ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace pro sledování SW projektů
Student:	Adam Valenta
Vedoucí:	Ing. Michal Petřík
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je návrh a implementace webové aplikace pro sledování SW projektů dle metodiky zadavatele (Profinit EU, s.r.o.).

Pokyny pro vypracování:
1. Analyzujte metodiku a nástroje používané pro řízení projektů ve společnosti Profinit.
2. Analyzujte potřebné integrace na interní systémy zadavatele.
3. Navrhněte architekturu a GUI požadované aplikace.
4. Zvolte technologie pro 3vrstvou web aplikaci a využijte je v implementaci.
5. Vytvořte dokumentaci a aplikaci otestujte.
6. Zhodnoťte řešení s ohledem na možný budoucí rozvoj.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 13. října 2017

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

Bakalářská práce

Webová aplikace pro sledování SW projektů

Adam Valenta

Vedoucí práce: Ing. Michal Petřík

2. ledna 2018

Poděkování

Děkuji vedoucímu této práce Ing. Michalu Petříkovi za bezproblémovou komunikaci, cenné rady, trpělivost a vřelý přístup při psaní této práce. Také děkuji firmě Profinit EU, s.r.o. za poskytnuté zdroje a možnost spolupráce. Děkuji mé přítelkyni za podporu během celého studia a za pomoc při jazykové korektuře této práce a v neposlední řadě také děkuji celé své rodině, bez které bych tyto řádky neměl možnost psát.

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 2. ledna 2018 .....

Odkaz na tuto práci

Valenta, Adam. *Webová aplikace pro sledování SW projektů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Práce se zabývá návrhem a implementací webové aplikace, která má za cíl zlepšit již existující zadavatelem (Profinit EU, s.r.o.) používané řešení pro řízení rozsahu projektů a sledování odvedené práce. Požadavkem řešení je dodržení stávající metodiky zadavatele a integrace na existující interní systémy. Výstupem práce je analýza, návrh a implementace webové aplikace, která pokryje výše uvedené požadavky. Aplikace je postavena na třívrstvé architektuře a implementována v programovacím jazyce Java spolu s technologií Spring Boot, Vaadin a MyBatis.

Klíčová slova Java, Webová aplikace, Spring Boot, projektové řízení, Vaadin, MyBatis, Work Breakdown Structure, Burndown Report

Abstract

The thesis deals with design and implementation of a web application. The purpose of the application is to improve the current solution of the project scope management and project tracking in the company Profinit EU, s.r.o. further called as the contracting authority. The requirement of the solution is to follow the current methodology of the contracting authority and to integrate with internal systems. The output of the work is analysis, design, and

implementation of a web application that meets the requirements mentioned above. The application uses a three layer architecture and is implemented in Java programing language with usage of Spring Boot, Vaadin, and MyBatis technologies.

Keywords Java, Web application, Spring Boot, project management, Vaadin, MyBatis, Work Breakdown Structure, Burndown Report

Obsah

Seznam obrázků

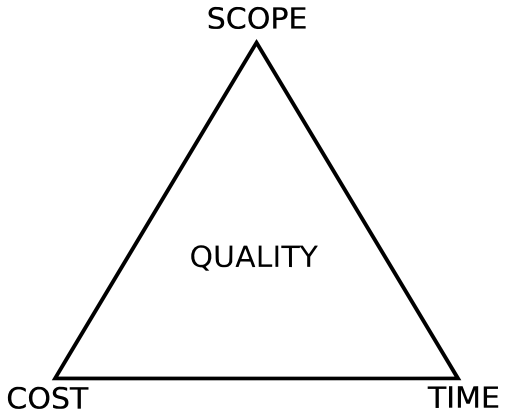
Úvod

Každá fungující firma podnikající v jakémkoli oboru dennodenně řeší nespočet úloh. Musí zajistit svůj provoz, svoje příjmy, svoje výnosy, svoji pozici na trhu atp. Úlohy se svým rozsahem, časovou náročností a v neposlední řadě i způsobem zadání zpravidla velmi liší. Existují úlohy, které jsou jasné, je pro ně známo efektivní řešení a může je snadno zvládnout jednotlivec. Naproti tomu nejsou žádnou výjimkou úlohy, jejichž zadání se vejde do jedné věty, ale o to více je velkou neznámou jejich rozsah, způsob řešení a nákladnost provedení. Takové druhy úloh můžeme za jistých níže uvedených okolností nazývat projekty. Jednou z takových úloh může být i potřeba vývoje nějakého softwaru.

Softwarové firmy se s poptávkou na řešení takových úloh potýkají dnes a denně. Tyto úlohy dříve nebo později dostávají na stůl lidé nazývaní projektoví manažeři. Jsou to odborníci, kteří svým zákazníkům pomáhají sjednotit myšlenky, sestavit jednoznačné zadání, řídit realizaci projektu a ve správném (zákazníkem schváleném) poměru udržet rozsah, cenu, kvalitu a termín dodání viz. obrázek 0.1.

Jednou z mnoha činností projektového manažera při řízení softwarového projektu je plánování rozsahu, kontrolování prováděných činností souvisejících s projektem a v neposlední řadě i řízení kapacit. Zadavatel pro detailní sledování rozsahu projektu využívá vlastní šablony nad aplikací MS Excel (dále jen Excel), které obsahují WBS rozpad daného projektu. Náplní této práce je seznámení se s metodikou zadavatele, analýza stávajícího nástroje a tvorba webové aplikace na platformě Java. Řešení musí splnit klíčové vlastnosti:

- Vytvářet a zobrazovat přehledy nových projektů.
- Vytvářet úkoly v přehledech.
- Editovat úkoly v přehledech.



Obrázek 0.1: Projektový trojimperativ [1]

- Propojit přehledy s interním systémem na sledování utilizace za účelem sledování stavu projektu.

Kapitola 1

Cíl práce a motivace

Cílem práce je získání aplikace, která bude vyhovovat speciálním firemním požadavkům a bude podporovat stávající metodiku zadavatele. To znamená provést analýzu a získat podrobnou představu zadavatele o výsledné aplikaci. Na základě analýzy provést návrh architektury a grafického uživatelského rozhraní a dle návrhu implementovat aplikaci, která bude náležitě otestována a zdokumentována.

1.1 Motivace

Motivací pro práci je především nedokonalost stávajícího řešení. Stávající řešení využívá nástroj MS Excel především pro jeho jednoduchost a rychlost možného nasazení. V Excelu je vytvořena šablona, která je po několika menších úpravách použita na projektu pro jednodušší definování rozsahu a sledování odpracovaných činností. Jak už bylo zmíněno, toto řešení má své nedostatky, které budou v následujícím textu detailně probrány.

1.1.1 Nejednotnost

První z nevýhod tohoto řešení je nejednotnost. Šablony jsou pouze návrh a nijak manažera neomezují ve způsobu jakým je použít. V praxi se to projevuje tak, že soubory od různých manažerů mají různou strukturu, nemusí splňovat základní pravidla pro tvorbu WBS apod. Jedním z požadavků je získat jednotné řešení problematiky skrze všechny projekty.

1.1.2 Decentralizace dat

Dalším problémem, který je spojený s nejednoznačností, je decentralizace dat. Zadavatel má mnoho projektů, pro každý projekt jeden soubor a v souboru jsou nejednotně uspořádaná data. Není možné se v datech organizovaně orientovat, efektivně v nich vyhledávat a centrálně je spravovat. Od řešení se tedy

očekává, že data všech projektů budou uspořádána na jednom místě a bude možné se v nich systematicky orientovat.

1.1.3 Lokální soubory

Je známo, že Excel vytváří lokální soubory. S lokálními soubory, i když jsou dostupné na sdíleném úložišti, může v daném okamžiku pracovat (editovat) pouze jeden uživatel, což způsobuje problémy při týmové práci. Nemluvě o tom, že všichni uživatelé mohou vše upravovat a nelze je v tom významně omezit. Při práci na dálku (z domova nebo u zákazníka) jsou soubory dostupné pouze skrze VPN připojení a nelze k nim jednoduše přistoupit například z mobilního telefonu. Nabízí se možnost soubory zkopírovat na svůj disk, tím ale riskujeme jejich neaktuálnost.

1.1.4 Problémy s rozhraním do Oracle databáze

Šablona má možnost pomocí ODBC rozhraní získávat data ze systému pro vykazování odpracovaných činností. Rozhraní se bohužel nedodává standardně k Excelu a musí být instalováno samostatně. Navíc musí být zvoleno to správné na základě druhu a verze OS a Excelu, což znepříjemňuje práci se šablonou.

Kapitola 2

Co je to projekt?

Tato kapitola se zabývá projekty a projektovým řízením v obecné rovině ve vztahu k běžné činnosti člověka v libovolném průmyslovém odvětví.

2.1 Projekt

V obecné rovině není vůbec snadné projekt definovat, v různých oborech je totiž slovo projekt vnímáno odlišnými významy. Navzdory praxi se o definici slova projekt snaží různé mezinárodní asociace či instituty. Například Mezinárodní asociace IPMA® definuje projekt následovně:

> *„Projekt je jedinečný a časově, nákladově a zdrojově omezený proces realizovaný za účelem vytvoření definovaných výstupů (rozsah naplnění projektových cílů) v požadované kvalitě a v souladu s platnými standardy a odsouhlasenými požadavky.“* [2, strana 13]

Ovšem i tato definice je velmi široká a mohly by ji splnit i činnosti, které se svou komplexností jeví jako projekt, např. dosažení určité úrovně vzdělání jednotlivce nebo položení 5 m^2 zámkové dlažby vlastníma rukama před vlastním domem. Organizace si tedy položily otázku: „Kdy činnost splňuje projektový charakter natolik, že je ku prospěchu věci, aby byla řízena jako projekt?“ Převedeno do IT oblasti: „Kdy je aplikace dostatečně veliká, aby se její vývoj řídili jako projekt?“ Dle [3] nám k zodpovězení této otázky mohou pomoci tzv. projektová kritéria:

- **Jedinečnost cíle**

Není to rutinně opakovaná akce. Odlišnost může být v prostředí, lokalitě, atd.

- **Vymezenost**

Termín, rozpočet, zdroje, legislativa, ...

- **Potřeba realizace projektovým týmem**

Potřeba několika pracovníků někdy i různých specializací a oborů.

- **Komplexnost a složitost**

Nejedná se o triviální problém.

- **Nadprůměrné riziko**

Daná věc se v daných podmínkách ještě nerealizovala, je omezen čas, peníze i zdroje, podílí se na problému celá řada různých lidí a je to komplikované, a je tedy vždy dost možné, že se něco pokazí.

2.2 Projektové řízení

Projektové řízení je uvažováno jako soubor norem, doporučení a rad pro koordinování jednotlivých činností v projektu, které povedou k jeho úspěšnému dokončení. Steve McConnell ve své knize Software Project Survival Guide [4] o úspěšném projektu říká:

> *„A successful project should be one that meets its cost, schedule, and quality goals within engineering tolerances and without padding its schedule or budget.“* [2, strana 13]

Projektové řízení vzhledem k různorodosti projektů neobsahuje žádné konkrétní postupy, jedná se spíše o všeobecně platné skutečnosti a určitou filozofii přístupu k řešení problematiky tak, aby bylo dosaženo požadovaných výstupů. Instituce PMI® [5] dělí řízení projektu do pěti základních oblastí:

1. zahájení,
2. plánování,
3. realizace,
4. sledování,
5. ukončení.

Na jednotlivé kroky se vážou činnosti, které je potřeba provést. Teoretická část této práce se nebude zabývat podrobným popisem každé z těchto oblastí, včetně všech činností prováděných v příslušných oblastech. V dalších sekcích se budeme zabývat pouze činnostmi v oblastech, které budou klíčové pro vyvinutou aplikaci. Konkrétněji tato práce poskytne zadavateli aplikaci, která si klade za cíl zjednodušit činnosti plánování rozsahu, harmonogramu, zdrojů a rozpočtu obsažené v kroku plánování a činnosti sledování a vyhodnocování obsažené v oblasti sledování. Vše taktéž bude respektovat interní procesy zadavatele.

2.3 Plánování rozsahu

Plánování je na projektu velmi často podceňovanou oblastí, a to jak u členů týmu, kteří budou provádět realizaci, tak u vedoucích projektů. Členové týmu by nejraději okamžitě začali s realizací a plánování realizace považují za zbytečné zdržení. Vedoucí pracovníci si naopak někdy ani neuvědomí, zvláště pokud nemají dostatečné technické znalosti, že čas strávený plánováním může pomoci kvalitě výstupu.

Navzdory přesvědčení projektového týmu praxe ukazuje, že oprava chyb způsobených špatným plánováním (neplánováním) může stát 50 až 200 krát více prostředků oproti odhalení chyby při plánování a vyřešení ještě před realizací. V případě neplánování můžeme považovat za chybu i odlišné představy zákazníka a projektového týmu. [4, strana 36]

Steve McConnell navíc říká, že při plánování projektu je možné odhalit i proveditelnost projektu.

> *„Planning is so critical to the success of a project that some experts report that a project's ultimate success or failure is determined as early as 10 percent of the way through the project.“* [4]

2.3.1 WBS dokument

Jedním ze způsobů plánování je tvorba WBS dokumentu (Work Breakdown Structure). Dle [6] dokument obsahuje hierarchický rozpad cíle projektu na jednotlivé dodávané výstupy. WBS je strom, v jehož kořeni je cíl projektu, v jeho uzlech jednotlivé dodávané výstupy a pod výstupy a na nejnižší úrovni jsou prvky, jež nazýváme pracovní balíky. Důležité je upozornit na základní pravidla pro tvorbu WBS.

- **100% práce**

Dokument musí obsahovat všechny dodávané výstupy, tedy všechnu práci, která se bude muset odevzdat.

- **Výstupy se nepřekrývají**

Práce vykonaná v jednotlivých uzlech se nesmí překrývat. Tato situace by mohla zapříčinit duplicitní konání práce.

- **Dokument obsahuje pouze výstupy**

Důležité je si uvědomit, že dokument má obsahovat pouze výstupy a ne činnosti, tedy jak výstupů dosáhnout. Lze ho připodobnit k nákupnímu seznamu. Ten obsahuje položky, které nakoupit, nikoliv jak je nakoupit.

V projektu plánujeme také činnosti. Vstupem pro plánování činností je WBS

dokument, ovšem pouze prvky na jeho nejnižší úrovni - pracovní balíky. Ty už z logiky věci musí obsahovat nějaké činnosti, které povedou k dodání příslušného pracovního balíku. Na činnosti se poté vážou informace o tom, jak dlouho daná činnost potrvá a kolik bude stát peněz. Na základě těchto doplňujících informací můžeme vytvořit harmonogram projektu.

Kapitola 3

Fáze sledování projektu

Při realizaci projektu je manažer zodpovědný za sledování činností projektového týmu a porovnávání skutečného průběhu realizace s plánem realizace. Sleduje se zejména množství reálně odpracovaného času s plánovanou pracností. Nedílnou součástí sledování je i komunikace s projektovým týmem a řešení vyskytujících se problémů. Projektový manažer by také měl v průběhu realizace znát odpovědi na základní otázky. Příkladem může být otázka: „Půlka rozpočtu projektu je vyčerpaná, je v projektu vykonáno alespoň 50% rozsahu?“ nebo „Je vyčerpáno 50% času, je také vykonáno odpovídající množství práce?“.

Motivací pro sledování projektu je zpravidla výskyt situací, kdy člen projektového týmu řeší úlohu odhadnutou na delší časový interval (např. 10 MD). Vžijme se do role projektových manažerů, kteří nesvědomitě sledují projekt. Den před odevzdáním se dozvíme, že se úloha nestihne, protože se na začátku vyskytl problém a dosud se ho nepodařilo vyřešit. Den před odevzdáním už z pozice manažera můžeme jenom zanedbatelně minimalizovat riziko posunutí termínu, kdežto druhý den bychom s projektovým týmem mohli navrhnout možná řešení nebo přiřadit další zdroje.

U takto rozsáhlých úloh se také špatně určuje stupeň rozpracování. Úloha by v ideálním případě měla být dále nedělitelnou jednotkou malého rozsahu. Jisté je, že se bude snadněji určovat stupeň rozpracování u úloh odhadnutých na 2 MD, tzn. na konci pracovního dne zhodnotit zda bude stačit už jenom jeden pracovní den, než stupeň rozpracování u úloh odhadnutých na 10 MD, kde je potřeba určit zda postačí další týden.

Důkladné a pravidelné sledování tedy může včas odhalit mnohá rizika, jako je posunutí termínu, nedostatek zdrojů apod. Pro sledování se uplatňuje několik metodik. V této práci bude popsána metoda analýzy dosažené hodnoty (EVM) a princip WIP.

3.1 Analýza dosažené hodnoty

Dle [7] je analýza dosažené hodnoty, označovaná jako EVM (Earned value management), metodou pro sledování a odhadu stavu projektu. Metodiku lze využít pro sledování projektu s mnoha úkoly a činnostmi z časového i finančního hlediska.

Princip metodiky spočívá v zavedení ukazatelů: [8]

PV (Planned value) - plánovaná hodnota,

AC (Actual cost) - aktuální náklady,

BAC (Budget at completion) – plánovaná výše rozpočtu,

POC (Percent of completion) - procento dokončené práce,

EV (Earned value) - získaná hodnota,

CV (Cost variance) - odchylka od nákladů,

CPI (Cost Performance Index) - index nákladových výkonů,

SV (Schedule variance) - odchylka od časového plánu,

SVI (Schedule performance index) - index výkonnosti z pohledu časového plánu,

EAC (Estimate at completion) - odhad nákládů na dokončení projektu,

ETC (Estimate to complete) - odhad zbývajících nákladů.

Hodnoty prvních třech ukazatelů získáme z plánu projektu a vykázané práce členů týmu. Ukazatele lze sledovat v peněžních jednotkách nebo v jednotkách úsilí (např. odpracované hodiny), je však nutné používat stejnou jednotku pro všechny ukazatele. Hodnotu POC typickou u softwarového projektu získáme odhadem. Ostatní hodnoty získáme následujícím výpočtem.

3.1.1 Určení hodnoty EV

Earned value je procento dokončené práce z celkového rozpočtu.

$$EV = POC * BAC$$

Hodnota EV je dále použita ve výpočtech všech ostatních ukazatelů. [7]

3.1.2 Určení hodnot CV a CPI

$$CV = EV - AC$$

Hodnota CV se interpretuje:

- $CV > 0$ projekt šetří (nedočerpává rozpočet),
- $CV = 0$ projekt čerpá rozpočet podle plánu,
- $CV < 0$ projekt přečerpává rozpočet.

Podobný pohled přináší hodnota CPI udávající relativní odchylku od nákladů.

$$CPI = \frac{EV}{AC}$$

Hodnotu CPI interpretujeme jako:

- $CPI = 1$ náklady odpovídají množství získané hodnoty,
- $CPI < 1$ náklady jsou vyšší, než odpovídá množství získané hodnoty,
- $CPI > 1$ náklady jsou nižší, než odpovídá množství získané hodnoty.

Hodnota se zpravidla převádí na procenta. Pokud hodnota vyjde $0,1$ lze říct, že projekt překročil plánovaný rozpočet o 10%. Analogicky hodnota $1,1$ znamená, že projekt ušetřil 10% plánovaných nákladů. CPI se využívá pro porovnání projektu s ostatními projekty v portfoliu. Jistě bude veliký rozdíl, pokud bude projekt s milionovým rozpočtem přečerpávat o 50 000 Kč, než pokud o tu samou částku přečerpá projekt se sta tisícovým rozpočtem. [7]

3.1.3 Určení hodnot SV a SPI

Hodnoty SV a SPI se získají obdobně jako CV a CPI, s tím rozdílem, že se na projekt dívají z pohledu časového plánu.

$$SV = EV - PV$$

Hodnota CV se interpretuje:

- $SV > 0$ Práce postupuje rychleji, než bylo plánováno,
- $SV = 0$ Práce postupuje podle plánu,
- $SV < 0$ Práce postupuje pomaleji, než bylo plánováno.

Obdobně jako v předchozím případě získáme index jako podíl.

$$SPI = \frac{EV}{PV}$$

Hodnotu SPI interpretujeme jako:

- $SPI = 1$ práce postupuje v souladu s časovým plánem,
- $SPI < 1$ práce je vykonávána pomaleji než bylo předpokládáno,
- $SPI > 1$ práce je vykonávána rychleji než předpokládal plán.

Opět je pravidlem hodnoty převést na procenta a např. při hodnotě $0,8$ mluvit o 20% zpoždění nebo při hodnotě $1,1$ mluvit o 10% předstihu. Hodnota SPI se stejně jako CPI využívá pro porovnávání projektů mezi sebou. [7]

3.1.4 Určení hodnot EAC a ETC

Pro získání představy konce projektu, pokud by probíhal současným tempem, slouží hodnoty EAC a ETC.

$$EAC = \frac{BAC}{CPI}$$

Hodnota říká, kolik projekt celkově spotřebuje zdrojů, pokud se bude realizovat současným tempem. Z této hodnoty se pak následovně získá hodnota zbývajících nákladů na dokončení.

$$ETC = EAC - AC$$

Jedná se tedy o rozdíl odhadu nákladů na dokončení projektu a aktuálních nákladů. [7]

3.2 Princip Work In Progress

Work In Progress, někdy také označovaný jako Work In Process a známý pod zkratkou WIP, je označení pro množství všech zdrojů vložené do dosud rozdělané ale nekompletní práce na libovolném produktu. Do zdrojů se započítávají jednak suroviny pro výrobu a jednak i cena odvedené práce zaměstnanců a další režijní náklady. Produkt může mít několik fází, kterými ve vývoji prochází a WIP reprezentuje hodnotu do něj vkládaných zdrojů, což zprůhlední postup realizace tím, že je vidět aktuální stav: kolik je odhadnuto, jaká část je odpracována a kolik ještě zbývá. [9]

Zdroje mohou být do WIP zahrnovány buďto jako náklady na množství odvedené práce nebo jako náklady na provedený proces. Ve vývoji software to typicky bývá druhý způsob. Práce na vývoji software se provádí v několika úkonech, a to: analýza, návrh, implementace, testování,... a každý z těchto úkonů spotřebovává zdroje. WIP poskytne koncept pro popis toku zdrojů mezi jednotlivými fázemi využitými pro vývoj produktu. [10]

3.3 Burndown report

Burndown report je graf zobrazující vztah mezi zbývající prací a zbývajícím časem potřebným pro dokončení projektu. Podle [11] burndown report poskytuje snadno pochopitelnou vizualizaci pokroku projektu a lze z něj odhadnout termín dokončení při pevně daném rozsahu projektu. Množství práce lze měřit v libovolných jednotkách (peníze, MD, MH,...).

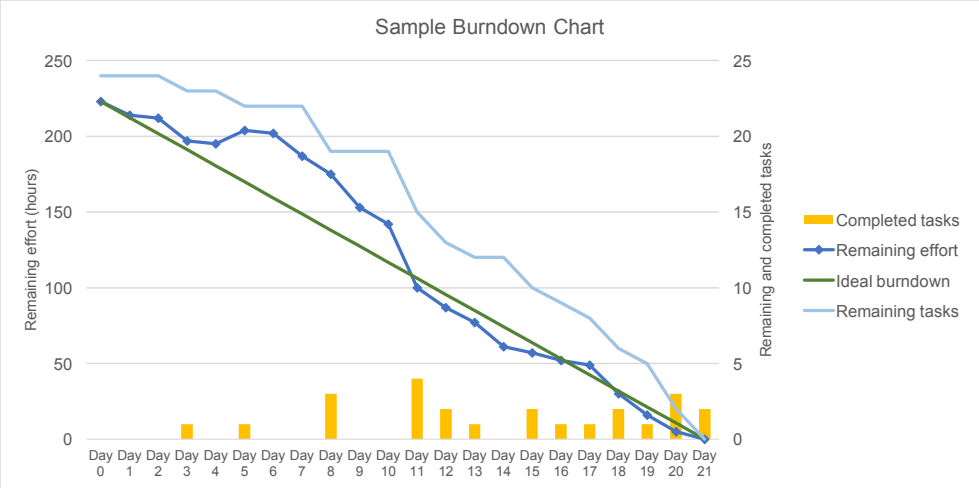
Jednotka času se volí na základě velikosti projektu. V praxi se projekt většího rozsahu dělí zpravidla do několika stejně trvajících iterací (např. 3 týdenní iterace), které dodávají nějaký zvolený funkční celek. Má tedy smysl množství zbývající práce sledovat po dnech. Lze také sledovat projekt jako celek a v tu chvíli je možné sledovat zbývající práci v řádů týdnů nebo měsíců.

3.3.1 Princip

Na ose Y se nachází množství zbývající práce a na ose X je čas. První bod se získá jako $[0, celkovéMnožstvíPráce]$ a poslední bod jako $[termínDokončení, 0]$. Základem grafu jsou dvě křivky: Ideální průběh a skutečný průběh. Křivka ideálního průběhu vznikne jako spojnice prvního a posledního bodu. Skutečný průběh je zobrazením historie projektu.

Pokud je křivka skutečného průběhu nad ideální křivkou, znamená to, že projekt je pozadu oproti plánu a lze očekávat zpoždění. Naopak je-li skutečná křivka pod ideální křivkou, znamená to, že realizace probíhá rychleji než bylo plánováno. Jak je vidět na grafu 3.1, lze sledovat i jiné hodnoty ve vztahu ke zbývající práci. Například počet zbývajících úkolů.

Pro projektového manažera je zpravidla výhodné průběh skutečné práce proložit regresní přímkou a odhadnout tak trend, jakým se projekt ubírá na základě jeho historie. Pokud byl projekt například v minulosti napřed oproti plánu, ale data z posledních dní se objevují nad ideální křivkou, tak regresní přímka ukáže, kdy by mohl projekt skončit, pokud by práce probíhaly jako doposud.



Obrázek 3.1: Ukázka burndown reportu [12]

Kapitola 4

Současný stav

Tato část bude popisovat dosavadní stav řešení problematiky a dosud používané nástroje v prostředí zadavatele.

4.1 Metodika

Zadavatel pro plánování úloh využívá princip WIP. Vstupem pro něj je WBS rozpad cíle projektu. K listům WBS stromu jsou vytvořeny úlohy potřebné pro realizaci. K úlohám jsou přidány odhady jejich pracnosti. Uvedený počet hodin je brán jako závazný odhad a projektový tým na něj musí brát zřetel. Interní pravidlo říká, že úloha by neměla trvat déle než 2 MD, tedy 16 hodin. Dále se u úloh definuje priorita, zodpovědná osoba, stav činnosti (hotovo, probíhá), milník, zbývající čas a čas strávený na úloze. Zbývající čas a čas strávený na úloze se sčítá a porovnává s originálním odhadem, což pak dává informaci o tom, v jakém vztahu je skutečná práce a odhad.

4.2 Používané nástroje

Zadavatel má v Excelu vytvořenou šablonu dokumentu. V šabloně jsou předvyplněna záhlaví tabulek a definovány vzorce pro výpočet hodnot (např. rozdíl odhadovaného a reálně stráveného času). Dalším nástrojem je systém Profis, který v organizaci slouží k vykazování odpracovaného času zaměstnance na dané úloze. Šablona využívá rozhraní ODBC do databáze Oracle, skrze které získává data z Profisu a vypočítává odpracovaný čas viz. 4.3. Projektový tým tedy vykazuje strávený čas pouze na jednom místě.

Mimo vykazování v Profisu ostatní členové týmu do takto připraveného dokumentu, pokud je to možné, doplňují stavy řešených úloh a v případě potřeby upravují čas potřebný k dokončení. Pokud se čas potřebný k dokončení činnosti liší od odhadu, manažer tuto informaci uvidí a má možnost řešit nastalou situaci.

												Utilizace:
	Priorita	Stav	Název činnosti (popis je ve sloupci F)	Osoby	TM	Hotovo [%]	Celkem	Odpracováno [čd]	Zbývající pracnost [čh]	Původní odhad [čh]	Rozdíl [čd]	KEYNOTE
			PM									
		Hotovo	Obecné PM	MH	OST	100%	5,0	0,63	0,00	8,00	-0,4	
		Probíhá	PKM3	VJ	OST	38%	6,0	0,75	10,00	16,00	0,0	IGNORE
		Hotovo	PKM3	VJ	OST	100%	7,0	0,88	0,00	5,00	0,3	IGNORE
			Analýza									
	1	Hotovo	PKM3	MH	TEST	90%	9,0	1,13	1,00	5,00	0,6	IGNORE
	1	Probíhá	PKM3	VJ	TEST	42%	5,0	0,63	7,00	12,00	0,0	IPKM3ZRAnalyza
			Design									
	1	Hotovo	PKM3	MH	TEST	0%	0,0	0,00	18,00	18,00	0,0	IGNORE

Obrázek 4.1: Ukázka současného řešení

Pokud tým nemá přístup k dokumentu (např. pracuje u zákazníka a nemá přístup na sdílené úložiště), tak úlohy aktualizuje projektový manažer.

4.3 Utilizace

Projektový manažer má možnost definovat vše, co je popsáno v sekci 4.1. U každé úlohy je navíc možné vyplnit čtyři hodnoty:

- zakázka,
- část zakázky,
- podčást,
- keynote.

Hodnoty těchto atributů slouží jako jednoznačný identifikátor úlohy v systému Profis. Rozhraní databáze dodá odpracované hodiny včetně těchto čtyř ukazatelů a zařadí je ke správné úloze.

Kapitola 5

Analýza

Na základě analýzy současného řešení a komunikace se zadavatelem byly identifikovány požadavky na novou aplikaci. Možnost tvorby harmonogramu byla zadavatelem určena jako nepotřebná pro tuto fázi projektu a nebude do nového řešení prozatím zahrnuta. Nové řešení bude navíc oproti současnému podporovat tvorbu WBS rozpadu a generování burndown reportu.

5.1 Funkční požadavky

F01 - Tvorba nových projektů - Uživatel s příslušným právem bude mít v aplikaci možnost vytvořit projekt.

F02 - Tvorba WBS stromu projektu - Systém bude uživateli dávat možnost vytvořit WBS rozpad.

F03 - Zobrazení projektu - Uživatel s příslušným právem bude moci vidět detailní zobrazení projektu.

F04 - Tvorba úloh v projektu - V projektech bude možné vytvářet úlohy.

F05 - Vazba úloh v projektu na listy WBS - Nové úlohy mohou být vytvořené pouze na listech WBS.

F06 - Zobrazování celkového součtu sledovaných hodnot - Systém bude zobrazovat celkové součty u sledovaných hodnot (Celkový odhad, celkový odpracovaný čas apod.). Bude možné zobrazit součty i jednotlivých podstromů projektu.

F07 - Filtrování úkolů v projektu - Úkoly v projektu mohou být filtrovány. Bude možné zobrazit úkoly pouze některých podstromů WBS a bude možné hledat úlohy obsahující konkrétní text. Při vyhledávání podle textu

není nutné přepočítávat celkové součty.

F08 - Sledování historie projektu - Systém bude zaznamenávat časy změn hodnot v projektu. Konkrétně to budou hodnoty určující originální a aktuální odhad.

F09 - Generování Burndown grafu - Systém na základě historie a uživatelem zadaného období vygeneruje burndown report projektu.

F10 - Zobrazit všechny projekty v aplikaci - Uživatel s příslušným právem bude mít možnost zobrazit přehled všech projektů.

F11 - Vyhledávat projekty v aplikaci - Uživatel bude moci vyhledávat konkrétní projekty.

F12 - Řízení uživatelského přístupu - Systém bude přístupný pouze po zadání příslušného jména a hesla.

F13 - Podpora importu dat ze systému Profis - Systém bude schopný importovat data ze systému Profis ve formátu CSV.

5.2 Nefunkční požadavky

NF01 - Webová aplikace - Aplikace musí být přístupná a ovladatelná ve webovém prohlížeči.

NF02 - Třívrstvá architektura s použitím MVC - Aplikace musí být postavena na třívrstvé architektuře a musí dodržovat návrhový vzor MVC nebo jeho obdobu (MVP, MVVM,...).

NF03 - Data uložena v databázi - Veškerá data musí být uložená v databázi.

NF04 - Spustitelná bez nutnosti nasazení na serveru - Aplikaci bude možné využívat jednak jako klasickou webovou aplikaci nasazenou na serveru a jednak jako desktopovou aplikaci bez nutnosti nasazení.

5.3 Uživatelské role

V této fázi projektu budou všichni do aplikace přihlášení uživatelé disponovat administrátorskými právy. Později v systému budou vystupovat uživatelé ve třech rolích.

5.3.1 Administrátor

Uživatel, který má přístup do každého okna aplikace a má právo na všechny úkony, které lze provést.

5.3.2 Projektový manažer

Projektový manažer může vytvářet projekty a ve svých vytvořených projektech má neomezená práva. Může vidět všechny projekty v aplikaci, ale může vidět detaily pouze svých projektů nebo projektů, kterých je součástí.

5.3.3 Developer/tester

Vidí a může zobrazit pouze projekty, kterých je součástí. Nemůže projekty vytvářet. Může upravovat pouze úkoly, u kterých je projektovým manažerem přiřazen jako řešitel. Může vyhledávat pouze v projektech, kterých je součástí.

Kapitola 6

Možnosti řešení

6.1 Existující řešení

Jedna z možnost řešení je například využití již existujících softwarů zabývajících se touto problematikou, kupříkladu program Microsoft Project. Microsoft Project ovšem vyžaduje nemalou investici: pro cloudové řešení Microsoft Project je to 46,40 euro měsíčně za uživatele [13]. Pro firmu čítající přibližně 450 zaměstnanců je to investice v řádu milionů ročně. Navíc není možné Microsoft Project přizpůsobit současné metodice zadavatele. Také by bylo možné využít existujících issue tracking systémů, případně dodatečných pluginů, které by byly schopné požadavky splnit.

6.2 Vlastní řešení

6.2.1 Databáze

Otázka, na kterou je potřeba znát odpověď, je, zda využít relační nebo grafovou (NoSQL) databázi. Dle analýzy dat uložených v současných souborech je zřejmé, že bez pochybností lze použít relační databázi. Relační databáze jsou velmi rozšířené a jejich princip je v oboru dobře známý a existuje pro ně nespočet implementací, které lze využít pro vývoj (H2, Derby,...) nebo ostré nasazení (MySQL, Postgres, Oracle,...).

Problém konverze mezi tabulkami relační databáze a objektově orientovaným programovacím jazykem se bude řešit knihovnami pro objektově relační mapování, dostupnými pro zvolený programovací jazyk. Zadavatel má možnost využít databázi Oracle a aplikace bude v budoucnu tuto databázi využívat. V současné verzi projektu je využita databáze H2 [14] v souborovém režimu. Projekt je v počáteční fázi a nejsou v něm žádná skutečná podniková data a použitím H2 lze začít snadno a rychle vyvíjet bez nutnosti instalace serveru.

Relační datový model je vytvořen v programu Enterprise Architect, který nabízí možnost generování DDL skriptu pro různé databáze včetně Oracle a

H2. Model je tedy odstíněn od konkrétní implementace a lze tak v budoucnu pohodlně přejít na Oracle databázi či jakoukoli jinou databázi, pro kterou lze v EA generovat skript.

6.2.2 Programovací jazyk

6.2.2.1 PHP

Zadavatel požaduje webovou aplikaci postavenou na třívrstvé architektuře schopnou ukládat data v databázi. Toho lze docílit kombinací technologií Apache, MySQL a PHP ve spolupráci s knihovnou Symfony 2 a knihovnou Doctrine 2 pro objektově relační mapování. Vznikne tím aplikace postavená na open-source technologiích ([15],[16], [17]) s možností snadného nasazení na cenově dostupném hostingu. [18] Symfony 2 navíc obsahuje buid-in server [19], který lze spustit z příkazové řádky a tím by tedy byl splněn i požadavek NF04 v 5.2.

6.2.2.2 Java

Další možností by mohla být technologie Java. V kombinaci s open-source knihovnou Spring [20], což je knihovna usnadňující vývoj podnikových aplikací, lze dostat nástroj pro tvorbu robustních aplikací se silnou uživatelskou základnou.

Vzhledem k velikosti projektu je možné využít i technologii Spring Boot. Knihovna Spring vyžaduje při startu projektů náročnou konfiguraci a zpravidla trvá delší dobu vytvořit první funkční prototyp. Knihovna Spring Boot je nadstavba nad Spring, která toto nevyžaduje, neboť je ve svém základu nastavená tak, „aby vše fungovalo“. [21] Má v sobě dokonce zabudovaný aplikační server a aplikaci je tak možné spustit i bez konfigurace aplikačního serveru. Zároveň ve Spring Bootu je možné nastavit vše co lze nastavit ve Spring. Je to tedy plnohodnotný Spring, ve kterém je ve svém základu nakonfigurovaná funkční aplikace.

Pro ORM lze využít knihovny Hibernate nebo MyBatis a pro uživatelské rozhraní lze použít knihovny PrimeFaces nebo Vaadin.

6.2.2.3 Zvolený programovací jazyk

Po dohodě se zadavatelem byla zvolena technologie Java ve spolupráci s knihovnou Spring Boot, MyBatis a H2 databáze (později Oracle). Uživatelské rozhraní bude zajišťovat knihovna Vaadin 7. Programovací jazyk Java byl zvolen zejména pro dosavadní zkušenosti a zaměření zadavatele a řešitele. Ze stejného důvodu byla zvolena i knihovna Spring Boot a navíc i pro splnění požadavku NF04 v 5.2.

Na databázovém serveru Oracle jsou již interní systémy nasazené a lze ho tedy bez větších licenčních či finančních problémů použít. Knihovna Hy-

bernate byla zamítnuta pro její velkou komplexnost a téměř úplné odstínění programátora od relační databázové vrstvy a bylo rozhodnuto nasadit MyBatis, který programátorovi umožňuje alespoň trochu zasáhnout do mapovacího mechanismu a mít nad ním větší kontrolu.

Knihovna Vaadin byla zvolena díky velkému počtu komponent, které lze použít bez většího zásahu do nich. Konkrétně jsou to komponenty pro zobrazení stromové struktury [22] a zobrazení tabulky [23]. Vaadin aplikace jsou také ve svém základu navržené tak, aby se daly bez problémů ovládat pomocí klávesnice, čímž se zhodnotí komfort uživatele při používání aplikace.

Kapitola 7

Návrh

7.1 Návrh obrazovek aplikace

Před samotnou implementací byl vytvořen prototyp aplikace. Ten pomohl předvést možnou podobu jednotlivých oken aplikace, umístění použitých grafických komponent a jejich nejpravděpodobnější vzhled bez nutnosti psaní zdrojového kódu.

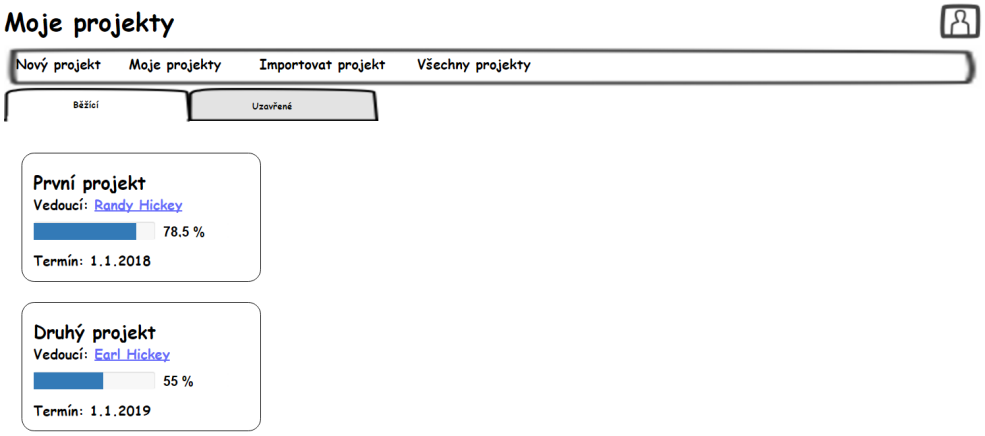
Pro návrh obrazovek a komponent byl použit nástroj Justinmind Prototyper [24]. Nástroj dává uživateli možnost zdarma vytvořit návrh aplikace, do které je možné vložit odkazy na ostatní navrhovaná okna a simulovat tak zadavateli reálné chování aplikace. Nástroj dává možnost i exportu návrhu do HTML včetně odkazů apod., ovšem jen v placené verzi. Více informací lze najít na [25]. Na přiloženém CD je také umístěn soubor spustitelný v Justinmind Prototyper.

7.1.1 Přihlašovací okno

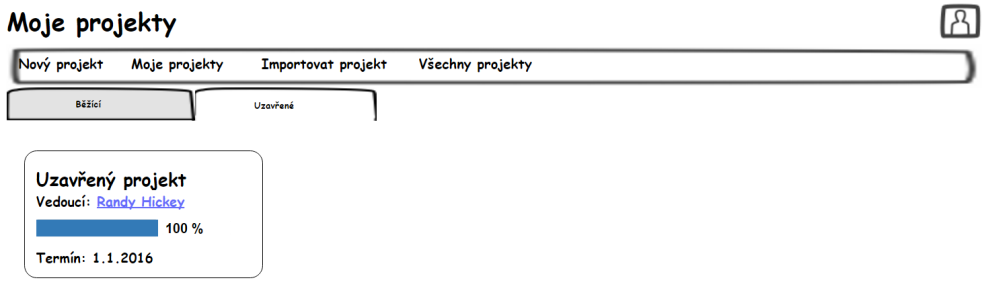
Obrázek 7.1: Přihlašovací okno

7.1.2 Okno přidělených projektu

Návrh okna, kde uživatel uvidí všechny projekty, u kterých je členem projektového týmu.



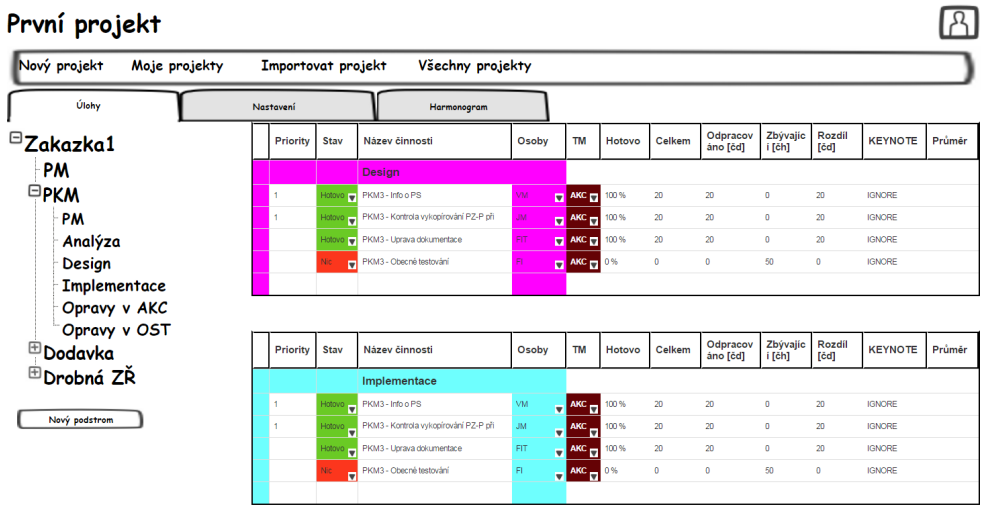
Obrázek 7.2: Okno přidělených a stále běžících projektů



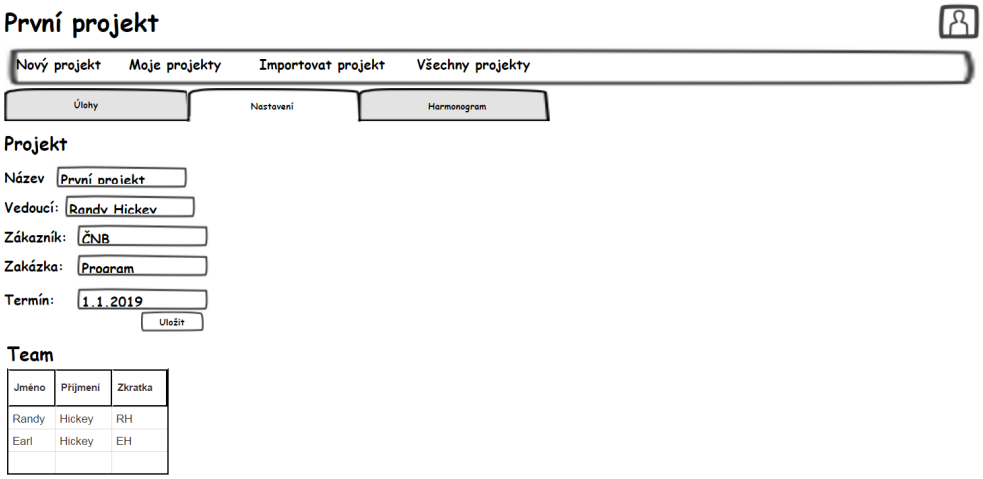
Obrázek 7.3: Okno přidělených a uzavřených projektů

7.1.3 Detail projektu

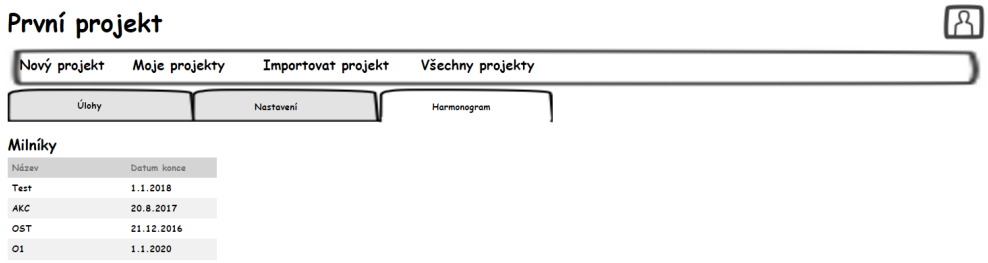
Návrh okna, kde bude zobrazen detail projektu. Na tomto místě by měla být veškerá funkcionalita, která je implementovaná v současném řešení pomocí nástroje Excel. Je zde i záložka harmonogram, kde by v budoucnu byla implementovaná tato funkcionalita.



Obrázek 7.4: Přehled úkolů



Obrázek 7.5: Nastavení



Obrázek 7.6: Harmonogram

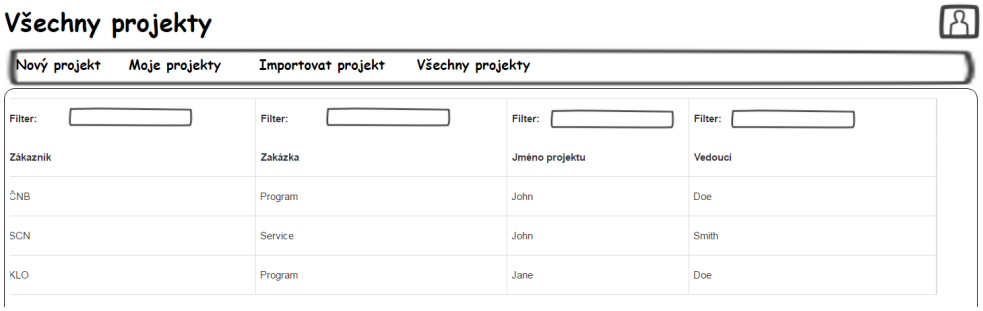
7.1.4 Detail úkolu

Návrh modálního okna, ve kterém budou zobrazeny všechny informace, které se vztahují ke konkrétnímu úkolu.

Obrázek 7.7: Detail úkolu

7.1.5 Přehled všech projektů

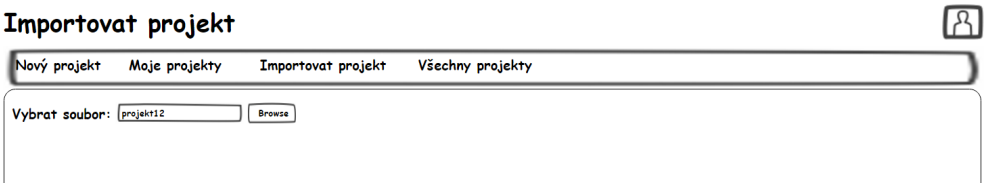
Okno zobrazující všechny projekty v aplikaci. Je zobrazena i možnost vyhledávání v projektech.



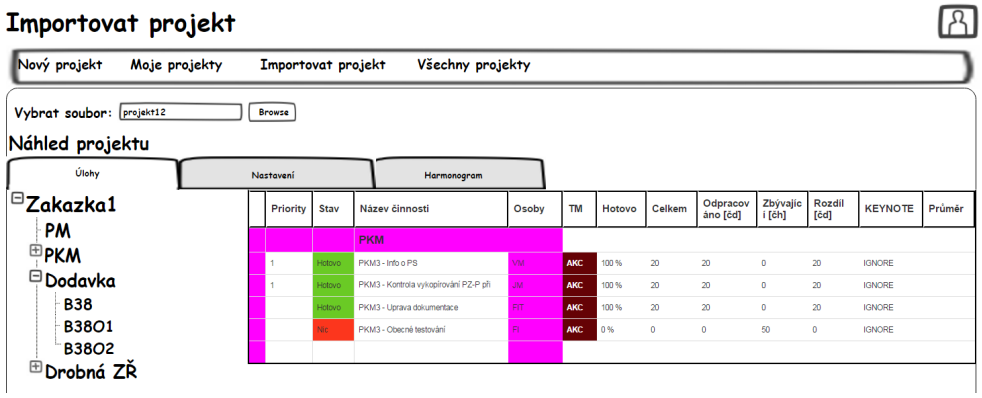
Obrázek 7.8: Přehled všech projektů

7.1.6 Importovat projekt

Návrh okna, ve kterém bude v budoucnu možné importovat projekt.



Obrázek 7.9: Import projektu - před specifikováním souboru



Obrázek 7.10: Import projektu - náhled importovaného projektu

7.2 Relační datový model aplikace

Jak již bylo zmíněno - relační datový model je vytvořen v programu Enterprise Architect. Kromě zřejmé výhody přehledné vizualizace datového modelu a tedy zlepšení celkové kvality dokumentace, lze navíc použitím EA částečně oddělit návrh modelu dat od konkrétního databázového serveru, čehož se s výhodou využije pro použití databáze H2 pro pilotní fázi projektu a implementaci a pozdější nasazení na databázový server Oracle.

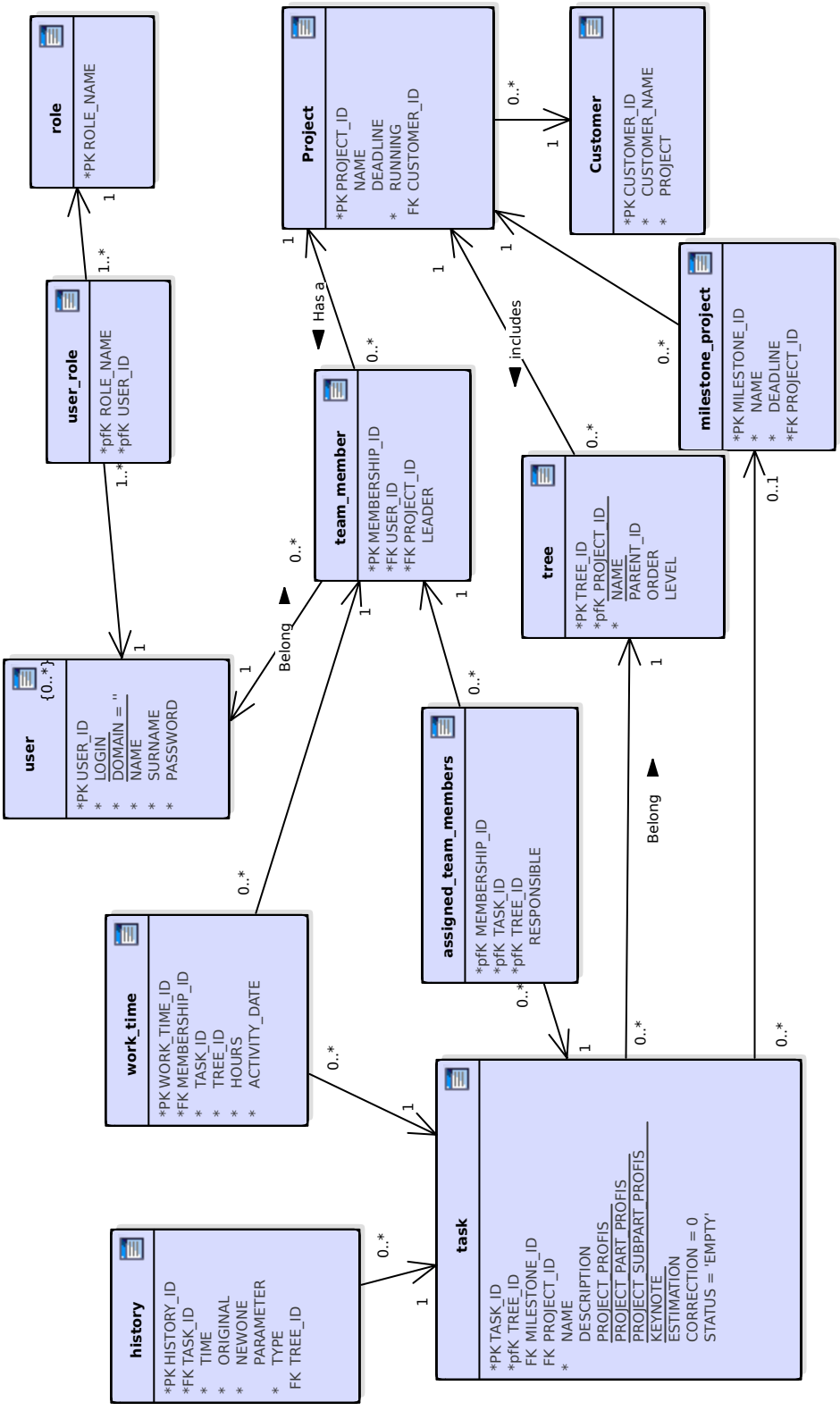
Při vytváření datového modelu bylo definováno několik jeho případů užití a na nich bylo demonstrováno a odladěno možné používání tohoto modelu. Pár ukázek z testovaných případů z EA:

- DMUC2 - Vypsat úkoly ve stromech s daným ID.
- DMUC7 - Přesun úkolu/ů mezi stromy.
- DMUC4 - Kolik je na úkolu/projektu odpracováno hodin.
- DMUC10 - Mazání úkolu bez historie a odpracovaného času.
- DMUC11 - Mazání již rozpracovaného úkolu.

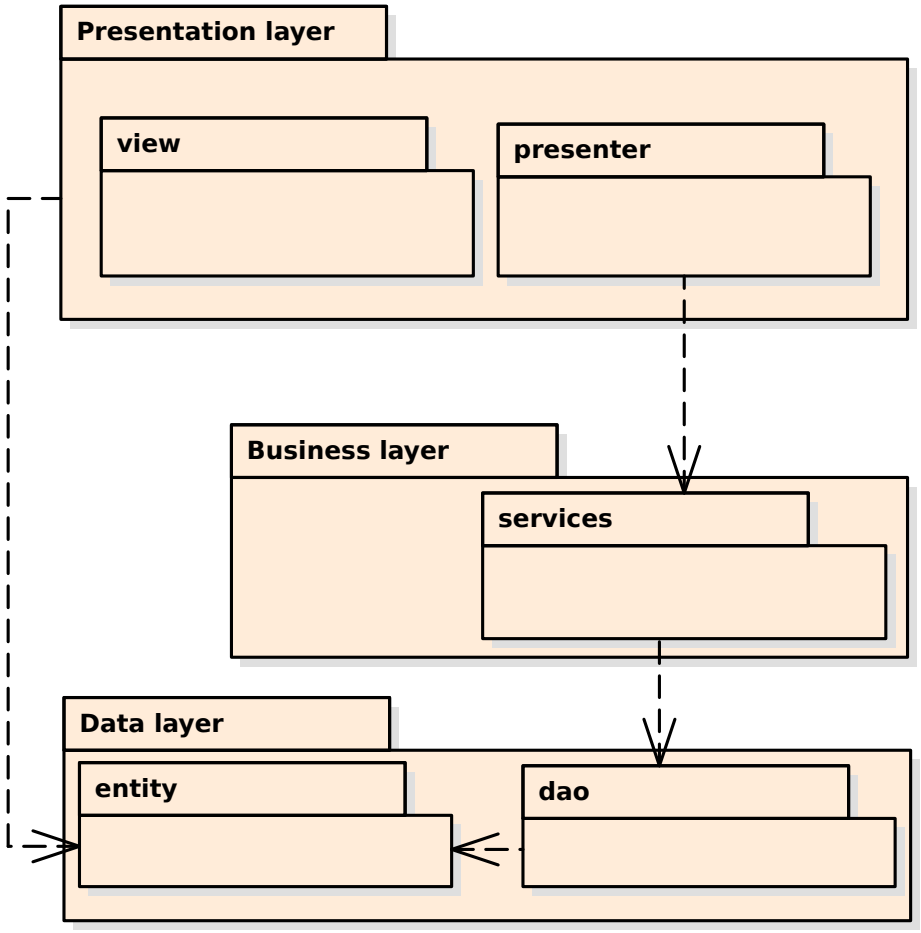
Za zmínku také stojí realizace konkrétních funkčních požadavků. Například požadavek F08 na sledování historie je splněn vytvořením tabulky `history`. Ta v sobě bude uchovávat historii změn, které se na jednotlivých úkolech udály. Pokud pro uživatele nemá smysl sledovat historii některých konkrétních změn, jako například změna názvu úkolu, změna KEYNOTE a jiné, zapíší se příslušné změny přímo do tabulky `task`.

Dále je požadavek F02 na tvorbu WBS rozpadu realizován tabulkou `tree`, která v sobě ukládá jednotlivé WBS stromy. Tyto stromy mají formou cizích klíčů odkazy na svoje předky, což umožňuje určit hierarchii mezi jednotlivými uzly. Navíc každý uzel má pomocné atributy ORDER a LEVEL. Atribut ORDER udává pořadí uzlu v daném podstromu. To v budoucnu dává možnost definovat v jakém pořadí se mají uzly vypisovat uživateli. Atribut LEVEL udává stupeň zanoření uzlu. Více o problematice ukládání stromových struktur v relačních databázích lze nalézt na [26].

Více uživatelský přístup vyplývající z požadavků (F01, F03, ...) je realizován pomocí tabulek `user`, `role`, `team_member`, ... Aplikace má tedy v současné implementaci možnost uživateli přidělit systémovou roli a určit k jakým konkrétním projektům, úkolům a částem aplikace bude mít přístup. Se zadavatelem byla diskutována i myšlenka rozdílných uživatelských rolí pro aplikaci a pro konkrétní sledovaný projekt. Toto vylepšení nebude podporováno v současné verzi aplikace, ale v budoucnu ho bude možné implementovat přidáním dodatečné tabulky s vazbou na `team_member` a v případně ještě větší granularity práv s vazbou na `assigned_team_members`.



Obrázek 7.11: Relační datový model



Obrázek 7.12: Diagram architektury

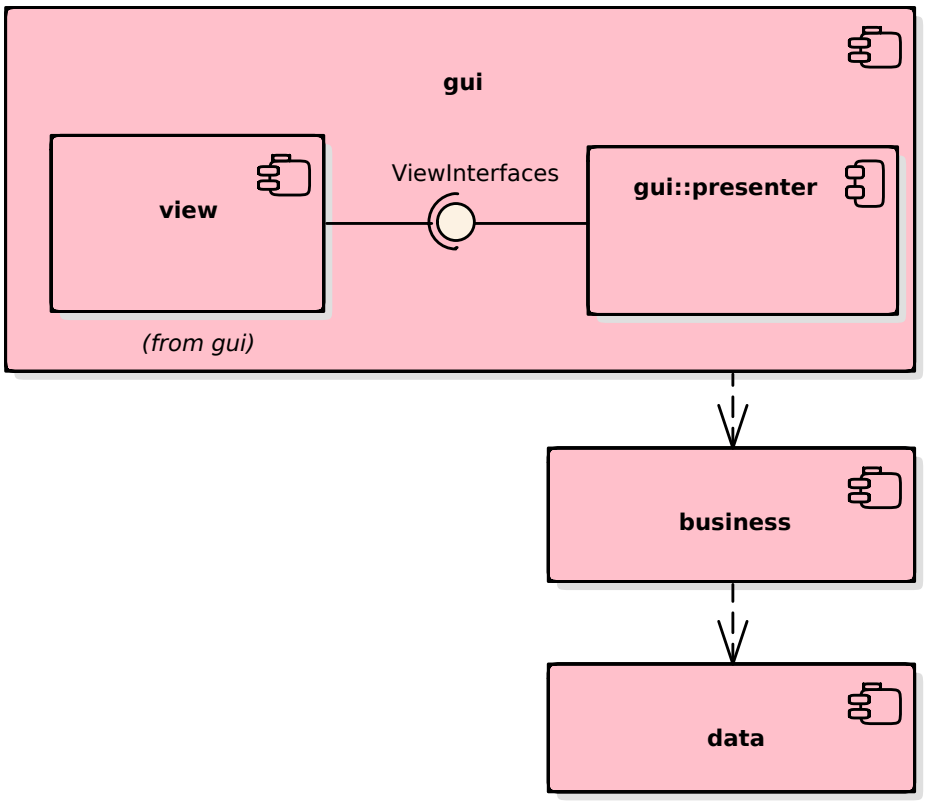
7.3 Architektura aplikace

Vzhledem k požadavku NF02 v 5.2 je aplikace postavená na třívrstvé architektuře s použitím MVP.

7.3.1 MVP

Návrhový vzor MVP je pro Vaadin knihovnu doporučený [27]. Princip spočívá v rozdělení prezentační vrstvy na komponenty - View a Preseter a business vrstvy na komponentu Model. Z obrázku 7.13 je zřejmé, že:

- View a Presenter spolu komunikují skrze rozhraní.
- Presenter je prostředník mezi View a Modelem.



Obrázek 7.13: Diagram komponent

- View a Model spolu nekomunikují přímo, ale pouze pomocí Presenteru.
- Model v aplikaci představují třídy balíku services.

V komunikaci mezi View a Presenterem lze uplatnit různé implementační přístupy. V této práci budou zmíněny dva.

7.3.1.1 Princip aktivního View

View má referenci na Presenter a Presenter má referenci na ViewInterface. Ve View je definice všeho, co uživatel vidí, a pokud View vyžaduje zobrazit nějaká data nebo předat data vložená uživatelem, zavolá si příslušnou metodu Presenteru a ten provede veškerou potřebnou komunikaci s vyšší vrstvou a pokud je to potřeba zavolá příslušnou metodu View pro zobrazení zpětné vazby uživateli. Výhodou tohoto přístupu je jednoduchost a přímočarost. Programátor jasně vidí, jaká metoda Presenteru se volá a má možnost snadno odhalit, co se děje na pozadí. Nevýhodou tohoto přístupu je málo striktní oddělení

View a Presenteru, což by mohlo způsobit problém při unit testování. Tento princip je v aplikaci používán a pro praktickou ukázku lze nahlédnout do tříd `LoginPresenter`, `LoginViewInterface` a `LoginView`. [28]

7.3.1.2 Princip pasivního View

Komunikace probíhá jako v předchozím případě s tím rozdílem, že View nemá referenci na Presenter. Rozhraní View poskytuje metody pro přidání posluchače na stisknutí tlačítek nebo ostatních událostí, které rozhraní definuje a Presenter má povinnost přidat svoje metody jako vykonavatele příslušných událostí. Výhodou tohoto přístupu je velice striktní oddělení View od logiky Presenteru. Je zřejmé, že takto implementované View by mohlo být použito na jakémkoli jiném projektu dokonce bez nutnosti implementace Presenteru (View by bylo statické). View a presenter lze testovat naprosto odděleně. Nevýhodou může být nepřímočarost - pokud programátor nahlédne do implementace View, tak nemá tušení, jaké metody Presenteru se ve skutečnosti volají. [29]

Kapitola 8

Realizace

8.1 Verzování

Aplikace je verzována pomocí softwaru Git. Zadavatel Git využívá pro verzování ostatních vyvíjených aplikací, má pro něj podporu ve své infrastruktuře a veškeré zdrojové soubory této aplikace mají povahu textových souborů a není tedy na místě přemýšlet např. o nástroji SVN, který se umí dobře vypořádat s verzováním binárních souborů.

8.2 Vytvoření projektu

Spring Boot projekt je vytvořen použitím Spring Boot inicializátoru [30]. Je to nástroj, ve kterém za pomoci uživatelského rozhraní zvolíme druh projektu (Maven, Groovy,...), identifikátor projektu a potřebné závislosti na ostatní technologie. Pro tento projekt byl zvolen buildovací nástroj Maven. Je hojně rozšířený, řešitel i zadavatel jsou s ním dobře seznámeni, je využívaný na nespočtu projektů a je tedy pro něj velké množství zdrojů dokumentace a velká komunita uživatelů, kteří ho používají. Více o Mavenu a jeho nastavení se zabývá sekce 8.4. Jako závislosti jsou specifikovány Vaadin, MyBatis a H2 databáze.

8.3 Konfigurace

V projektu jsou následující konfigurační soubory

8.3.1 Konfigurační soubor Springu

Soubor `application.properties` je konfigurační soubor Springu. V současnosti je aplikace postavena tak, že v případě nasazení bude nutné měnit konfiguraci pouze v tomto souboru. Projekt obsahuje dva takovéto soubory, jeden

pro aplikaci a druhý pro unit testy. Jejich konfigurace se liší pouze ve využití konkrétní databáze.

Spring také umožnuje předat konfigurační soubor již zkompilované aplikaci. Na [31] se mimo jiné udává, že Spring hledá `application.properties` ve složce, kde se uživatel nachází při spuštění aplikace a to ještě předtím, než použije konfigurační soubor zkompilovaný společně s aplikací. Toho se dá využít třeba při nasazování produkční verze aplikace nebo testovacího prostředí.

8.3.2 Konfigurační soubor Logu

V aplikaci je využit logovací nástroj Logback [32]. V jeho konfiguračním souboru `logback-spring.xml` je nastavení pro všechny loggery použité v aplikaci.

8.4 Maven

Maven je nakonfigurován tak, aby uživatel pouze s pomocí maven příkazů dokázal následující:

- Zkompilovat aplikaci příkazem `mvn clean install` nebo `mvn package`.
- Vytvořit H2 databázi i s počátečními daty příkazem `mvn sql:execute`.
- Vygenerovat javadoc příkazem `mvn javadoc:javadoc`.
- Spustil unit testy příkazem `mvn test`.
- Vytvořil soubor se styly pro Vaadin příkazem `mvn vaadin:compile-theme`.
- Zkompiloval Vaadin addony příkazem `mvn vaadin:update-widgetset`.
- Spustil aplikaci příkazem `mvn spring-boot:run`.

Je toho docíleno použitím pluginů pro jednotlivé funkcionality. V souboru `README.md` na přiloženém CD je příručka pro spuštění projektu, kde se mimo jiné píše o těchto příkazech a i o tom jak je použít.

Kapitola 9

Datová vrstva

9.1 Integrace knihovny MyBatis

Prvním krokem pro využití knihovny MyBatis v projektu bylo definování závislosti na knihovně a databázi při vytváření projektu viz. sekce 8.2. Jenom to ovšem k úspěšné integraci nestačí. V databázi je potřeba vytvořit schéma a v aplikaci je potřeba vytvořit konfigurační soubor, entitní třídy, Mappery a DAO třídy. V dalších částech bude popsán postup, jak úspěšně integrovat knihovnu do aplikace.

9.1.1 Vytvoření databáze

Nejprve bude vytvořena databáze. Konkrétně to bude soubor, ve kterém bude uloženo schéma i s daty. Při implementaci bylo pro vytvoření databáze využito pluginu DBeaver [33] pro Spring Tool Suite vývojové prostředí [34]. Je to plugin, který dokáže komunikovat s jakýmkoli databázovým serverem, který poskytuje JDBC driver. DBeaver se také dá pořídit jako java aplikace. Tato práce se nebude dále zabývat tímto nástrojem, ale bude popsán způsob, jak vytvořit databázi pomocí Mavenu.

9.1.1.1 Vytvoření databáze pomocí Mavenu

Pro vytvoření databáze bylo využito maven pluginu: `sql-maven-plugin`. Je to plugin, kterému specifikujeme JDBC driver, url do databáze, uživatelské jméno a heslo pod kterým bude plugin vystupovat a cesty k SQL skriptům, případně konkrétní SQL dotazy, které bude potřeba vykonat. Poté z příkazové řádky zavoláme příkaz `mvn sql:execute` a plugin vytvoří databázi a spustí vše, co bylo potřeba. Nutno dodat, že url je prostá cesta v souborovém systému. JDBC driver poté buďto najde soubor ve kterém je H2 databáze uložená nebo příslušný soubor vytvoří. Není tedy nutné nijak explicitně vytvářet databázový soubor. Implementace je na přiloženém CD v souboru `pom.xml` pod tagem `<plugins>`.

9.2 Příprava aplikace

9.2.1 Entitní třídy

ORM knihovny mapují tabulky v databázi na entitní třídy. Entitní třída představuje řádek konkrétní tabulky v databázi. Entitní třídy pro MyBatis a nejenom pro něj mají mít následující předepsanou podobu (Stejnou strukturu mají i entitní třídy pro Hibernate): [35]

- sloupec = atribut třídy,
- cizí klíč = atribut typu cílové entity nebo jejich kolekce,
- bezparametrický konstruktor,
- getr a setr pro každý atribut.

9.2.2 Mappery

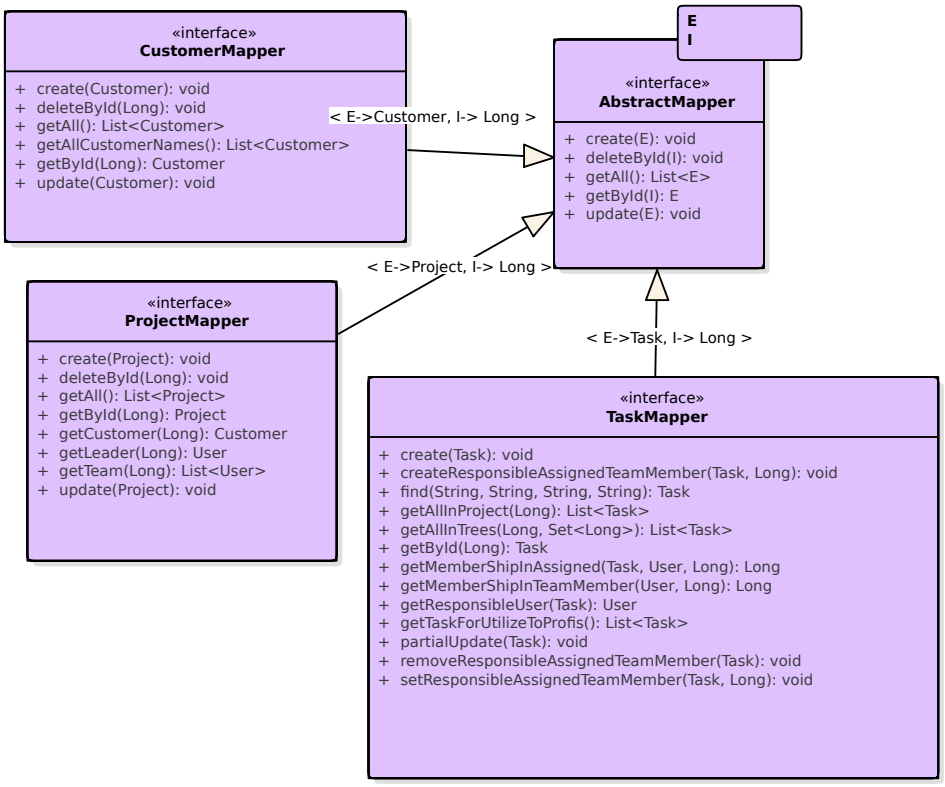
Mappery [36] jsou rozhraní pro každou entitu. Mappery definují jména metod, které bude možné volat. K metodám se doplní anotace ve kterých definujeme SQL dotaz prováděný v databázi. MyBatis tyto volání provede a pokud mají nějaké výstupy, tak je namapuje na entitní třídy. Tedy udělá přesně to, co lze očekávat od ORM knihovny. Pro účely projektu bylo vytvořeno rozhraní `AbstractMapper`. To definuje základní metody, které by měla být každá entita schopna provést. Toto abstraktní rozhraní dědí všechny ostatní mappery konkrétních entit viz. obrázek 9.1.

9.2.3 DAO

DAO jsou třídy, skrze které voláme Mappery. Opět platí, že pro každou entitu by měla být jedna třída DAO. DAO otevře Session do databáze, zavolá příslušné metody mapperu, předá výstupy ostatním třídám a relaci ukončí [37]. Všechny DAO třídy v této aplikaci implementují `BasicDAO` rozhraní, které má i svoji defaultní implementaci. Každá entita pro níž existuje DAO třída má tedy základní operace již zprovozněny.

9.2.4 MyBatisUtil

DAO třídy z předchozí sekce využívají pro komunikaci s databází Session. Tato Session je poskytována knihovnou MyBatis ve formě třídy faktory třídy `SqlSessionFactoryBuilder` [38], které předáme konfigurační soubor viz. 9.2.5. MyBatis vytvoří Session dle předaného konfiguračního souboru (url, JDBC driver,...) a při volání metody `openSession()` ji otevírá s požadovaným nastavením. Názornou ukázku lze nalézt ve třídě `MyBatisUtil` na přiloženém CD.



Obrázek 9.1: Diagram některých mapovacích tříd

9.2.5 Konfigurační soubor pro MyBatis

Soubor `mybatis-config.xml` je konfigurační soubor pro knihovnu MyBatis. V souboru specifikujeme:

- JDBC driver,
- url do databáze,
- přístupové údaje,
- balíček obsahující mappery nebo konkrétní rozhraní jednotlivě.

V projektu je dále ještě navíc specifikován balíček obsahují Type Handlery. To jsou třídy, jež mapují datové typy Javy na datové typy databázového serveru. V projektu je využit jediný Type Handler a to na mapování datového typu LocalDateTime na databázový typ Timestamp viz. soubor `LocalDateTimeTypeHandler.java`. Více informací o konfiguračních souborech lze nalézt v dokumentaci [39].

9.3 Integrace Profisu

Pro podporu importu dat ze systému Profis a tedy splnění požadavku F13 z 5.1 je použit JDBC driver pro CSV soubory [40]. Byla implementována třída `ProfisUtil`, která vytvoří Connection do CSV souboru. (V budoucnu implementováno vytvoření Connection do Oracle databáze). Při vytváření je předána cesta do složky s CSV soubory. Driver se chová k jednotlivým CSV souborům jako k tabulkám (Co jeden soubor to jedna tabulka) a hlavička CSV souboru se považuje za definici sloupců. Nad soubory lze vytvářet SQL dotazy jako nad databázovým serverem.

9.4 Struktura balíků datové vrstvy

- `Projektová složka`
  - `src`
    - `main/java` ..... Zdrojové kódy implementace
      - `ppt.data.util` ..... Deficice Session
      - `ppt.data.entity` ..... Entitní třídy
      - `ppt.data.dao` ..... DAO třídy
      - `ppt.data.mappers` ..... Mappery
      - `ppt.data.typehandlers` ..... TypeHandlery
      - `ppt.data.profis` ..... Třídy poskytující přístup k Profisu
    - `main/resources` ..... Konfigurační a ostatní soubory
      - `database` ..... DDL a datový skript
      - `profis` ..... Složka s csv soubory
      - `mybatis-config.xml` ..... Konfigurační soubor MyBatisu
      - `application.properties` ..... Konfigurační soubor Springu
    - `test/java` ..... Zdrojové kódy unit testů
      - `ppt.data.util` ..... Třída pro správu testovací DB
      - `ppt.data.dao` ..... Testy balíku DAO
    - `test/resources` ..... Konfigurace testu

Kapitola 10

Vrstva business logiky

Na této vrstvě není použita žádná knihovna třetích stran. Tato vrstva implementuje balík tříd, které slouží jako prostředník mezi datovou vrstvou a vrstvou uživatelského rozhraní. Dále je zde definice rozhraní tříd zprostředkovávajících uživatelský přístup.

10.1 Servisní třídy

K implementaci těchto tříd přistupují Presentery. Servisní třídy jím zprostředkují volání datové vrstvy a případné Exception vyhozené datovou vrstvou zaobalí do Exception bussiness vrstvy, které bude schopná prezentační vrstva odchytit a reprodukovat chybu uživateli.

10.2 Řízení uživatelského přístupu

Řízení uživatelského přístupu je implementováno pomocí Vaadinu. Na této vrstvě jsou pouze definice rozhraní a výčtových typů.

10.3 Struktura balíků business vrstvy

```
Projektová složka
└── src
    ├── main/java ..... Zdrojové kódy implementace
    │   ├── ppt.business.services ..... Třídy pro komunikaci s DAO
    │   ├── ppt.business.accesscontrol .... Kontrola uživatelského přístupu
    │   └── ppt.business.messages ..... Zprávy poskytované uživateli
    └── main/resources ..... Konfigurační a ostatní soubory
        └── messages ..... Jazykové soubory
```

Kapitola 11

Vrstva uživatelského rozhraní

Uživatelské rozhraní je zprostředkováno knihovnou Vaadin 7 a jejími komponentami převážně v jejich základním vzhledu definovaném Vaadinovským stylem Valo. Výjimkou je pouze komponenta AbstractGrid a TreeGrid, jejíž původní implementace byla rozšířena o pomocné metody.

11.1 Integrace knihovny Vaadin

Vaadin je na integraci s knihovnou Spring připraven a nemělo by to tedy způsobovat větší komplikace. Pokud je projekt vytvořen dle 8.2 tak jsou všechny závislosti připravené a je pouze nutné vytvořit základní UI třídu například dle tutoriálu na [41]. Pro jednotlivá okna aplikace se vytvoří třídy implementující rozhraní `View` a v těchto třídách bude definice vzhledu jednotlivých oken. Pro navigaci mezi konkrétními `View` používá Vaadin třídu `Navigator`. Instance třídy `Navigator` se předá vytvořenému UI a nastaví se jí příslušná `View`, na které je potřeba se v aplikaci odkazovat.

11.2 Integrace DCharts addonu

DCharts addon je oficiální addon pro Vaadin vyvinutý třetí stranou a šířený pod licencí Apache License 2.0 [42]. Doplněk se nachází v maven repozitáři s názvem Vaadin Add-ons. Pro jeho integraci je nutné přidat do `pom.xml` následující závislosti.

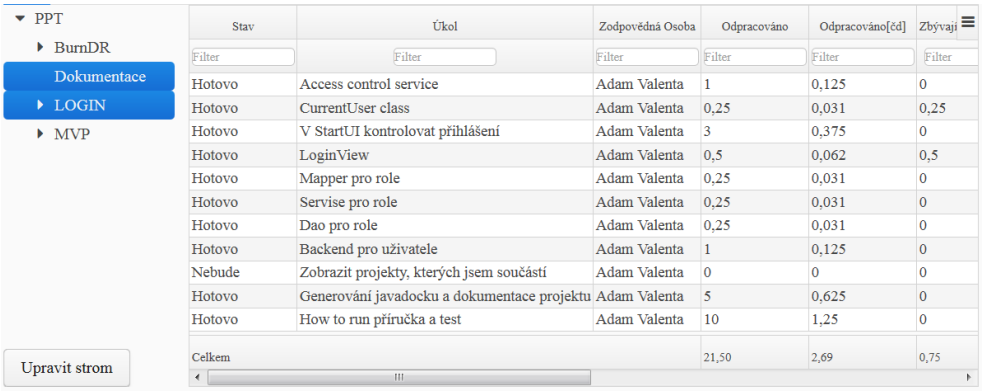
```
<dependency>
    <groupId>org.vaadin.addons</groupId>
    <artifactId>dcharts-widget</artifactId>
    <version>1.7.0</version>
</dependency>

<dependency>
    <groupId>com.google.gwt</groupId>
    <artifactId>gwt-servlet</artifactId>
    <version>2.8.2</version>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.1</version>
</dependency>

<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>commons-codec</artifactId>
</dependency>

<repositories>
    <repository>
        <id>vaadin-addons</id>
        <url>http://maven.vaadin.com/vaadin-addons</url>
    </repository>
</repositories>
```

Po zkompilování projektu již bude dostupná knihovna DCharts a bude možné využívat všechny její výhody. Více se touto knihovnou zabývá sekce 11.6.

11.3 Využívání vlastních stylů

Vaadinu lze předat vlastní definici stylů následujícím způsobem. Do UI třídy vytvořené v předchozí části přidáme anotaci `@Theme("mytheme")`. Tím Vaadin dostane informaci o tom, že nemá používat defaultní soubor se styly a má hledat stylový soubor ve složce `VAADIN/themes/mytheme/mytheme.scss`.

Soubor `mytheme.scss` má předepsanou strukturu. Je velice nepravděpodobné, že by v projektu bylo potřeba přepsat styly všech komponent Vaadinu. V souboru je tedy příkaz `@import "../valo/valo.scss";` který říká, že importuje styly z Vaadinovského stylu valo. Pokud tedy není specifikován vlastní



Obrázek 11.1: Vzhled komponenty TreeGrid použité v aplikaci

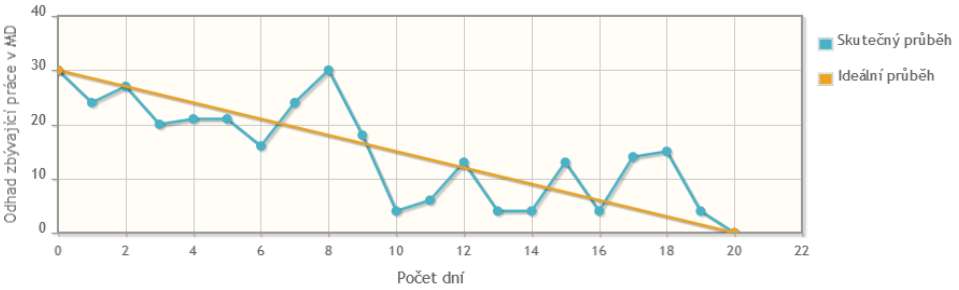
styl, tak Vaadin zvolí defaultní styl. Je také nutné do projektu přidat i zdrojové soubory stylu Valo. Provede se to obdobně jako při přidání stylu `mytheme`. Tedy vytvořením složky: `VAADIN/themes/valo/` a vložením příslušných zdrojových souborů do ní. Zdrojové soubory lze nalézt například zde [43].

11.4 Komponenta AbstraktGrid

Komponenta AbstraktGrid ve svém jádru používá Vaadinovskou komponentu Grid. Je to komponenta, která má za úkol zobrazovat uživateli data ve formě tabulky. AbstractGrid definuje metody pro snadnější používání Gridu. Přidává možnost jednoduché definice sloupců, které budou v tabulce zobrazovány. Dále možnost jednoduché definice sloupců, které bude možné filtrovat pomocí textového vyhledávání a metodu pro přidávání řádků do tabulky.

11.5 Komponenta TreeGrid

Komponenta TreeGrid je komponenta, která má zajišťovat zobrazování detailu projektu. Podobně jako je tomu na návrhu obrazovky 7.7. Obsahuje v sobě komponentu Tree, která je získána z Vaadin book [44]. Komponenta Tree v sobě uchovává WBS rozpad projektu. Dále TreeGrid obsahuje komponentu TaskGrid, která je implementací AbstraktGridu a slouží pro zobrazení úkolu. Na Tree jsou navázané údálosti kliknutí. Kliknutím myší nebo pomocí šipek a klávesy enter je možné v TreeGridu označovat jednotlivé uzly WBS stromu. Komponenta Tree z označených uzlů získá uzly nejnižšího řádu (listy) a ty pak předá k zobrazení TaskGridu. TaskGrid tedy vždy zobrazuje všechny úkoly, které spadají do označených uzlů. Pokud není vybrán ani jeden uzel, zobrazí se všechny úkoly v projektu. Je to tedy stejná situace jako kdyby se označil kořen WBS stromu. Na obrázku 11.1 je grafická podoba této komponenty.



Obrázek 11.2: Vzhled komponenty BurnDownChart použité v aplikaci

11.6 Komponenta BurnDownChart

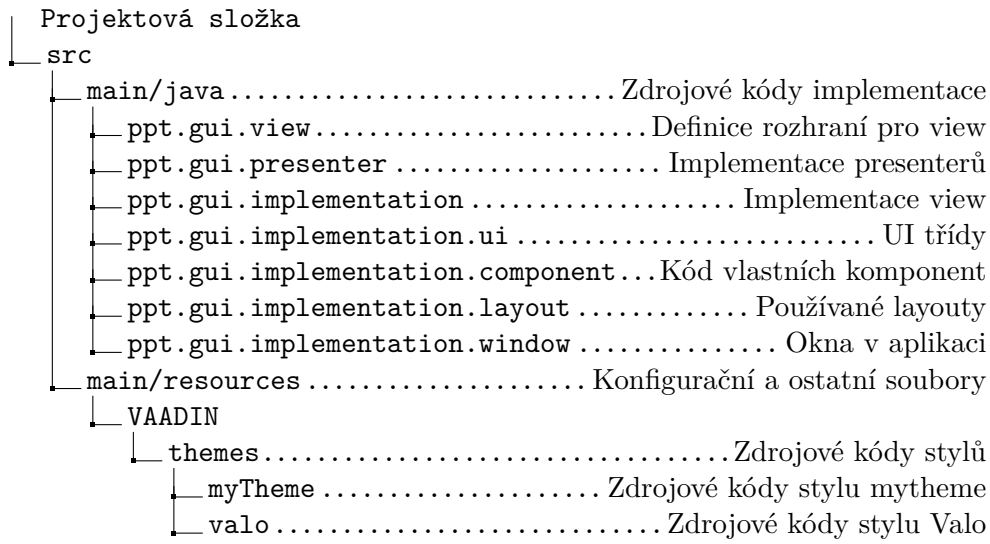
Komponenta BurnDownChart ve svém jádru využívá Vaadin addon DCharts. Obsahuje definici uživatelského rozhraní Burndown reportu tak, jak je použit v aplikaci. Přidává mu metody pro definici zobrazovaných dat a formulářové prvky pro upřesnění sledovaného období. Na obrázku 11.2 je grafická podoba této komponenty.

Regresní přímku pro odhad trendu popsanou v sekci 3.3 lze díky DCharts získat také a to přidáním následujících řádků.

```
SeriesDefaults seriesDefaults = new SeriesDefaults()
    .setTrendline(
        new Trendline().setShow(true)
    );
```

Generování ovšem má své nedostatky. Nelze se dostat k datům, které knihovna zvolí jako vstup pro generování a je tedy nemožné vykreslování přímky do větší míry ovlivňovat. Nelze také přímku graficky upravovat. V budoucnu tedy bude generování trendové křivky prováděno jiným způsobem než za pomoci DCharts a to knihovnami pro jazyk Java zabývající se regresí.

11.7 Struktura balíků prezentační vrstvy



Kapitola 12

Testování

12.1 Unit testy

V aplikaci je implementována sada unit testů pro ověření správné funkčnosti tříd DAO a Mapperů. Pro psaní testů bylo využito knihovny JUnit [45]. Testovací třídy pracují s H2 testovací databází, kterou si samy vždy před spuštěním inicializují a po vykonání testu ji mažou. Na inicializaci databáze se používají skripty ze složky `scr/main/resources/database`. Konfigurace databáze je uložena v souboru `application.properties` umístěném ve složce `src/test/resources/`. Testy lze spustit voláním mavenu:

- příkazem `mvn test`,
- příkazem `mvn -Dtest=CustomerDAOTest test` pro spuštění konkrétní třídy.

12.2 Uživatelské testování

Jakmile vývoj pokročil, tak se přistoupilo k uživatelskému testování následujícím způsobem. Další úkoly potřebné pro vykonání už byly vkládány přímo do již implementované aplikace a tím se ověřilo, zda aplikace splňuje požadavky vyjmenované v sekci 5.1. Používání aplikace bylo vyhodnoceno a byl sepsán seznam možných vylepšení, která by vedla ke zlepšení uživatelského komfortu, a tím tedy zlepšení celé aplikace. Také se tím pomohla stanovit priorita budoucích úkolů, což může vést k dřívějšímu nasazení sice nekompletní, ale ve své podstatě již použitelné aplikace. Při testování se přišlo mimo jiné na následující:

- Nelze přejmenovat strom.
- Nelze přesunout úkol mezi podstromy.

- Názvy jednotek by mohly být u čísel a nejenom v záhlaví.
- Pole pro popis úkolů je moc krátké (50 znaků).
- Pokud si uživatel definuje zobrazení jen některých sloupců, tak by se tato definice mohla uložit a při opětovném navštívení opět zobrazit poslední definované zobrazení.
- Nelze psát desetinná čísla s tečkou, ale jen s čárkou.
- Optimalizovat rozmístění komponent s cílem co nejvíce zvětšit pracovní plochu.
- ...

12.3 Dokumentace

Pro dokumentaci datového modelu, analýzy, architektury je použit nástroj Enterprise Architect. Pro dokumentaci zdrojových kódů je využit nástroj Javadoc. Javadoc je propojen s Mavenem skrze `maven-javadoc-plugin` a dokumentaci tedy lze vygenerovat příkazem `mvn javadoc:javadoc`. Vytvořené dokumentační soubory se uloží do `../pptDoc`. Pro hlídaní kvality kódu byl využit doplněk sonarlint [46], což je odlehčená verze SonarQube použitelná na lokálním prostředí.

Kapitola 13

Současná podoba aplikace

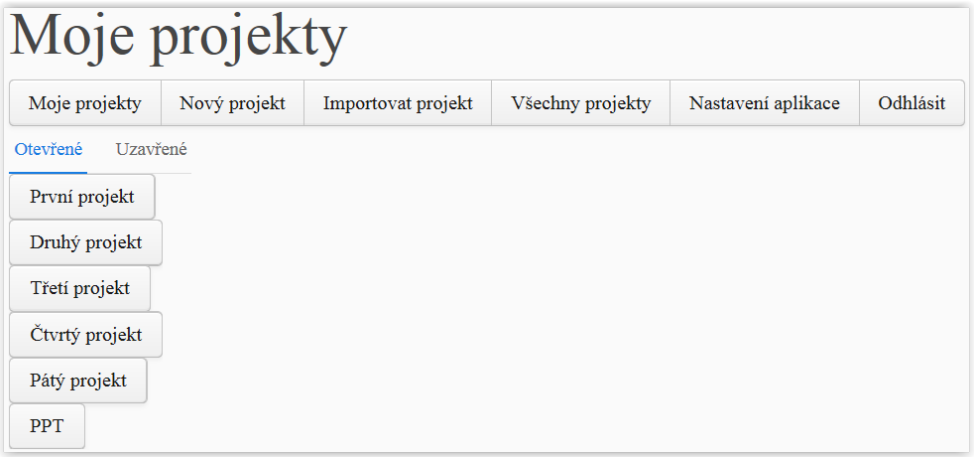
V této kapitole bude předvedena současná podoba aplikace. Je zde grafická forma každého okna nebo podokna aplikace. Z níže uvedeného je zřejmé, že se vzhled jednotlivých oken mírně liší od návrhu v kapitole 7.1. Po dohodě se zadavatelem nebylo do současné verze zahrnuto okno s importem projektu, s detailem úkolu a harmonogram. Naproti tomu bylo navíc o proti návrhu přidáno okno z burndown reportem, filtrování úkolů v projektu, sumarizační zápatí v projektu a okno pro nastavení aplikace.

13.1 Přihlašovací okno

Obrázek 13.1: Přihlašovací okno

13.2 Okno přidělených projektů

Okno se oproti návrhu zjednodušilo. Místo obdélníků s informací o názvu projektu, vedoucím projektu a rozpracovaností jsou zde pouze tlačítka, která po stisknutí vedou na zobrazení detailu projektu. V současné verzi jsou zde vidět všechny projekty v aplikaci a nejenom ty, kterých je přihlášený uživatel součástí. Důvodem pro to je zjednodušení testovaní a fakt, že momentálně všichni přihlášení uživatelé vystupují v roli administrátora a ten má ke všem projektům přístup. Nicméně implementace počítá v budoucnu se změnou bez nutnosti většího zásahu do kódu.



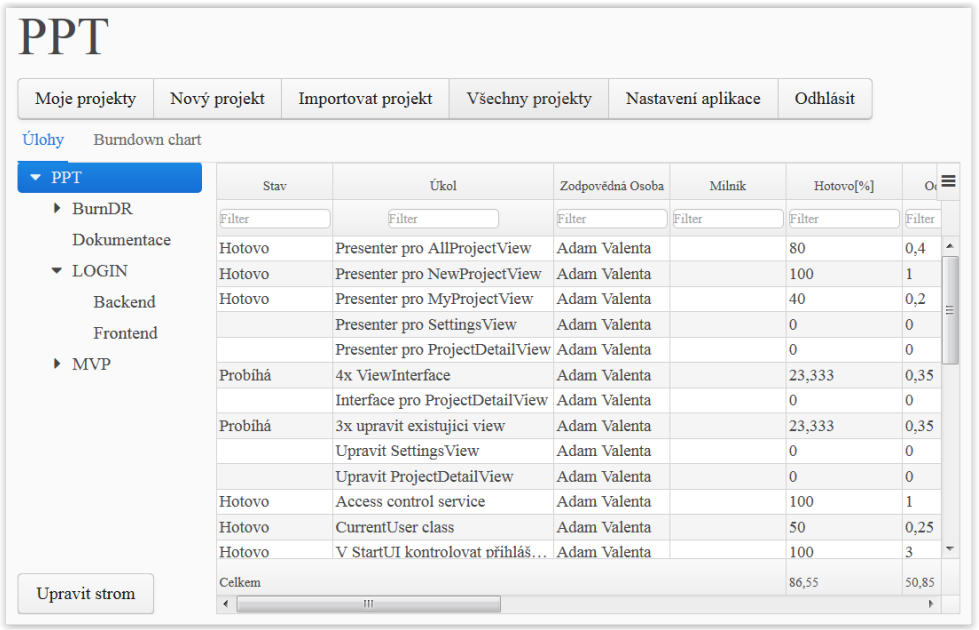
Obrázek 13.2: Okno přidělených projektů

13.3 Detail projektu

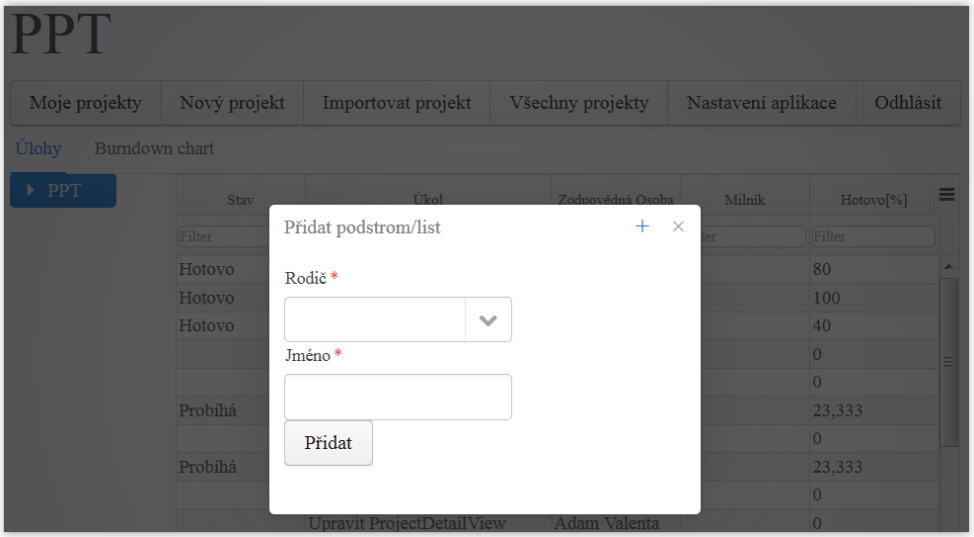
Okno pro zobrazení detailu projektu je to nejpodstatnější okno v aplikaci a při implementaci mu bylo věnováno nejvíce času. Oproti návrhu je do něj přidáno filtrovací záhlaví a zápatí se sumarizací, které se přepočítává pouze při každé změně označení uzlů WBS stromu. Přepočítávat zápatí při filtrování není zatím potřeba s tím, že filtrování s největší pravděpodobností bude sloužit převážně k vyhledání jednoho konkrétního úkolu a pokud bude nějakou skupinu úloh potřeba sledovat společně, tak pro ně bude vytvořen uzel ve WBS. Úlohy je možné dvojklikem editovat.

Dále je možné v okně 13.4 přidat podstrom pouze těm uzlům, které na sebe nemají navázané žádné úlohy. Tato zkutečnost mírně omezuje uživatele při přidávání nových podstromů do již běžícího projektu a po implementaci přesunu úkolů mezi podstromy bude řešena méně omezujícím způsobem.

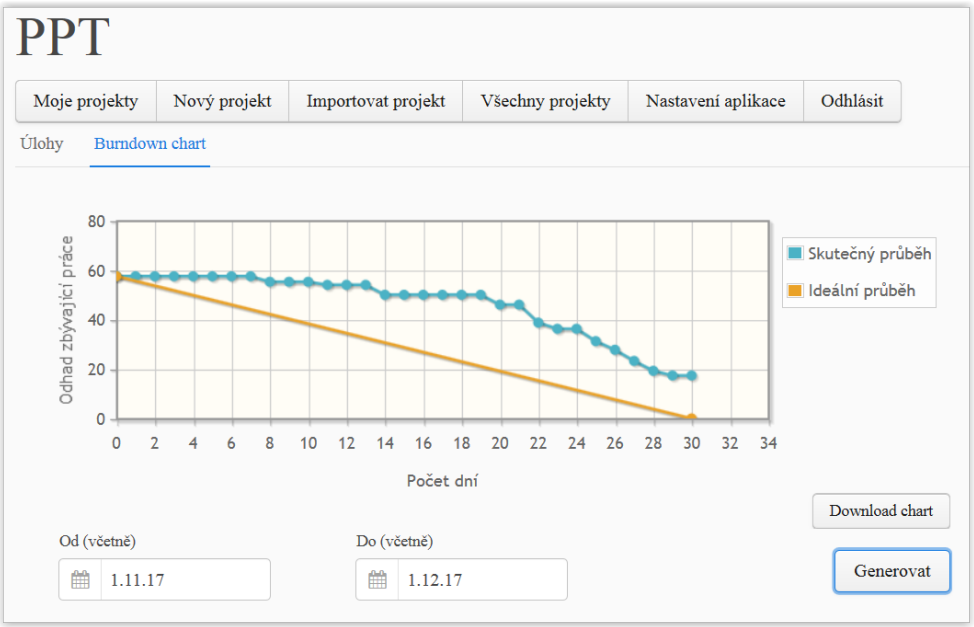
V neposlední řadě je zde i okno pro burndown report 13.5. V něm je možné po zadání mezí znázornit historii projektu. V této verzi implementaci burndown report zobrazuje hodnotu zbývající práce na základě historie odpracovaného času.



Obrázek 13.3: Přehled úkolů



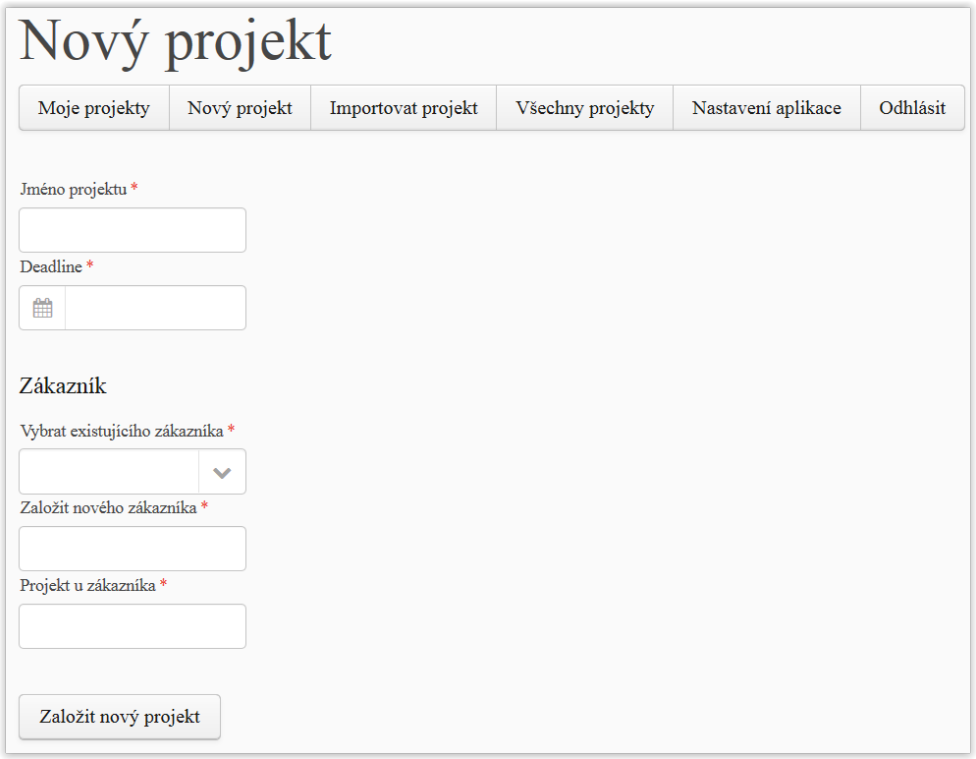
Obrázek 13.4: Úprava WBS



Obrázek 13.5: Burndown report

13.4 Tvorba nového projektu

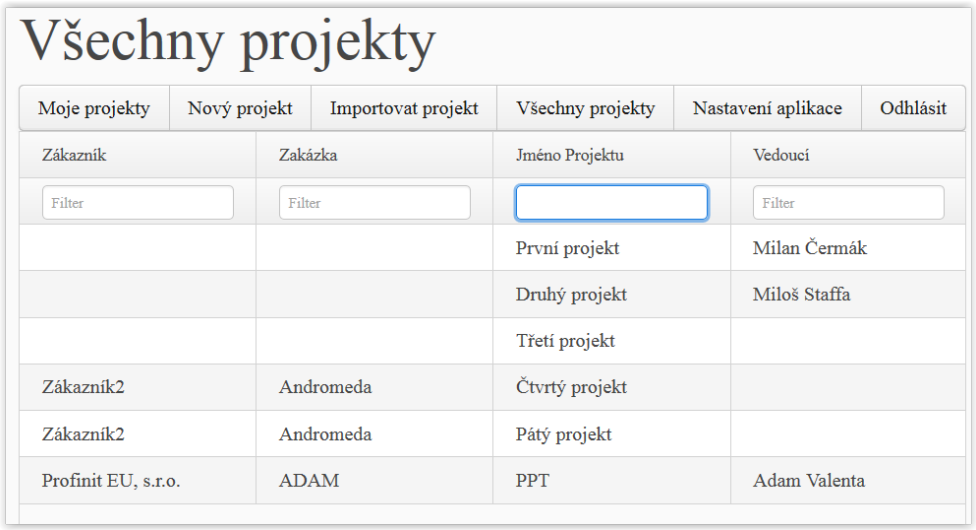
V tomto okně je možné vytvořit projekt po vyplnění příslušných polí. Chybí zde přidání členů týmu a specifikace vedoucího projektu, což bude řešeno v dalších verzích aplikace.



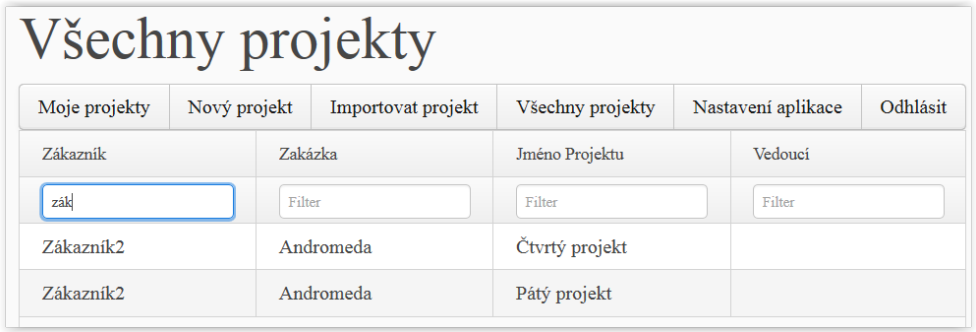
Obrázek 13.6: Tvorba nového projektu

13.5 Přehled všech projektů v aplikaci

Okno se od návrhu neliší vůbec. Dvojklikem na řádek tabulky se lze dostat na okno zobrazující detail projektu. Na obrázku 13.8 je ukázáno jakým způsobem funguje filtrování v tabulce.



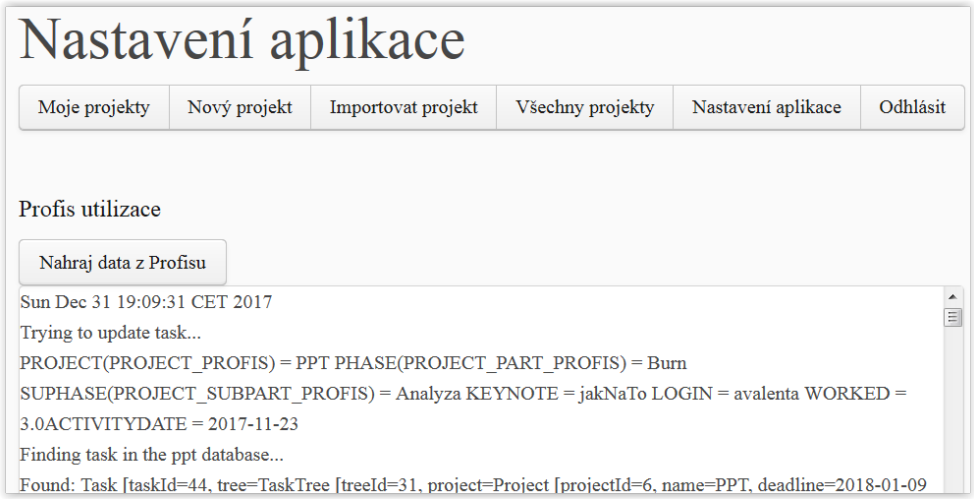
Obrázek 13.7: Přehled všech projektů v aplikaci



Obrázek 13.8: Ukázka filtrování

13.6 Nastavení aplikace

V nastavení aplikace je zatím možné nahrát data ze systému Profis. Respektive tedy z CSV souboru, který je Profis schopný vygenerovat. Po stisku tlačítka se zobrazí přidávací log, ve kterém je možné dohledat případné chyby při importu. Chybou může být situace, kdy úkol v Profisu neodpovídá žádnému úkolu v aplikaci nebo pokud na úkol přispíval člen, který nemá úkol přidělen. V budoucnu zde bude celé nastavení aplikace v němž bude zobrazen i přehled uživatelů a jejich rolí. Rovněž bude také v budoucnu vylepšen plán logu: barevně rozlišit chybové zprávy, poskytnout filtrování logu apod.



Obrázek 13.9: Nastavení aplikace

Závěr

Cílem práce bylo získání aplikace, která by vyhovovala speciálním požadavkům zadavatele a podporovala by jeho stávající metodiku. Tento cíl byl splněn v plném rozsahu. Byla provedena analýza potřebného řešení a na základě toho byla popsána současná metodika, současné používané nástroje a nalezeny interní systémy, se kterými bude aplikace spolupracovat. Dále byly sestaveny požadavky na budoucí aplikaci a bylo určeno v jakých rolích budou uživatelé v aplikaci vystupovat. Povedlo se tedy získat podrobnou představu zadavatele o výsledné aplikaci.

Dále byla provedena diskuze možností řešení, ze které vyplynulo, že nejlepší volbou bude implementace vlastní aplikace v programovacím jazyce Java v interakci s knihovnami Spring Boot, MyBatis, Vaadin 7. Byl realizován návrh řešení, který spočíval v tvorbě jednoduchého prototypu aplikace v programu Justinmind Prototyper a dále v tvorbě datového modelu. V rámci návrhu byl také vytvořen model architektury aplikace. Byla zvolena třívrstvá architektura s MVP vzorem.

Na základě návrhu byla provedena implementace aplikace. Povedlo se vytvořit aplikaci, která splňuje požadavky sepsané v analýze. Aplikace podporuje stávající metodiku zadavatele a je v současné verzi použitelná. Nepovedlo se pouze odstranit všechny nedostatky grafického rozhraní, jako například obarvit buňky tabulky, pokud je hodnota menší než nula, nebo implementovat okno s detailním přehledem všech informací o konkrétním úkolu. V současnosti je aplikace připravená tak, aby jednotlivec nebo tým vývojářů bez větších komplikací pokračoval ve vývoji, opravil všechny nedostatky a dodal další vylepšení, která nebyla v rámci práce řešena. Jedná se například o možnost tvorby harmonogramu projektu, možnost importu projektu ze souboru, detailnější sledování historie úkolů a na základě těchto dat vylepšení burndown reportu.

Literatura

[1] Mapto: *Project triangle*. Wikipedia, [cit. 2017-11-08]. Dostupné z: `https://commons.wikimedia.org/wiki/File:Project-triangle-en.svg`

[2] IPMA: *ICB - IPMA Competence Baseline, Version 3.0*. International Project Management Association, 2004, ISBN 0-9553213-0-1.

[3] Ing. Jan Doležal, a. k., Ph.D.: *Projektový management Komplexně, prakticky a podle světových standardů*. Grada Publishing, a.s., 2016, ISBN 978-80-271-9066-9.

[4] McConnell, S.: *Software Project Survival Guide*. Microsoft Press, 1998, ISBN 1-57231-621-7.

[5] PMI: *What is Project Management?* [cit. 2017-10-28]. Dostupné z: `https://www.pmi.org/about/learn-about-pmi/what-is-project-management`

[6] Pitaš, J.: *Národní standard kompetencí projektového řízení verze 3.2*. Společnost pro projektové řízení, 2012, ISBN 978-80-260-2325-8.

[7] Doležal, J.; Pavel, M.; Lacko, B.: *Projektový management podle IPMA*. Grada, 2012, ISBN 978-80-247-4275-5.

[8] Čermák, M.: *Analýza dosažené hodnoty: nejčastěji používané vzorce*. WebStat, [cit. 2017-11-08]. Dostupné z: `http://www.cleverandsmart.cz/analyza-dosazene-hodnoty-nejcasteji-pouzivane-vzorce/`

[9] Investopedia, L.: *Work In Progress - WIP*. Investopedia, LLC., [cit. 2018-01-01]. Dostupné z: `https://www.investopedia.com/terms/w/workinprogress.asp`

[10] Investopedia, L.: *The Differences Between Job Costing and Process Costing*. Investopedia, LLC., [cit. 2018-01-01]. Dostupné z: `https://www.investopedia.com/terms/w/workinprogress.asp`

[11] Dinwiddie, G.: Feel the Burn: Getting the Most Out of Burn Charts. *Magazine*, 2009.

[12] PabloStraub: *File:SampleBurndownChart.svg*. Wikipedia, [cit. 2017-12-11]. Dostupné z: https://commons.wikimedia.org/wiki/File:SampleBurndownChart.svg

[13] Microsoft: *Plány a ceny*. Microsoft, [cit. 2017-12-11]. Dostupné z: https://products.office.com/cs-cz/project/compare-microsoft-project-management-software

[14] *H2 Database Engine*. [cit. 2017-12-11]. Dostupné z: http://www.h2database.com/html/main.html

[15] Foundation, T. A. S.: *Licensing of Distributions*. The Apache Software Foundation, [cit. 2017-12-11]. Dostupné z: https://www.apache.org/licenses/

[16] Oracle: *MySQL Editions*. Oracle, [cit. 2017-12-11]. Dostupné z: https://www.mysql.com/products/

[17] Group, T. P.: *The PHP License*. The PHP Group, [cit. 2017-12-11]. Dostupné z: https://secure.php.net/license/3_01.txt

[18] Janovský, D.: *H2 Database Engine*. Jak psát web, [cit. 2017-12-11]. Dostupné z: https://www.jakpsatweb.cz/katalog/hosting-php.html

[19] Symfony: *How to Use PHP's built-in Web Server*. Symfony, [cit. 2017-12-11]. Dostupné z: https://symfony.com/doc/current/setup/built_in_web_server.html

[20] Pivotal Software, I.: *Spring Framework*. Pivotal Software, Inc., [cit. 2017-12-11]. Dostupné z: https://projects.spring.io/spring-framework/

[21] Pivotal Software, I.: *Spring Boot*. Pivotal Software, Inc., [cit. 2017-12-11]. Dostupné z: https://projects.spring.io/spring-boot/

[22] Ltd., V.: *Tree*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: https://vaadin.com/docs/framework/components/components-tree.html

[23] Ltd., V.: *Grid*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: https://vaadin.com/docs/framework/components/components-grid.html

[24] JUSTINMIND: *All-in-one Prototyping Tool for web and mobile apps*. 2017. Dostupné z: https://www.justinmind.com/

[25] JUSTINMIND: *Justinmind FREE vs Justinmind PRO edition*. 2017. Dostupné z: https://www.justinmind.com/blog/justinmind-free-vs-pro-edition/

[26] Zelenka, P.: *Metody ukládání stromových dat v relačních databázích*. ZONER software, a.s., [cit. 2017-12-11]. Dostupné z: https://www.interval.cz/clanky/metody-ukladani-stromovych-dat-v-relacnich-databazich/

[27] Ltd., V.: *Advanced Application Architectures*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: https://vaadin.com/docs/v8/framework/advanced/advanced-architecture.html

[28] sockeqwe: *Model View Presenter: The Basics*. GitHub, Inc., [cit. 2017-12-11]. Dostupné z: https://github.com/sockeqwe/Vaadin-MVP-Lite/wiki/Model-View-Presenter:-The-Basics

[29] sockeqwe: *Model View Presenter: Using MVP Lite*. GitHub, Inc., [cit. 2017-12-11]. Dostupné z: https://github.com/sockeqwe/Vaadin-MVP-Lite/wiki/Model-View-Presenter:-Using-MVP-Lite

[30] Initializr, S.; Services, P. W.: *Spring Initializr*. Spring Initializr and Pivotal Web Services, [cit. 2017-12-11]. Dostupné z: https://start.spring.io/

[31] Pivotal Software, I.: *Spring Initializr*. Application property files, [cit. 2017-12-11]. Dostupné z: https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

[32] QOS.ch: *Logback Project*. QOS.ch, [cit. 2017-12-11]. Dostupné z: https://logback.qos.ch/

[33] Foundation, T. E.: *DBeaver*. The Eclipse Foundation, [cit. 2017-12-11]. Dostupné z: https://marketplace.eclipse.org/content/dbeaver

[34] Pivotal Software, I.: *Spring Tool Suite*. Pivotal Software, Inc., [cit. 2017-12-11]. Dostupné z: https://spring.io/tools

[35] MyBatis.org: *Getting started*. MyBatis.org, [cit. 2017-12-11]. Dostupné z: http://www.mybatis.org/mybatis-3/getting-started.html

[36] Rai, A.: *MyBatis 3 Annotation Example*. concretepage.com, [cit. 2017-12-11]. Dostupné z: https://www.concretepage.com/mybatis-3/mybatis-3-annotation-example-with-select-insert-update-and-delete

[37] MyBatis.org: *Acquiring a SqlSession from SqlSessionFactory*. MyBatis.org, [cit. 2017-12-11]. Dostupné z: http://www.mybatis.org/mybatis-3/getting-started.html

[38] MyBatis.org: *Building SqlSessionFactory from XML*. MyBatis.org, [cit. 2017-12-11]. Dostupné z: http://www.mybatis.org/mybatis-3/getting-started.html

[39] MyBatis.org: *typeHandlers*. MyBatis.org, [cit. 2017-12-11]. Dostupné z: http://www.mybatis.org/mybatis-3/configuration.html

[40] sourceforge.net: *A Java database driver for reading comma-separated-value files*. sourceforge.net, [cit. 2017-12-11]. Dostupné z: http://csvjdbc.sourceforge.net/

[41] Ltd., V.: *Vaadin Spring*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: http://vaadin.github.io/spring-tutorial/

[42] Vejnovič, D.: *dCharts Widget*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: http://vaadin.github.io/spring-tutorial/

[43] Ltd., V.: *vaadin/framework*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: https://github.com/vaadin/framework/tree/master/themes/src/main/themes/VAADIN/themes/valo

[44] Ltd., V.: *Book of Vaadin Examples*. Vaadin Ltd., [cit. 2017-12-11]. Dostupné z: https://demo.vaadin.com/book-examples/book/

[45] JUnit: *JUnit*. JUnit, [cit. 2017-12-11]. Dostupné z: http://junit.org/junit4/

[46] SonarSource: *SonarLint*. SonarSource, [cit. 2017-12-11]. Dostupné z: https://www.sonarlint.org/

Příloha A

Seznam použitých zkratek

WBS Work Breakdown Structure

MS Microsoft

EVM Earned value management

MD Man day

MH Man hour

OS Operační systém

EA Enterprise Architect

WIP Work In Progress

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... Stručný popis obsahu CD
- `sablona.xlsm` ..... Šablona původního řešení v MS Excel
- `src`
  - `impl` ..... Zdrojové kódy implementace
  - `thesis` ..... Zdrojová forma práce ve formátu LaTeX
- `text` ..... Text práce
  - `BP_valenta_adam_2017.pdf` ..... Text práce ve formátu PDF

Příloha C

Struktura balíků projektu

- PPT
  - src
    - main/java — Zdrojové kódy implementace
      - ppt — Spouštěcí třída
      - ppt.data.util — Deficice Session
      - ppt.data.entity — Entitní třídy
      - ppt.data.dao — DAO třídy
      - ppt.data.mappers — Mappery
      - ppt.data.typehandlers — TypeHandlery
      - ppt.data.profis — Třídy poskytující přístup k Profisu
      - ppt.business.services — Třídy pro komunikaci s DAO
      - ppt.business.accesscontrol — Kontrola uživatelského přístupu
      - ppt.business.messages — Zprávy poskytované uživateli
      - ppt.gui.view — Definice rozhraní pro view
      - ppt.gui.presenter — Implementace presenterů
      - ppt.gui.implementation — Implementace view
      - ppt.gui.implementation.ui — UI třídy
      - ppt.gui.implementation.component — Kód vlastních komponent
      - ppt.gui.implementation.layout — Používané layouty
      - ppt.gui.implementation.window — Okna v aplikaci
    - main/resources — Konfigurační a ostatní soubory
      - database — DDL a datový skript
      - profis — Složka s csv soubory
      - messages — Jazykové soubory
      - VAADIN
        - themes — Zdrojové kódy stylů
          - myTheme — Zdrojové kódy stylu mytheme
          - valo — Zdrojové kódy stylu Valo
      - logback-spring.xml — Konfigurační soubor logu
      - mybatis-config.xml — Konfigurační soubor MyBatisu
      - application.properties — Konfigurační soubor Springu
    - test/java — Zdrojové kódy unit testů
      - ppt.data.util — Třída pro správu testovací DB
      - ppt.data.dao — Testy balíku DAO
    - test/resources — Konfigurace testu
  - pom.xml — Konfigurační soubor Mavenu
  - README.md — Příručka pro start projektu