

Sem vložte zadání Vaší práce.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
KATEDRA SOFTWAREVÉHO INŽENÝRSTVÍ



Bakalářská práce

GTD - Mít vše hotovo

Michal Sláma

Vedoucí práce: Ing. Jiří Mlejnek

6. května 2015

Poděkování

Děkuji vedoucímu své práce za hodnotné rady a své rodině za podporu při vytváření této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2015

.....

České vysoké učení technické v Praze

Fakulta informačních technologií

© 2015 Michal Sláma. Všechna práva vyhrazena.

Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Sláma, Michal. *GTD - Mít vše hotovo*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Má bakalářská práce se zabývá návrhem a realizací systému osobního plánování na základě metodiky [1]. V této práci vytvořím aplikaci pro centrální uložení dat dostupné přes REST API a mobilní aplikaci pro OS Android napojenou na zmíněné API a poskytují funkce osobního plánování. Zároveň umožním uživateli publikovat své úkoly na Facebook a do Google kalendáře.

Klíčová slova webová aplikace, mobilní aplikace, osobní plánování, spring, android, REST API, Facebook, Google kalendář

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords web application, mobile application, personal planning, spring, android, REST API, Facebook, Google Calendar

Obsah

Úvod	1
1 Cíl práce	3
2 Analýza metodiky <i>GTD</i>	5
2.1 Metodika <i>GTD</i> - Mít vše hotovo	5
2.2 Současné implementace	9
2.3 Shrnutí analýzy metodiky <i>GTD</i>	12
3 Návrh	15
3.1 Rozdělení aplikace	15
3.2 Model nasazení	15
3.3 Technologie	16
4 Návrh	23
4.1 REST API	23
4.2 Datový model	23
5 Realizace	25
5.1 Hosting	26
5.2 GitHub	26
Závěr	27
Literatura	29
A Seznam použitých zkratk	31
B Obsah příloženého CD	33

Seznam obrázků

2.1	Digram zpracování schránky dle metodiky <i>GTD</i> [2]	7
2.2	Podoba webové aplikace Toodledo.[3]	10
2.3	Mobilní aplikace Doit.[?]	11
2.4	Wunderlist [4]	12
3.1	Model nasazení GTD (šedivě zbarvená nasazení ukazují úvahy dalšího rozšíření)	16
4.1	Data model (generováno pomocí MySQL Workbench)	24

Úvod

„Ze všeho nejvíce požírá čas a energii neustále neproduktivní přemýšlení o věcech, které musíme udělat.“

Kerry Glesson

V současném informačním věku naše práce přestává být fyzickou. Práce už není ohraničena jasnými hranicemi a existuje velké množství informačních zdrojů použitelný pro její zpracování. Řešitel je tím vystaven tlaku nejasného zadání i rozsahu a musí si sám určit cíle své práce.

Naše mysl nejasné nepřijímá, jakékoliv takové úkoly nám neustále připomíná, vyžaduje se jejich řešení a znemožňuje nám tím koncentraci. Potřebujeme nový přístup. Přístup, abychom naši mysl dokázali přesvědčit, že máme vše jasné a právě teď se můžeme koncentrovat pouze na aktuální úkol.

Jedním z takových přístupů je metodika vytvořená Davidem Allenem *GTD* [1]. Jeho motto *„Jak zvládnout práci i život a cítit se při tom dobře“* je mým kýženým životním cílem.

Moje práce se zabývá návrhem a implementací informačního systému pro osobní plánování založeném na metodice *GTD*. Provedu analýzu samotné metodiky, navrhnu vhodnou infrastrukturu splňující potřebná kritéria *GTD* [1], implementuji aplikaci pro centrální uložení dat a mobilní aplikaci pro uživatele.

Cíl práce

První cílem mé práce je analyzovat metodiku *GTD* [1]. Prozkoumám existující implementace osobního plánování podporující *GTD* [1]. Navrhnou vlastní informační systém.

Vytvořím aplikaci poskytující WS pro centrální uložení dat. Vyberu nejvhodnější typ webových služeb. Pro tuto aplikaci najdu potřebný hosting a na něm provedu konfiguraci aplikačního a databázového serveru.

Vytvořím mobilní aplikaci pro Android využívající WS služeb předešlé aplikace. Aplikace umožní uživateli správu jeho osobního plánování.

Dnešní internetový svět je propojen skrze sociálních sítí. Proto umožním uživateli publikovat své úkoly do sítí *Facebooku* a *Google kalendáře*. K tomu nastudují potřebné specifikace poskytovaných API

Mým cílem není vytvoření odladěné aplikace, ale získání know-how s vývojem takového systému.

Analýza metodiky *GTD*

V této kapitole se zabývám analýzou metodiky *GTD*, hledám její základní stavební prvky a její přínos pro osobní plánování. Definuji důležité pojmy a postupy pro využití v dalším návrhu aplikace. Dále srovnávám existující implementace osobního plánování a hledám jejich výhody, kterými podporují kvalitnější a přehlednější osobní plánování. Na závěr spojuji výsledky obou předchozích bodů a navrhuji základní strukturu a funkce budoucí aplikace.

2.1 Metodika *GTD* - Mít vše hotovo

Kniha *GTD* [1] vznikla v roce 2001 a jejím autorem je David Allen. Samotná metodika nepředstavuje jasné řešení, podle kterého se můžeme okamžitě začít řídit a změnit tím svůj život. Snaží se vysvětlit obecné zákonitosti lidské mysli a najít takových způsobů plánování, který s ní bude v souladu.

Pojďme se na metodiku podívat podrobněji.

2.1.1 Změna práce

Naše práce přestává být fyzikou a přesunuje se do roviny sběru informací, jejich zpracování a vytvoření závěrů. Zároveň nejsme schopni jednoznačně určit prioritu, protože ta se stává proměnnou závislou na mnoha vnějších i vnitřních faktorech. Naše práce přestává mít jasné hranice, které naše mysl potřebuje. Bez jasných hranic nám mysl bude takové věci neustále sama připomínat a vyžadovat se o jejich řešení. Paradoxem je, že naše mysl v tomto ohledu nejedná rozumně a připomíná nám naše nedořešené věci bez ohledu na vhodnost situace. Při takovém stavu si říkáme „*memohu se soustředit*“, mysl nám ruší naši vlastní koncentraci. A koncentrace je nutnou podmínkou pro vysokou výkonnost a my potřebujeme aktuálnímu úkolu věnovat 100% pozornosti.

Jak toho ale dosáhnout? Potřebujeme naši mysl přesvědčit, že o všech úkolech víme, jsou zaznamenány mimo ní, víme přesně co máme dělat a není riziko zapomenutí.

2.1.2 Produktivita

„Ve znalostní práci není úkol dán, je třeba jej určit. Jaký je očekávaný výsledek práce? Je klíčovou otázkou pro produktivitu znalostních pracovníků. Je to otázka, která si žádá riskantního rozhodnutí. Obvykle na ni neexistuje správná odpověď, místo ní jsou tu možnosti. A má-li být dosaženo produktivity, je třeba jasně specifikovat očekávané výsledky.“

Petr Ducky[1]

Mysl potřebuje jasný cíl, aby mohla udržet potřebné soustředění, nepřemohla ji únava a dokázala rozlišit důležité informace pro zapamatování.

Co udělat při zpracování úkolu:

1. vyjasnit si požadovaný výsledek
2. mít jasno o dalším kroku
3. uložit si úkol do důvěryhodného systému

Začít je třeba od nezákladnější věci. To nám pomůže zvládnout a odpoutat se od každodenních závazků a poté se můžeme lépe soustředit na větší projekty a vize. Při zpracování potřebujeme využít kontroly ve dvou rovinách: horizontální a vertikální. V horizontální přemýšlíme, co všechno potřebujeme udělat. Ve vertikální řešíme konkrétní věc a co je potřeba vykonat k jejímu splnění.

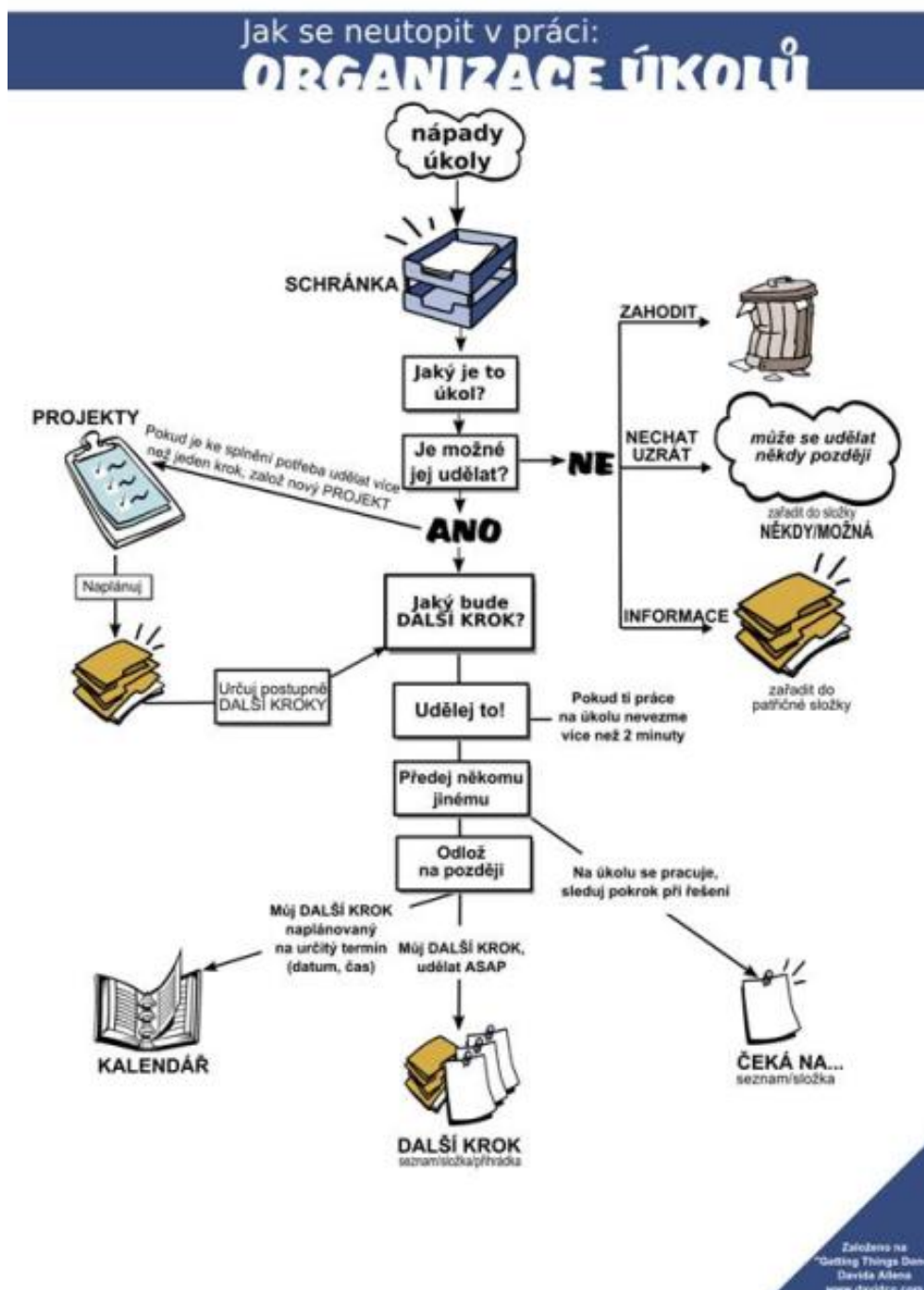
Typický horizontální postup:

1. sběr věcí, které si získaly naši pozornost
2. hodnotíme jejich význam a kroky k jejich splnění
3. uspořádáme si výsledky
4. máme je v plánu a hledáme vhodný čas k provedení
5. provedeme

2.1.2.1 Sběr věcí

Pozornost si získávají věci u kterých si řekneme „*měl bych*“, „*potřebuji*“, „*musím*“. Je potřeba sesbírat všechny věci a žádnou nelze vynechat. Pro naši mysl je důležitým slovem „*všechno*“ a nesmí vzniknout jakákoliv pochybnost. V tuto chvíli ještě nepotřebujeme pouze jedno místo uložení, protože spojujeme všechny naše zdroje např. emaily, fyzické dokumenty, ... Označme taková místa naší schránkou.

2.1.2.2 Zpracování schránky



Obrázek 2.1: Digram zpracování schránky dle metodiky GTD [2]

2. ANALÝZA METODIKY *GTD*

Jak *GTD* graf interpretovat (popíši důležité body):

1. Co je to za úkol? Potřebujeme se nad úkolem zamyslet, čeho se týká a co bude potřeba k jeho splnění.
2. Realizovatelný/ nerealizovatelný. Zde se rozhodujeme, jestli úkol můžeme a chceme splnit. Nerealizovatelný úkol může být pouhá informace. Něco o čem zatím neuvažujeme a nebo pro nás není úkol relevantní a zahazujeme jej.
3. Pro realizovatelné úkoly zjistíme kolik kroků bude potřeba k jejich splnění. V případě, že jich je více zakládáme projekt a pod ním více úkolů. Hierarchie projektů a úkolů má stromovou strukturu, kde úkoly jsou vždy listy stromu.
4. Provedení úkolu
 - a) trvá-li splnění úkolu kratší dobu než 2 minuty - splňme ho
 - b) úkol musí vyřešit někdo jiný - deleguj ho
 - c) úkol řeším a rozhodnu se o dalším kroku nebo zařadím do kalendářePo vyhodnocení úkol končí vždy ve stavu, kdy je jasné jeho další zpracování.

Pravidelné vyprazdňování našich schránek představuje práci navíc a z počátku k němu bude existovat odpor. Musíme docílit návyku a plné důvěry, proč to děláme.

Zde už potřebujeme jednotný systém pro správu našich úkolů a projektů. Jeho navržením se budu zabývat v další kapitole.

Důležité pojmy z *GTD*

- *Činnost/ závazek/ věc/ úkol* – něco, co máme vykonat a k čemu jsme se zavázali
- *Úkol* – zde ve smyslu konkrétního kroku směřujícího ke splnění činnosti
- *Projekt* – může obsahovat další *projekty* a úkoly společné pro jednu činnost
- *Schránka* – místo/ místa na kterých se nacházejí naše činnosti. Zpravidla se jedná o email, šanony,...

2.1.2.3 Pravidelná kontrola úkolů

Kritickým klíčem k úspěchu jsou pravidelná zhodnocení všech aktuálních úkolů. Ideálně každý týden. Naším cílem je udržet si plný přehled o úkolech a zohlednit do nich současné priority.

Stejně jako u vyprazdňování schránky se snažíme o vytvoření návyku.

2.1.2.4 Co mám dělat?

Otázkou, kterou si nyní musíme zodpovědět je, jak nám náš seznam úkolů pomůže v rozhodnutí, co nyní dělat.

K tomu nám budou sloužit *kontexty*.

Představme si, že máme chvíli času před schůzkou a po ruce mobilní telefon. Musíme mít možnost rychle zjistit úkoly splnitelné zavoláním z mobilního telefonu. U úkolů zavedeme kontext představující prostředek k jeho splnění. Např. mobilní telefon, notebook, internet, domov,...

Kontexty jsou pro každého z nás specifické a potřebujeme mít možnost si je spravovat.

Dalšími možnými filtry jsou *dostupný čas*, *dostupná energie* (jak moc „svěží“ musíme být jeho splnění) a *priorita*.

2.1.3 Shrnutí metodiky

- *vše máme mimo naši mysl* – cokoliv potřebujeme udělat máme poznamenáno mimo naši hlavu
- *známe další krok*
- *umíme naše schránky zpracovat*
- *jednotný systém*. Máme jednotný systém, kam zaznamenáváme všechny úkoly/ projekty při zpracování schránky. Máme k němu 100% důvěru a umíme ho správně použít.

2.2 Současné implementace

2.2.1 Toodledo

Odkaz: www.toodledo.com

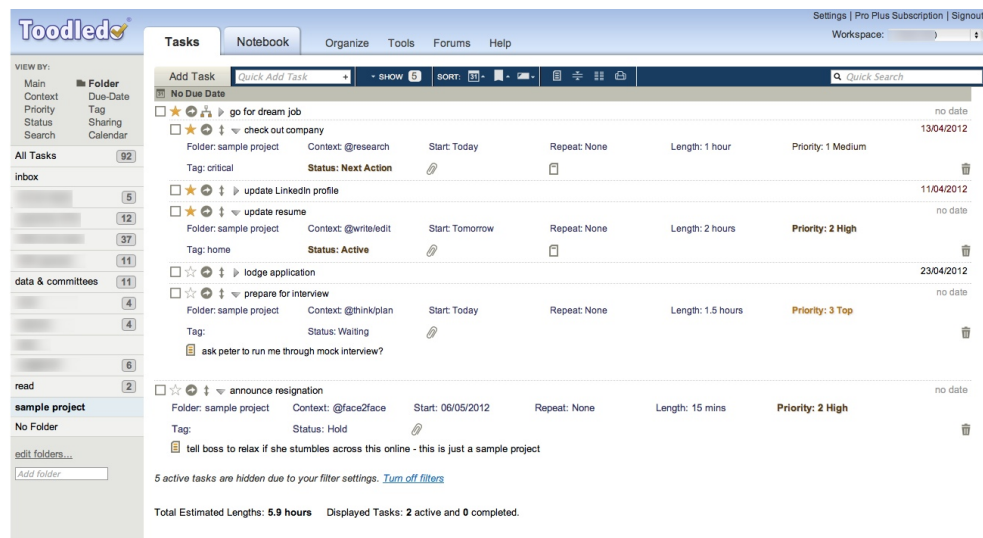
Podporovaná zařízení web: ano, iOS: ano, Android: ano

Výhody: Plně podporuje metodiku GTD. Zakládání a editace úkolů je zde velmi snadná a vše je ukládané okamžitě při změně. U úkolů lze nastavit široké spektrum filtrů od základních kontextů až po definování lokace. Podporuje Google Calendar (widget).

Nevýhody: Neobsahuje integraci se sociálními sítěmi. Projekty jsou až od placené verze.

Tento systém v současnosti používám pro své osobní plánování.

2. ANALÝZA METODIKY *GTD*



Obrázek 2.2: Podoba webové aplikace Toodledo.[3]

2.2.2 Doit.im

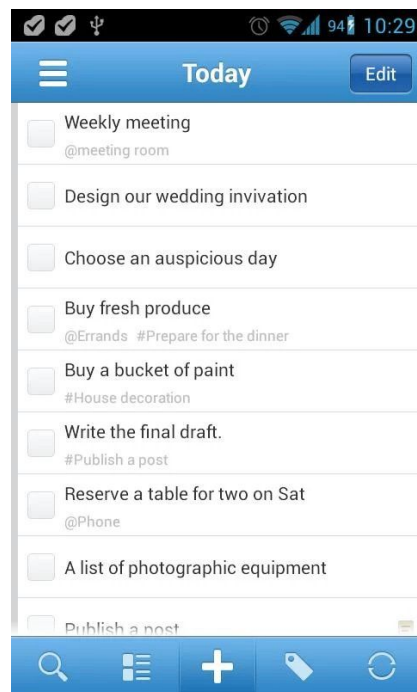
Odkaz: doit.im

Podporovaná zařízení web: ano, iOS: ano, Android: ano

Výhody: Plně podporuje metodiku GTD. Příjemná minimalistická aplikace, která je ihned po registraci připravena k použití.

Nevýhody: Mírně nepřehledná webová aplikace. Podpora Google Calendar až do placené verze.

Tento systém v současnosti používám pro své osobní plánování.



Obrázek 2.3: Mobilní aplikace Doit.[?]

2.2.3 Wunderlist

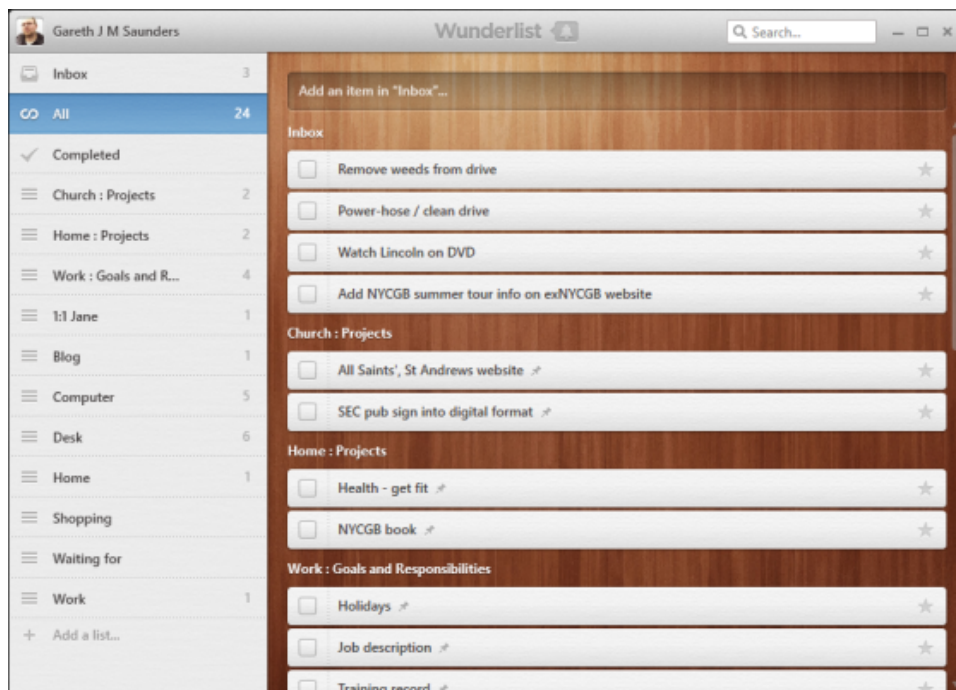
Odkaz: www.wunderlist.com

Podporovaná zařízení web: ano, iOS: ano, Android: ano

Výhody: Plná podpora *GTD*. Pěkné grafické zpracování s další možností přizpůsobení. Po registraci má vše potřebné k okamžitému použití. Jako u Toodledo jsou změny okamžitě ukládány. Zakládání úkolů zasláním emailu. Sdílení úkolů.

Nevýhody: Bez sociálních sítí.

2. ANALÝZA METODIKY *GTD*



Obrázek 2.4: Wunderlist [4]

2.2.4 Todoist

Odkaz: todoist.com

Podporovaná zařízení web: ano, iOS: ano, Android: ano

Výhody: Projekty jsou zde součástí základní verze. Pokud chceme jednoduché ovládání bez další funkcí.

Nevýhody: Nesplňuje nároky *GTD*, ale umožňuje všechny potřebné základní funkce. Desktopová aplikace vychází z omezení mobilních zařízení a nevyužívá velkou plochu.

2.3 Shrnutí analýzy metodiky *GTD*

Z samotné metodiky *GTD* jsem odvodil tyto nároky na aplikaci:

- *Základní funkce.* Není třeba vytvářet složité funkce. Ale ty které existují, musí být spolehlivé a uživatel nesmí mít problém s jejich ovládáním.
- *Dostupnost.* Aplikace se nemůže soustředit pouze na jednu platformu a musí být dostupná z více platforem (web, mobilní zařízení,...).
- *Spolupráce s aplikacemi třetích stran.* U uživatele lze očekávat facebook/gmail účet a aplikace může toto využít.

Současné implementace můj závěr potvrzují. I u malých projektů je podpora pro všechny platformy téměř samozřejmostí. Minimalistické na funkčnost zaměřené uživatelské rozhraní. Synchronizace k emailovými účty je často také přítomna, ačkoliv je už součástí placené verze.

Implementace bude obsahovat

- *Aplikace pro centrální uložení dat vystavující WS pro jejich správu*
- *Aplikace pro OS Android*

Návrh

Tato kapitola popisuje volbu technologií pro implementovanou aplikaci poskytující centrální uložení dat skrze WS a aplikaci na OS Android. Nároky na technologie vycházejí ze závěrů analýzy předešlé kapitoly a dále je rozvíjejí. Současně hledám vhodný server pro hostování WS aplikace.

3.1 Rozdělení aplikace

Ze závěrů analýzy vzešlo, že aplikace musí být pro uživatele dostupná více platformách zahrnující např. mobilní zařízení a web. Proto jsem se rozhodl rozdělit návrh aplikace na dvě části. První bude zajišťovat centrální uložení dat. Druhá pak zajistí pro uživatele samotné GUI pro osobní plánování a správu dat deleguje na první aplikaci. Díky tomuto návrhu se zbavím nutnosti implementovat uložení dat na každé uživatelské aplikaci.

3.1.1 Aplikace pro centrální uložení dat

Skrze WS bude poskytovat centrální uložení dat do databáze. Služby budou dostupné přes internet a samotná aplikace bude nasazena do aplikačního serveru.

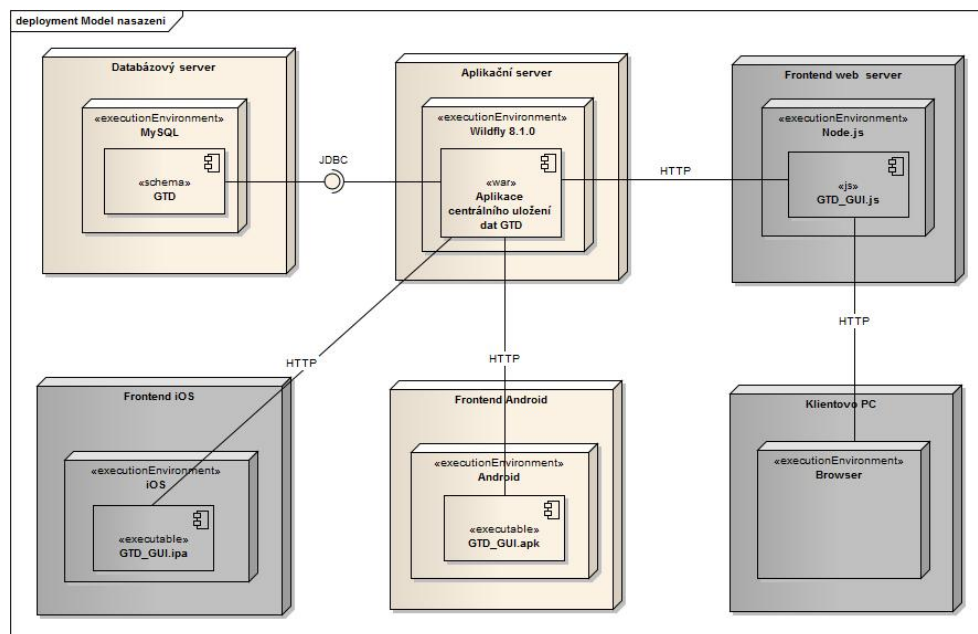
3.1.2 Mobilní aplikace pro Android

První implementace GUI cílená na mobilní zařízení se systémem Android. Fungovat bude na principu tenkého klienta[5].

3.2 Model nasazení

Vycházení z návrhu rozdělení aplikace. GUI aplikace jsou unikátní pro každou platformu s centrální aplikací pro správu data komunikují přes HTTP protokol.

3. NÁVRH



Obrázek 3.1: Model nasazení GTD (šedivě zbarvená nasazení ukazují úvahy dalšího rozšíření)

)s

3.3 Technologie

3.3.1 Aplikace pro centrální uložení dat

3.3.1.1 REST API

Při volbě webového API jsem se rozhodoval mezi dvěma typy webových služeb.

REST API

Bezstavová komunikace přes protokol HTTP využívající typicky metody GET/ PUT/ POST a DELETE. Ty umožňují CRUD operace nad zdrojem jednoznačně identifikovaným pomocí URI. V našem případě je zdrojem např. úkol.

Výhodnou RESTu je jeho škálovatelnost a pro klienta snadná implementace konzumující RESTové služby. Přenosovým formátem nemusí být jenom XML, ale také JSON a další.

Nevýhodou je nejednotná specifikace služby. Existuje WADL a RAML, ale v současnosti nemůžeme žádnou z nich považovat za hlavní.

SOAP

Bezstavová komunikace přes protokol HTTP založená na XML formátu.

Důležité charakteristiky:

- *platformová nezávislost* – klient a server mohou být zcela rozdílné systémy
- *nezávislost na jazyku* – přenosovým formátem SOAP je jednoduché XML, které není složité generovat
- *pro komunikaci používá HTTP protokol* – může využít i jiné protokoly, ale HTTP je nejběžnější
- *zabezpečení* – podporuje zabezpečení pomocí hlavičky soap požadavku. Využití je například podepsání požadavku certifikátem klienta.
- *specifikace pomocí WSDL* – popisuje dostupné služby, jejich zabezpečení a podobu požadavků a odpovědí

Nevýhody:

- *závislost na XML* – jeho zpracování stojí výpočetní čas

Má volba:

Rozhodl jsem se pro REST. Funkce centrálního uložení dat přesně odpovídají specifikaci RESTového API. Navíc klientů v tomto případě bude více a jednoduchá implementace pro konzumaci RESTového API mi pomůže při jejich rychlejší vývoji. Nutno zmínit, že REST API se stává internetovým standardem a dříve převládající SOAP je nyní na ústupu.

Pro návrh REST API jsem se rozhodl použít RAML. Jedná se nový jazyk založený na formátu YAML, který v říjnu 2014 přešel do verze 1.0. Díky YAML je výsledný kód dobře čitelný. Pro vývoj jsem použil webové rozhraní od firmy MuleSoft¹.

3.3.1.2 Java

Volba programovacího jazyka byla pro mě jasná: *Java*.

Pro vývoj aplikace jako webové služby založené na RESTovém API můžeme využít celou škálu programovacích jazyků. Od nejběžnějšího PHP/ Python/ ASP, přes Javu k méně známým jako Ruby a další...

Java je perspektivní, velice rozšířený a univerzální jazyk s rozsáhlou komunitou. Komunita, velké množství frameworků a další rozšíření dělají z Javy, tak silný programovací jazyk, kterým bezesporu je.

Přesto má volba pro Javu vzešla hlavně z mého zaměstnání a potřeby se v tomto programovacím jazyce zdokonalit.

3.3.1.3 Spring

Nyní hledám framework, který mi umožní rychle implementovat REST API. Ve světě Javy na podobné otázky existuje vždy více možných odpovědí. Podívejme se na hlavní možnosti:

¹www.mulesoft.com

- *JAX-RS*² – REST API od Javy
- *Jersey*³ – implementace REST API založená na JAX-RS
- *Spring MVC*⁴ – vlastní implementace z rodiny Spring

Všechny řešení podobný postup. Pomocí anotací ovlivňují chování tříd a method pro příjem/ odeslání zpráv v kontextu REST API.

Rozhodl jsem se pro Spring. Hlavním důvodem byl rozsah celého Spring frameworku. Díky jeho univerzálnosti, snadné konfigurovatelnosti a dobré dokumentaci je použití Springu a Javy téměř standardem.

Méně častou volbou je mé rozhodnutí pro rozšíření Spring boot⁵. Slouží pro vytváření stand-alone Spring aplikací a zlehčuje základní konfiguraci aplikace. Pozitivním efektem je také odstranění XML konfigurací.

3.3.1.4 Hibernate

Pro persistenci dat jsem zvolil framework Hibernate⁶.

Používá ORM, což je způsob, kdy je objektový model mapován na relační databázi. Programátor pomocí anotací (nejčastěji nad třídami modelu) definuje podobu databázových objektů, ale o samotnou správu na úrovni databáze se už nestará. Hibernate také zajišťuje nezávislost na použité databázi. Nicméně např. pro Oracle databázi je potřeba použít speciálních anotací (databáze nemá auto inkrementální sloupce a pracuje přes definici sekvencí hodnot).

Použití Hibernate je Springem přímo podporováno a jeho konfigurace je velice snadná.

3.3.1.5 MySQL

Jako databázi jsem si zvolil MySQL.

Jedná se o relační databázi, která si v současnosti získává obrovskou oblibu. Je to díky její nenáročnosti, jednoduchosti, výkonnosti a stabilitě. Vlastníkem je firma Oracle, vydávající také databázi Oracle DB, která už vyžaduje specializovanou správu a není pro tento projekt vhodná. Pro administraci databáze jsem zvolil phpMyAdmin. A to díky mým dřívějším zkušenostem s jeho používáním.

²jax-rs-spec.java.net

³jersey.java.net

⁴<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>

⁵projects.spring.io/spring-boot/

⁶hibernate.org

3.3.1.6 Aplikační server Wild Fly

Jako aplikační server jsem zvolil WildFly⁷ ve verzi 8.2.

Jedná se pokračování známého JBoss AS[6].

Potřebné vlastnosti

- *rychlá instalace*
- *funkce auto deploy* – zkopírování waru⁸ aplikace do určité složky serveru dojde k jejímu automatickému nasazení
- *administrátorské rozhraní*
- *dokumentace* – díky jeho rozšíření existuje velké množství návodů
- *podpora Java EE*⁹

3.3.1.7 Hosting - DigitalOcean

Pro hosting jsem zvolil služby DigitalOcean¹⁰.

Jako další možnost byl server OpenShift¹¹. Ten poskytuje pevně dané aplikace, které je možné na virtuální server nainstalovat. Instalace serveru probíhá formou konfigurace a je díky tomu velmi snadná. Bohužel omezení systémových prostředků u neplacené verze vedlo k nespolehlivosti mého serveru a hledání jiné varianty.

DigitalOcean poskytuje tzv. droplet. Funguje jako virtuální server a správce si volí OS, který je něm automaticky nainstalován. Další instalace se už provádějí přímo ve vybraném OS. Mou volbou byl Ubuntu verze 14. Jde jednoduchý a spolehlivý linuxový systém se kterým mám jen kladné zkušenosti.

DigitalOcean je placený, ale v rámci GitHub Education packu¹² je pro studenty k dispozici 100 dolarový kapitál. To stačí na mnohaměsíční provoz i silnějších variant dropletů s více systémových prostředků. Pro naši aplikaci jsem zvolil variantu 2 GB paměti, 2 CPU, 40 GB SSD disk za 20 dolarů na měsíc.

3.3.2 Mobilní aplikace pro Android

3.3.2.1 OS Android

Rozhodl jsem se svou mobilní aplikaci implementovat pro OS Android.

Vybíral jsem mezi OS Android do firmy Google¹³ a iOS od firmy Apple¹⁴.

⁷wildfly.org

⁸<http://docs.oracle.com/javaee/6/tutorial/doc/bnaby.html>

⁹<http://www.oracle.com/technetwork/java/javaee/overview/index.html>

¹⁰www.digitalocean.com

¹¹www.openshift.com

¹²education.github.com/pack

¹³www.android.com

¹⁴www.apple.com/cz/ios/

3. NÁVRH

Celkové množství aplikací je u obou OS téměř stejné a ani náročnost vývoje neznamena výrazné rozdíly.[7]. Rozdíl programovacích jazyků (Android:Java a iOS:Objective-C) pro mě znamená důležitý faktor. U předešlé aplikace jsem se rozhodl pro Javu a díky Androidu mohu zůstat ve stejném programovacím jazyce.

Trh Android má silnou převahu v zastoupení na světovém trhu. Zatímco iOS si drží 15% podíl, Android se pohybuje na 80% [8]. Silnější pozice dosahuje iOS na trhu v U.S., kde můžeme zastoupení obou OS považovat za vyrovnané [9]. Pro mě je důležité, že Android při stejném množství aplikace poskytuje širší uživatelskou základnu.

Závěrem analýzy existujících implementací osobního plánování je důraz na podporu co nejširšího spektra platform. Z tohoto pohledu chápu mé rozhodnutí pro vývoj na OS Android jako volbu první mobilní platformy pro implementaci. A v rámci dalšího rozvoje projektu plánuji podporu pro další.

Shrnutí

- *Java* – mohu stavět na současných znalostech tohoto jazyka a využít stejné vývojové prostředí
- *zastoupení na trhu* – více uživatelů mi dává vyšší šance si najít své zákazníky
- *osobní preference* – vlastním mobil s OS Android

3.3.2.2 Android-bootstrap

Pro implementaci jsem se rozhodl použít framework Android-bootstrap od Donna Felkera ¹⁵.

Framework představuje implementaci základních funkcí aplikace pro OS Android. Používá se jako základ při vytváření aplikace a umožňuje rychlý vývoj. Zdrojové kódy jsou přístupné na GitHubu, a autor se společně s komunitou podílí na dalším rozvoji.

Obsahuje

- *napojení na REST API*
- *implementace přihlašování*
- *stránkování jako hlavní komponenta uživatelského rozhraní* – snadné rozšíření o další stránky
- *vlastní implementace zobrazení seznamu dat* – načtení dat v asynchronním vlákně, specifikace zobrazení pomocí vlastního layoutu¹⁶
- *jednotný grafický vzhled*

Nevýhoda je nárok na verzi Android SDK 15 a vyšší.

¹⁵<http://www.androidbootstrap.com/>

¹⁶<http://developer.android.com/guide/topics/ui/declaring-layout.html>

3.3.2.3 Verze SDK Android

Rozhodl jsem pro minimální verzi SDK 16 (Android verze 4.1 s označením Jelly bean) a cílovou verzi SDK 19 (Android verze 4.4 s označením Kitkat). Toto rozhodnutí vzešlo z nároků mnou zvolené technologie a současného stavu trhu.

Použitím frameworku Bootstrap (viz 3.3.2.2) jsem stanovil minimální verzi SDK na 15 a vyšší. V rámci vývoje jsem následně použil funkce vyžadující verzi SDK 16+.

Autoři Androidu doporučují volbou SDK pokrýt 90% aktivních zařízení[10]. V aktuálním zastoupení SDK vede verze 19 a suma od verze 4.1 výše je cca 88%[11].

3.3.2.4 Aplikace

- java potřebuje aplikační server
- potřebujeme webový server
- ideálně na jednom místě

3.3.2.5 openshift

- jednoduché ovládání bez nutnosti ruční instalace
- zdarma
- ale ukázalo se jako nestabilní

3.3.2.6 Digiocean

- studentský pack s 100 dolary pro založení a užívání server
- slouží jako virtuální server na který si uživatel vše potřebné nainstaluje
- složitější instalace
- funkční

Návrh

Krátká kapitola pouze zmíním základní

4.1 REST API

tabulky volání

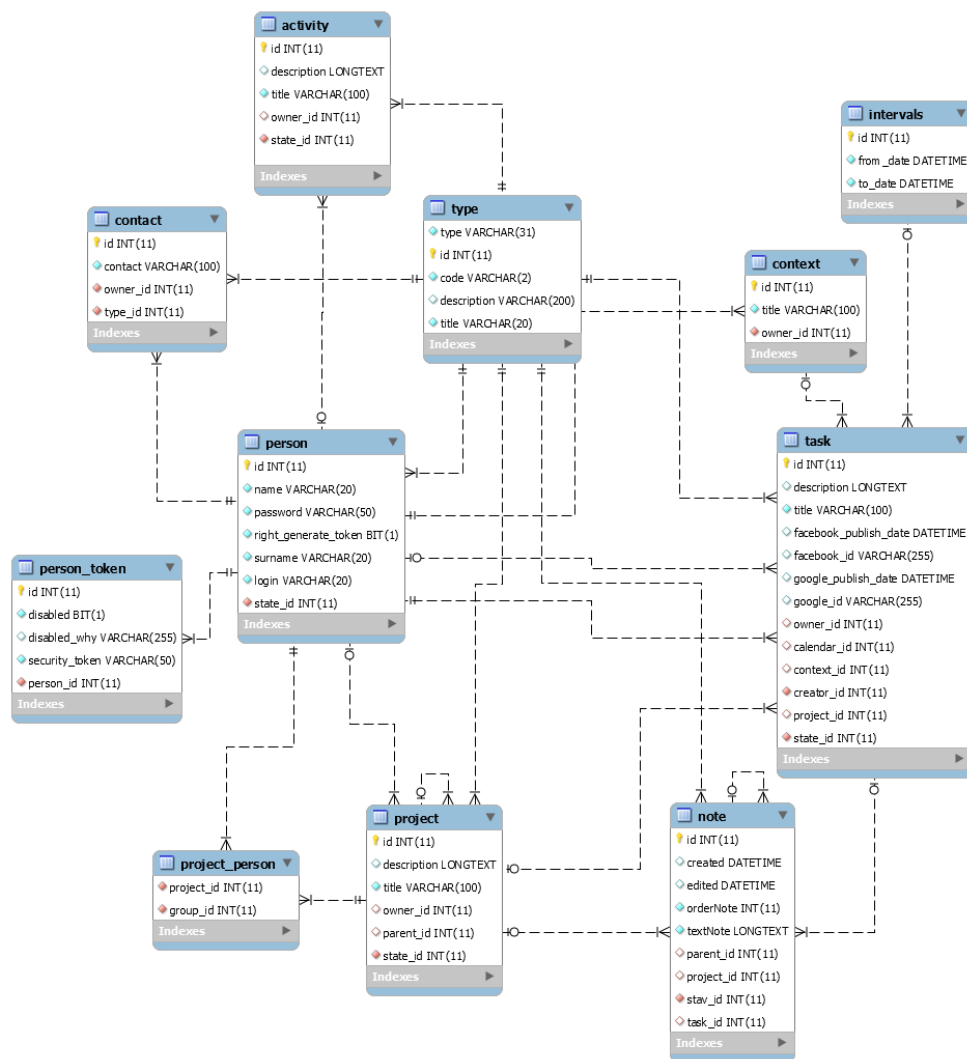
- způsob zabezpečení pomocí tokenu
- SOAP UI jako testy řízený vývoj?

4.2 Datový model

lehce popsat

- středobot jsou tasky person token

4. NÁVRH



Obrázek 4.1: Data model (generováno pomocí MySQL Workbench)

)

Realizace

5.0.1 Aplikace pro centrální uložení dat

5.0.1.1 Rozložení projektu

–složky a k čemu slouží

5.0.1.2 REST API

ukázka kontroleru

odchycení výjimek

ukázka servisy pro obsluhu volání (např add person)

5.0.1.3 Hibernate

práce s DAO.

5.0.1.4 Facebook

práce s access tokenem

vypublikování tasku na zeď uživatele

5.0.1.5 Google

práce s access tokenem

vypublikování tasku do google kalendáře

5.0.2 Mobilní aplikace pro Android

5.0.2.1 Rozložení projektu

–složky a k čemu slouží

5.0.2.2 Konzumace REST API

- konzumace rest api

5.0.2.3 Asynchronní načítání dat

- ukázka takového volání
 - jak se při získání/ nezískání výsledku
 - Bus listenery

5.0.2.4 Facebook

- jak založí aplikaci na facebooku a správně ji nastavit
 - nastavení projektu
 - získání access tokenu přihlášením uživatele

5.0.2.5 Google

- jak založí aplikaci na facebooku a správně ji nastavit
 - nastavení projektu
 - získání access tokenu přihlášením uživatele

5.1 Hosting

- postup instalace
- správa

5.2 GitHub

- zpřístupnění kódu na GitHubu ano?

5.2.1 title

Závěr

Metodika *GTD* do mého života přinesla nový směr a podle něj se snažím řídit. Mysl potřebujeme použít k řešení problému a ne jako diář, který ještě funguje velmi podivně a události nám připomíná velmi nahodile. Diář si musíme vytvořit mimo ni. Na internetu nám k tomu pomůže celá řada produktů. Je jen na nás, který si vybereme. A volba je důležitá. Protože metodika nás nabádá, že musíme mít naprostou důvěru ke spolehlivosti svého systému a používat ho rádi. Plánování a revize úkolů se musí stát pravidelnou činností.

Mnou navržený systém se ukázal jako vhodný.

Framework Spring, který jsem použil pro aplikaci poskytující centrální uložení dat, mi pomohl k vytvoření dobře škálovatelné a udržitelné aplikace. Vybrané REST API dostalo mým předpokladům, pro které jsem ho vybral. Implementace není náročná na straně serveru ani klienta a plně vyhovuje potřebám aplikace. Stejně tak má volba Hibernatu pro ORM řešení mě dobře odstínila od databázového prostředí. Zde se ukazuje síla anotací. Pomocí nich lze provádět širokou škálu konfigurací bez nutnosti externích konfiguračních souborů. Proto jsem pomocí anotací konfiguroval samotnou aplikaci a zde mi práci ulehčil vybraný Spring Boot.

Pro hosting jsem zvolil server DigiOcean. Zde jsem vytvořil vlastní Linux server, nainstaloval a nakonfiguroval aplikační server WildFly a databázi MySQL s phpAdminem a nasadil REST API aplikaci.

Mobilní aplikaci jsem vyvinul a testoval na lokálním emulátoru Genymotion s instalovanou verzí Androidu 19. Volba Bootstrapu byla výborná a pro účely mé aplikace plně vyhovující. Framework poskytuje dostačující infrastrukturu a zároveň je sám založen na další knihovnách usnadňujících vývoj.

Nastudoval jsem specifikace API Facebooku a Google kalendáře. Umožňují uživateli publikovat úkol na jeho vlastní zeď Facebooku. Stejně může publikovat úkol do Google kalendáře. Zde jsem správně navrhl oddělit funkci publikování na dvě části a to získání přístupových práv a samotné publikování. Přístupová práva získává Android aplikace a k tomu využívá komponenty přímo

od Facebooku a Googlu. Správně jsem analyzoval, že proces nelze automatizovat, ale vytvořením přístupového tokenu lze následné publikování delegovat na jinou aplikaci. O samotné publikování se stará REST API aplikace.

Práce s API Facebook i Googlu pro mě znamenala cennou zkušenost a vidím v něm velký potenciál dalšího rozvoje. V moderním internetovém prostředí už nemůže aplikace existovat izolovaně, ale je třeba ji integrovat do existujícího prostředí. A tato integrace nemá být pouze pasivní, ale také aktivní.

Literatura

- [1] Allen, D.: *Mít vše hotovo - Jak zvládnout práci i život a cítit se při tom dobře*. Brno: Jan Melvil Publishing, s.r.o., 2008, ISBN 978-80-903912-8-4.
- [2] Gregor, L.: Stručný průvodce po metodě GTD. [online], 2008, [cit. 2015-04-29]. Dostupné z: <http://www.mitvsehotovo.cz/2008/08/strucny-pruvodce-po-metode-gtd/>
- [3] toodledo: weird name, solid app. [online], 2012, [cit. 2015-04-29]. Dostupné z: <http://purplezengocat.com/2012/04/06/toodledo-weird-name-solid-app/>
- [4] Saunders, G. J. M.: Wunderlist — UI peculiarities. [online], 2013, [cit. 2015-04-29]. Dostupné z: <http://blog.garethjmsaunders.co.uk/tag/wunderlist/>
- [5] Thin client. [online], 2015, [cit. 2015-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Thin_client
- [6] What is WildFly? [online], 2015, [cit. 2015-04-29]. Dostupné z: <http://wildfly.org/about>
- [7] Hamlyn, M.: Should You Develop for Android, iOS or both? [online], 2014, [cit. 2015-04-29]. Dostupné z: <http://www.appmakr.com/blog/develop-android-ios/>
- [8] Comparison of mobile operating systems. [online], 2015, [cit. 2015-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems
- [9] McCracken, H.: Who's Winning, iOS or Android? All the Numbers, All in One Place. [online], 2013, [cit. 2015-04-29]. Dostupné z: <http://techland.time.com/2013/04/16/ios-vs-android/>

LITERATURA

- [10] Supporting Different Platform Versions. [online], 2015, [cit. 2015-04-29]. Dostupné z: <http://developer.android.com/training/basics/supporting-devices/platforms.html>
- [11] Dashboards. [online], 2015, [cit. 2015-04-29]. Dostupné z: <http://developer.android.com/about/dashboards/index.html>

Seznam použitých zkratk

Obsah přiloženého CD

	readme.txt.....	stručný popis obsahu CD
	exe	adresář se spustitelnou formou implementace
	src	
	impl.....	zdrojové kódy implementace
	thesis	zdrojová forma práce ve formátu L ^A T _E X
	text	text práce
	thesis.pdf	text práce ve formátu PDF
	thesis.ps	text práce ve formátu PS