VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJOVÁ PODPORA VÝVOJE SOFTWARU METODIKOU SCRUM

SCRUM PROJECT MANAGEMENT TOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ANDREA OSTEROVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK KOČÍ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19717/2016/xoster00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Osterová Andrea**
Obor: Informační technologie
Téma: **Nástrojová podpora vývoje softwaru metodikou Scrum**
Scrum Project Management Tool
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s problematikou agilních metodik vývoje softwaru. Podrobně prostudujte metodiku Scrum.
2. Dokumentujte základní model, praktiky a techniky používané metodikou Scrum.
3. V prostředí reálné firmy analyzujte způsob využívání metodiky Scrum, důvody pro zavedení, přínosy metodiky, slabé stránky.
4. Seznamte se s dostupnými aplikacemi pro podporu vývoje softwaru a vyhodnoťte jejich silné a slabé stránky.
5. Vytvořte nástroj pro podporu vývoje softwaru metodikou Scrum. Nástroj musí umožňovat kooperativní práci více uživatelů, práci se stavem projektu, definovat etapy, úlohy apod.
6. Navrhněte vhodný demonstrační příklad včetně vývojového týmu, na kterém předvedete zásadní vlastnosti aplikace.
7. Shrňte dosažené výsledky, analyzujte použitelnost výsledné aplikace a navrhněte směry další rozvoje tématu práce.

Literatura:

- A. Stellman, J. Greene. Learning Agile. O'Reilly, 2014.
- D. Leffingwell. Scaling Software Agility. Addison-Wesley, 2007.

Pro udělení zápočtu za první semestr je požadováno:

- První 3 body zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kočí Radek, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto bakalárska práca sa zaoberá návrhom a implementáciou webovej aplikácie slúžiacej ako nástrojová podpora pre vývoj softvéru na základe princípov metodiky Scrum. Je cielená najmä na tímy, ktoré s agilnou metodikou Scrum začínajú a snažia sa o jej nasadenie. Práca približuje čitateľovi oblasť agilných metodík a zameriava sa na podrobný popis praktík metodiky Scrum. Analyzuje prostredie reálnych firiem a ich zaužívané procesy pri vývoji produktov. Pri návrhu nástroja bol dôraz kladený na jednoduchosť používania, intuitívnosť a responzivitu. Aplikácia bola implementovaná za použitia frameworku Laravel ako webové rozhranie. Záver práce popisuje proces testovania, ktorý prebehol v reálnej firme na ich projekte.

Abstract

This bachelor thesis deals with the design and implementation of the application that serves as a Scrum-based project management tool. It aims particularly on teams that start with the Scrum agile methodology and try to implement it. This work illustrates the field of agile methodologies and focuses on a detailed description of the Scrum methodology practices. It also analyses the real companies' environment and their well-established processes of product development. Designing the tool, the main emphasis has been put on its intuitive usage and adaptability. The application has been implemented using the Laravel framework as a web interface. In the conclusion of this thesis, there is described the testing process that has taken place in the real company on their project.

Kľúčové slová

agilný, agilné metodiky, SCRUM, softwarový vývoj, webová aplikácia, projektový manažment, projekt

Keywords

agile, agile metodologies, SCRUM, software development, web application, project management, project

Citácia

OSTEROVÁ, Andrea. *Nástrojová podpora vývoje softwaru metodikou Scrum.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Kočí, Ph.D.

Nástrojová podpora vývoje softwaru metodikou Scrum

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Radeka Kočího, Ph.D. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

.....
Andrea Osterová
16. mája 2017

Poďakovanie

Ďakujem vedúcemu mojej bakalárskej práce Ing. Radkovi Kočímu, Ph.D. za konzultácie a vedenie mojej bakalárskej práce. Ďalej by som rada poďakovala vedúcim projektových tímov, ktorí mi poskytli informácie o procesoch spojených s realizovanými projektami. Veľká vďaka patrí aj firme, ktorá bola začlenená do testovania samotného nástroja.

Obsah

1 Úvod **3**

2 Agilné metodiky softvérového vývoja **5**
2.1 Manifest agilných metodík 5
2.2 Agilný prístup oproti tradičnému 6

3 Metodika Scrum **9**
3.1 Role v Scrume 9
3.2 Artefakty v Scrume 10
3.3 Schôdze v Scrume 11
3.4 Model softvérového vývoja 12
3.5 Meracie techniky metodiky Scrum 13

4 Skúmanie metodiky Scrum v praxi **17**
4.1 Analýza využívania metodiky vo firmách 18
4.2 Existujúce riešenia využívané v praxi 20

5 Návrh riešenia nástrojovej podpory **26**
5.1 Analýza požiadaviek 26
5.2 ER diagram 27
5.3 Návrh rozhrania 29

6 Implementácia nástrojovej podpory **32**
6.1 Koncept implementácie aplikácie 32
6.2 Databázová vrstva 34
6.3 Riadiaca vrstva 34
6.4 Funkcionalita a vzhľad výsledného rozhrania 36
6.5 Užívateľské testovanie 39
6.6 Budúci vývoj 41

7 Záver **43**

Literatúra **44**

Prílohy **45**

A Diagram a schéma **46**

B Obsah CD **49**

Zoznam obrázkov

2.1 Vodopádový model 6

3.1 Scrum tabuľa 11
3.2 Model procesu vývoja softvéru podľa agilnej metodiky Scrum 13
3.3 Velocity graf so stabilným výkonom tímu 14
3.4 Velocity graf s osciláciami výkonu tímu 14
3.5 Burndown graf s ideálnym a reálnym priebehom spaľovania User Stories . . 15
3.6 Burndown graf identifikujúci predbiehanie a zaostávanie za plánom 16

4.1 Parametre a ich vzájomný vzťah 17
4.2 Kancelárska tabuľa zobrazujúca aktuálne bežiaci projekt 21
4.3 Aplikácia ScrumDo so zobrazením Scrum tabule 23
4.4 Aplikácia Jira so zobrazením backlogu 23
4.5 Aplikácia Trello so zobrazením tabule Sprintu 24

5.1 Schéma databázy zobrazujúca základné entity s ich reláciami 28
5.2 Ručný návrh realizácie webového rozhrania 30

6.1 Prepojenie komponent MVC v Laravel aplikácii [1] 33
6.2 Nápoveda pre užívateľa s rolou Scrum Master 36
6.3 Modálne okno s podrobnosťami User Story 37
6.4 Scrum tabuľa s úlohami rozdelenými podľa stavu 38
6.5 Burndown graf Sprintu 2 a velocity graf pre vzorový projekt 40

A.1 ER diagram zobrazujúci štruktúru databázy 46
A.2 Use Case diagram pre role Product Owner a Scrum Master 47
A.3 Use Case diagram pre role člen tímu a Scrum Master 48

Kapitola 1

Úvod

„*Tento projekt trvá už príliš dlho, ja už za neho nie som ochotný platiť*“. Túto vetu počulo od svojich klientov už mnoho projektových manažérov. Predstavte si pozíciu vedúceho vývojového tímu, ktorý sa snaží spolu so svojim tímom úspešne dokončiť produkt s veľkým potenciálom. Môže sa zdať, že jeho pozícia je veľmi jednoduchá, stačí to len implementovať a predať klientovi. Skutočnosť je však iná. Klient to potrebuje rýchle, má obmedzený rozpočet a sám dobre nevie, čo všetko bude potrebovať. Úvodná fáza špecifikácií vyčerpá ako klienta, tak aj projektový tím. Nakoniec sa však všetci pustia do vývoja a s plným nasadením postupujú na projekte. Po pár mesiacoch ale prichádzajú zo strany klienta veľké námietky. Netuší aký je stav projektu, nevie na čom sa aktuálne pracuje a či už vôbec niečo v systéme funguje. Na strane tímu situácia nie je o nič pokojnejšia. Klient prišiel s novými požiadavkami, špecifikácia je už ale dávno hotová, pracujú s maximálnym nasadením, ale sú vyčerpaní. Projektový manažér vidí, že ani jeho tím a ani klient nie sú spokojní. Dôvod však nedokáže nájsť. Vie, že vytvárajú produkt, ktorý získa klientovi nemalé peniaze a že tímu zabezpečuje všetko čo potrebuje ku efektívnej práci. Po porade so známym zo spriatelenej spoločnosti však zistil kde je problém. Potrebujú zaviesť nové procesy, organizovať prácu efektívnejšie ako doteraz. Na základe ich rozhovoru a odporúčaní známeho sa rozhodol, že by sa mohli pokúsiť o nasadenie agilnej metodiky Scrum.

Cieľom mojej bakalárskej práce je vytvoriť nástroj, ktorý by projektovým manažérom, stretávajúcim sa s podobným problémom, pomohol pri zavádzaní agilnej metodiky Scrum. Celkový proces zavedenia nepredstavuje len čas strávený nad naštudovaním procesov, ale ide aj o fázu začlenenia tímu do tejto zmeny a o presvedčenie tímu o hodnote, ktorú im táto zmena prinesie. Tento proces nie je jednoduchý a pre členov tímu musí znamenať čo najmenšie možné zaťaženie na samotnú administratívu. Nástroj by mal od užívateľov preto vyžadovať len nevyhnutné množstvo práce. Získané informácie od užívateľa by mali byť využité v takej miere, aby uľahčili všetkým členom projektového tímu ďalšiu prácu spojenú so správou projektu. Vývojový tím často nemá dostatočný čas na vzdelávanie sa v oblasti projektového riadenia a procesov spojených s metodikami vedenia projektov. Ich hlavným zameraním je vyvíjať funkčný produkt a zabezpečovať firme dostatočný obrat. Mojim cieľom je vytvoriť nástroj, ktorý je schopný naviesť tím na správny proces, aj bez predošlej skúsenosti s metodikou Scrum. Komplexnosť je nahradená detailným dodržaním procesov tejto metodiky. Aplikácia by mala poskytovať ucelené informácie ako pre vývojárov, tak aj pre vedúceho tímu. Tieto informácie im zabezpečujú prehľadnosť o stave projektu, či podklady pre jednotlivé porady.

Samotná práca je rozdelená do šiestich častí. Kapitola 2 sa zaoberá princípom agilných metodík. Dochádza tu k porovnaniu tradičných postupov s agilnými. Kapitola 3 je

zameraná na teoretickú rovinu agilnej metodiky Scrum. Popisuje jednotlivé role, artefakty a postupy, ktoré sú pre túto metodiku typické a sú využité aj pri následnom návrhu a implementácií nástroja. Súčasťou je aj poskytnutie prehľadu o základných meracích technikách spolu s rozborom možných odchýlok, ktoré možno identifikovať ako problém pri realizácií projektu. Kapitola 4 je orientovaná na používanie metodiky v praxi v prostredí reálnych firiem a zhrnutie vlastností existujúcich nástrojov používaných aj pre metodiku Scrum. Nasledujúca kapitola 5 poskytuje prehľad o samotnom návrhu aplikácie. Mapuje proces analýzy požiadaviek a z nich odvodených funkcionalít systému so súčasnou definíciou databázy. Posledná kapitola 6 je venovaná oblasti implementácie nástroja, priblíženiu spôsobu fungovania použitého frameworku a výslednej vytvorenej aplikácií. Zahŕňa v sebe aj popis databázovej a riadiacej vrstvy spolu so zaujímavými funkcionalitami. Táto kapitola je doplnená o fázu užívateľského testovania.

Kapitola 2

Agilné metodiky softvérového vývoja

Novou etapou sa vo vývoji softvéru stali agilné metodiky, ktoré postupne nahrádzajú klasický prístup [5]. Agilitu je možné definovať ako schopnosť rýchleho reagovania na zmeny vo svojom okolí. Cieľom je rýchle doručenie kvalitného a flexibilného nástroja, ktorý odpovedá požiadavkám klienta a odráža aktuálne potreby trhu. V dnešnej dobe trhu presýteného firmami v oblasti softvérového vývoja je dôležité, aby bola firma schopná doručiť produkt spoľahlivo a k čo najväčšej spokojnosti klienta.

Agilné metodiky sa snažia o začlenenie klienta do procesu samotného vývoja, kedy nedochádza k zbytočnému predlžovaniu a predražovaniu projektu. Tento prístup je vhodné využiť pre projekty rozsiahleho typu, u ktorých zadávateľ nie je schopný presne špecifikovať všetky požiadavky ešte pred samotným začiatkom implementácie. Proces vývoja sa následne riadi požiadavkami a pripomienkami na zmeny, ktoré vznikajú v priebehu vytvárania produktu. Agilný proces sa nebráni zmenám, tak ako je tomu u tradičných postupov, naopak je týmto zmenám otvorený.

2.1 Manifest agilných metodík

Manifest agilného softvérového vývoja, častejšie známy ako agilný manifest bol vytvorený v roku 2011 [2]. Jeho cieľom je vytvoriť lepší spôsob návrhu a vývoja softvéru a navedenie nových spoločností, či tímov na základné praktiky agilného prístupu. Skladá sa zo **štyroch základných hodnôt**. Napriek tomu, že sú body na pravo hodnotné, tak body naľavo sú cenené vo väčšej miere [11].

„**Jednotlivci a interakcie**	pred procesmi a nástrojmi“
„**Fungujúci software**	pred vyčerpávajúcou dokumentáciou“
„**Spolupráca so zákazníkom**	pred vyjednávaním o zmluve“
„**Reagovanie na zmeny**	pred dodržiavaním plánu“

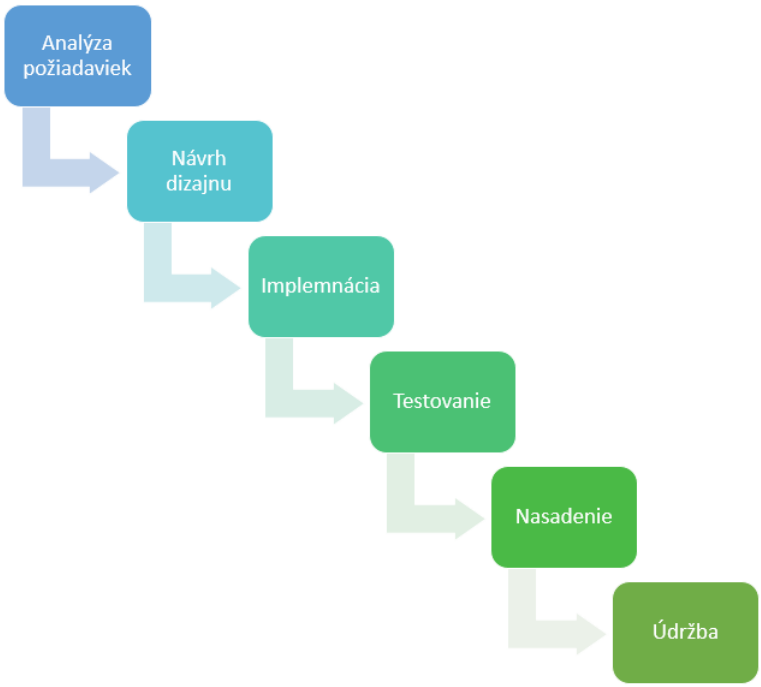
Hodnoty sa zameriavajú na členov tímu, pochopenie ich spôsobu komunikácie a spolupráce, vkladanie plnej dôvery do ich práce. Neoddeliteľnou súčasťou agilného prístupu je snaha o rýchly presun do implementačnej časti projektu, bez potreby zdĺhavej etapy plánovania a definovania požiadaviek. Umožňuje tak doručenie prvých funkčných častí softvéru v priebehu pár týždňov, či mesiacov. Pre samotný vývoj nie je dôležité ustanovovať zmluvy

s kompletnými špecifikáciami a pripravovať obsiahle návrhy. Aj samotný klient uprednostní aktívnu komunikáciu a informovanie o stave projektu pred záväzkami a dohodami, o ktorých plnení nemá prehľad. Agilná metodika je metodikou reaktívnou na zmeny. Nie je dôležité striktné dodržanie plánu, je dôležité uspokojiť klienta a tržný segment, pre ktorý je produkt určený. Je preto nevyhnutné reagovať na zmeny požiadaviek a stav trhu čo najrýchlejšie.

Agilný manifest neposkytuje presný popis procesu vývoja alebo riešenia problémov. Je nositeľom základných myšlienok a princípov, ktoré napomáhajú ku správnej voľbe postupov a riešení situácií spojených s vývojom softvéru. Tieto princípy sa často výrazne odlišujú od myšlienok a pravidiel tradičných metodík.

2.2 Agilný prístup oproti tradičnému

Tradičné prístupy k vývoju softvéru, využívané mnohými firmami aj v dnešnej dobe, sú založené na presne definovaných pravidlách a fázach. Jedným z príkladov tradičného prístupu je **vodopádový model**, ktorý sa typický skladá z fáz vytvárania požiadaviek, návrhu dizajnu a implementácie, testovania, nasadenia produktu a jeho následnej údržby [8].



Obr. 2.1: Vodopádový model

Je to sekvenčný model, v rámci ktorého dochádza k prechodu na nasledujúcu fázu po úplnom ukončení predchádzajúcej. Jeho dôležitou súčasťou je dokumentácia, ktorá uchováva všetky informácie o naplánovanej a navrhnutej implementácií. V priebehu vývoja softvéru je takmer **nemožné zahrnúť výraznú zmenu** do celkového procesu vývoja a preto často dochádza k návratu na úplný začiatok realizácie [9]. Veľkou nevýhodou tohto modelu je **neskorá spätná väzba** od klienta, ktorá neumožňuje rýchlu reakciu na identifikovanú chybu. K predaniu produktu dochádza až po jeho úplnom dokončení. Klient nijakým spôsobom ne-

zasahuje do procesu vývoja softvéru, čím vývojári strácajú možnosť konzultácie správnosti fungovania určitej funkcionality.

Na grafe vodopádového modelu, zobrazeného na obrázku 2.1, je možné vidieť, že k definovaniu požiadaviek a zostrojeniu dizajnu dochádza ešte pred samotnou etapou implementácie. Tento proces je výrazne zdĺhavý a spojený s pomerne obsiahlou dokumentáciou, ktorá vzniká ako výsledok týchto etáp. Táto dokumentácia následne slúži pre ďalšie etapy vývoja. V agilnom svete však dochádza k výraznej zmene vo veľkosti vytváranej dokumentácie i v časovom horizonte vymedzenom na jej prípravu. **Uprednostňuje sa funkčnosť softvéru, pred vyčerpávajúcou dokumentáciou**. Uprednostnená je skôr priama osobná konverzácia, kedy dochádza k zdieľaniu vedomostí, myšlienok a riešení problémov. Požiadavky a samotný návrh je vytváraný postupne na základe požiadaviek klienta, ktoré dokáže podrobnejšie špecifikovať až v dobe behu projektu a existencie určitej časti funkcionalít systému. Následne teda nedochádza k predĺženiu realizácie projektu a teda ani k jeho predraženiu.

Princípy agilných metodík určujú odlišnosti od tradičných prístupov. Tieto princípy boli popísané aj v knihe *Learning Agile* [9], z ktorej vychádza táto kapitola. Základný rozdielom medzi tradičným a agilným prístupom je existencia **iterácie**. Iterácia predstavuje typicky krátku časovo ohraničenú etapu, za ktorú má byť úspešne implementovaná určitá funkcionalita systému, ktorú klient môže následne testovať a využívať. Je dôležité jednotlivé hodnotné funkčné časti **predávať klientovi priebežne**. Cieľom takéhoto predania časti funkcionalít nie je len ubezpečenie klienta o dobrom stave projektu, ide aj o otestovanie funkcionality v reálnom prostredí, kedy je užívateľ systému schopný identifikovať, či implementovaná funkčnosť odpovedá pôvodným požiadavkám a či jej chovanie nevykazuje chybovosť. Súčasne je klientovi umožnené zhodnotiť aktuálnu verziu a pozmeniť svoje požiadavky. Následné návrhy na zmeny a opravy, často spojené so zmenou požiadaviek trţného segmentu a zachovaním konkurenčnej výhody klienta, sú **zahrnuté do nasledujúcej iterácie bez potreby návratu na úplný začiatok** procesu analýzy požiadaviek. Takéto zavádzanie zmien požiadaviek a špecifikácií je možné realizovať počas celej doby vývoja softvéru.

Pracovný tím sa skladá nie len zo samotných vývojárov a manažmentu, ale tvoria ho aj užívatelia, pre ktorých je konkrétny vyvíjaný produkt určený. Naproti tradičnému prístupu je v agilnom svete uprednostňovaná neustála a **aktívna komunikácia s klientom**, či nutnosť pochopenia firemnej kultúry, pracovných procesov a hodnôt, ktoré má vyvíjaný softvér objednávateľovi priniesť. Takéto pochopenie je dôležité pre samotnú vnútornú motiváciu členov tímu. Pri realizácií projektu nie je potrebné zavádzať presné a prísne postupy práce, či rozdelenie rolí ako je tomu u tradičných metodík. Je uprednostnený tzv. samoorganizovaný tím, ktorý je sám schopný rozhodnúť o postupoch a ich riešeniach.

Z pohľadu vedenia tímu je v prípade vodopádového modelu možné identifikovať manažérsku pozíciu, ktorej úlohou je koordinovať prácu tímu, definovať úlohy a náležite ich plnenie kontrolovať. Jednotlivé termíny sú presne dané, rovnako ako nariadenia pre implementáciu. Agilná metodika naproti tomu zavádza tzv. **samoorganizovaný tím**, ktorý je zodpovedný za rozhodnutia a definuje koľko práce je schopný vykonať za určitý pridelený čas. Prevažuje snaha o vytváranie záväzku k prácam, ktoré je tím schopný realizovať. V prípade, že nedochádza k pracovnému preťažovaniu jednotlivých členov, je možné trvalo udržať ich pracovné nasadenie a výkon. Charakteristickou črtou tímu s agilným prístupom je organizovanie častých stretnutí za účelom zhodnotenia určitej iterácie, či naplánovania ďalšieho behu projektu a zhodnotenia jeho aktuálneho stavu. Typicky dochádza k prejednávaniu a návrhu riešení jednotlivých granulárnych častí systému a ich postupnej implementácií.

Granularita definuje náročnosť splnenia danej úlohy v rozsahu menej ako jeden pracovný deň. Takéto úlohy sú následne jednoducho dosiahnuteľné a merateľné, súčasne zabezpečujú dodávateľovi a aj klientovi jasný prehľad o stave projektu.

Agilná metodika odporúča **udržiavať kód neustále funkčný, prehľadný a modifikovateľný**. Testovanie nie je odsúvané až do ukončenia celkovej implementácie, je vykonávané priebežne tak, aby bolo zaručené, že vnesenie novej funkcionality nijak neovplyvnilo funkčnosť existujúceho systému. V prípade vzniku chyby dochádza k jej okamžitému odstráneniu. Agilný tím nie je zameraný na striktné dodržiavanie plánu a riešenie tak nie je odkladané na iné časové obdobie. Ideálnym prístupom je využívanie **automatizovaných testov**, ktoré sú spustené vždy po modifikácií určitej časti projektu, aby bolo zaručené, že do systému neboli zanesené nové chyby. V prípade vodopádového modelu je táto etapa poslednou pred samotným nasadením systému do prevádzky. Identifikovanie akejkoľvek chyby vo funkčnosti systému vedie na úplný začiatok realizácie projektu, čo výrazne predlžuje čas potrebný pre úspešné dokončenie projektu a zvyšuje s tým spojené náklady.

V ktoromkoľvek momente vývoja softvéru je nutné, aby všetci členovia tímu, vrátane manažmentu a klienta mali jasný prehľad o stave projektu. Určenie stavu projektu u agilných metodík je výrazne zjednodušené vďaka využívaniu princípov iterácie. Miera funkčnosti vytvoreného softvéru za realizované iterácie určuje úspešnosť projektu. Beh celkového projektu nie je ovplyvňovaný len jeho stavom, ale aj mierou prínosu vytváranej časti pre klienta. V prípade, že daná časť je pre objednávateľa podstatná a nevyhnutná pre využívanie softvéru, tak musí byť implementovaná čo najrýchlejšie a dostáva tak najvyššiu prioritu. Funkcionality s nižšou prioritou sú odsúvané na neskôr. Takáto prioritizácia napríklad v prípade vodopádového modelu nie je typická, vychádza sa z pôvodných požiadaviek a systém sa vytvára na základe logickej postupnosti. K hodnoteniu projektu a výkonu práce nedochádza až na konci realizácie. **Tím spoločne hodnotí efektivitu ich postupov a riešení**, ale aj celkovú vytvorenú prácu. Spätný pohľad na odvedenú prácu umožňuje tímu napredovať a rozvíjať svoje schopnosti, či naprávať chyby, ktoré vznikli v priebehu iterácií.

Rozdiel medzi klasickým a agilným prístupom je výrazný. Dochádza k zmene vzťahu medzi klientom a dodávateľom, stávajú sa partnermi. Ich partnerstvo je podporované neustálym prehľadom na projektovom postupe a možnosti vidieť funkčnú časť systému za krátky časový okamih. Prechod od klasického vnímania k vnímaniu agilnému však predstavuje zložitý a zdĺhavý proces, do ktorého musí byť začlenený každý článok organizácie.

Kapitola 3

Metodika Scrum

Scrum predstavuje jednu z agilných metodík projektového vývoja. Typickou charakteristikou tejto metodiky je definovanie Sprintov, v rámci ktorých sa samoorganizovaný tím snaží klientovi dodať funkčnú časť aplikácie [9]. Tímu výrazne napomáha osoba zvaná Scrum Master. Základné požiadavky od klienta sprostredkováva Product Owner. Pre úspešnosť používania tejto metodiky a dosiahnutie skutočných myšlienok agilného manifestu musí byť projektový tím otvorený zmenám, ktoré je nutné aplikovať v priebehu vývoja, na základe požiadaviek klienta alebo užívateľov. Jednotlivé role, artefakty a stretnutia budú popísané v nasledujúcich podkapitolách vychádzajúc z knihy zameranej na agilné metódy riadenia projektov [11] a z knihy zameranej špeciálne na metodiku Scrum [7]. Z týchto prvkov metodiky Scrum vychádzal aj samotný návrh aplikácie, jej funkcionalít a spôsobu užívateľského využívania aplikácie.

3.1 Role v Scrume

V tejto podkapitole sú priblížené jednotlivé osoby, ktoré sú súčasťou projektového tímu. Každý z nich zastáva svoju vlastnú pozíciu a sú mu pridelené určité práva a zodpovednosti.

Scrum Master vytvára nárazový bod medzi vývojárskym tímom, vyšším manažmentom a klientom. Nepredstavuje však klasickú manažérsku pozíciu, jeho náplňou práce nie je určovať a rozdeľovať úlohy, ale motivovať tím k tomu, aby potrebné úlohy splnil a podporovať ho pre dosahovanie lepších výsledkov. Je dôležité aby vývojárov chránil pred vonkajšími vplyvmi, ktoré by mohli ovplyvniť ich pozornosť k vykonávaniu úloh. V prípade vzniku problémov, napríklad pri komunikácií, je za ich vyriešenie zodpovedný práve Scrum Master. Napriek tomu, že nie je typickým manažérom je dôležité, aby mal autoritu u členov tímu a bol schopný viesť jednotlivé stretnutia.

Product Owner predstavuje dôležitú osobu, ktorá aktívne komunikuje s realizátormi v plnom zastúpení klienta. Typické je, že pracuje mimo aktívneho pôsobiska vývojárov, ale stretáva sa s nimi na pravidelnej báze. Jeho úlohou nie je len definícia klientových požiadaviek, jeho úloha zahŕňa definovanie vízií, úloh a priorít. Je dôležité, aby Product Owner predstavil víziu vytváraného projektu. To znamená, aby dodávateľovi objasnil prečo je daný produkt potrebné vytvoriť a aký prínos má mať pre jeho užívateľov. Podieľa sa aj na vytváraní Product Backlogu, na základe ktorého pracuje vývojový tím. Súčasne s definíciou požadovaných funkcionalít systému zavádza aj ich prioritu. Tieto priority odlišujú dôležité funkcionality vytváranej aplikácie od funkcionalít, ktorých implementovanie nie je okamžite

potrebné. Napriek tomu, že Product Owner pozná klientove požiadavky typicky najlepšie, mal by akceptovať vývojový tím a ich technologické postupy.

Člen tímu alebo tzv. **Team Member** tvorí veľmi dôležitú súčasť projektového tímu. Úlohou členov tímu nie je len implementácia jednotlivých funkcionalít, ktoré sú klientom požadované, ale aj hodnotenie behu projektu a úloh, ktoré majú byť implementované. Z toho vyplýva, že samotný člen môže ovplyvniť spôsob svojej práce, sám sa rozhoduje a určuje akým spôsobom svoju prácu vykoná. Na rozdiel od klasického tímu u Scrum tímu nie sú striktne oddelené úlohy jednotlivých členov. Členovia tímu sa vzájomne dopĺňajú ich znalosťami a schopnosťami. Je dôležité, aby bol tím schopný rozhodnúť o tom, ktorý člen bude zodpovedný za užívateľský príbeh. Takéto zdieľanie vedomostí následne vedie k vytvoreniu tímu, ktorý je v prípade odchodu kolegu naďalej schopný plnohodnotnej práce.

Zákazník sa v rámci agilného procesu môže stať súčasťou realizačného tímu. Výrazne sa tak zlepšuje porozumenie potrieb samotného klienta a predchádza sa možnej zlej implementácií určitej funkcionality. Veľmi dôležitou je komunikácia medzi zákazníkom a dodávateľom. Je s ním možné zdieľať aktuálny stav projektu, prípadné pripomienky a návrhy na zmeny z technického pohľadu vývojárov, ktoré môžu vylepšiť pôvodne zamýšľané funkcionality. Rovnako zo strany zákazníka je v prípade potreby doplnenia novej funkcionality pomerne jednoduché danú informáciu predať vývojárom.

3.2 Artefakty v Scrume

Metodika Scrum manipuluje s niekoľkými základnými artefaktami, ktoré využíva v celkovom procese riadenia pre rozdelenie projektu do menších častí, ktoré sú merateľné a organizovateľné.

User Story popisuje funkcionalitu systému, ktorú je potrebné implementovať alebo zmeniť. Určuje úkon, ktorý užívateľ vykoná v systéme spolu s prínosom pre jeho pracovnú činnosť. Je teda interpretovaná z pohľadu užívateľa. V priebehu prechodu od vízie k samotnej realizácií dochádza k deleniu User Story na menšie časti, ktoré sú lepšie a presnejšie popísateľné.

Product Backlog zlučuje naplánované User Stories do jedného celku. Poverenou osobou za udržanie aktuálnych informácií v Product Backlogu je Product Owner. Pri jeho vytváraní dochádza k rozdeleniu User Stories na základe ich priority, kedy vyššia priorita indikuje nutnosť skorej implementácie konkrétnej funkcionality. Súčasťou User Stories v Product Backlogu je aj odhad ich náročnosti, ktorý je zostavený na základe konzultácie s vývojovým tímom.

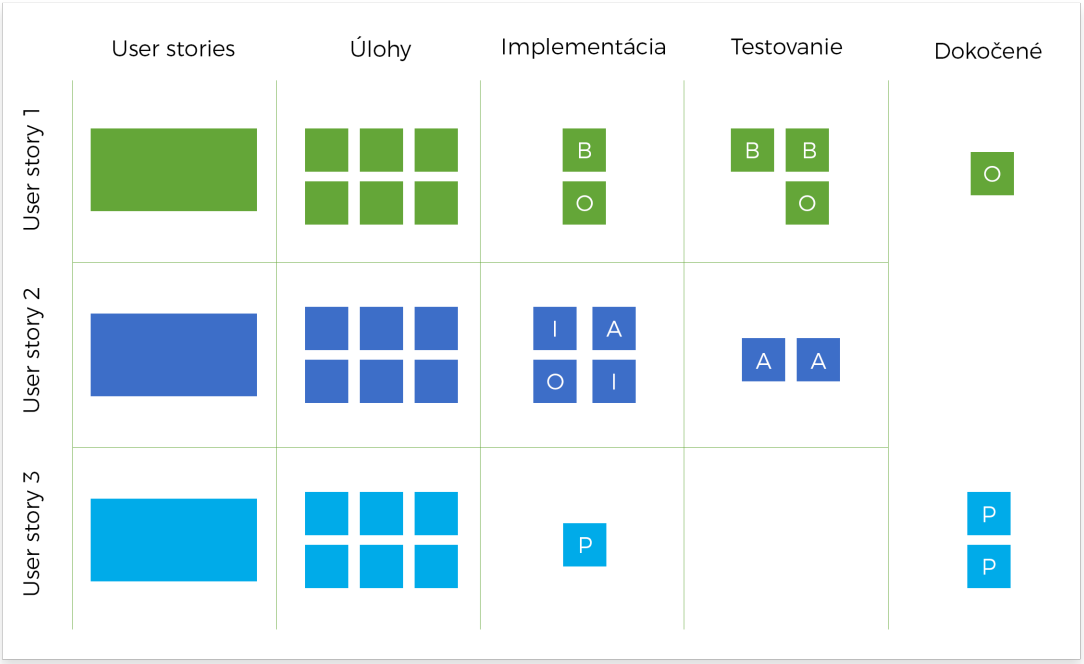
Sprint Backlog je zostavovaný pre každý Sprint pred jeho zahájením. Vychádza z Product Backlogu, z ktorého sú vyberané User Stories s najvyššou prioritou tak, aby nepresiahli maximálne schopnosti vývojového tímu a aby ich tím v priebehu Sprintu zvládol plne implementovať. Vzniknutý Sprint Backlog je počas celého behu Sprintu nemenný.

Sprint predstavuje opakujúce sa iterácie v rámci implementácie produktu. Je definovaný na fixný časový úsek, nastavený na dobu typicky troch až piatich týždňov, za ktorý je potrebné dokončiť všetky User Stories v naplánovanom Sprint Backlogu. Pre preverovanie plnenia plánu je každý deň zvolávaná porada – tzv. Daily Meeting. Na konci Sprintu dochádza k jeho zhodnoteniu pomocou tzv. Sprint Review a k predaniu hotovej funkcionality klientovi.

Scrum tabuľa je využívaná ako vizualizačný prostriedok, ktorý zabezpečuje prehľadnosť o funkcionalitách v rôznych fázach vývoja. Odlišuje úlohy, ktoré je potrebné implementovať od tých , ktoré sú aktuálne rozpracované alebo už boli úspešne implementované. Je

využívaná ako fyzická kancelárska tabuľa alebo ako súčasť softvérového nstroja pre správu projektov.

Typické prevedenie Scrum tabule je možné vidieť na obrázku 3.1. Je rozdelená na 5 častí. Prvá sekcia je určená pre zobrazenie User Story s jej krátkym popisom. Ostatné časti určujú fázu spracovania konkrétnej User Story a sú v nej umiestnené s ňou súvisiace úlohy. Odporúča sa využívanie kartičiek pre popis konkrétnej úlohy, ktoré sa postupne presúvajú po tabuli. Pri spracovaní jednotlivých funkcionalít dochádza k vzniku menších úloh, ktoré sú potrebné pre implementovanie danej funkcionality. Členovia tímu sa zaväzujú k vykonaniu danej úlohy tým, že si príslušnú kartičku odoberú zo sekcie úloh do sekcie určujúcej, že je úloha aktuálne spracovaná. Je vhodné, aby boli jednotlivé kartičky jednoznačne označené symbolom, ktorý určuje osobu zodpovednú za spracovanie príslušnej úlohy. Po jej dokončení musí prebehnúť otestovanie naprogramovanej sekcie a schválenie akceptačných kritérií. V prípade potvrdenia kritérií je možné považovať danú úlohu za splnenú. V prípade implementácie všetkých úloh určujúcich User Story je aj táto User Story považovaná za plne implemenetovanú.



Obr. 3.1: Scrum tabuľa

3.3 Schôdze v Scrume

Podkapitola sa zaoberá schôdzami realizovanými v procese vývoja produktu. Sú organizované v rôznych fázach vývoja a sú zamerané napríklad na definovanie cieľov projektu, zistenie stavu projektu, či spätné zhodnotenie realizovaného projektu. Z dôvodu zachovania terminológie je pojem schôdza nahradený pojmom meeting so súčasným zachovaním anglického pomenovania.

Zahajovací Meeting alebo aj **Pre-planning Meeting** je prvým krokom pri realizácií projektu. Je dôležitý z dôvodu potreby vytvorenia samotného vývojárskeho tímu a ich zoznámenia s celkovým projektom. Po zložení tímu, pribratí špecialistov a začlenení Product Ownera, je potrebné objasniť tímu vízie celkového projektu. Predstavenie vizionárskeho cieľa je plne v rukách Product Ownera, ktorý interpretuje požiadavky klienta. Výsledkom tohto stretnutia je predbežný Product Backlog, ktorý je zostavený za pomoci celého tímu. Členovia tímu sú pri vytváraní Backlogu dôležitou súčasťou diskusie, nakoľko môžu prinášať zásadné technické poznámky.

Backlog Refinement Meeting prebieha za účelom vytvorenia finálnej verzie Product Backlogu. Typicky sa prevádza pred samotným spustením produktu, ale je možné sa k nemu navracať aj počas jeho realizácie a daný Backlog aktualizovať a dopĺňať o nové úlohy. V tejto fáze dochádza k prevedeniu odhadu náročnosti užívateľských príbehov spolu s definovaním priorít a akceptačných kritérií.

Sprint Planning Meeting má za úlohu výber užívateľských príbehov z Product Backlogu. Tieto užívateľské príbehy sú vybrané na základe priority a sú zostavené tak, aby spolu určitým spôsobom súviseli s cieľom, ktorý má byť na konci Sprintu dosiahnutý.

Daily Meeting je zvolávaný každodenne a mali by sa na ňom zúčastniť všetci členovia vývojárskeho tímu a Scrum Master. Tohto meetingu sa môže zúčastniť aj Product Owner, ale len ako nestranný pozorovateľ. Jeho cieľom je oboznámiť všetkých členov tímu o aktuálnom stave práce so súčasným zaviazaním sa k práci na daný pracovný deň. V prípade, že zamestnanec nesplnil konkrétnu úlohu, ku ktorej sa zaviazal, je dobré aby predniesol kolegom z akého dôvodu k tomu došlo. Pri Daily Meetingu sa odporúča využívať Scrum tabuľu, ktorá uľahčuje prehľadnosť o stave projektu a o participácií členov na jednotlivých úlohách.

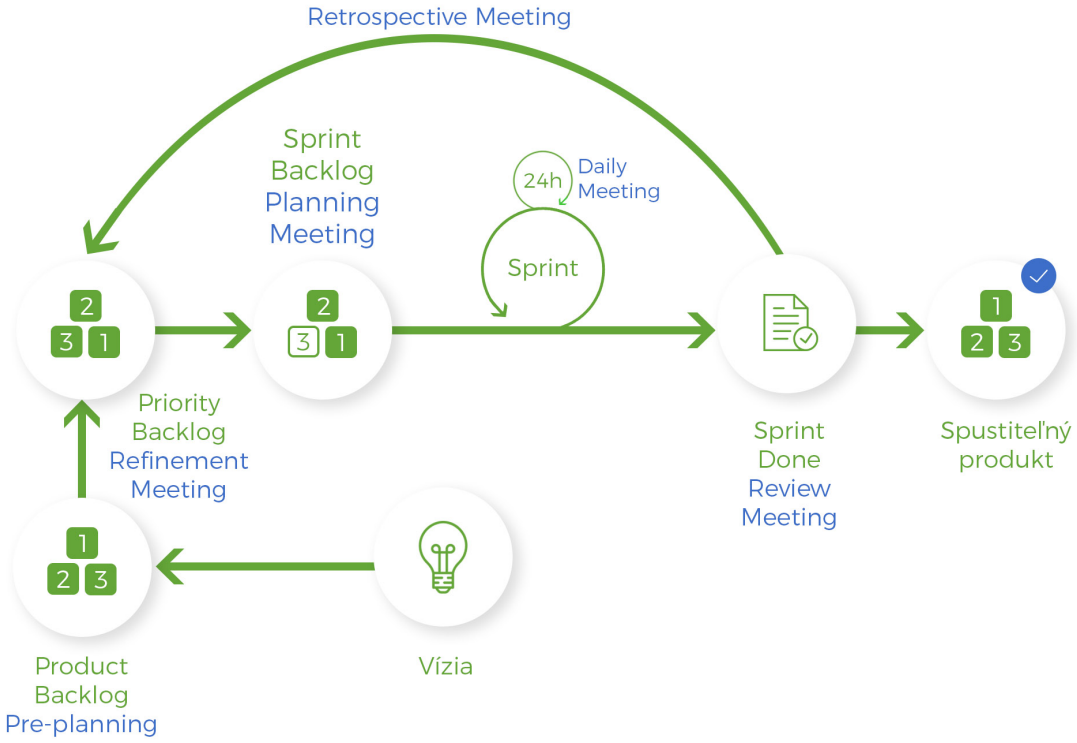
Review Meeting tvorí dôležitý prvok analýzy úspešnosti Sprintu. Dochádza pri ňom k predávaniu hotovej funkcionality vytvorenej v priebehu posledného cyklu. Zúčastňujú sa na ňom všetci členovia tímu, Product Owner, Scrum Master a samotný klient, ktorému je funkcionalita predstavovaná.

Retrospective Meeting je zvolávaný za účelom zhodnotenia priebehu Sprintu. Pri hodnotení je nutné brať do úvahy výkonnosť tímu ako celku, ale aj výkonnosť jednotlivcov. Dochádza k identifikovaniu chýb, ktoré sa v priebehu Sprintu vyskytli. Záverom tohto stretnutia by malo byť ustanovenie, čomu je potrebné sa pri budúcom postupe vyhnúť, čo je potrebné zmeniť a čo nové zaviesť tak, aby bol budúci Sprint úspešnejší ako ten predchádzajúci.

Predávajúci Meeting predstavuje vyvrcholenie celého procesu vývoja dodávanej aplikácie, či softvéru. Je etapou oficiálneho predávania finálnej verzie klientovi spolu s dokumentáciou a akceptačnými protokolmi. Týmto stretnutím sa ukončuje realizácia projektu.

3.4 Model softvérového vývoja

Priebeh vývoja softvéru v metodike Scrum je znázornený pomocou čiastočných krokov od vízie až ku produktu na obrázku 3.2. Tento model vychádza zo základneho empirického modelu pre procesnú kontrolu vývoja, súčasne ho dopĺňa o dvojitý cyklus kontroly [5]. Skladá sa z artefaktov a je sprevádzaný stretnutiami popísanými v predchádzajúcich podkapitolách. Pri tomto procese dochádza k pretvoreniu vízie do kompletného spustiteľného produktu. Z diagramu je zreteľné, že Scrum pracuje na princípe iterácie a neustáleho opakovania krokov plánovania a stretnutí, ktorých cieľom je vytvorenie Product Backlogu a Sprint



Obr. 3.2: Model procesu vývoja softvéru podľa agilnej metodiky Scrum

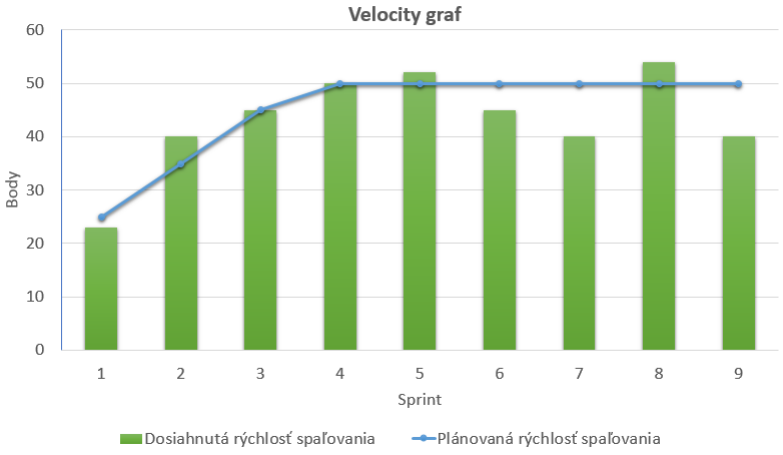
Backlogu. Po ich vytvorení je možné plánovať a spúšťať jednotlivé Sprinty. Každý z nich vrcholí odovzdaním spustiteľnej časti a otestovaním funkčnosti zo strany užívateľa.

3.5 Meracie techniky metodiky Scrum

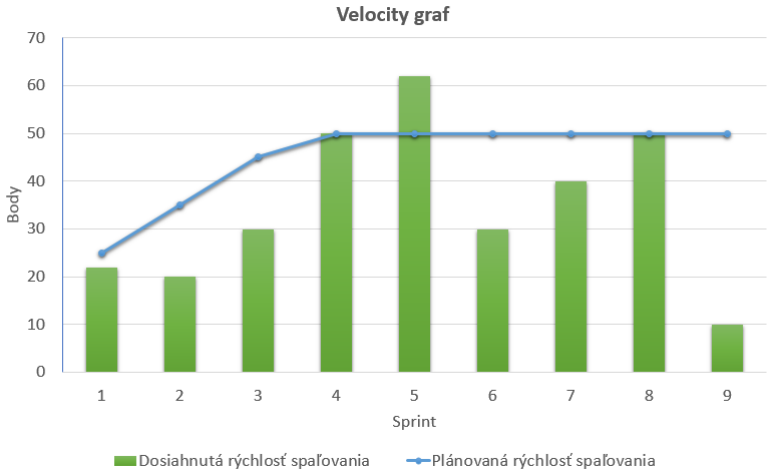
Úspešnosť vývoja projektu je možné merať po každom ukončení Sprintu za pomoci zhodnotenia naplánovaných a skutočne implementovaných funkcionalít vyvíjaného softvéru. Takýto náhľad na stav projektu je ale nárazový a neumožňuje v priebehu Sprintu objektívne zhodnotiť v akom stave sa projekt nachádza. Rovnako nezabezpečuje spôsob určenia dlhodobej efektívnosti fungovania vývojového tímu. Pre takéto účely je možné využiť tzv. velocity alebo burndown graf.

Velocity graf slúži pre určenie rýchlosti pracovania tímu v jednotlivých Sprintoch. Sleduje plánovanú rýchlosť tímu a porovnáva ju s aktuálnymi hodnotami, ktoré bol tím schopný dosiahnuť [10]. Poskytuje prehľad o stálosti výkonu tímu alebo o prípadných výkyvoch, ktoré je následne nutné analyzovať a prispôsobiť im Sprint Backlog pre ďalší beh.

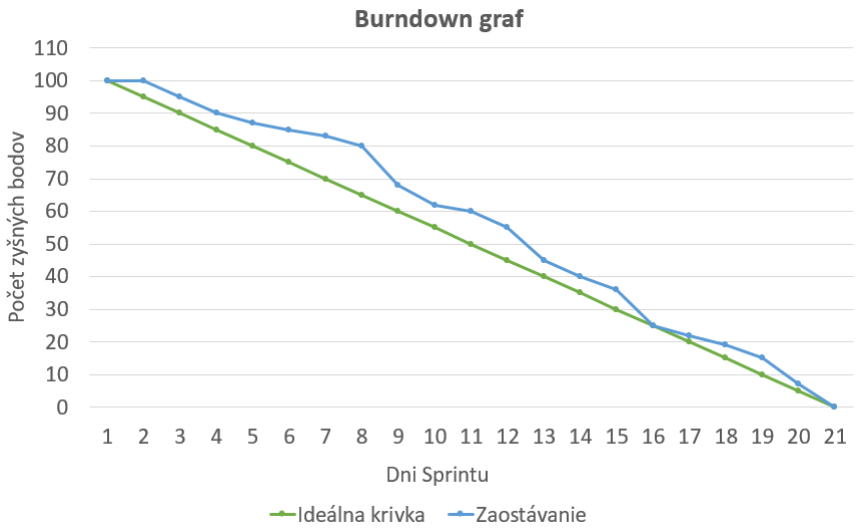
Na obrázku 3.3 je možné vidieť výkonnosť tímu, ktorý je stabilný a v každom Sprinte je schopný dosiahnuť rovnaké bodové hodnotenie. Nedochádza teda ku žiadnym komplikáciám v rámci tímu a Sprint Backlog je vytváraný vhodne na základe možností a schopností vývojárov. Pri začiatku vývoja Sprintu je nutné brať do úvahy, že sa tím potrebuje zaučiť a zoznámiť so samotným projektom a preto je ich výkon slabší.



Obr. 3.3: Velocity graf so stabilným výkonom tímu



Obr. 3.4: Velocity graf s osciláciami výkonu tímu



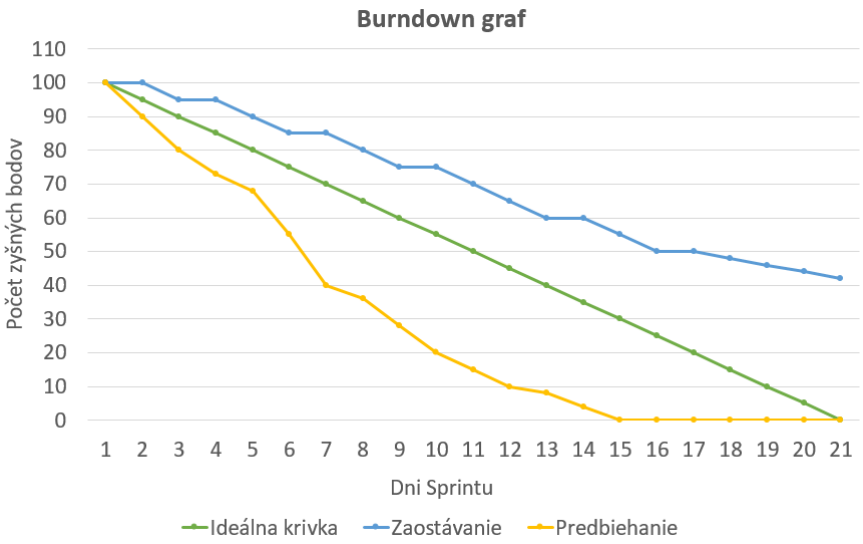
Obr. 3.5: Burndown graf s ideálnym a reálnym priebehom spaľovania User Stories

Obrázok 3.4 identifikuje oscilácie vo výkone počas jednotlivých Sprintov. Tieto oscilácie môžu byť spôsobené rôznymi udalosťami. Napríklad to môže byť problém s určitým členom tímu alebo problém, ktorý súvisel s určitou User Story, ktorá výrazne ovplyvnila prácu tímu a zdržala ho pri vykonávaní ďalších úloh. Opačným prípadom môže byť vysoká výkonnosť prác, ktorá však môže znamenať zníženú kvalitu. Po každom Sprinte je potrebné identifikovať príčiny, ktoré spôsobili takúto osciláciu a snažiť sa ich minimalizovať na takú mieru, aby bolo možné projekt dokončiť v stanovenom čase a v čo najlepšej kvalite.

Burndown graf sa na efektivitu práce tímu pozerá z iného pohľadu. Zobrazuje predikciu počtu úloh, ktoré ostávajú do ukončenia Sprintu v závislosti na fáze v akej sa aktuálne projekt nachádza [7]. Určuje ako rýchle tím „spaľuje“ jednotlivé User Stories a teda, či sa pohybuje plánovanou rýchlosťou, prípadne, či stagnuje. Burndown graf je viazaný ku konkrétnemu Sprintu a nenazerá na úspešnosť dokončenia celkového projektu, ale aktuálne bežiaceho Sprintu.

Porovnanie ideálneho priebehu s priebehom reálnym zobrazuje obrázok 3.5. Priebeh zobrazený zelenou farbou predstavuje ideálny stav spaľovania jednotlivých User Stories. Tento stav je však v skutočnosti takmer nedosiahnuteľný, nakoľko do procesu vývoja vstupujú rôzne rušivé faktory, ktoré ovplyvňujú výkon tímu. Tieto rušivé faktory je možné vidieť v reálnej krivke, ktorá sa len mierne odchyľuje od ideálnej priamky v burndown grafe. Odráža faktory, ktoré môžu ovplyvniť fungovanie tímu, jeho výkonnosť a následné dosiahnutie denného cieľa.

Pri samotnom procese vývoja však dochádza k častým výkyvom vo výkonnosti. Tieto zmeny oproti plánu je možné vidieť na obrázku 3.6. Správny Scrum Master by mal vedieť tieto problémy identifikovať a zaviesť prípadné opatrenia. Aj napriek tomu, že sa priebeh Sprintu uberá spočiatku správnym tempom, tak je možné že počas neho nastanú výrazné výkyvy. V grafe je zobrazená priamka identifikujúca zaostávanie tímu za plánom, tempo je teda výrazne nižšie ako by sa očakávalo. Hlavným problémom, ktorý môže spôsobiť takéto zaostávanie je zlé naplánovanie Sprintu a precenenie schopností tímu. Druhým problémom môže byť nízka efektivita práce jednotlivých členov tímu. Rovnako ako zaostávanie, tak



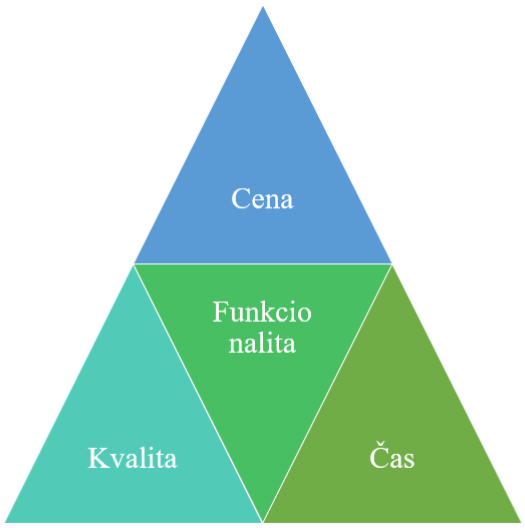
Obr. 3.6: Burndown graf identifikujúci predbiehanie a zaostávanie za plánom

ani predbiehanie zobrazené druhou priamkou, nie je ideálnym priebehom Sprintu. Napriek tomu, že značí rýchle tempo práce a tým pádom rýchle dokončenie Sprintu, môže značiť nízku kvalitu práce alebo v lepšom prípade len podcenenie schopností tímu. Tento problém môže byť odstránený opätovným prehodnotením zostavovania Sprint Backlogu a lepšou kontrolou správnej funkčnosti implementovaných User Stories.

Kapitola 4

Skúmanie metodiky Scrum v praxi

V šesťdesiatich rokoch minulého storočia bola identifikovaná výrazná odlišnosť charakteru továrenskej výroby od vývoja softvéru. Napriek tomu, že sa jedná o výrobu produktu, sú na tento proces kladené iné systematické a koordinačné požiadavky ako je tomu napríklad pri výrobe spotrebného tovaru. Zlé plánovanie a požiadavky na kvalitu softvéru vyústili v minulom storočí do tzv. **softvérovej krízy**, o ktorej boli vedené diskusie už v roku 1968 na konferencií NATO [6]. Produkty vyvíjané v tomto období niesli charakter pomerne **nízkej kvalitatívnej úrovne** z hľadiska čitateľnosti zdrojových súborov, či možnosti rýchlych opráv a modifikácií. Realizácia projektov často výrazne **prekračovala pôvodne stanovené termíny a náklady** na vývoj softvéru, napriek tomu, že výsledný produkt často úplne neodpovedal špecifikáciám a požiadavkám klienta.



Obr. 4.1: Parametre a ich vzájomný vzťah

Zvýšením akosti vyvíjaných softvérov a produktivity softvérových inžinierov s dôrazom na finančné hľadisko sa zaoberá softvérové inžinierstvo. Práve počiatky vzniku tohto vedeckého smeru boli spojené so softvérovou krízou. Dochádza k snahe o vytvorenie základných postupov pre efektívne riadenie životného cyklu produktu. Napriek existencií softvérových

metodík problémy s vývojom kvalitného, málo chybového produktu, ktorý je dodaný na čas a nedôjde k jeho niekoľkonásobnému predraženiu, stále v praxi pretrvávajú.

Dnešným trendom sa stávajú agilné metodiky. Vychádzajúc z ich samotného charakteru a definície sa v praxi firmy zameriavajú na **3 fixné parametre (cena, čas a kvalita) a premenný parameter (funkcionalita)**. Úlohou projektového manažéra je kontrolovať stav fixných parametrov a na základe dohody s klientom určiť funkcionalitu, ktorá je potrebná pre odovzdanie produktu v stave, kedy je klientovi plne umožnené využívať vytvorený softvér. Výsledný vzťah na obrázku 4.1 určuje, že funkcionalita je obmedzená investovanou finančnou čiastkou, dostupným časom a potrebou dosiahnutia určitej kvalitatívnej úrovne.

4.1 Analýza využívania metodiky vo firmách

Nasledujúca kapitola sa zaoberá porovnaním spôsobu využívania metodiky Scrum a jej prvkov vo firmách vytvárajúcich softvérové produkty. Účelom bolo preskúmať nie len druhy využívaných postupov, ale aj schopnosť adaptácie členov tímu na celkový životný cyklus produktu. Skúmanie bolo realizované v dvoch firmách s rozdielnym zameraním, ale aj veľkosťou tímov, či dobou dodávania produktu.

4.1.1 Korporát

Prvou skúmanou firmou bola spoločnosť zameraná na prekladateľské služby a vývoj prekladateľských systémov. Konzultácia prebiehala s vývojovým manažérom, ktorý ma na starosti projektový tím, riadenie implementácie interných, ale aj externých projektov. Dôvodom nasadenia agilných metodík u tejto firmy bola potreba **zníženia časovej náročnosti** na realizáciu projektu. Nakoľko vyvíjaný produkt je definovaný potrebami zákazníka, bolo nutné, aby boli jednotlivé funkcionality otestované a schválené zo strany klienta v čo najkratšej dobe. Takýto inkrementálny prístup k realizácií projektu zabezpečuje postupné zdokonaľovanie funkcionalít produktu, na základe reálnych skúsenosti s využívaním časti fungujúceho softvéru v praxi. Súčasťou je **začlenenie samotného klienta** do celkového procesu tak, aby mali obe strany – teda zadávateľ aj realizátor, rýchlu spätnú väzbu.

Hlavnou orientáciou v rámci softvérového inžinierstva je zameranie sa na princípy agilného manifestu a využívania jeho metodík. Osvojili si najmä metodiku **Kanban** a **Scrum**, ale využívajú aj iné prvky agilných metodík na základe aktuálnych potrieb. Kanban vychádza zo samotného Scrumu a výrazne sa odlišuje absenciou štruktúrovanosti, nedefinuje pravidlá a ani role. Stanovuje hornú hranicu pracovnej výkonnosti, čím zamedzuje zahlteniu pracovných síl členov tímu. Čistý princíp Kanbanu je často doplnený stretnutiami definovanými metodikou Scrum ako je napríklad retrospektíva, plánovací meeting alebo daily standup meeting. Takého kombinovanie prvkov jednotlivých metodík je typické aj pre túto spoločnosť.

V prípade implementácie produktu, ktorý ešte nie je nasadený do produkčného prostredia je volená čistá metodika Scrum. Táto metodika im zabezpečuje možnosť dobrého a efektívneho plánovania krátkych iterácií na základe časových požiadaviek klienta. S klientom je možné po jednotlivých cykloch produkt konzultovať a riešiť prípadné problémy, meniť požiadavky alebo ich upresňovať. Mnou konzultovaná spoločnosť však z väčšej časti pracuje s produktami, ktoré už sú nasadené v produkčnom prostredí a je nutné na jednotlivé zmeny reagovať takmer okamžite. Nie sú preto schopní využívať princíp dlhších iterácií odporúčaných metodikou Scrum. Pre potreby okamžitého zanesenia zmien do produktu

preto využívajú princípy metodiky Kanban, ktoré im umožňujú **dennú reprioritizáciu backlogu**, v niektorých prípadoch dochádza k zmene priority User Stories niekoľkokrát za deň.

Podnet na využívanie agilných metodík a konkrétne aj metodiky Scrum prišiel z vnútra samotného tímu a tím si ich používanie výslovne vyžiadal. Projektový manažér hodnotil túto situáciu ako veľmi pozitívnu, pretože z jeho skúseností vedel, že v prípade nátlaku na projektový tím môže dôjsť k vzniku odporu u jednotlivých členov a odmietnutiu používania jednotlivých princípov.

Pri celkovom životnom cykle vývoja produktu, či jeho úprave sa však stále zachovávajú charakteristiky typické pre Scrum ako sú role, User Stories alebo stretnutia, ako bolo spomenuté vyššie. Z celkového hľadiska sa projektový tím výrazne prispôsobil agilným princípom. Jedným z problematických prvkov sa stali tzv. grooming meetingy, ktoré sa niektorým tímom zdali **zdĺhavé**. Následne teda došlo k ich miernej úprave, kedy boli definované jednotlivé User Stories, ktoré mali byť na stretnutí prejednané a vývojárom boli poskytnuté deň vopred, aby sa s nimi oboznámili a pripravili si prípadné otázky a nápady. Výsledkom bolo výrazné zníženie časovej náročnosti na dané stretnutie. V prípade, že sa ale jednalo o komplexnejší projekt, tak sa rozdelili implementované časti na menšie funkčné celky a boli postupne prejednávané na separátnych grooming stretnutiach.

Využívaným nástrojom pre vizualizáciu a koordináciu vývoja produktu je Target Process. Celkovo sú s daným nástrojom spokojní, vyhovuje im jeho flexibilita, možnosť nastavenia vlastných tabúľ, či možnosť vytvárania prehľadov s grafmi a trendami. Dôležitou je pre nich integrácia repozitára Gitlab a systému TeamCity určeného pre automatizované testovanie. Jednou z fáz zavádzania agilných princípov bola aj adaptácia na tento nástroj a osvojenie si návykov pre prácu s ním.

4.1.2 Malá firma

Pre porovnanie využívania agilných metodík v rôznych prostrediach som sa zamerala aj na firmu, ktorá je zložená z menšieho počtu zamestnancov (celkovo 10) a existuje v nej len jeden tím, ktorý spoločne pracuje na rôznych projektoch. Ich zameranie je vývoj nových softvérových produktov na základe špecifikácií klienta, teda sa nejedná o produkty, ktoré by už boli nasadené. Táto firma sa rozhodla pre zavedenie agilných metodík už pri jej vzniku, na základe rád a odporúčaní mentorov z podnikateľského inkubátora. V rámci tohto podporného programu pre novo vznikajúce inovatívne projekty im bolo poskytnuté špeciálne vzdelávanie aj v oblasti riadenia projektov.

Prvou zvolenou agilnou metodikou, ktorú využívali bola práve metodika Scrum. V horizonte dvoch mesiacov aktívne zavádzali a testovali tieto prístupy spolu so svojim vývojovým tímom. Prechádzali klasickým procesom Scrumu, vyvíjali projekt v cykloch, realizovali stretnutia. Po dobe dvoch mesiacov zamestnanci stále neboli ochotní túto agilnú metodiku prijať. Jednotlivé procesy pre nich boli príliš **náročné**, rovnako ako samotné stretnutia, predpokladateľne aj z dôvodu, že daný zamestnanci **nemali skúsenosť s metodikou Scrum** v praxi. Technický riaditeľ firmy sa teda rozhodol pre zmenu a zavedenie kombinácie Scrumu a Kanbanu, ktorý výrazne zmenšil množstvo procesov. Opätovne ako pri prvej firme bolo potrebné zachovať porady, preto sú čiastočne realizované ďalej. Konkrétne bol zachovaný plánovací meeting a review meeting. Súčasne využívajú aj klasické štatistické grafy stavu projektu a Sprintov ako je burndown graf a velocity graf.

Nástroj využívaný na správu behu projektu je Trello, ktorý je popísaný v podkapitole 4.2. Tím využíva na jednotlivé projekty tabule, ktoré si prispôsobuje na základe potrieb a

charakteru vyvíjanej aplikácie. V začiatkoch nasadzovania daného nástroja boli vytvorené zásuvné moduly, ktoré dopĺňajú jeho funkcionalitu a pomáhajú pri efektívnom plánovaní implementovania funkcionalít. Ako doplňujúci prvok si samotná spoločnosť vytvorila vlastné nástroje pre generovanie štatistických grafov potrebných pre plánovanie a revíziu projektov. Dôvodom pre ich implementovanie bola absencia možnosti generovania grafov definovaných metodikou Scrum v samotnom nástroji Trello.

4.1.3 Vyhodnotenie

Využívanie agilných metodík a teda aj metodiky Scrum si vyžaduje mnoho úsilia pre ich zavedenie a následné udržanie. Obe firmy sa stretli s **náročnosťou realizácie meetingov a určitých praktík metodiky Scrum**. Napriek tomu, že im samotná metodika prišla náročná na zavedenie a druhá menšia firma sa vzdala jej priameho používania, tak firma korporátneho charakteru zvolila vhodný kompromis na základe potrieb vývoja. Metodika Scrum napomáha tímom určiť jednotlivé procesy, vedie klienta k postupnej definícií požiadaviek a prípadnému pripomienkovaniu už existujúcich funkcionalít, umožňuje plánovať v určitých cykloch a je vhodná najmä na vytváranie rozsiahlych projektov s veľkým množstvom funkcionalít.

Druhou voľbou u oboch firiem bolo využitie metodiky Kanban, ktorej jadro vychádza práve z metodiky Scrum. Dôdovodom pre výber tejto metodiky spolu s kombináciou prvkov Scrumu bola menšia procesná náročnosť Kanbanu. Jedinou podmienkou pre správne fungovanie metodiky Kaban je odvádzať kvalitnú prácu. Táto zmena bola identifikovaná ako veľmi prínosná pre projekty menšieho rozsahu, ktoré obsahujú menšie množstvo funkcionalít a sú z pohľadu časového horizontu kratšie. Produkty, ktoré sú len modifikované si vyžadujú rýchlu zmenu a odozvy na vzniknutý problém.

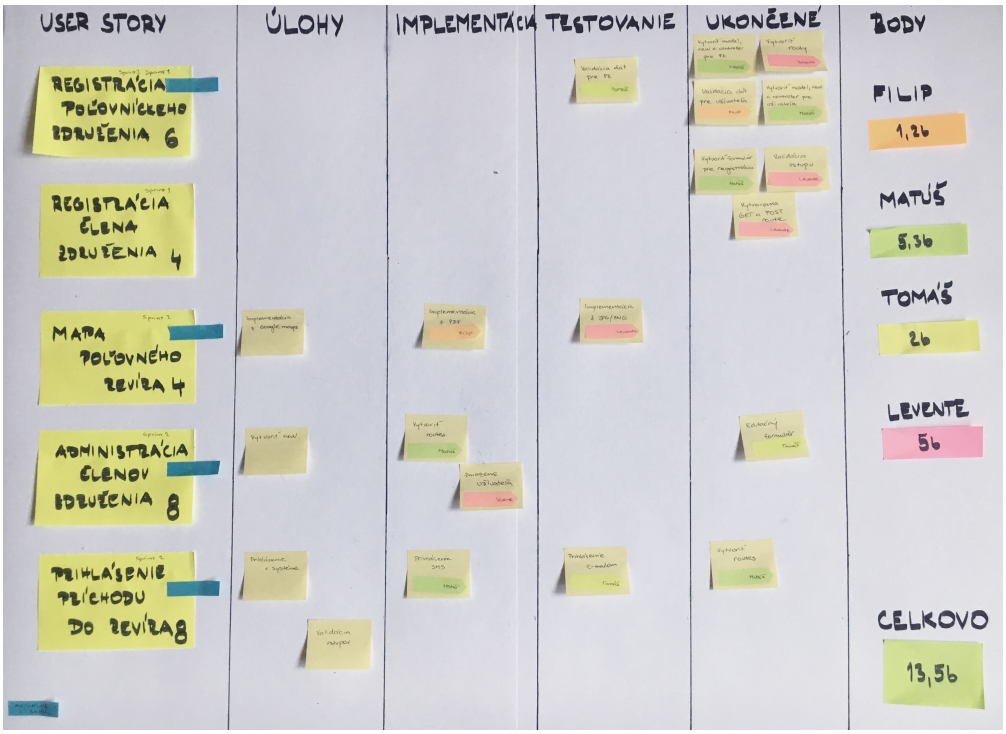
Obe firmy si pre riadenie procesu vybrali dostupný nástroj, ktorý výrazne uľahčuje komunikáciu vo vývojovom tíme, prenášanie informácií, či uchovávanie dát o projekte. Korporátna firma si zvolila nástroj s možnosťou integrácií aplikácií, ktoré aktívne využívajú. Nástroj Target Process je výrazne komplexnejší, ale i finančne náročnejší oproti nástroju Trello, ktorý si zvolila druhá malá firma. Voľba v druhom prípade bola prispôsobená nie len finančnému hľadisku, ale aj potrebe voľby nástroja, na ktorý sa jednoducho adaptujú aj vývojári, ktorý s podobným nástrojom, či metodikou ešte nemali skúsenosť.

4.2 Existujúce riešenia využívané v praxi

Jednou z dôležitých praktík pre správne fungovanie a používanie nie len metodiky Scrum, ale celkovo agilných metodík je dobrá **vizualizácia** [11]. Na trhu existujú rôzne nástroje, ktoré umožňujú správu behu vývoja aplikácie, či softvéru. Charakteristika týchto nástrojov je odlišná. Jedná sa buď o nástroje jednoduché, či komplexné, o nástroje so zameraním na jednu agilnú metodiku, či nástroje, ktoré umožňujú užívateľovi nastaviť vlastný spôsob agilného prístupu. Základnou myšlienkou je však poskytnutie určitej prehľadnej tabule, ktoré boli v skorších érach využívané v kancelárskom prostredí a u niektorých firiem pretrvávajú až do dnes. V určitých prípadoch dochádza ku kombinácií viacerých nástrojov, príkladom je súčasné používanie kancelárskej tabule so softvérovou aplikáciou, ktorá slúži ako úložisko dát o projekte a jeho behu.

4.2.1 Kancelárska tabuľa

Pre dobrú vizualizáciu aktuálneho stavu projektu, či backlogu je vhodná klasická kancelárska tabuľa umiestnená na viditeľnom mieste v dosahu celého projektového tímu [3]. Zabezpečuje prehľadnosť a čitateľnosť aj pre nestranného pozorovateľa. Náklady na jej zavedenie sú minimálne a preto je vhodné, aby takúto tabuľu začali využívať najmä tie tímy, ktoré s agilnou metodikou nemajú ešte žiadne skúsenosti a snažia sa o jej nasadenie. Jej výhodou je možnosť okamžitej modifikácie a zmeny prístupu k manipulácií s touto tabuľu, či k používanej metodike. Aktívne zapája členov tímu do denného hodnotenia, pretože sa nachádzajú v jej bezprostrednej blízkosti. Nevyžaduje si žiadnu časovú náročnosť na adaptáciu na nový nástroj, nakoľko je jej používanie prirodzené a intuitívne.



Obr. 4.2: Kancelárska tabuľa zobrazujúca aktuálne bežiaci projekt

V praxi je na tejto tabuli zobrazený beh projektu alebo Sprintu v rôznych fázach, ktorými prechádza počas vývoja. Fázy sú definované pomocou stĺpcov. Do nich sú umiestnené kartičky s popisom úlohy a unikátnou značkou člena tímu. Toto označenie slúži pre identifikáciu osoby, ktorá aktuálne príslušnú časť implementuje. Vzhľad tejto tabule je možné vidieť na obrázku 4.2. Počas doby behu projektu dochádza k presunu úloh po tabuli na základe ich aktuálneho stavu. Je možné ich presúvať v podstate do ľubovoľnej fázy v prípade, že spĺňajú odpovedajúce kritéria. Napríklad do fázy ukončenia je možné danú úlohu preniesť až v momente, kedy bola riadne otestovaná a spĺňa všetky akceptačné kritériá. Túto zmenu pritom registrujú všetci členovia tímu, ktorí sa aktuálne nachádzajú v kancelárií a majú tak vždy aktuálny prehľad o dianí.

Jej výraznou nevýhodou je nemožnosť jednoduchého generovania grafov a štatistík, či uchovávania informácií o zmenách stavu projektu. Je možné vyčleniť ľudské zdroje, ktoré

budú za zber dát zodpovedné. Takáto praktika je však časovo náročná a často nereflektuje aktuálne hodnoty alebo stav.

4.2.2 Softvérové nástroje

Úlohou softvérového nástroja je transformácia kancelárskej tabule do elektronickej podoby. Jedným z dôvodov pre využívanie elektronickej verzie je možnosť efektívnejšieho uchovávania dát, zabezpečenie prehľadu o stave projektu pre distribuované tímy alebo dosah nad dianím projektu pre manažérov, ktorí sa fyzicky nenachádzajú v rovnakej lokalite ako samotný vývojársky tím. Nevýhodou týchto nástrojov je menšia interakcia tímu ako celku s tabuľou. Každý člen tímu manipuluje so svojimi úlohami na svojom vlastnom elektronickom zariadení bez aktívnej pozornosti ostatných, ako je tomu pri reálnej tabuli. Prehľad o zmenách je v niektorých nástrojoch zabezpečený upozorneniami o aktivitách, ktoré členom tímu aj v prípade absencie umožnia zistiť aktuálny stav projektu.

Na trhu sú dostupné rôzne softvérové riešenia, s rôznou úrovňou komplexnosti, či možnosťami integrácií. Typicky v prípade komplexného nástroja, ktorý neposkytuje len prehľadnú tabuľu, ale umožňuje napríklad generovať štatistiky a grafy, či integrovať iné nástroje, platí, že je spoplatnený. Každý z týchto nástrojov má vlastné charakteristické vlastnosti a ich funkcionalita je odlišná. Vyžadujú teda alokovanie určitého času na adaptáciu členov tímu na jeho používanie aj v prípade, že už mali skúsenosť s iným nástrojom zameraným na agilnú metodiku.

ScrumDo[1]

Aplikácia ScrumDo je ucelený nástroj pre správu projektov so zameraním na metodiku Scrum. Poskytuje možnosť správy iterácií a ich plánovania. Prehľad o konkrétnom Sprinte je vizualizovaný pomocou tabule s kartami, ktoré predstavujú jednotlivé User Stories. Táto vizualizácia je zobrazená na obrázku 4.3. Prehľadná tabuľa je doplnená aj o grafy behu projektu, ktoré napomáhajú projektovému plánovaniu.

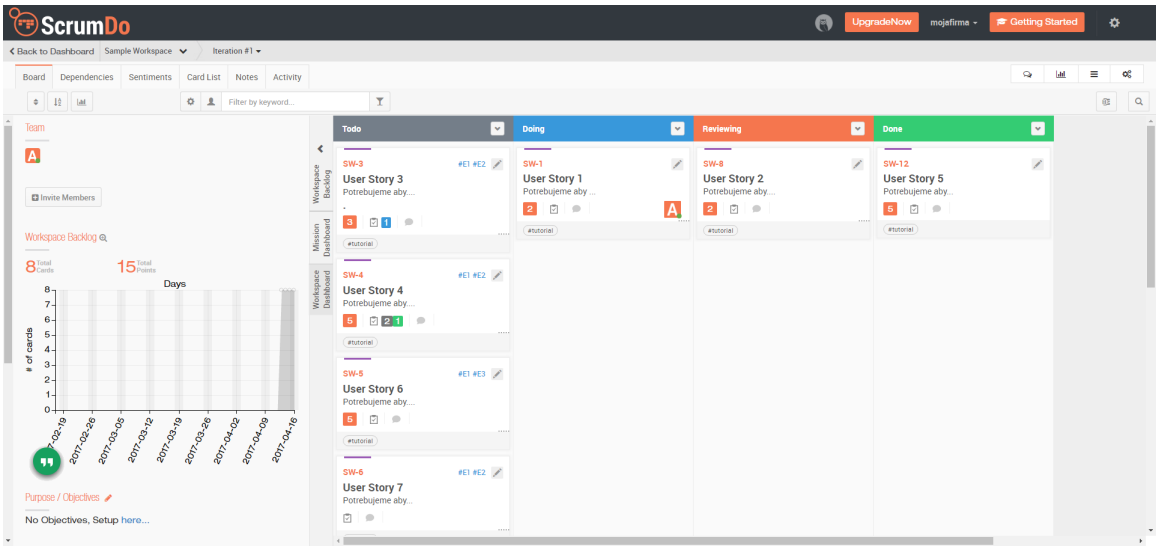
Neoddeliteľnou súčasťou aplikácie sú úlohy patriace konkrétnym User Stories. Sú zobrazené pomocou osobitnej zmenšenej tabule v rámci každej User Story. Prácu nad týmito úlohami zaznamenáva časť aktivít. Zavedenie prehľadu prác zabezpečuje členom tímu veľmi rýchly spôsob zorientovania sa v stave projektu, informovania sa o prípadných zmenách, či problémoch v implementáciách.

Nástroj je podľa môjho názoru veľmi vhodný pre tímy, ktoré sa chcú naučiť pracovať s metodikou Scrum, dodržať jej postupy a na ich základe viesť projekt. Koncepcia aplikácie prevádza užívateľa celkovým procesom. Na názornom projekte mu ukazuje ako ho má správne spravovať, na aké časti je ho potrebné rozdeliť a ako s týmito časťami pracovať. Súčasne poskytnutie štatistík a grafov a možnosť integrácie základných nástrojov používaných pri vývoji projektov na správu zdrojových kódov, komunikáciu a meranie vyťaženosti, umožňuje tímom efektívne plánovať a spravovať ich projekty bez vysokých časových nárokov.

JIRA[2]

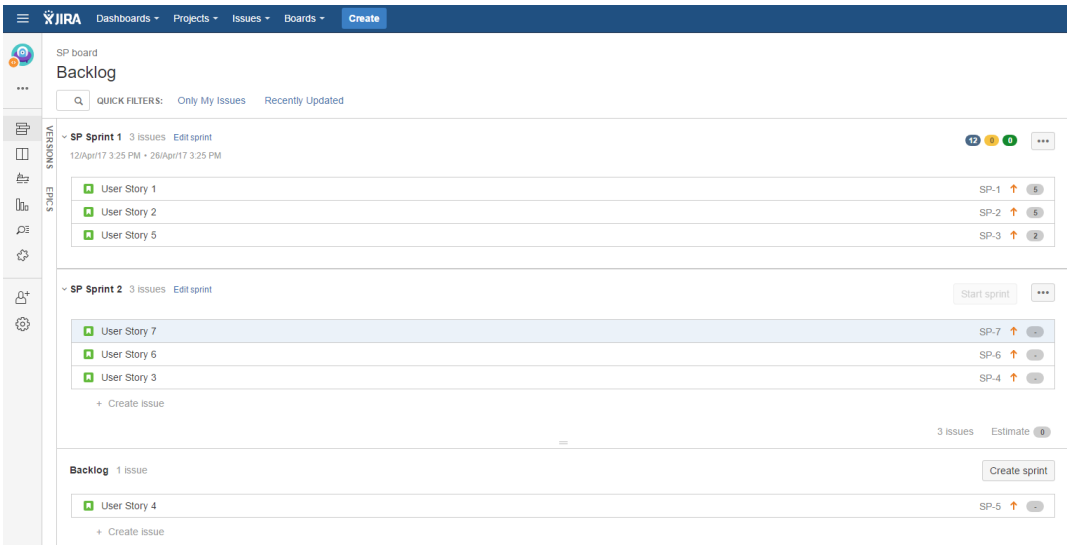
Jedná sa o komplexný nástroj, ktorý umožňuje spravovať projekt na základe rôznych metodík. V oblasti softvérového vývoja poskytuje podporu pre Scrum a Kanban a základný

[1] Dostupné na: https://www.scrumdo.com
[2] Dostupné na: https://www.atlassian.com/software/jira



Obr. 4.3: Aplikácia ScrumDo so zobrazením Scrum tabule

softvérový vývoj. Pracuje na princípe vytvárania a postupného dopĺňania požadovaných funkcionalít a ich následného roztriedenia do naplánovaných Sprintov. Súčasne je možné ku každej User Stories priradiť prioritu spracovania, prípadne ohodnotenie náročnosti. Na obrázku 4.4 je možné vidieť naplánované funkcionality a ich rozdelenie do príslušných Sprintov. Práca s týmto prehľadom je intuitívna, jednotlivé User Stories je možné premiestňovať jednoduchým ťahaním. O každej z nich je možné zobraziť podrobné informácie.



Obr. 4.4: Aplikácia Jira so zobrazením backlogu

Prevedenie tabule odpovedá fyzickej tabuli v kancelárií. Je delená do sekcií podľa jednotlivých User Stories, kde sú zobrazené úlohy potrebné pre implementovanie danej funkci-

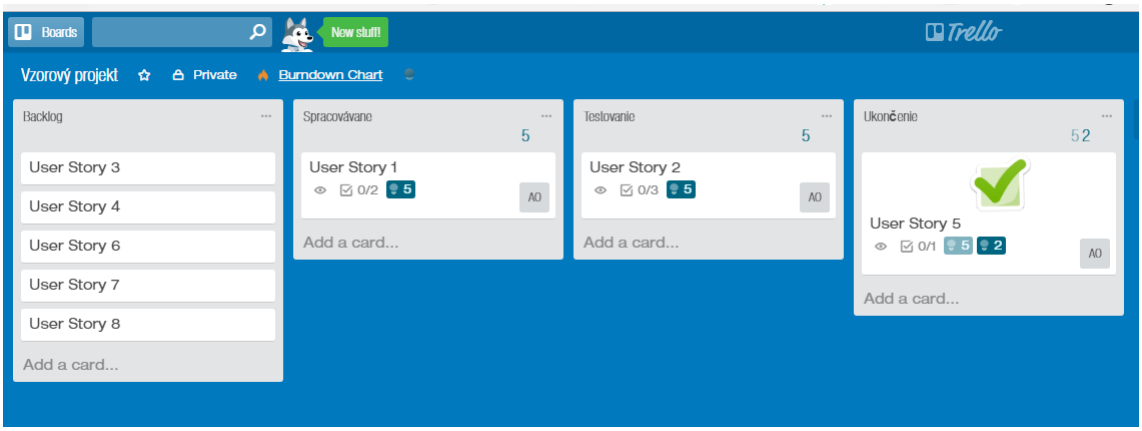
onality. V nastaveniach je možné definovať rôzne štádiá, ktorými úloha prechádza, prípadné je možné definovať druhy problémov, ktoré je potrebné v rámci vývoja vyriešiť.

Výhodou nástroja JIRA je poskytovanie štatistických a grafických výstupov na základe behu projektu. Jedná sa o klasické grafy ako je napríklad burndown graf alebo velocity graf, ale aj grafy slúžiace projektovým manažérom pre správu pracovného výkonu zamestnancov.

Aplikácia je podľa môjho názoru vhodná pre tímy, ktoré majú znalosť o agilnej metodike Scrum. Ponúka im možnosti pre celkovú správu projektu, identifikovanie stavu projektu a riadenie jeho priebehu. Jeho komplexnosť u tímov, ktoré s metodikou Scrum začínajú môže vyžadovať väčšiu časovú náročnosť na adaptáciu. Keďže nástroj ponúka možnosti správy jednotlivých úloh, zaznamenávanie informácií o nich a o priebehu ich implementácie, môže dôjsť k odpútaniu pozornosti tímu od samotnej realizácie projektu z dôvodu ich sústredenia sa na správu projektu v tomto nástroji.

Trello[3]

Aplikácia je založená na princípe práce s tabuľami. Poskytuje základné rozhranie pre správu projektov s umožnením sledovania ich stavu na základe priradenia úloh k jednotlivým skupinám. Je možné ho prispôsobiť vlastným potrebám, stanoviť si jednotlivé etapy procesu vývoja, prípadne pridať rozšírenia a pripojenia na iné vývojové nástroje. V prípade, že sa vývoj riadi metodickými postupmi Scrumu, je nástroj Trello možné používať v zmysle tejto metodiky. Tím, ktorý sa snaží o zavedenie metodiky Scrum môže mať mierne problémy s pochopením metodiky. Z dôvodu neexistencie súčasného prehľadu o Product Backlogu a jednotlivých Sprintov môže dôjsť k určitej nekonzistencií dát a úloh a k zmenšeniu prehľadnosti o celkovom stave projektu.



Obr. 4.5: Aplikácia Trello so zobrazením tabule Sprintu

Existujú dve varianty k prístupu správy tabule. Prvou možnosťou je považovať tabuľu za celkový projekt. V tomto prípade má síce tím prehľad o všetkých funkcionalitách, ktoré treba implementovať, ale nie je možné konkrétnu User Story priradiť nejakému Sprintu. Druhou možnosťou je vytvorenie niekoľkých tabúľ pre jeden projekt, kde každá tabuľa predstavuje jeden Sprint. Tento prípad je zobrazený na obrázku 4.5. Tím má následne jasný prehľad o tom, čo je potrebné v danom Sprinte implementovať. Stráca sa však komplexný prehľad o celkovej funkcionalite a zvyšuje sa náročnosť potrebná pre plánovanie.

[3]Dostupné na: https://trello.com

V oboch prípadoch je možné zobraziť len jednotlivé User Stories. Úlohy, ktoré súvisia s ich implementáciou sú skryté v karte.

Vďaka jednému z bezplatných rozšírení je možné pri User Story definovať predpokladanú náročnosť a i výslednú spálenú náročnosť. Spracovanie týchto hodnôt do grafov ako je burndown graf je už spoplatneným rozšírením a nie je priamou súčasťou aplikácie.

Celkovo je tento nástroj pre potreby riadenia projektu podľa metodiky Scrumu podľa mňa priemerný. V prípade, že sa má tím správne naučiť princípy čistého Scrumu, tak ich táto aplikácia nenavádza. Zároveň neposkytuje celkový štatistický prehľad o priebehu a celkovom stave projektu. Trello by však mohlo byť vhodné pre tímy, ktoré už majú skúsenosti s agilným prístupom Scrumu, poznajú postupy a riadia sa nimi. Následne im tento nástroj slúži len ako podporný.

Výsledok analýzy nástrojov

Cieľom analýzy existujúcich nástrojov bolo zhodnotenie silných a slabých stránok pre vedenie projektov na základe princípov metodiky Scrum. Dostupné nástroje na trhu zabezpečujú pomerne veľký výber funkcionalít a pomáhajú tímom s koordináciou ich práce, plánovaním a hodnotením projektov. Na základe ich komplexnosti by som ich charakter rozlíšila do dvoch skupín. Prvou skupinou sú nástroje veľmi jednoduché, neorientované na konkrétnu metodiku. Tieto nástroje zabezpečujú typicky základné funkcie pre prehľad implementovaných úloh, často však už neposkytujú grafické hodnotenia. Druhou skupinou nástrojov sú tie, ktoré sú orientované na konkrétnu metodiky (v tomto prípade metodiku Scrum). Takáto orientácia umožňuje väčšiu komplexnosť nástroja, ale je spojená s vyššou náročnosťou pre adaptáciu.

Na základe testovania funkcionalít na vzorovom projekte vo všetkých troch vyššie spomenutých nástrojoch som sa postupne snažila o vytvorenie základného konceptu mojej aplikácie. Ako slabšiu stránku som u nástrojov brala absenciu grafov, ktoré som následne zahrnula do svojej aplikácie. Snažila som sa však využiť len základné grafy potrebné pre vedenie a nepridávať časti, ktoré by zaťažovali ako systém, tak aj užívateľa. Veľmi inšpiratívnou bola pre mňa forma prevedenia nastavením projektu v aplikácií ScrumDo, ktorá bola veľmi vhodná najmä pre užívateľov, ktorí s takýmito systémami nemajú skúsenosť. Takáto nápoveda bola zahrnutá aj do návrhu mojej aplikácie, avšak v inej forme. Snažila som sa o spojenie zodpovedností, ktoré má vykonávať užívateľ s konkrétnou rolou, a funkcionalít, ktoré má v systéme využívať. Súčasne som sa mu snažila priblížiť proces, ktorým si má ako člen projektového tímu prejsť.

Kapitola 5

Návrh riešenia nástrojovej podpory

Cieľom bakalárskej práce bolo vytvorenie nástroja pre projektovú podporu vývoja softvéru, ktorý pomôže firmám a ich tímom k osvojeniu základných praktík metodiky Scrum. Pri výbere správnej softvérovej podpory je nutné brať v úvahu schopnosti a znalosti vývojárov. Tím, ktorý má skúsenosti s určitou agilnou metodikou, či priamo metodikou Scrum, dokáže používať aj jednoduchý nástroj, obsahujúci len základné prvky ako sú tabule a karty, pretože sú schopní sami riadiť projekt podľa princípov metodiky. Rovnako je pre nich vhodný aj komplexný nástroj, pretože majú prehľad nad jednotlivými poskytovanými funkcionalitami, disponujú znalosťou s ich manipuláciou a práca s nimi neodvádza väčšinovo ich pozornosť od implementácie softvéru. Je však dôležité zamerať sa aj na tímy, ktoré s metodikou Scrum skúsenosť nemajú a snažia sa len o jej nasadenie. Navrhnutá nástrojová podpora sa snaží poskytnúť užívateľom prevod celkovým procesom na základe princípov metodiky Scrum tak, aby boli používané správnym spôsobom.

Vychádzajúc zo znalosti potrieb v reálnych firmách pre prácu na projektoch v distribuovaných tímoch, či potreby prístupu do systému v rôznych lokalitách, bolo riešenie navrhnuté ako webová aplikácia. Takéto rozhranie zabezpečuje možnosť pripojenia z akéhokoľvek zariadenia s prístupom na internet. Aplikácia je navrnutá ako klient-server riešenie. Očakávané je teda vytvorenie základného rozhrania, ktorého úlohou bude získavanie a poskytovanie dát klientskej aplikácií. Z dôvodu predpokladu poskytovania dynamického obsahu klientskému rozhraniu bolo potrebné navrhnúť samotný databázový model. Tento model určuje spôsob uchovávania dát a ich vzájomné prepojenia. Komunikácia s databázou následne zabezpečuje aktualizáciu dát a ich získavanie pre ďalšie potreby. Analýza požiadaviek určila celkový prehľad funkcionalít, ktoré majú byť užívateľovi sprístupnené. Tieto funkcionality pracujú s databázou, spracovávajú jej dáta a následne ich predávajú klientovi v čo najzrozumiteľnejšej forme. Počíta sa aj s umožnením viac užívateľského prístupu do aplikácie s rozlíšením rolí jednotlivých užívateľov. Pre prípad potreby práce na inom zariadení ako je stolný počítač, či notebook bolo rozhranie aplikácie navrhnuté s ohľadom na potrebu responzívneho zobrazenia. Celkové riešenie bolo zamerané na vytvorenie základných funkcionalít spojených s potrebami koordinácie procesu vývoja softvéru pomocou metodiky Scrum.

5.1 Analýza požiadaviek

Aplikácia by mala rozlišovať celkovo 3 základné role užívateľov. Administrátorské práva sú udelené užívateľovi, ktorý v projektovom tíme figuruje ako Scrum Master. Pred možnos-

ťou prístupu do aplikácie je zodpovedný za registráciu firmy a zvolenie tímového mena, na základe ktorého sa registrujú ďalší zamestnanci spoločnosti. Jeho úlohou by malo byť vytváranie všetkých projektov a ich následná správa. Pod správou sa rozumie spúšťanie a ukončenie projektu, úprava jeho detailov, či projektového tímu. Napriek tomu, že metodika nedefinuje Scrum Mastera ako čiste manažérsku pozíciu, tak je dôležité, aby niekto sledoval postup na projekte, či výkonnosť zamestnancov. Z tohto dôvodu je mu umožnené pristupovať ku typickým grafom ako je burndown a velocity graf s možnosťou ich selekcie podľa zvolených parametrov. Sú mu zobrazené aj súhrnné informácie o množstve vykonanej práce na jednotlivých projektoch.

Za spúšťanie a ukončenie Sprintu je rovnako zodpovedný Scrum Master, za jeho plánovanie je už však zodpovedný Product Owner. Jeho úlohou je definícia User Stories, ktoré majú byť implementované pre úspešné dodanie vytváraného softvéru. Tieto User Stories následne prispôsobuje a plánuje za pomoci ich editácie a udržiava tak Product Backlog neustále aktuálny. V prípade potreby naplánovania ďalšieho Sprintu mu je umožnené jeho pridanie. Product Owner je zodpovedný za udržiavanie aktuálneho Product Backlogu na základe požiadaviek klienta, pričom priority User Stories by mal uspôsobovať na základe nutnosti implementácie určitej funkčnosti. Scrum Master v prípade absencie Product Ownera zodpovedá za jeho úlohy a sú mu sprístupnené všetky potrebné funkcionality pre ich realizáciu. Tento vzťah generalizácie je možné vidieť v prílohe A. Sú v ňom znázornené všetky operácie, ktoré môžu byť v systéme vykonané jednotlivými užívateľmi.

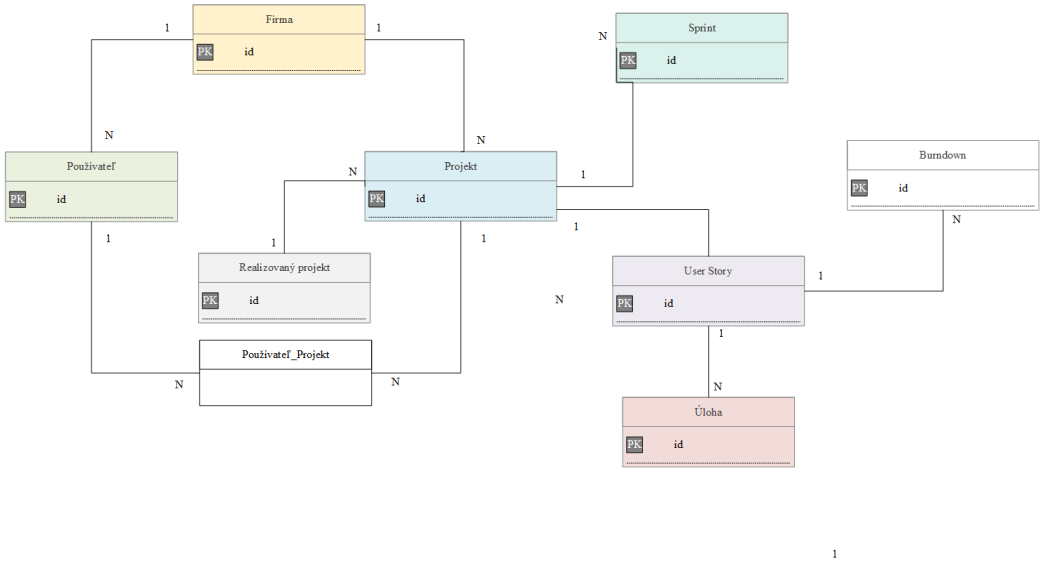
Poslednú uživateľskú rolu v nástroji predstavuje samotný vývojár alebo teda Team Member (člen tímu). Pre potreby sledovania behu projektu a implementácie jeho jednotlivých častí je mu zverená správa úloh prislúchajúcich k User Story. Tieto úlohy si môže prevziať, čím sa zaväzuje k plnej zodpovednosti za ich implementáciu. V prípade, že úlohu nedokáže spracovať, tak je mu umožnené túto úlohu odovzdať a následne si ju môže prevziať iný člen tímu.

Ostatné operácie v systéme sú spoločné pre všetky role užívateľov. Jedná sa o operácie typu prihlásenia a odhlásenia zo systému a následnej správy osobného profilu spolu so zobrazením osobných štatistík. Každý užívateľ si môže zobraziť aktuálne projekty, na ktorých pracuje a následne si zobraziť detail projektu. Sprístupnený je i Product Backlog, kde si môžu prezerať detaily jednotlivých User Stories. Hlavnou časťou je zobrazenie tabule aktuálne bežiaceho Sprintu. Dôležitou je aj história, ktorú si všetci užívatelia môžu zobraziť. Prestavuje prehľad o realizovanom projekte, jeho Sprintoch a User Stories, ktoré boli implementované a následne naplánové.

Výstupom analýzy požiadaviek bol Use Case diagram priložený v prílohe A. Tento diagram súčasne odráža aj zmeny zavedené vo fáze testovania.

5.2 ER diagram

Pri návrhu databázy bolo identifikovaných 9 entitných množín. K základným entitným množinám patrí používateľ, firma, projekt, sprint, user story a úloha. Pre potreby uchovávania štatistických údajov bola vytvorená entitná množina burndown. Databáza bola navrhovaná tak, aby bol prístup k jednotlivým položkám čo najefektívnejší. Znázornenie všetkých entít a ich vzájomných vzťahov je možné vidieť na obrázku 5.1. Entity spolu s ich atribútmi sú uvedené v prílohe A.



Obr. 5.1: Schéma databázy zobrazujúca základné entity s ich reláciami

Firma

Firma predstavuje unikátne zoskupenie zjednocujúce všetky projekty a užívateľov, ktoré súvisia s danou spoločnosťou. Medzi jej atribúty patrí meno firmy a meno tímu. Pomenovanie tímu je využívané vo formulári pre registráciu užívateľa do časti systému, ktorá prislúcha jeho firme. Súčasťou návrhu databázy bolo i previazanie projektu a užívateľa so samotnou firmou pre potreby zabezpečenia informácií iných firiem.

Používateľ

Entitná množina používateľa je určená atribútmi ako meno a priezvisko pracovníka, jeho kontaktné informácie (e-mail, telefón) a relatívna cesta k súboru s jeho profilovou fotografiou. Súčasne má každý používateľ identifikovanú svoju rolu – Scrum Master, Product Owner alebo Team Member. Login a e-mail predstavujú unikátnu hodnotu v rámci databázy. Z dôvodu potreby autorizácie užívateľa pri vstupe do systému je medzi atribúty zaradené heslo, ktoré je kódované hash funkciou, aby bola zabezpečená bezpečnosť v prípade neoprávneného získania dát z databázy. V prípade, že by užívateľ pracoval vo viacerých firmách súčasne, musia byť pre neho vytvorené osobitné účty. Počet účastí na projektoch však nie je nijakým spôsobom obmedzený.

Projekt

Je identifikovaný pomocou názvu projektu a mena firmy, ktorá predstavuje stranu objednávateľa. V tejto entitnej množine je možné pridať stručný popis zamerania projektu pre upresnenie jeho cieľa a odovzdávanej hodnoty. Každý projekt má priradeného Scrum Mastera a Product Ownera, tieto hodnoty predstavujú cudzie kľúče do tabuľky používateľov. Je možné definovať preferovanú dĺžku Sprintu, ktorá je následne použitá pri ich spúšťaní ako základná hodnota. Dôležitým atribútom je identifikácia stavu projektu, projekt sa môže nachádzať v stave čakanie, realizácia alebo ukončené.

Realizovaný projekt

Slúži ako pomocná entitná množina určená k definícií aktuálne bežiacich projektov. Jej vytvorenie bolo navrhnuté z dôvodu optimalizácie a zmenšenia náročnosti dotazov do databáze. V určitých prípadoch postačuje zistiť, či je daný projekt realizovaný a nie je potrebné získavať jeho ďalšie atribúty. Zaznamenáva len samotnú identifikáciu projektu, pričom jej inštancia by mala byť vytváraná v prípade spustenia projektu a a zmazaná v prípade jeho ukončenia.

Sprint

Základným stavebným kameňom agilnej metodiky Scrum je princíp iterácie. Pre potreby ich realizácie bola vytvorená entitná množina Sprint. Môže sa nachádzať v stavoch čakanie, realizácia alebo ukončenie podobne ako projekt. Dĺžka Sprintu je stanovená vo forme počtu dní od dňa zahájenia. Sprint je vrámci projektu identifikovaný aj svojim poradovým číslom.

User Story

V rámci každého projektu sú vytvárané User Stories. Dôležitými atribútmi pri vytváraní novej User Story je jej názov, ktorý definuje aký cieľ má dosiahnuť jej implementácia a obsiahlejší popis, ktorý podrobnejšie špecifikuje požadovanú funkcionalitu. Priorita definuje mieru potreby implementácie danej User Story a jej náročnosť sa odráža v bodovom ohodnotení. Stav definuje, či čaká na zaradenie do Sprintu, je v určitom Sprinte naplánovaná alebo už bola ukončená, t.j. bola úspešne implementovaná.

Úloha

Táto entitná množina je vytváraná pre potreby definovania implementačných postupov, ktoré je potrebné vykonať pre realizáciu požadovanej funkčnosti systému v rámci určitej User Story. Každá úloha má definovaný stav, v ktorom sa nachádza. Tieto stavy sú čakanie, implementovanie, testovanie a ukončené. Za úlohu je zodpovedný určitý člen tímu, ktorý je zaznamenaný ku úlohe v momente jej prevzatia.

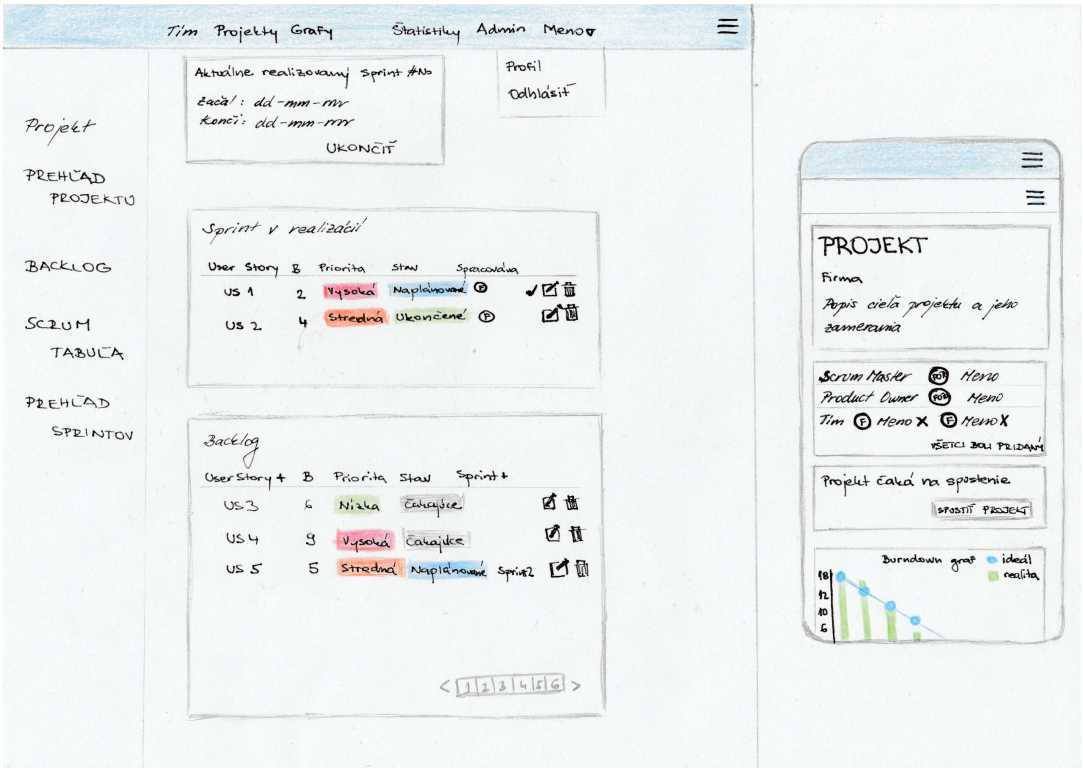
Burndown

Pre potreby uchovávania informácií o jednotlivých funkcionalitách bola vytvorená entitná množina burndown. Záznam v tejto tabuľke je vytvorený vždy po ukončení User Story. Pri každom zázname je uvedený počet bodov, ktorý prislúcha členovi tímu za odvedenú prácu na danej User Story. Z dôvodu optimalizácie a potreby výberu veľkého množstva záznamov z databázy, boli do tejto entitnej množiny vsunuté aj identifikátory do tabuľky Sprintov, projektov a firiem. Dátum zápisu určuje deň, kedy bol daný záznam zapísaný do databázy.

5.3 Návrh rozhrania

Samotnému procesu implementácie nástroja predchádzal návrh jeho rozhrania, rozvrhnutie častí nástroja a určenie rozmiestnenia prvkov v týchto častiach. Všetky návrhy boli realizované pre administrátorskú rolu, teda boli určené všetky prvky pre jednotlivé funkcionality systému. Rozhranie bolo rozvrhnuté do sekcií podľa potrieb jednotlivých členov tímu. Ako

základný prvok boli zvolené karty pre potrebu možnosti manipulácie s nástrojom na mobilnom zariadení alebo tablete. Počítalo sa aj so skrytím bočného panela na mobilnej verzii a jeho nahradením navigačnou lištou.



Obr. 5.2: Ručný návrh realizácie webového rozhrania

Scrum Master je zodpovedný za celkový beh projektu a veľmi dôležitou je pre neho časť s informáciami o projekte. Okrem stručného popisu projektu a určenia firmy, pre ktorú je produkt dodávaný, je potrebné Scrum Mastera upovedomiť o stave projektu. Pre tento účel bola zvolená osobitná karta, ktorá určuje či je projekt spustený, čaká na spustenie alebo bol ukončený. V prípade, že je projekt v realizácií, tak je užívateľovi poskytnutá informácia o dátume zahájenia projektu o dátume predpokladaného ukončenia, či o bežiacom Sprinte. Správne projektové riadenie a plánovanie je podporené zobrazením burndown a velocity grafu. Pre účely analýzy celkového priebehu projektu sú Scrum Masterovi sprístupnené oddelené časti systému, ktoré umožňujú generovanie grafov na základe výberu projektu a k nemu prislúchajúcemu Sprintu. Táto časť systému je zobrazená na obrázku 5.2 vo forme mobilného rozhrania.

Zobrazenie backlogu spolu s naplánovanými a nezaradenými User Stories bola navrhnutá ako osobitná časť. Je rozdelená na dve sekcie. Jedna sekcia zobrazuje informácie o User Stories, ktoré sú priradené do aktuálne realizovaného Sprintu. Ich vizualizácia je znázornená na obrázku 5.2 v ľavej časti. V prípade, že sa nejaká User Story nachádza v Sprinte, ktorý má byť realizovaný alebo je doposiaľ nezaradená, tak je umiestnená v druhej sekcii zobrazujúcej celkový backlog. Prídavná karta v hornej časti okna zobrazuje informácie o Sprinte. V rámci konkrétneho projektu je pridaná pravá bočná navigačná lišta, ktorá umožňuje pohyb medzi základnými funkcionalitami súvisiacimi s projektom.

Napriek tomu, že sa firma typicky zameriava na aktuálne bežiace Sprinty je dôležité, aby sa tímy spätne vracali k projektom a ich výsledkom. Preto bola navrhnutá časť prehľadu Sprintov, ktorej úlohou je zobraziť všetky informácie o projekte. Obsah zobrazenia pre projekt v realizácií sa odlišuje od zobrazenia pre ukončený projekt. Zobrazenie pre aktuálne realizovaný projekt výrazne pripomína zobrazenie Product Backlogu. Rozdielom je, že sa tu zobrazujú usporiadane Sprinty, ktoré sú naplánované, prípadne, ktoré boli ukončené spolu s prislúchajúcimi User Stories. Ak bol projekt už ukončený, zobrazuje sa súhrná informácia o tom, koľko Sprintov a User Stories bolo naplánovaných a koľko z nich sa v skutočnosti zrealizovalo. V jednej časti sú následne zobrazené tie User Stories, ktoré sa implementovali a v druhej tie, čo neboli zaradené alebo prislúchali ku Sprintu, ktorý nebol počas behu projektu spustený.

Kapitola 6

Implementácia nástrojovej podpory

Nasledujúca kapitola popisuje implementáciu nástroja pre podporu vývoja softvéru a vychádza z princípov a návrhov popísaných v kapitole 5. Aplikácia bola navrhnutá ako klient-server riešenie, pričom klienta predstavuje webový prehliadač užívateľa. Výsledná aplikácia je dostupná na adrese scrum.osterova.sk. V aplikácií je pripravený vzorový projekt, ktorý vychádzal z testovacej fázy. Prihlásiť sa do neho možno pomocou testovacích účtov uvedených v tabuľke 6.1.

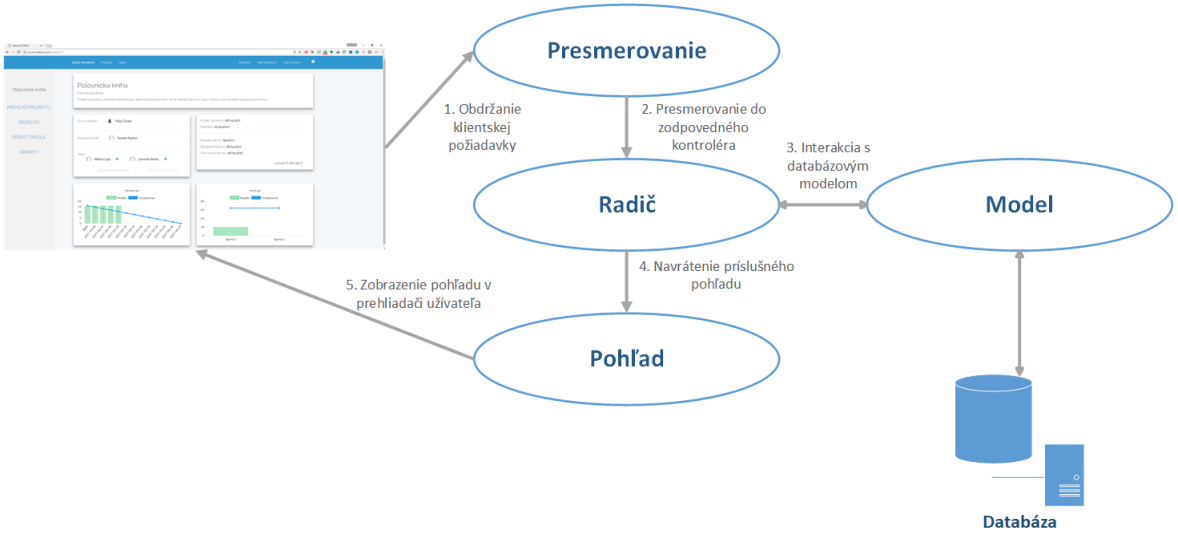
Rola	**E-mail**	**Heslo**
Scrum Master	*scrum@test.sk*	secret
Product Owner	*product@test.sk*	secret
Team Member	*team@test.sk*	secret

Tabuľka 6.1: Prihlasovacie údaje do vzorového projektu aplikácie

6.1 Koncept implementácie aplikácie

Pre implementáciu bol zvolený PHP framework Laravel[4], ktorý pracuje na princípe model-view-controller (ďalej len MVC). Návrhový vzor MVC sa skladá z troch základných komponent, ktorých prepojenie je možné vidieť na obrázku 6.1.

- **Model** je založený na objektoch reálneho sveta. Typicky sa jedná o trvalé komponenty uložené mimo samotnej aplikácie, napr. v databáze. Súčasťou modelu sú dáta, ale aj pravidlá určujúce množinu operácií nad týmito dátami.
- **View** (pohľad) predstavuje vizuálnu reprezentáciu modelu. Jedným z príkladov je využitie zobrazenia stránky pomocou jazyka HTML. Táto vrstva zodpovedá za zobrazenie dát klientovi.
- **Controller** (radič) vytvára prepojenie medzi modelom a pohľadom. Po odoslaní požiadavky z klientskej strany dochádza k jej spracovaniu kontrolérom, pričom je vykonaná príslušná operácia a navrátený nový pohľad.



Obr. 6.1: Prepojenie komponent MVC v Laravel aplikácii [1]

Proces implementácie nástrojovej podpory sa skladal z vytvorenia jednotlivých častí popísaných vyššie. Prvou vytváranou komponentou boli modely, ktoré určovali jednotlivé entitné množiny definované v priebehu návrhu aplikácie. Po implementácii modelov a vytvorení k nim príslušných migrácií, ktoré definovali položky záznamov v databáze, bolo možné vytvoriť jednotlivé pohľady. Tieto pohľady sú implementované pomocou jazyka HTML. Ich zobrazenie je umožnené vďaka obdržaniu požiadavky Laravel smerovačom, ktorý na základe príslušnej adresy určí pomocou smerovacieho súboru zodpovedný kontrolér. Kontrolér po získaní príslušných parametrov typicky spolupracuje s modelom, ktorý umožňuje komunikáciu s databázou. Funkconalita kontrolérov bola vytváraná priebežne na základe operácií, ktorými jednotlivý užívatelia systému potrebovali disponovať. Celkovo je možné typy týchto funkcionalít rozdeliť do troch skupín:

- **Získanie zobrazenia** - predstavuje najjednoduchšiu funkcionalitu, kedy prostredníctvom modelu dochádza k získaniu základných dát z databázy a k ich predaniu pohľadu.
- **Zmena databázy** - umožňuje pridanie nových prvkov do databázy, zmenu jednotlivých položiek, či úplne zmazanie záznamu. Takúto funkcionalitu bolo potrebné zaviesť v prípade registrácie a administrácie užívateľov, vytvárania nových projektov, či udržiavania aktuálneho stavu backlogu.
- **Vykonanie algoritmu** - bolo potrebné najmä pre štatistické účely. Boli vytvorené jednoduché algoritmy pre generovanie štatistických údajov na základe záznamov databázy. Opätovne po komunikácií s modelom bolo možné získať potrebné dáta, spojiť ich do štatistických celkov a následne ich odovzdať pohľadu, ktorý obsahoval skript napríklad pre generovanie grafov.

Zreteľ bol braný aj na samotnú bezpečnosť aplikácie a na zamedzenie prístupu k neoprávneným dátam. Kontrola oprávnenia pre prístup k dátam alebo funkcionalite je vykonávaná ešte pred samotným presmerovaním do konktroléra pomocou tzv. middleware.

Ich súčasťou je definícia pravidiel, ktoré musia byť splnené, aby bol umožnený prístup k dotazovanej časti systému. Jedným z príkladov je CompanyMiddleware, ktorý slúži pre overenie, či dáta, ku ktorým sa užívateľ snaží pristúpiť, patria firme, do ktorej je on sám zaregistrovaný. Vyhľadáva tak rôzne položky v databáze, napríklad na základe zadaného identifikátora úlohy, či Sprintu a porovnáva ho s identifikačným číslom firmy uvedeného v zázname prihláseného užívateľa. Ak dôjde k ich zhode sú preposlané do zobrazenia dáta, prípadne je vykonaná príslušná sekvencia príkazov pre zmenu parametrov databázy. Ak však zhoda nenastane, je užívateľ presmerovaný na hlavnú stránku a upozornený, že pre prístup k daným dátam nemá oprávnenie.

6.2 Databázová vrstva

Databáza bola vytvorená na základe schémy dátového modelu, ktorá je umiestnená v prílohe A. Jednotlivým entitným množinám odpovedajú modely aplikácie. Pre každý model boli definované relácie s inými modelmi. Databáza obsahuje všetky druhy relácií, pričom pre reláciu N:N bola vytvorená pomocná tabuľka. Nasledujúce ukážky kódu predstavujú spôsob vytvorenia relácie 1:N medzi dvoma modelmi.

```
/* model Sprint*/
public function project(){
    return $this->belongsTo('App\Project');
}

/* model Controler*/
public function sprint(){
    return $this->hasMany('App\Sprint');
}
```

Súčasťou modelov sú aj statické funkcie navracajúce dáta z databázy, ktoré sú použité v pohľadoch. Využívanie kontroléra v takýchto prípadoch nie je vhodné. Jedná sa o operácie ako zistenie, či niekto pracuje na danom projekte alebo, či je v rámci projektu aktuálne realizovaný Sprint.

6.3 Riadiaca vrstva

Za výkon funkcionality, získavanie dát z databázy a jej modifikáciu sú zodpovedné kontroléry, ktoré vytvárajú prepojenie medzi modelom a samotným zobrazením. Nadradenou triedou kontrolérov je `BaseController`.

- `AdminController` zodpovedá za administráciu užívateľov, spravovanie ich priradenej role, či prípadné zmazanie užívateľa z firmy s následnou úpravou záznamov v databáze. Ak dochádza k zmazaniu užívateľa, tak úlohy, ktoré spracovával sú prenechané na ďalšie spracovanie pre iných užívateľov.
- `BacklogController` jeho úlohou je výber aktuálne bežiaceho Sprintu a k nemu prislúchajúcich User Stories a ostatných Sprintov a neimplementovaných User Stories

z databázy. Tieto dáta sú následne odovzdané do backlogového zobrazenia a spracované.

- `GraphsController` je využitý pre zobrazovanie grafov a výber projektu a k nemu patriacich Sprintov na základe zadaných parametrov. Tieto údaje sú využívané v skriptoch pre generovanie grafu.
- `ProfilController` zabezpečuje nie len možnosť zobrazenia profilu užívateľa, ale aj jeho modifikáciu a prípadné uloženie fotografie užívateľa. Jeho súčasťou sú aj funkcie pre generovanie personalizovaných grafov, ktoré slúžia užívateľovi na zobrazenie jeho vlastnej výkonnosti. Je možné zvoliť výber výkonnosti na všetkých projektoch v konkrétnom mesiaci alebo na konkrétnom projekte za celý aktuálny rok.
- `ProjectController` predstavuje najobsiahlejší kontrolér. Zodpovedá za pridávanie, editáciu a mazanie projektu. Dôležitou je však aj samotná správa projektu, ktorá v sebe zahŕňa jeho spustenie a zastavenie, či možnosť pridania člena tím. V prípade zmazania osoby z tímu sú jej úlohy prenechané ďalším spolupracovníkom. Je v ňom zahrnutá funkcionalita pre generovanie velocity grafu a burndown grafu viazaného na aktuálne bežiaci Sprint. Pre potreby analýzy celkového behu projektov Scrum Masterom bola pridaná možnosť výberu konkrétneho projektu a Sprintu, na základe ktorých sú vygenerované dáta pre graf.
- `SprintController` je zameraný na správu Sprintov, ich spúšťanie a zastavovanie.
- `StatisticsController` zodpovedá za získanie základných dát o projektoch ako sú množstvá odpracovaných bodov, či pracovníkov, ktorí na daných projektoch pracovali
- `TaskController` umožňuje užívateľovi nie len zobraziť samotnú tabuľu, ale aj manipuláciu s úlohami. Manipuláciou sa rozumie prevzatie úlohy do spracovania, či jej prenechanie a presúvanie karty s úlohou na tabuli so súčasnou zmenou ich stavu. Každú úlohu je však do systému nutné zaviesť a v prípade potreby odobrať.
- `UserStoryCreateController` má za úlohu užívateľovi poskytnúť základnú funkcionalitu pre pridávanie, editáciu a mazanie User Stories. Pre Scrum Mastera je dôležitá možnosť schválenia User Story pomocou špecializovanej funkcie, ktorá zabezpečí vytvorenie záznamov pre štatistické potreby.

Ďalšie nespomenuté kontroléry zabezpečujú registráciu, či prihlásenie alebo získanie dát a ich zaslanie do pohľadu, čo zabezpečí ich následné zobrazenie.

Jednou z najzaujímavejších častí aplikácie je poskytnutie štatistických nástrojov pre užívateľov, ktoré im umožnia analýzu stavu stávajúceho projektu, či retrospektívnu analýzu s celkovým vyhodnotením úspešnosti projektov. Okrem základného zobrazenia údajov o počte odpracovaných bodov rozdelených po projektoch a konkrétnych vývojároch, sú údaje poskytnuté aj vo forme grafov. Grafy sú generované pomocou knižnice `chart.js`[1]. Patrí k nim napríklad burndown a velocity graf zobrazené na obrázku 6.5 umiestneného v podkapitole 6.5 zaoberajúcou sa užívateľským testovaním. Sú spúšťané pomocou skriptu, ktorý je umiestnený priamo v odpovedajúcom pohľade. Tento skript získava údaje za pomoci funkcie umiestnenej napr. v `ProjectControllery`. Po zavolaní tejto funkcie dochádza k výberu dát z databázy z tabuliek Burndown a UserStory. Následne prechádzajú algoritmom, ktorého

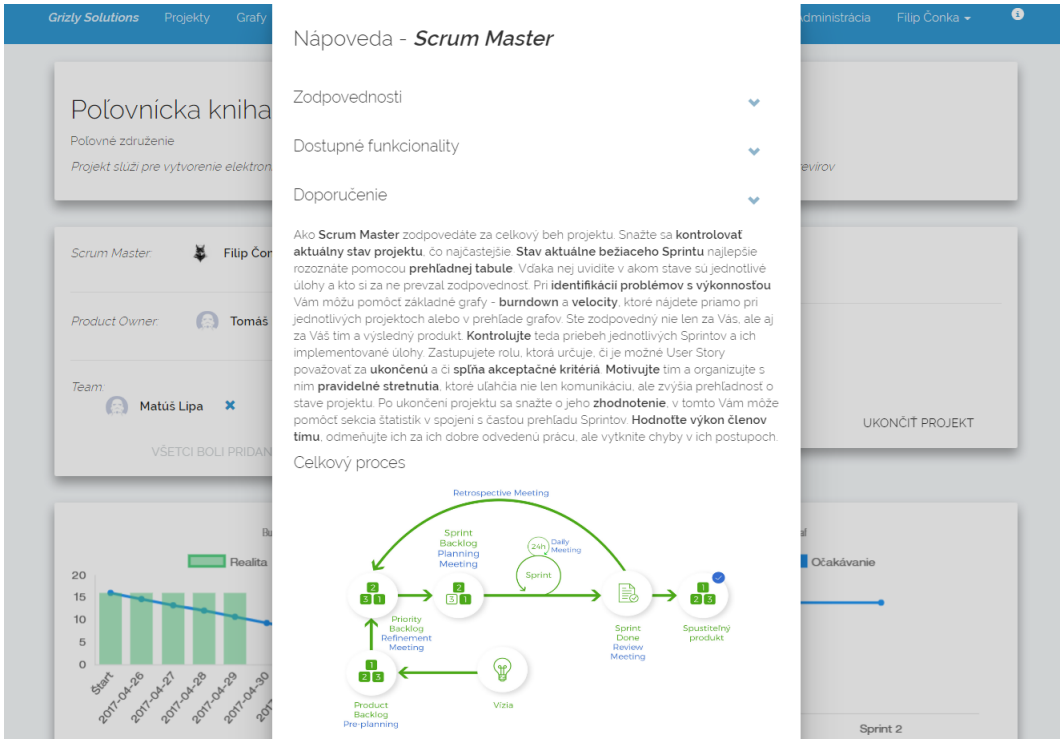
[1] Dostupné na: http://www.chartjs.org/docs/

úlohou je vytvorenie poľa s prednastavenými hodnotami pre ukladanie dát. Tieto dáta sú následne modifikované a doplnené na základe burndown záznamov o príslušné body v miestach zhody dátumov. Návratovou hodnotou funkcie je multidimenzionálne pole, ktoré sa skladá z poľa uchovávajúceho reálne štatistické hodnoty a druhého poľa, ktoré definuje ideálnu priamku vypočítanú na základe pomeru súčtu bodov a dĺžky Sprintu.

6.4 Funkcionalita a vzhľad výsledného rozhrania

Celkový vzhľad nástroja a rozloženie prvkov bolo volené na základe návrhov, ktorých časť bola popísaná v kapitole 5. Prvky sú vybrané v súlade s princípmi materiálneho dizajnu[2].

Nástroj je cielený najmä na tímy, ktoré sa snažia o nasadenie agilnej metodiky Scrum a hľadajú nástroj, ktorý im pomôže pri osvojení praktík tejto metodiky. Preto je do nástroja pridaná sekcia s radami pre prácu s aplikáciou. Výber rád je uspôsobený na základe role užívateľa. Jeho vzhľad je možné vidieť na obrázku 6.2. Dôvodom pre toto odlíšenie je rozdielny charakter vykonávanej činnosti Scrum Mastera, Product Ownera a člena tímu.



Obr. 6.2: Nápoveda pre užívateľa s rolou Scrum Master

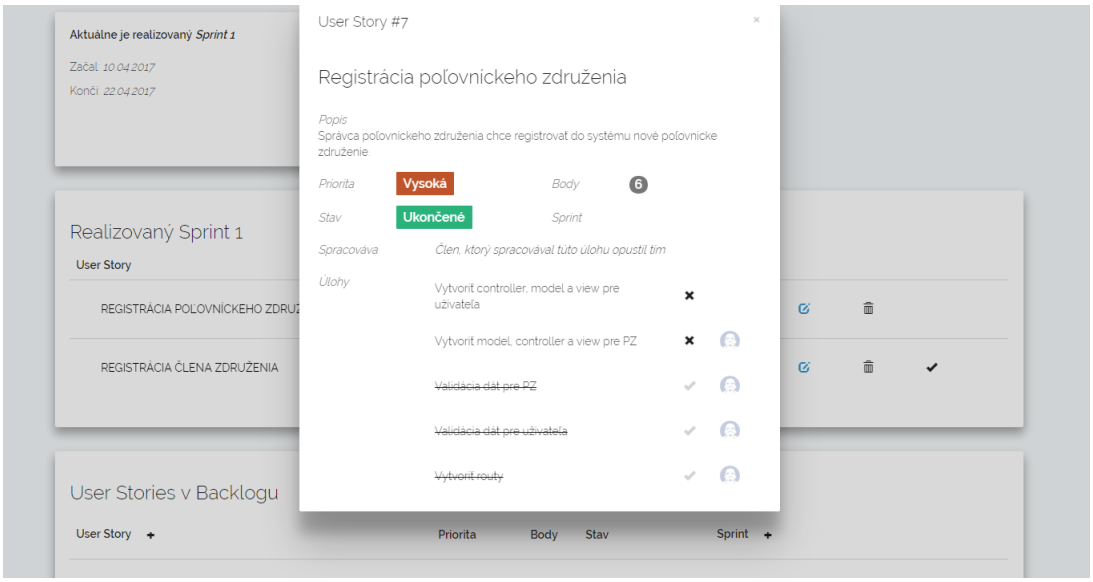
Registrácia pre Scrum Mastera je zvolená ako osobitná sekcia. Okrem vytvárania vlastného účtu je zodpovedný za registráciu samotnej firmy. Product Owner a člen tímu sa registruje ako klasický užívateľ so zadaním tímu, ku ktorému patrí. Pri vstupe na úvodnú stránku dochádza k nasmerovaniu na príslušnú registráciu na základe potreby vytvorenia nového tímu alebo pridania sa do existujúceho tímu. Po úspešnom prihlásení do systému

[2]Dostupné na: http://fezvrasta.github.io/bootstrap-material-design/

je zobrazený prehľad aktuálnych projektov, na ktorých užívateľ pracuje a prehľad v minulosti realizovaných projektov, ktorých bol súčasťou. V prípade, že do systému vstúpi užívateľ s administrátorským oprávnením, tak dostáva prístup k funkcionalitám pre editáciu a pridanie projektu realizovaných pomocou otvorenia modálneho okna. Každý projekt je zobrazený ako karta, ktorá užívateľa presmeruje na detail konkrétneho projektu.

Detail projektu poskytuje prehľad základných informácií ako je názov projektu, názov firmy zadávateľa, či podrobnejší popis projektu. Realizačný tím je pre každý jeden projekt podstatný, pričom jeho zloženie sa môže pri jednotlivých projektoch meniť. Zmena Scrum Mastera a Product Ownera si vyžaduje editáciu samotného projektu. Pre zmenu zloženia vývojárskeho tímu je Scrum Masterovi umožnené pridanie nového člena na základe výberu z ešte nepridaných zamestnancov. Poskytnutá je aj funkcionalita pre odstránenie člena z tímu a prenechanie všetkých jeho nedokončených úloh. V spodnej časti sú zobrazené grafy, ktoré umožňujú každému užívateľovi identifikovať prípadné problémy v behu aktuálneho Sprintu alebo v rámci celého projektu. Jedná sa o velocity a burndown graf, ktorých spôsob generovania bol popísaný v podkapitole 6.3. Celkové prevedenie a rozloženie prvkov odpovedá pôvodnému návrhu.

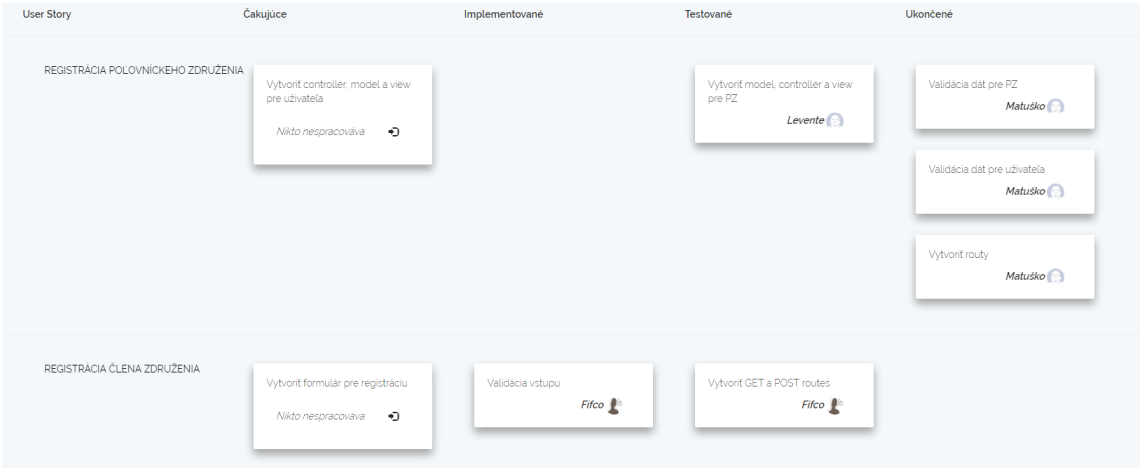
Plánovaný backlog je zobrazený v osobitnej sekcii. K jeho zmenám je oprávnený Product Owner, ale aj Scrum Master. Môžu pridávať, editovať alebo odoberať User Story, či vytvoriť nový Sprint. Vizualizácia User Stories bola zvolená vo forme tabuliek. Tieto tabuľky oddeľujú nenaimplementované User Stories od akutálne realizovaných. Prehľadnosti majú napomáhať farebné odlíšenia vlastností ako je priorita, či stav User Story. Ak sa ešte nenachádza v stave ukončené, je možné, aby Scrum Master túto User Story schválil, čím dochádza k jej preneseniu do štatistických záznamov a odzrkadleniu v grafe. V prípade aktuálne realizovanej User Story sú zobrazení tí členovia tímu, ktorí implementujú jej úlohy. Ak je však požadovaná funkcionalita ešte v backlogu, tak sa zobrazuje Sprint, do ktorého je User Story zaradená.



Obr. 6.3: Modálne okno s podrobnosťami User Story

Súhrnné informácie o jednotlivých User Stories sú zobrazené vo forme modálneho okna viditeľného na obrázku 6.3. Obsahujú jej podrobnejší popis, teda špecifikáciu funkcionality, prioritu a náročnosť vo forme bodov. Súčasťou sú aj úlohy, ktoré spoločne smerujú k vytvoreniu požadovanej vlastnosti systému. V prípade, že niekto úlohu spracováva, zobrazí sa pri danej úlohe jeho fotografia. Ak bola úloha ukončená, tak je od iných odlíšená prešKrtnutím. Toto modálne okno je možné aj v časti samotnej tabule, či v prehľade Sprintov.

Dôležitou časťou aplikácie je samotná tabuľa. Umožňuje manipuláciu s jednotlivými úlohami jednoduchým spôsobom. Každý člen tímu môže úlohu prevziať a spracovávať ju za pomoci tlačidla. V prípade, že sa rozhodne ďalej úlohu neimplementovať môžu ju prenechať na ďalšie spracovanie. Pre prehľadnosť spracovávania jednotlivých úloh je na ich karte zobrazené meno člena tímu spolu s jeho fotkou. Zmena stavu úlohy je realizovaná jednoduchým presunom medzi stĺpcami, každý presun je následne reflektovaný v databázovom systéme. Pre možnosti presunu kariet bola využitá knižnica `dragula`[3]. Vyžadovala si definíciu základných stĺpcov, do ktorých sa môže karta presúvať. Následné spustenie skriptu umožňuje určenie prvku a jeho parametrov. Za tieto parametre sa považuje identifikačné číslo úlohy, identifikačné číslo User Story, ku ktorej prináleží a stĺpec, do ktorého bola premiestnená. Funkcia v TaskControllery je následne zodpovedná za vyhľadanie príslušnej úlohy v databáze a úpravu jej stavu.



Obr. 6.4: Scrum tabuľa s úlohami rozdelenými podľa stavu

Pre uľahčenie práce s nástrojom a zabránenie vykonania nechcenej akcie bez upovedomenia užívateľa, sú zobrazované upozornenia pri vykonaní určitej akcie. Na základe príslušného úkonu je zobrazená hláška s určitým podfarbením, ktorá popisuje zmenu, ktorá bola vykonaná. V prípade, že sa jedná o pridanie, či editáciu, tak je užívateľ upozornený hláškou so zeleným zafarbením, predpokladateľne teda vykonal operáciu, ktorú mienil. Pri zmazaní prvku dochádza k upozorneniu výraznejším sfarbením, pretože sa predpokladá, že v prípade používania mobilného zariadenia alebo tabletu môže dôjsť k nechcenému stlačeniu tlačidla a vyvolaniu operácie, ktorá zmení zloženie tímu, či obsah backlogu.

[3]Dostupné na: https://github.com/bevacqua/dragula

6.5 Užívateľské testovanie

Testovacia firma

Aplikácia bola podrobená celkovo dvom fázam skúmania užívateľskej prívetivosti a správnosti fungovania. Obe tieto fázy prebehli vo firme orientovanej na softvérový vývoj najmä v oblasti webových riešení. Firma má celkovo 8 zamestnancov, avšak testovanie prebiehalo vrámci jedného tímu zloženého zo 4 ľudí. Dvaja z týchto členov zastávali čisto vývojársku pozíciu, zvyšné dve osoby vykonávali práce na vedení projektu a komunikácií s klientom. Táto firma pre vedenie svojich projektov aktuálne využíva nástroj JIRA, ktorý je popísaný v kapitole 4. Tím nemá zavedený presný proces fungovania od uzatvorenia zmluvy s klientom až po odovzdanie hotového projektu. Aktuálny nástroj je využívaný len ako skladisko úloh, ktoré majú vývojári splniť a implementovať. Súčasne je využívaný aj ako systém pre správu zamestnaneckej výkonnosti, čo umožňuje vedúcemu stanoviť mzdu zamestnancov. Vo firme prebehlo ešte pred samotnými fázami testovania sedenie, ktorého účelom bolo zaškoliť členov vývojového tímu do praktík agilnej metodiky Scrum. Výstupom bola napríklad tabuľa popisovaná v kapitole 4. Toto sedenie poskytlo členom tímu prehľad o spôsobe rozdelenia projektu na menšie časti a o náležitostiach, ktoré by mali praktikovať pre úspešnú realizáciu projektu pomocou metodiky Scrum.

Konzultácia a testovanie rozhrania

Ako bolo spomenuté cieľom nástroja je pomáhať najmä tímom, ktoré sa zameriavajú na agilnú metodiku Scrum, snažia sa o jej nasadenie bez predošlej skúsenosti s touto metodikou a hľadajú vhodný prostriedok na vizualizáciu celkového behu ich projektov, Preto bol vytvorený návrh konzultovaný priamo s členmi tímu. Jednalo sa o konzultáciu v oblasti požiadaviek, ktoré na nástroj kladú. Majiteľ spoločnosti uviedol, že jednou z dôležitých častí systému je pre neho hodnotenie výkonnosti zamestnancov a možnosť celkového prehľadu prác nie len po projektoch, ale aj po mesiacoch. Dôvodom potreby prezerania výkonnosti pracovníkov na jednotlivých projektoch bola nutnosť hodnotenia výslednej ziskovosti projektu, pričom časové vyťaženie mu umožnilo následne rýchle určiť náklad na zamestnancov. Súčasná možnosť identifikovať celkovú výkonnosť zamestnancov v rôznych mesiacoch roka umožnila hodnotenie ich práce a k nim príslušných mzdových odmien. Z tohto dôvodu bol pôvodne zamýšľaný jednoduchý štatistický prehľad o celkovom množstve vykonaných prác na určitom projekte nahradený podrobnejším popisom o množstvách prác jednotlivých zamestnancov.

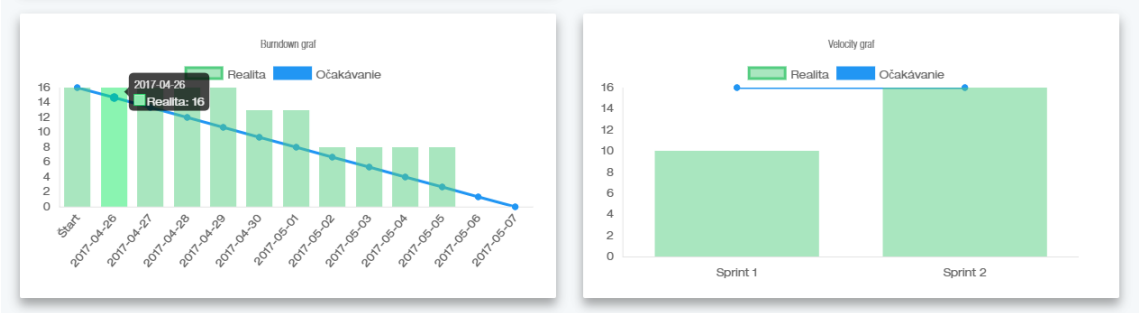
Členovia tímu v tejto fáze deklarovali, že nie je pre nich problematické s nástrojom pracovať, ale očakávali by od neho možnosť približného určenia mesačného zárobku tak, aby sa nemuseli obracať na vedúceho firmy. Z tohto dôvodu bolo do samotného profilu užívateľa zaradené aj poskytnutie informácií o ich pracovnej výkonnosti v rámci celého roka, prípadne mesiacov príslušného roka. Z dôvodu, že projekty typicky netrvajú len jeden mesiac a typickou črtou tejto spoločnosti bolo naberanie členov tímu len na určité projekty, bolo umožnené prezeranie výkonu po jednotlivých projektoch. Súčasťou tohto prehľadu je grafické znázornenie odvedenej práce po mesiacoch so súčasným doplnením textového popisu o množstve User Stories, na ktorých sa zamestnanec podieľal v porovnaní so všetkými naplánovanými User Stories pre daný projekt.

V rámci tejto etapy bola vytváraná funkcionalita postupne testovaná, pričom bola u užívateľov pozorovaná schopnosť intuitívnej práce so systémom. Akákoľvek práca so systémom im nebola vopred vysvetľovaná a teda pracovali s nástrojom ako nezávislý užívatelia. Tento

proces odhalil napríklad aj potrebu pridania väčšieho množstva užívateľských upozornení pri vykonaní určitých operácií, či upozornenie na prázdnosť backlogu a neexistenciu Sprintov. V prípade, že užívateľ chcel vytvoriť novú User Story a chcel ju zaradiť do Sprintu, tak až v editačnom okne zistil, že žiaden Sprint nemá naplánovaný. Bol následne nútený toto okno zavrieť, vytvoriť si nový Sprint a navrátiť sa naspäť k vytváraniu User Story. Tento proces bol obídený osobitnou kartou s upozornením na prázdnosť backlogu, či nulový počet naplánovaných Sprintov.

Testovanie na projekte

Za testovací projekt bol zvolený projekt nízkej náročnosti zameraný na vytvorenie webovej aplikácie pre zadávateľa. Tento projekt bol časovo málo náročný a išiel rozložiť celkovo do troch Sprintov. Realizácia projektu bola spustená 08.04.2017 s predpokladaným ukončením projektu 20.05.2017. Po dobu testovania a sledovania boli realizované celkovo dva Sprinty, jeden spustený dňa 10.04.2017 a druhý 26.04.2017 oba s časovým rozsahom 12 dní. Účelom tejto fázy bolo odskúšať funkcionalitu vytvoreného nástroja ako komplexného celku, kedy užívatelia využili všetky jeho časti. Do systému boli zanášané reálne dáta, ktoré odrážali požiadavky klienta na funkcionalitu vytváraného produktu. Výkonnosť prác je možné vi-



Obr. 6.5: Burndown graf Sprintu 2 a velocity graf pre vzorový projekt

dieť na obrázku 6.5, kde je uvedený burndown graf pre druhý Sprint a velocity graf pre celý projekt. Je možné vidieť že výkonnosť práce je v prvom Sprinte výrazne nižšia. Bola spojená najmä s potrebou adaptácie členov na nový programovací jazyk a s naštudovaním nových technológií. V druhom Sprinte bola zaznamenaná vyššia výkonnosť, ktorá naplnila očakávanú rýchlosť spracovania. Na burndown grafe možno identifikovať, že došlo k miernemu odklonu od očakávanej priamky spaľovania. Príčina tohto javu bola spojená s pomerne malou náročnosťou projektu a teda aj s malým množstvom naplánovaných User Stories. Tieto User Stories boli rozdelené do podúloh, ktoré boli spracovávané denne. K ich zaznamenaniu došlo však až pri schválení User Story v príslušných dňoch poklesu grafu.

Celkovo bolo v priebehu tohto testovania odhalených niekoľko drobných chýb spojených predovšetkým so získavanými dátami, ktoré tvorili podklad pre generovanie grafov. Jednalo sa o nesprávne odčítavanie hodnôt vykonaných User Stories v burndown grafe, či nezanášanie hodnôt do grafu v dňoch, kedy nebola vykonaná žiadna práca. Testovacia skupina rovnako udala, že do velocity grafu by bolo vhodné zahrnúť aj aktuálne bežiaci Sprint. Tieto zmeny a opravy boli zanesené do výsledného nástroja.

V priebehu realizácie prvého Sprintu bola identifikovaná potreba vykonávania prác a realizácie implementácie funkcionalít aj Scrum Masterom a Product Ownerom. Pôvodne za-

mýšľaný proces možnosti manipulácie s úlohami, ich preberanie do zodpovednosti a správa stavu len pre členov vývojového tímu, bola doplnená už v priebehu tohto Sprintu o možnosť manipulácie aj v prípade, že sa jednalo o pozície Scrum Mastera a Product Ownera. Návrh aplikácie bol zostavený tak, že za celkový beh projektu a definíciu úloh pre jednotlivé User Stories bol zodpovedný Scrum Master. Pri realizácií projektu však členovia tímu udali, že aj oni sami potrebujú definovať úlohy. Tieto úlohy predstavovali skôr pomocné práce, ktoré im umožňovali ešte viac rozdeliť funkcionalitu do zložiek, ktoré bolo potrebné implementovať.

Keďže členovia tímu často pracovali aj z domu, bolo potrebné zaznačenie zistených problémov, či riešení ku jednotlivým úlohám. Pôvodný návrh bol teda doplnený o možnosť konverzácie ku jednotlivým úlohám vo forme jednoduchých komentárov. Tieto komentáre sú oprávnení pridávať všetci užívatelia systému v danej firme, ktorí aktuálne na tomto projekte pracujú.

Celkový dojem

Ako bolo spomenuté, aplikáciu testoval tím, ktorý sa skladá zo štyroch ľudí. Preto som sa rozhodla, že nie je potrebné vytvárať online dotazník a s každým členom tímu som sa osobne stretla a opýtala som sa na jeho dojmy z používania aplikácie. Medzi otázky, ktoré som týmto ľudom kládla patria:

- *Používa sa aplikácia intuitívne?*
- *Ohodnoťte vzhľad aplikácie ako v škole.*
- *Je niečo, čo vám v aplikácii chýba?*
- *Pristupovali ste k aplikácii aj z mobilného telefónu?*
- *Aký je váš celkový dojem z aplikácie?*

Traja opýtaní odpovedali, že aplikáciu považujú za intuitívnu a jednoducho sa ovláda. Štvrtý člen tvrdí, že z počiatku mal s orientáciou v aplikácii problém, ale po prečítaní návodu dostupného po prihlásení sa v aplikácii dokázal pohybovať bez väčších problémov. Zaujímavým faktom je, že všetci členovia tímu ohodnotili vzhľad aplikácie známkou B. Po nadväzujúcej otázke *„Ako by ste vzhľad vylepšili?“*, som však zistila, že sa odpovede líšia. Jednému členovi tímu vadí farba, inému zase tiene alebo vybrané ikony. V princípe ale žiaden člen tímu nemal výhrady voči rozloženiu prvkov. Z mobilného zariadenia pristupoval k aplikácii iba Scrum Master a to vtedy, keď potreboval skontrolovať stav Sprintu počas služobnej cesty. Scrum Master a majiteľ firmy v jednej osobe súhlasil s využívaním aplikácie aj po skončení oficiálneho testovania, teda po skončení druhého Sprintu v prípade, že bude do aplikácie doimplementovaný prehľad vykonaných aktivít a upozornenia na zmeny emailom. Člen tímu, ktorý pracoval ako Product Owner pri otázke na celkový dojem z aplikácie vyzdvihol proaktivitu pri implementácii chýbajúcej funkcionality, teda napríklad implementácia možnosti pridávania komentárov k jednotlivým úlohám.

6.6 Budúci vývoj

Na základe fázy testovania a analýzy celkového behu projektu v prostredí reálnej firmy vnímam vytvorený nástroj ako jednoduchú formu vizualizácie behu projektov a ich štatistických náležitostí. Tento nástoj je možné ďalej rozvíjať a dopĺňať o nové funkcionality.

Ďalšie funkcionality

- **Pridanie prehľadu aktivít** - umožní členom tímu, ale najmä Scrum Mastrovi vidieť posledné zmeny a pokroky na projektoch
- **E-mailové upozornenia** - určené napríklad pre Scrum Mastera v prípade potreby analýzy správnosti implementácie a schválenia User Story, ktorej úlohy boli implementované
- **Rozsiahlejšie štatistiky** - štatistické súbory v rozmedzí viacerých rokov s možnosťou detailného pohľadu pre určitý rok, či kvartálne obdobie
- **Hodinové sadzby** - možnosť pridania hodinových ohodnotení jednotlivých zamestnancov, či hodinovej sazby firmy za výkon určitej práce umožňujúce okamžité generovanie príjmov a nákladov
- **Meranie času** - rozšírenie, ktoré umožňuje sledovanie výsledného času stráveného na realizáciách jednotlivých úloh, porovnávanie odhadu a reality, či určenie výslednej mzdy v prípade ohodnocovania zamestnancov na základe odpracovaných hodín
- **Pridávanie dokumentov** - ukladanie dokumentov súvisiacich s projektom (zápisy z porád, špecifikácie produktu, spätné väzby klienta)
- **Plánovanie stretnutí** - zahrnutie kalendára spolu s upozorneniami na plánované stretnutie s klientom, či projektovým tímom

Na základe kladnej odozvy od spoločnosti, ktorá aplikáciu testovala som sa rozhodla, že by som na aplikácii chcela ďalej pracovať a rozširovať ju. Mojím cieľom je dostať aplikáciu do takého stavu, že ju bude možné distribuovať medzi ďalšie spoločnosti, ktoré majú záujem o zavedenie metodiky Scrum v jej tímoch.

Kapitola 7

Záver

Hlavným cieľom tejto práce bolo vytvorenie nástroja, ktorý napomáha tímom a firmám pri koordinácií ich projektov s orientáciou na metodiky Scrum. Snažila som sa zamerať na pomoc tímom, ktoré s celkovým procesom tejto agilnej metodiky nemajú predošlú skúsenosť. V prvej fáze bolo potrebné naštudovať jednotlivé praktiky metodiky Scrum a celkový proces prechodu od vízie až ku hotovému produktu. Súčasne došlo k naštudovaniu moderných webových technológií a zvoleniu výsledného frameworku Laravel. Jednotlivé naštudované poznatky som podložila diskusiami a konzultáciami v prostredí dvoch firiem s odlišným prístupom k vývoju softvérových projektov. Dôležitým krokom bolo aj naštudovanie existujúcich riešení, ktoré sú voľne dostupné. Došlo k ich otestovaniu na mnou zvolenom vzorovom projekte, kedy bola hodnotená intuitívnosť práce s týmto nástrojom, či jej cielenie priamo na metodiku Scrum.

Proces návrhu zohľadňoval potreby reálnych firiem. Snažila som sa o začlenenie všetkých funkcií ktoré sú denne v prostredí firiem využívané pri realizácii projektov. Tieto funkcie systému boli aktívne konzultované s firmou, ktorá následne nástroj testovala. Prostredie bolo vytvárané tak, aby zobrazované informácie boli prehľadné a ľahko pochopiteľné. Dôležitým prvkom bolo zabezpečenie responzivity nástroja z dôvodu predpokladu využívania nástroja na mobilnom zariadení. Súčasťou návrhu boli aj prvky, ktoré napomáhajú užívateľom, ktorí nemajú predchádzajúcu skúsenosť s metodikou Scrum. Úlohou týchto prvkov je navedenie užívateľa na správny proces a pomoc pri osvojení tejto agilnej metodiky.

Navrhovaný nástroj bol úspešne implementovaný, pričom výsledok spĺňa stanovený cieľ. Nástroj poskytuje komplexné množstvo funkcií pre podporu vedenia projektu a plánovanie jeho fáz. Zahŕňa aj časti, ktoré slúžia ako podklady pre hodnotenie úspešnosti projektu alebo jeho časti, či hodnotenie výkonnosti tímu a jeho členov. Postupným procesom dochádzalo k dopĺňaniu funkcionalít, ktoré boli dôležité pre vedenie projektu. Aktuálne tento nástroj vnímam ako platformu, ktorá obsahuje všetky nevyhnutné prvky pre úspešnú realizáciu projektu po stránke projektového manažmentu. V budúcom postupe plánujem aplikáciu doplniť o ďalšie prvky ako sú napríklad prehľady aktivít, či meranie a zaznamenávanie odpracovaného času na jednotlivých implementovaných častiach.

Testovanie prebiehalo na reálnom projekte. Odozva na vytvorený nástroj bola lepšia než som pôvodne očakávala. Bolo odhalených pár drobných chýb, ktoré boli spojené najmä s funkcionalitami pre štatistické potreby. Po dobu testovania bol nástroj schopný naviesť užívateľov na proces metodiky Scrum bez väčších problémov.

Výsledná aplikácia je dostupná na webovej adrese scrum.osterova.sk.

Literatúra

[1] *Architecture of Laravel Applications*. [Online; navštívené: 08.05.2017]. URL http://laravelbook.com/laravel-architecture

[2] BECK, K.; aj.: *Manifesto for Agile Software Development*. [Online; navštívené: 20.04.2017]. URL http://agilemanifesto.org/

[3] KNIBERG, H.: *Scrum and xp from the trenches: how we do scrum* . S.I.: C4Media Inc., 2007, ISBN 1430322640, 978430322641, [Online; navštívené 02.05.2017]. Dostupné z: wwwis.win.tue.nl/2R690/doc/ScrumAndXpFromTheTrenchesonline07-31.pdf.

[4] *Laravel 5.4 documentation*. [Online; navštívené: 08.05.2017]. URL https://laravel.com/docs/5.4

[5] LEFFINGWELL, D.: *Scaling Software Agility: best Practices for large enterprises*. Agile Software Development Series, Upper Saddle River, NJ: Addison-Wesley, 2007, ISBN 0321458192.

[6] MCCLURE, R.: *Nato software engineering coference 1968*. [Online; navštívené: 29.04.2017]. URL http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

[7] MYSLÍN, J.: *Scrum: průvodce agilním vývojem softwaru*. Brno: Computer Press, 2016, ISBN 9788025146507.

[8] PETERSEN, K.; WOHLIN, C.; BACA, D.: *The waterfall model in large-scale development. In: 10th International Conference on Product-Focused Software Process Improvement*. Oulu: Springer, 2009, ISBN 9783642021510, [Online; navštívené 01.05.2017]. Dostupné z: http://bth.diva-portal.org/smash/get/diva2:835760/FULLTEXT01.pdf.

[9] STELLMAN, A.; GREENE, J.: *Learning agile*. Sebastopol, CA: O'Reilly Media, 2014, ISBN 9781449363840.

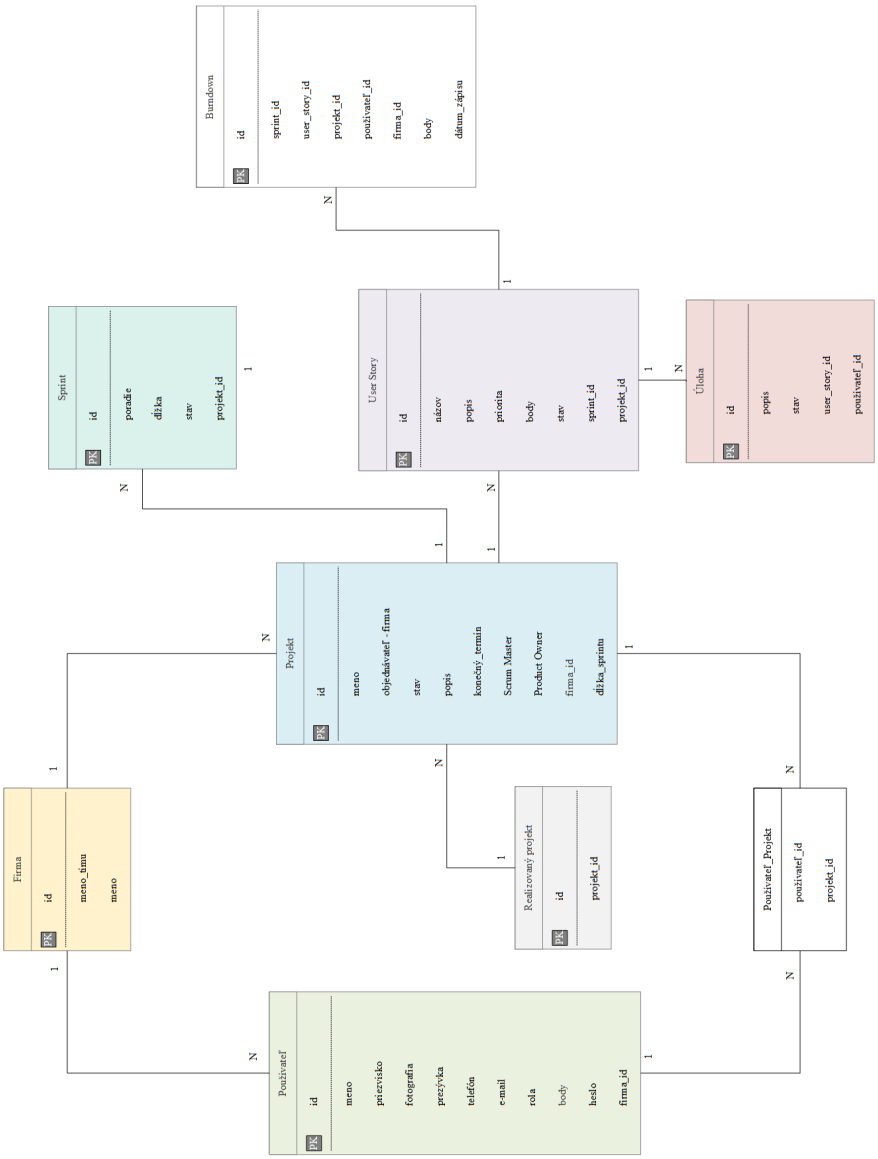
[10] ŠOCHOVÁ, Z.: *Burndown grafy*. [Online; navštívené: 28.04.2017]. URL http://soch.cz/blog/management/agile/scrum-management/burndown-grafy/

[11] ŠOCHOVÁ, Z.; KUNCE, E.: *Agilní metody řízení projektů*. Brno: Computer Press, 2014, ISBN 9788025141946.

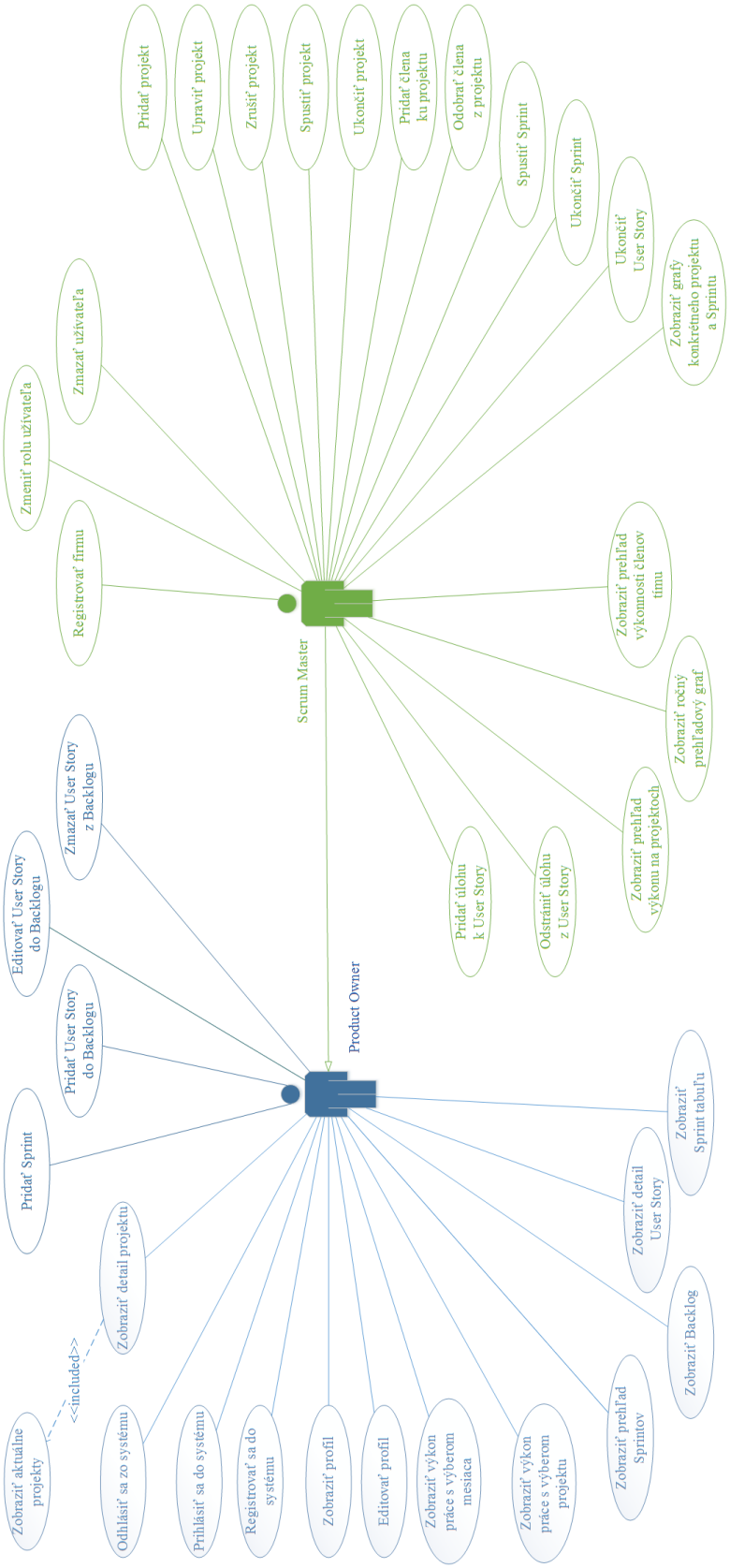
Prílohy

Príloha A

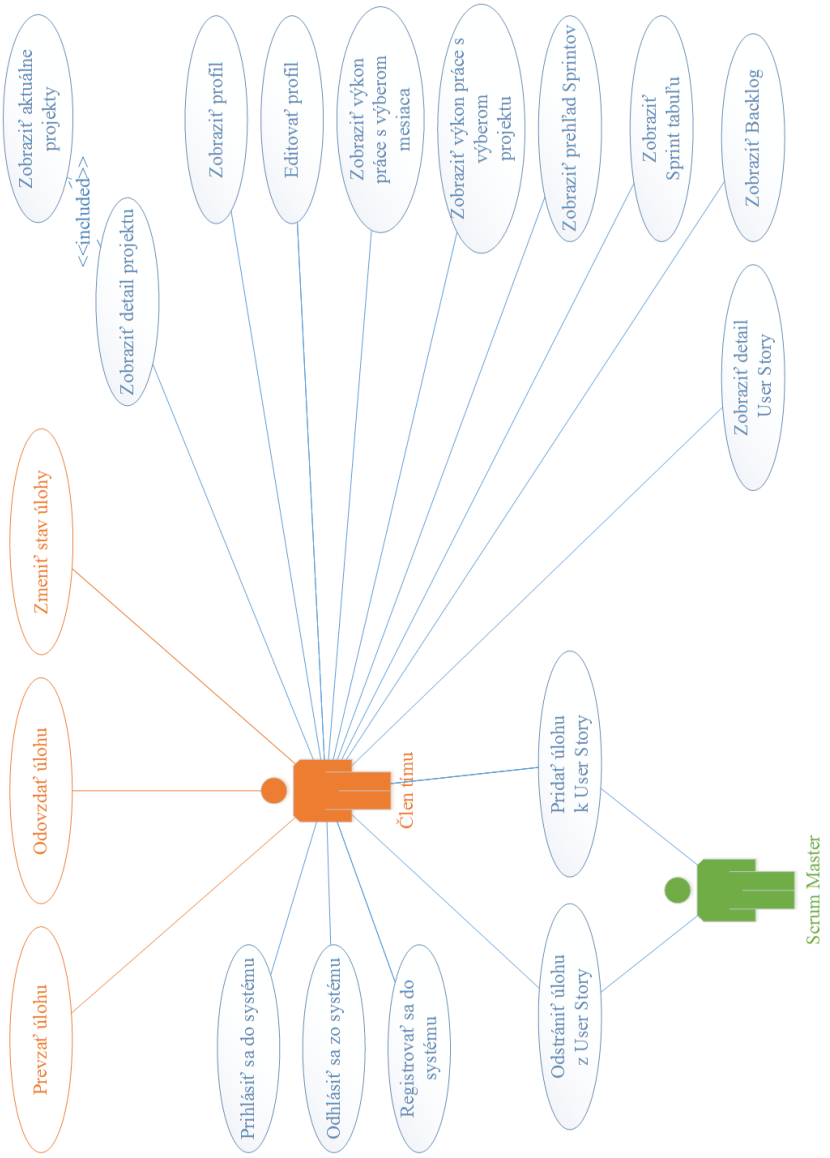
Diagram a schéma



Obr. A.1: ER diagram zobrazujúci štruktúru databázy



Obr. A.2: Use Case diagram pre role Product Owner a Scrum Master



Obr. A.3: Use Case diagram pre role člen tímu a Scrum Master

Príloha B

Obsah CD

- Zložka `documents`
  - Dokumentácia kódu
  - Technická správa
- Zložka `src`
  - Zdrojové súbory aplikácie
- Zložka `srcTZ`
  - Zdrojové súbory technickej správy