České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Michal Chvála**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Kanban board pro Gitlab**

Pokyny pro vypracování:

Seznamte se s API [2], které poskytuje Gitlab pro práci s úkoly (issues) a s principy organizace práce pomocí metodiky Kanban [3]. Následně analyzujte, navrhněte a implementujte aplikaci, která umožní využívat principy Kanban pro úkoly uložené v systému Gitlab. Implementaci proveďte pomocí frameworku AngularJs [1]. Následně implementaci otestujte pomocí vhodných testů (minimálně pomocí Selenium testů).

Seznam odborné literatury:

Ken Schwaber and Mike Beedle. 2001. Agile Software Development with Scrum (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

Vedoucí: Ing. Ondřej Macek, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

[redacted]
doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

L.S.

[redacted]
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 4. 2015

České vysoké učení technické v Praze

Fakulta Elektrotechnická

Katedra Počítačů

Bakalářská práce

Kanban Board pro Gitlab

Michal Chvála

Květen 2015

Vedoucí práce: Ing. Ondřej Macek, Ph.D.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 11. 5. 2015

Poděkování

Tímto bych velice rád poděkoval vedoucímu práce Ing. Ondřeji Mackovi, Ph. D. za poskytnuté konzultace a pozitivní přístup.

Abstrakt

Tato práce se zabývá tvorbou webové aplikace pro správu a řízení vývoje softwarových projektů, propojenou s issue tracking systémem aplikace Gitlab. Aplikace využívá koncept Kanban, který pomáhá optimalizovat vývojové procesy.

Abstract

This project realizes web application for management and control of software projects. Application is synchronized with issue tracking system of application Gitlab. Application uses the concept of Kanban, which helps to optimize development processes.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Správa a řízení vývojových procesů jsou důležité součásti vytváření software. Existuje mnoho technik a nástrojů usnadňujících tyto činnosti. Jedním z takových nástrojů je open source webová aplikace Gitlab, která je využívána i na této fakultě a je přístupná studentům a zaměstnancům[1]. Cílem práce je vytvořit aplikaci, která bude vycházet z funkcionality Gitlabu a rozšiřovat ji o další prvky. Rozšiřující funkcionalita bude inspirována technikou Kanban, jejíž cílem je optimalizovat vývojové procesy. Hlavní komponentou bude kanban board, který umožňuje vizualizovat tok práce a zvýšit efektivitu týmu.

Funkcionalita vyházející z Gitlabu bude s Gitlabem synchronizována, takže k jednotlivým projektům půjde přistupovat jak z nové aplikace, tak z Gitlabu. To umožní snadný přechod k nové aplikaci a to třeba jen části týmu. Například osoba, která bude pouze přidávat issues, nebude potřebovat účet v nové aplikaci.

Ve vývoji software je velice důležitá komunikace se zákazníkem. Správné pochopení a specifikování požadavků je základem pro vytvoření produktu, se kterým bude zákazník spokojen. Osobní schůzky se zákazníkem mohou být časově náročné a většinou u nich nemůže být přítomen celý vývojový tým. Nepříjemnosti můžou způsobovat i změny požadavků v průběhu projektu. Proto bude aplikace obsahovat část pro správu uživatelských požadavků, která bude umožňovat i zapojení zákazníka. Zákazník bude moci vložit do aplikace svůj požadavek a případně o něm diskutovat s vývojovým týmem. Sníží se tím potřeba osobních setkání a do diskuze se bude moci zapojit celý tým.

[1] `https://gitlab.fel.cvut.cz/users/sign_in`

2 Správa a řízení vývoje software

Cílem správy a řízení je zajistit, aby vývoj probíhal co nejefektivněji a bylo dosaženo výsledku v požadované kvalitě a v požadovaném čase. Jedním z cílů práce je vhodně zkombinovat klasický issue tracking s pokročilejšími metodami Kanban a SCRUM.

2.1 Issue Tracking System

Issue tracking system (ITS) je systém pro správu issues. Issue může být jakýkoliv problém, který je potřeba vyřešit. Například odstranění chyby nebo přidání nové funkcionality. ITS umožňuje shromažďovat issues na jednom místě a sdílet je mezi členy týmu. Každé issue by mělo obsahovat tyto údaje [6]

- co se má udělat
- co nefunguje a jak by to mělo fungovat správně, pokud se jedná o chybu
- kdo issue vytvořil a kdo je za něj zodpovědný
- kdy bylo issue vytvořeno
- jaké změny byly provedeny
- jak dlouho trvalo vyřešení issue

Správné používání ITS pomáhá zvýšit kvalitu software a spokojenost zákazníka [6]. Mezi již existující ITS patří například Assembla[2] nebo Bugzilla[3].

2.2 Kanban Board

Kanban [1] je technika pro optimalizaci procesů vycházející z konceptu just-in-time [4]. Just-in-time říká, že produkt by měl být vyroben a dodán právě tehdy, kdy je třeba, a tím minimalizovat náklady na skladování a zamezit plýtvání prostředky. Kanban je technika, která podává návod, jak tohoto cíle dosáhnout. Tento koncept se dá aplikovat i na vývoj software. D. J. Anderson ve své knize [2] definuje pět základních principů Kanbanu použitého pro vývoj software:

1. Vizualizovat tok práce

   K vizualizaci je v praxi použita tabule (Kanban Board) a kartičky, na kterých jsou napsány úkoly. Na tabuli jsou vyznačeny sloupce s názvy fází, ve kterých se můžou úkoly nacházet. Toto řešení je vidět na obrázku 1. Úkoly lze posouvat mezi sloupci podle toho, v jaké fázi se zrovna nachází. Kartičky by měly být co

[2] https://www.assembla.com/
[3] https://www.bugzilla.org/

nejnázornější. Můžeme například použít různé barvy kartiček pro různé druhy úkolů nebo dopsat na kartičky jména pracovníků, kteří se úkolem zabývají a podobně. Díky vizualizaci je okamžitě vidět, kdo na kterých úkolech právě pracuje. Také můžeme sledovat postup při řešení úkolů a snadněji odhalovat případné problémy.

2. Omezit rozdělanou práci

   Nad sloupce je možno napsat čísla, která určí maximální počet úkolů v daném sloupci, a tím omezit rozdělanou práci. Omezení rozdělané práce je jednou z klíčových součástí Kanbanu. Je to nástroj, který dokáže zamezit nadprodukci a udržet proces just-in-time. Zásadní ovšem je omezení dobře nastavit. Správným nastavením se zabývají další definované body. Omezení rozdělané práce navíc zabraňuje pracovníkům v přeskakování mezi různými úkoly, a tím pádem jsou úkoly dokončeny rychleji a v lepší kvalitě [2].

3. Měřit a optimalizovat tok práce

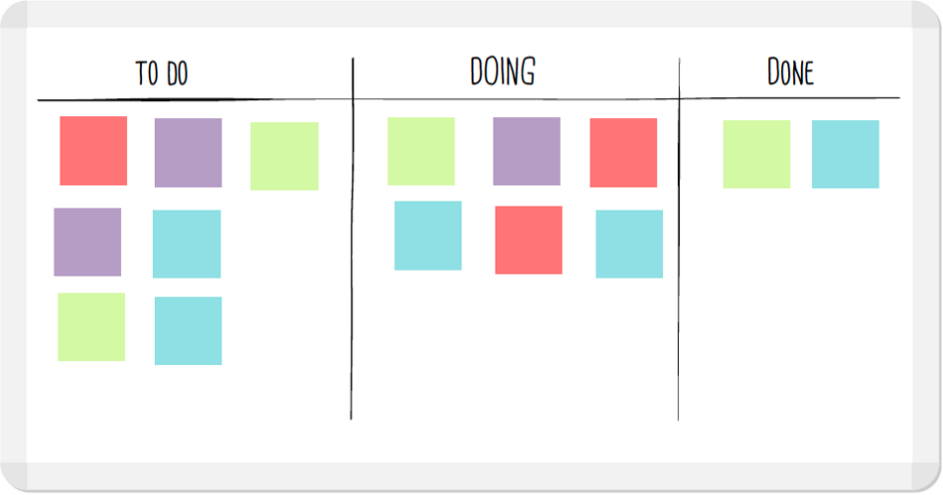
   Díky vizualizaci lze snadno pozorovat postup práce a případné problémy. Je možné vidět, kde se úkoly zasekávají nebo které sloupce jsou naopak málo vytížené a na základě těchto údajů můžeme tok optimalizovat. Jedním z úkolů optimalizace je nalezení a odstranění případných bootlenecků. Bootleneck je část procesu, která zpomaluje celkový tok práce. Podle článku [1] umožňuje kanban board objevit bootleneck. Na obrázku 2 lze vidět, že analytici a vývojáři nemohou začít pracovat na dalším úkolu, dokud testeři nějaký nedokončí. Analytici a vývojáři tedy mohou pomoci s testy a zajistit plynulejší tok práce. Pokud taková situace nastává často, může to znamenat, že fáze test je bootleneck a měly by se přehodnotit zdroje.

4. Udělat procesní politiky explicitní

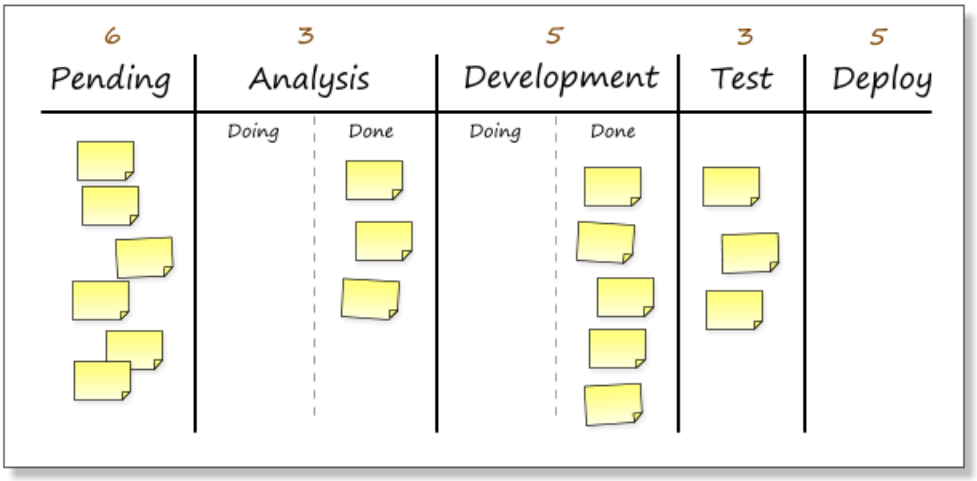
   Tento bod říká, že politiky by měly být jasně definované, snadno dohledatelné a každý člen týmu by jim měl rozumět. Jednou z důležitých politik je například Definiton of Done, tedy za jakých podmínek může být úkol uzavřen. U každého sloupce kanban boardu mohou být definovány podmínky, které je třeba splnit před přesunem úkolu do tohoto sloupce. Procesní politikou je i omezení rozdělané práce popsané v bodu 2. Je důležité, aby omezení bylo viditelné a všem jasné. Případně je možno definovat podmínky, za jakých může úkol do sloupce vstoupit, i když překročí limit.

5. Kontrolovat a vylepšovat kvalitu procesu

   Předchozí body by nebyly tak užitečné, pokud bychom nekotrolovali kvalitu procesu a nesnažili se o jeho vylepšování. Můžeme například sledovat, kolik je celkem rozdělané práce a kolik rozdělané práce připadá v průměru na jednoho člena týmu, a na základě toho upravit omezení počtu úkolů ve sloupcích na kanban boardu. Také můžeme měřit propustonst procesu, tedy celkový počet dokončené práce v určitém časovém období. Další měřenou veličinou je lead time. Lead time je čas,



Obrázek 1: Základní Kanban Board, převzato z [9]



Obrázek 2: Kanban Board - bootleneck, převzato z [10]

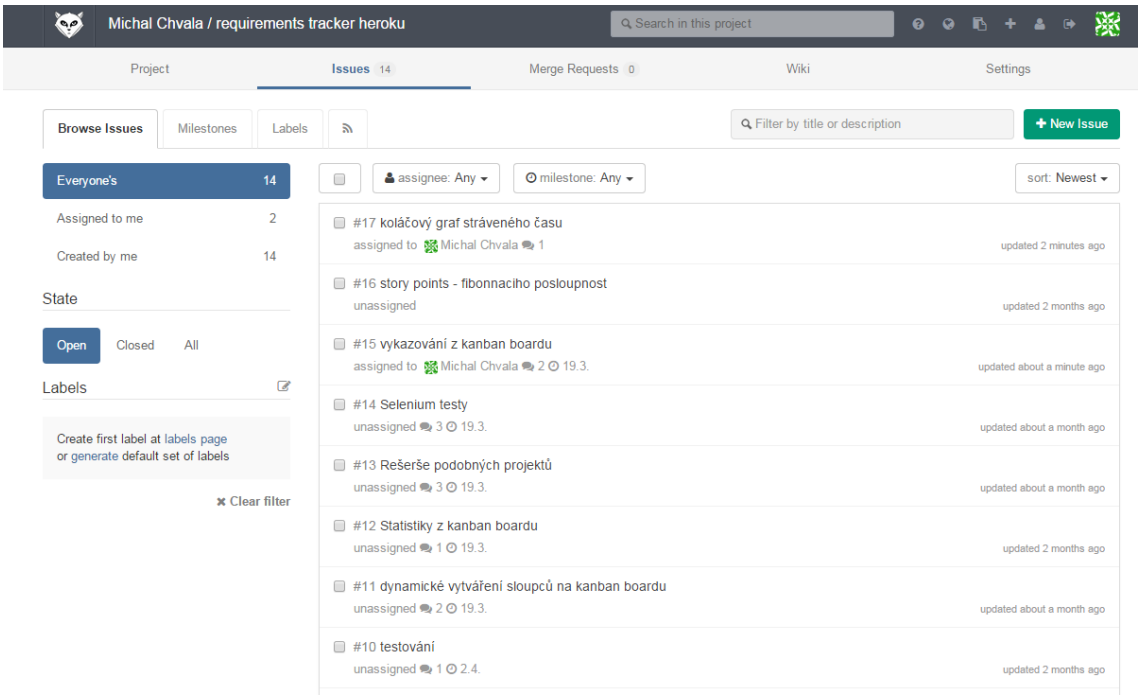
za který kartička s úkolem projde celý kanban board. Tento údaj můžeme porovnat se skutečným stráveným časem na úkolu. Rozdíl mezi těmito hodnotami by měl být ideálně co nejnižší.

Kanban Board je tedy jeden z prostředků k realizování techniky Kanban. Slouží zejména k vizualizaci, omezení rozdělané práce a pomáha při meření a zlepšování kvality procesů.

2.3 SCRUM

SCRUM[7] je agilní[4] metodika řízení projektů, vyžívaná zejmená pro vývoj software.

[4] http://agilemanifesto.org/iso/cs/



Obrázek 3: Gitlab issue tracking

Základní prvky scrumu jsou product backlog a sprint. V product backlogu jsou seřazeny uživatelské požadavky podle priority. Sprint je časové období dlouhé maximálně jeden měsíc uvnitř kterého probíhá implementace. Před zahájením se z product backlogu do sprint backlogu přesunou požadavky, které budou v rámci sprintu implementovány. Po skončení sprintu se provede zhodnocení a v případě potřeby může proběhnout další sprint.

2.4 Gitlab

Gitlab [11] je open source webová aplikace pro správu kódu obsahující také jednoduchý Issue tracking system. Umožňuje vytvářet projekty a přiřazovat do nich členy týmu pod různými uživatelskými právy. V rámci projektu je možno vytvářet issues. K vytvořenému issue lze přiřadit pracovník, štítky a stav, který může být buď open nebo closed. K issue lze také přidávat komentáře. Issues lze organizovat pomocí milestones. Každý milestone může obsahovat datum, do kdy by měl být splněn. Pomocí milestones můžeme tedy simulovat vývojové iterace. Většina funkcionality ITS je přístupná přes REST API.

3 Požadavky

Požadavek je specifikace toho, co by mělo být implementováno [8]. Požadavky určují pouze to, co by měl výsledný systém umožňovat, nikoliv jak toho bude dosaženo. Požadavky na tuto aplikaci vycházejí z kombinace metod Kanban a SCRUM, zadání práce a požadavků vedoucího práce.

3.1 Propojení s Gitlabem

REQ-1: Propojení s účtem na Gitlabu

Systém bude umožnovat propojení s existujícím účtem na Gitlabu.

REQ-2: Vytvořit issue

Systém bude umožnovat vytvoření issue. Issue bude synchronizováno s Gitlabem a to včetně štítků.

REQ-3: Přiřazení pracovníkovi

Systém bude umožňovat přiřadit issue konkrétnímu uživateli. Pokud issue nebude nikomu přiřazeno, může ho kdokoliv přiřadit sám sobě. Přiřazení bude synchronizováno Gitlabem.

REQ-4: Komentovat issue

Systém bude umožňovat komentovat jednotlivá issue. Komentáře budou synchronizovány s Gitlabem.

REQ-5: Uzavřít issue

Systém bude umožnovat uzavřít issue. Uzavření issue bude synchronizováno s Gitlabem.

3.2 Kanban

REQ-6: Backlog

Všechna issue budou po vytvoření přiřazena do backlogu.

REQ-7: Odhad pomocí story points

Systém bude umožnovat přidat k issue odhad pomocí story points.

REQ-8: Vytvoření iterace

Systém bude umožnovat vytvořit iteraci. Bude možné nastavit délku iterace.

REQ-9: Kanban Board

Systém bude obsahovat kanban board. Na kanban boardu se budou zobrazovat issues z právě probíhající iterace. Pozice na kanban boardu bude určovat stav issue.

REQ-10: Přijmutí/nepřijmutí

Systém bude umožňovat pro issues nacházejících se v sekci done možnosti přijmout nebo nepřijmout. Pokud bude zvoleno přijmout, issue se uzavře a bude považováno za splněné. Pokud se nepřijme, uživatel ho bude moci posunout na jinou pozici kanban boardu nebo vrátit do backlogu.

REQ-11: Druhy issue

Systém bude umožňovat k issue přiřadit druh. Druhy budou: feature, bug, (chore, release)

REQ-12: Automatická tvorba iterace

Systém bude umožňovat automatické vytvoření iterace podle pořadí issues v backlogu.

REQ-13: Přidání issue do iterace

Vedoucí pracovník bude moci přiřadit issue z backlogu do konkrétní iterace.

REQ-14: Statistiky z kanban boardu

Systém bude umožňovat zobrazit statistiku o pohybu issue na kanban boardu. Bude možné zobrazit v jakém časovém intervalu a na jaké pozici v kanban boardu se issue nacházelo.

REQ-15: Mapování issue na požadavek

Systém bude umožňovat přiřadít k issue konkrétní požadavek, ke kterému se issue vztahuje.

REQ-16: Přidání nového uživatele

Systém bude umožnovat osobám s právy administrátora přídání nového uživatele. Při přidávání se budou nastavovat uživatelská práva (zákazník, zaměstnanec, administrátor).

REQ-17: Smazání uživatele

Systém bude umožnovat osobám s právy administrátora smazat uživatele.

3.3 Fórum se zákazníkem

Aplikace bude umožňovat zapojení zákazníka prostřednictvím diskuzního fóra, které bude zároveň sloužit ke správě požadavků. Tato sekce byla přidána na základě požadavků

vedoucího práce.

REQ-18: Přihlášení přes Google

Systém bude umožňovat přihlášení přes Google účet. Uživatel bude přihlášen jen tehdy, pokud bude jeho e-mail zaznamenán v lokální databázi.

REQ-19: Přidání požadavku

Systém bude umožnovat přidat požadavky. Požadavek bude obsahovat název, popis, prioritu a identifikátor osoby, která požadavek vytvořila.

REQ-20: Připojení souboru k požadavku

Systém bude umožnovat přidat k požadavku obrázky, pdf sobory a rar/zip archivy.

REQ-21: Stav požadavku

Systém bude umožnovat přiřazení a změnu stavu k požadavku. Aktuální stav bude ovlivňovat barvu požadavku. Stavy budou: nový, rozpracovaný, k posouzení, přijatý, odmítnutý, smazaný.

REQ-22: Komentování požadavku

Systém bude umožnovat komentovat jednotlivé požadavky i jednotlivé komentáře. Komentáře budou mít stromovou strukturu.

REQ-23: Připojení souboru ke komentáři

Systém bude umožnovat přidat ke komentáři obrázky, pdf sobory a rar/zip archivy.

REQ-24: Stav komentáře

Systém bude umožnovat ke komentáři přiřadit stav. Stavy budou vytvořený, vyřešený a neplatný.

REQ-25: Rozdělení komentářů

Systém bude komentáře rozdělovat do dvou sloupců podle stavu komentáře. Neplatné a vyřešené komentáře budou v samostatném sloupci a bude je moci zobrazit jen vývojář a administrátor, ne zákazník.

3.4 Výkazy a statistiky

REQ-26: Výkaz hodin

Systém bude umožnovat přidat k issue počet hodin, které pracovník strávil při jeho plnění. Na jednom issue může pracovat více lidí.

REQ-27: Statistika odpracovaných hodin

Systém bude umožnovat zobrazení statistiky odpracovaných hodin. Odpracovaný čas bude možno filtrovat podle pracovníka a data.

REQ-28: Stažení výkazů

Systém bude umožňovat stažení a vytisknutí výkazu hodin.

REQ-29: Burndown chart

Systém bude zobrazovat burndownt chart pro jednotlivé iterace i pro celkový projekt.

4 Podobné projekty

Aplikace pro řízení vývoje software čerpající z techniky Kanban již existují. Cílem rešerše je zjistit, jak existující aplikace vyhovují zadaným požadavkům, případně se inspirovat zajímavou funkcionalitou.

4.1 Taiga

Taiga [12] je open source aplikace zaměřená především na scrum. Umožňuje vytvářet user stories a přiřazovat je do sprintů. Každý sprint má vlastní taskboard, kde lze na základě user stories vytvářet úkoly. Úkoly můžeme posouvat po taskboardu, a tím měnit jejich stav. Taiga obsahuje i kanban board, ale od taskboardu se liší jen tím, že zobrazuje stavy user stories, ne stavy tasků. Aktivity prováděné s user stories i tasky jsou zaznamenávány, včetně pohybu po kanban boardu. Aplikace umožňuje propojení s Gitlabem, ale ne s jeho issue trackerem. Jediná funkcionalita je změna stavu úkolu pomocí zprávy při commitu kódu.

Nedostatky

- Zatím jen v beta verzi, dochází k chybám
- Nelze propojit s issue trackerem na Gitlabu
- Nepřehledné zobrazení historie posunů po kanban boardu
- Nepodporuje vykazování času

Přednosti

- Orientace na SCRUM
- Možnost mapovat úkoly na user story
- Konfiguravatelné uživatelské role umožňující zapojení zákazníka do projektu

4.2 Pivotal Tracker

Pivotal Tracker [13] je aplikace pro agilní řízení vývoje software. Umožňuje vkládat user stories a přidávat k nim odhad pomocí story points, komentáře, pracovníka, štítky a chcecklist s podúkoly. User stories jsou řazena v backlogu. Pořadí v backlogu určuje prioritu. Iterace jsou tvořeny automaticky na základě pořadí v backlogu. Iteraci lze nastavit délka v počtu týdnů. Počet user stories které se přiřadí do iterace závisí na velocitě týmu. User stories se přiřadí do iterace tak, aby se součet jejich story points rovnal

velocitě. Na začátku projektu musíme velocitu odhadnout, pozdější se vypočítává automaticky jako průměr jedné až čtyř předchozích iterací. V probíhající iteraci můžeme postupně měnit stav user story na started, finished a delivered. Poté může být user story přijmuto nebo odmítnuto. Při přijmutí se uzavře, v případě odmítnutí zůstane v současné iteraci a může být znovu zahájeno.

Nedostatky

- Nelze propojit s issue trackerem na Gitlabu
- Nepodporuje vykazování času
- Neobsahuje uživatelskou roli pro zákazníka
- Neobsahuje Kanban Board

Přednosti

- Automatické vytváření iterace na základě vypočítané velocity

4.3 LeanKit

LeanKit [14] je placená aplikace pro řízení projektů pomocí metody Kanban. Umožňuje vytvářet kartičky s úkoly a posouvat je po kanban boardech. Kartičky s úkoly mohou obsahovat všechny potřebné informace a data jako odhadovaný začátek a konec, prioritu, druh úkolu, složitost, zodpovědné pracovníky, podúkoly, soubory a komentáře. Každá kartička má historii provedených akcí, včetně posunů po kanban boardu. U každého sloupce můžeme omezit počet kartiček, které se v něm můžou nacházet. Sloupce mohou obsahovat podsloupce a mohou být řazeny i pod sebou. Díky tomu můžeme například vizualizovat paralelní práci dvou týmů. Aplikace také obsahuje pokročilé grafické nástroje pro analýzu procesů.

Nedostatky

- Nelze propojit s issue trackerem na Gitlabu
- Nepodporuje vykazování času
- Neobsahuje uživatelskou roli pro zákazníka
- Vše se odehrává na kanban boardech, backlog můžeme vytvořit jen jako sloupec na kanban boardu, nelze vytvořit společný backlog pro více kanban boardů

Přednosti

- Výborné vizuální provedení
- Možnost omezit počet kartiček ve sloupci
- Možnost ke každému sloupci připsat procesní politiku
- Grafické nástroje pro analýzu procesů
- Velké množství připravených šablon pro Kanban Boardy
- Sloupce mohou obsahovat podsloupce a sloupce se mohou nacházet i pod sebou

4.4 Trello

Trello [15] je aplikace pro organizování práce pomocí kanban boardu. Není nijak zaměřena na vývoj sotware. Umožňuje vytvářet tabule a posouvat po ní úkoly. Kartička s úkolem může obsahovat pracovníky, štítky, komentáře, soubory, datum expirace a checklist s podúkoly.

Nedostatky

- Nelze propojit s issue trackerem na Gitlabu
- Nepodporuje vykazování času
- Neobsahuje uživatelskou roli pro zákazníka
- Neumožňuje přidat časový odhad
- Využívá jen kanban boardy, žádný centrální backlog

Přednosti

- Jednoduché a intuitivní uživatelské rozhraní
- Možnost vložit obrázek přímo na kartičku zobrazenou na Kanban Boardu

4.5 Shrnutí rešerše

Tabulka 1 ukazuje, které aplikace splňují jaké požadavky. Terminologie se v aplikacích liší. Při posuzování splnění požadavku byl brát v potaz účel a použitelnost funkcionality, ne její přesný název. Žádná z aplikací neumožňuje propojení s issue trackerem na Gitlabu. Požadavky, které jsou splněny kromě sychnronizace s Gitlabem, jsou v tabulce považovány za splněné.

Vzhledem k tomu, že žádná z nalezených aplikací nepodporuje propojení s issue trackerem na Gitlabu, je k uspokojení požadavků potřeba vytvořit novou aplikaci.

	Taiga	Trello	Pivotal Tracker	LeanKit
REQ-1:	✗	✗	✗	✗
REQ-2:	✓	✓	✓	✓
REQ-3:	✓	✓	✓	✓
REQ-4:	✓	✓	✓	✓
REQ-5:	✓	✓	✓	✓
REQ-6:	✓	✗	✓	✗
REQ-7:	✓	✗	✓	✓
REQ-8:	✓	✗	✓	✗
REQ-9:	✓	✓	✗	✓
REQ-10:	✗	✗	✓	✗
REQ-11:	✓	✓	✓	✓
REQ-12:	✗	✗	✓	✗
REQ-13:	✓	✗	✓	✗
REQ-14:	✗	✗	✗	✓
REQ-15:	✓	✗	✗	✗
REQ-16:	✓	✓	✓	✓
REQ-17:	✓	✓	✓	✓
REQ-18:	✗	✓	✓	✗
REQ-19:	✓	✗	✗	✗
REQ-20:	✓	✗	✗	✗
REQ-21:	✓	✗	✗	✗
REQ-22:	✓	✗	✗	✗
REQ-23:	✗	✗	✗	✗
REQ-24:	✗	✗	✗	✗
REQ-25:	✗	✗	✗	✗
REQ-26:	✗	✗	✗	✗
REQ-27:	✗	✗	✗	✗
REQ-28:	✗	✗	✗	✗
REQ-29:	✓	✗	✓	✗

Tabulka 1: Přehled podobných aplikací a jejich vyhovování požadavkům

5 Analýza

Cílem analýzy je co nejlépe pochopit zadaný projekt a definovat základní pojmy. Analýza neříká nic o tom, jak má být požadovaných výsledků dosaženo.

5.1 Uživatelské role

Uživatelské role popisují uživatele systému a jejich práva v aplikaci.

Zákazník Osoba, která je zadavatelem projektu. Do aplikace může vkládat požadavky diskutovat o nich s vývojáři. Zákazník nemusí mít účet na Gitlabu.

Vývojář Osoba pracující na vývoji projektu, která má účet na Gitlabu. Uživatelská práva u jednotlivých projektů závisí na právech přidělených na Gitlabu[5].

Administrátor Osoba, která má právo přidávat do aplikace nové uživatele. Musí mít účet na Gitlabu. Uživatelská práva u jednotlivých projektů závisí na právech přidělených na Gitlabu.

5.2 Doménový model

Doménový model popisuje základní entity, se kterými bude aplikace pracovat, a vztahy mezi nimy. Doménový model je zobrazen na obrázku 4.

Osoba symbolizuje uživatele aplikace. Může se jednat o vývojáře, administrátora nebo zákazníka.

Požadavek popisuje, co má vytvářený produkt splňovat. Může ho vytvářet zákazník i pracovník. Možné stavy a přechody mezi nimi jsou vyjádřeny obrázkem 5.

Projekt seskupuje informace týkající se jednoho konrétního zadání.

Issue symbolizuje úkol, který je potřeba splnit. Může se jednat například o implementaci nové funkcionality nebo o opravu chyby.

Sprint rozděluje vývoj do iterací. Jedná se o časově ohraničený úsek v rámci kterého by měly být splněny úkoly, které jsou do sprintu přiřazeny.

Backlog shromažďuje issues, které ještě nebyly přiřazeny do žádného sprintu.

[5] `https://gitlab.fel.cvut.cz/help/permissions/permissions.md` - zde je po přihlášení dostupná specifikace přístupových práv různých rolí na Gitlabu

KanbanBoard seskupuje sloupce a spojuje je s konkrétním sprintem. Představuje kanban board, který je popsán v kapitole 2.2.

Sloupec je sloupec na kanban boardu. Může obsahovat issues v určitém pořadí.

Odpracovaný čas představuje skutečný čas, který osoba strávila prací na konkrétním issue.

Gitlab představuje issue tracker aplikace Gitlab přístupný přes REST api. Pomocí tohoto api bude probíhat synchronizace.

5.3 Případy užití

Případy užití říkají, kdo a jakým způsobem bude moci systém používat. Zachycují aktéry a činnosti, které mouhou v systému vykonávat. Případ užití vždy zahajuje aktér. Jedná se o další způsob zaznamenání požadavků zadavetele. Případy užití také často slouží jako podklad pro testování. [8]

5.3.1 Fórum se zákazníkem

UC1: Vložit požadavek

Uživatel může vložit požadavek. Požadavek obsahuje název, popis a prioritu. Všechna pole jsou povinná. K požadavku může být také nepovinně připojen obrázek, pdf soubor nebo rar/zip archiv.

UC2: Komentovat požadavek

Uživatel může k požadavku přidat komentář. Komentář může odpovídat i na jiný komentář. Ke komentáři může být připojen obrázek, pdf soubor nebo rar/zip archiv.

UC3: Zobrazit požadavky

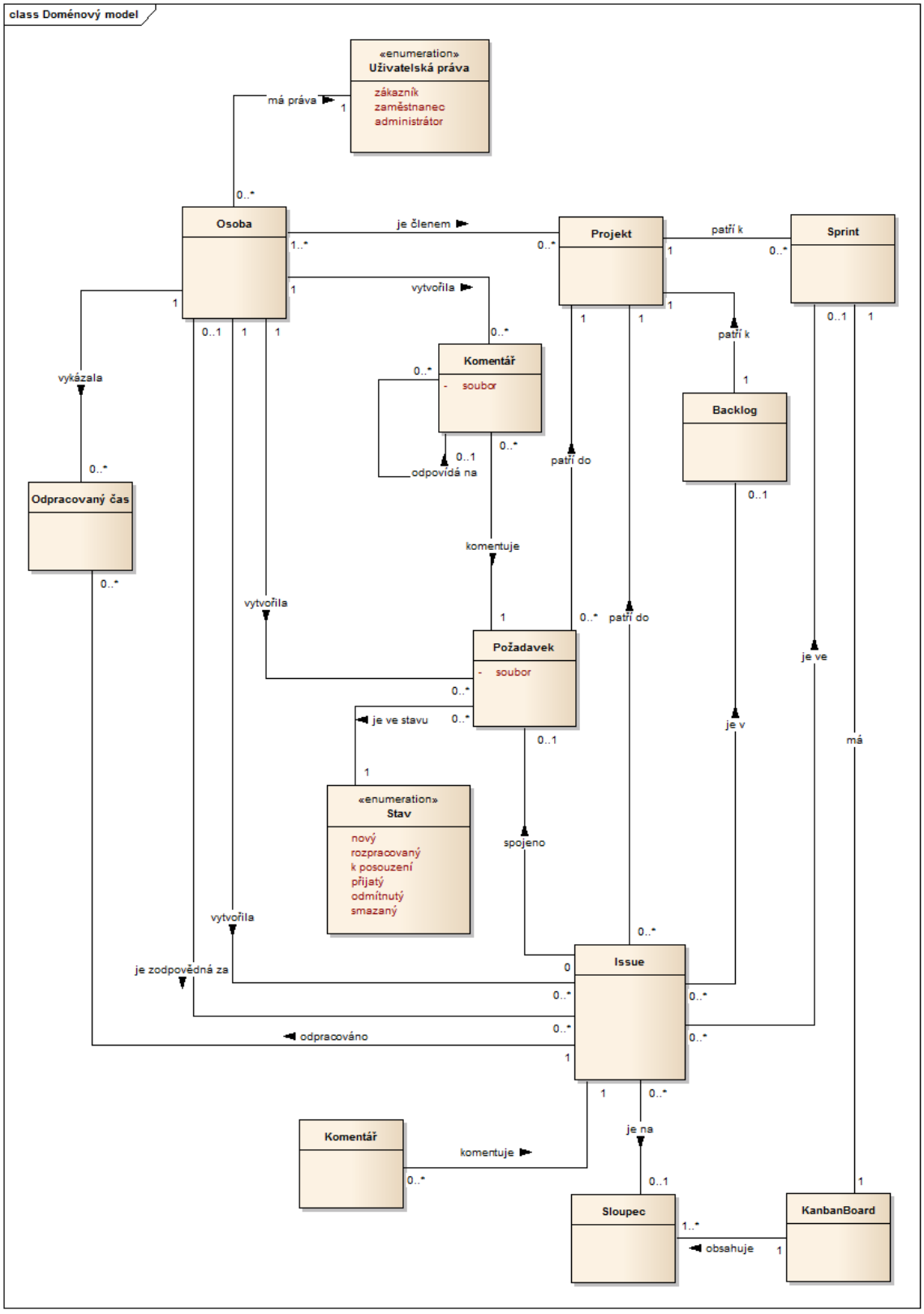
Uživatel může prohlížet náhledy požadavků seřazené podle data vytvoření. Náhled požadavku obsahuje název, stav, prioritu, datum vytvoření a jméno uživatele, který požadavek vytvořil. Požadavky jsou barevně odlišeny podle stavu, ve kterém se nachází.

UC4: Filtrovat požadavky podle stavu

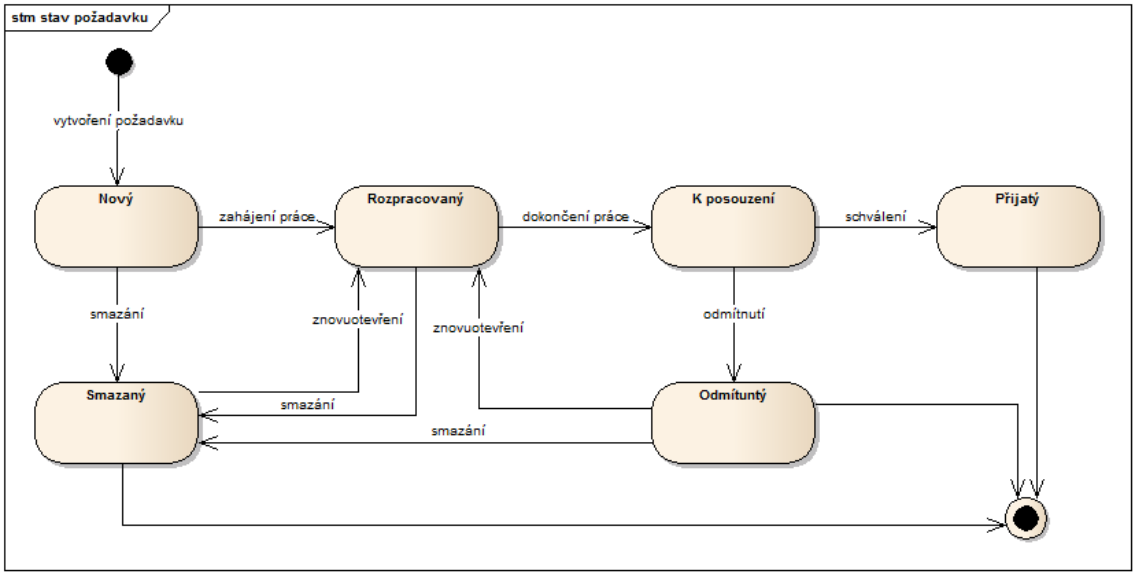
Uživatel může filtrovat zobrazené požadavky podle jejich stavu.

UC5: Zobrazit detail požadavku

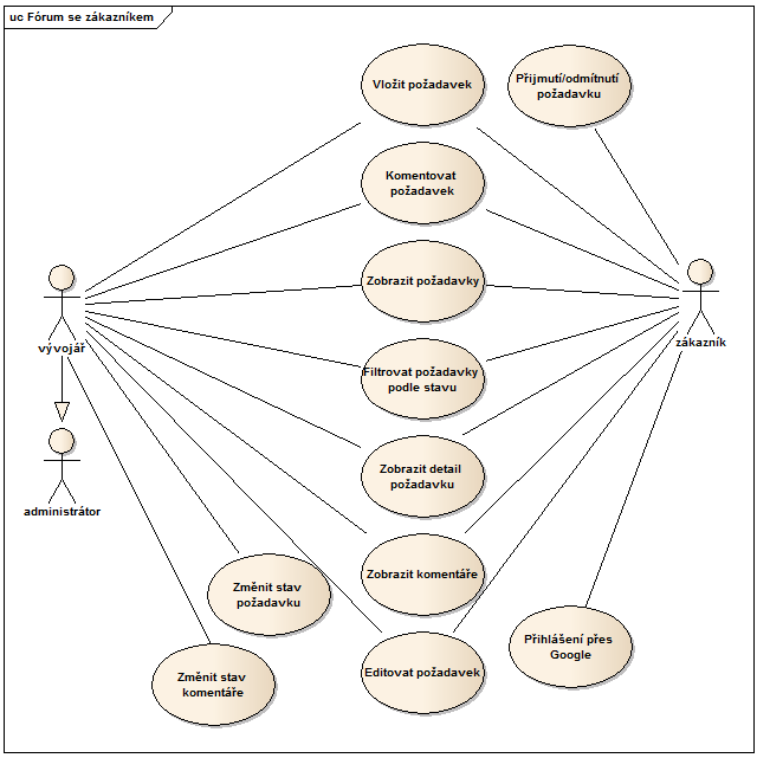
Uživatel může zobrazit detail požadavku. Detail obsahuje název, popis, prioritu, stav, datum vytvoření a jméno osoby, která požadavek vytvořila.



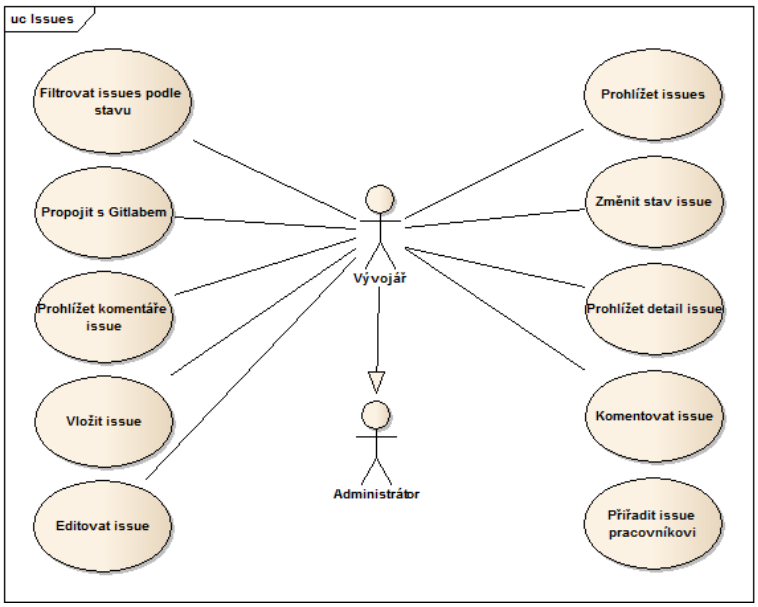
Obrázek 4: Doménový model



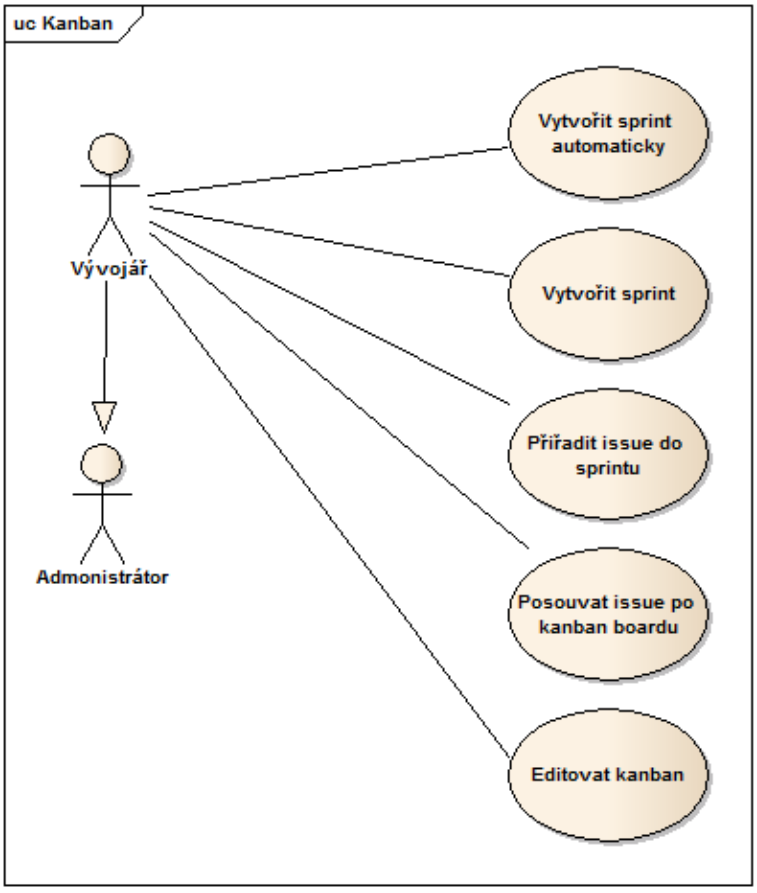
Obrázek 5: Stavový diagram požadavku



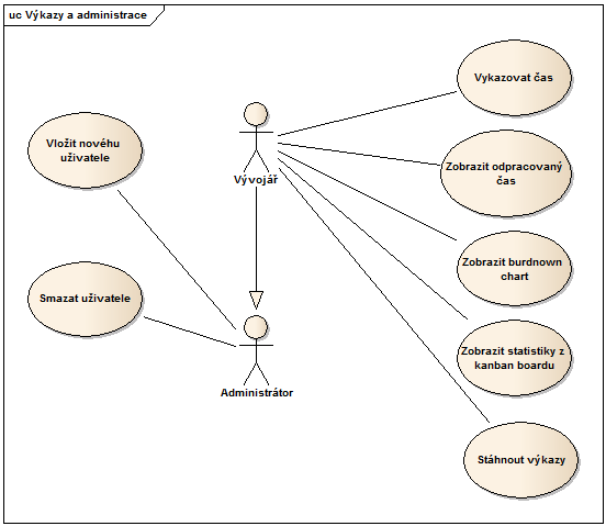
Obrázek 6: Případy užití - fórum se zákazníkeml



Obrázek 7: Případy užití - issues



Obrázek 8: Případy užití - Kanban



Obrázek 9: Případy užití - výkazy a administrace

UC6: Zobrazit komentáře

Uživatel může zobrazit komentáře. Komentáře jsou zobrazeny ve vláknech. Lze tedy vidět, jestli komentář reaguje na jiný komentář, případně na který. Komentáře se rozdělují do dvou sekcí podle stavu. V jedné sekci jsou vytvořené komentáře, ve druhé jsou vyřešené a neplatné komentáře.

UC7: Editovat požadavek

Uživatel může editovat požadavek. Lze změnit název, popis i prioritu.

UC8: Změnit stav požadavku

Uživatel může změnit stav požadavku.

UC9: Změnit stav komentáře

Uživatel může změnit stav komentáře na vyřešený nebo neplatný.

UC10: Přijmutí/odmítnutí požadavku

Pokud se požadavek nachází ve stavu dokončeno, uživatel ho může přijmout nebo odmítnout.

UC11: Přihlášení přes Google účet

Uživatel se bude moci přihlásit přes Google účet.

5.3.2 Issues

UC12: Vložit issue

Uživatel může vložit issue, které obsahuje název, popis, story points, druh, zodpovědnou osobu a požadavek, ke kterému se issue vztahuje. Název je povinný, ostatní volitelné. Vytvořené issue se vloží do backlogu.

UC13: Editovat issue

Uživatel může editovat issue. Editovat lze název, popis, story points, zodpovědnou osobu a související požadavek.

UC14: Prohlížet issues

Uživatel může prohlížet náhledy issues seřazené podle data přidání. Náhled obsahuje název, stav, story points, zodpovědnou osobu a sprint, ve kterém se issue nachází.

UC15: Změnit stav issue

Uživatel může změnit stav issue. Stavy jsou opened, closed, reopened.

UC16: Prohlížet detail issue

Uživatel může zobrazit detail issue. Detail obsahuje název, popis, story points, osobu, která issue vytvořila, čas vytvoření, stav, zodpovědnou osobu, sprint a související požadavek.

UC17: Komentovat issue

Uživatel může přidat komentář k issue.

UC18: Prohlížet komentáře issue

Uživatel může prohlížet komentáře issue.

UC19: Přiřadit issue pracovníkovi

Uživatel může issue přiřadit sobě nebo jinému pracovníkovi.

UC20: Filtrovat issues podle stavu

Uživatel může filtrovat issues podle jejich stavu.

UC21: Propojit s Gitlabem

Uživatel může vložit přístupový token z Gitlabu a URL adresu Gitlabu. Aplikace tedy lze propojit i s lokálně nasazeným Gitlabem.

5.3.3 Kanban

UC22: Vytvořit sprint

Uživatel může vytvořit sprint. Sprint obsahuje název, popis a datum ukončení.

UC23: Přiřadit issue do sprintu

Issue lze přesovat mezi backlogem a sprinty.

UC24: Vytvořit sprint automaticky

Uživatel může nechat sprint vytvořit automaticky na základě pořadí issues v backlogu.

UC25: Posouvat issue po kanban boardu

Issue lze posovat mezi sloupci na kanban boardu.

UC26: Editovat kanban

Kanban board umožňuje přidávat sloupce, mazat sloupce a měnit pořadí sloupců.

5.3.4 Výkazy a administrace

UC27: Vložit nového uživatele

Uživatel může do systému přidat nového uživatele.

UC28: Smazat uživatele

Uživatel může ze systému vymazat jiného uživatele.

UC29: Vykazovat čas

Uživatel může k issue vykázat odpracovaný čas.

UC30: Zobrazit odpracovaný čas

Uživatel může zobrazit celkový odpracovaný čas u konkrétního issue. Bude moci zobrazit i odpracovaný čas jednotlivých uživatelů u konkrétního issue.

UC31: Zobrazit burdnown chart

Uživatel může zobrazit burndown chart pro jednotlivé sprinty i pro celý projekt.

UC32: Zobrazit statistiky z kanban boardu

Uživatel může u každého issue zobrazit historii pohybu po kanban boardu.

UC33: Stáhnout výkazy

Uživatel může stáhnout výkazy hodin.

5.3.5 Mapování případů užití na požadavky

Mapováním případů užití na požadavky ověřujeme, jestli každému požadavku náleží alespoň jeden případ užití.

	UC1:	UC2:	UC3:	UC4:	UC5:	UC6:	UC7:	UC8:	UC9:	UC10:	UC11:
REQ-18:											✓
REQ-19:	✓	✓			✓		✓				
REQ-20:	✓										
REQ-21:			✓	✓	✓			✓		✓	
REQ-22:		✓				✓					
REQ-23:		✓									
REQ-24:						✓			✓		
REQ-25:						✓					

Tabulka 2: Mapování případů užití na požadavky 1

	UC12:	UC13:	UC14:	UC15:	UC16:	UC17:	UC18:	UC19:	UC20:	UC21:
REQ-1:										✓
REQ-2:	✓	✓	✓		✓				✓	
REQ-3:	✓		✓		✓			✓		
REQ-4:						✓	✓			
REQ-5:				✓						
REQ-7:	✓		✓		✓					
REQ-11:	✓		✓		✓					
REQ-15:	✓				✓					

Tabulka 3: Mapování případů užití na požadavky 2

	UC22:	UC23:	UC24:	UC25:	UC26:
REQ-6:		✓			
REQ-8:	✓				
REQ-9:				✓	✓
REQ-10:				✓	✓
REQ-12:			✓		
REQ-13:		✓			

Tabulka 4: Mapování případů užití na požadavky 3

	UC27:	UC28:	UC29:	UC30:	UC31:	UC32:	UC33:
REQ-14:						✓	
REQ-16:	✓						
REQ-17:		✓					
REQ-26:			✓	✓			
REQ-27:				✓			
REQ-28:							✓
REQ-29:					✓		

Tabulka 5: Mapování případů užití na požadavky 4

6 Návrh

Tato kapitola popisuje použité technologie a návrh databáze.

6.1 Použité technologie

Tato sekce popisuje zásadní technologie, které budou využity.

6.1.1 MEAN Stack

Mean stack je soubor technologií (MongoDB, Express, AngularJS, Node.js) založených na JavaScriptu, které se používají k tvorbě webových aplikací. Umožňuje tvořit servery i klienty. Klient komunikuje se serverem pomocí rest api. Mezi hlavní přednosti patří rychlost, jednoduchost a testovatelnost.

AngularJS Angular[6] je MVC framework pro tvorbu webových aplikací. Výrazně usnadňuje práci a také podporuje vysokou modularitu a testovatelnost. Vzhled prezentační vrstvy můžeme měnit pomocí direktiv. Direktivy určují, jak bude zacházeno s modelem. Zapisujeme je jako atributy HTML značek. Pokud máme například model items a chceme položky vypsat do seznamu, použijeme direktivu ng-repeat.

```
<ul>
        <li ng-repeat='item in items'>{{item}}</li>
</ul>
```

Angular umožňuje i tvorbu uživatelských direktiv, což zjednodušuje znovupoužitelnost kódu. Další výhodou angularu je two-way data binding[7]. Two-way data binding zajišťuje synchronizaci modelu a prezentační vrstvy. Změna modelu se tedy okamžitě projeví na obrazovce. Angular za nás také řeší závislosti mezi jednotlivými komponenty pomocí dependency injection.

Node.JS Node[8] je prostředí, umožňující běh JavaScriptu mimo webový prohlížeč, využívané zejména pro běh back-end serverů pro webové aplikace. Node běží vždy jen v jednom vlákně, souběžnost zajišťuje architektura řízená událostmi. Pokud vlákno provede nějaký požadavek (například dotaz do databáze), vlákno se nezablokuje, ale může vykonávat něco jiného. Pokud přijde odpověď na náš požadavek, vyvolá se událost a provede se kód, který na událost reaguje. Například získání uživatele z databáze provedeme takto:

[6] https://angularjs.org/
[7] https://docs.angularjs.org/guide/databinding
[8] https://nodejs.org/

```
User.FindById(id, function(user) {
        //kod, ktery se provede po ziskani uzivatele z databaze
})
```

Express Express[9] je framework pro Node.js, umožňující tvorbu serverů pro webové a mobilní aplikace.

MongoDB MongoDb[10] je dokumentově orientovaná databáze. Jedná se o NoSQL databázi. Data se ukládají do dokumentů, které jsou tříděny do kolekcí. Filozofie spočívá v tom, že data, která spolu souvisí, jsou uložena v jednom dokumentu a měla by odpovídat požadovaným dotazům na databázi. Při dotazu na databázi tedy obvykle vracíme jen jeden dokument, což je mnohem snažší a rychlejší, než v klasických SQL databázích.

6.1.2 NPM

NPM je balíčkový manažer pro node.js a další technologie. Umožňuje jednoduše stahovat přídavné moduly pro naší aplikaci. Modul naistalujeme příkazem npm install nazevModulu. Závislosti na modulech můžeme zapsat do souboru package.json a poté naistalovat najednou příkazem npm install.

6.1.3 Bower

Bower je balíčkový manažer pro frontend aplikace. Funguje podobně jako NPM. Pro instalaci modulu použijeme příkaz bower install nazevModulu. Závislosti můžeme ukládat do souboru bower.json. Bower balíčky lze stahovat i pomocí NMP. Bower jsem použil kvůli oddělení závislostí pro backend a frontend.

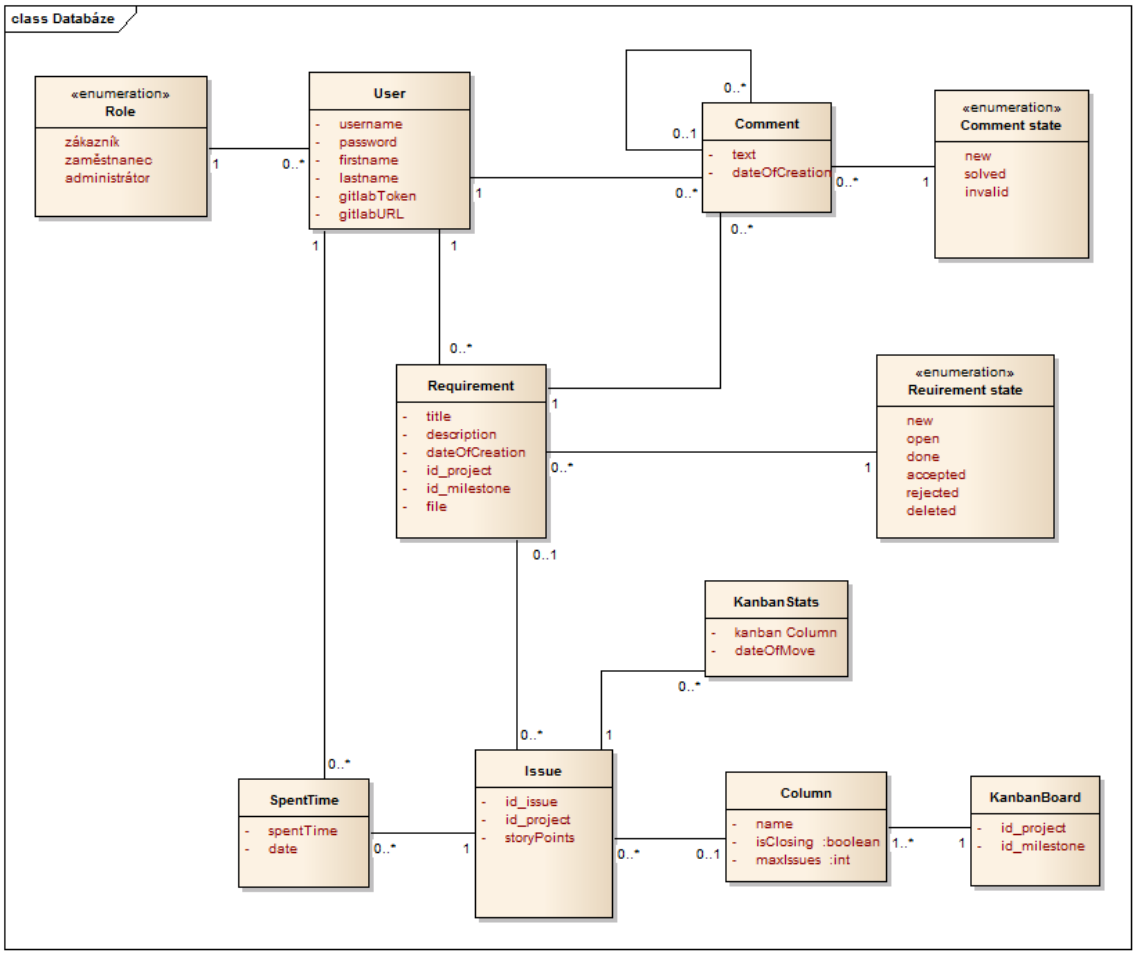
6.1.4 Gulp

Gulp je nástroj pro automatizaci sestavovacího procesu. Umožňuje spojovat kód z více souborů do jednoho, minifikaci JavaScriptu, automatický rebuild při změně sledovaného souboru atd.

6.2 Databáze

Databáze, jak je vidět na obrázku 10, je jednodušší než doménový model. To je způsobeno tím, že některé třídy jsou ukládány pouze do databáze Gitlabu. Atributy s předponou id představují identifikátory objektů na Gitlabu. Data, která lze ukládat na

[9] http://expressjs.com/
[10] https://www.mongodb.org/



Obrázek 10: Model databáze

Gitlab, jsou ukládána na Gitlab. Díky tomu jsou údaje vložené z mé aplikace přístupné i na Gitlabu. Lokální databáze představuje rozšiřující funkcionalitu. Hlavní změnou oproti doménovému modelu je to, že chybí třídy Projekt, Backlog a Sprint. Sprinty jsou ukládány na Gitlab jako milestones a v Backlogu se nachází všechny issues, které nejsou přiřazeny do žádného milestone. Tyto třídy tedy nejsou potřeba v lokální databázi. Stejně tak jsou údaje o projektech přebírány z Gitlabu. Příslušnost objektů ke konktrétnímu projektu je zaznamenána atributem id project.

7 Implementace

Tato kapitola popisuje nejdůležitější prvky aplikace a strukturu kódu. K implementaci byly použity technologie popsané v kapitole 6.1.

7.1 Server

Server poskytuje REST API pro klientskou část aplikace. Probíhá zde komunikace z Gitlabem a spojování dat z lokální databáze a Gitlabu. Klientská aplikace je tak od Gitlabu odstíněna. Ke komunikaci s Gitlab API jsem použil balíček node-gitlab[11], který dotazy maximálně zjednodušuje. Pro komunikaci s lokální databází jsem použil framework mogoose[12].

Klient v každém dotazu musí v hlavičce poslat privátní gitlab token a gitlab URL. Tyto údaje se na serveru přečtou z hlavičky a s jejich pomocí se vytvoří gitlab klient.

Kód je rozdělen na modely a routy. Modely určují podobu objektů ukládaných do databáze a specifikují metody, které mohou s objektem manipulovat. Routy jsou metody, obsluhující konkrétní adresy REST API, mají přístup k datům zaslaným v požadavku a zajišťují odeslání odpovědi. Routy tedy zajišťují zpracování požadavku, získání potřebných dat a odeslání odpovědi klientovi.

Zde je seznam dalších modulů, které jsem v aplikaci použil:

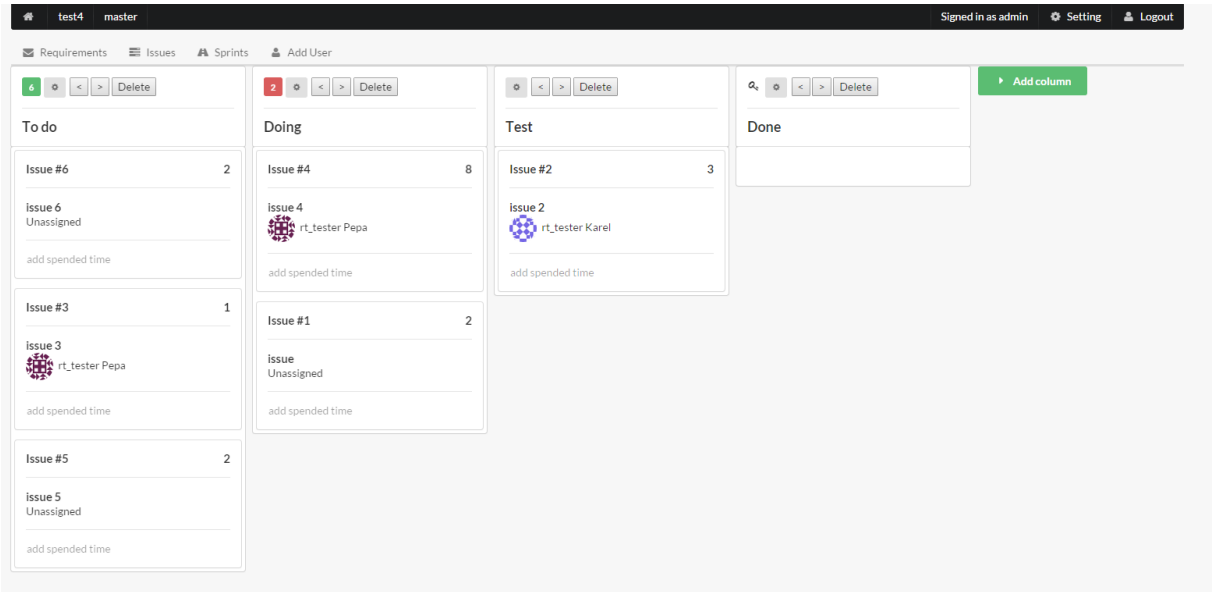
- bcrypt - obsahuje hash funkci, používanou k šifrování hesel.
- body-parser - umožňuje automaticky číst JSON v POST dotazu.
- jwt-simple - umožňuje vytvářet a ověřovat JWT (JSON Web Token). Používá se pro autentizaci uživatele.

7.2 Klient

Kód v klientské aplikaci je rozdělen servisy, kontrolery a html stránky. Servisy zajišťují komunikaci se serverem prostřednictvím dotazů na REST API. Controlery ke komunikaci se serverem využívají tyto servisy. Tím jsou odstíněny od konkrétní podoby dotazu na REST API, což ulehčuje případné změny. Controlery zajišťují výměnu dat mezi serverem a zobrazenými stránkami. Každé stránce přísluší jeden kontroler. Existuje jeden globální kontroler, který zajišťuje přihlašování do aplikace a udržuje údaje o přihlašeném uživateli.

Pro tvorbu uživatelského rozhraní jsem použil css framework Semantic-UI[13]

[11] `https://github.com/node-gitlab/node-gitlab`
[12] `http://mongoosejs.com/`
[13] `http://semantic-ui.com/`



Obrázek 11: Kanban Board

7.2.1 Kanban Board

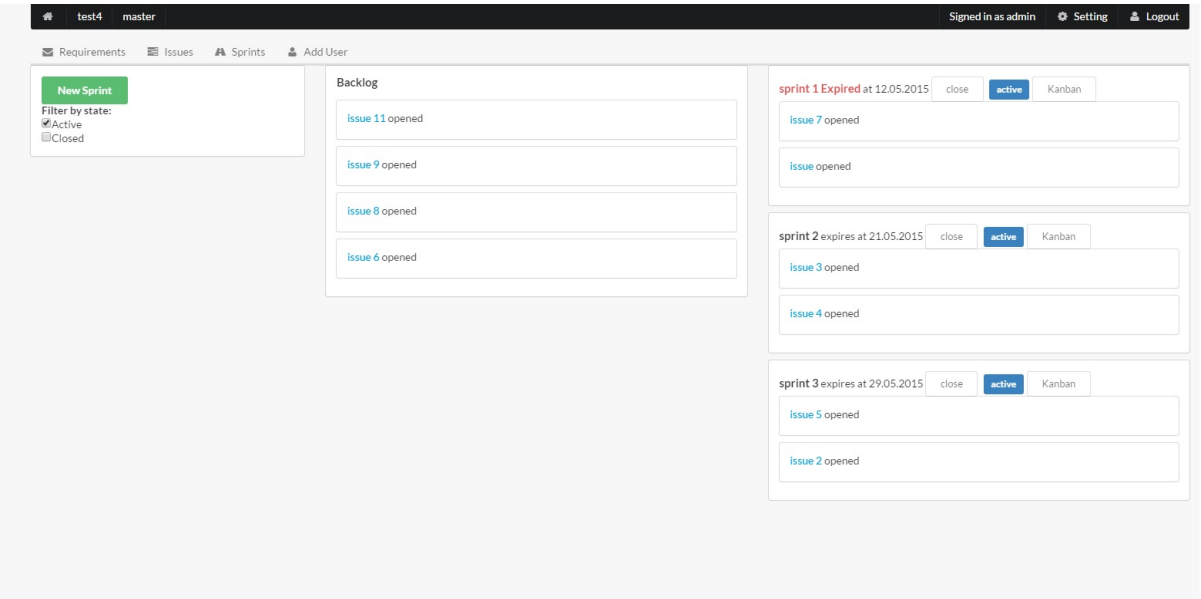
Vzhled implementovaného kanban boardu je zobrazen na obrázku 11. Čísla v levém horním rohu sloupců představují maximální počet issues v daném sloupci. Sloupec Doing má kapacitu 2 a již v něm jsou dvě issues. Proto je číslo zobrazeno červeně. Do sloupce už nelze přidat žádné další issue, dokud se alespoň jedno místo neuvolní. Klíček v levém horním rohu sloupce Done značí, že issue přesunuta do něj, se automaticky uzavřou na Gitlabu. V pravém horním rohu každého issue se nachází story points.

Při implementaci kanban boardu jsem využil balíček ui-sortable[14]. Balíček umožňuje řadit položky v poli pomocí drag & drop. Umožňuje i přesouvat položky mezi různými poli. To se přesně hodí pro implementaci kanban boardu, kde je potřeba řadit položky ve sloupcích a přesouvat je mezi sloupci. Během provádění drag & drop jsou vyvolávány události, na které lze reagovat v kontroleru. Všechny vyvolávané události jsou pospsány v dokumentaci balíčku. Já jsem využil události start, update a stop. Při obsluze události start se zaznamená id posouvaného objektu. Při obsluze update se vyhodnocuje, jestli po přesunutí issue nebude překročena kapacita sloupce. Pokud by měla být překročena, přesunutí se neprovede. Při události stop se provedené změny odešlou na server.

7.2.2 Backlog a Sprinty

Stránka se sprinty je zachycena na obrázku 12. Mezi backlogem a sprinty lze issues přesouvat pomocí drag & drop.

[14] `https://github.com/angular-ui/ui-sortable`



Obrázek 12: Sprints

K implementaci jsem využil modul ngDraggable [15] umožňující drag & drop. Elementy které lze posouvat jsou označeny direktivou ng-drag='true' a elementy, do kterých lze posouvané prvky vkládat jsou označeny direktivou ng-drop='true'. Direktiva ng-drop-success='onDropComplete($data,$event, m)' registruje metodu, která se zavolá po dokončení drag & drop. K přesunu issues v modelu, dojde až v této metodě.

Modul má několik nevýhod a to zejména tu, že drag & drop nelze nijak zakázat, což znemožňuje nastavení přístupu pro různé uživatelské role.

7.3 Nedostatky

Největším nedostatkem implementace je nízká úroveň zabezpečení klienta i serveru. Uživatelské role jsou zajištěny jen různým přístupem k HTML elementům. Konkrétní URL adresy a RESP API uživatelské role nekontroluje.

[15] `https://github.com/fatlinesofcode/ngDraggable`

8 Testy

Úkol testování je zjistit, do jaké míry výsledek odpovídá požadovanému zadání a objevit případné chyby.

8.1 Selenium testy

Selenium testy umožňují testovat uživatelské rozhraní, výstupy na obrazovku a uživatelské vstupy. Umožňují automatizovat průchod aplikací, který může být proveden i ručně.

V případě testování dynamických aplikací mají však Selenium testy jistá omezení. K zaměření elementu je potřeba ho jednoznačně identifikovat, což lze provést nastavením atributu id, nebo odkazováním na pozici v HTML dokumentu. V dynamickém webu někdy ovšem nelze určit jednoznačné id, ani pozice v dokumentu předem a testování je tak omezeno. Také nelze otestovat drag & drop funkcionalitu.

K vytvoření testu jsem použil Selenium IDE[16]. Jedná se o doplněk pro Firefox umožňující nahrávání sekvence kroků a vkládání příkazů z kontextového menu. Vytvořené testy lze ve Firefoxu rovnou spustit.

Vytvořil jsem celkem 9 testů ověřujících základní funkcionalitu a všechny úspěšně prošly, jak je vidět na obrázku 13. Testy pro kanban board a přesování issues mezi backlogem a sprinty se mi nepodařilo vytvořit z důvodů omezení Selenium IDE.

8.2 Akceptační testy

Akceptační testy se provádí po dokončení aplikace. Jejich účel je zhodnotit, jak byly naplněny specifikované požadavky. Akceptační testy jsem provedl ručně vzhledem k požadavkům a případům užití.

Požadavky označené v tabulce 6 hvězdičkou byly splněny jen částečně.

- REQ-22: komentáře nemají stromovou strukturu
- REQ-27: výkazy nelze filtrovat
- REQ-2: není iplementováno vkládání šítků

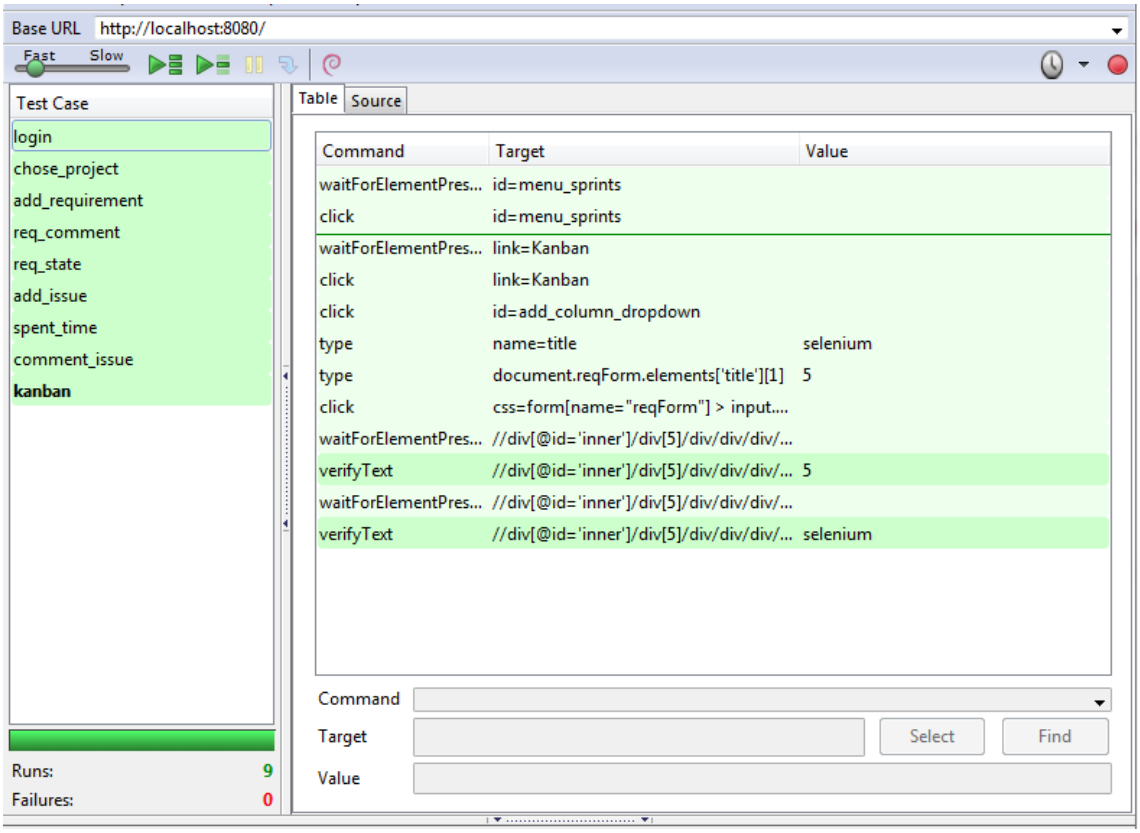
Požadavky byly celkem splněny na 70%[17]

Případy užití označené v tabulce 7 hvězdičkou byly splněny jen částečně.

- UC1: nelze vložit soubor
- UC2: nelze odpovídat na jiný komentář

[16] `http://www.seleniumhq.org/projects/ide/`

[17] Částečně splněné požadavky byly považovány za splněné na 75%



Obrázek 13: Selenium testy

- UC6: komentáře netvoří vlákna
- UC30: nelze filtrovat podle uživatele

Případy užití byly celkem splněny na 80%[18]. Vyššího čísla bylo dosaženo proto, že případy užití a požadavky nejsou svázány jedna ku jedné. Jeden požadavek může mít několik přpadů užití, a proto se výsledek liší.

[18] Částečně splněné byly považovány za splněné na 75%

	Splněno	Nesplněno
REQ-18:		✗
REQ-19:	✓	
REQ-20:		✗
REQ-21:	✓	
REQ-22:	✓*	
REQ-23:		✗
REQ-16:	✓	
REQ-17:		✗
REQ-24:	✓	
REQ-25:	✓	
REQ-1:	✓	
REQ-2:	✓*	
REQ-3:	✓	
REQ-4:	✓	
REQ-5:	✓	
REQ-6:	✓	
REQ-7:	✓	
REQ-8:	✓	
REQ-9:	✓	
REQ-10:	✓	
REQ-11:		✗
REQ-12:		✗
REQ-13:	✓	
REQ-14:	✓	
REQ-15:	✓	
REQ-26:	✓	
REQ-27:	✓*	
REQ-28:		✗
REQ-29:		✗
celkem	21	8

Tabulka 6: Splnění požadavků

	Splněno	Nesplněno
UC1:	✓*	
UC2:	✓*	
UC3:	✓	
UC4:	✓	
UC5:	✓	
UC6:	✓*	
UC7:	✓	
UC8:	✓	
UC9:	✓	
UC10:	✓	
UC11:		✗
UC12:	✓	
UC13:	✓	
UC14:	✓	
UC15:	✓	
UC16:	✓	
UC17:	✓	
UC18:	✓	
UC19:	✓	
UC20:	✓	
UC21:	✓	
UC22:	✓	
UC23:	✓	
UC24:		✗
UC25:	✓	
UC26:	✓	
UC27:	✓	
UC28:		✗
UC29:	✓	
UC30:	✓*	
UC31:		✗
UC32:	✓	
UC33:		✗
celkem	28	5

Tabulka 7: Splnění use cases

9 Zhodnocení

Tato kapitola hodnotí výsledky práce, upozorňuje na některé nedostatky a navrhuje jejich budoucí řešení.

9.1 Současné problémy a náměty na zlepšení

Pokud je změněn milestone z Gitlabu, změna se neprojeví na Kanban Boardu v mé aplikaci. Gitlab API neumožňuje načítat issues příslušící do konkrétního milestone. Issues musí být načteny všechny. Není možné tedy efektivně zjistit, jestli issues na kanban boardu odpovídají issues v příslušném milestone na Gitlabu. Řešením by mohla být asi jedině úprava Gitlab API.
Modul ngDraggable neumožňuje vypnutí drag & drop, což znemožňuje zabránit některým uživatelským rolím přesunu issues mezi sprinty a backlogem. Nejlepším řešením by podle mého názoru bylo stránku předělat a použít modul ui-sortable, který byl použit pro implementaci kanbanu. Díky tomu by mohla být implementována i prioritizace issues v backlogu pomocí pořadí. To by umožnilo i automatické tvoření sprintů na základě pořadí v backlogu.
V kódu se nacházejí nepoužívané metody, případně zakomentované části kódu, které jsem nestihl odstranit. U některých částí kódu by bylo vhodné před dalším postupem zvážit refactoring.
Ve vykazování času a tvorbě sprintu chybí komponenta k výběru data. V době vývoje Semantic-ui nenabízel komponentu pro výběr data. Pole je označeno typem date. Webový prohlížeč Chrome zobrazuje date picker automaticky, Firefox a Safari zatím ne. Bylo by dobré nějaký date picker zakomponovat přímo do aplikace, případně zjistit, jestli Firefox s Safari mají v plánu podporu typu date.

9.2 Osobní zhodnocení

Aplikace splňuje zhruba tři čtvrtiny specifikovaných požadavků. Většina nesplněných požadavků ovšem nebyla příliš podstatná a nebyla nutná ke splnění zadání práce. Důležitým faktorem, který ovlivnil kvalitu práce je fakt, že jsem na začátku práce neovládal žádnou z využívaných technologií, což se rozhodně projevilo na kvalitě kódu. V průběhu práce jsem se stále učil a zlepšoval, ale na refaktoring většinou nezbýval čas.

Přestože aplikace ještě není dokonalá, tak vzhledem k rozsahu požadavků a výše zmíněným faktům, považuji výsledek své práce za úspěšný.

Literatura

[1] What is Kanban?. PETERSON, David. *Kanban Blog* [online]. © 2009-2015 [cit. 2015-04-12]. Dostupné z: `http://kanbanblog.com/explained/`

[2] ANDERSON, David J. *Kanban: successful evolutionary change in your software business*. Sequim, Wash: Blue Hole Press, 2010. ISBN 09-845-2140-2.

[3] What is Kanban?. *EVERYDAY KANBAN* [online]. © 2015 [cit. 2015-04-28]. Dostupné z: `http://www.everydaykanban.com/what-is-kanban/`

[4] JIT (Just-in-time). *ManagementMania* [online]. © 2011-2013, 25.04.2013 [cit. 2015-04-21]. Dostupné z: `https://managementmania.com/cs/just-in-time`

[5] KOTAČKA, Vít. Kanban, lehký úvod. *SoftWare Samuraj* [online]. 4.3.2014 [cit. 2015-04-21]. Dostupné z: `http://www.sw-samuraj.cz/2014/03/kanban-lehky-uvod.html`

[6] JANÁK, Jiří. *Issue Tracking Systems*. Brno, 2009. Dostupné z: `http://is.muni.cz/th/60778/fi_m/thesis.pdf`. Diplomová práce. Masarykova Univerzita. Vedoucí práce RNDr. Jan Pavlovič.

[7] Scrum Guides [online]. 2015 [cit. 2015-05-18]. Dostupné z: `http://www.scrumguides.org/`

[8] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Vyd. 1. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.

[9] A physical Kanban board with a basic, three-step workflow [obrázek]. *LeanKit* [online]. © 2014 [cit. 2015-04-23]. Dostupné z: `http://leankit.com/kanban/kanban-board/`

[10] Kanban-board-1 [obrázek]. PETERSON, David. *Kanban Blog* [online]. © 2009-2015 [cit. 2015-04-13]. Dostupné z: `http://kanbanblog.com/explained/`

[11] *GitLab* [online]. 2015. [cit. 2015-04-23]. Dostupné z: `https://about.gitlab.com/`

[12] *Taiga* [online]. 2014. [cit. 2015-05-13]. Dostupné z: `https://taiga.io/`

[13] *Pivotal Tracker* [online]. 2015. [cit. 2015-05-14]. Dostupné z: `http://www.pivotaltracker.com/`

[14] *LeanKit* [online]. 2015. [cit. 2015-05-14]. Dostupné z: `http://leankit.com/`

[15] *Trello* [online]. 2015. [cit. 2015-05-14]. Dostupné z: `https://trello.com/`