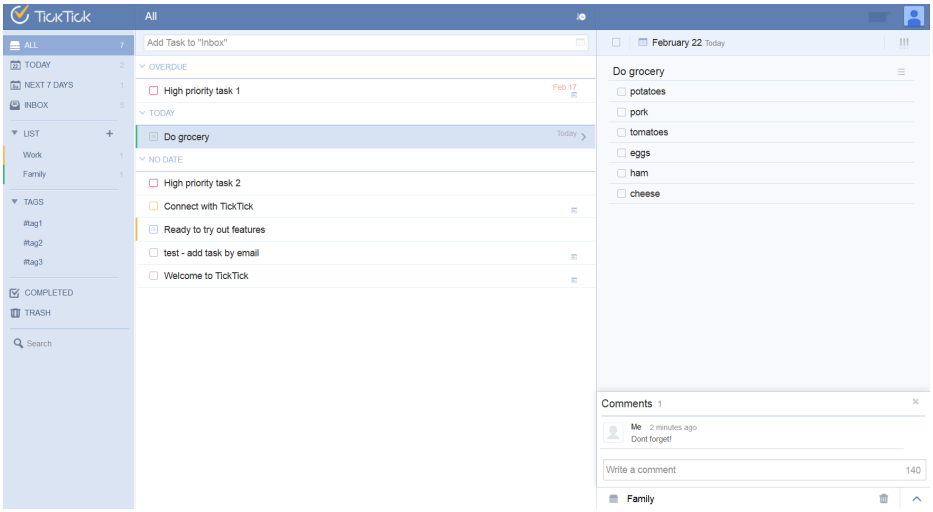
3.3 Přehled nástrojů

Následuje přehled vybraných otestovaných nástrojů. U všech byla otestována základní verze pro prohlížeč a v případě, že existovala, i aplikace pro Android. Pokud byla k dispozici další alternativa, například doplněk do prohlížeče, byla aplikace vyzkoušena i na této platformě.

Text u jednotlivých nástrojů se nezaměřuje na podrobný popis obecné funkčnosti těchto aplikací, ale spíše na specifické aspekty, nadstandardní či jinak neobvyklé vlastnosti, nalezené chyby a problémy při použití.

3.3.1 TickTick

Prvním testovaným nástrojem byla aplikace *TickTick* [17]. Služba je dostupná ve webovém prohlížeči, jako aplikace pro Android a iOS a také ve formě rozšíření do webového prohlížeče Google Chrome. Lze se přihlašovat bez předchozí registrace přes Google, Facebook a Twitter. *TickTick* v základní verzi zdarma poskytuje ve srovnání s ostatními službami nadstandardní množství funkcí.



Obrázek 3.1: *TickTick* - přehled webového rozhraní.

[3] Výjimku v rámci rešerše tvoří nástroj Remember The Milk (3.3.10), který podporuje i platformu BlackBerry

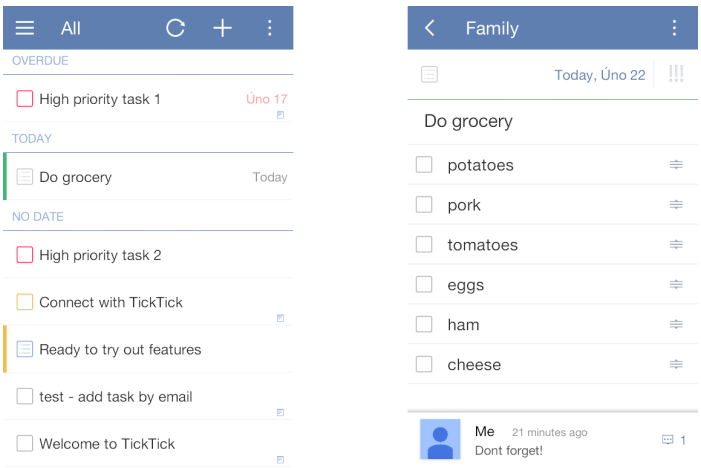
3.3.1.1 Funkcionalita

Program nabízí poměrně široké možnosti správy úkolů, ze základních funkcí sledovaných u všech programů (podrobněji v tabulce 3.1) podporuje všechny. *TickTick* však nabízí podúkoly pouze jedné úrovně, navíc se nejedná o plnohodnotné úkoly se všemi atributy, ale pouze seznam textových odrážek, které lze označit za splněné. Druhým nedostatkem v této oblasti je jednoduchost filtrů, položky se dají selektovat pouze pomocí časových skupin (dnes, příštích 7 dní) a tagů.

S ostatními uživateli nelze sdílet jednotlivé úkoly, ale spolupráce je možná nad celými listy. Dobře funguje i integrace úkolů do Google kalendáře. Lze vygenerovat zálohu a data lokálně uložit na zařízení. Zajímavou funkcí je přidání úkolu pomocí odeslání běžného emailu na uživateli přidělenou adresu. Další specialitou je funkce upozornění závislá na tom, jestli uživatel vstupuje do nějaké lokality nebo ji opouští.

Mobilní verze funguje offline a následná synchronizace po připojení k internetu probíhá většinou automaticky, v případě nutnosti ji lze manuálně spustit tlačítkem. Obsah interního mobilního kalendáře lze přidat mezi úkoly.

Ve verzi zdarma má aplikace některá omezení, například na počet listů nebo počet položek v listu. Lze také přidávat pouze 1 přílohu denně a sdílet list pouze s jedním uživatelem. V placené verzi je pak navíc i kalendářní přehled úkolů a historie revizí.



Obrázek 3.2: *TickTick* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu obsahující podúkoly.

3.3.1.2 Grafické uživatelské rozhraní

Aplikace se vyznačuje čistým a přehledným designem, ve kterém není problém se snadno a rychle zorientovat bez nutnosti úvodních tutoriálů či pomoci uživatelské podpory. Webová verze je rozdělena na tři sloupce (viz obrázek 3.1), díky čemuž lze mít současně zobrazené menu, seznam úkolů i detail úkolu. Layout je responzivní a při snižování šířky okna mění rozložení prvků na stránce. Další výhodou je jednoduché přidávání úkolů pouze zadáním

názvu, veškerá editace parametrů pak probíhá přímo ve stránce tam, kde jsou zobrazeny výsledné hodnoty. Za nedostatek v návrhu GUI by šlo považovat příliš malé a někdy špatně srozumitelné ikonky.

Jednoduchý a srozumitelný design si aplikace zachovává i v mobilní verzi (obrázek 3.2). Rozhraní se dobře ovládá, gesta jsou přesná a odezva je okamžitá. Dobré prostředí nabízí i doplněk do prohlížeče, chybí v něm však některé prvky nastavení.

3.3.1.3 Přehled výhod a nevýhod

Výhody

- Jednoduchý a přehledný design.
- Pohodlná práce s aplikací v režimu offline.
- Integrace s Google kalendářem.

Nevýhody

- Absence plnohodnotných podúkolů.
- Slabé možnosti filtrování.

3.3.2 Todoist

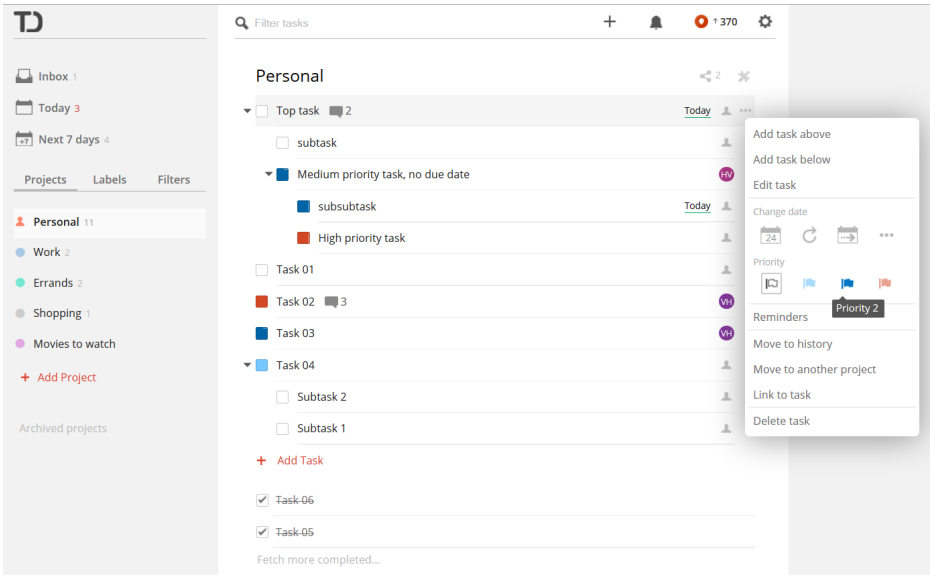
Druhým otestovaným a v rámci rešerše nejpropracovanějším nástrojem byl *Todoist* [18]. V rámci rešerše se jedná o nejvíce multiplatformní službu. *Todoist* podporuje webový prohlížeč, mobilní platformy Android i Apple iOS, dále existuje ve formě doplňků do prohlížečů Firefox a Chrome a do poštovních klientů Outlook, Thunderbird, Gmail a Postbox. Kromě toho lze stáhnout i desktopové aplikace pro operační systémy Windows a Mac. Specialitou jsou verze pro Chromebook a *Android Wear*[4]. Z externích služeb se lze přihlásit přes Google účet. Mimo placené verze, kterou je možné vyzkoušet, je dostupná i varianta zdarma, ovšem bez řady funkcí.

3.3.2.1 Funkcionalita

Z testovaných základních funkcí, shrnutých v tabulce 3.1, podporuje *Todoist* všechny. Jako jeden z mála nástrojů podporuje plnohodnotné podúkoly se všemi atributy do libovolné úrovně. Úkoly jsou rozdělené do několika časových kategorií, uživatel může vytvářet vybírat i podle projektů (listů), tagů, připravených filtrů a může dokonce vytvářet i vlastní filtry. Práce s časem obsahuje inteligentní rozpoznávání zadaných údajů v textové formě.

Stejně jako u předchozího nástroje lze sdílet s ostatními uživateli jednotlivé listy, na aktivity ve sdílených listech je uživatel automaticky upozorňován. K úkolům lze přiřazovat

[4] *Android Wear* je verze operačního systému Android určená pro inteligentní hodinky a další miniaturní přenosná zařízení.



Obrázek 3.3: *Todoist* - přehled webového rozhraní. Vpravo zobrazené menu pro úpravu atributů úkolu.

uživatele. Úkoly je možné přidávat pomocí emailu, skrz obsah emailu lze pomocí speciálních značek přidávat i pokročilejší atributy úkolů, například jeho časové zařazení. Uživatel může dále vytvářet zálohu a tuto proceduru zautomatizovat. Úkoly lze exportovat do některých kalendářních služeb (Google kalendář, Sunrise), *Todoist* lze dále propojit s různými zálohovacími službami (Google Drive, Dropbox), službami pro nastavování akcí[5] a dalšími. K dispozici je i API pro propojení s dalšími aplikacemi. Zajímavostí jsou grafické i textové přehledy aktivity a produktivity uživatelů.

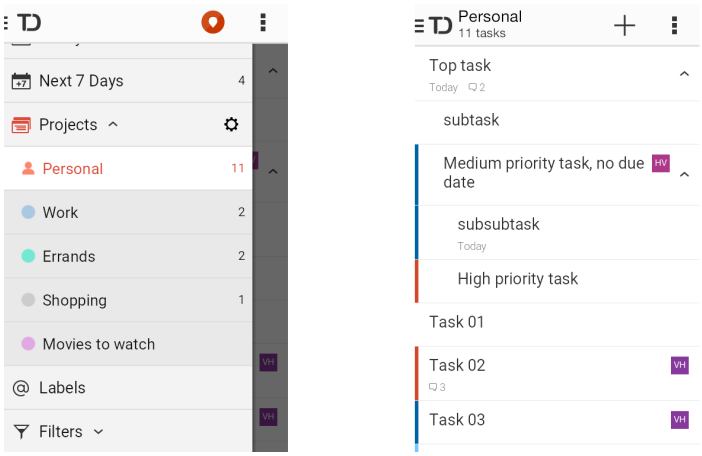
Todoist mobilní aplikace zvládá práci v offline režimu. Následná synchronizace je automatická a probíhá bez intervence uživatele.

Verze zdarma je oproti placené poměrně značně omezená. Chybí všechny pokročilé funkce integrace, zálohy dat nebo přidávání úkolů pomocí emailu. Nelze použít ale i celou řadu v ostatních nástrojích základních funkcí jako jsou tagy, poznámky k úkolům, upozornění nebo přílohy.

3.3.2.2 Grafické uživatelské rozhraní

Design aplikace je velmi přehledný a kompaktní, orientace je okamžitá a ovládání bezproblémové (viz obrázek 3.3). Přidávání a úprava atributů položek probíhá přímo v místě jejich zobrazení. Velká část ovládání se uskutečňuje pomocí drag and drop akcí, například řazení listů a úkolů nebo vytváření hierarchie podúkolů. Design je responzivní, nevýhodou může být nedostatečné využití prostoru na velkých obrazovkách.

[5] Jedná se o tzv. „If This Then That“ nástroje, které umožňují navěsit posluchač na událost v nějaké aplikaci a provést úkon v jiné. Například služba *IFTTT* dostupná na adrese https://ifttt.com/ nebo *Zapier* na https://zapier.com/.



Obrázek 3.4: *Todoist* - dvě vybrané obrazovky mobilní aplikace, menu a přehled úkolů.

Mobilní aplikace je propracovaná a velmi dobře se ovládá (obrázek 3.4). Pohodlné drag and drop akce fungují i na této platformě. Práce s doplňky do prohlížečů je shodná jako práce s webovou verzí v menším okně.

3.3.2.3 Přehled výhod a nevýhod

Výhody

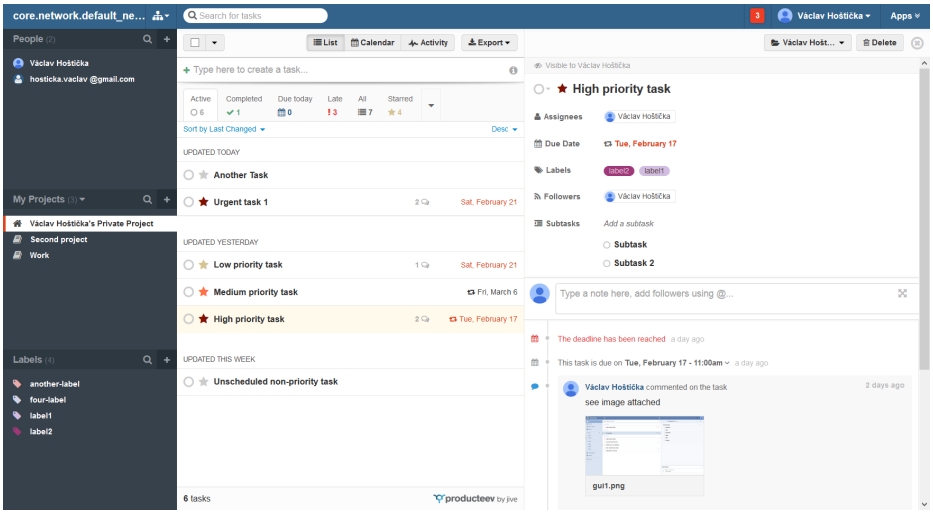
- Propracovaný a přehledný design.
- Integrace s velkým množstvím služeb.
- Vlastní programátorské API.

Nevýhody

- Ve verzi zdarma značně omezené funkce.
- Nedostatečné využití prostoru na velkých obrazovkách.

3.3.3 Producteev

Dalším poměrně rozšířeným nástrojem je *Producteev* [12]. Podporovanými platformami jsou kromě webového prohlížeče operační systémy Android, Apple iOS, Mac a rozšíření pro Outlook. Mimo standardní registraci a přihlášení se uživatel může přihlásit přes Google a Facebook účet. K dispozici je placená varianta i verze zdarma.



Obrázek 3.5: *Producteev* - přehled webového rozhraní.

3.3.3.1 Funkcionalita

Producteev obsahuje všechny funkce vymezené touto rešerší jako základní. Nelze ovšem vytvářet stromovou strukturu podúkolů, k dispozici jsou pouze v jednoduché textové formě a do jedné úrovně. Poměrně široká nabídka je v oblasti řazení a filtrování.

K jednotlivým projektům (obdoba listů) lze přidělovat přístup přidaným uživatelům nebo je celé označit jako veřejné či privátní. Úkolům je možné přiřazovat jednoho či více pracovníků a také tzv. „followerů“, kteří pak obdržují upozornění na změny. Velmi pěkné je zpracování s úkoly v kalendářním přehledu. K dispozici je i přehled historie aktivit. Aplikace obsahuje jednoduchý export do CSV.

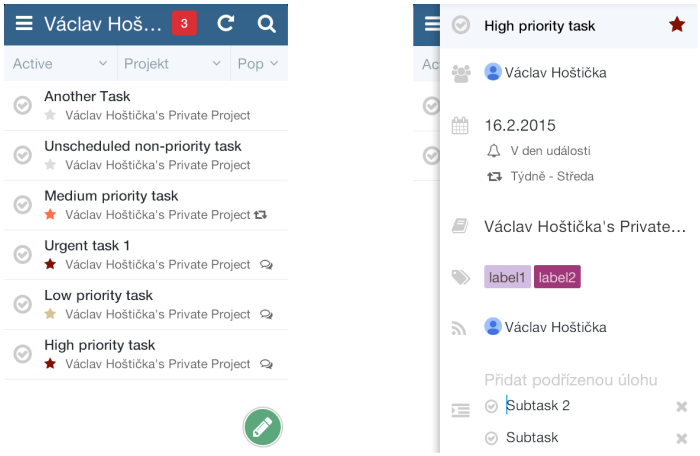
Velkou nevýhodou této služby je, že mobilní aplikace nefunguje offline. Po připojení obsahuje mobilní verze plnou funkcionalitu.

Producteev obsahuje velkou většinu svých funkcí už ve verzi zdarma. V placené variantě obdrží uživatel navíc možnost použití doplňku do Outlooku, personifikaci některých nastavení a zejména plnou zákaznickou podporu.

3.3.3.2 Grafické uživatelské rozhraní

Grafické rozhraní je na první pohled mírně chaotické a orientace v prostředí trvá delší dobu než u předchozích dvou nástrojů. Vzhled připomíná spíše tradiční desktopové programy (viz obrázek 3.5). Aplikace nepůsobí uceleným dojmem, lepší orientaci a ovládání brání rozložení některých prvků na stránce či použití barev. Přeskakování do jiných režimů vzhledu, například při vyhledávání, působí rušivě. Aplikace nereaguje na změny šířky okna.

Mírně nesourodě působí i vzhled a rozložení prvků v mobilní aplikaci (obrázek 3.6). Nesrozumitelné je zejména doplňkové menu pod horní lištou.



Obrázek 3.6: *Producteev* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu.

3.3.3.3 Přehled výhod a nevýhod

Výhody

- Velká nabídka funkcí ve verzi zdarma.
- Snadné a přehledné sdílení projektů.

Nevýhody

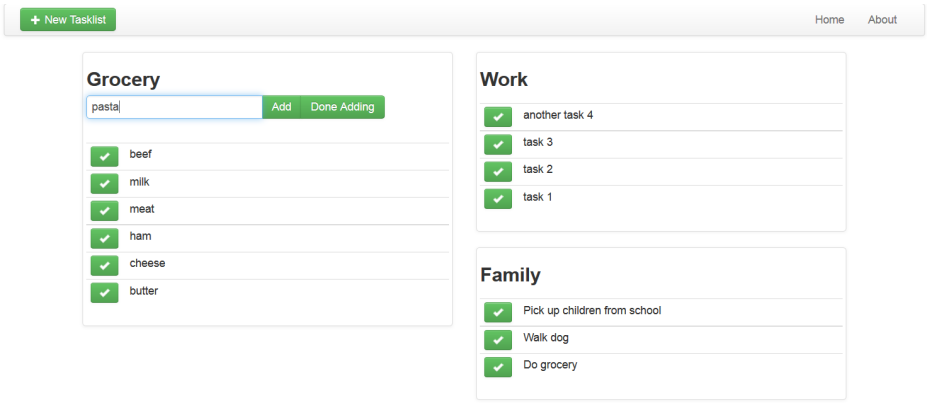
- Nemožnost práce s mobilní aplikací offline.
- Mírně nepřehledný vzhled na všech platformách.

3.3.4 Takshare

Nejjednodušším testovaným nástrojem byl *TaskShare* [16]. K dispozici je pouze webové rozhraní, aplikace pro mobilní platformy neexistují. Služba je zcela zdarma. Jedná se o open-source projekt pod MIT licencí.

3.3.4.1 Funkcionalita

Nástroj slouží pro sdílení listů s textovými úkoly bez dalších doprovodných funkcí (viz tabulka 3.1). Tomu odpovídá i jednoduchost jeho používání. Aplikace nevyžaduje žádné přihlašování, pro každý projekt je vytvořena unikátní URL adresa, která slouží jako přístupový bod do projektu. Přes tuto adresu je možné projekt sdílet mezi libovolným počtem uživatelů.



Obrázek 3.7: *Tasksharе* - přehled webového rozhraní.

3.3.4.2 Grafické uživatelské rozhraní

TaskShare sází na minimalistický přístup a to platí i o vzhledu (viz obrázek 3.7). I při jeho jednoduchosti jsou však některé aspekty používání zbytečně těžkopádné. Po přidání nového listu musí uživatel dvakrát kliknout, aby zvolil jméno. Plovoucí menu pro přidání úkolů, editaci a mazání by mohlo být zobrazeno permanentně, ne až po kliknutí na název listu, design disponuje dostatkem místa a zobrazení skrytého menu je zbytečně neintuitivní, především při používání na dotykových zařízeních. Layout je responzivní a z mobilního prohlížeče jej lze bez větších obtíží používat. Pouze některá tlačítka graficky vyjíždějí z obrazovky.

3.3.4.3 Přehled výhod a nevýhod

Výhody

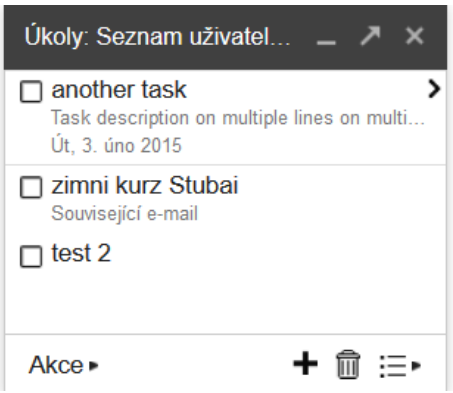
- Vhodný jako zcela jednoduchý úkolovník.
- Snadné sdílení pomocí zaslání URL adresy.
- Okamžité použití bez nutnosti přihlášení/registrace.

Nevýhody

- Absence jakékoliv složitější správy úkolů.
- Nutná samospráva URL.
- Nedomyšlený design rozhraní.

3.3.5 Google Tasks

V rámci rešerše byly také odzkoušeny Google úkoly (*Google Tasks*). Přístup ke Google úkolům má každý uživatel Google přes svůj účet, užívání není nijak zpoplatněno. Mobilní obdoba této služby od Google neexistuje. K dispozici je však řada mobilních aplikací od externích vývojářů, které umožňují synchronizaci s Google úkoly, nebo na nich přímo staví. Jejich komplexní otestování je nad rámec této rešerše, za všechny byla zvolena služba *GmailSharedTasks* (3.3.6).



Obrázek 3.8: *Google Tasks* - rozhraní integrované do Gmailu nacházející se v pravé dolní části obrazovky.

3.3.5.1 Funkcionalita

Jedná se o nepříliš propracovaný nástroj s nevelkou nabídkou funkcí. Jak je patrné ze srovnávací tabulky 3.1, ze základních funkcí nabízí úkoly od Google pouze sdružování do listů a jednoduché řazení. K úkolům lze přidat datum a poznámku, dají se též poslat emailem přes volbu v menu.

Google nabízí podobnou a propracovanější službu Google Kalendář. Přes tu lze vytvářet události a sdílet je s ostatními uživateli. U událostí je možné nastavit termíny od a do, opakování, upozornění, zadat polohu a popis události. Svojí filosofií je tato služba už mimo zamýšlenou oblast rešerše.

3.3.5.2 Grafické uživatelské rozhraní

Nástroj je integrován v layoutu Gmailu. Grafické rozhraní je velmi strohé a působí zastaralým a nekompaktním dojmem (viz obrázek 3.8). Celá aplikace i jednotlivé ovládací prvky jsou velmi malé a ovládání není příliš pohodlné. Příjemným prvkem je drag and drop řazení úkolů.

3.3.5.3 Přehled výhod a nevýhod

Výhody

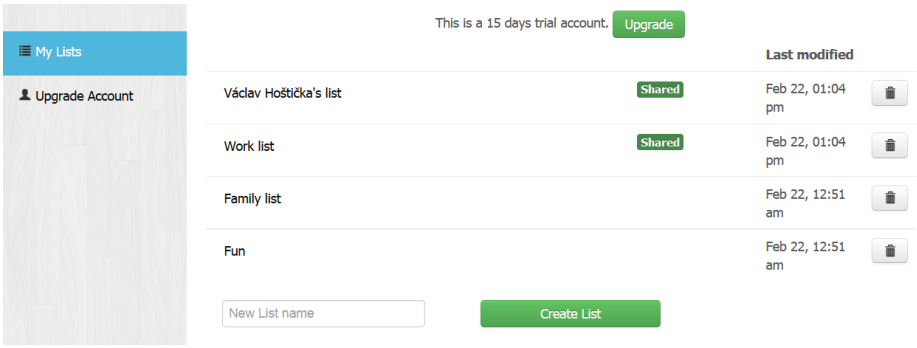
- Integrace přímo v Gmailu.
- Jednoduché poslání listů přes email.

Nevýhody

- Slabá nabídka funkcí.
- Nepřehledné ovládání.
- Neumožňuje sdílení.

3.3.6 GmailSharedTasks

Služba *GmailSharedTasks* se zaměřuje na jeden z hlavních nedostatků úkolů od Google a to zprostředkovat jejich sdílení [8]. Svým pojetím je tedy poněkud odlišná od ostatních, nemůže fungovat samostatně. Přihlašování probíhá přes Google účet ve webovém prohlížeči, mobilní aplikace není k dispozici. Zdarma lze vyzkoušet 15 denní trial verzi, poté je služba zpoplatněna.



Obrázek 3.9: *GmailSharedTasks* - přehled webového rozhraní.

3.3.6.1 Funkcionalita

Program sám o sobě nevytváří úkoly ani s nimi nijak nepracuje. Manipuluje ale s listy (kategoriemi) úkolů, které vytváří, maže a především nastavuje, mezi jakými uživateli budou listy sdílené.

Sdílení probíhá na základě emailů s pozvánkou. Nepříjemný je fakt, že všechny testovací pozvánky skončily ve spamu. K první synchronizaci došlo až přibližně po 5 minutách. U každého listu a uživatele lze nastavit zasílání upozornění přes email na nově vytvořené úkoly.

3.3.6.2 Grafické uživatelské rozhraní

Rozhraní je ucelené a přehledné, dobře se ovládá (obrázek 3.9). Jeho účel je však velmi prostý, nenabízí tedy téměř žádné ovládací prvky. Aplikace nereaguje dobře na zmenšování šířky okna, jednotlivé prvky se překrývají.

3.3.6.3 Přehled výhod a nevýhod

Výhody

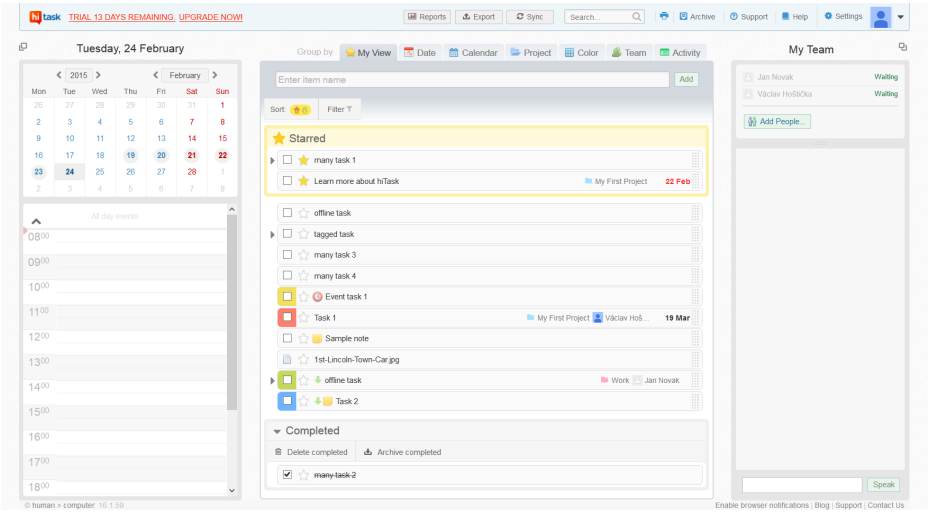
- Snadný způsob, jakým sdílet listy Google úkolů.

Nevýhody

- Plná závislost na Google úkolech.
- Neexistence nezpoplatněné verze.

3.3.7 hiTask

Nástroj *hiTask* podporuje je webové rozhraní a operační systémy Android a Apple iOS [9]. Služba je placená, odzkoušet ji lze na 15 denní trial verzi.



Obrázek 3.10: *hiTask* - přehled webového rozhraní.

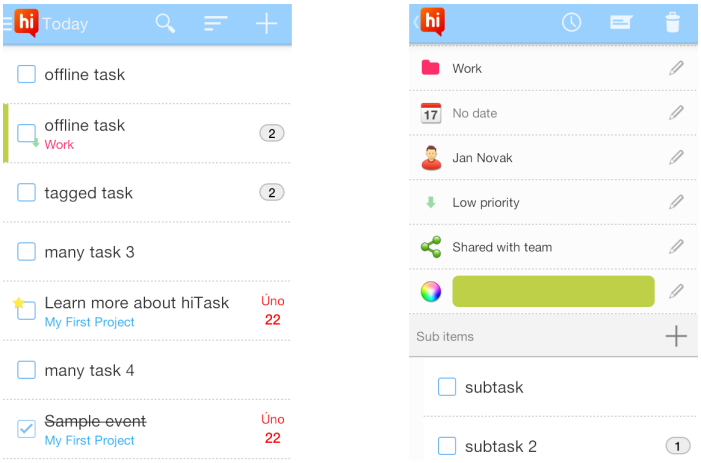
3.3.7.1 Funkcionalita

Přestože služba podporuje dle tabulky 3.1 všechny základní funkce, některé splňuje jen částečně. Poměrně ojedinělá je možnost libovolně zanořovat podúkoly. Těžkopádné je však například rozdělení do listů nebo práce s tagy. Ty působí jako pouze řetězce oddělené mezerami

vložené k úkolu, dále se nikde neobjevují a nelze podle nich vyhledávat ani filtrovat. V mobilní verzi tagy zcela chybí.

Výjimečnou funkcí je sdílení nejen celých listů, ale i jednotlivých úkolů mezi uživatele pozvané do projektu. Obsažen je export do jednoduchého CSV souboru i do formátu Excel. Aplikace dále obsahuje kalendářní pohled i přehled historie aktivit. Velmi zdařilé je generování výkazů obsahujících souhrn vývoje práce na projektech, mezi dostupné formáty patří Excel, PDF a HTML. *hiTask* také umožňuje synchronizaci s Google kalendářem a poskytuje vývojářské API.

Aplikace pro mobil funguje bez připojení k internetu, ale po opětovném připojení poměrně dlouho trvá, než dojde k synchronizaci. Obzvláště pokud byl pokyn v zařízení zadán ještě ve stavu offline. Ani k aktualizaci přehledu ve webovém prostředí nedochází ihned automaticky.



Obrázek 3.11: *hiTask* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a detail úkolu.

3.3.7.2 Grafické uživatelské rozhraní

Domovská stránka služby i aplikace samotná působí chaoticky, obsahuje velké množství prvků a aklimatizace uživatele v prostředí poměrně dlouho trvá. Jedním z možných důvodů je snaha zobrazit vše na jedné obrazovce (viz obrázek 3.10). Je také použito příliš mnoho barev. Odezva je poměrně pomalá. Výhodou pro některé uživatele může být permanentní zobrazení kalendáře a denního přehledu.

Design mobilní aplikace vykazuje větší propracovanost a ovládání je na první pohled přehlednější (viz obrázek 3.11). Některé funkce v mobilní verzi chybí.

3.3.7.3 Přehled výhod a nevýhod

Výhody

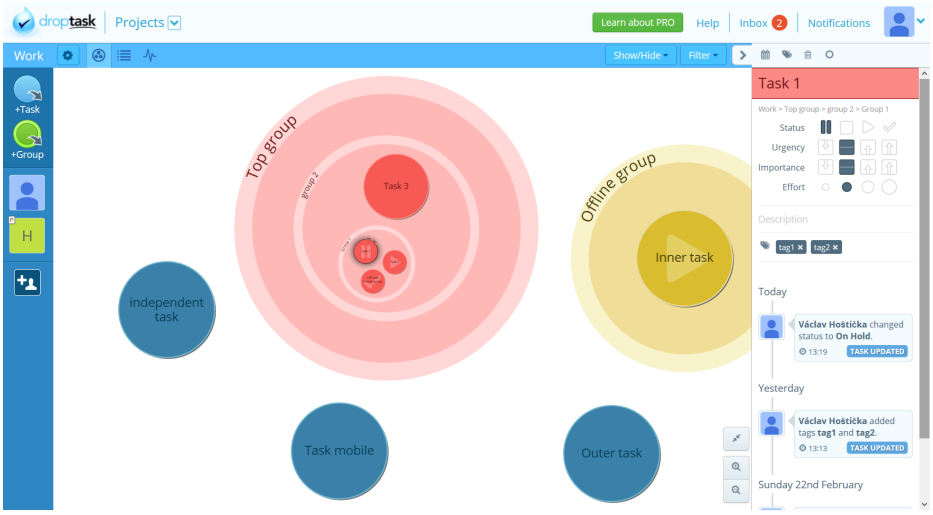
- Sdílení i pro jednotlivé úkoly.
- Export výkazů.

Nevýhody

- Nepohodlné ovládání základních funkcí programu.
- Synchronizace neprobíhá automaticky.
- Neexistence verze zdarma.

3.3.8 DropTask

Odlišným grafickým přístupem k ovládání se vyznačuje testovaný nástroj *DropTask* [6]. Podporovanými platformami jsou webový prohlížeč, Android a Apple iOS. Připravovaná je integrace do poštovního klienta Outlook. K dispozici je bezplatná i placená verze.



Obrázek 3.12: *DropTask* - přehled webového rozhraní. Vpravo je zobrazené menu pro editaci atributů úkolu.

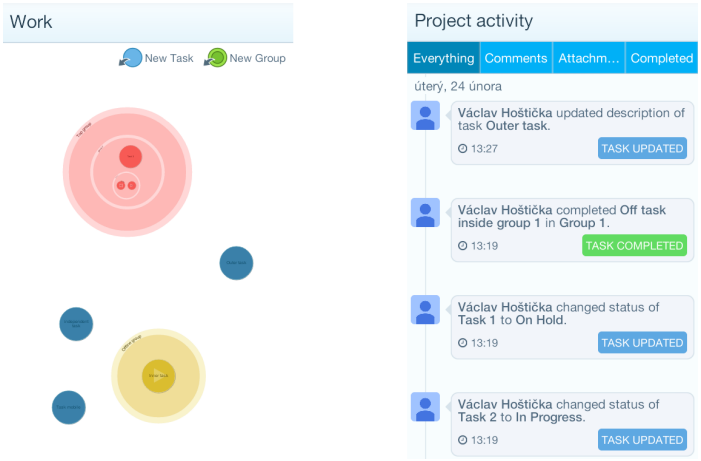
3.3.8.1 Funkcionalita

Ze základních funkcí (viz tabulka 3.1) aplikace nepodporuje pouze možnosti řazení, pravděpodobně kvůli netradičnímu pojetí GUI. Velmi dobře zpracované je filtrování, jednotlivé filtry se dají kombinovat. Jako jeden z mála nástrojů podporuje *DropTask* libovolné zanořování plnohodnotných úkolů.

Sdílení a spolupráce probíhají na úrovni jednotlivých projektů (listů). Mimo čtyři priority zavádí aplikace i výběr ze 4 kategorií potřebného úsilí a stejného množství označení pro naléhavost úkolu. Úkoly lze označovat nejen jako splněné, ale i jako probíhající či pozastavené. *Droptask* se dokáže synchronizovat s úkoly a kalendářem od Googlu a zálohovat na Dropbox.

Část mobilní aplikace zabývající se vytvářením a správou úkolů funguje i offline a následná synchronizace je automatická a velmi svižná. Historie aktivit, komentáře a přílohy fungují pouze s připojením k internetu.

Placená verze poskytuje celou řadu zadarmo nedostupných funkcí. Jedná se například o integraci s externími službami, souborové přílohy, vzájemné závislosti úkolů, šablony projektů, exklusivní podporu a další. Verze zdarma má také omezený počet projektů a kolaborantů.



Obrázek 3.13: *DropTask* - dvě vybrané obrazovky mobilní aplikace, grafické přidávání úkolů a přehled aktivit.

3.3.8.2 Grafické uživatelské rozhraní

DropTask umožňuje unikátní grafickou manipulaci při vytváření a sdružování úkolů. Úkoly i skupiny lze vytvářet, zanořovat, přeskupovat a oddělovat pomocí drag and drop operací. K dispozici je krom toho přehled úkolů v seznamu seřazený podle priorit. Design aplikace je přehledný a moderní (viz obrázek 3.12).

Mobilní aplikace, navzdory dobré funkčnosti, působí zastaralým a trochu nepřehledným dojmem (obrázek 3.13). Struktura je poměrně odlišná od webové verze. S grafickými prvky se ale pracuje bez problémů.

3.3.8.3 Přehled výhod a nevýhod

Výhody

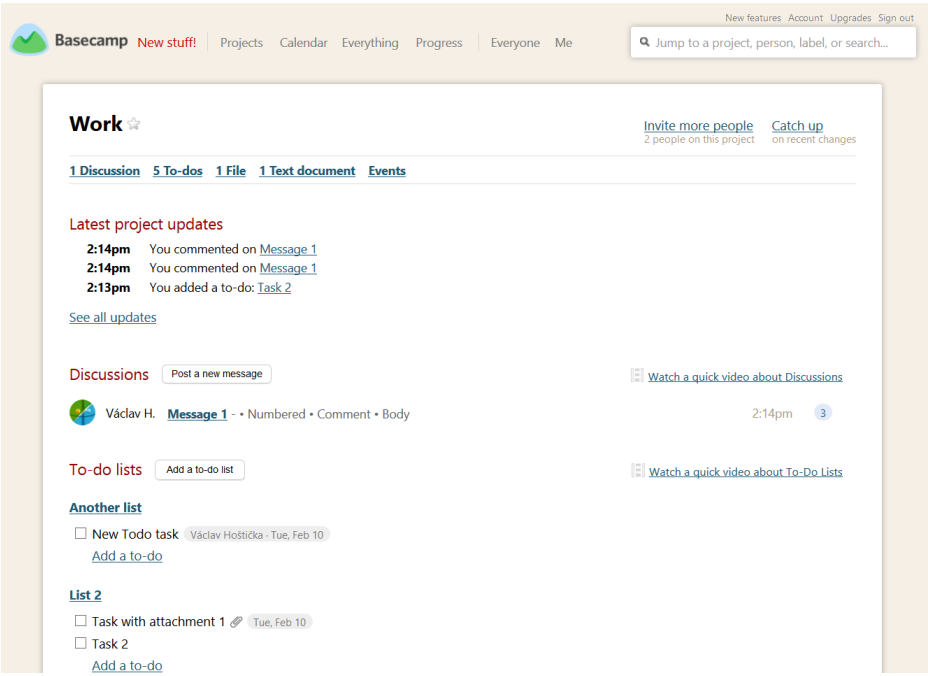
- Netradiční přístup k manipulaci s úkoly.
- Plnohodnotné víceúrovňové podúkoly.

Nevýhody

- Vzhled a ovládání mobilní verze.
- Chybí možnosti řazení v textovém přehledu úkolů.

3.3.9 Basecamp

Odlišným přístupem se zaměřením na projekty se vymezuje služba *Basecamp* [3]. Služba podporuje mobilní platformy Android, iOS a jako jediná z testovaných i Windows Phone. Registrace je vyžadována, možnost přihlášení přes Google účet nebo Facebook chybí. Zdarma je 60 denní trial verze.



Obrázek 3.14: *Basecamp* - přehled webového rozhraní.

3.3.9.1 Funkcionalita

Ze základních funkcí (přehled v tabulce 3.1) podporuje *Basecamp* listy, tagy, vyhledávání, komentáře a přílohy.

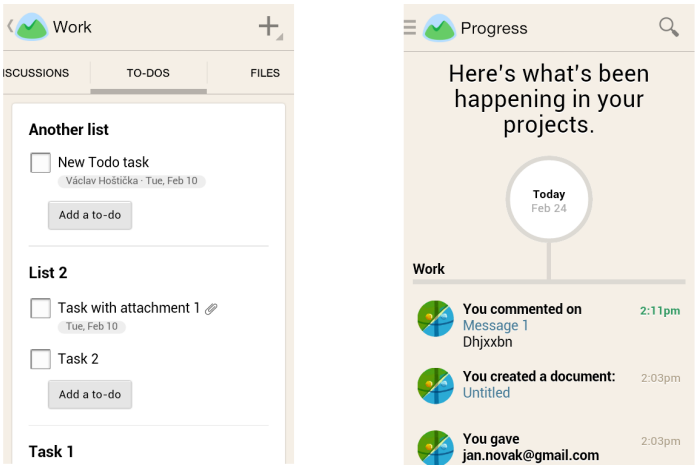
Basecamp se vyznačuje jinou strukturou než ostatní testované programy, zaměřuje se spíše na organizaci řízení projektů. Tyto projekty se v aplikaci skládají z tzv. „todo“ listů, diskusí, událostí a souborových příloh. Pro každý projekt lze přizvat další uživatele. Zajímavostí je spolupráce s uživatelem označeným jako „klient“, před kterým lze jednotlivé části projektu skrývat.

Mobilní aplikace není funkční bez připojení k internetu. Některé okrajové funkce, jako formátování při editaci textových příloh, nejsou dostupné.

3.3.9.2 Grafické uživatelské rozhraní

Vzhled webové stránky je tradiční, ale přehledný a dobře uspořádaný (viz obrázek 3.14). Po adaptaci na jiný přístup je ovládání snadné a intuitivní.

Vzhled mobilní aplikace působí modernějším dojmem a jednotlivé prvky jsou dobře uspořádané a propojené (obrázek 3.15). Rušivě působí pouze vzhled některých tlačítek a formulářových polí.



Obrázek 3.15: *Basecamp* - dvě vybrané obrazovky mobilní aplikace, přehled *todo* listů a nedávných aktivit.

3.3.9.3 Přehled výhod a nevýhod

Výhody

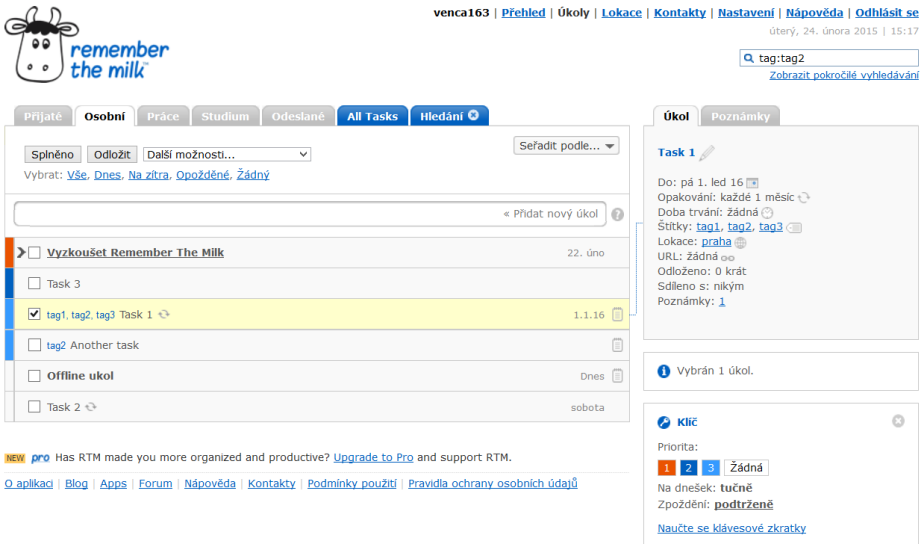
- Vhodný pro organizaci řízení projektů.
- Bohatější struktura, jeden projekt obsahuje více listů.

Nevýhody

- Chybí většina funkcí pro práci s úkoly.
- Chybí integrace s externími nástroji.
- Mobilní aplikace nefunguje offline.

3.3.10 Remember The Milk

Posledním testovaným nástrojem byl projekt *Remember The Milk* [13]. Mimo běžné platformy včetně Andoridu a Apple iOS je podporováno i BlackBerry, k dispozici je i integrace do celé řady dalších služeb (např. s Gmailem, Google kalendářem a dalšími) pomocí rozšíření do prohlížečů. Pro přidání rychlých poznámek lze využít bookmarklet. Existuje placená i bezplatná verze. Aplikace je dostupná ve více než 20 jazycích včetně češtiny.



Obrázek 3.16: *Remember The Milk* - přehled webového rozhraní.

3.3.10.1 Funkcionalita

Remember The Milk podporuje celou řadu ze základních funkcí, jejich používání je ale poměrně komplikované. Například pro výběr data neexistuje žádný klikací nástroj ve formě kalendáře, uživatel zadává hodnoty ručně. K dispozici je inteligentní rozpoznávač různých textových forem času (dnes, v pondělí), ale s leckterými vstupy má problém, nebo je interpretuje špatně a nápověda není v místě editace dostupná. Aplikace obsahuje i interní jazyk pro vyhledávání, vložený tag nelze vyhledávat prostým zadáním jeho názvu, nutná je textová předpona „tag:“.

Sdílet je možné celé listy mezi kontakty přidané do projektu. Kromě toho lze pro ostatní uživatele označit list jako veřejný, pak je list přístupný k nahlížení pro ostatní uživatele, kteří jej však nemohou měnit. K dispozici je výběr lokalit z mapy, které pak lze přidat k jednotlivým úkolům.

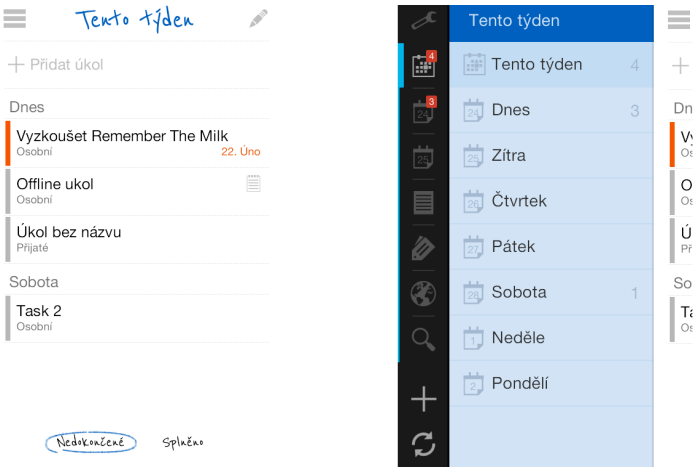
Mobilní verze obsahuje téměř plnohodnotnou funkcionalitu a lze ji používat i v offline režimu. Synchronizace je ve výchozím nastavení ruční, lze zapnout i automatický režim.

Placená verze obsahuje funkce upozornění, dřívější přístup k novinkám a přednostní podporu. Upgrade obsahuje extra funkce i pro mobilní platformy. Platící uživatelé mají také větší váhu při hlasování o směru dalšího vývoje.

3.3.10.2 Grafické uživatelské rozhraní

Webové rozhraní pro prohlížeče působí zastarale (viz obrázek 3.16). Ovládací prvky nejsou seskupené a někdy se těžko rozlišují od informativních částí stránky. Tam, kde aplikace neobsahuje klasické výběrové nástroje (např. kalendář, seznamy), je její použití poměrně složité a neintuitivní. Design není responzivní.

Mobilní aplikace je přehlednější, ale obsahuje celou řadu vad. Hlavní menu se dvěma úrovněmi je těžko pochopitelné a složité (viz obrázek 3.17). Některé výběrové nástroje (např. kalendář) už jsou přítomny, ale jejich zapnutí se musí explicitně povolit.



Obrázek 3.17: *Remember The Milk* - dvě vybrané obrazovky mobilní aplikace, přehled úkolů a menu.

3.3.10.3 Přehled výhod a nevýhod

Výhody

- Sdílení pro spolupráci i pouze pro nahlížení.
- Aplikace pro BlackBerry.
- Česká lokalizace.

Nevýhody

- Nepřehledný vzhled a složité ovládání.
- Slabá nabídka funkcí.

3.4 Shrnutí rešerše nástrojů

Z testovaných nástrojů pro organizaci a sdílení úkolů nabídly největší funkcionalitu *TickTick* (3.3.1) a *Todoist* (3.3.2). Druhý jmenovaný disponuje nejvíce funkcemi, i když velká část z nich je dostupná pouze v placené variantě. Obsahuje také jedno z nejpropracovanějších rozhraní, které je dostupné na 15 platformách.

Dobrou nabídku funkcí obsahují také nástroje *Producteev* (3.3.3) a *hiTask* (3.3.7), velkou nevýhodou prvního nástroje je nemožnost práce offline s mobilní aplikací, u obou potom ne příliš přehledný vzhled.

Svojí naprostou jednoduchostí vyniká *TaskShare* (3.3.4), přístupem k ovládání *DropTask* (3.3.8) a jinou filosofií *Basecamp* (3.3.9). V dostupné konkurenci je nabídkou funkcí nejslabší *Remember The Milk* (3.3.10), podporuje však jako jediný nástroj platformu BlackBerry a obsahuje českou lokalizaci.

	Tick	**Todo**	**Prod**	**TSh**	**GT**	**GST**	**hiT**	**DT**	**Base**	**RTM**
listy	ano	ano	ano[6]	ano	ano	ano	ano[6]	ano	ano	ano
tagy	ano	ano	ano	ne	ne	ne[7]	ano	ano	ano	ano
priority	ano	ano	ano	ne	ne	ano	ano	ne	ne	ano
řazení	ano	ano	ano	ne	ne	ano	ne	ano	ne	ano
filtry	ano[8]	ano	ano	ne	ne	ano	ano	ano	ne	ne
vyhledávání	ano	ano	ano	ne	ne	ano	ano	ne	ano	ano
upozornění	ano	ano	ano	ne	ne	ano	ano	ne	ne	ano
podúkoly	ano[9]	ano	ano[9]	ne	ne	ano	ano	ne	ne	ne
komentáře	ano	ano	ano	ne	ne	ano	ano	ne	ano	ne
přílohy	ano	ano	ano	ne	ne	ano	ano	ne	ano	ne

Tabulka 3.1: Přehled podpory základních funkcí v testovaných nástrojích.

[6] Nástroje neumožňují přímo kategorizaci úkolů do listů, ale obsahují tzv. projekty, ve kterých lze sdružovat skupiny úkolů.

[7] Nástroj nepodporuje faktickou práci s tagy, nepodporuje vyhledávání ani filtrování dle tagů.

[8] Nástroj umožňuje filtrování pouze podle tagů.

[9] Nástroje umožňují práce s podúkoly pouze jedné úrovně.

	Tick	**Todo**	**Prod**	**TSh**	**GT**	**GST**	**hiT**	**DT**	**Base**	**RTM**
sdílení listů	ano	ano	ano	ano	ne	ano	ano	ano	ano	ano
sdílení úkolů	ne	ne	ne	ne	ne	ne	ano	ne	ne	ne
offline applikace	ano	ano	ne	ne[10]	ne[10]	ne[10]	ano	ano	ne	ano
export/zálohy	ano	ano	ano	ne	ne	ne	ano	ne	ne	ne
externí služby	ano	ano	ne	ne	ne	ne	ano	ano	ano	ano

Tabulka 3.2: Přehled podpory vybraných pokročilých funkcí v testovaných nástrojích.

	Tick	**Todo**	**Prod**	**TSh**	**GT**	**GST**	**hiT**	**DT**	**Base**	**RTM**
prohlížeč	ano	ano	ano	ano	ano	ano	ano	ano	ano	ano
Android	ano	ano	ano	ne	ne	ne	ano	ano	ano	ano
iOS	ano	ano	ano	ne	ne	ne	ano	ano	ano	ano
Windows Phone	ne	ne	ne	ne	ne	ne	ne	ne	ano	ne
BlackBerry	ne	ne	ne	ne	ne	ne	ne	ne	ne	ano
Firefox doplněk	ne	ano	ne	ne	ne	ne	ne	ne	ne	ano[11]
Chrome doplněk	ano	ano	ne	ne	ne	ne	ne	ne	ne	ano[11]
Windows desktop	ne	ano	ne	ne	ne	ne	ne	ne	ne	ne
Mac	ne	ano	ano	ne	ne	ne	ne	ne	ne	ne

Tabulka 3.3: Podporované platformy u testovaných služeb.

[10] Nástroje mobilní aplikace vůbec nenabízejí.

[11] Nástroj nenabízí plnohodnotné rozhraní, ale doplněk slouží jako integrace do ostatních služeb.

Kapitola 4

Použité technologie

V této kapitole jsou popsány použité technologie se zaměřením na jejich základní aspekty, obecné výhody či nevýhody. Odůvodnění jejich výběru a popis konkrétního začlenění do projektu následuje v kapitole *Analýza a realizace* (5).

4.1 Saltarelle

Saltarelle [14] je překladač zdrojového kódu ze `C#` do JavaScriptu. Jedná se o open-source projekt přístupný pod permisivní Apache licencí verze 2.0. Projekt je starý necelé 4 roky a je stále aktivně vyvíjen, k dispozici je podpora v podobě fóra, uživatelská komunita je však poměrně malá. *Saltarelle* se nainstaluje přidáním balíčků do vývojového prostředí Microsoft Visual Studio (od verze 2010) pomocí správce balíčků *Nuget*.

Díky tomuto kompilátoru lze využívat výhody vysokoúrovňového jazyka při psaní webových a jiných aplikací, které běží na JavaScriptu. Při implementaci je možné používat plnohodnotný systém tříd včetně dědičnosti, generické typy, statickou typovou kontrolu, lambda výrazy a další. Velkým přínosem oproti psaní zdrojového kódu v JavaScriptu je pokročilé refaktorování. *Saltarelle* umí dokonce překládat asynchronní `C#` kód používající konstrukt `async/await`, a to vytvořením konečného automatu. Seznam podporovaných vlastností společně s ukázkovými příklady lze najít na www [14] v sekci *dokumentace*.

4.1.1 Použití JavaScriptových knihoven

Saltarelle přináší i drobné nevýhody, mimo nutnosti počátečního nastavení projektu je to zejména oblast používání externích JavaScriptových knihoven. Pro každou knihovnu je třeba napsat *stub* obsahující metadata o jejím použití. To je třída obsahující atributy a metody takové, jaké má kód v JavaScriptu, s odpovídajícími typy. Neobsahuje však žádný výkonný kód, těla metod jsou prázdné. Externí knihovna je pak připojena do výsledného projektu[1].

[1] Např. do webové stránky může být nalinkována standardně pomocí HTML tagu `<script>`, do Node.js aplikace může být přidána pomocí funkce pro načítání modulů `require`.

Povinnou knihovnou pro běh přeloženého kódu je s překladačem dodávaný soubor *mscorlib.js*. Od autora a komunity jsou k dispozici knihovny pro práci s DOM[2], Node.js, jQuery a řada dalších. Existuje i možnost pracovat s externími knihovnami bez stubů a to s využitím klíčového slova *dynamic*, tímto přístupem se ale ztrácí výhody implementace v C#. S využitím *stubů* programátor vidí dostupné funkce knihoven, očekávané atributy a jejich typy, či typy návratových hodnot.

Pro správné generování výsledného JavaScriptu lze modifikovat chování překladače pomocí atributů, které se aplikují na celé třídy nebo i pouze na jejich jednotlivé metody[3]. Běžně používanými atributy jsou *Imported* pro označení kódu, který bude využívat volání externí knihovny (žádný výkonný kód nebude vygenerován), *ScriptNamespace* pro nastavení jmenného prostoru ve výsledném kódu, *ScriptName* pro nastavení jména třídy nebo metody (implicitně se jméno v přeloženém kódu shoduje se jménem metody ve *stubu*, pouze počáteční velké písmeno je převedeno na malé) nebo například *ModuleName* pro správné načítání modulů pro Node.js (4.4). Pro jinak neřešitelné situace lze použít také *InlineCode* a zapsat v požadovaném formátu kód JavaScriptu přímo.

Listing 4.1: Ukázka aplikace atributů na třídu a metodu

```
[Imported]
[IgnoreNamespace]
[ModuleName("socket.io")]
[ScriptName("")]
public static class SocketIo
// ...

[InlineCode("{this}.on('connection', {handler})",
   GeneratedMethodName = "on")]
void OnConnection(Action<ISocketIoServerSocket> handler);
```

4.2 ExtBrain Framework

Extbrain Framework je knihovna napsaná v jazyce C# určená pro manipulaci s daty a jejich prezentaci. Součástí jádra frameworku je i podpora pro psaní značkovacího jazyka HTML a XML, k dispozici je téměř kompletní sada elementů, atributů a jejich hodnot a podpůrných tříd pro práci s elementy. Framework bohužel neposkytuje dokumentaci a zdrojový kód neobsahuje komentáře.

Framework je připraven pro použití kompilátorem *Saltarelle* a obsahuje celou řadu stubů pro začlenění externích knihoven do projektu. Patří mezi ně například moduly do Node.js (4.4) jako jsou *Connect*, *Sqlite3* (4.4.3) nebo *Socket.IO* (4.4.2). Součástí rozšířené části frameworku je i *AppController* zajišťující základní logiku webové aplikace (vytváření odkazů, přechody mezi stránkami, navěšování událostí na načtení stránky apod.). Tato technologie

[2] **Document Object Model**, tj. objektová reprezentace dokumentu ve stromové struktuře standardizovaná organizací W3C poskytující API pro práci s obsahem HTML či XML dokumentů a umožňující jeho modifikaci.

[3] Jednotlivé atributy mají nastaveno, na jaké prvky jazyka je lze aplikovat. Kromě tříd a metod jsou to například rozhraní, výčtové typy, struktury, fieldy, kontruktory, parametry a další.

tedy představuje způsob, jakým psát dynamické webové aplikace a celý kód, včetně HTML šablon, vytvářet a udržovat v C#.

4.3 Cordova

Apache Cordova [5] je platforma pro vývoj mobilních aplikací za použití standardních webových technologií, HTML a CSS pro prezentační vrstvu a JavaScriptu pro logiku aplikace. Výsledná aplikace běží ve webovém prohlížeči operačního systému, který je zapouzdřen do nativní aplikace. Jedinou rozdílnou částí kódu vzniklé aplikace je tedy kontejner, který v sobě obsahuje *WebView*[4] pokrývající celou část obrazovky. Zobrazený výsledek se však může lišit v závislosti na vykreslování jednotlivých prohlížečů.

Framework se používá pomocí nástroje integrovaného do příkazové řádky, pro build pro jednotlivé cílové platformy je třeba mít nainstalované jejich SDK. Aktuálně podporovanými operačními systémy jsou především Android a iOS, u kterých je zajištěn přístup k většině API zařízení (např. souborový systém, fotoaparát, akcelerometr a další), velmi dobrou podporu nabízí *Cordova* i pro Windows Phone, Windows 8, BlackBerry OS a Ubuntu Touch, částečnou podporu pak pro Firefox OS, Badu, Symbian a Tizen[5]. Projekt je také známý pod označením *PhoneGap*[6].

Cordova poskytuje JavaScriptové API, pomocí kterého lze využívat platformové API zařízení. Základní API je poskytováno přímo od Apache Cordova, doplňkové od externích vývojářů. Oboje se přidává do projektu jako plugin pomocí nástroje pro příkazovou řádku. Následující příkaz, demonstrující přidání pluginu pro přístup k fotoaparátu přístroje, se zadává z root adresáře cordova projektu.

```
cordova plugin add org.apache.cordova.camera
```

U některých pluginů je potřeba pro některé platformy ještě manuálně upravit konfigurační soubory, ve kterých se definuje začlenění zásuvného modulu do mobilní aplikace (na platformě Android se jedná o soubory *app/AndroidManifest.xml* a *config.xml* v kořenovém adresáři aplikace). Výjimečně je nutné i ručně připojit JavaScriptovou knihovnu. Následuje stručné představení hlavních použitých pluginů.

4.3.1 File API

File API je poměrně rozsáhlé rozhraní k souborovému systému zařízení, které je postaveno na W3C File API[7]. K úložišti se přistupuje přes metodu *window.requestFileSystem()*, druhou nestandardní metodou oproti specifikaci je *window.resolveLocalFileSystemURI()*, která

[4] *WebView* je komponenta systému, která se stará o zobrazování webových stránek. Název je napříč platformami velmi podobný, např. *WebView* pro Android a Windows Phone, nebo *UIWebView* pro iOS.

[5] Aktuální přehled dostupnosti API pro jednotlivé platformy je dostupný na https://cordova.apache.org/docs/en/4.0.0/guide_support_index.md.html.

[6] **PhoneGap** je původní označení projektu do doby, než byl začleněn pod Apache Software Foundation. Dnes lze software používat jako nástroj do příkazové řádky pod obojím označením - *cordova* i *phonegap* - přičemž nástroj *phonegap* zapouzdřuje *cordovu* a oba obsahují téměř totožnou funkcionalitu, *phonegap* obsahuje navíc možnosti pro vzdálený build aplikací.

[7] **W3C File API** odkaz: http://www.w3.org/TR/FileAPI/.

dokáže pracovat s URI ve formátu lokálního zařízení. API dále obsahuje standardní objekty jako File, FileEntry, FileReader, FileWriter a další. Soubory jsou načítány a ukládány asynchronně.

4.3.2 SQLite plugin

Cílem *SQLite pluginu* je poskytovat rozhraní shodné s HTML5/Web SQL API[8] pro práci s SQLite databází. Jedinou odchylkou v tomto ohledu je vlastní metoda inicializace databáze:

Listing 4.2: Ukázka inicializace databáze.

```
// JavaScript
window.sqlitePlugin.openDatabase()
```

Ostatní metody jsou v souladu se specifikací. Projekt má řadu problémů a omezení na různých operačních systémech, nespolehlivá práce je napříč všemi platformami s datovým typem *BLOB*[9]. Doplněk je stále aktivně vyvíjen.

4.3.3 Camera API

Camera API umožňuje vývojáři přístup k fotoaparátu zařízení. Zachycenou fotografii lze získat přímo jako binární data v textové podobě (zakódované pomocí base64[10]), nebo jako adresu do lokálního dočasného souborového úložiště. Lze nastavit kvalitu snímaného obrázku i výsledný formát. Pomocí tohoto API lze přistupovat i k obrázkům uloženým v galerii zařízení.

4.3.4 BarcodeScanner plugin

Pro jednoduché načtení URL adresy z čárového kódu lze využít plugin *BarcodeScanner*. Plugin podporuje celou řadu typů kódů, mezi nejznámější patří QR kód (Quick Response Code), UPC (Universal Product Code), EAN (European Article Number), Data Matrix nebo například Code 128[11].

4.3.5 WebIntent plugin

WebIntent plugin slouží pro práci s *Android Intenty*. Posílání Intentů je způsob, jakým lze v systému Android zajistit mezi-procesní komunikaci. Intent je objekt přenášející data, který

[8] W3C ovšem přestalo pracovat na **Web SQL API** standardizaci, projekt ustoupil aktuálně vyvíjenému standardu *Indexed Database API*, více na http://www.w3.org/TR/IndexedDB/.

[9] **BLOB** je databázový datový typ. V SQLite, podobně jako v ostatních databázích, lze do sloupce typu BLOB ukládat libovolná data, která budou uložena přesně ve formátu, ve kterém byla vložena. Využívá se zejména pro ukládání binárních dat. Seznam známých problémů pluginu je dostupný na https://github.com/litehelpers/Cordova-sqlite-storage#known-issues.

[10] **Base64** je kódování binárních dat na tisknutelné znaky, které jsou poté přenositelné kanály, které vyžadují standardní text.

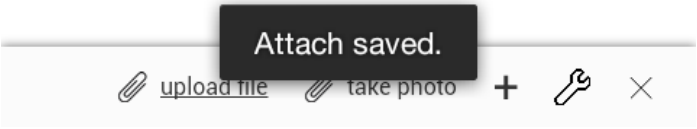
[11] Úplný seznam pro jednotlivé platformy lze prohlédnout na https://github.com/wildabeast/BarcodeScanner#android.

má specifikovaný objekt *action*, případně *extras*. Podle těchto objektů se registrují posluchači, kteří jsou zařazeni do seznamu aplikací, které jsou schopny na konkrétní Intenty reagovat[12].

Plugin umožňuje jednak vytvářet aktivity, jednak registrovat posluchače pro jednotlivé akce. Původní WebIntent plugin obsahuje bug při práci s klíčem *EXTRA_STREAM* (stav k 1.3.2015), který se využívá pro sdílení souborů. Pro sdílení souborů mezi aplikacemi s využitím tohoto klíče je tedy třeba nainstalovat opravenou verzi[13].

4.3.6 Toast plugin

Jedná se o malý plugin pro zobrazování krátkých upozornění, tzv. *Toast messages*, pro uživatele. Zobrazování lze pozicovat na horní, prostřední či dolní část obrazovky a dále lze nastavit dobu zobrazení na dlouhou či krátkou.



Obrázek 4.1: Ukázka detailu *Toast message*.

4.4 Node.js

Node.js je JavaScriptová událostmi řízená platforma pro vytváření rychlých a dobře škálovatelných síťových aplikací [11]. Běhovým prostředím je engine V8 od Googlu[14]. Běh programu je řízen asynchronními vstupně výstupními operacemi, každá taková operace používá *callback*[15], příkazy se vykonávají paralelně. S touto architekturou zvládá *Node.js* obsluhovat velké množství požadavků, i když běží na jednom vlákně, řízení dostane v jednom čase vždy jen jeden callback, není tedy třeba řešit problémy souběhu. Projekt vznikl v roce 2009 a od té doby zaznamenal strmý vzestup v popularitě[16].

[12] Uživatel např. vybere v zařízení obrázek a chce ho sdílet do aplikace. V seznamu aplikací pro sdílení Android nabízí všechny takové, které se v *Android Manifestu* zaregistrovaly pro konkrétní typ souboru. Uživatel jednu vybere a tato aplikace obdrží Intent s přiloženými daty, v tomto případě URL obrázku. Je na aplikaci, jak s těmito daty naloží. Více na http://developer.android.com/guide/components/intents-filters.html.

[13] Fork původního repozitáře na adrese https://github.com/florentvaldelievre/virtualartifacts-webintent.

[14] Interpretr JavaScriptu **V8** je považován za jeden z nejrychlejších v současnosti, i když vyhledat prokazující nezávislé testy je obtížné. Základní designové principy, které podporují rychlost běhu, jsou sepsané na adrese https://developers.google.com/v8/design.

[15] **Callback** je funkce, která se předává při spouštění nějaké asynchronní operace. Program se vykonává dál a ve chvíli, kdy obslužná rutina dokončí asynchronní operaci, zavolá předaný callback. Callbacky mohou přijímat parametry, se kterými jsou volány.

[16] Alternativní platformou pro JavaScriptové aplikace na straně serveru je například projekt *io.js* (https://iojs.org/en/index.html), který vznikl jako fork právě z Node.js. a jehož filosofií je přinášet nové verze v pravidelných cyklech a reagovat na změny enginu V8.

Funkcionalita platformy se rozšiřuje pomocí modulů. Samotný framework obsahuje řadu základních modulů, např. *File System* pro přístup k souborovému systému nebo *HTTP* pro práci s tímto protokolem. Mnoho dalších modulů je vyvíjeno externími vývojáři, dostupné jsou přes balíčkovací systém *npm*[17]. V obou případech se pak balíček připojuje do aplikace pomocí metody `require()`. Níže jsou stručně představeny některé z použitých modulů.

Listing 4.3: Ukázka připojení modulu Socket.IO

```
// JavaScript
var socketio = require('socket.io');
```

4.4.1 Express

Express rozšiřuje možnosti webového serveru a usnadňuje tak práci s přijímáním a vyřizováním HTTP požadavků, obsahuje například jednoduchou konstrukci jak navěšovat posluchače pro základní HTTP metody GET, POST, PUT a DELETE a jak poslouchat na jednotlivých, i parametrizovaných, URL adresách. Dále rozšiřuje standardní Node.js objekty pro HTTP požadavek/odpověď (request/response) o další funkcionalitu. Aktuální verze 4.x je oproti předchozím výrazně odlehčená, filosofií vývojářů je dodat malou knihovnu s možností přidání funkcionality na vyžádání pomocí dalších standardních balíčků.

4.4.2 Socket.IO

Socket.IO je knihovna pro obousměrnou komunikace mezi serverem a klientem v reálném čase. Pro přenos jsou využívány různé technologie[18], v případě přítomnosti v běhovém prostředí se používají webové sockety[19], uživatel však vždy využívá jednotné Socket.IO rozhraní. Klientem může být prohlížeč, ale také například jiný Node.js server. K dispozici jsou 2 balíčky pro serverové platformy (včetně Node.js), *socket.io* a *socket.io-client*. Pro prohlížeč lze klientský kód stáhnout manuálně, Socket.IO server taktéž vrací klientský kód na URL adrese *"http://«adresa-serveru»/socket.io/socket.io.js"*.

4.4.3 Sqlite3

Sqlite3 poskytuje řešení pro práci s databází SQLite. Tento modul není jediný, který umožňuje práci s touto databází, mezi jeho výhody patří stálý vývoj, dobrý výkon nebo například možnost spustit a testovat databázi přímo v paměti.

4.5 SQLite databáze

SQLite [15] je relační databázový systém. Hlavní výhodou oproti klasickým SQL databázím jako MySQL nebo PostgreSQL je, že *SQLite* existuje ve formě malé C knihovny a vloží se přímo do systému, který ji využívá. Databáze tedy nepotřebuje žádný server, na kterém by

[17] Odkaz na balíčkovací systém *npm* https://www.npmjs.com/.
[18] Mezi takové patří *AJAX long-polling*, *Flash socket* nebo např. *JSONP polling*.
[19] **WebSockets** odkaz http://en.wikipedia.org/wiki/WebSocket.

běžela, po běhovém prostředí vyžaduje pouze základní sadu funkcí C[20]. Tím odpadá celý proces instalace a nastavení, databáze nepotřebuje žádnou konfiguraci.

Díky tomu je *SQLite* jednou z nejrozšířenějších SQL databází, která je vhodná především pro vestavěné systémy, mobilní a jiná přenosná zařízení, technologie může být použita i pro webové stránky nebo pro analýzu dat. Využívají ji velké společnosti pro své produkty, mezi nejznámější patří Firefox od Mozilly, Safari a OS X/iOS od Applu nebo například Chrome a Android od společnosti Google. Pracovníci Googlu se dokonce podíleli na vývoji fulltextového vyhledávání databáze.

SQLite podporuje serializovatelné transakce s vlastnostmi ACID. Oproti jiným SQL databázím nabízí méně typů, a to[21]:

- **NULL**: hodnota NULL
- **INTEGER**: celé číslo se znaménkem, uložené v 1, 2, 3, 4, 6 nebo 8 bajtech podle velikosti
- **REAL**: reálné číslo s pohyblivou desetinnou čárkou, uloženo v souladu s IEEE standardem
- **TEXT**: textový řetězec, uložené v kódování UTF-8 nebo UTF-16
- **BLOB**: vstup uložen přesně ve formě, ve které byl vložen

4.6 jQuery

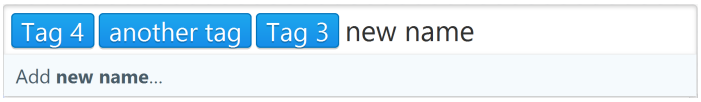
jQuery je velmi rozšířená JavaScriptová knihovna, oblíbená pro svou bohatou nabídku funkcí, které usnadňují manipulaci s elementy, obsluhu událostí, AJAX a přináší mnoho dalšího. Popularitu si získal zejména kvůli své rychlosti, relativně malé velikosti a především spolehlivou funkcionalitou napříč prohlížeči.

4.7 Selectize

Selectize je JavaScriptová knihovna, která se používá k vytváření ovládacích prvků sloužících pro výběr z položek, případně i pro vytvoření nové položky. Vhodná je tedy například pro tagování elementů. Knihovna ke svému běhu vyžaduje jQuery.

[20] SQLite při spuštění v paměti vyžaduje pouze C funkce `memset()`, `memcpy()`, `memcmp()`, `strcmp()`, `malloc()`, `free()` a `realloc()`. Pro přístup k souborovému systému používá rozšiřitelnou vrstvu VFS (Virtual File System). VFS moduly pro Windows a Unix systémy jsou součástí technologie. Zdroj https://www.sqlite.org/selfcontained.html.

[21] Přehled typů převzat a přeložen z https://www.sqlite.org/datatype3.html.



Obrázek 4.2: Ukázka detailu použití knihovny *Selectize*.

4.8 LESS

Jazyk LESS rozšiřuje možnosti jazyka CSS, do kterého se kompiluje [10]. Slouží tedy také k tvorbě stylů, přidává ale pokročilejší konstrukce jako používání proměnných, funkcí, vytváření skupin pomocí zanořování a další. Odpovídá tedy na dlouholetou snahu uživatelů přidat některé tyto vlastnosti přímo do CSS[22]. Označení LESS se používá jednak pro samotný jazyk, jednak pro jeho překladač do kaskádových stylů. Alternativou je například Sass[23].

4.9 Bootstrap

Bootstrap je robustní knihovna pro vývoj webových aplikací [4]. V dnešní podobě poskytuje množství CSS stylů, připravených HTML komponent a dokonce i jQuery pluginů. Styly lze stáhnout a integrovat do projektu ve formě čistého CSS, LESS nebo Sass. Jedná se velmi aktivně vyvíjený open-source projekt hostovaný na portálu GitHub.

[22] O tom, proč by mohlo být začlenění těchto konstrukcí do CSS škodlivé, pojednává např. tento článek http://www.w3.org/People/Bos/CSS-variables. V současném stavu jsou si tvůrci CSS vědomi existence dynamických jazyků pro tvorbu stylů jako LESS nebo Sass a nejsou zprávy, že by se v blízké době plánovalo rozšíření přímo CSS. Společnost Mozilla sice přišla s experimentální implementací proměnných (https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_variables), ale Firefox je momentálně jediným prohlížečem, který je podporuje.

[23] **Sass** je dalším CSS preprocesorem, podobně jako LESS. Domovská stránka se nachází na webové adrese http://sass-lang.com/.

Kapitola 5

Analýza a realizace

V této kapitole jsou nejprve shrnuty použité programovací jazyky, následuje popis architektury systému, u kterého je ukázáno konkrétní využití technologií představených v předešlé kapitole. Dále se v kapitole nachází popis schématu databáze a konečně popis funkcionality systému společně s uživatelským rozhraním.

5.1 Použité programovací jazyky

Výsledný program je vykonáván v jazyce JavaScript. Ten zajišťuje jak funkčnost klienta ve webovém prohlížeči, tak i serverovou část (5.2.1). JavaScript je použit i jako zdrojový jazyk pro mobilního klienta, do Android aplikace jsou JavaScriptové zdrojové kódy převedeny pomocí frameworku *Cordova* (4.3).

Systém byl však implementován v jazyce `C#` a do JavaScriptu byl překládán pomocí kompilátoru *Saltarelle* (4.1). Toto rozhodnutí přináší řadu výhod, ty zásadní jsou shrnuty v následujícím seznamu.

- `C#` je vysokoúrovňový programovací jazyk s plnohodnotným systémem tříd, dědičností a rozhraními. JavaScript toto může pouze simulovat. Použití `C#` v tomto případě přispívá k lepšímu a čistšímu návrhu.
- `C#` je staticky typovaný jazyk, překladač provádí statickou typovou kontrolu. JavaScript je dynamicky typovaný a typová kontrola probíhá až za běhu. Díky statické typové kontrole je velká řada chyb odhalena snáze a opravena dříve.
- `C#` obsahuje jmenné prostory. Ty představují výhodu zejména při integrování většího množství externích komponent.
- Možnost využití ExtBrain Frameworku (4.2), jehož zdrojový kód je napsán v `C#`.
- Visual Studio umožňuje použití nástroje *IntelliSense*, který se stará o doplňování kódu při implementaci. *IntelliSense* ale i různá IDE určená přímo pro JavaScript[1] do jisté

[1] Mezi IDE zaměřující se na JavaScript patří například *Sublime Text* (`http://www.sublimetext.com/`) nebo JetBrains WebStorm (`https://www.jetbrains.com/editors/javascript_editor.jsp`).

míry také zvládají doplňování kódu JavaScriptu, ale svými schopnosti zatím nedosahují tak dobrých výsledků jako doplňování pro C#. Napovídání dostupných atributů a metod u C# funguje stoprocentně.

- Od verze 5.0 obsahuje C# podporu pro asynchronní programování pomocí klíčových slov *async/await*. Jejich využitím lze předejít mnohonásobnému zanořování callbacků[2], výsledek asynchronní operace je získán standardním voláním funkce. Následují 2 ukázky zdrojového kódu.

Listing 5.1: Ukázka C# kódu pro zápis do souboru.

```
// Za pouziti callbacku.
static void CreateFile(string dirName, string fileName,
    string content)
{
    FileApi.RequestFileSystem(0, fileSystem =>
    {
        fileSystem.Root.GetFile(dirName + "/" + fileName,
            fileEntry =>
        {
            fileEntry.CreateWriter(fileWriter =>
            {
                fileWriter.Write(content);
            });
        });
    });
}

// Za pouziti async/await.
static async void CreateFile(string dirName, string
    fileName, string content)
{
    var fileSystem = await FileApi.RequestFileSystem(0);
    var fileEntry = await fileSystem.Root.GetFile(dirName
        + "/" + fileName);
    var fileWriter = await fileEntry.CreateWriter();

    fileWriter.Write(content);
}
```

Pouze jedna část aplikace, mimo kaskádových stylů pro jejichž vývoj byl použit LESS (4.8), nebyla napsána v jazyce C#. Jedná se o extrakční nástroj (5.2.5), který byl implementovaný přímo v čistém JavaScriptu.

[2]Existují i další náhrady za callbacky při asynchronním programování. Např. *jQuery promises* (http://www.htmlgoodies.com/beyond/javascript/making-promises-with-jquery-deferred.html) použité i v rámci této práce. Další konstrukce se nacházejí i v různých jazycích, které se do JavaScriptu kompilují. Mezi ně patří např. *LiveScript* (http://livescript.net/) nebo *TypeScript* (http://www.typescriptlang.org/).

5.2 Architektura

Sekce *Architektura* se zaměřuje na popis hlavních komponent systému a jejich vzájemné provázanosti. V této části je také popsáno, jakým způsobem byly jednotlivé technologie z předešlé kapitoly použity ve vyvíjeném programu.

5.2.1 Node.js server

Serverová část aplikace byla postavena na technologii Node.js (4.4). Architektura systému umožňuje spouštět více rovnocenných serverů a synchronizovat data mezi libovolnými dvěma z nich. Více o spouštění serverů v sekci C, o synchronizaci v části 5.4.3.2.

Po spuštění server nejprve inicializuje modul Express (4.4.1) a zaregistruje příslušné middlewary pro zpracovávání požadavků. Dále vytvoří databázi, pokud ještě neexistuje, případně vloží testovací data. Poté začne server poslouchat na portu, který mu byl předán jako argument. Nakonec se zaregistruje posluchač čekající na příchozí *Socket.IO* spojení.

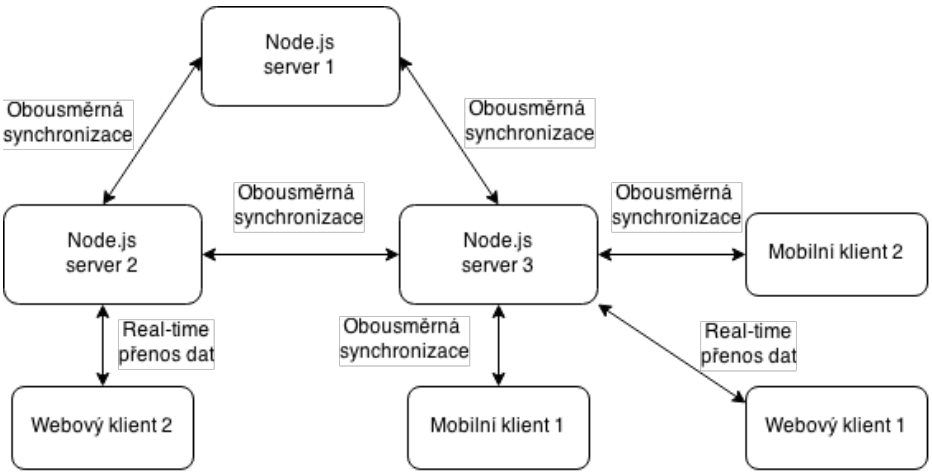
Server obsluhuje příchozí *Socket.IO* spojení, pro každého klienta ověří jeho totožnost a vytvoří novou instanci obslužné třídy pro přenos dat. Té je předána i totožnost uživatele, aby server mohl rozlišovat, jaká data může uživateli vracet a jak má manipulovat s daty přijatými. Identita je použita i v případě pozdější synchronizace s dalším serverem.

Server využívá následující moduly, hlavní z nich jsou podrobněji představeny v sekci Node.js (4.4). Seznam slouží jako stručný přehled konkrétního použití modulů, tučně vysázená jména odpovídají názvům modulů v balíčkovacím systému *npm*. Moduly s označením „externí“ nejsou součástí Node.js a musely být do systému nainstalovány.

- **http** - vytvoření serveru a poslouchání na příslušném portu
- **fs** - přístup k souborovému systému, využití pouze při vývoji pro automatickou aktualizaci stylů
- **buffer** - servírování binárních souborů při stahování příloh
- **os** - přístup k některým komponentám operačního systému, konkrétně pro získání URL, na kterém běží server[3]
- **express** (externí) - obsluha HTTP požadavků
- **body-parser** (externí) - zpracování dat z HTTP požadavků
- **socket.io** (externí) - obsluha *Socket.IO* spojení
- **sqlite3** (externí) - práce s databází SQLite
- **jsonwebtoken** (externí) - vytváření a podepisování tokenů
- **socketio-jwt** (externí) - ověřování tokenů při vytváření socketového spojení
- **bcrypt-nodejs** (externí) - implementace hashovacího algoritmu bcrypt pro ukládání a ověřování hesel

[3] Toto URL je pak uživateli zobrazeno textově i pomocí QR kódu. Více v sekci C.

- **cors** (externí) - zpřístupnění URL pro volání skripty jiného původu, pro přijímání dat z extraktoru
- **node-uuid** (externí) - generování Uuid[4], tj. unikátních identifikátorů pro entity v aplikaci



Obrázek 5.1: Ukázka možné komunikace jednotlivých částí systému.

5.2.1.1 Server API

Server kromě obsluhy Socket.IO spojení vyřizuje i HTTP požadavky, jmenovitě GET požadavek na adrese `/downloadAttach` pro stahování či zobrazování souborů a POST požadavky na adresách `/login` pro přihlášení uživatele, `/register` pro registraci uživatele a `/saveTask` pro uložení nového úkolu z extraktoru. Poslední jmenované URL tudíž vyžaduje platný autentizační token. Jako jediná adresa má také povolený přístup pro skripty z cizích domén (více v sekci 5.2.5 o extraktoru, heslo CORS).

Na adrese *http://localhost:«cislo portu»/* je přístupný samotný webový klient, servírována jsou i další statická data, zejména soubory potřebné pro běh klienta, např. CSS soubory nebo různé JavaScriptové knihovny.

5.2.2 Databáze

Systém běží na databázi SQLite (4.5), která je použita na serveru i jako lokální úložiště mobilního klienta. Serverová databáze využívá *sqlite3* Node.js modul (4.4.3), mobilní databáze využívá SQLite plugin (4.3.2), tyto nemají shodné API a vyžadují proto vlastní obslužný kód. Obě varianty používají modul *FTS3* pro vytváření tabulek s podporou fulltextového vyhledávání.

[4]**Uuid**, volně přeloženo jako **Univerzální unikátní identifikátor**, je standard, podle kterého lze generovat 128 bitové hodnoty sloužící jako identifikátory entit. Používá se k vytváření ID s velmi nízkou pravděpodobností kolizí. V aplikaci je použita verze 4. Více na http://en.wikipedia.org/wiki/Universally_unique_identifier.

5.2.3 Webový klient

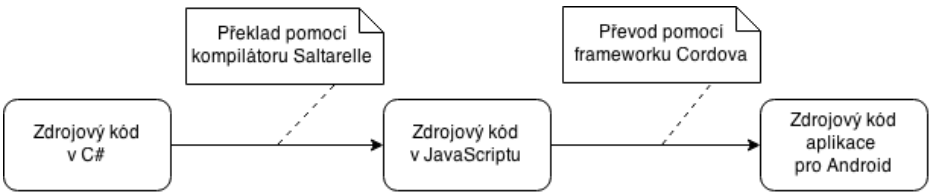
Klient pro prohlížeč[5] je postavený pomocí standardních webových technologií (HTML, CSS a JavaScript). HTML a JavaScriptová část je implementována v `C#` za použití mnoha knihoven. Primárně je využíván ExtBrain Framework (4.2) pro elementární logiku aplikace a vytváření jednotlivých stránek klienta. Práci s elementy ulehčuje jQuery (4.6), práci se styly LESS (4.8), reponzivní design a lepší vzhled aplikace podporuje použití knihovny Bootstrap (4.9).

Webový klient neobsahuje žádnou databázi ani jiné lokální úložiště. Používán může být bez připojení internetu, ke svému běhu potřebuje mít spuštěný pouze lokální Node.js server. Komunikace s tímto serverem probíhá s využitím knihovny Socket.IO (4.4.2).

5.2.4 Mobilní klient

Mobilní klient sdílí velkou část kódu s klientem do prohlížeče. Do aplikace pro Android byl kód transformován pomocí nástroje Cordova. Mezi největší změny patří přizpůsobení vzhledu pro malé obrazovky a použití pluginů pro přístup ke komponentám zařízení. Z popsaných pluginů pro Cordovu (4.3) byly použity všechny.

Aplikace je koncipována pro použití bez připojení k serveru (v offline režimu) a proto obsahuje vlastní lokální SQLite (4.5) databázi[6]. Toto je zásadní změna oproti webovému klientu. Po připojení komunikuje se serverem opět pomocí Socket.IO. Po připojení k libovolnému serveru dochází k synchronizaci dat.



Obrázek 5.2: Diagram procesu implementace mobilního klienta.

5.2.5 Extraktor

Extrakční nástroj, který slouží pro získávání částí nebo celých HTML stránek a jejich ukládání do vyvíjené aplikace, je napsán jako jediná komponenta přímo v JavaScriptu. Použit byl navíc pouze čistý JavaScript, bez jakýchkoliv externích knihoven, aby nevznikaly zbytečné závislosti a aby extraktor nemusel před použitím čekat na načtení závislých knihoven.

Extraktor je implementovaný jako bookmarklet. Bookmarklety jsou JavaScriptové programy obsažené v záložkách prohlížeče. Oproti klasickým doplňkům do prohlížeče, které jsou

[5] Klient pro prohlížeč je v textu pro zjednodušení často nazýván *webový klient*.

[6] První volbou byla databáze IndexedDB, protože z pohledu specifikace W3C je Web SQL standard zastaralý a nahradit ho má právě IndexedDB (`http://www.w3.org/TR/IndexedDB/`). Nakonec se ale zvolila SQLite databáze, na které běží i serverová část aplikace. IndexedDB plugin pro Cordovu navíc obsahuje mnoho bugů a na některých verzích operačního systému Android nebyl vůbec použitelný.

nabízející se alternativní technologií pro extrahování webových stránek[7], přináší toto řešení řadu výhod i omezení. Následuje jejich seznam s podrobnějším vysvětlením, případně se srovnáním s doplňky do prohlížečů.

- **Jednotná implementace.** Přestože se některé rozšíření pro prohlížeče navzájem podobají (např. Firefox Add-on a rozšíření pro Google Chrome), mají zpravidla lehce odlišnou architekturu a API. Některé prohlížeče mají architekturu doplňků i zcela odlišnou. Bookmarklety umožňují psát skutečně jednotný kód pro všechny cílové platformy.

- **Velká podpora**. Bookmarklety jsou podporovány ve všech hlavních prohlížečích současnosti[8].

- **Jednoduchá instalace**. Bookmarklety se nemusí instalovat v běžném slova smyslu, existují dvě možnosti, jak je začít používat. Buď lze manuálně vytvořit záložku a do políčka pro URL zkopírovat kód, nebo vytvořit HTML odkaz, který má kód ve svém atributu „href“ a tento odkaz přetáhnout myší do záložek. V obou případech musí být kód uveden klíčovým slovem „`javascript:`“. Instalace současných moderních doplňků bez nutnosti restartu prohlížeče je z uživatelského hlediska však také velmi snadná.

- **Přístup ke stránce.** Bookmarklety běží přímo ve stránce, společně s veškerým ostatním JavaScriptem, který využívá webová stránka. To vyžaduje opatrnost a zodpovědnost vývojáře, který musí zajistit, aby jeho kód nekolidoval s ostatními skripty. Na druhou stranu to přináší výhody, protože vložený kód má přímý přístup ke všem prvkům stránky, se kterou chce manipulovat. Rozšíření prohlížečů právě z důvodu bezpečnosti na sebe kladou různá omezení.

- **Same origin policy (SOP).** *Same origin policy* je bezpečnostní koncept prohlížečů pro kontrolu práv na komunikaci a přístupu k datům skriptů z různých stránek[9]. Z pohledu extraktoru to znamená, že je třeba na serveru explicitně povolit CORS[10], tedy přijímání požadavků od skriptů jiného původu.

- **HTTPS**. Extraktor se ve většině prohlížečů ve výchozím nastavení nespustí, pokud aktuální stránka bude využívat protokol HTTPS. Prohlížeče totiž implicitně blokují nešifrovaný obsah na zabezpečených webech, i když po zablokování většinou nabídnou možnost, jak toto chování dočasně zrušit. Druhou možností, jak obejít toto omezení, je zkusit přístupnost dané stránky na stejné adrese, ovšem s protokolem HTTP.

[7] Extraktor byl zpočátku vyvíjen jako doplněk do prohlížeče Firefox, toto řešení ale ustoupilo právě bookmarkletu.

[8] Jako kritérium „hlavního“ prohlížeče byla uvažována podmínka, že produkt zabírá alespoň 1% trhu. Podpora bookmarkletů bude však pravděpodobně ještě širší. První bookmarklety šlo spouštět už v prohlížeči Netscape.

[9] Skript z první stránky může přistupovat k datům z druhé stránky, pouze pokud mají oba stejný „původ“ (*origin*). Jako stejný původ se považuje URL se stejným protokolem, doménou (včetně subdomén) a portem. Podrobnější informace na http://en.wikipedia.org/wiki/Same-origin_policy.

[10] Cross-origin resource sharing - sdílení zdroje pro jiný původ. Jedná se o mechanismus, kterým server povolí přijímání požadavků z jiných zdrojů, které by jinak nebyly povoleny kvůli *Same origin policy*. Na rozdíl od JSONP, tedy dalšího způsobu, kterým lze obcházet SOP, umožňuje CORS posílat nejenom GET, ale všechny HTTP metody.

Obrázek 5.3: Ukázka instalace bookmarkletu přetažením do lišty záložek.

Instalace extraktoru tedy probíhá pouhým přetažením odkazu do záložek prohlížeče (viz obrázek 5.3). Samotný bookmarklet obsahuje jen minimální část kódu, který po spuštění dotazuje server, aby mu vrátil samotný výkonný kód extraktoru. Tímto způsobem je zajištěna automatická aktualizace extraktoru bez jakýchkoliv nároků na zásah uživatele. Aktuální kód se stáhne ze serveru pokaždé, když je extraktor zapnut.

Komunikace se serverem je zajištěna pomocí HTTP POST požadavků, v jehož těle jsou extrahovaná data společně s názvem úkolu, do kterého se data uloží. Součástí požadavku je i bezpečnostní token sloužící k identifikaci uživatele.

5.3 Databázové schéma

V této sekci se nachází popis schématu a jednotlivých entit databáze. Schéma je v obou případech stejné, pouze verze pro mobilního klienta neobsahuje tabulku pro uložení uživatelů[11]. Návrh databáze, společně s obslužnými metodami, zahrnuje širší funkčnost, než kterou je možné používat ze současné verze GUI. Toto je jedním z aspektů, u kterých byl kladen důraz na snadné budoucí rozšíření.

5.3.1 Tabulka Node

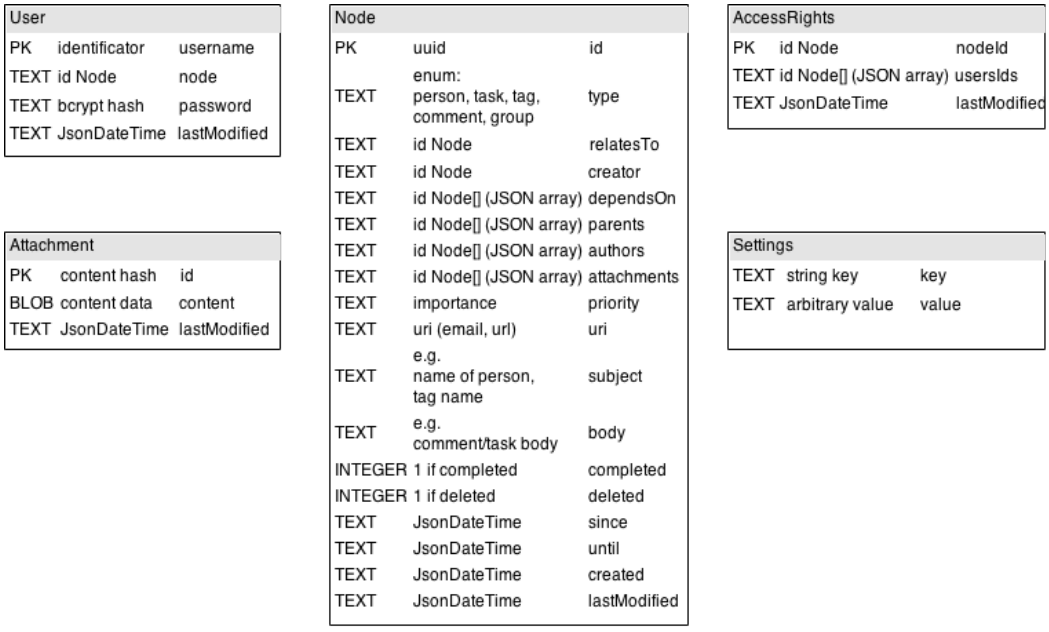
Základní entitou v aplikaci je Node[12]. Jednoznačným identifikátorem prvků v tabulce (id) je vygenerované uuid. Entita se používá jednak pro uložení úkolů, jednak pro uložení skupin, tagů, komentářů i informací o uživateli (bez přihlašovacích dat).

Úkoly mají u sebe uložené id svých rodičů, mohou mít více předchůdců (i když GUI zatím pracuje pouze s konceptem jednoho rodiče), jedná se tedy o vztah *m:n*. Skupiny u sebe drží id úkolů, které do nich patří, podobně jako tagy ukládají id přináležících úkolů. Úkol může patřit do více skupin (GUI zatím pracuje s přiřazováním úkolů pouze k jedné skupině) nebo mít více tagů a vice versa, jedná se tedy opět o vztah *m:n*. Komentáře se přímo vztahují vždy právě k jedné Node entitě, drží u sebe ale zároveň i id hlavní entity, ke které náleží, aby nemusely být z databáze vybírány rekurzivně[13]. Pole identifikátorů je uloženo jako serializovaný JSON. U každého záznamu u všech tabulek se udržuje údaj o jeho poslední modifikaci, tato informace slouží pro sdílení dat.

[11] K registraci a přihlašování uživatelů dochází vždy oproti serveru, tato data by neměla v lokální mobilní databázi smysl. Dalším důvodem, proč nejsou na tomto místě ukládána, je bezpečnost.

[12] Zde byl použit lehce zavádějící název entity, který v kontextu aplikace evokuje příslušnost k technologii Node.js, nemají spolu však nic společného.

[13] Typickým příkladem je diskuse pod úkolem. Diskutující na sebe vzájemně odkazují, většina komentářů tedy přímo náleží jinému komentáři. Všechny ale u sebe mají uložené id i hlavní Nody, v tomto případě úkolu. Takto mohou být získány z databáze jedním jednoduchým dotazem.



Obrázek 5.4: Schéma databáze, shodné pro server i mobilní aplikaci. První sloupec obsahuje SQLite typ, druhý popis uložených dat a třetí název sloupce v tabulce.

5.3.2 Tabulka User

Tabulka User je přítomná pouze v serverové databázi a drží v sobě čtyři hodnoty. Unikátní uživatelské jméno, bcrypt hash hesla, datum poslední změny a odkaz na entitu Node, kde mohou být uložena o uživateli další data, například jméno.

5.3.3 Tabulka Attachment

Tabulka Attachment je používána pro ukládání příloh. Soubory jsou ukládány v datovém typu BLOB ve formě řetězce zakódovaného pomocí base64[14]. Přílohy mají unikátní id vytvořené jako hash svého obsahu. Pokud je tedy nějaká příloha vytvořena opakovaně, nebo přidána k více úkolům, v databázi je uložena pouze jednou.

5.3.4 Tabulka AccessRights

V tabulce AccessRights jsou uložená přístupová práva pro jednotlivé Nody. Pro každou entitu Node existuje právě jeden záznam udržující seznam id uživatelů, kteří mají k entitě přístup. Minimálně tedy vždy obsahuje autora entity. Více o sdílení v sekci 5.4.4.

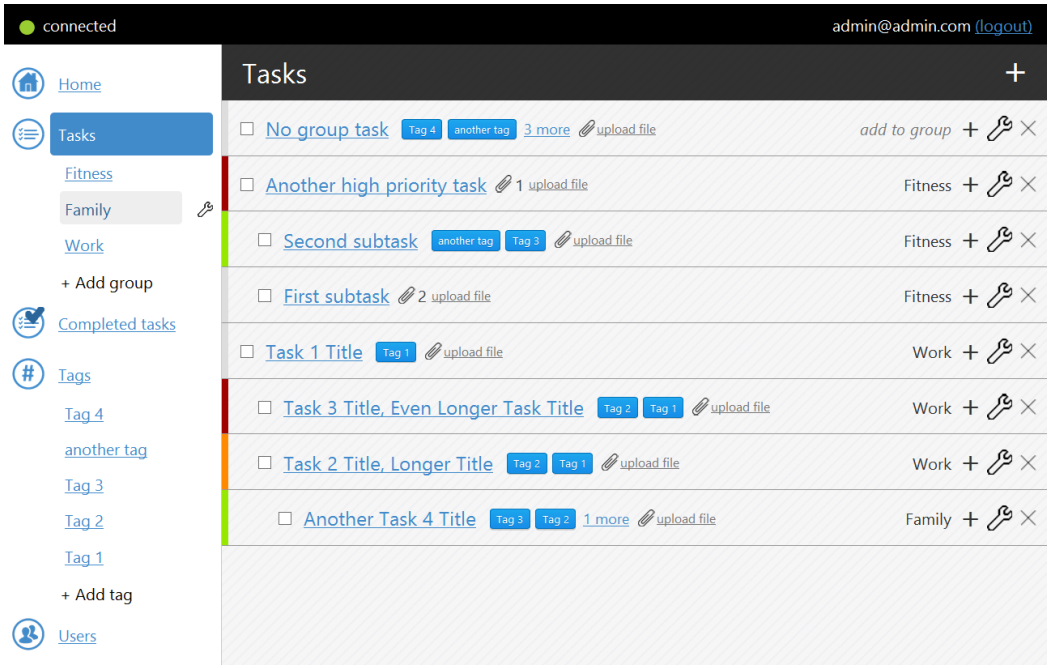
[14] Data příloh byla před aktualizací knihovny Socket.IO na novější verzi, která zvládá i přenos binárních dat, posílána přes AJAX a byla proto kódována a poté ukládána zakódovaná, aby nedocházelo k nadbytečnému převodu dat. Po přechodu na novější verzi knihovny už byl formát ponechán. Výhodou je, že některá mobilní API vrací soubory přímo v base64 řetězci a nemusí se proto převádět. Nevýhodou je o něco větší velikost ukládaných dat.

5.3.5 Tabulka Settings

Tabulka Settings slouží pro uložených dat ve formě klíč a hodnota. V aplikaci je použita pro ukládání posledních časů úspěšných synchronizací, nebo v mobilní aplikaci například pro uložení poslední adresy, na kterou byla aplikace připojena.

5.4 Funkcionalita, realizace a uživatelské rozhraní

Tato sekce přináší souhrn základní funkcionality nástroje společně s ukázkami GUI. Nachází se zde také popis některých implementačních detailů. Grafické uživatelské rozhraní bylo vytvořeno v anglickém jazyce[15].



Obrázek 5.5: *IAM Toolkit* - přehled webového rozhraní.

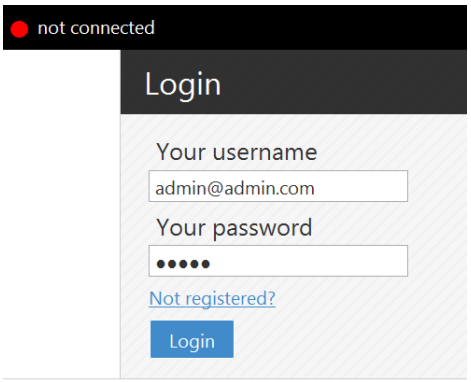
5.4.1 Registrace a přihlášení

Před prvním použitím aplikace se uživatel musí přihlásit. K dispozici jsou 3 testovací účty, nebo lze registrovat nový. Seznam účtů poskytnutých pro snazší testování:

- **admin@admin.com (heslo admin)** - tento účet obsahuje testovací data, pokud je spuštěn server na portu 3000
- **email2@email.com (heslo 12345)**

[15] GUI a zdrojový kód byly vytvořeny v anglickém jazyce, aby byly co nejvíce univerzální pro budoucí vývoj.

- **email3@email.com (heslo 12345)**



Obrázek 5.6: *IAM Toolkit* - přihlašování. Aplikace značí, že není připojena přes Socket.IO kanál.

Přihlašovací data jsou poslána na server HTTP POST požadavkem, kde jsou ověřena oproti databázi. Použitý hashovací algoritmus pro ukládání hesel je *bcrypt*[16], který umožňuje i automatické generování soli pro hesla, výsledný řetězec v sobě obsahuje sůl, hash zadaného hesla a informaci o počtu iteracích použitých ke generování hesla. Pokud je uživatel úspěšně ověřen, aplikace vygeneruje autentizační token[17] podepsaný tajným klíčem uloženým na straně serveru. Token má platnost nastavenou na 14 dní. V případě úspěchu vrací server objekt *Identity* obsahující kromě tokenu také základní informace o uživateli.

Na straně klienta je token uložený do *LocalStorage*[18]. Způsob uložení přihlášení je tedy na obou klientech shodný. Identita uživatele je poté ověřována přímo na úrovni vytváření Socket.IO spojení, token je poslán jako parametr metody zajišťující vytvoření spojení.

5.4.2 Správa úkolů

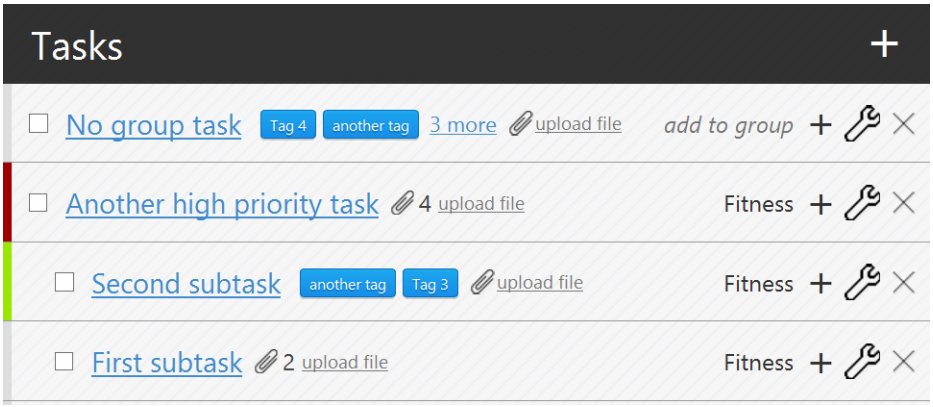
Po přihlášení může uživatel vytvářet, upravovat a mazat úkoly. Úkoly mají různé atributy a jsou rozdělené do skupin, lze je také označovat tagy (viz obrázek 5.7). Ve většině prohlížečů je možné k úkolům nahrávat přílohy. Konkrétně v těch co implementují HTML5 FileReader API[19], podpora shrnuta v příloze C.

[16] **bcrypt** je už mnoho let doporučován jako jeden z nejbezpečnějších hashovacích algoritmů pro ukládání hesel. Jedním ze článků popisující sílu algoritmu z praktického hlediska je např. http://codahale.com/how-to-safely-store-a-password/.

[17] Metoda použití **tokenu** oproti cookies byla zvolena zejména z toho důvodu, že cookies nemají podporu při psaní aplikace pro Android za využití Cordovy. Ověřování pomocí tokenu přináší i další výhody, více o tokenech ve článku [19].

[18] **LocalStorage** je standardizovaný způsob, jakým na straně klienta ukládat data ve formě *klíč + hodnota*. V současné době má velmi rozsáhlou podporu, viz. http://caniuse.com/#search=localstorage.

[19] **FileReader API** je moderní standardizovaný způsob, jakým načítat a ukládat soubory a manipulovat s nimi na straně klienta. Toto řešení bylo zvoleno jako ukázka možností stále více podporovaného standardu HTML5.



Obrázek 5.7: *IAM Toolkit - detail přehledu úkolů.*.

5.4.2.1 Práce s daty

Manipulace s entitami probíhá přímo na straně klienta, ovládací prvky jsou zobrazovány a skrývány dynamicky a stejně tak přechod mezi stránkami je realizován pomocí JavaScriptu.

Načtené stránky nebo ovládací prvky poté získávají data asynchronně. U webového klienta jsou data získána pomocí volání API, veškerý přenos dat probíhá přes Socket.IO a to včetně souborů jako příloh. V případě mobilního klienta jsou data dotazována přímo z lokální databáze. Některé stránky nebo ovládací prvky dostávají data získaná dříve přímo jako parametr[20]. Implementačně je toto asynchronní řešení ale složitější, protože klient musí jednotlivé části stránky doplňovat po prvotním vykreslení.

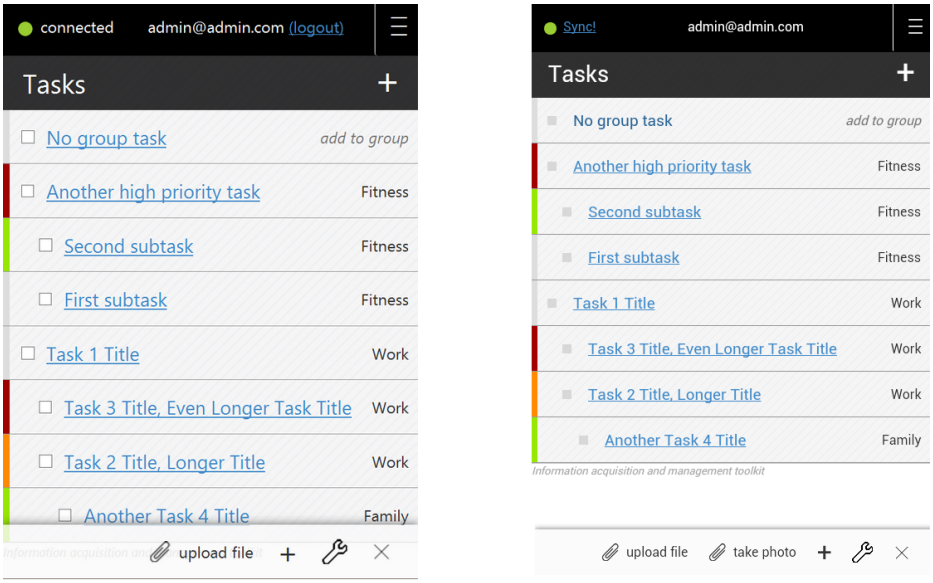
Data vytvořená klientem se zpracují, dočasně uloží a rovnou vykreslí ve stránce. Poté jsou propagována k permanentnímu uložení. V případě webového klienta jsou odeslány přes Socket.IO, v případě mobilního klienta jsou uložené v lokální databázi. Pokud je mobilní klient připojen k serveru, vysílá data zároveň Socket.IO kanálem.

Při používání klienta na menších obrazovkách (zejména u mobilního klienta, ale i u webového klienta při malé šířce okna prohlížeče) je prostor pro vykreslování menší a některé ovládací prvky či atributy úkolů jsou skryté. V přehledu úkolů chybí na velkých obrazovkách zobrazované tagy a ovládací prvky pro načtení souborů, přidání podúkolů a tlačítek pro editaci a mazání úkolů. Tyto prvky jsou přístupné v dalším menu, které se zobrazí v dolní části obrazovky dlouhým podržením konkrétního úkolu (viz obrázek 5.8). Většina prvků je také dostupná z detailního zobrazení úkolu.

5.4.2.2 Aktualizace klientů v reálném čase

Mezi klienty, které jsou v Socket.IO spojení se serverem, probíhá aktualizace některých dat v reálném čase. Server po obdržení a uložení dat totiž zároveň vysílá tato data příslušným posluchačům. Posluchači musí být přihlášení na stejný účet, případně se musí jednat o sdílená data, ke kterým mají přístup. Funkční a ozkoušená je zejména automatická změna po přidání nových úkolů nebo editaci jejich atributů a přidání/editaci komentářů, a to i mezi více účty

[20] Tímto způsobem se minimalizuje přenos dat.



Obrázek 5.8: *IAM Toolkit* - ukázka rozhraní pro malé obrazovky, vlevo prohlížeč, vpravo mobilní aplikace. Obě mají zobrazené dolní menu pro akce s úkolem.

(pokud se jedná o sdílené prvky). Automatická aktualizace příloh funguje pouze v rámci jednoho účtu a aktualizace po synchronizaci může vyžadovat opětovné načtení stránky.

Listing 5.2: Ukázka serverové části kódu. Přijatý objekt je vysílán do všech klientů pro všechny zainteresované uživatele.

```
socket.On<NodeSaved>(async ns =>
{
    // ...

    ar.UsersIds.ForEach(userId =>
    {
        socket.Broadcast.To(userId).Emit(ns);
    });
});
```

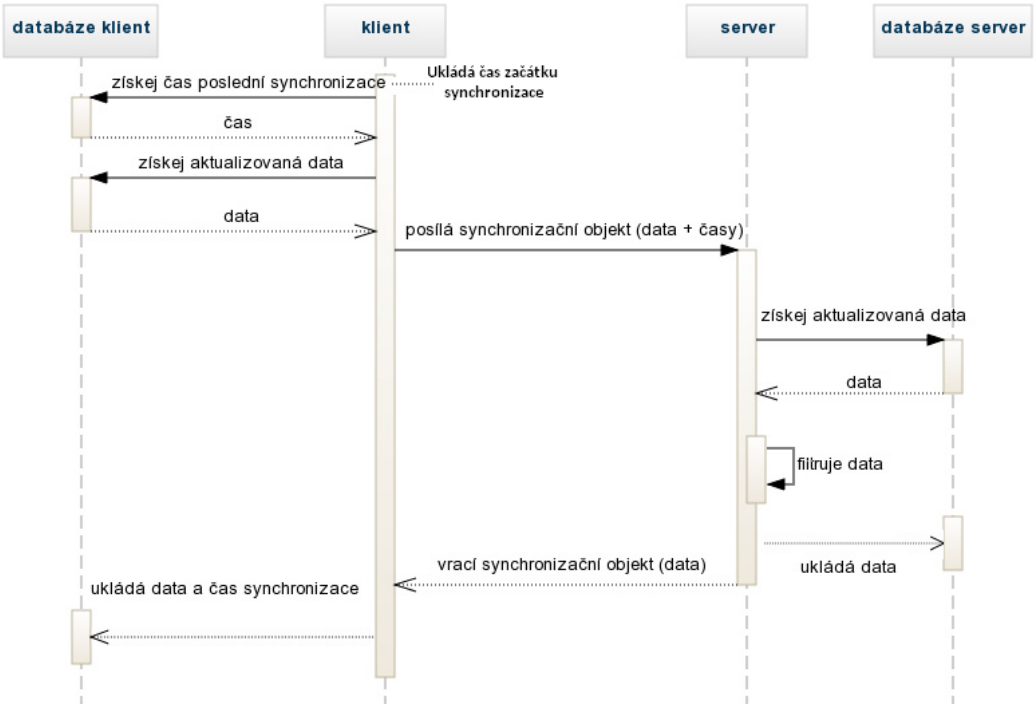
5.4.3 Synchronizace

Synchronizace probíhá na dvou úrovních. Jednak mezi mobilním klientem a serverem, jednak mezi dvěma servery navzájem. V obou případech dochází k obousměrné synchronizaci. Konflikty jsou řešeny zachováním nověji upravené verze.

Klient[21] nejprve zkontroluje čas poslední synchronizace a vybere ze svého úložiště všechny nové elementy z tabulek Node, Attachment a AccessRights. V případě, že je klientem Node.js

[21] Zde myšlen klient ve smyslu Socket.IO spojení, v rámci aplikace jím může být mobilní klient i Node.js server.

server, vybere data i z tabulky User. Data (dále označována jako *data-c*) jsou soustředěna do objektu pro přenos, tomu jsou přidány i dva časové údaje, čas poslední synchronizace a čas začátku této synchronizace. Tento objekt je poslán na server a klient čeká na odpověď. Server nejprve vybere z databáze svoje data[22](dále označována jako *data-s*), která mají čas poslední modifikace novější než přijatý údaj. *Data-s* jsou následně filtrována podle toho, jestli k nim má klient autentizovaný v tomto spojení přístup. Poté jsou serverem procházena *data-c* a pokud neexistuje jejich novější ekvivalent v *data-s*, jsou ukládána na serveru s časem poslední modifikace nastaveným na přijatý údaj začátku celé operace. Pokud naopak existuje novější verze stejné entity v *data-c*, je její varianta vyřazena z *data-s*. Na konci jsou vyfiltrovaná *data-s* odeslána klientovi. Klient přijatá *data-s* ukládá na své úložiště, tato data mají už ze serveru nastavený čas poslední změny na čas začátku celé operace. Po úspěšné proceduře u sebe klient aktualizuje čas poslední synchronizace pro konkrétní server na hodnotu začátku synchronizace.



Obrázek 5.9: Sekvenční diagram průběhu synchronizace.

5.4.3.1 Mobilní klient - server

Protože mobilní klient byl napsaný pro práci bez připojení k serveru, základním požadavkem je umožnění přenosu takto vytvořených dat na server. K synchronizaci dochází ihned po připojení k serveru a poté v online režimu kdykoliv na vyžádání uživatele stiskem tlačítka. Mobilní klient informuje uživatele o probíhající synchronizaci i o jejím ukončení krátkými

[22]Z totožných tabulek jako klient, ovšem bez tabulky User. Pro synchronizaci přihlašovacích účtů musí požadavek vycházet aktivně z účtu, ze kterého mají být odeslána.

zprávami zobrazenými v dolní části obrazovky. Po dokončené operaci se mobilní klient aktualizuje, aby zobrazil nová data. Mobilní klient odesílá veškerá data, server vrací data, ke kterým má klient přístup (pokud je vlastník nebo jsou s ním sdílená).

5.4.3.2 Server - server

Synchronizace mezi dvěma servery je inicializovaná z webového klienta. Uživatel musí dodat pouze číslo portu serveru. Node.js server, ke kterému je klient připojený, je v tomto Socket.IO spojení v pozici klienta, a Node.js server, jehož port byl zadaný uživatelem, je v pozici obslužného serveru spojení. Synchronizace je opět obousměrná, klientský server odesílá všechna data a cílový server opět vrací pouze data, ke kterým je klient autorizován.

```
Connected successfully to server http://localhost:5002
Syncing...
sending nodes num: 23
sending users num: 3
sending attachs num: 0
sending accessRights num: 20
received nodes num: 1
received attachs num: 0
received accessRights num: 0
Last sync time 2015-05-10T17:51:44.171
Syncing done.
```

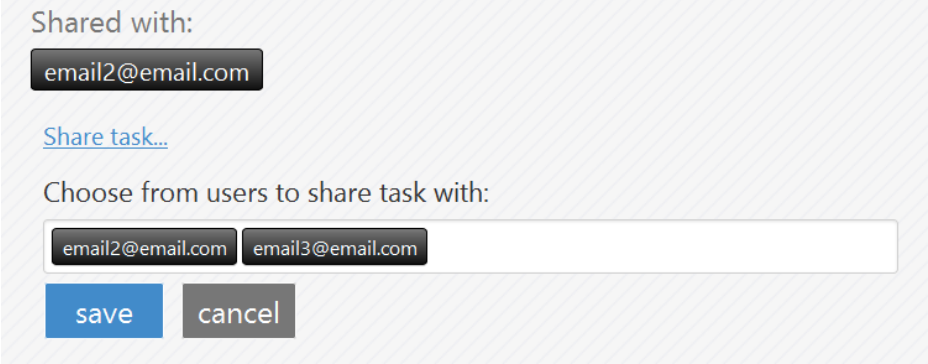
```
Syncing...
received nodes num: 23
received attachs num: 0
received accessRights num: 20
filtering nodes...
filtering attachs...
filtering access rights...
processing data, please wait few seconds...
users done
attachs done
access rights done
nodes done
sending nodes num: 1
sending attachs num: 0
sending accessRights num: 0
Syncing done.
```

Obrázek 5.10: *IAM Toolkit* - logy serverů při synchronizaci, nahoře klient, dole server.

5.4.4 Sdílení a spolupráce

Sdílením se rozumí zpřístupnění úkolů pro jiné uživatele. Toto je zásadní výhoda oproti testovaným službám, u kterých šlo s výjimkou jednoho nástroje (hiTask 3.3.7) sdílet pouze na úrovni celých listů. V detailu úkolu (viz obrázek 5.11) je zobrazen ovládací prvek, který uživateli nabízí seznam dostupných (tj. registrovaných) účtů, kterým lze udělit přístup. Přístupová práva se rekurzivně propagují na podúkoly a jejich komentáře. Sdíleným uživatelům se zobrazí i případné tagy a skupiny, do kterých je úkol (či jeho podúkoly) zařazen. Pro zobrazení nově sdílených úkolů je vyžadováno opětovné načtení stránky. Nastavovat práva na sdílení lze i z mobilního klienta, uplatněná budou však až po synchronizaci.

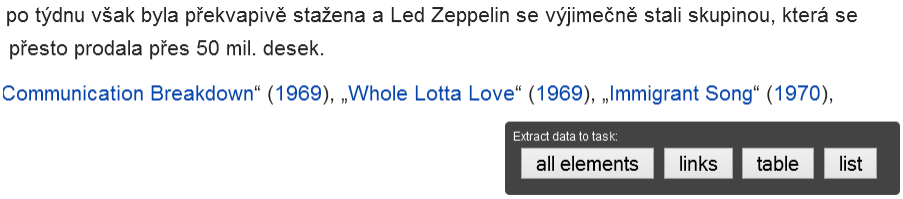
U nově vytvořených komentářů se přístupová práva nastavují podle úkolu, ke kterému se vztahují, to platí i pro zanořené komentáře. Nově vytvořené podúkoly sdílených úkolů práva nepřebírají, uživatel má tak větší kontrolu i možnosti v tom, co chce s ostatními sdílet.



Obrázek 5.11: *IAM Toolkit* - nastavování sdílení z webového rozhraní.

5.4.5 Extrakce

Po přetažení odkazu obsahujícího kód bookmarkletu z přihlášeného webového klienta do záložek lze používat v prohlížeči extraktor. Po kliknutí na bookmarklet se v pravém dolním rohu obrazovky zobrazí malé menu[23]. Rozhraní nabízí čtyři tlačítka, pro extrakci veškerého HTML, pouze odkazů, pouze tabulek a pouze seznamů. Ve všech případech se extrahuje buď z myší označené oblasti (lze označovat i nesouvisející oblasti pomocí držení klávesy CTRL), nebo z celé stránky, pokud není nic označeno[24].



Obrázek 5.12: *IAM Toolkit* - ovládací prvek extraktoru vložený do stránky na wikipedii.

Po kliknutí na některé z tlačítek je uživateli zobrazen náhled extrahovaného HTML a políčko pro název úkolu, který se má z těchto dat vytvořit. Uživatel může název upravit a data odeslat na server, nebo operaci zrušit. Data jsou odeslána na server, ze kterého byl bookmarklet přetažený do záložek.

[23] Extraktor bude blokován samotnými prohlížeči, pokud bude spouštěn na stránkách využívajících protokol HTTPS. Více v sekci 5.2.5.

[24] Pokud už uživatel dříve něco označil, ve většině prohlížečů se nelze označení zcela zbavit. I pokud uživatel klikne myší a vizuálně označení zmizí, z pohledu JavaScriptového API vykazuje prohlížeč označenou oblast v místě kliknutí. Nejjednodušším způsobem, jak se označení zbavit, je refresh stránky.

Extraktor ze stránky vybírá vždy celé elementy, pokud jsou některé označené jen částečně, oblast se doplní. Program se dále snaží rozumně interpretovat uživatelské příkazy. Pokud uživatel žádá například tabulku, která se v označené oblasti vůbec nevyskytuje, aplikace se snaží najít nejbližší vyhovující prvek. Prohledávání z hlediska stromové struktury prvků probíhá na sousedících, vnořených i nadřazených elementech.

Extraktor lze experimentálně spustit i v mobilním prohlížeči, ačkoliv pro tuto platformu nebyl zamýšlen. Instalace probíhá stejně, nástroj se musí uložit jako bookmarklet do záložek. V případě mobilních prohlížečů je tato operace o něco složitější. Doporučený postup (může se v závislosti na konkrétním mobilním prohlížeči mírně lišit):

1. Přejít na webového klienta z mobilního prohlížeče. Lokální URL serveru lze zobrazit na stejném místě jako QR kód výběrem položky v menu.
2. Zobrazit stránku s bookmarkletem obsaženým v odkazu.
3. Podržet tento odkaz a vybrat možnost „kopírovat URL odkazu“.
4. Kliknout na volbu „přidat do záložek“ aktuální (jakoukoliv) stránku.
5. Do políčka „url odkazu“ vložit zkopírovaný kód bookmarkletu.

Takto je extraktor připraven k použití i z mobilního prohlížeče zařízení, které je připojené k lokálnímu serveru. Ovládání extraktoru je potom shodné, ale z uživatelského hlediska obtížnější. Extraktor byl zkušebně zprovozněn na mobilním prohlížeči od Androidu a data extrahoval, i když vždy pouze z celé stránky a ne z vyznačené oblasti[25]. Na aktuální verzi Chrome pro mobilní zařízení se extraktor nepodařilo spustit.

5.4.6 Mobilní získávání dat

Mobilní aplikace získává data třemi způsoby. Nahráváním souborů, pořízením fotografie a z Android Intentů s klíčem *EXTRA_STREAM* (více o Android Intentech v sekci 4.3.5).

Stisknutím tlačítka „upload file“ se otevře dialog, ze kterého lze přejít do *Galerie* zařízení. Na tuto událost mohou být v systému zaregistrováni i posluchači jiných aplikací, proto mohou být v otevřeném okně nabízeny i další aplikace, například Dropbox. Tlačítko se nachází v dolním menu zobrazeném podržením úkolu na některé z přehledových stránek s více úkoly a také v detailním přehledu úkolu. Ve druhém případě se nahraný soubor vykreslí ihned po jeho kompletním načtení.

V dolním menu je k dispozici také tlačítko „take photo“ pro vytvoření přílohy pomocí fotoaparátu zařízení. Z otevřeného dialogu lze vybrat opět z některé ze zaregistrovaných aplikací pro pořízení fotografie, v nabídce je minimálně základní Android aplikace *Fotoaparát*.

Pro uložení souborů do aplikace pomocí Adroid Intentů nevychází uživatelská akce z aplikace, ale z některé části systému, která umožňuje sdílení souborů, tedy například z aplikace *Galerie* nebo *Moje soubory*. Po nalezení konkrétního souboru uživatel vybere možnost sdílení a z nabídnutého seznamu aplikací vybere *IAM Toolkit*. Následně je otevřena obrazovka už

[25] K tomu dochází pravděpodobně proto, že prohlížeč nezná metodu `window.getSelection()`. Extraktor byl testován Android verzi 4.1.2, ve verzi 4.4 by měla být tato funkce podporována.

ve vyvíjené aplikaci, s náhledem souboru (pokud je to obrázek) a s ovládacím prvkem pro přiřazení k úkolu nebo vytvoření nového úkolu, ke kterému bude příloha uložena. Po uložení je uživatel přesměrován na detail úkolu.

5.5 Licence

Aplikace byla vyvíjena pod licencí *Mozilla Public License* verze 2.0. Plné znění licence se nachází na adrese http://mozilla.org/MPL/2.0/.

Kapitola 6

Testování

V poslední fázi této diplomové práce bylo provedeno testování se třemi uživateli. V kapitole je nejprve popsána motivace testování a cílová skupina, následuje seznam úkolů pro test a popis průběhu testu s jednotlivými uživateli. Na konci kapitoly je souhrn nalezených nedostatků.

6.1 Účel testu

Test měl odhalit jednak chyby v logice programu, jednak zejména problémy s použitelností a celkově s ovládáním aplikace. Z průběhu testu a vyplněných dotazníků vzešlo i několik námětů pro vylepšení a budoucí vývoj systému.

6.2 Cílová skupina

Aplikace cílí na uživatele zvládající alespoň základní práci s počítačem. Měli by být schopni orientovat se v prostředí internetu a webových stránek. Nutnou podmínkou je také zkušenost s používáním operačního systému Android pro mobilní zařízení. V obou případech by pak měli být zvyklí pravidelně pracovat s nějakými nástroji či službami, které alespoň okrajově souvisí s organizováním dat, například s emailem nebo kalendářem.

6.2.1 Screener

K otestování, zda kandidát vyhovuje kritériím pro participanta testu, slouží *Screener*. Jedná se o prvotní dotazník, který získá základní informace o uživateli pomocí sady otázek s připravenými odpovědmi, s nichž má kandidát vždy právě jednu vybrat. V přiloženém *Screeneru* je u odpovědí v závorce zobrazený údaj o ideálním rozložení odpovědí od uživatelů, tučně jsou zvýrazněny požadované odpovědi.

1. **Používáte pravidelně počítač nebo notebook s připojením k internetu?**
   a) **Ano (3)** b) Ne (0)
2. **Používáte mobilní telefon nebo jiné přenosné zařízení s operačním systémem Android?**
   a) **Ano (3)** b) Ne (0)

3. **Jak často pracujete se službami jako email či kalendář?**
   a) **Pravidelně (2)** b) **Příležitostně (1)** c) Nikdy (0)

4. **Dorozumíte se anglicky?**
   a) **Ano** (3) b) Ne (0)

6.2.2 Pre-test dotazník

Z kandidátů, kteří odpovídají profilu uživatele definovaného Screenerem, se stávají účastníci testu. Těmto je ještě podán krátký předtestový dotazník, který o nich zjistí další doplňující informace. Ty lze využít například při snaze bližšího pochopení, jaký typ uživatelů má problém s různými aspekty používání aplikace. Test má opět připravené odpovědi, ze kterých se vybírá právě jedna.

1. **Považujete se za zkušeného uživatele IT technologií?**
   a) Rozhodně ano b) Spíše ano c) Spíše ne d) Rozhodně ne
   e) Nedokážu posoudit

2. **Jak často používáte různé aplikace pro organizování času a úkolů na počítači? (kalendáře, úkolovníky a jiné)**
   a) Velmi často b) Občas c) Zřídka d) Nikdy
   e) Nedokážu posoudit

3. **Jak často používáte různé aplikace pro organizování času a úkolů v mobilním zařízení? (kalendáře, úkolovníky a jiné)**
   a) Velmi často b) Občas c) Zřídka d) Nikdy
   e) Nedokážu posoudit

4. **Vyhledáváte novinky v této oblasti, které by Vám ulehčily práci?**
   a) Velmi často b) Občas c) Zřídka d) Nikdy
   e) Nedokážu posoudit

5. **Za jak zkušeného uživatele mobilních zařízení se systémem Android se považujete?**
   a) začátečník (volání, psaní zpráv, pořizování fotek)
   b) pokročilý (vyhledávání a instalace nových aplikací)
   c) expert (znalosti o vývoji aplikací)

6.3 Průběh testování

Testování probíhalo vždy na vlastním počítači a mobilním zařízení participanta, ten byl proto ve známém prostředí. Dalším přínosem tohoto přístupu je otestování aplikace v různých prostředích. Funkci moderátora testu zastupoval vývojář aplikace, tedy autor této práce. Moderátor zaznamenával průběh testu, pokládal doplňující dotazy a v případně nutnosti napovídal participantům jak pokračovat. Každý test zabral jednu až dvě hodiny.

6.3.1 Úkoly

Pro potřeby testu byla vytvořena sada úkolů, které museli participanti pokud možno samostatně splnit. Nejprve byl program vyzkoušen v prohlížeči, poté se test zaměřil na mobilní aplikaci a nakonec na interakci mezi oběma platformami, tedy na synchronizaci. Byla testována i synchronizace mezi více servery a extrakční nástroj.

Úkoly na sebe navazovaly, ale test se podřizoval uživateli. Pokud měl s něčím problém, nebo naopak chtěl něco vyzkoušet, byl mu ponechán prostor a moderátor podle toho volil doplňující úkoly a dotazy. Moderátor se také snažil zasadit konkrétní formulaci jednotlivých úkolů do kontextu profese nebo znalostí jednotlivých participantů. Následující přehled tedy zahrnuje minimální sadu úkolů, kterou každý participant někdy v průběhu testu splnil.

1. Spustit server a zobrazit přihlašovací stránku webového klienta.
2. Registrovat nového uživatele a přihlásit se na něj.
3. Vytvořit úkol v nové skupině. Poté vytvořit další úkoly ve stejné skupině.
4. Vytvořit úkoly v jiné skupině, přesouvat úkoly mezi skupinami, odebírat úkoly ze skupin, vytvořit úkol bez skupiny. Orientovat se v přehledu právě zobrazených skupin.
5. Přiřadit nové i existující tagy. Filtrovat úkoly dle tagů.
6. Vytvořit a rozpoznat podúkoly (i do více úrovní).
7. Nastavovat všechny parametry úkolů, toto nastavení později měnit či úkoly smazat. Označovat úkoly jako splněné.
8. Přidávat a odebírat přílohy k úkolům.
9. Nainstalovat a spustit mobilní aplikaci. Připojit aplikaci přes QR kód a přihlásit se na dříve vytvořený účet.
10. Přidávat a editovat úkoly na obou platformách a data synchronizovat.
11. Přidávat a editovat přílohy na obou platformách a data synchronizovat. V případě mobilní aplikace vyzkoušet i pořízení fotografie či načtení přílohy z galerie či souborového systému mobilního zařízení.
12. Spustit další server, synchronizovat na něj data a zobrazit je ve webovém rozhraní.
13. Sdílet úkol pro jiného uživatele. Komunikace pomocí komentářů s jiným uživatelem (tuto roli zastal moderátor na svém vlastním zařízením).
14. Použít extraktor k vyextrahování části participantem zvolené stránky.

6.3.2 Post-test dotazník

Participant byl při testu neustále vyzýván ke sdělování všech jeho postřehů, které ho napadly při používání aplikace. Takto bylo od každého uživatele získáno velké množství podnětů, proto byl dotazník po testu omezen už pouze na tři krátké otázky.

1. Co se vám na aplikaci nejméně líbilo? Uveďte 2-3 problémy.
2. Co se vám na aplikaci nejvíce líbilo? Uveďte 2-3 pozitiva.
3. Jak byste ohodnotil/a aplikaci na stupnici 0-10 (10 nejlepší)?

6.4 Participanti

U každého participanta je představen jeho uživatelský profil (na základě dotazníků) a zachyceno v jakém prostředí bylo testováno. Poté je velmi krátce shrnut průběh testování. Nalezené problémy i s informací, u kterých uživatelů se objevily, jsou v samostatné sekci.

6.4.1 Participant 1

Participant 1 je středoškolský žák, muž ve věku 12 - 18 let. S aplikacemi pro organizování času a úkolů na počítači nepracuje nikdy, v mobilním zařízení zřídka. Novinky v této oblasti nevyhledává. Považuje se za pokročilého uživatele mobilních zařízení a spíše zkušeného uživatele IT technologií obecně.

Aplikace byla testována na operačním systému Windows 7 a Android 4.1.2. Použité mobilní zařízení bylo Samsung Galaxy S III mini.

Participant se v prostředí rychle a snadno orientoval a úkoly plnil rychle. I když řešení našel, dával aktivně podněty k ovládání či funkcionalitě. Participant považoval za největší problém pomalost odezvy mobilní aplikace. Dále potom fakt, že extraktor nefunguje na stránkách využívajících protokol HTTPS. Uživatel dále zmínil, že vzhled je příliš jednoduchý a ne zcela doladěný. Jako největší klady uvedl participant automatickou aktualizaci nových či změněných entit (úkolů, komentářů...) u připojených klientů a fungování obou typů synchronizace (mobilní klient-server i server-server). Participant celkově hodnotí aplikaci známkou 7/10.

6.4.2 Participant 2

Participant 2 je vysokoškolsky vzdělaná žena ve věku 18 - 30 let. S aplikacemi pro organizování času a úkolů pracuje zřídka na počítači i mobilním zařízením. Novinky v této oblasti také nevyhledává. Považuje se za spíše méně zkušeného uživatele IT technologií obecně, v případě mobilních aplikací za pokročilého.

Aplikace byla testována na operačním systému Windows 7 a Android 4.3. Použité mobilní zařízení bylo Samsung Galaxy S III.

Participantce trvalo trochu déle, než se v aplikaci zorientovala, ale po krátké chvíli už ji ovládala bez problémů. Uživatelka neměla dle svých slov problém ani s rychlostí odezvy

mobilní aplikace, ani s velikostí ovládacích prvků. Jako největší problém viděla neschopnost mobilní aplikace pracovat s velkými přílohami. Dále potom matoucí vzhled checkboxů pro splnění úkolů a absenci editace některých atributů v místě jejich zobrazení (zejména u tagů). Uživatelka také oceňuje synchronizaci a real-time zobrazování entit mezi více klienty, vhodné jí připadá i rozložení menu aplikace. Participantka celkově hodnotí aplikaci známkou 8/10.

6.4.3 Participant 3

Participant 3 je student vysoké školy, muž ve věku 18 - 30 let. Aplikace pro organizování času a úkolů používá na počítači zřídka, na mobilu častěji. Novinky v této oblasti však sám aktivně nevyhledává. Považuje se za velmi zkušeného uživatele IT technologií.

Aplikace byla testována na operačním systému Windows 7 a Android 4.1.2. Použité mobilní zařízení bylo Samsung Galaxy S II.

Participant se v aplikaci orientoval poměrně rychle a se základním použitím aplikace (bez synchronizace) neměl téměř žádné problémy. Jako největší problémy zmínil fakt, že by očekával automatickou synchronizaci mobilního klienta pokud je online a ne až po stisknutí tlačítka. Uživatel by také očekával, že v editaci úkolu bude moci měnit všechny jeho atributy, včetně příslušnosti do skupiny. Dále by uvítal lepší nápovědu při práci s extraktorem. Jako největší přínosy aplikace vidí její multiplatformnost a intuitivní ovládání. Participant celkové hodnotí aplikaci známkou 8/10.

6.5 Nálezy

Poslední sekce kapitoly o testování je věnována nalezeným problémům. U každého nálezu je popsán problém, okolnosti projevení a návrh na řešení problému. Některé problémy byly po testování ještě opraveny. Výpis je řazen dle závažnosti nálezů. U nálezů jsou v závorce v názvu uvedeny čísla participantů, u kterých byl problém objeven.

6.5.1 Kritické

Tyto nálezy představují velmi závažný problém v aplikaci a je nutné je opravit před ostrým nasazením aplikace.

6.5.1.1 Synchronizace mezi více než dvěma subjekty (1)

Synchronizace mezi 2 subjekty, i opakovaná, probíhá bez problémů. Data ze serveru 1, která už byla jednou synchronizována na server 2, se však nesynchronizují na server 3.

Toto bylo způsobeno tím, že původní návrh nepracoval s variantou rovnocenných serverů, ale s lokálními servery a centrálním serverem, přičemž synchronizace probíhala vždy pouze mezi lokálním a centrálním serverem. Každý server obsahoval pouze jednu časovou stopu poslední úspěšné synchronizace. Problém byl opraven, každý subjekt si nyní pamatuje čas poslední synchronizace pro každý server zvlášť.

6.5.2 Střední

Nálezy v této kategorii jsou problémy, které sice bezprostředně nebrání nasazení aplikace, ale jejich oprava je také nutná.

6.5.2.1 Přechod na stejnou stránku pomocí odkazu (1, 2, 3)

Pokud se uživatel nachází na nějaké stránce a klikne na odkaz směřující na tu samou stránku, nic se nestane.

Toto je způsobeno samotným Extbrain Frameworkem. Vytváření odkazů a přechod mezi stránkami je realizován pomocí hashe v URL adrese (část na konci adresy nacházející se za znakem #). Metoda pro přechod na novou stránku reaguje na změnu hashe, po kliknutí na odkaz se stejným hashem se hash v URL nezmění a posluchač navěšený na *onhashchange* není zavolán. Obtížnost opravy je střední, rychlou opravou by mohlo být připojení nějaké proměnné části při generování odkazu, která by se při interpretaci routování ignorovala, posluchač by však byl zavolán, protože by došlo ke změně hashe.

6.5.2.2 Používání knihovny selectize (1, 2, 3)

Výběr položek probíhá v pořádku, ale při vytváření nového prvku ho uživatel často ztratí tím, že chce operaci potvrdit kliknutím vedle. Není také zřejmé, že nové elementy lze vytvořit stejným ovládacím prvkem sloužícím pro výběr. V mobilním prohlížeči je s ovládáním ještě větší problém, pro smazání aktuální položky je na některých zařízeních nutno kliknout alespoň jednou na jinou klávesu než *backspace*.

V prohlížeči lze očekávat, že se uživatelé s ovládacím prvkem naučí pracovat, mohla by se přidat nápověda. Bug při mazání na mobilu je nutné opravit přímo v knihovně, nebo knihovnu vyměnit. Náročnost opravy je vysoká.

6.5.2.3 Velké přílohy na mobilním klientovi (1, 2)

Mobilní aplikace nezvládá práci s velkými přílohy, některé přílohy neuloží, nebo při jejich načítání spadne.

Toto je způsobeno buďto limity SQLite databáze, nebo doplňku pro přístup k této databázi z Cordova aplikace. Aplikace vykazuje toto chování, i když jsou přílohy načítány z databáze jednotlivě. Nejlepším řešením by bylo přílohy ukládat v souborovém systému a v databázi držet pouze cesty, tato změna by si vyžádala rozsáhlejší změny v aplikaci.

6.5.2.4 Mobilní GUI, velikost prvků (1, 3)

Velikost některých ovládacích prvků na mobilních zařízeních není stále dostatečná, nebo tyto prvky kolem sebe nemají dostatečný volný prostor.

Oprava by vyžadovala opakované procházení rozhraním mobilní aplikace a snadné, ale časově náročné, ladění vzhledu jednotlivých stránek.

6.5.2.5 Dolní menu úkolů zobrazované na podržení (2, 3)

Při podržení úkolu na malých obrazovkách se zobrazí menu v dolní části obrazovky, uživatel o této funkci ale neví.

Menu musí být někde schované, na jeho permanentní zobrazení není prostor. Řešením by mohlo být poskytnout ikonku pro jeho zobrazení, nebo na jeho existenci upozornit nějakým průvodcem.

6.5.2.6 Rychlost odezvy mobilní aplikace (1)

Mobilní aplikace nereaguje tak rychle, jako webový klient.

U tohoto problému není důvod znám. Problém se projevuje i bez použití příloh. Jednou z možných příčin může být velikost použitých knihoven. Náročnost opravy velmi vysoká.

6.5.2.7 Vícenásobné spuštění synchronizace (1)

Po opakovaném kliknutí na tlačítko pro synchronizaci v mobilním zařízení je proces spuštěn vícekrát.

Řešením by bylo tlačítko deaktivovat do doby ukončení synchronizace, náročnost lehká.

6.5.3 Mírné

Tyto nálezy nepředstavují závažný problém v aplikaci, nemusí být opraveny ihned.

6.5.3.1 Absence skupin v editaci úkolu (2, 3)

Při editaci úkolu není přítomna možnost pro přiřazení úkolu. Toto ovládání je k dispozici v přehledu úkolů. Řešením je rozšířit formulář pro úpravu úkolu, to však vyžaduje rozsáhlé změny při jeho vykreslování i zpracování.

6.5.3.2 Možnost schovat podúkoly (1)

V seznamu úkolů není možnost pro schování/zobrazení podúkolů. Při větším množství úkolů se seznam stává méně přehledným.

Oprava spočívá v implementaci schovávání skupin podúkolů a nalezení vhodného místa pro takový ovládací prvek v GUI, obtížnost střední.

6.5.3.3 Vytváření úkolu z přehledu tagu (1)

Pokud se vytváří nový úkol ze stránky přehledu úkolů pro jednotlivý tag, je mu tento tag přiřazen. V formuláři pro nový úkol ale není tento tag vizuálně předvyplněný.

Při vytváření úkolu tímto způsobem je už informace o aktuálním tagu přítomna, stačí tedy konkrétní tag vykreslit do formuláře, obtížnost lehká.

6.5.3.4 Extrahování selže pro velká data (1)

Při pokusu o odeslání velkých extrahovaných dat je tento požadavek serverem zamítnut. Problém vyřešen nastavením limitu pro velikost dat, které server přijímá.

6.5.3.5 Nevýrazné tlačítko pro splnění úkolu (2)

Tlačítko pro splnění úkolu je malé a není zřejmá jeho funkce. Řešením by bylo prvek zvětšit a lépe graficky zobrazit. Náročnost je střední, HTML input typu *checkbox* totiž nelze stylovat pomocí CSS, standardní řešení používají HTML tag `<label>`, který zobrazují a stylují místo checkboxu.

6.5.3.6 Editace všech atributů přímo z detailu úkolu (2)

V detailu úkolu nelze přímo editovat všechny atributy, nelze například přidávat a odebírat tagy tam, kde jsou vykreslené, ale je nutné zobrazit formulář. Řešením by bylo kromě formuláře přidávat další malé ovládací prvky pro přidávání/editaci/mazání ke všem místům, kde jsou atributy zobrazovány. Oprava by byla náročná.

6.5.3.7 Automatická synchronizace v online režimu (3)

Mobilní zařízení neodesílá některá data rovnou, i když je online, ale až po stisku tlačítka pro synchronizaci.

Problém byl opraven pomocí Socket.IO emitování, musely se však implementovat obslužné metody pro real-time překreslení dalších prvků ve stránce.

6.5.3.8 Absence nových tagů v menu bez refreshe (3)

Po přidání nového tagu vytvořeného přes formulář úkolu se tento tag nezobrazí v menu ihned, ale až po refreshi. Řešením je na příslušnou akci spustit metodu aktualizace této části menu.

6.5.4 Návrhy

V této kategorii se nachází spíše návrhy uživatelů, než nalezené chyby. Koncipovány jsou jako stručný přehled pomocí seznamu, v závorce opět čísla participantů, kteří změnu navrhují.

- Zavedení druhého políčka pro heslo při registraci kvůli kontrole překlepů (1).
- Drag and drop řazení skupin a tagů v menu (1).
- Sdílené úkoly by se mohly objevovat i ve speciální skupině zobrazované v menu (1).
- Synchronizovat pouze přihlášeného uživatele (1).
- Formulář pro úpravu úkolu nebo přidání podúkolu zobrazit hned pod souvisejícím úkolem (1).
- Pro extraktor přidat nápovědu (3).

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat za použití daných technologií nástroj pro správu a sdílení úkolů, společně s možností extrakce informací z webu. Výstupem je aplikace spustitelná ve webovém prohlížeči i na operačním systému Android a extraktor určený pro běh ve webovém prohlížeči. Výsledný program je v souladu nejen se zadáním této práce, ale splňuje i další požadavky vyplývající z rešerše podobných nástrojů a zejména z průběžných konzultací s vedoucím této práce. Strukturovaný popis těchto požadavků se nachází v kapitole 2.

Aplikace byla v závěru testována s uživateli. Proběhlé testy se týkaly alespoň v základní míře všech funkcionalit systému, odhalily několik nedostatků a vzešla z nich řada podnětů pro vylepšení aplikace. Nálezy byly kategorizovány a analyzovány v kapitole 6.

Hlavním přínosem práce je fungující aplikace, při jejíž vývoji byl kladen důraz na architekturu a čistotu kódu. Budoucí rozšíření by měl usnadňovat i návrh databáze. Dílčím přínosem práce je i rešerše (kapitola 3) vypracovaná na téma nástrojů pro správu a sdílení úkolů. Popis u každého nástroje dodržuje pevně danou strukturovanou formu. V rámci práce bylo také vytvořeno nebo rozšířeno několik *stubů* pro překladač *Saltarelle*. Tyto třídy mohou být jednoduše začleněny do dalších projektů. V poslední řadě slouží práce jako ukázka použití a propojení poměrně nových technologií.

Další vývoj by se měl zaměřit především na dorovnání některých nedostatků, které aplikace má v porovnání s existujícími řešeními. Nástroj přináší několik výhod, zejména serverové řešení a existenci extrakčního nástroje, ale v mnohém zaostává. Odrazovým můstkem pro budoucí práci by měly být nalezené chyby při testování. Vzhledem k situaci na trhu je stěžejním problémem absence aplikace pro iOS. Díky použité technologii pro vývoj mobilní aplikace (*Cordova*) by rozšíření na tuto platformu mohlo být usnadněné. Aplikace by se měla dále zaměřit na integrování dat z různých externích zdrojů.

Literatura

[1] V. Hoštička. *Extrakce webu pomocí ExtBrain Extractoru.* 2012. Bakalářská práce, Fakulta elektrotechnická, České vysoké učení technické v Praze, Praha. Vedoucí práce: Ing. Tomáš Novotný.

[2] J. Mašek. *Extrakce webu pomocí Mozilla Application Frameworku.* 2010. Bakalářská práce, Fakulta Elektrotechnická, České vysoké učení technické v Praze, Praha. Vedoucí práce: Ing. Tomáš Novotný.

[3] Nástroj Basecamp — domovská stránka [online]. [cit. 12. 2. 2015]. Dostupné z: https://basecamp.com/.

[4] Framework Bootstrap — domovská stránka [online]. [cit. 22. 4. 2015]. Dostupné z: http://getbootstrap.com/.

[5] Apache Cordova — domovská stránka [online]. [cit. 20. 4. 2015]. Dostupné z: https://cordova.apache.org/.

[6] Nástroj Droptask — domovská stránka [online]. [cit. 11. 2. 2015]. Dostupné z: https://www.droptask.com/.

[7] Projekt ExtBrain — domovská stránka [online]. [cit. 20. 4. 2015]. Dostupné z: http://www.extbrain.net/.

[8] Nástroj GmailSharedTasks — domovsk stránka [online]. [cit. 11. 2. 2015]. Dostupné z: http://gmailsharedtasks.com/.

[9] Nástroj hiTask — domovská stránka [online]. [cit. 11. 2. 2015]. Dostupné z: https://hitask.com/.

[10] Jazyk LESS — domovská stránka [online]. [cit. 22. 4. 2015]. Dostupné z: http://lesscss.org/.

[11] Node.js — domovská stránka [online]. [cit. 20. 4. 2015]. Dostupné z: http://nodejs.org/.

[12] Nástroj Producteev — domovská stránka [online]. [cit. 10. 2. 2015]. Dostupné z: https://producteev.com/.

[13] Nástroj Remember The Milk — domovská stránka [online]. [cit. 12. 2. 2015]. Dostupné z: http://www.rememberthemilk.com/.

[14] Překladač Saltarelle — domovská stránka [online]. [cit. 20. 4. 2015]. Dostupné z: http://www.saltarelle-compiler.com/.

[15] SQLite databáze — domovská stránka [online]. [cit. 21. 4. 2015]. Dostupné z: http://www.sqlite.org/.

[16] Nástroj TaskShare — domovská stránka [online]. [cit. 12. 2. 2015]. Dostupné z: http://taskshare.org/.

[17] Nástroj TickTick — domovská stránka [online]. [cit. 9. 2. 2015]. Dostupné z: https://www.ticktick.com/.

[18] Nástroj Todoist — domovská stránka [online]. [cit. 10. 2. 2015]. Dostupné z: http://todoist.com/.

[19] M. Woloski. 10 things you should know about tokens [online]. [cit. 15. 4. 2015]. Dostupné z: https://auth0.com/blog/2014/01/27/ten-things-you-should-know-about-tokens-and-cookies/.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

AJAX Asynchronous JavaScript and XML

BLOB Binary large object

CORS Cross-origin resource sharing

CSS Cascading Style Sheets

CSV Comma-separated values

DOM Document Object Model

GUI Graphical User Interface

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

JSON JavaScript Object Notation

JSONP JSON with padding

SDK SoftwareDevelopment Kit

SOP Same Origin Policy

SQL Structured Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

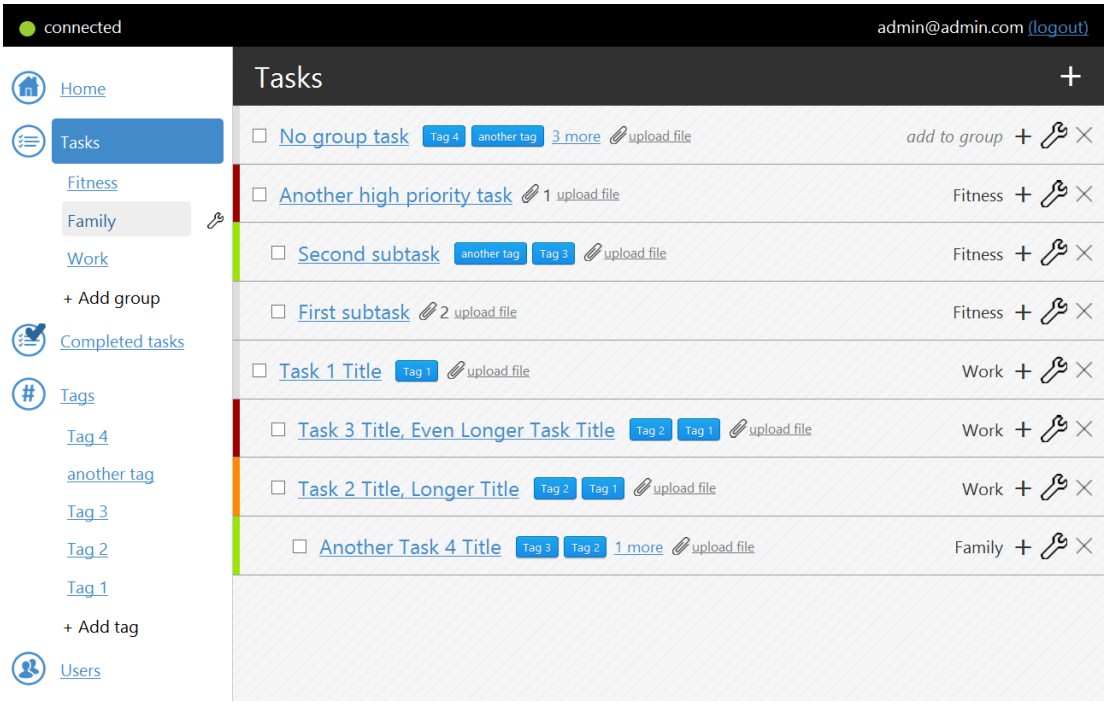
UUID Universally Unique Identifier

W3C World Wide Web Consortium

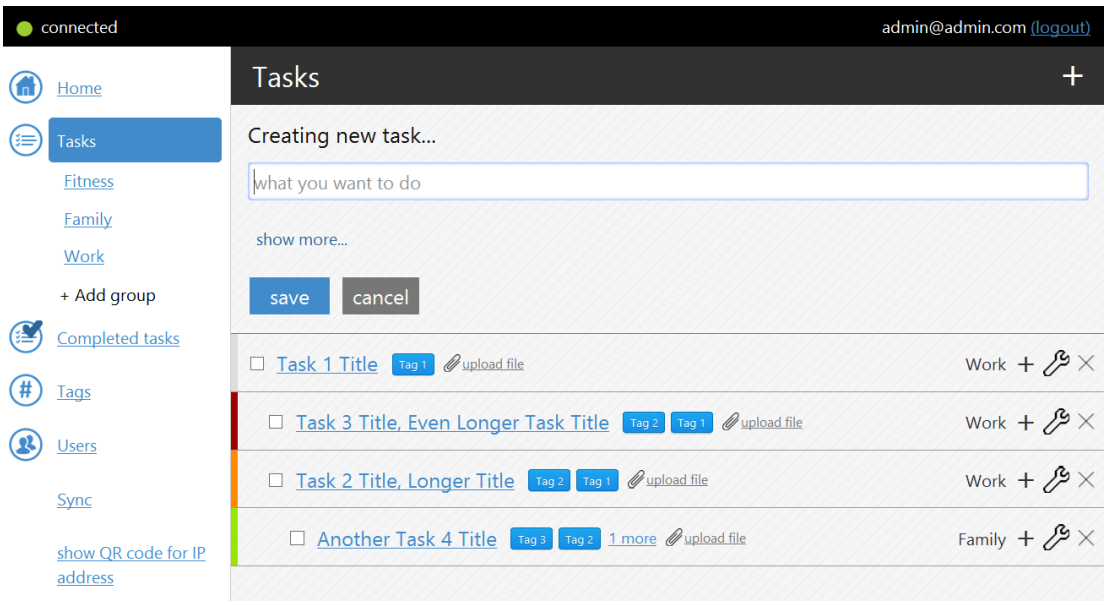
XML Extensible Markup Language

Příloha B

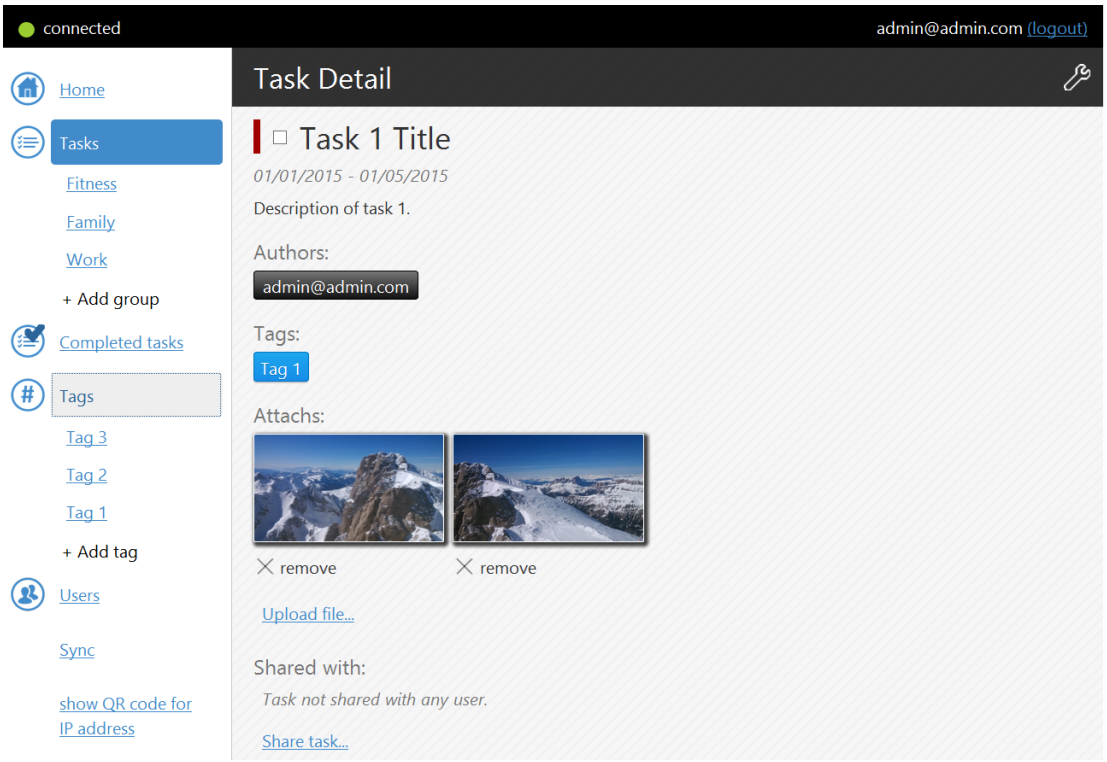
Screenshoty uživatelského rozhraní



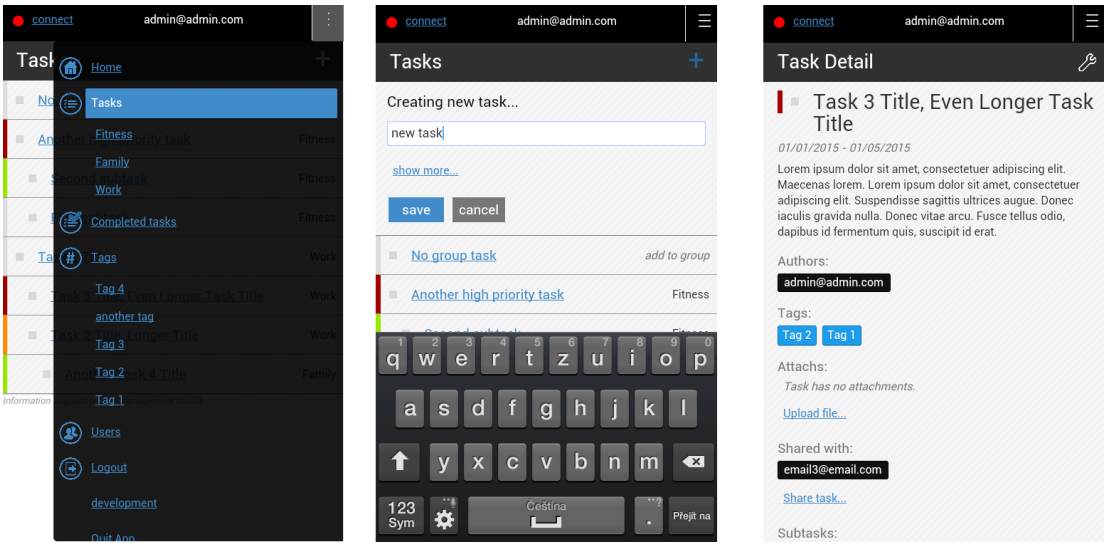
Obrázek B.1: *IAM Toolkit* - přehled webového rozhraní.



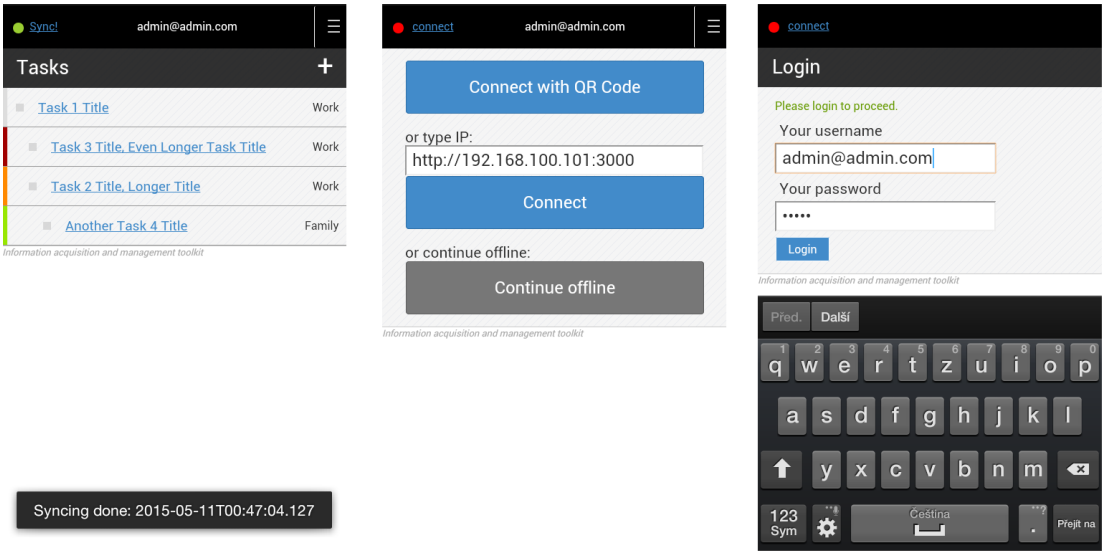
Obrázek B.2: *IAM Toolkit* - přidání nového úkolu.



Obrázek B.3: *IAM Toolkit* - detail úkolu.



Obrázek B.4: *IAM Toolkit* - mobilní aplikace. Menu, přidávání úkolu, detail úkolu.



Obrázek B.5: *IAM Toolkit* - mobilní aplikace. Dokončená synchronizace, obrazovka připojení, přihlašování.

Příloha C

Popis instalace a spuštění

V této příloze jsou popsány nároky prostředí, ve kterém lze spustit aplikaci, společně s návodem na instalaci a spuštění aplikace. Při popisu cest bude použita zkratka *«root»*, která označuje cestu *«kořenový adresář přiloženého CD»/SP/Build/Debug*.

Instalace a spuštění serverového prostředí

Server je napsaný pro platformu Node.js (4.4) s použitím velkého množství modulů, které jsou předinstalované. Serverová část aplikace je připravena pro okamžité spuštění na 64 bitových systémech Windows, v *«root»/* se nachází standalone verze Node.js pro tuto platformu[1]. Server se spouští příkazem

```
node run-server.js <<číslo portu>>
```

tedy například

```
node run-server.js 3000
```

Pro pohodlí jsou připraveny dva *.cmd soubory pro jednoduché spuštění na Windows. Jedná se o soubory *«root»/run-server-3000.bat* a *«root»/run-server-4000.bat*, které spouští server na portu, který mají v názvu. Adresa serveru je potom např. `http://localhost:3000/`.

Aplikace je nastavena tak, že server na portu 3000 se vždy spustí i s testovacími daty. Ta jsou uložena pro uživatele **„admin@admin.com“** s heslem **„admin“**. Ostatní servery neobsahují po spuštění žádná data. Po opětovném spuštění se data na serveru s portem 3000 obnoví do původní podoby, data na ostatních serverech zůstanou zachována[2]. Pro smazání dat nějakého serveru stačí ručně smazat soubor *«root»/storage/data_ «číslo-portu».sqlite*.

[1] Standalone verze Node.js pro Windows je ve formě jediného souboru *node.exe* a nevyžaduje instalaci. K dispozici je varianta pro 64 i 32 bitové systémy.

[2] Toto řešení je zvoleno z důvodu snazšího testování aplikace.

Alternativní spuštění

Aplikaci lze spustit i bez použití standalone Node.js verze. V takovém případě je nutné soubor *node.exe* smazat a mít nainstalované prostředí Node.js ve verzi 0.10.28[3]. Před prvním spuštěním může být třeba zadat příkaz z *«root»/*

```
npm install
```

pro build Node.js modulů. *Npm*[4] je nainstalovaný společně s Node.js.

Spuštění webového klienta

Webový klient je přístupný z prohlížeče na adrese *http://localhost:«číslo-portu»*. Program funguje bez problému v prohlížečích Firefox, Chrome a Opera. V prohlížečích Internet Explorer a Safari (verze pro Windows) nelze nahrávat přílohy, pro manipulaci se soubory je totiž použito HTML5 FileReader API, které v těchto prohlížečích není podporováno, nebo je podpora pouze částečná[5].

Instalace a spuštění mobilního klienta

Klient pro Android se instaluje jako standardní aplikace. Soubor *IAMToolkit-debug.apk*[6] je třeba nahrát do mobilu a spustit. V nastavení mobilního zařízení je nutné povolit instalaci z cizích zdrojů. Pracovní označení aplikace je *IAM Toolkit*, aplikaci zatím nebyla vytvořena vlastní ikonka, ponechána je výchozí ikonka z *Cordovy* (4.3).

Při prvním spuštění je třeba připojit aplikaci k serveru, přihlásit se do aplikace a tedy autentizovat uživatele[7]. Adresu serveru je možné načíst přes QR kód, který lze zobrazit v prohlížeči (odkaz se nachází přímo v hlavním menu), případně zadat ručně. Poté už lze aplikaci používat kdykoliv v offline režimu, tzn. bez nutnosti připojení na některý ze serverů.

[3] Aplikace byla vyvíjena ve Windows na Node.js verzi 0.10.28, testována i na verzích 0.10.36 a 0.12.0 ve Windows a na verzi 0.10.26 v operačním systému Ubuntu. Aplikace poběží pravděpodobně i na ostatních Node.js verzích 10 a novějších, autor práce to ale nezaručuje. Verzi 0.10.28 lze stáhnout na http://blog.nodejs.org/2014/05/02/node-v0-10-28-stable/.

[4] *Npm* je balíčkovací systém pro instalování modulů pro různé platformy, Node.js ho využívá jako svůj hlavní balíčkovací systém. Viz. https://www.npmjs.com/.

[5] Kompletní přehled podpory FileReader API lze najít na http://caniuse.com/#feat=filereader. V tabulce lze vidět i od jakých verzí jednotlivých prohlížečů je API podporováno. Safari na platformě OS X od Applu by dle tabulky mělo API podporovat, nebylo to ale vyzkoušeno.

[6] Soubor aplikace se nachází v adresáři */SP/cordova_ client/platforms/android/ant-build/*.

[7] Přihlásit se lze na některý z účtů vytvořený pomocí webového klienta, nebo z účtů obsažených v testovacích datech systému.

Příloha D

Obsah přiloženého CD

- **DP**
  - **Client**: `C#` projekt pro klientskou část aplikace
  - **Server**: `C#` projekt pro serverovou část aplikace
  - **Framework**: `C#` projekt použitého a rozšířeného Extbrain Frameworku
  - **cordova_klient**: Cordova projekt mobilní aplikace
    - **IAMToolkit-debug.apk**: Mobilní aplikace pro Android
  - **Build/Debug**: adresář s vygenerovaným JavaScriptem ze všech projektů, obsahující další soubory pro spuštění aplikace
- **text**
  - **DP_hostivac_2015.pdf**: digitální verze tohoto dokumentu v PDF
  - **source**: adresář se zdrojovým kódem pro *DP_hostivac_2015.pdf*
- **README.txt**: soubor popisující instalaci a spuštění nástroje