Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového Inženýrství

Bakalářská práce

Systém správy úkolů pro jednotlivce a malé týmy

Martin Melka

Vedoucí práce: Ing. Josef Pavlíček, Ph.D.

21. března 2016

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

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 21. března 2016 .....

Odkaz na tuto práci

Melka, Martin. *Systém správy úkolů pro jednotlivce a malé týmy.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá srovnáním existujících aplikací a tvorbou nové aplikace pro správu úkolů. Uživateli této aplikace budou jednotlivci a menší pracovní skupiny, a přidělovat a sledovat průběh práce na společných úkolech. Aplikace umožní lidem sdružovat se do skupin, spolupracovat na sdílených úkolech a zaznamenávat odvedenou práci. Součástí této práce je definice požadavků na aplikaci, srovnání navrhovaného řešení s existujícími aplikacemi, dále návrh, implementace, testování a nasazení aplikace. Výsledkem práce bude aplikační backend, vystavující funkcionalitu skrze REST rozhraní.

Klíčová slova Správa úkolů, produktivita, organizace týmů, backend, REST, Java Spring Framework

Abstract

The aim of this thesis is to compare available software applications for task management and to subsequently create an original one. The users of this application will be individuals and small-scale workgroups, who need to assign responsibilities for and track the progress of shared tasks. The application will allow users to form groups, work together on shared tasks and report the

work done on them. This thesis consists of a definition of application requirements, comparison of current task management solutions and design, implementation, testing and deployment of the proposed application. The outcome of this thesis will be an application backend, which exposes its functionality through a REST interface.

Keywords Task management, productivity, team organization, backend, REST, Java Spring Framework

Obsah

Seznam obrázků

Úvod

Výpočetní technika umožnila rozvoj rychlejší a efektivnější komunikace, práce a vůbec způsobu života. Je běžné mít svůj kalendář on-line a sdílet ho s ostatními, případně používat některý nástroj s funkcí úkolníčku. Zejména ve velkých firmách, kde existuje silná potřeba koordinovat úsilí mnoha lidí, tak vznikla poptávka po nástrojích pro správu úkolů, které by jim umožnily efektivnější rozdělování zodpovědností a práce. Řešení, které na tento popud vznikly, slouží právě potřebám velkých firem. Potřeby menších skupin a jednotlivců jsou ale jiné. Pro ně jsou tyto nástroje příliš komplexní, těžkopádné a nedostatečně intuitivní.

Tato práce se zabývá přehledem existujících řešení pro správu úkolů malých i velkých skupin a tvorbou backendu nové aplikace. Aplikace bude zaměřena na potřeby menších týmů a jednotlivců a bude umožňovat uživatelům vytvářet úkoly, sdílet je s ostatními uživateli, sdružovat se do skupin a zaznamenávat průběh práce na úkolech.

V první části porovnám současná řešení pro správu úkolů z pohledu malých týmů a jednotlivců.

Ve druhé části se zabývám analýzou problému. Definuji požadavky na aplikaci ve formě uživatelských příběhů a podrobím je analýze.

Ve třetí části vypracuji návrh řešení aplikace. Zde vybírám technologii pro implementaci a popisuji architekturu aplikace, návrhový model tříd a databázový model.

Ve čtvrté části popisuji implementaci aplikace.

V páté části se věnuji testování aplikace za účelem zjištění její správné funkčnosti. Popíšu zde použité technologie a způsob testování.

V poslední, šesté části, vysvětlím, jak výslednou aplikaci nasadit a spustit.

Kapitola 1

Slovník pojmů

Deadline je časová lhůta, do které je nutné dokončit nějakou práci. Jeho synonyma jsou uzávěrka a konečný termín.

Assignee je člověk přidělený k určitému úkolu, je zodpovědný za práci na něm.

Open source software je softwarové dílo, k němuž jsou veřejně dostupné jeho zdrojové kódy.

Closed source software je softwarové dílo, jehož zdrojové kódy nejsou veřejně dostupné.

Gamifikace je uplatňování technik z herního designu a herních principů do neherních oblastí. Slouží například k motivaci formou získávání bodů za určité činnosti a zobrazováním grafů jejich vývoje.

Backend je část webové aplikace, která nekomunikuje přímo s uživatelem. V kontextu této práce zprostředkovává aplikační logiku a s okolím komunikuje skrze definované rozhraní.

Manhour je jednotka práce. Jeden člověk odpracuje za hodinu jeden *manhour*. Vypočte se jako: $manhours = lidi \times hodiny$.

Embedded znamená vestavěný, zabudovaný.

Kapitola 2

Současný stav

Způsobů, jak řešit správu úkolů, existuje spoustu a liší se podle toho, kdo je má využívat. V této části práce představím několik zástupců pro každou ze tří kategorií. Těmi jsou:

1. Řešení pro větší firmy s množstvím pracovníků
2. Řešení pro střední a menší týmy, jednotky až desítky pracovníků
3. Řešení pro jednotlivce

Zástupce vybírám podle osobních zkušenosti a podle výsledků získaných z vyhledávače `Google`, na základě jejich pořadí a popularity mezi uživateli. U uvedených zástupců uvedu krátký popis a jejich použitelnost cílovou skupinou této práce, tj. menšími týmy a jednotlivci.

Pracuji s tím, že pro cílovou skupinu této práce jsou důležitá následující kritéria:

1. **Nevyžaduje vlastní infrastrukturu, rychlé zprovoznení** – Uživatelé nechtějí spravovat vlastní hardware, kde by jim aplikace běžela. Začít používat aplikaci má být otázka maximálně několika málo minut.
2. **Použitelnost** – Kritérium jsem hodnotil subjektivně, podle mého názoru na pět kvalitativních částí použitelnosti[1]. Uživatel by se neměl ztratit ve funkcích aplikace a měl by být schopen rychle pochopit, jak s aplikací pracovat.
3. **Absence nepotrebné funkcionality** – Aplikace by měla obsahovat jen základní funkce, které uživatel využije. Větší množství funkcí, které uživatele nezajímají, ztěžují orientaci v aplikaci, což souvisí s předchozím bodem.
4. **Použití zdarma** – Aplikace by měla být použitelná zdarma. V případě, že se jedná o *freemium* model, měla by její neplacená část stačit k běžnému používání a neomezovat výrazně uživatele.

5. **REST API** – Aplikace by měla nabízet rozhraní REST API pro možnost vlastní integrace na její funkce.

6. **Open-source** – Aplikace by měla mít veřejně dostupné zdrojové kódy.

Uvedený přehled není vyčerpávající, věnuji se jen některým z těch nejznámějších řešení.

2.1 Řešení pro firmy

Řešení této kategorie se zaměřují na větší počet uživatelů a mimo základní správy úkolů nabízí často další funkce pro řízení projektů a integraci s dalšími systémy. Používají se zejména v oblasti vývoje software, ale dají se využít i v jiných oblastech.

2.1.1 JIRA

JIRA [2] je software, který nabízí bug tracking, issue tracking a funkce pro správu projektů. Je možné ho používat jak na vlastní HW infrastruktuře, tak on-line. V prvním případě je použití zdarma za určitých podmínek[1], v druhém případě je použití placené.

Nabízí širokou funkcionalitu a např. možnost upravovat podle potřeb životní cyklus úkolů. To ho činí využitelným i mimo vývoj software. Množství nabízených funkcí jde ale nad potřeby cílové skupiny této práce a technicky méně zdatné uživatele může mást. Na úkolech lze pracovat ve více lidech, ale přiřazen může být v jednu chvíli jen jednomu uživateli (*assignee*). Úkolům lze také přiřadit *deadline*.

JIRA nabízí REST API a je closed source.

2.1.2 Bugzilla

Bugzilla [5] je bug tracking nástroj, který je zaměřen hlavně na vývoj software. Je podobný nástroji JIRA, nicméně nenabízí takovou flexibilitu a i když by mohl být s dobře nastavenou politikou použitý pro správu úkolů u jiných než softwarových projektů, nebylo by použití intuitivní. Samotná správa a práce s úkoly funguje stejně jako u nástroje JIRA.

Bugzilla je open source, licencovaná pod MPL, a lze ji využít zdarma i pro komerční účely. Je nutné ji ale provozovat na vlastním hardware. Existují hosting služby, které jsou ale neoficiální a placené. Bugzilla nabízí REST API.

[1] Zdarma pro veřejně dostupný open-source software projekt[3] a pro neziskové, nevládní, neakademické, nekomerční a sekulární instituce, které by si jinak nemohly software dovolit. [4]

2.1.3 Redmine

Redmine [6] je issue tracking nástroj, který nabízí více flexibility než Bugzilla a obsahuje i některé nástroje pro řízení projektů. Tyto nástroje mohou být přínosné pro větší projekty, které mají danou strukturu, ale nepočítám s tím, že by cílovou skupinu této práce zajímaly. Funkcionalita, která se týká správy úkolů, je srovnatelná s předchozími dvěma nástroji.

Použití je zdarma, ale je nutné nainstalovat na vlastním hardware. Stejně jako v případě Bugzilly není Redmine oficiálně použitelný on-line, soukromé hostingy jsou placené. Projekt je vyvíjen jako open source a nabízí REST API.

2.2 Řešení pro střední a menší týmy

Nástroje v této kategorii se snaží cílit na týmy, spíše než celé firmy, a práce s nimi není tak formální. Používat je lze on-line, není nutné vlastní instalace. Oproti řešením v bodu 2.1 obsahují tyto méně funkcí, chybí hlavně různé manažerské nástroje a integrace s dalšími systémy.

2.2.1 Trello

Trello [7] je on-line aplikace, která vznikla v roce 2011. Způsob správy úkolů staví na konceptu *kanban*[8]. Umožňuje vytvářet nástěnky (boards), které reprezentují projekty. K nástěnkám lze přizvat další uživatele a pracovat na nich společně. Na nástěnkách se dají vytvářet seznamy (lists) a v nich karty (cards), které představují nejmenší jednotku práce - úkol. Ten má svou prioritu, *deadline* a zodpovědné uživatele. Těch může být libovolný počet.

Funkčně bohatý nástroj, s jednoduchým ovládáním i pro netechnické uživatele. Na profilu uživatele lze zobrazit všechny přiřazené karty a ty seřadit podle nástěnky, kam patří, nebo podle *deadlinu*.

Trello je možné používat zdarma s libovolným počtem spolupracovníků. Placené varianty přináší určité výhody[9], ale menší týmy se moho obejít bez nich. Existuje i REST API[10], aplikace je *closed source*.

2.2.2 Trackie

Trackie [11] je on-line aplikace, která je určena pro správu úkolů na společných projektech. Ty lze zakládat a zvát do nich uživatele, v rámci projektů pak tvořit úkoly. Úkol může být někomu přiřazen, ale vždy jen jednomu uživateli. Funkčností i vzhledem jednoduché na použití, ale některé funkce oproti předchozím nástrojům chybí. Úkolům například nelze nastavit *deadline*. Zobrazit je možné jen úkoly každého projektu zvlášť, nelze zobrazit přehled všech úkolů uživatele.

Aplikace nabízí 30denní zkušební dobu, po její uplynutí je placená. Nenabízí REST API a je closed source.

2.2.3 FogBugz

FogBugz [12] je nástroj řízení projektů, který kromě issue trackingu nabízí i možnost agilního plánování[13], správu podpory a zpětné vazby zákazníků, vytváření dokumentu ve stylu Wiki a další. Nabídkou funkcí je nejbohatší z trojice nástrojů v této kategorii, i přesto se v aplikaci uživatel neztratí.

Uživatelé spolu mohou pracovat na úkolech (cases), které se dělí do projektů (projects). Na úkolu lze evidovat informace potřebné pro využití agilní metodiky plánování, včetně *story points*. Úkolu je možné určit zodpovědného uživatele, odhad pracnosti a deadline. Na úkolu je možné průběžně zaznamenávat odpracovanou práci a zobrazovat kolik času na něm zbývá.

FogBugz nabízí 7denní zkušební dobu zdarma, poté je nutné za používání platit. Je možné ho používat jak on-line, tak na vlastním hardware. Nabízí REST API a je closed source.

2.3 Řešení pro jednotlivce

Poslední kategorií jsou nástroje pro správu úkolů jednotlivců. Některé z nich mohou umožňovat sdílení úkolů, takže by se daly zařadit i do kategorie 2.2, nicméně svým zaměřením cílí primárně na využití jako osobní úkolníček, proto jsou zařazeny zde. Také jsem do této kategorie zařadil nástroje, které nejsou primárně určeny pro správu úkolů, ale někteří je k tomuto účelu využívají, například kalendář nebo poznámky.

2.3.1 Todoist

Todoist [14] je on-line aplikace pro organizaci úkolů, která má uživatele motivovat k lepší produktivitě. Za splněné úkoly jsou přidělovány body (karma), jejichž historický vývoj je možné sledovat v grafu, lze nastavit denní a týdenní cíl a získávat další „odměny“ za jeho splnění. Úkoly lze rozdělit do projektů, nastavit jim *deadline*, prioritu a opakování. Projekty je možné sdílet s dalšími uživateli a dají se zobrazit buď po projektech, ke kterým patří, nebo všechny na jednom místě – ve schránce (inbox).

Vytvoření nového úkolu se může provést zadáním (anglického) textu, aplikace sama rozpozná klíčová slova a není tak nutné nastavovat vlastnosti úkolů ručně. Například heslo *Go jogging at 1 PM every day* vytvoří úkol *Go jogging*, který má *deadline* ve 13 hodin a opakuje se každý den.

Placená verze nabízí větší množství otevřených projektů a úkolů, hledání v úkolech, notifikace a další.[15] Todoist nabízí REST API[16] a je closed source.

2.3.2 Toodledo

Toodledo [17] je on-line aplikace, která kromě úkolů umožňuje vytvářet si zvyky (habits). To jsou úkoly, které se opakují ve volitelné dny každý týden,

jež mají uživatelům pomoci vypěstovat si a dodržovat dobré návyky. Po vykonání zvyku je možné označit ho za splněný a přidat k jeho splnění číslo nebo hodnocení. Mezi další možnosti patří poznámky (notes), seznamy (lists), nebo nástiny (outlines).

Úkoly lze třídit do složek, nastavit jim *deadline* a prioritu. Spolupráce s dalšími uživateli vyžaduje placenou verzi aplikace a Webové rozhraní aplikace je oproti nástroji Todoist poměrně nepřehledné. Toodledo nabízí REST API[18] a je closed source.

2.3.3 Google Inbox Reminders

Inbox [19] je e-mailový klient od společnosti Google, který kromě práce s e-maily umožňuje vytváření jednoduchých úkolů (reminders). Ty lze kromě aplikace Inbox také zobrazit v kalendáři Google Calendar.[20] Úkolům nelze nastavit *deadline*, ale je možné je odložit (snooze) tak, aby se zobrazily později. Aktivní úkoly se zobrazují mezi příchozími e-maily.

Úkoly nelze nijak třídit, ani u nich určovat prioritu. Jediná informace v úkolu je jeho popis. Pro základní potřeby dostačující, ale jinak funkčně chudý nástroj zatím nenabízí API[21] a je closed source.

2.3.4 Poznámky, kalendář

Tato sekce nepopisuje jeden konkrétní produkt, ale jejich skupinu. Uvádím je pro úplnost, jelikož je stále využívá velké množství lidí. Patří sem všechny nástroje pro psaní poznámek a vytváření událostí v kalendáři, a to jak elektronické, tak papírové.

Některé elektronické nástroje umožňují sdílení poznámek či kalendářů, takže se dají při dobře definovaných pravidlech použít i pro týmovou práci. Jejich použití je ale složitější s rostoucím počtem úkolů a projektů, protože neumožňují žádné filtrování, řazení úkolů ani přiřazování zodpovědností. Pro nenáročného uživatele, který si chce zapisovat nejdůležitější úkoly a sám se postará o to, že na nich nezapomene včas začít, může být toto řešení dostačující.

Pro papírové „nástroje“ platí předchozí odstavec podobně, jen možnost spolupráce je ještě více omezena. A pokud celá skupina nemá společnou místnost pro práci, pak prakticky vyloučena. Navíc vzniká problém s archivací a uchováváním historie úkolů.

2.4 Shrnutí

Přehled kritérií stanovených na začátku kapitoly 2 a jejich splnění uvedenými nástroji prezentuji v tabulce B.1 (příloha).

Všechny nástroje umožňují vytvářet úkoly, ale v možnostech se liší. Nyní zvážím, které funkce jsou užitečné pro cílovou skupinu této práce a mají být

obsaženy ve výsledné aplikaci. Při odkazování se k nástrojům nebo aplikacím v této sekci mám na mysli ty představené výše.

Většina nástrojů umožňuje úkoly sdílet s jinými uživateli a společně s nimi pracovat na projektu. Možnost seskupit se do jednoho celku se spolupracovníky je užitečná, proto budou v aplikaci existovat pro uživatele pracovní skupiny.

V aplikacích je u úkolu možné určit uživatele za něj odpovědné. Většinou lze přiřadit k úkolu v jednu chvíli jen jednoho uživatele, ale v nástroji Trello jich lze přiřadit libovolný počet. To může být pro některé typy úkolů užitečné.

Kromě *deadlinu* a priority jako ve většině ostatních aplikací, lze ve FogBugz u úkolu nastavit i odhad pracnosti – dobu, jakou by měla práce na úkolu trvat. Uživatel pracující na úkolu pak může průběžně zaznamenávat odpracovanou dobu a celá skupina tak vidí, v jaké fázi se úkol nachází. Toto chci zahrnout do výsledné aplikace a ještě rozšířit o další funkci. Na základě doby zbývající do *deadlinu* a zbývající pracnosti na úkolu lze určit „urgentnost“, tedy jakou má ještě uživatel rezervu, než *deadline* zmešká. Tato informace o rezervě může uživatelům pomoct s rozhodováním, kterému úkolu se prioritně věnovat, aby jej ještě stihli.

Například Todoist nabízí možnost vytvořený úkol opakovat v zadaném intervalu. Google Inbox kromě opakování umožňuje také úkol odložit na později, čímž uživateli zmizí z očí. V aplikaci chci vyjít z těchto funkcí a vytvořit funkci podobnou. Myšlenka je, že opakující se úkoly nebudou mít pevně daný *deadline*. Uživatel se jim chce věnovat zhruba po nějakých intervalech, ale ne na den přesně. Při vytvoření úkol nebude důležitý, ale bude postupně nabývat na urgentnosti, až se důležitým stane. Poté uživatel úkol buď ukončí, nebo vynuluje a úkol znovu poroste.

Backend část aplikace, jež bude výstupem této práce, bude zaměřením spadat na pomezí sekcí 2.2 a 2.3. Bude obsahovat vybrané funkce pro správu práce týmů i jednotlivců, které analyzuji v kapitole 3.

Kapitola 3

Analýza

V této sekci se budu věnovat specifikování požadavků na aplikaci a jejich analýze. To mi poskytne konkrétní přehled o tom, co má výsledná aplikace umět a jaké entity v ní budou existovat.

3.1 Požadavky na aplikaci

Požadavky na aplikaci popíšu pomocí *user stories* (uživatelské příběhy)[22]. Jedná se o jednoduchý způsob, jakým vyjádřit požadavek na aplikaci z pohledu jejího uživatele. Měl by být krátký a napsán v terminologii uživatele budoucí aplikace.

Formální způsob zápisu user story se dá definovat [23] například takto:

Jako ⟨role⟩ chci ⟨něco⟩, abych dosáhl ⟨něčeho⟩.

Pomocí *user stories* lze také testovat, zda aplikace obsahuje všechny požadované funkce. Pokud jsou všechny *user stories* uspokojeny, pak je aplikace z hlediska funkčnosti hotova.

3.1.1 User stories

1. **Uživatel**

   1.1. Jako uživatel chci mít svůj účet, abych mohl spravovat své úkoly a pracovní skupiny. Účet bude jednoznačně identifikován e-mailem uživatele a pro přihlášení bude potřeba heslo. Dále bude účet obsahovat jméno uživatele. Svůj účet budu moci zobrazit a upravit svoje jméno a heslo.

   1.2. Jako uživatel chci u úkolu mít vždy jednu ze dvou rolí – pozorovatel a pracovník – abych dal najevo, v jakém vztahu k úkolu jsem. Jako pozorovatel mám jen pasivní roli, úkol vidím ve svém seznamu. Jako pracovník se na úkolu aktivně podílím.

1.3. Jako uživatel chci vytvářet úkoly soukromé nebo sdílené s ostatními uživateli a pracovními skupinami, abych nezapomněl na své nebo týmové úkoly, které je potřeba vypracovat. Úkol bude obsahovat následující informace:
   - Název
   - Popis
   - Datum a čas začátku
   - Pracnost (*manhours*)
   - Priorita
   - Typ
     - S deadlinem – úkol má daný *deadline*, kdy má být splněn
     - Rostoucí – úkol nemá daný *deadline*, ale jeho urgentnost postupně roste
   - Datum a čas *deadlinu* (pro úkoly „s deadlinem“)
   - Rychlost růstu urgentnosti (pro „rostoucí“ úkoly)
   - Stav
     - Otevřený
     - Splněný
     - Zavřený

1.4. Jako pozorovatel nebo pracovník úkolu ho chci sdílet s dalšími uživateli

1.5. Jako pracovník úkolu ho chci modifikovat, abych mohl reagovat na případné změny. Změnit budu moct všechny údaje, kromě jeho typu.

1.6. Jako uživatel chci znát *urgentnost* úkolů, abych věděl, kterým úkolům se mám věnovat tak, abych stihl jejich deadline. Čím méně času zbývá na vypracování úkolu vzhledem k jeho pracnosti a deadlinu, tím více je urgentní.

1.7. Jako uživatel chci zobrazit všechny úkoly, kterých jsem součástí – moje osobní i sdílené se mnou nebo pracovní skupinou, již jsem členem – abych měl přehled o tom, co je potřeba udělat. Zobrazení by mělo jít filtrovat a řadit podle následujících kritérií:
   - Filtr
     - Role (pozorovatel / pracovník)
     - Typ
     - Stav
     - Vlastnictví (můj / sdílený se mnou)
     - Priorita
   - Řazení

  - Název
  - Datum začátku úkolu
  - Datum *deadlinu*
  - Rozpracovanost (kolik procentuelně odpracováno)
  - Priorita
  - Urgentnost

1.8. Jako uživatel chci přijímat nebo zamítat příchozí návrhy na sdílení úkolů, abych se mohl rozhodnout, na kterých se chci podílet a na kterých ne.

1.9. Jako pracovník úkolu na něm chci zaznamenávat odvedenou práci, abych měl já i mí spolupracovníci lepší představu o tom, kolik práce na něm zbývá.

1.10. Jako pracovník úkolu ho prohlašovat úkol za splněný, uzavřený, nebo znovuotevřený, abych dal najevo, v jakém stavu se úkol nachází.

1.11. Jako pracovník na „rostoucím“ úkolu chci vynulovat jeho urgentnost, aby poté, co jsem ho splnil, dočasně přestal být důležitý. Tuto funkci využiji při opakovaných úkolech.

1.12. Jako uživatel chci zakládat pracovní skupiny a po založení se stát jejich adminem, abych se mohl sdružovat s dalšími uživateli a pracovat společně. Pracovní skupina obsahuje tyto informace:

  - Název
  - Popis

1.13. Jako uživatel chci vidět seznam skupin, jichž jsem členem, manažerem nebo adminem, a jejich detaily, abych rychle viděl, se kterými skupinami spolupracuji.

2. **Manažer skupiny**

2.1. Jako manažer skupiny do ní chci zvát a odebírat z ní uživatele. Členům chci přiřazovat role pozorovatel nebo pracovník na úkolech, které jsou se skupinou sdíleny, abych tím mohl organizovat skupinu a rozdělovat zodpovědnost za práci mezi její členy.

2.2. Jako manažer skupiny chci vytvářet úkoly pro skupinu a přijímat nebo zamítat návrhy na sdílení úkolů s mou skupinou, abych mohl určovat, na čem má skupina pracovat.

2.3. Jako manažer skupiny chci mít možnost odebrat ji z úkolu, který je s ní sdílen, aby úkoly, které již nejsou relevantní, nezůstávaly v seznamu úkolů skupiny.

3. **Admin skupiny**

3.1. Jako admin skupiny chci mít možnost měnit její název a popis, abych je mohl udržovat aktuální.

3.2. Jako admin skupiny chci jmenovat manažery skupiny, abych mohl pověřit další uživatele správou skupiny.

3.3. Jako admin skupiny chci mít možnost přenechat svou roli jinému členovi skupiny, abych mohl skupinu opustit a ta pořád měla admina.

3.4. Jako admin skupiny chci mít možnost skupinu zrušit.

3.1.2 Nefunkční požadavky

1. Aplikace bude dostupná přes REST API. Jedná se o pouze o backend, takže k dispozici nebude žádné grafické rozhraní.

2. Aplikace nebude z důvodu bezpečnosti uchovávat hesla jako prostý text. Oveření bude provedeno vhodným způsobem, který znemožní únik hesel.

3. Zdrojové kódy aplikace budou veřejně dostupné pod MIT licencí.

3.2 Model případů užití

User stories ze sekce 3.1.1 nyní budu specifikovat detailněji pomocí modelu případů užití (*use case model*). *User story* by měl obsahovat tolik informací, aby i technicky nezkušený uživatel dokázal pochopit, co má aplikace dělat. Oproti tomu případy užití popisují konkrétní interakce uživatele s aplikací, tedy chování, které má aplikace mít, aby splnila potřeby na ní kladené. [24]

Případů užití typicky bude více než *user stories*[25], protože *user stories* jsou více obecné, a tak na pokrytí jednoho může být potřeba více interakcí s aplikací a tedy více případů užití.

3.2.1 Účastníci

Aplikace bude rozlišovat účastníky na základě několika kritérií a těm bude dostupné různé množství funkcí. Účastníci jsou zobrazeni podle standardu UML na obrázku 3.1. Účastníky jsou v tomto případě uživatelé, kteří budou aplikaci používat. Každý uživatel může v různých případech užití vystupovat jako jiný účastník, v rámci jednotlivých případů užití ale musí být neměnný. [26]

Účastníci od sebe mohou dědit, v tom případě potomek může kromě svých akcí vykonávat navíc všechny akce svého rodiče.

Nepřihlášený uživatel
Nepřihlášený uživatel je účastník, který používá aplikaci poprvé, nebo se z ní po předchozím používání odhlásil.

Přihlášený uživatel
Uživatel se stane přihlášeným po zadání svého jména a hesla. Jsou mu dostupné funkce aplikace a může ji používat.

Člen skupiny
Uživatel je Člen skupiny, pokud k této skupině patří. Členové skupiny mohou mít také další role v rámci této skupiny.

Manažer skupiny
Uživatel je Manažer skupiny, pokud mu Admin tuto roli přiřadil. Jako Manažer má více privilegií pro správu skupiny.

Admin skupiny
Uživatel je Admin skupiny, pokud ji založil, nebo mu současný Admin tuto roli přenechal. Jako Admin má nejvyšší privilegia ve skupině a tato role v rámci skupiny patří vždy právě jednomu uživateli.

Účastník úkolu
Účastník úkolu je abstraktní účastník. Reprezentuje uživatele, který je součástí úkolu v nějaké roli.

Pozorovatel úkolu
Pozorovatel úkolu je součástí úkolu v pasivní roli. Na úkolu přímo nepracuje, ale má zájem sledovat jeho průběh.

Pracovník úkolu
Pracovník úkolu je součástí úkolu v aktivní roli. Na úkolu pracuje a má tak zpřístupněny některé funkce navíc oproti Pozorovateli.

Čas
Účastník Čas nereprezentuje žádného fyzického uživatele, ale spouští události, které mají nastat s uplynulým časem. [26]

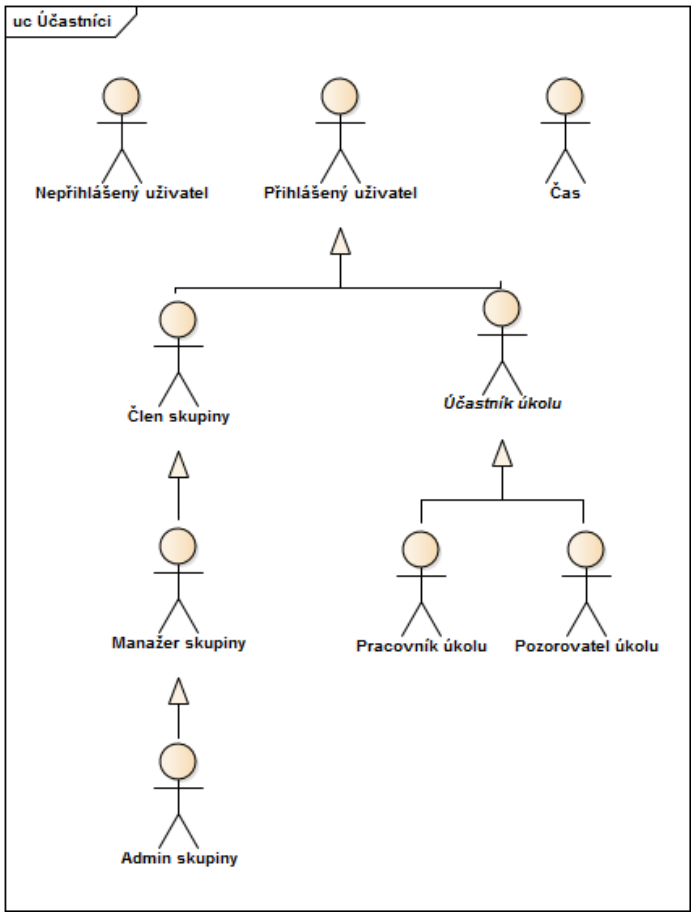
3.2.2 Případy užití

Případy užití získané analýzou *user stories* zobrazuji v diagramech případů užití. Rozdělil jsem je do čtyř balíků podle jejich účastníků pro lepší přehlednost.

3.2.2.1 Nepřihlášený uživatel

V diagramu 3.2 jsou zachyceny případy užití pro Nepřihlášeného uživatele.

UC 1.01 Registrovat se
Umožní uživateli aplikace zaregistrovat se. Zadá svůj e-mail, heslo a jméno.



Obrázek 3.1: Diagram účastníků aplikace

UC 1.02 Přihlásit se
Umožní uživateli přihlásit se pomocí svého e-mailu a hesla.

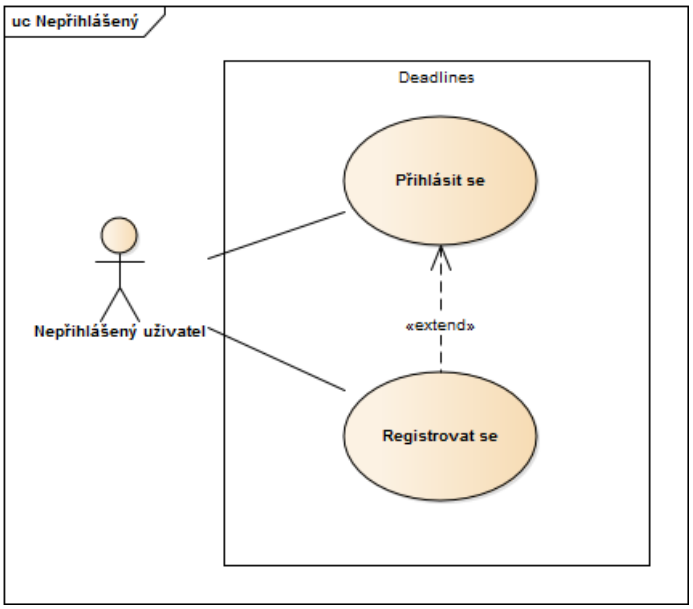
3.2.2.2 Případy užití pro jednotlivce

V diagramu 3.3 a 3.4 jsou zachyceny případy užití aplikace jednotlivci. V prvním diagramu jsou případy užití z pohledu Přihlášeného uživatele v druhém pak z pohledu Pozorovatele a Pracovníka úkolu.

UC 2.01 Odhlásit se
Odhlásí uživatele z aplikace.

UC 2.02 Upravit svůj profil
Umožní uživateli upravit si své jméno a heslo.

UC 2.03 Vytvořit úkol
1. Uživatel chce vytvořit úkol



Obrázek 3.2: Diagram případů užití pro nepřihlášeného uživatele

2. Uživatel vyplní informace k úkolu, povinně název a typ. Pokud je typ „S deadlinem“, pak je povinné i datum a čas deadlinu. Volitelně může zadat popis, pracnost v *manhours* a prioritu jako celé číslo v intervalu 1–5.
3. <Sdílet úkol>
4. Aplikace vytvoří úkol, uživatel se stane jeho Pracovníkem.

UC 2.04 Sdílet úkol

1. Uživatel chce sdílet úkol.
2. <Zobrazit seznam uživatelů>
3. <Zobrazit seznam skupin>
3. Uživatel určí uživatele a skupiny, se kterými chce úkol sdílet.
4. Aplikace pošle vybraným uživatelům a skupinám nabídku ke sdílení úkolu.

UC 2.05 Zobrazit seznam mých úkolů

1. <Filtrovat úkoly>
2. <Řadit úkoly>
3. Aplikace zobrazí seznam všech úkolů, kterých je uživatel součástí, se zvoleným filtrem a řazením. Výchozí řazení je podle urgentnosti.

UC 2.06 Filtrovat úkoly

Úkoly je možné filtrovat podle role uživatele v nich (pozorovatel nebo

pracovník), podle jejich typu (s deadlinem nebo rostoucí), stavu (otevřený, splněný nebo zavřený) a priority (celé číslo 1–5).

UC 2.07 Řadit úkoly

Úkoly je možné řadit podle zvoleného kritéria. Tím může být název, datum začátku nebo *deadlinu* úkolu, rozpracovanost (v procentech odpracováno ku celkové pracnosti), priorita a urgentnost.

UC 2.08 Rozhodnout o příchozí nabídce sdílení úkolu

1. <Zobrazit seznam příchozích nabídek na sdílení úkolu>
2. Uživatel rozhodne, zda chce nabídku přijmout, nebo zamítnout.
3. Pokud je nabídka přijata, stává se uživatel pozorovatelem sdíleného úkolu. V opačném případě je nabídka smazána.

UC 2.09 Zobrazit seznam nabídek na sdílení úkolu

Zobrazí seznam nabídek pro uživatele k připojení se ke sdílenému úkolu, v němž bude u každé nabídky uvedeno jméno sdílejícího uživatele a název úkolu.

UC 2.10 Založit pracovní skupinu

1. Uživatel chce založit pracovní skupinu
2. <Zobrazit seznam uživatelů>
3. Uživatel zadá jméno skupiny a volitelně její popis. Dále zadá seznam e-mailů uživatelů, které chce do skupiny pozvat.
4. Aplikace vytvoří novou skupinu s daným názvem a popisem. Zakládající uživatel se stane jejím Adminem.
5. Aplikace rozešle nabídky ke vstupu do skupiny zadaným uživatelům.

UC 2.11 Rozhodnout o příchozí nabídce ke vstupu do skupiny

Uživatel rozhodne, zda chce přijmout nebo odmítnout nabídku ke vstupu do skupiny. Pokud ji přijme, stane se jejím Členem.

UC 2.12 Zobrazit seznam nabídek ke vstupu do skupiny

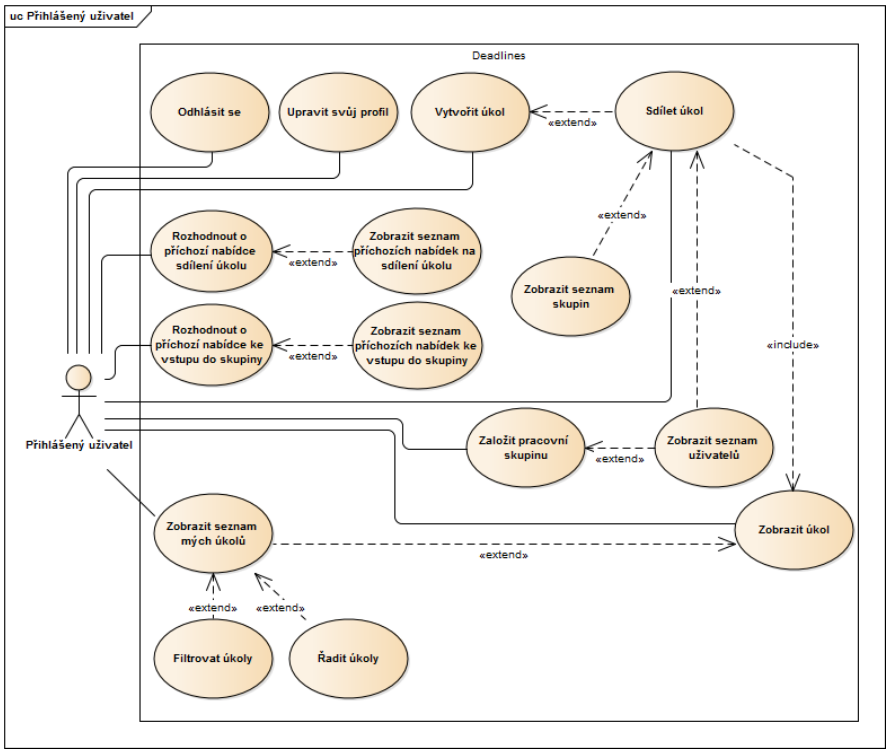
Zobrazí seznam nabídek ke vstupu do skupin. U každé nabídky bude uvedeno jméno skupiny a jméno a e-mail uživatele, od něhož nabídka pochází.

UC 2.13 Zobrazit seznam uživatelů

Zobrazí seznam uživatelů aplikace. U uživatele bude uvedeno jeho jméno a e-mail.

UC 2.14 Zobrazit seznam skupin

Zobrazí seznam skupin v aplikaci. U skupiny bude uvedeno její jméno a admin.



Obrázek 3.3: Diagram případů užití pro přihlášeného uživatele

UC 2.15 Zobrazit úkol

1. Uživatel chce zobrazit detaily úkolu
2. <Zobrazit seznam mých úkolů>
3. Aplikace zobrazí všechny informace o úkolu, tedy název, popis, datum a čas začátku a *deadlinu*, pracnost, prioritu, typ, stav a seznam jeho Pracovníků a Pozorovatelů.

UC 2.16 Stát se pracovníkem úkolu

Umožňuje Pozorovateli úkolu stát se jeho Pracovníkem.

UC 2.17 Změnit stav úkolu

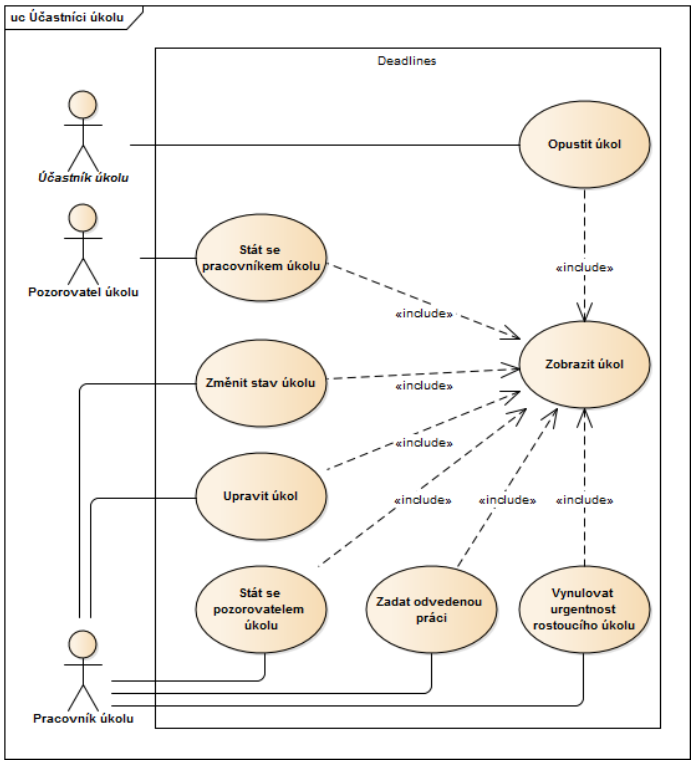
Umožňuje Pracovníkovi změnit stav úkolu na jakýkoliv z jeho možných stavů.

UC 2.18 Upravit úkol

Umožňuje Pracovníkovi upravit u úkolu jeho popis, *deadline*, pracnost a prioritu.

UC 2.19 Stát se pozorovatelem úkolu

Umožňuje Pracovníkovi úkolu stát se jeho Pozorovatelem.



Obrázek 3.4: Diagram případů užití pro účastníky úkolu

UC 2.20 Zadat odvedenou práci
Umožňuje Pracovníkovi zadat u úkolu odvedenou práci.

Hlavní scénář : Pracovník chce zadat počet odpracovaných hodin.
1. Include(Zobrazit úkol)
2. Uživatel zadá počet odpracovaných hodin a potvrdí.
3. Aplikace k úkolu přidá zadaný počet hodin.

Alternativní scénář : Pracovník chce měřit odpracovaný čas. 1. Include(Zobrazit úkol) 2.

UC 2.21 Vynulovat urgentnost rostoucího úkolu
Umožňuje Pracovníkovi vynulovat urgentnost rostoucího úkolu.

UC 2.21 Opustit úkol
Umožňuje Účastníkovi úkolu opustit ho. Účastníkem úkolu je možné být přímo a/nebo skrze skupinu. Opustit úkol lze pouze, pokud je Účastník součástí úkolu přímo.

1. Případ užití začíná, když se Účastník úkolu rozhodne opustit úkol.
2. Include(Zobrazit úkol)

3. Účastník zašle požadavek na opuštění úkolu.
4. Pokud nebyl úkol sdílen přímo s Účastníkem (ale skupinou, ve které je Členem), scénář končí.
5. Aplikace odebere Účastníka z úkolu. Pokud byl ale Účastník navíc součástí úkolu skrze skupinu, jíž je členem, nadále zůstane součástí úkolu skrze tuto skupinu.

3.2.2.3 Případy užití pro skupiny

V diagramu 3.5 jsou zachyceny případy užití aplikace členy pracovních skupin. Týkají se správy skupinových úkolů a skupin samotných.

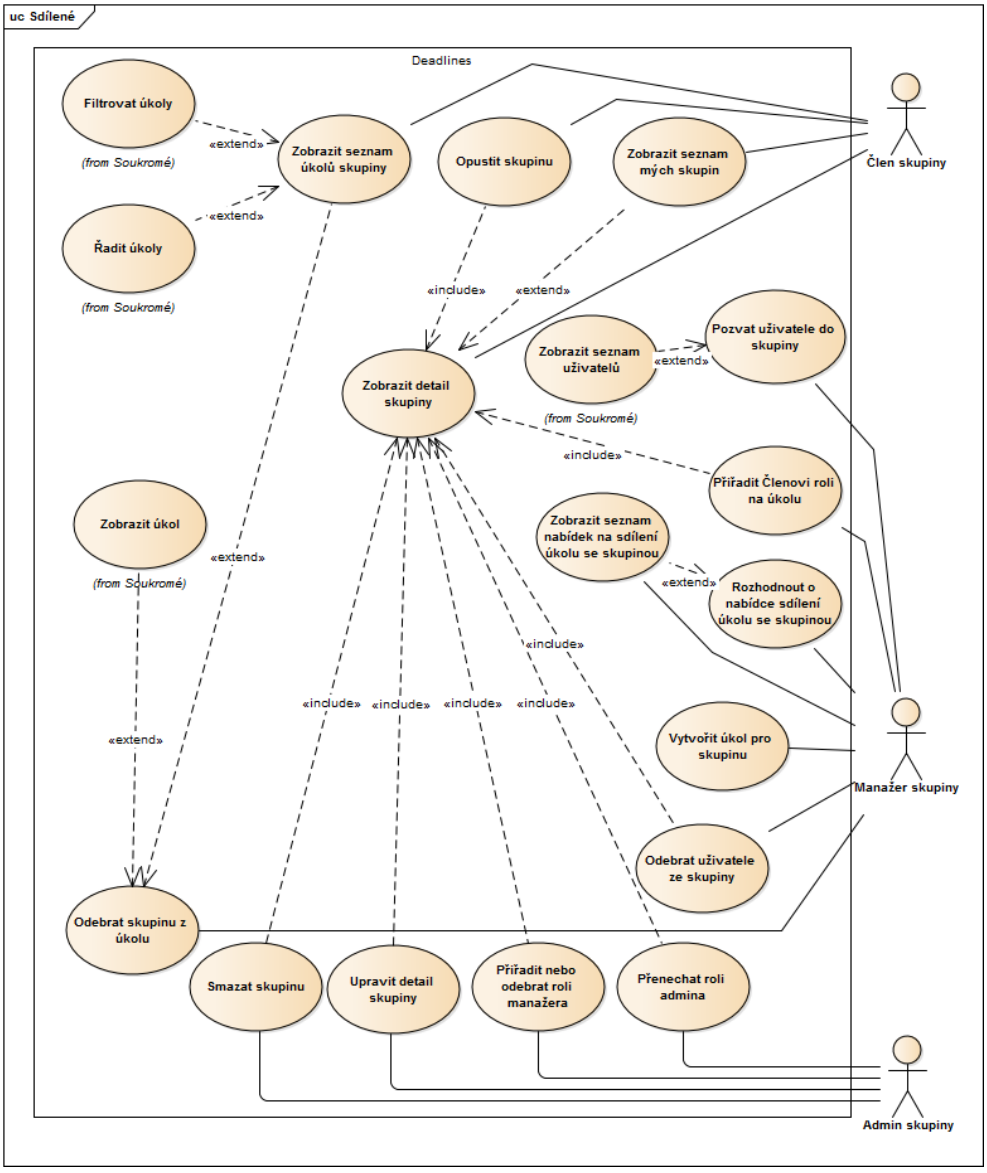
UC 3.01 Zobrazit seznam úkolů skupiny
1. Člen skupiny chce zobrazit úkoly své skupiny
2. <Filtrovat úkoly>
3. <Řadit úkoly>
4. Aplikace zobrazí seznam všech úkolů skupiny se zvoleným filtrem a řazením. Výchozí řazení je podle urgentnosti.

UC 3.02 Opustit skupinu
1. Člen chce opustit skupinu.
2. Include(Zobrazit detail skupiny)
3. Člen zašle požadavek na opuštění skupiny.
4. Pokud je Člen Adminem skupiny, opuštění je zamítnuto a scénář končí.
5. Aplikace odebere Člena ze skupiny a ze všech úkolů, jichž byl součástí skrze opouštěnou skupinu.

UC 3.03 Zobrazit seznam mých skupin
Zobrazí Členovi seznam názvů všech skupin, ve kterých je Členem.

UC 3.04 Zobrazit detail skupiny
1. <Zobrazit seznam mých skupin>
2. Člen zvolí skupinu, u které chce zobrazit detaily, a pošle požadavek.
3. Pokud Člen nepatří do zvolené skupiny, je zobrazení odepřeno a scénář končí.
4. Aplikace zobrazí detaily skupiny, tedy její jméno, popis, seznam Členů, Manažerů a Admina.

UC 3.05 Pozvat uživatele do skupiny
1. Případ užití začíná, když Manažer chce pozvat uživatele do skupiny.
2. <Zobrazit seznam uživatelů>
3. Manažer vybere uživatele, kterého chce pozvat, a pošle požadavek.
4. Aplikace zašle uživateli nabídku ke vstupu do skupiny.



Obrázek 3.5: Diagram případů užití pro členy pracovní skupiny

UC 3.06 Přiřadit Členovi roli na úkolu
Umožňuje přiřadit Členům skupiny roli Pozorovatele nebo Pracovníka na úkolech skupiny.

UC 3.07 Zobrazit seznam nabídek na sdílení úkolu se skupinou
Zobrazí seznam příchozích nabídek ke sdílení úkolů. U nabídky bude jméno uživatele, od kterého pochází, a název úkolu, kterého se týká.

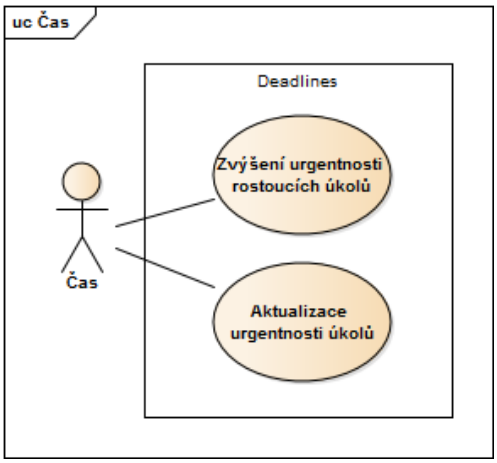
UC 3.08 Rozhodnout o nabídce sdílení úkolu se skupinou
1. Případ užití začíná, když Manažer chce rozhodnout o nabídce sdílení úkolu se skupinou.
2. <Zobrazit seznam nabídek na sdílení úkolu se skupinou>
3. Pokud Manažer nabídku odmítne, je smazána a scénář končí.
4. Pokud Manažer nabídku přijme, Aplikace úkol zařadí mezi úkoly skupiny a všichni Členové skupiny se stanou jeho Pozorovateli.

UC 3.09 Vytvořit úkol pro skupinu
1. Případ užití začíná, když Manažer chce vytvořit úkol pro skupinu.
2. Manažer vyplní informace k úkolu, povinně název a typ. Pokud je typ „S deadlinem“, pak je povinné i datum a čas deadlinu. Volitelně může zadat popis, pracnost v *manhours* a prioritu jako celé číslo v intervalu 1–5. Zadá také skupinu, pro kterou chce úkol vytvořit.
3. Pokud Manažer není Manažerem zvolené skupiny, vytvoření úkolu selže a scénář končí.
4. Aplikace vytvoří úkol se zadanými parametry a přidá ho do úkolů skupiny, s níž byl sdílen.
5. Aplikace přiřadí na úkolu všem Členům skupiny roli Pozorovatel.

UC 3.10 Odebrat uživatele ze skupiny
Umožňuje Manažerovi odebrat zvoleného uživatele ze skupiny. Ten je odebrán ze všech úkolů, jichž byl součástí skrze skupinu. Pokud se jedná o Manažera nebo Admina, je odebrání zakázáno.

UC 3.11 Odebrat skupinu z úkolu
1. Případ užití začíná, když chce Manažer odebrat skupinu z úkolu, aby už nebyla jeho součástí.
2. <Zobrazit úkol>
3. <Zobrazit seznam úkolů skupiny>
4. Manažer vybere úkol, ze kterého chce skupinu odebrat, a zašle požadavek.
5. Aplikace u úkolu odebere sdílení se skupinou a odebere z úkolu všechny Členy skupiny, pokud nejsou součástí úkolu i mimo skupinu.

UC 3.12 Upravit detail skupiny
Umožňuje Adminovi měnit popis skupiny.



Obrázek 3.6: Diagram případů užití pro časově závislé události

UC 3.13 Přiřadit nebo odebrat roli manažera
Umožňuje Adminovi přiřazovat a odebírat roli Manažera u všech Členů skupiny.

UC 3.14 Přenechat roli admina
Umožňuje Adminovi přenechat svou roli jinému členovi skupiny.

UC 3.15 Smazat skupinu
1. Případ užití začíná, když se Admin rozhodne smazat skupinu.
2. Include(Zobrazit detail skupiny)
3. Admin zašle požadavek na smazání skupiny.
4. Pokud Admin není Adminem ve skupině, kterou chce smazat, scénář končí.
5. Aplikace odebere všechny ostatní členy ze skupiny („Odebrat uživatele ze skupiny“).
6. Aplikace odebere skupinu ze všech jejích úkolů („Odebrat skupinu z úkolu“).
7. Aplikace odebere Admina ze skupiny.
8. Aplikace smaže skupinu.

Diagram 3.6 zachycuje časově závislé případy užití, které se pravidelně vykonávají bez přičinění uživatelů aplikace.

UC 4.01 Zvýšení urgentnosti rostoucích úkolů
1. Případ užití začíná po uplynutí časového intervalu pro zvyšování urgentnosti rostoucích úkolů.
2. Aplikace u všech úkolů typu „rostoucí“ zvýší urgentnost o hodnotu

danou nastavením úkolu.

UC 4.02 Aktualizace urgentnosti úkolů

1. Případ užití začíná po uplynutí časového intervalu pro aktualizaci urgentnosti úkolů.
2. Aplikace u všech úkolů typu „s deadlinem“ vypočte novou hodnotu urgentnosti.

3.3 Doménový model

Doménový model slouží k popisu dat, zachycení hlavních entit, se kterými bude aplikace pracovat, a vztahů mezi nimi. [27] Jedná se o abstraktní pohled na problém, a tak diagram obsahuje jen entity a jejich vlastnosti dané předchozími kroky analýzy. Nejsou v něm implementační detaily, jako například konkrétní atributy a metody.

Diagram doménového modelu je znázorněn na obrázku C.1. Znázorňuje vztahy mezi entitami a některá omezení, jako například vzájemnou výlučnost. [28] V této části se dále budu věnovat popisu jednotlivých entit modelu.

3.3.1 User

Entita User reprezentuje uživatele aplikace. Uchovává jeho e-mail, který je unikátní v rámci aplikace, jméno a heslo k jeho přihlášení.

3.3.2 Group

Entita Group reprezentuje pracovní skupinu. Má název a popis. Uživatelé s ní mohou být ve třech různých vztazích – jako Člen, Manažer a Admin. Tyto vztahy jsou vzájemně vylučující, takže uživatel nemůže být zároveň například Člen a Manažer.

3.3.3 Task

Entita Task reprezentuje úkol. Obsahuje informace definované v předchozích krocích analýzy. Uživatel může být součástí úkolu jako pozorovatel, nebo pracovník. Nemůže být ale oboje zároveň.

3.3.4 DeadlineTask

Entita DeadlineTask je typ entity Task, reprezentující úkol s *deadlinem*.

3.3.5 GrowingTask

Entita GrowingTask je typ entity Task, reprezentující rostoucí úkol. Obsahuje informaci o rychlosti růstu jeho urgentnosti.

3.3.6 TaskParticipant

TaskParticipant je entita reprezentující dvojici entit User–Group. Tato dvojice je v aplikaci unikátní, takže nemůžou existovat dvě stejné dvojice zároveň.

Skrze tuto entitu je uživatel součástí úkolu. Součástí úkolu může být sám, pokud není k entitě TaskParticipant přiřazena žádná skupina, nebo může být součástí úkolu jako člen skupiny, pokud k entitě skupina přiřazena je. Uživatel také může být součástí úkolu jako člen více skupin zároveň.

3.3.7 TaskWork

Entita TaskWork reprezentuje práci vykonanou na úkolu uživatelem. Je spjata s entitou User, místo TaskParticipant, takže i když uživatel úkol opustí, pořád bude záznam o jeho práci na úkolu existovat.

3.3.8 Offer

Entita Offer reprezentuje nabídku, kterou uživatel někomu zasílá.

3.3.9 TaskSharingOffer

Entita TaskSharingOffer je typ nabídky, která reprezentuje nabídku ke sdílení úkolu.

3.3.10 UserTaskSharingOffer

Entita UserTaskSharingOffer je typ nabídky ke sdílení úkolu, která je určena pro uživatele. Vznikne, pokud jeden uživatel chce k úkolu, na kterém pracuje, pozvat jiného uživatele.

3.3.11 GroupTaskSharingOffer

Entita UserTaskSharingOffer je typ nabídky ke sdílení úkolu, která je určena pro skupinu. Vznikne, pokud jeden uživatel chce k úkolu, na kterém pracuje, pozvat skupinu.

3.3.12 MembershipOffer

Entita MembershipOffer je typ nabídky, která reprezentuje nabídku ke vstupu do skupiny. Vznikne, pokud manažer skupiny do ní chce pozvat nového uživatele.

Kapitola 4

Návrh řešení

V této kapitole představím technologie zvolené k vypracování aplikace a navrhnu REST API, které bude aplikace nabízet. Dále se budu věnovat architektuře aplikace, návrhovému modelu tříd a databázovému modelu, což bude představovat podklad pro následující implementaci prototypu aplikace ve zvolené technologii.

4.1 Technologie

4.1.1 Java

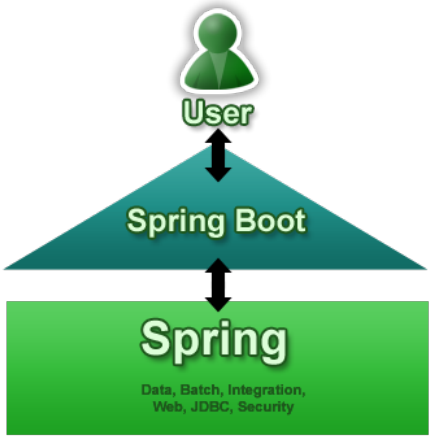
Již v zadání práce je avizováno, že aplikace bude postavena na platformě Java. [29] Jazyk Java byl vyvinutý společností Sun Microsystems a veřejnosti představen v roce 1996. Syntaxí je podobný C/C++ a je silně objektově orientovaný. Program napsaný v Javě je kompilován do podoby *bytecode*, který je následně spouštěn na *Java Virtual Machine* (JVM), který bytecode interpretuje. To znamená, že jazyk je platformně nezávislý – *bytecode* vzniklý kompilací zdrojového Java kódu je možné spustit na jakékoliv platformě, pokud na ní je přítomný odpovídající *JVM*.

Oproti jazykům C/C++ navíc Java disponuje automatickou správou paměti, nazvanou *garbage collection*, takže vývojáři se nemusí tolik starat o životní cyklus objektů a následné *memory leaks*. [30] [31]

4.1.2 Spring

Spring je framework, který zjednodušuje tvorbu enterprise aplikací v Javě. [32] Jednou z výhod Springu je jeho podpora návrhového vzoru *dependency injection*, díky kterému je program lépe modularizovatelný, testovatelný a znovupoužitelný.[33]

Další součástí Springu je webový modul *Web MVC framework*. [**?**] Ten umožňuje tvorbu webových aplikací podle architektonického vzoru MVC. Je



Obrázek 4.1: Vztah Spring a Spring Boot [36]

také možné využít ho pro tvorbu *RESTful* aplikace, což je cílem implementační části této práce.

4.1.2.1 Spring Boot

Nadstavbou k frameworku Spring je projekt Spring Boot. [34] Vztah mezi Spring a Spring Boot je zobrazen na obrázku 4.1. Byl vyvinut k usnadnění tvorby Spring aplikací. Představuje více „dogmatický“ (*opinionated*) pohled na Spring, což znamená, že od vývojáře očekává „standardní“ chování podle zaběhlých postupů. [35] Vývojáře za to ale odstiňuje od nutnosti (nejen) počáteční konfigurace a nabízí navíc některé funkce, jako *embedded server*, bezpečnost, metriky, kontrolu stavu aplikace a další. V případě, že výchozí konfigurace nevyhovuje, je možné si ji ručně přizpůsobit. [36]

Spring Boot také umožňuje sestavit aplikaci do podoby jednoho JAR souboru („*fat jar*“), který obsahuje vše, co je nutné k jejímu spuštění. Není tedy potřeba mít pro nasazení zprovozněný aplikační server, jako například Tomcat či GlassFish. Stále je ale možné aplikaci sestavit do WAR souboru a vlastní server použít.

4.1.3 Hibernate

Hibernate [37] je framework pro platformu Java, usnadňující persistenci dat. Jedná se o nástroj pro ORM (objektově-relační mapování), který se stará o transformaci objektů aplikace do podoby vhodné pro uložení do relační databáze, jejich uložení, a následné načtení a transformaci z databáze zpět na objekty. Způsob mapování objektů do databáze lze nastavit buď v konfiguračním XML souboru, nebo přímo ve zdrojovém kódu anotacemi.

Kromě usnadnění práce s databází je další výhodou Hibernate možnost jednoduché změny uložiště. Kód aplikace komunikuje s jeho API stejným způsobem, nehledě na typ databáze, ke které je připojen.

4.1.4 apiary/dredd?

4.1.5 MySQL

Persistence dat aplikace bude zprostředkována relační databází. Pro výběr databáze jsem se řídil jejich popularitou a podmínkou použití zdarma. Nejpopulárnější takovou databází je MySQL. [38] Aplikace nebude mít takové požadavky na persistentní vrstvu, aby nějak ovlivnily výběr databáze. Funkce, které budou potřeba, splňují všechny uvažované možnosti. Větší popularita MySQL bude znamenat lepší podporu komunity a snažší řešení problémů během implementace.

4.1.6 Apache Tomcat

Pro nasazení aplikace bude použit server Apache Tomcat. [39] Je to *lightweight* server, nenáročný na hardware a jednoduchý na zprovoznění.

V případě spuštění aplikace jako JAR souboru je v něm Tomcat automaticky zahrnutý.

4.2 REST API

V této sekci nejprve krátce vysvětlím pojmy API a REST API. Poté přistoupím ke konkrétnímu návrhu REST API pro aplikaci Deadlines.

4.2.1 API

API (*Application Programming Interface*) slouží k jednoduché a spolehlivé komunikaci mezi aplikacemi nebo moduly aplikace. API představuje kontrakt, kterým se *producent* zavazuje k nabízení svých služeb v dané zdokumentované podobě, jež mohou být využity *konzumenty*.

Autoři knihy *APIs: A Strategy Guide* [40] definují API takto:

API je v podstatě kontrakt. Ve chvíli, kdy tento kontrakt existuje, vývojáři tíhnou k využívání API, protože ví, že se na něj mohou spolehnout. Kontrakt zvyšuje důvěru, což zlepšuje použitelnost. Tento kontrakt také zefektivňuje spojení mezi producentem a konzumentem, protože rozhraní jsou dokumentována, jsou konzistentní a předvídatelná.

4.2.2 REST

Termín REST (REpresentational State Transfer) byl poprvé zaveden v roce 2000 Royem Fieldingem v jeho dizertační práci. [41] Ve zkratce se jedná o

koncept designu distribuované architektury. Definuje rozhraní, které je použitelné pro jednotný přístup ke zdrojům (resources). Těmi mohou být data nebo stavy aplikace, popsatelné konkrétními daty. Zdroje v kontextu REST jsou identifikované pomocí URI a komunikovány mezi klientem a serverem v podobě reprezentací v libovolné textové, strojově zpracovatelné podobě (například JSON nebo XML). [42]

Architektonický styl REST popisuje šest podmínek:

- Uniform Interface (Jednotné rozhraní)
- Stateless (Bezstavovost)
- Cacheable (Cachovatelnost)
- Client-Server
- Layered System (Systém z vrstev)
- Code on Demand (Kód na vyžádání)

Popis architektury REST není předmětem této práce, takže se mu dále nebudu věnovat.

4.2.3 REST API aplikace Deadlines

API aplikace budu dokumentovat ve formátu API Blueprint. [43] Ten představuje jednoduchý způsob, jak popsat zdroje, jejich *endpointy*, návratové hodnoty a další informace. Jeho syntaxe vychází z jazyka Markdown. [44]

Kompletní popis API, včetně požadovaného obsahu požadavků a možných odpovědí, je vypsán v příloze [....] a dostupný na webu Apiary[2]. Zde jen uvedu *endpoints* a popíšu jejich funkci.

Pokud není uvedeno jinak, všechna volání API musí být autentifikovaná použitím *HTTP Basic Access Authentication*. [45]

4.2.3.1 `/user [GET, POST]`

GET vrátí seznam všech uživatelů v systému.
POST vytvoří nového uživatele z poskytnutých dat.

Tento *endpoint* je přístupný i neautentifikovaným voláním.

[2] `http://docs.deadlines.apiary.io/`

4.2.3.2 /user/{user_id} [GET, PUT]

GET vrátí informace uživatele s ID `user_id`.
PUT upraví profil uživatele s ID `user_id` předanými daty.

4.2.3.3 /task [GET, POST]

GET vrátí seznam všech úkolů, které odpovídají zadanému filtru. Zobrazit lze jen úkoly, jichž je volající účastníkem.
POST vytvoří nový úkol z poskytnutých dat.

4.2.3.4 /task/{task_id} [GET, PUT]

4.2.3.5 /task/share/{task_id [POST]

4.2.3.6 /task/leave/{task_id} [POST]

4.2.3.7 /task/userrole/{task_id} [PUT]

4.2.3.8 /task/status/{task_id} [PUT]

4.2.3.9 /task/report/{task_id} [POST]

Reports work done on a task.

4.2.3.10 /offer/task [GET]

4.2.3.11 /offer/task/{task_offer_id} [POST]

4.2.3.12 /offer/group [GET]

4.2.3.13 /offer/group/{membership_offer_id} [POST]

4.2.3.14 /group [GET, POST]

4.2.3.15 /group/{group_id} [GET, PUT, DELETE]

4.2.3.16 /group/manager/invite/{group_id} [POST, DELETE]

4.2.3.17 /group/manager/offer/task [GET]

4.2.3.18 /group/manager/removetask/{group_id}/{task_id} [POST]

4.2.3.19 /group/admin/setmanager/{group_id}/{user_id} [POST]

4.2.3.20 /group/admin/setadmin/{group_id}/{user_id} [POST]

4.3 Architektura aplikace

4.4 Návrhový model

4.5 Databázový model

Kapitola 5

Implementace

Kapitola 6

Testování

Kapitola 7

Nasazení

Závěr

TODO závěr

Literatura

[1] Nielsen, J.: Usability 101: Introduction to Usability [online]. 2012-01-04, [cit. 2016-03-16]. Dostupné z: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

[2] Atlassian: JIRA Software [online]. © 2016, [cit. 2016-03-04]. Dostupné z: https://www.atlassian.com/software/jira/

[3] Atlassian: Open Source Project License Request [online]. © 2016, [cit. 2016-03-04]. Dostupné z: https://www.atlassian.com/software/views/open-source-license-request

[4] Atlassian: Community License Request [online]. © 2016, [cit. 2016-03-04]. Dostupné z: https://www.atlassian.com/software/views/community-license-request

[5] Mozilla Foundation: Bugzilla [online]. Poslední aktualizace 2015-12-22, [cit. 2016-03-04]. Dostupné z: https://www.bugzilla.org/

[6] Lang, J.-P.: Redmine [online]. Poslední aktualizace 2015-11-15, [cit. 2016-03-04]. Dostupné z: http://www.redmine.org/

[7] Trello Inc.: Trello [online]. © 2016, [cit. 2016-03-06]. Dostupné z: https://trello.com/

[8] What is Kanban? [online]. © 2009–2015, [cit. 2016-03-06]. Dostupné z: http://kanbanblog.com/explained/

[9] Trello Pricing [online]. © 2016, [cit. 2016-03-06]. Dostupné z: https://trello.com/pricing

[10] API Reference | Trello Developers [online]. © 2016, [cit. 2016-03-06]. Dostupné z: https://developers.trello.com/advanced-reference

[11] Barzooka: Trackie [online]. [cit. 2015-11-21]. Dostupné z: https://trackieapp.com/

[12] Fog Creek: FogBugz [online]. [cit. 2016-03-06]. Dostupné z: www.fogcreek.com/fogbugz/

[13] Agile Planning and Project Management [online]. ©1998–2016, [cit. 2016-03-06]. Dostupné z: https://www.mountaingoatsoftware.com/presentations/agile-planning-and-project-management

[14] Doist: todoist [online]. Version 723, [cit. 2016-03-07]. Dostupné z: https://en.todoist.com/

[15] Compare Todoist Free and Todoist Premium [online]. [cit. 2016-03-07]. Dostupné z: https://todoist.com/compareVersions

[16] API Documentation | Todoist Developer [online]. [cit. 2016-03-07]. Dostupné z: https://developer.todoist.com/

[17] Toodledo: Toodledo [online]. © 2004–2016, [cit. 2016-03-07]. Dostupné z: https://www.toodledo.com/

[18] Toodledo: Toodledo [online]. © 2004–2016, [cit. 2016-03-07]. Dostupné z: http://api.toodledo.com/3/index.php

[19] Google Inc.: Google Inbox [online]. [cit. 2016-03-07]. Dostupné z: https://www.google.com/inbox/

[20] Google Inc.: Google Calendar [online]. [cit. 2016-03-07]. Dostupné z: https://www.google.com/calendar/

[21] How to read reminders in google calendars - Stack Overflow [online]. Poslední aktualizace: 2016-02-21. Dostupné z: http://stackoverflow.com/questions/35534774/how-to-read-reminders-in-google-calendars

[22] User Stories: An Agile Introduction [online]. © 2003–2014. Dostupné z: http://www.agilemodeling.com/artifacts/userStory.htm

[23] Cohn, M.: *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, první vydání, ISBN 978-0321205681.

[24] Stellman, A.: Requirements 101: User Stories vs Use Cases [online]. Poslední aktualizace: 2009-05-03. Dostupné z: http://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases/

[25] Mlejnek, J.: Analýza a sběr požadavků [online]. © 2011. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-SI1/prednasky/BI-SI1-P03m.pdf

[26] Jim Arlow, I. N.: *UML 2 a unifikovaný proces vývoje aplikací*. Computer Press (CPress), první vydání, ISBN 978-80-251-1503-9.

[27] Čápka, D.: 4. díl - UML - Doménový model [online]. © 2011. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/uml/uml-domenovy-model-diagram/

[28] Constraints - UML Domain model - how to model multiple roles of association between two entities? [online]. © 2016. Dostupné z: http://stackoverflow.com/questions/36100826/uml-domain-model-how-to-model-multiple-roles-of-association-between-two-entiti

[29] java.com: Java + You [online]. 2016, [cit. 2016-03-20]. Dostupné z: https://www.java.com/en/

[30] What Is Java? [online]. 2007, [cit. 2016-03-20]. Dostupné z: http://searchsoa.techtarget.com/definition/Java

[31] Balík, M.: Programování v jazyku Java [online]. 2011, [cit. 2016-03-20]. Dostupné z: https://edux.fit.cvut.cz/oppa/BI-PJV/prednasky/pjv01uvod.pdf

[32] Spring Framework [online]. © 2016, [cit. 2016-03-20]. Dostupné z: https://projects.spring.io/spring-framework/

[33] Hanák, D.: Dependency injection (předávání závislostí) [online]. 2016, [cit. 2016-03-20]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/dependency-injection-navrhovy-vzor/

[34] Spring Boot [online]. © 2016, [cit. 2016-03-20]. Dostupné z: http://projects.spring.io/spring-boot/

[35] design - What is opinionated software? - Stack Overflow [online]. 29-04-2009, [cit. 2016-03-20]. Dostupné z: http://stackoverflow.com/questions/802050/what-is-opinionated-software

[36] Spring Boot – Simplifying Spring for Everyone [online]. 06-08-2013, [cit. 2016-03-20]. Dostupné z: https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone

[37] Hibernate ORM - Hibernate ORM [online]. [cit. 2016-03-20]. Dostupné z: http://hibernate.org/orm/

[38] DB-Engines Ranking - die Rangliste der populärsten Datenbankmanagementsysteme [online]. 04-2016, [cit. 2016-03-20]. Dostupné z: http://db-engines.com/de/ranking

[39] Apache Tomcat - Welcome! [online]. © 1999–2016, [cit. 2016-03-20]. Dostupné z: http://tomcat.apache.org/

[40] Daniel Jacobson, D. W., Greg Brail: *APIs - A Strategy Guide*. O'Reilly Media, Inc., první vydání, ISBN 978-1-449-30892-6.

[41] Architectural Styles and the Design of Networbased Software Architectures [online]. 2000, [cit. 2016-03-21]. Dostupné z: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[42] Intro to REST (aka. What Is REST Anyway?) [online]. 30-05-2012, [cit. 2016-03-21]. Dostupné z: https://www.youtube.com/watch?v=llpr5924N7E

[43] API Blueprint | API Blueprint [online]. © 2013–2016, [cit. 2016-03-21]. Dostupné z: https://apiblueprint.org/

[44] Daring Fireball: Markdown [online]. © 2002–2016, [cit. 2016-03-21]. Dostupné z: http://daringfireball.net/projects/markdown/

[45] RFC 2617 - HTTP Authentication: Basic and Digest Access Authentication [online]. © 1999, [cit. 2016-03-21]. Dostupné z: https://tools.ietf.org/html/rfc2617#section-2

Příloha A

Seznam použitých zkratek

HW Hardware

SW Software

MVC Model-View-Controller

Příloha B

Splnění kritérií rešerše

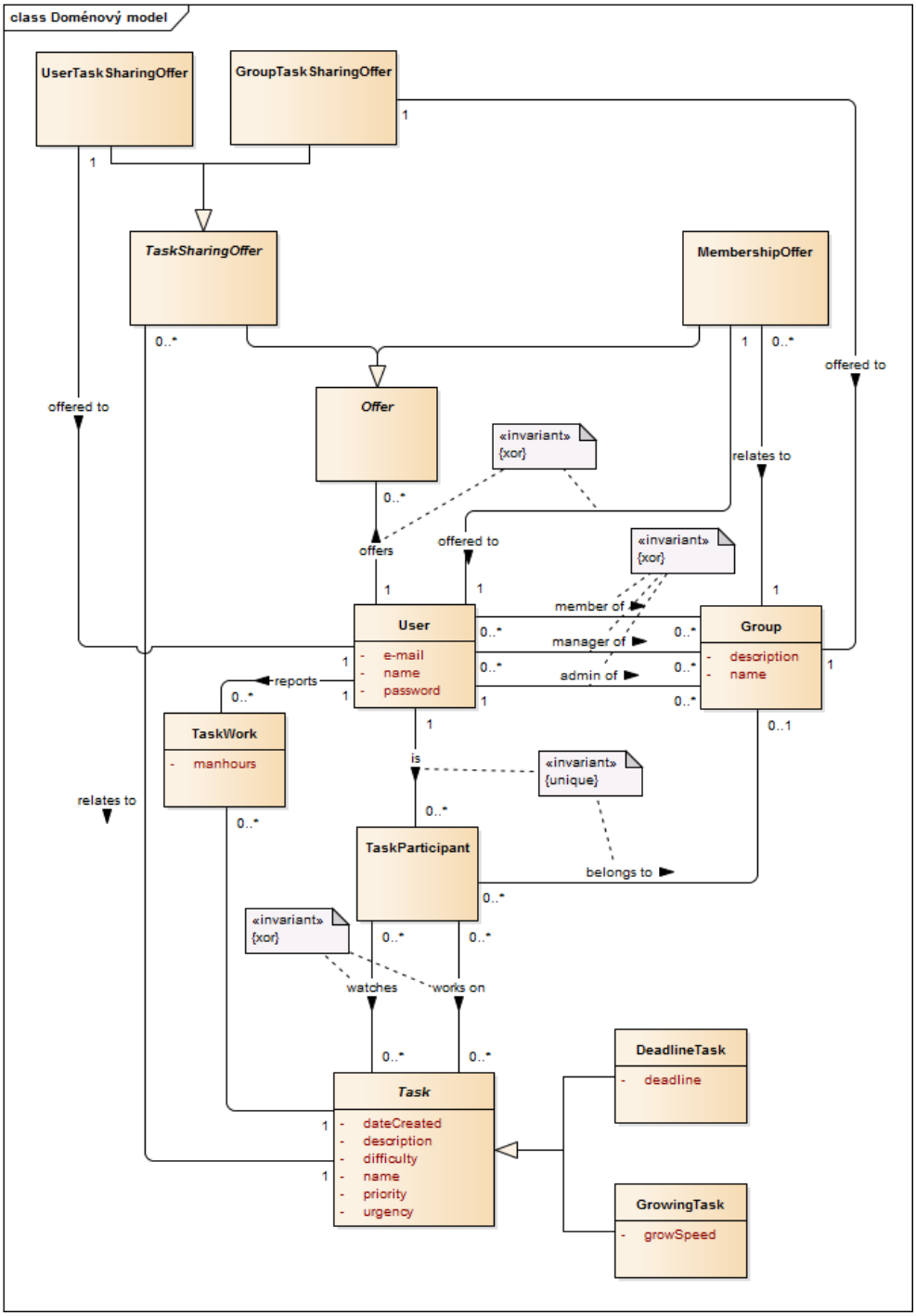
Tabulka B.1: Tabulka splnění kritérií rešerše

	Kr. 1	Kr. 2	Kr. 3	Kr. 4	Kr. 5	Kr. 6
JIRA	+				+	
Bugzilla	+			*	+	+
Redmine	+			*	+	+
Trello	+	+	+	+	+	
Trackie	+	+	+			
FogBugz	+	+			+	
Todoist	+	+	+	+	+	
Toodledo	+		+	**	+	
Inbox	+	+		+		
Poznámky	+	+	+	+	N/A	N/A

[+] Splněno
[*] Zdarma na vlastním HW.
[**] Zdarma jen bez spolupráce na úkolech
Kr. 1 – Nevyžaduje vlastní infrastrukturu, rychlé zprovoznení
Kr. 2 – Použitelnost
Kr. 3 – Absence nepotrebné funkcionality
Kr. 4 – Použití zdarma
Kr. 5 – REST API
Kr. 6 – Open-source

Příloha C

Doménový model



Obrázek C.1: Doménový model aplikace Deadlines

Příloha D

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