Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

GTD - Mít vše hotovo

Michal Sláma

Vedoucí práce: Ing. Jiří Mlejnek

10. 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 10. května 2015 .....

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 appliaction, personal planning, spring, android, REST API, Facebook, Google Calendar

Obsah

Seznam obrázků

Seznam tabulek

Todo list

Co je obsaženo v práci, nikoliv na čem budu dělat

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 přístupu 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.

Kapitola 1

Cíl práce

První cílem mé práce je analyzovat metodiku *GTD* [1]. Prozkoumám existující implementace osobního plánování podporující *GTD* [1]. Navrhnu 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 sítě. Proto umožním uživateli publikovat své úkoly do sítí *Facebooku* a *Google kalendáře*.K tomu nastuduji potřebné specifikace poskytovaných API

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

Kapitola 2

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 změníme tím svůj život. Snaží se vysvětlit obecné zákonitosti lidské mysli a najít takových způsob 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 Ducken[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

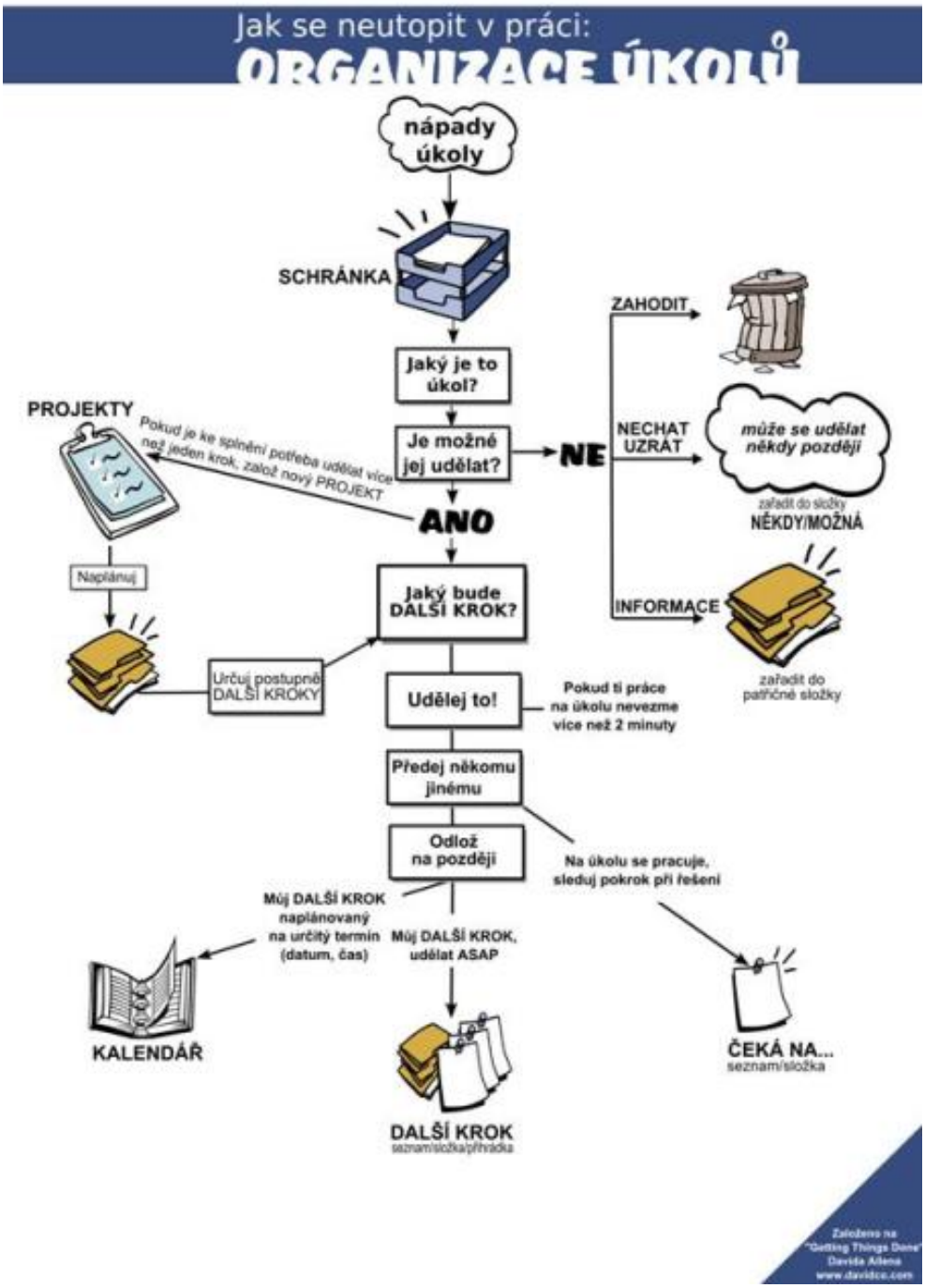
Začít je třeba od nezákladnější věcí. To nám pomůže zvládnout a odpoutat se od každodenních závazků a poté se můžeme lépe soustředit na větší projekty a vize. Při zpracování potřebujeme využít kontroly ve dvou rovinách: horizontální a vertikální. V horizontální přemýšlíme, co všechno potřebujeme udělat.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

Typický horizontální postup:
1. sběr věcí, které si získaly naší 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]

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 kalendáře

   Po 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 úkolem 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

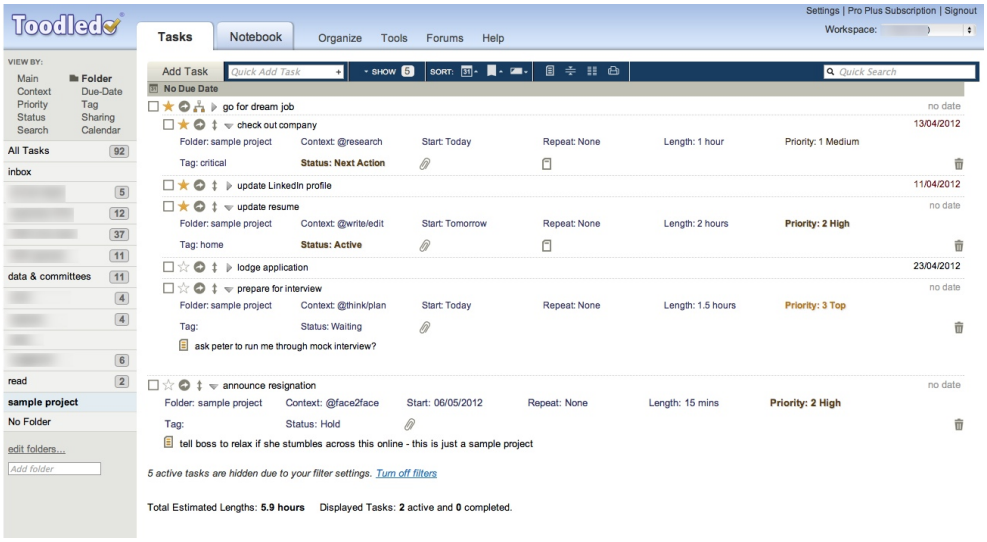
2.2.1 Toodledo

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.



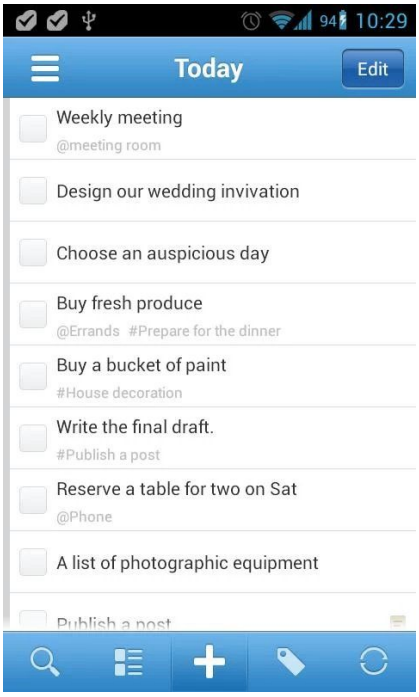
Obrázek 2.2: Podoba webové aplikace ToodleDo.[3]

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.



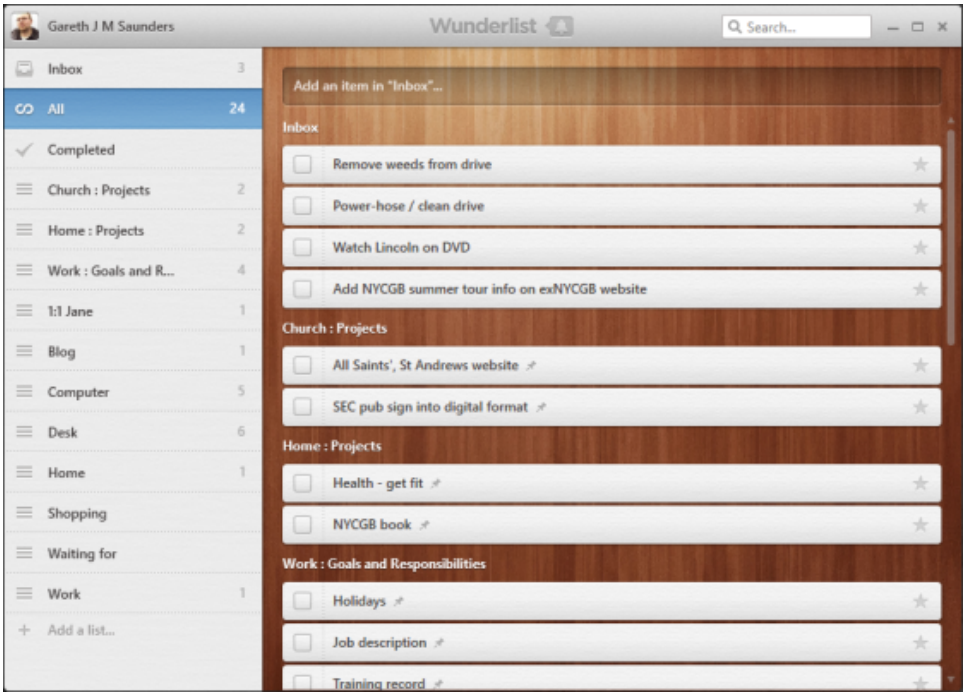
Obrázek 2.3: Mobilní aplikace Doit.[**?**]

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.



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 verzí. Pokud chceme jednoduché ovládání bez další funkcí.

Nevýhody: Nesplňuje národy *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 existující, 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 součásti placené verze.

Implementace bude obsahovat

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

Kapitola 3

Návrh

Tato kapitola popisuje volbu technologií pro implementovanou aplikaci poskytující centrální uložení dat skrze WS a aplikaci pro OS Android. Nároky na technologie vycházejí ze závěrů analýzy předešlé kapitoly a dále je rozvíjejí. Protože jedna z aplikací potřebuje hosting, budeme hledat vhodný server splňující naše potřeby.

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.

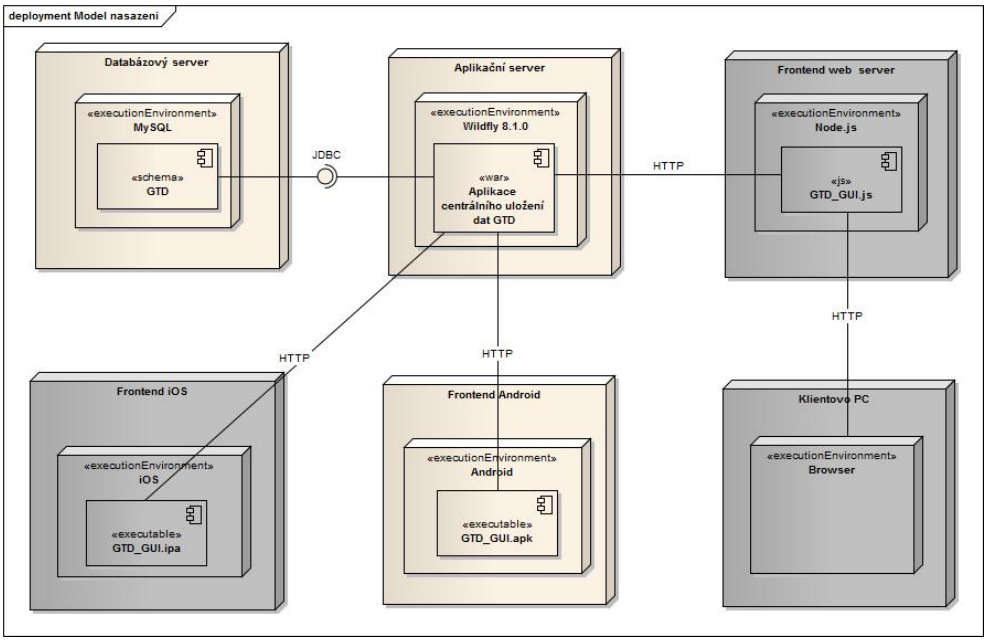
3.1.2 Mobilní aplikace pro Android

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 kaž-

Komentář, že se jedná o navazující projekt z SP2



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

)s

dou platformu s centrální aplikací pro správu data komunikují přes HTTP protokol.

3.3 Technologie

3.3.1 Aplikace pro centrální uložení dat

3.3.1.1 Webová služba – REST API

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

- *SOAP*
- *REST*

Nyní se soustředím na hlavní rozdíly mezi těmito službami.

link na zabezpeceni

Návrh nepočítá s využitím zabezpečení (přihlášení bude řešeno pomocí tokenu) a komunikace bude probíhat přes HTTP protokol (SOAP ztrácí výhodu). Myšlenkou RESTu je poskytnout CRUD[14] operace nad vystavenými zdroji[16] a to přesně odpovídá potřebám navrhovaného centrální uložení dat.

Tabulka 3.1: Rozdíly mezi SOAP a REST

Typ	SOAP	REST
Format dat[6]	XML	nezávislý (XML, JSON, plain)
Protokol[6]	více možných (HTTP, SMTP,..)	HTTP
Možnost cachování[6]	ne	ano
Formát specifikace[7][8]	WSDL[9]	existují formáty jako WADL[10], Swagger[11], RAML[12] a další, ale žádný nemůžeme považovat na hlavní [13]
Vystavuje[14]	operace	zdroje
Zabezpečení[14]	SSL + WS-Security	SSL
Rozšířenost[14]	REST získává na stále větší popularitě . Příkladem uvedu např. Google, který se nahradil své dřívější SOAP API právě RESTem.	
Rychlost zpracování[15]	XML parsing je pomalejší než JSON	
Jednodušší implementace[6]	u RESTu na straně klienta i serveru	

Další výhody jako rychlá implementace a přenos dat ve formátu JSON je podpořily volbu.

Volba pro WS je REST.

Pro návrh REST API bude použit jazyk RAML. Jedná se nový jazyk založení na formátu YAML, který v říjnu 2014 přešel do verze 1.0[12] a na jehož vývoji se podílí technologičtí odborníci ze známých IT firem[17]. Díky YAML se kód dobře píše a výsledný dokumentace je čitelná. Pro vývoj bude použito webového prostředí od firmy MuleSoft[18]. Jedná se o firmu, která se na RAML úzce propojena, protože na specifikaci RAML se podílí její CTO[1].

tabulka rest api

3.3.1.2 Programovací jazyk – 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...

[1] www.linkedin.com/in/sarid

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 Základní framework – 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:

- *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 boot[5]. Slouží pro vytváření stand-alone Spring aplikací a zlehčuje základní konfiguraci aplikace. Pozitivním efektem je také odstranění XML konfigurací.

3.3.1.4 Persistence dat – Hibernate

Pro persistenci dat jsem zvolil framework Hibernate[6].

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.

[2] jax-rs-spec.java.net
[3] jersey.java.net
[4] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html
[5] projects.spring.io/spring-boot/
[6] hibernate.org

3.3.1.5 Databáze – 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.

3.3.1.6 Aplikační server Wild Fly

Jako aplikační server jsem zvolil WildFly[7] ve verzi 8.2.

Jedná se pokračování známého JBoss AS[19].

Potřebné vlastnosti

- *rychlá instalace*
- *funkce auto deploy* – zkopírování waru[8] 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 DigitalOcean[10].

Jako další možnost byl server OpenShift[11]. Ten poskytuje pevně dané aplikace, které je možné na virtuální server nainstalovat. Instalace serveru probíhá formou konfigurace a je díku 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 packu[12] je pro studenty k dispozici 100 dolarový kapitál. To stačí na mnohaměsíční provoz i silnějších variant dropletů s více systémových prostředků. Pro naši aplikaci

[7] wildfly.org
[8] http://docs.oracle.com/javaee/6/tutorial/doc/bnaby.html
[9] http://www.oracle.com/technetwork/java/javaee/overview/index.html
[10] www.digitalocean.com
[11] www.openshift.com
[12] education.github.com/pack

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 Google[13] a iOS od firmy Apple[14].

Celkové množství aplikací je u obou OS téměř stejné a ani náročnost vývoje neznamená výrazné rozdíly.[20]. 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% [21]. Silnější pozice dosahuje iOS na trhu v U.S., kde můžeme zastoupení obou OS považovat za vyrovnané [22]. 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 platforem. Z tohoto pohledu chápu mé rozhodnutí pro vývoj na OS Android jako volbu první mobilní platformy pro implementaci. A v rámci další rozvoje projektu plánuji podporuji 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 dodává 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 podílejí na dalším rozvoji.

Obsahuje

[13] www.android.com
[14] www.apple.com/cz/ios/
[15] http://www.androidbootstrap.com/

- *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 asychronním vlákně, specifikace zobrazení pomocí vlastního layoutu[16]
- *jednotný grafický vzhled*

Nevýhodou je nárok na verzi Android SDK 15 a vyšší.

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[23]. V aktuálním zastoupení SDK vede verze 19 a suma od verze 4.1 výše je cca 88%[24].

3.3.2.4 Facebook

Facebook

3.3.2.5 Google

Google navrh

[16] http://developer.android.com/guide/topics/ui/declaring-layout.html

Kapitola 4

Návrh

Tato kapitola se zaměřuje na definování základních charakteristik aplikace. Najdete zde návrh specifikace REST API a webového vývojového prostředí od společnosti Mulesoft, které je k návrhu použito.

Kapitola dále popisuje použitý datovým model.

Nakonec se seznámíte s postupem pro propojení mobilní aplikace na Facebook a Google kalendář. Přidělíme každé aplikaci oddělené úkoly. Mobilní aplikace zajistí přihlášení uživatele a získání access tokenu. Tento token předáme aplikaci pro centrální uložení dat a ta zajistí samotnou práci s API protistrany.

4.1 REST API

Budeme pracovat s těmito čtyřmi metodami HTTP 4.1:

Tabulka 4.1: Funkce HTTP method

HTTP metoda	**Popis**
GET	získá informace daného zdroje
POST	vytvoří nový zdroj
PUT	aktualizuje daný zdroj
DELETE	smaže daný zdroj

Navržené REST API rozhraní je zobrazeno v tabulkách 4.2 a 4.2 (* – představuje společnou URL adresu určenou nasazením a verzí API).

Na základě volby v 3.3.1.1 vznikl návrh REST API ve webovém vývojovém prostředí Mulesoft.

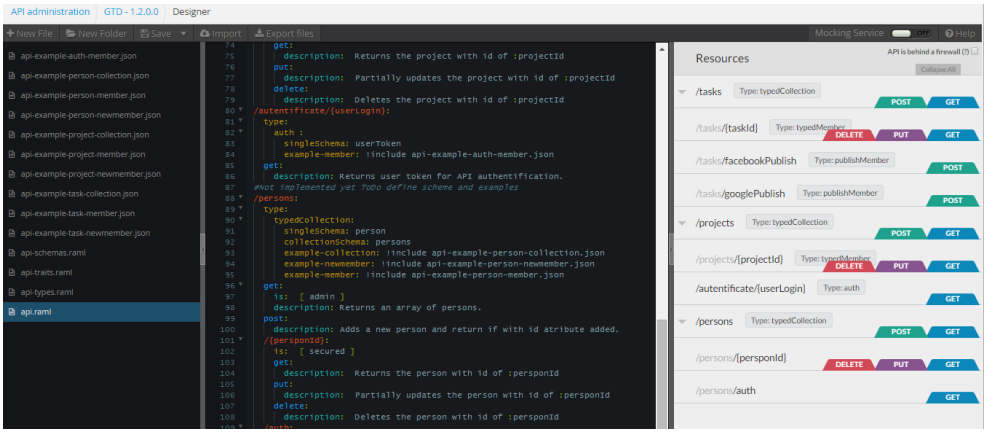
Tabulka 4.2: Rozhraní REST API přístupné s tokenem

Zdroj URI	**HTTP metoda**	
1	*/projects/	POST
2	*/projects/	GET
3	*/projects/{id}	GET
4	*/projects/{id}	PUT
5	*/projects/{id}	DELETE
6	*/tasks/	POST
7	*/tasks/	GET
8	*/tasks/{id}	GET
9	*/tasks/{id}	PUT
10	*/tasks/{id}	DELETE
11	*/tasks/{id}/facebookPublish	POST
12	*/tasks/{id}/googlePublish	POST
13	*/persons/	GET
14	*/persons/{id}	GET
15	*/persons/{id}	PUT
16	*/persons/{id}	DELETE
17	*/persons/auth	GET

Tabulka 4.3: Rozhraní REST API přístupné bez tokenu

Zdroj URI	**HTTP metoda**
*/authenticate/{userLogin}	GET
*/persons	POST

Přístupné přes:	Mulesoft API Designer
Příhlašovací jmeno:	drugnanov
Heslo:	Qazwsxedc369



Obrázek 4.1: Webové prostředí pro návrh REST API ve specifikaci RAML od firmy Mulesoft

Práce s tímto prostředím je příjemná. Některé funkce potřebují ještě vylepšit. Chybí inline nápověda v kontextu specifikace RAML a např. auto complete nepracuje s objekty vytvořenými uživatelem. Zároveň příkladů návrhu API v jazyku RAML na webu není mnoho. To bohužel prodlužuje čas učení a samotného návrhu. Přesto má tento nástroj vykročeno stát se součásti DDD[25].

4.2 Databázový model

V 3.3.1.4 jsem se rozhodli pro použití Hibernatu. To nám umožňuje specifikovat strukturu databáze přímo v datovém modelu aplikace. Specifikace se provádí pomocí anotací nad objekty tříd, metod a atributů[26].

Kapitola 5

Realizace

V této kapitole se seznámíme s praktickým použitím dříve zvolených technologií. U aplikace pro centrální uložení dat si ukážeme strukturu projektu, podobu práce s REST API, generování struktury databáze pomocí hibernatu, identifikace uživatele s použitím tokenu a práci s API Facebooku a Google kalendáře. U android aplikace si uk

co s ukážeme u android aplikace

5.0.1 Aplikace pro centrální uložení dat

Zde si ukážeme části implementace aplikace pro centrální uložení dat.

5.0.1.1 Struktura projektu

Struktura projektu respektuje Java konvence

- `libs` ..... externi knihovny používané aplikací
- `src` ..... základní balík implementace projektu
  - `main` ..... balík aplikace
    - `java` ..... složka s kódy aplikace
      - `GTD` ..... základní balík GTD
        - `BL` ..... balík buisiness vrstvy
        - `DL` ..... balík datové vrstvy
        - `restapi` ..... balík pro REST API
    - `resources` ..... složka obsahující konfigurace
      - `database` složka obsahující pomocné skripty pro správu databáze
      - `messages` ..... složka pro stringové konstanty aplikace
      - `gtd.api.properties` ..... property aplikace
      - `hibernate.cfg.xml` ..... konfigurace hibernatu
      - `logback.xml` ..... konfigurace logování pomocí logbacku
  - `test` ..... složka obsahující unit testy
- `build.gradle` ..... build soubor pro Gradle
- `settings.gradle` ..... nastavení Gradlu

5.0.1.2 REST API

ukázka kontroleru

odchycení výjímek

ukázka servisy pro obsluhu volání (např add person)

5.0.1.3 Databáze

Příklad použití anotací na základě 5.1.

Listing 5.1: Příklad použití Hiberante

```
@Entity
@Table(name = "person")
@JsonSerialize(include = JsonSerialize.Inclusion.NON_NULL)
@JsonIgnoreProperties(ignoreUnknown = true)
public class Person {

    public static final int MAX_LENGTH_FIRST_NAME = 20;
    public static final int MAX_LENGTH_LAST_NAME = 20;
    public static final int MAX_LENGTH_LOGIN = 20;
```

```
    public static final int MAX_LENGTH_PASSWORD = 50;

    @Id
    @GeneratedValue
    private int id;

    @Column(length = MAX_LENGTH_FIRST_NAME, nullable = false,
        name = "name")
    private String name;

    @Column(length = MAX_LENGTH_PASSWORD, nullable = false)
    private String password;

    @Transient
    private String passwordRaw;

    @Column(nullable = false, name = "right_generate_token")
    @JsonIgnore
    private Boolean rightGenerateToken;

    @OneToMany(mappedBy = "person", cascade =
        CascadeType.ALL, fetch = FetchType.EAGER)
    private List<PersonToken> tokens;

    @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL,
        fetch = FetchType.EAGER)
    private List<Contact> contacts;

    @Column(length = MAX_LENGTH_LOGIN, nullable = false,
        unique = true, name = "login")
    private String username;

    @Column(length = MAX_LENGTH_LAST_NAME, nullable = false,
        name = "surname")
    private String surname;

    @ManyToOne
    @JoinColumn(nullable = false, name = "state_id")
    @JsonProperty(value = ApiConstants.STATE)
    private PersonState state;

    @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL)
    @JsonIgnore
    private List<Project> projects;

    @OneToMany(mappedBy = "owner", cascade = CascadeType.ALL)
    @JsonIgnore
    private List<Context> contexts;

    ....
}
```

Význam anotací dle [26].

Doplněním anotací k prvkům datového modelu a konfigurací hibernatu byla v MySQL vygenerována následující databázová struktura 5.1.

5.0.1.4 Identifikace uživatele

Identifikace uživatele je realizována formou tokenu v hlavičce HTTP požadavku. Token lze získat pomocí dvou způsobů (viz. 4.3):

- *registrací* – po vytvoření účtu je ve vráceném objektu *osoby* obsažen také token
- *přihlášení* – autentifikace pomocí loginu a hesla uživatele, v takovém případě je zpět vrácen pouze token a login uživatele.

Při vytvoření uživatele je token vygenerován automaticky. V případě přihlášení je vrácen aktivní token uživatele. Pokud uživatel token nemá (např. token byl deaktivován) a má právo si token vytvořit, je mu automaticky vygenerován.

Pokud uživatel už aktivní token má, může použít dotaz číslo 17 z **??**. API na základě tokenu rozpozná *osobu* a vrátí její objekt *osoby* obsahující všechny tokeny.

Token je generován pomocí kódu v 5.2 (person je objekt typu Person).

Listing 5.2: Generování uživatelského tokenu

```
String stringToCrypt = person.toString() +
    PersonToken.tokenSalt + currentTimeMillis();
String hash = HashConverter.md5(stringToCrypt);
```

5.0.1.5 Facebook

práce s access tokenem

vypublikování tasku na zeď uživatele

5.0.1.6 Google

práce s access tokenem

vypublickování tasku do google kalendáře

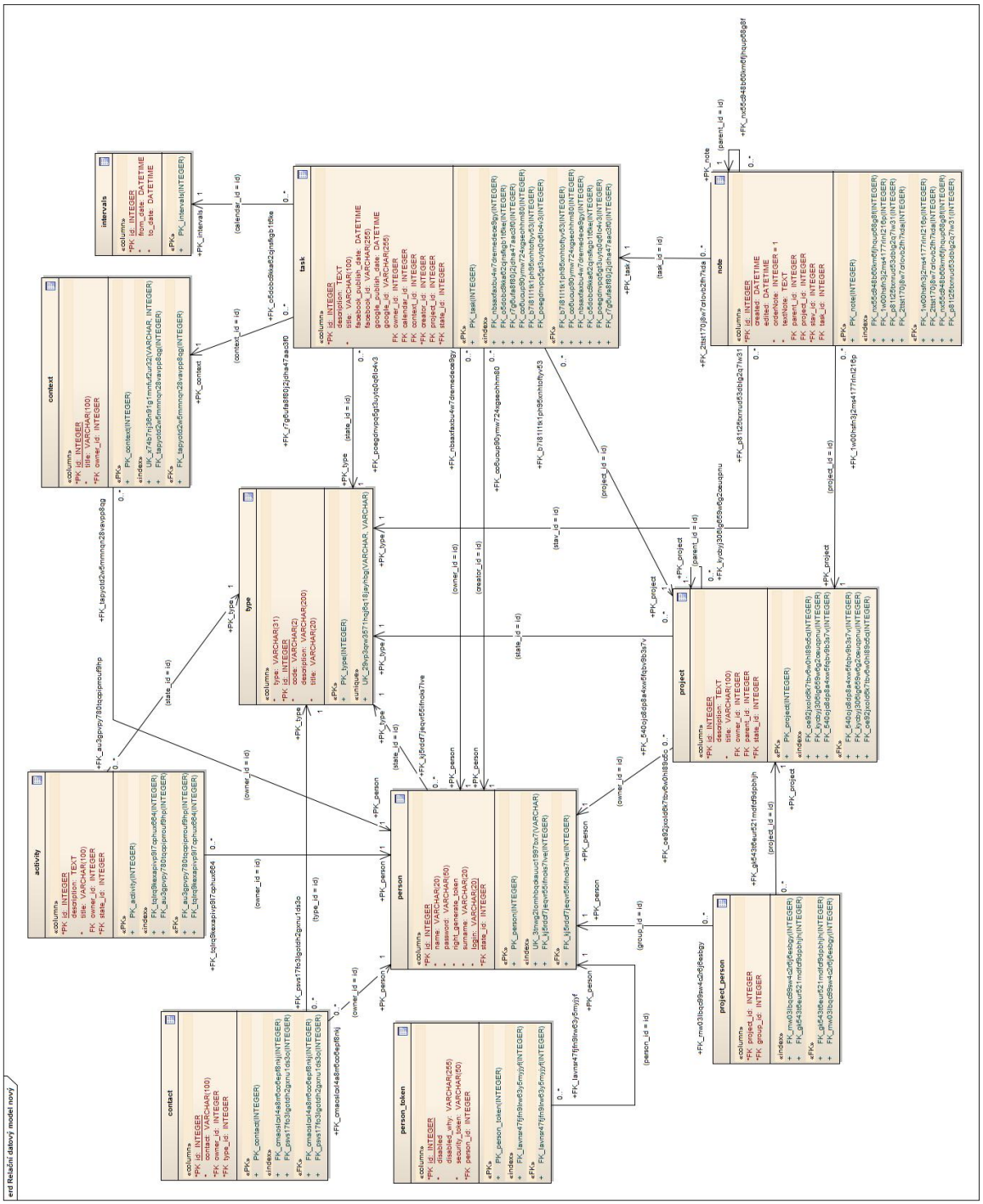
5.0.2 Mobilní aplikace pro Android

5.0.2.1 Rozložení projektu

–složky a k čemu slouží

5.0.2.2 Konzumace REST API

– konzumace rest api



Obrázek 5.1: Databázový model

5.0.2.3 Asynchronní načítání dat

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

5.0.2.4 Facebook

– jak zalozi aplikaci na facebooku a spravne ji nastavit
– nastaveneni projektu
– ziskani access tokenu přihlášením uživatele

5.0.2.5 Google

– jak zalozi aplikaci na facebooku a spravne ji nastavit
– nastaveneni projektu
– ziskani access tokenu přihlášením uživatele

5.1 Hosting

–postup instalace
–správa

5.2 GitHub

–zpřístupnění kódu na GitHubu ano?

AnyPoint

Program pro převod JSON do objedků d:01.Skola01.Bakalarka.30.Vyvoj BP GTD GeneratorPOJOs

5.2.1 title

5.3 Návrhy k rozvoji

Apple aplikace Testování na reaálných zařízeních a úprava kódu aby to fungovalo
refactoring Hibernate
Fungování s email načtením

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 udržovatelné aplikace. Vybrané REST API dostálo 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. Umožňuji 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.

na čem je nasazeno, kolik lidí to používá a další rozvoj

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://purplezengoat.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] Dhingra, S.: REST vs. SOAP: How to choose the best Web service. [online], 2013, [cit. 2015-05-05]. Dostupné z: `http://searchsoa.techtarget.com/tip/REST-vs-SOAP-How-to-choose-the-best-Web-service`

[7] Motamarri, J.: Compare RESTful vs SOAP Web Services. [online], 2014, [cit. 2015-05-05]. Dostupné z: `http://java.dzone.com/articles/j2ee-compare-restful-vs-soap`

[8] SOAP vs. REST Challenges. [online], 2014, [cit. 2015-05-05]. Dostupné z: `http://www.soapui.org/testing-dojo/world-of-api-testing/soap-vs--rest-challenges.html`

[9] Web Services Description Language (WSDL) 1.1. [online], 2001, [cit. 2015-05-05]. Dostupné z: `http://www.w3.org/TR/wsdl`

[10] Web Application Description Language. [online], 2015, [cit. 2015-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Web_Application_Description_Language

[11] Swagger. [online], 2015, [cit. 2015-05-05]. Dostupné z: http://swagger.io

[12] RESTful API Modeling Language. [online], 2015, [cit. 2015-05-05]. Dostupné z: http://raml.org

[13] API Spec Comparison Tool. [online], 2014, [cit. 2015-05-05]. Dostupné z: http://www.mikestowe.com/2014/12/api-spec-comparison-tool.php

[14] Francia, S.: REST Vs SOAP, The Difference Between Soap And Rest. [online], 2001, [cit. 2015-05-05]. Dostupné z: http://spf13.com/post/soap-vs-rest

[15] Nene, A.: Web Services Architecture – When to Use SOAP vs REST. [online], 2014, [cit. 2015-05-05]. Dostupné z: http://java.dzone.com/articles/web-services-architecture

[16] Create, read, update and delete. [online], 2015, [cit. 2015-05-05]. Dostupné z: http://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[17] RAML (software). [online], 2014, [cit. 2015-05-05]. Dostupné z: http://en.wikipedia.org/wiki/RAML_(software)

[18] Mulesoft. [online], 2014, [cit. 2015-05-05]. Dostupné z: www.mulesoft.com

[19] What is WildFly? [online], 2015, [cit. 2015-04-29]. Dostupné z: http://wildfly.org/about

[20] 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/

[21] Comparison of mobile operating systems. [online], 2015, [cit. 2015-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Comparison_of_mobile_operating_systems

[22] 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/

[23] Supporting Different Platform Versions. [online], 2015, [cit. 2015-04-29]. Dostupné z: http://developer.android.com/training/basics/supporting-devices/platforms.html

[24] Dashboards. [online], 2015, [cit. 2015-04-29]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[25] RAML: DDD (Delight-Driven Development) For APIs, Uri Sarid 20140716. [online], 2014, [cit. 2015-05-05]. Dostupné z: https://www.youtube.com/watch?v=_RI5iZgOtis

[26] Hibernate Annotations. [online], 2010, [cit. 2015-05-05]. Dostupné z: https://docs.jboss.org/hibernate/annotations/3.5/reference/en/html_single

Příloha A

Seznam použitých zkratek

Příloha **B**

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 LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS