Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Dokument
k inžinierskemu dielu

Tím 24

Študijný program: Inteligentné softvérové systémy, Internetové technológie
Ročník: 1
Predmet: Tímový projekt
Cvičiaci: Ing. Eduard Kuric, PhD.

Obsah

Obsah obrázkov

1 Úvod

Prioritou nášho tímu je vytvoriť webovú aplikáciu, ktorá má v prvom rade slúžiť ľuďom, zjednodušovať ich každodennú prácu, resp. skvalitňovať služby. Z uvedených tém sme po diskusii vybrali tému “Kolaboratívne prototypovanie používateľských rozhraní [Collab-UI]”. Táto téma nás zaujala najmä kvôli kolaborácii používateľov a možnosti real-time interakcie, na ktorý by podľa nášho názoru bolo vhodné použiť technológiu, ktorá sa v súčasnej dobe dostáva do popredia a teda NodeJS, vďaka ktorej by sme vedeli zabezpečiť spomínanú interakciu používateľov v reálnom čase. K tejto skutočnosti nám pomôže knižnica socket.io, ktorá bude slúžiť najmä na komunikáciu prostredníctvom posielaní správ medzi serverom a klientom. Nástrojov k prototypovaniu existuje v dnešnej dobe mnoho, no napriek tomu ani jeden nauvažuje nad kolaboráciou v reálnom čase, resp. ju len imituje. Cieľom nášho tímu je vytvoriť schopný nástroj, ktorý túto skutočnosť dokáže v najlepšej možnej miere. Ďalej nám pride zaujímavé napr. pridanie verziovania jednotlivých prototypov, ukladanie histórie zásahov do prototypu aby sme vedeli process vývoja zachytiť a opätovne prehrať vo forme animácie.

2 Globálne ciele pre ZS

Keďže ide o tímový projekt, na ktorom spolupracujú ľudia, ktorí predtým spolu nepracovali, celý zimný semester, no najmä prvá polovica sa vyznačuje inicializáciou činností, dohadovaním a konfiguráciou mnohých komponentov a podporných nástrojov, no v neposlednom rade rozdelením si zodpovedností a začatím implementácie prvých verzií vybraných modulov systému.

Cieľom projektu pre zimný semester je najmä vytvorenie manažmentu používateľov, ktorý sa do nášho systému budú registrovať. Toto zahrňa registráciu a prihlásenie používateľov, vytvorenie úvodnej stránky pre nalákanie zákzníkov nato aby používali náš produkt.

Ďalej by sme sa radi venovali manažmentu projektov. Teda ak sa už používateľ registruje do systému prepokladáme, že chce vytvárať projekty, na ktorých bude pracovať. Okrem vytvárania projektov, bude potrebovať aj ich správu a teda mazanie alebo úpravu konkrétneho projektu.

Keď už máme systém, do ktorého sa dá prihlásiť ako aj používateľov, ktorí vedia vytvárať projekty, potrebujeme ďalej manažment kolaborantov tímu. Teda ak chce vlastník projektu pridávať svojich členov do tímu na projekt, na ktorom spoločne pracujú potrebuje ich o tejto akcii notifikovať. Taktiež pokiaľ dôjde k nezhodám medzi vlastníkom a kolaborantom, môže ho vlastník odstrániť z projektu alebo dočasne zablokovať. Taktiež mu vie meniť práva a rozhodovať o tom, či môže do projektu prispievať alebo ho len sledovať.

Posledným krokom, ktorým zamýšlame v zimnou semestri je samotný editor aplikácie, v ktorom bude používateľ vedieť editovať svoj vytvorený projekt a zároveň ho vedieť uložiť. Okrem editora sa budeme zaoberať aj samostatnou kolaboráciou, teda používateľ bude vidieť v reaálnom čase čo kolaborant robí, taktiež bude vidieť tiež zoznamov kolaborantov, ktorí akurát editujú prototyp.

3 Globálne ciele pre LS

Ako sme už spomínali v zimnom semestri dokončíme zjednodušenú verziu pre editor. V ďalšom semestri máme na plane nasledujúce user story:

- Možnosť vytvárania viacerých stránok v jednom projekte a ich hierarchické prelinkovanie
- Verziovanie, ktoré sme tento semester nestihli
- Ukladanie histórie akcií na prototype a následné prehranie (retrospektíva)
- Chat + Audio Chat
- Pridávanie poznámok k jednotlivým elementom
- Pridávanie hlasových poznámok k jednotlivým elementom
- Preview – interaktívny prototyp
- Profil používateľa

4 Celkový pohľad na systém

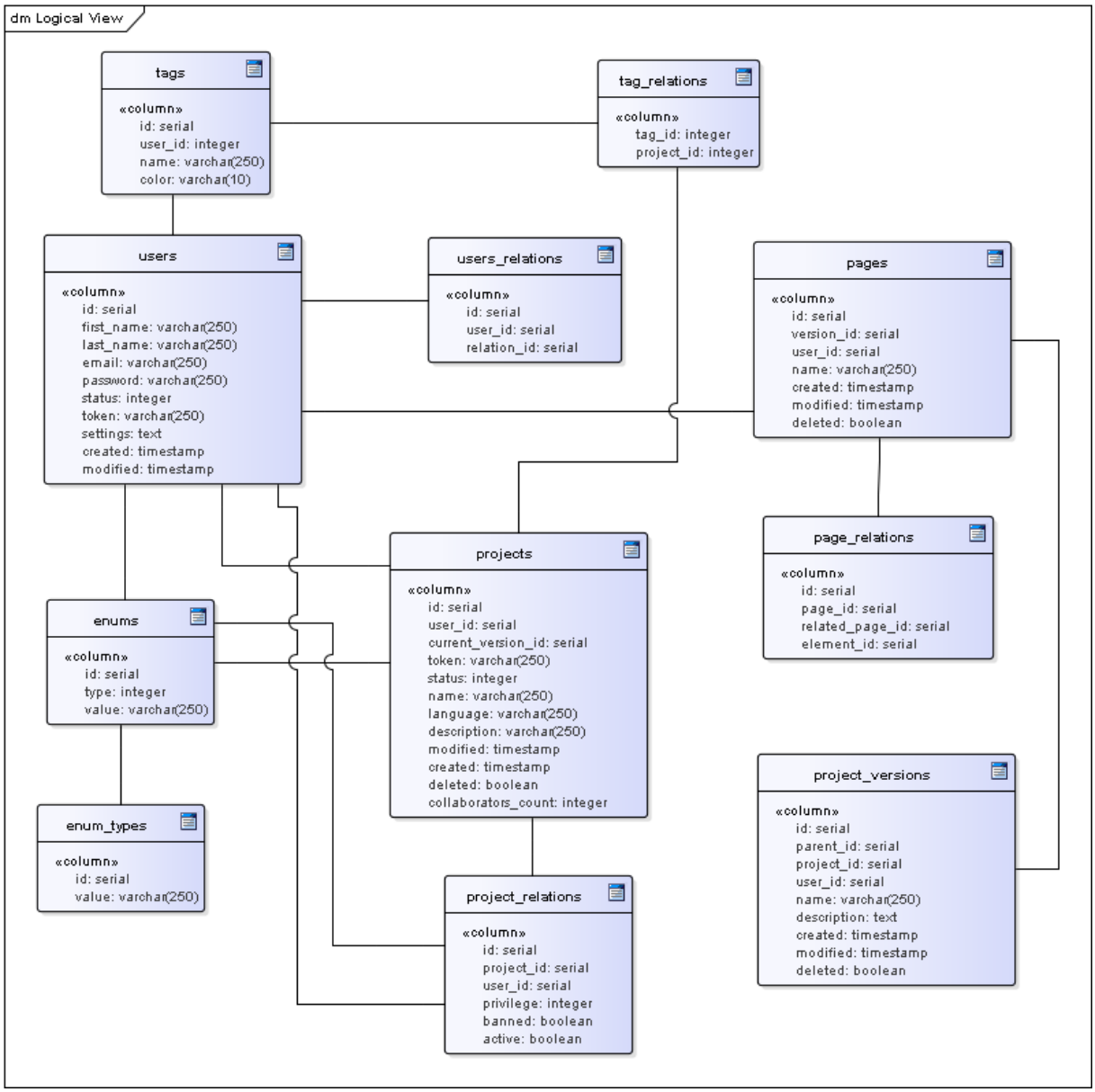
4.1 Architektúra

4.1.1 Backend

Webové riešenie je postavené na webovom MVC rámci CakePHP 3.5 Red Velvet. Používa 2 DB a to:

- PostgreSQL – relačná databáza
- Cassandra – nerelačná databáza

Okrem CakePHP rámca je súčasťou aplikácie NodeJS Server, ktorý beží nezávisle od webového riešenia.



Obrázok 1 - Celkový databázový model pre postgresql

4.1.1.1 Manažment BE knižníc (CakePHP)

Manažment závislostí sa rieši prostredníctvom nástroja Composer. Zoznam použitých knižníc sa nachádza v composer.json

4.1.1.2 Mechanizmus spracovania chýb

Štandardné CakePHP spracovanie chýb je v aplikácii nahradené vlastným mechanizmom. Nachádza sa v priečinku:
root/src/Error/AppErrorHandler.php.

Spracováva špeciálne výnimky definované v priečinku:
root/src/Error/Exception/...

Všetky chyby a mnoho iných informácií je možné nájsť v logoch v priečinku:
root/logs/...

V prípade ak nastala fatálna chyba pri prístupe na stránku aplikácia vyhadzuje chybovú stránku 404, 403 alebo 500. Šablóny sa nachádzajú v priečinku:
root/src/Template/Error/...

Keďže toto riešenie bolo prebraté z pôvodnej infraštruktúry tak ho časom budeme refaktorovať.

Ďalej ak nastane chyba pri požiadavke typu ajax, spracuje ju vlastná iplementácia spracovania chýb zasadená do frontendového rámca aoweb a to nasledovne:

- Status == 422 > niektorý atribút poslaný na backend nie je validný, jednotlivé chyby sú zobrazené používateľovi vo forme notifikácií

- Iný status > nastala neočakávaná chyba, ktorá je zobrazená vo forme modálneho okna, v ktorom sú bližšie informácie o chybe

4.1.1.3 Mechanizmus vykreslenia stránok

Zoberme si napríklad URL: http:\\domena\dashboard\index URL by sme mohli rozdeliť do 3 častí:

1. http:\\domena 2. \dashboard 3. \index
2. časť hovorí o tom, ktorý Controller stránka používa, teda v tomto prípade ide o root/src/Controller/DashboardController.php 3. časť hovorí o použitej metóde Controller-a, teda ide o metódu index()

Metódy ako také rozdeľujeme na tie, ktoré vykresľujú stránky, teda nazvime ich „stránkové“ a na tie, ktoré obhospodárujú AJAX-ové volania, nazvime ich pre jednoduchosť „ajaxové“.

Ak ide o stránkovú metódu posielame na FE dáta vždy v atribúte **$php_response**. Ak ide o ajaxovú metódu v atribúte **$result**.

S týmito atribútmi pracuje nadradená trieda root/src/Controller/AppController.php
Aby sme zabezpečili čitateľnosť kódu zoskupujeme spoločné funkcionality do tzv. Component-ov, ktoré sa nachádzajú v root/src/Controller/Component/...
Čo sa týka šablón, teda .CTP súborov. Nachádzajú sa v root/src/Template/... pričom priečinok musí byť pomenovaný ako Controller a súbor.ctp ako metóda. Teda ak si zoberieme vyššie spomínaný príklad, potom URL: http:\\domena\dashboard\index hľadá šablónu v root/src/Template/Dashboard/index.ctp

Keďže toto riešenie bolo prebraté z pôvodnej infraštruktúry tak ho časom budeme refaktorovať.

4.1.1.4 Databázový model, ORM

Ako ste sa už mohli vyššie dočítať používame relačnú DB PostgreSQL a nerelačnú DB Cassandra.

Štruktúru databáz manažujeme v priečinku, v ktorom sa nachádza vždy aktuálna verzia databázy:
root/src/Cassandra/collabui.sql a root/src/Database /collabui.cql

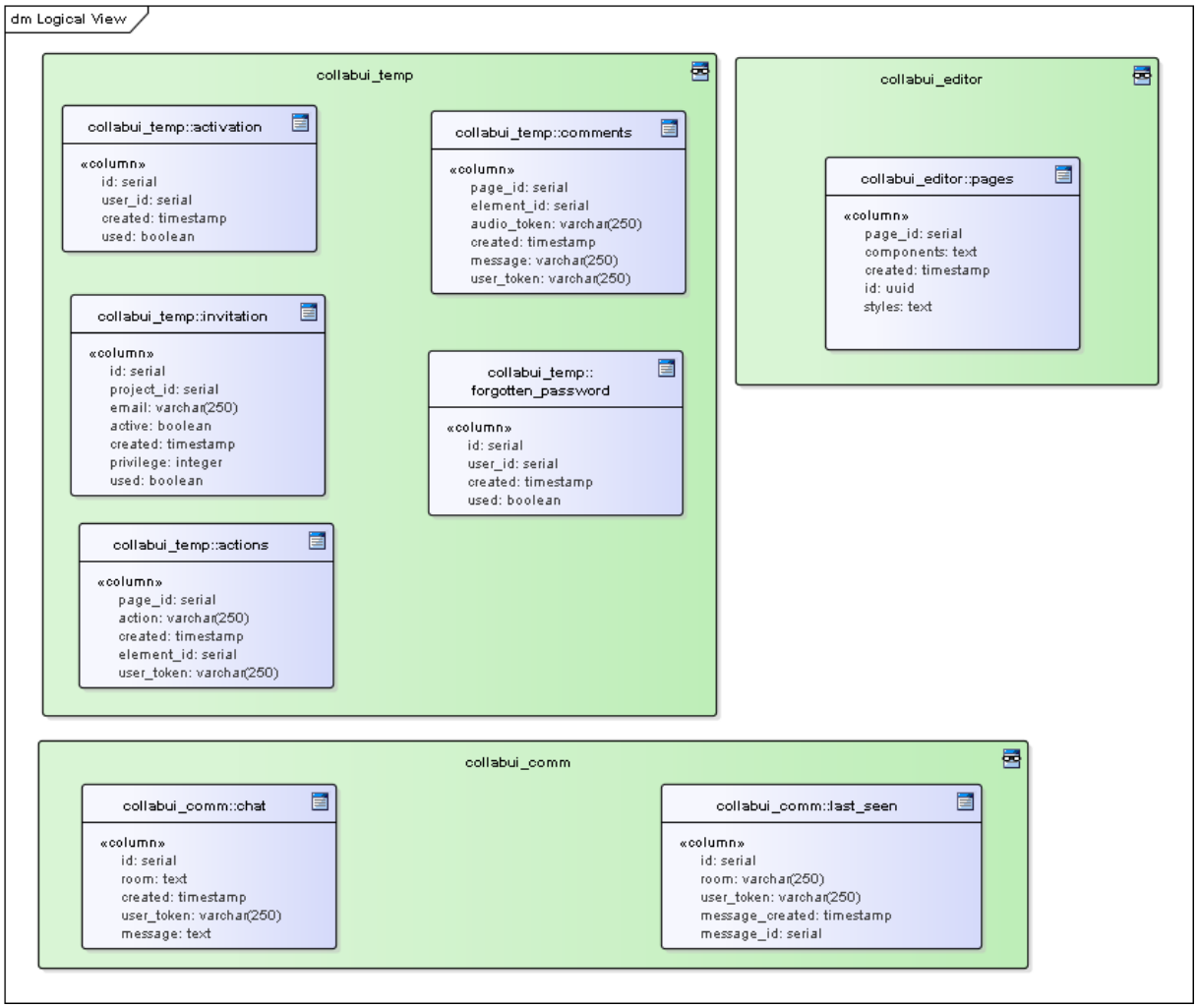
Každú tabuľku z RDB je potrebné „upiecť“, teda vytvoriť PHP Model, prostredníctvom príkazu z root adresára:
./bin/cake bake model nazov_tabulky

Databáza Cassandra obsahuje v súčasnosti 3 keyspace:

- collabui_temp
- collabui_comm
- collabui_editor

a disponuje nasledovnými tabuľkami.

Poznámka: Zelený štvorec v NoSQL diagrame znázorňuje keyspace



Obrázok 2 - Tabuľky pre databázu Cassandra

4.1.1.5 NodeJS Server

NodeJS Server slúži na obhospodárenie požiadaviek vznikajúcich pri kolaborácii, chat, atď. Server je rozdelený do modulov, ktoré sa nachádzajú v priečinku root/node/modules/... , pričom sa spájajú v súbore root/node/Server.js
NodeJS Server pracuje a komunikuje výlučne s nerelačnou DB Cassandra. Server štartujeme z adresára root/node/.. v konzole príkazom: npm start alebo
node Server.js

Keďže toto riešenie bolo prebraté z pôvodnej infraštruktúry tak ho časom budeme refaktorovať.

4.1.1.6 Manažment BE knižníc (NodeJS)

Manažment závislostí sa rieši prostredníctvom nástroja NPM. Zoznam použitých knižníc sa nachádza v priečinku:
root/node/package.json.

Taktiež každá z backendových knižníc, ktorá sa v aplikácii používa sa nachádza v priečinku:
root/node/node_modules/... a taktiež v root/node/package.json

Priečinok root/node/node_modules/... sa nachádza v .gitignore
Po pridaní knižnice musí každý člen tímu spustiť z adresára root/node/.. v konzole príkaz:

- npm update --dev

Ak potrebuje člen tímu pridať novú knižnicu použije príkaz:
npm install nazov_kniznice –-save

Hore uvedeným príkazom pridávame knižnice pre NodeJS Server.
Ak chceme pridať závislosť typu dev, použijeme:

- npm install nazov_kniznice –-save-dev

Závislosti typu DEV slúžia najmä pre buildovanie scss a js.

4.2 Frontend

Ako ste sa už dočítali vyššie šablóny sú v CakePHP označovené súbormi typu .ctp a nachádzajú sa v root/src/Template/...

Každá šablóna je zasadená do tzv. dispozície, ktorá sa definuje na backend rovno v metóde stránky a nachádza sa v priečinku :
root/src/Template/Layout/...
V dispozícii sa nachádzajú potrebné importy, ktoré sú špecifické pre danú dispozíciu.

Ďalšou zaujímavosťou sú tzv. elementy. Zjednodušujú komplexitu a veľkosť šablón, rozbíjajú ju na viacero menších súčiastok.
Nachádzajú sa v priečinku:
root/src/Template/Element/...

Okrem šablón patria k frontendu samozrejme aj css a js.

4.2.1.1 Manažment FE knižníc

Manažment závislostí sa rieši prostredníctvom nástroja Bower. Zoznam použitých knižníc sa nachádza v priečinku:
root/bower.json

Každá frontendová knižnica použitá v aplikácii sa nachádza v priečinku:
root/webroot/vendor/... a taktiež v root/bower.json

Priečinok root/webroot/vendor/... sa nachádza v .gitignore Po pridaní knižnice musí každý člen tímu spustiť z root adresára v konzole príkaz:
bower update

Ak potrebuje člen tímu pridať novú knižnicu použije príkaz:

bower install nazov_kniznice –-save

4.2.1.2 Buildovanie SASS a BrowserSync

Aplikácii využíva technológiu sass. Aby sme dokázali .scss súbory kompilovať využívame technológiu gulp.

gulp úlohy sa nachádzajú v súbore:
root/node/gulpfile.js

Ako predloha podobne ako pri app.php slúži root/node/gulpfile.default.js, ktorý je potrebné pri prvom rozbehávaní zduplikovať, premenovať a nastaviť potrebne atribúty.

Z dôvodu aby sme nemuseli po každej zmene v zdrojových súboroch obnovovať vyvíjanú stránku je použitá knižnica browser-sync, ktorá dokáže prebuildovať súbory a zároveň obnoviť stránku v reálnom čase.

Použijeme príkaz z adresára root/node/... :

- gulp watch

Príkaz nám otvorí aplikáciu na zadefinovanom porte a bude reagovať na zmeny v zdrojových súboroch .ctp, .js a .scss

Súbory .sccs sa následne kompilujú do .css do priečinka root/webroot/css/...

4.2.1.3 Minifikácia JS a CSS, optimalizácia obrázkov

Bude riešené prostredníctvom gulp taskov.

5 Moduly

Samotný projekt sa skladá z niekoľkých modulov. Tieto moduly predstavujú základ aplikácie v podobe vytvárania projektov, registrácie a prihlásenia ako aj správu kolaborantov v tíme. Jednotlivé moduly sú nasledujúce:

- Manažment používateľov
- Manažment projektov
- Manažment kolaborantov
- Editor
- Manažment tagov

5.1 Manažment používateľov

5.1.1 Úvod

Prvý bod, ktorý sme naplánovali v rámci projektu Collab-ui je nastavenie a naštýlovanie prezentačnej stránky a správa používateľského konta. Prezentačná stránka slúži nato aby používateľovi, ktorý ešte systém nepozná poskytla jednotlivé informácie o projekte. Tieto informácie sú opis základných funkcii systému, technológie, s ktorými bol systém vytvorený, dodatočné služby poskytované v rámci jednotivých častí projektu, kto na tomto projekte pracoval a vlastne ako dokáže sám používateľ tento system používať. Toto všetko by sa malo nachádzať na úvodnej stránke projektu, ktorá má používateľa zaujať a presvedčiť aby si založil účet a samotný systém aj vyskúšal. Používateľ sa z úvodnej stránky môže prekliknúť na registráciu, vytvoriť si účet a hneď potom ako bude tento účet aktivovaný sa môže prihlásiť a začať sa oboznamovať s jednotlivými funkaciami systému.

5.1.2 Analýza

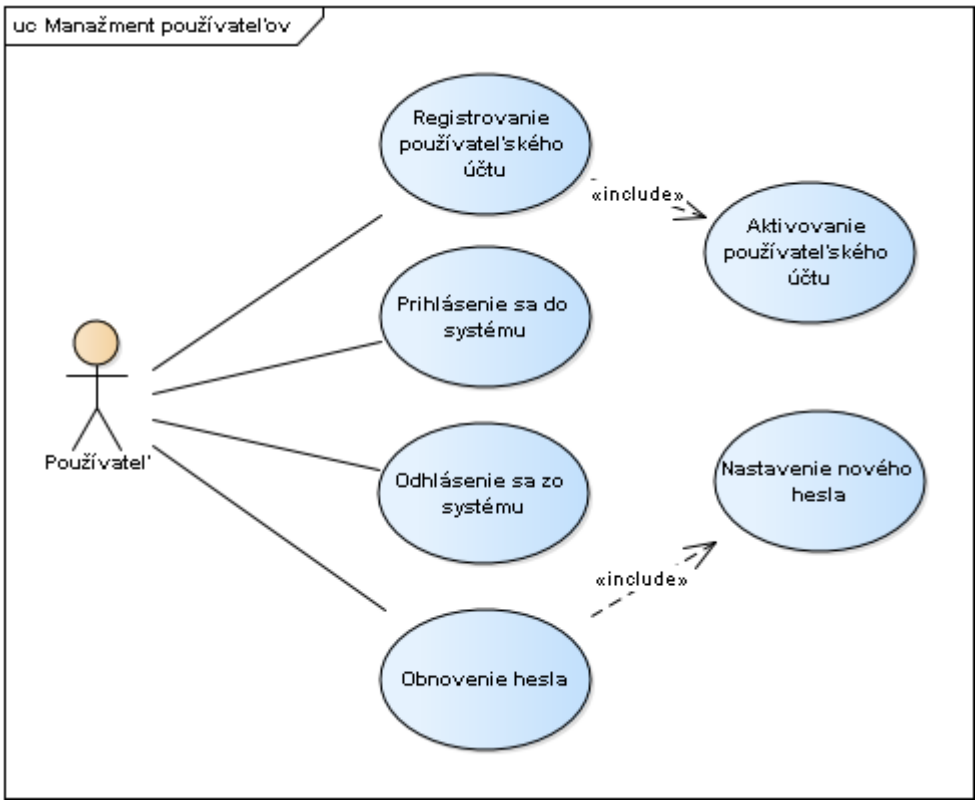
V rámci analýzi sme sa zameriavali nato kto bude nás hlavný používateľ. Náš používateľ by predovšetkým mal byť niekto kto je technicky zdatný. Primárne sa teda budeme sústredovať na dizajnérov a programátorov popr. frontend developerov. Týchto používateľov sme vybrali práve kvôli špecifickosti systému. Keďže systém bude slúžiť na kolaboratívnu prácu jednotlivých členov a teda na kolaboratívne vytváranie wireframov. Práve kvôli analýze koncového používateľa vieme lepšie zostaviť prezentačnú stránku.

5.1.3 Návrh

Manažment používateľov sa skladá z nasledujúcich UC:

- Registrovanie používateľského účtu
- Prihlásenie sa do systému
- Odhlásenie sa zo systému
- Obnovenie hesla
- Aktivovanie používateľského účtu
- Nastavenie nového hesla

Nasledujúci use case diagram znázorňuje základný sled akcií používateľa pri vytváraní používateľského účtu.



Obrázok 3 - UC diagram pre manažment používateľov

UC Registrovanie používateľského účtu

Hlavný tok:

1. Používateľ navštívil registračnú podstránku
2. Používateľ vyplní potrebné polia (meno, priezvisko, email, heslo)
3. Používateľ odošle formulár
4. Systém odošle používateľovi email s potvrdením registrácie
5. Pokračuje s UC Aktivovanie používateľského účtu

UC Aktivovanie používateľského účtu

Hlavný tok:

1. Používateľ potvrdí URL pre aktivovanie účtu
2. Systém zaktivuje používateľov účet a presmeruje používateľa na prihlasovací formulár
3. Prípad použitia končí.

UC Obnovenie hesla

Hlavný tok:

1. Používateľ zvolí možnosť zabudnuté heslo
2. Systém vyzve používateľa pre zadanie emailu pre dané konto
3. Používateľ vyplní pole pre email a odošle požiadavku
4. Systém zverifikuje používateľa
5. Ak sa v systéme používateľ nachádza s vytvoreným kontom, systém pošle na email obnovu hesla
6. Pokračuje s UC Nastavenie nového hesla

UC Nastavenie nového hesla

Hlavný tok:

1. Používateľ potvrdí URL pre obnovu hesla
2. Používateľ je pomocou potvrdenej URL presmerovaní na podstránku nastavenia nového hesla
3. Používateľ zadá nové heslo a potvrdí svoju akciu
4. Systém zvlaliduje vpísané údaje
5. Prípad použitia končí

UC Prihlásenie sa do systému

Hlavný tok:

1. Používateľ navštívi prihlasovaciu podstránku
2. Používateľ sa prihlási do systému
3. Systém presmeruje používateľa na dashboard aplikácie
4. Používateľ sa odhlási zo systému
5. Prípad použitia končí

UC Odhlásenie sa zo systému

Hlavný tok:

1. Používateľ sa odhlási zo systému
2. Systém presmeruje používateľa na úvodnú stránku
3. Prípad použitia končí

5.1.4 Implementácia

5.1.4.1 Úvodná stránka

Za úlohu bolo rozumne rozložiť elementy úvodnej stránky a naštýlovať ju. Tento proces sa dá rozdeliť do niekoľkých bodov:

- Naštýlovanie navigačného baru, v ktorom boli prepísané texty a pridaný script pre flawless scrolling na ukotvenie ďalej v tele stránky

- Pridanie obsahu do samotného tela stránky. To je rozdelené na niekoľko sekcií. Prvá sekcia obashuje carousel, ktorý zabezpečuje rotovanie multimediálneho obsahu. Na tejto sekcii a potom každom obrázku je aplikovaný parallax scrolling effect. Ďalej sú tam sekcie prístupné z navigačného baru. Sekcie bez pozadia zvyčajne obsahúju textový popis a buttony. Na konci je zoznam technológií.

- V dolnej časti stránky je footer, kde prebehli len menšie úpravy, čo sa týka odstránenia niektorých prvkov. Footer je tiež rozdelený na dve časti aby mohol byť v krátkej forme využívaný aj v ostatných častiach projektu.

5.1.4.1.1 Akceptačné kritéria

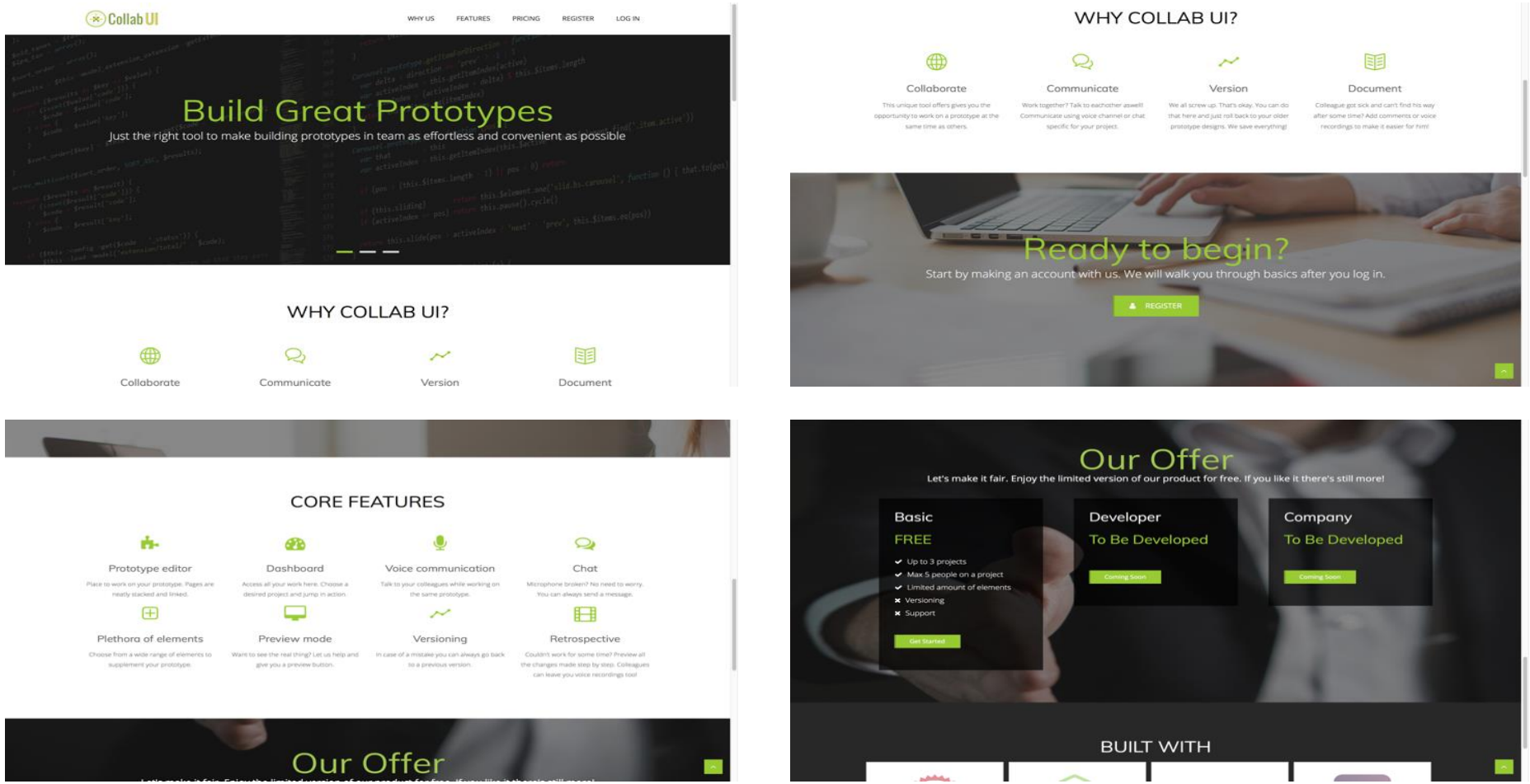
- Po príchode na URL sa používateľovi zobrazí úvodná stránka s hlavnými informáciami, ktoré budú obsahovať dôležité fakty potrebné pre prvotné získanie záujmu používateľa.

- Používateľ sa vie z úvodnej stránky dostať na jednotlivé podstránky.

5.1.4.1.2 Validácia

V rámci user story “Úvodná stránka” nebolo nutné vykonávať akékoľvek validácie, keďže sa jedná najmä o prezentačnú časť projektu.

5.1.4.1.3 Obrazovky úvodnej stránky



Obrázok 4 - Obrazovka pre úvodnú stránku

5.1.4.2 Registrovanie a aktivovanie používateľského účtu

Cieľom tohto user story bolo vytvorenie najzákladnejšej časti manažmentu používateľov a teda vytvorenie možnosti registrovania sa pre používateľa, po ktorej bude nasledovať akcia aktivácie používateľského účtu. K tejto udalosti prislúcha konkrétna URL pre registráciu: /registration

Registračný formulár pozostáva z nasledujúcich polí:

- Meno
- Priezvisko
- Email
- Heslo

5.1.4.2.1 Akceptačné kritéria

- Používateľ si dokáže zaregistrovať nový účet na URL: /registration
- Po vytvorení účtu obdrží používateľ email "Welcome" s aktivačnou URL linkou
- Používateľ si dokáže zaktivovať účet
- Po aktivácii je používateľ presmerovaný na URL: /log-in

5.1.4.2.2 Validácia

Samozrejme pri každej registrácii je doležitá samotná validácia používateľa. V tomto prípade sme robili 2 typy validácie a to na:

- Frontend
- Backend

V rámci frontendovej validácie sa jedná o skontrolovanie polí ako je:

- Meno
- Priezvisko
- Email
- Heslo

Taktiež v rámci kvality hesla sa kontroluje aj sila hesla, ktorá by mala byť aspoň 10 znakov. Sila hesla je taktiež graficky reprezentovaná farebným loading barom. Táto funkcionalita sa nachádza v utilities.js a potom sa len binduje na inputy s heslom.

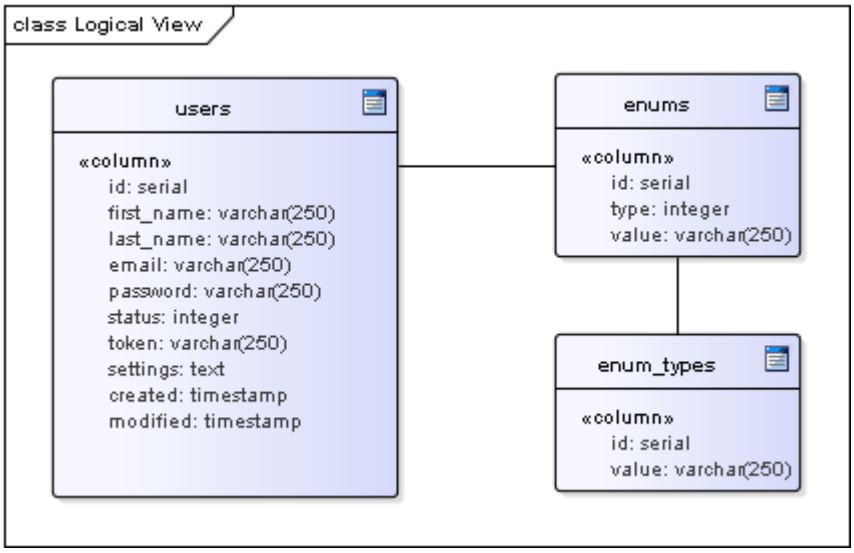
Čo sa týka backendovej validácie používateľa, tá je vykonávaná pomocou validačných funkcii v UserTable.php, kde sa pre každý typ formuláru zavolá iná validačná funkcia. Naviazanie na formulár a validačných funkcii prebieha v Usercontroller.

V prípade, že by sa používateľ snažil prihlásiť bez registrácie alebo potvrdenia registračného emailu dostane správu “This account has not been activated.” a pokiaľ chce ďalej pokračovať, musí sa zaregistrovať.

Keď sa používateľ zaregistruje, systém mu vygeneruje token a pošle aktivačný link pre jeho konto, ktorý obdrží na svoj mail. Po presmerovaní sa používateľský účet stane aktívnym. Okrem spomínaných skutočností sa neustále kontroluje aj stav používateľského účtu a teda ak administrator používateľa zabanoval, pri prihlásení uvidí správu “This account has been banned.”

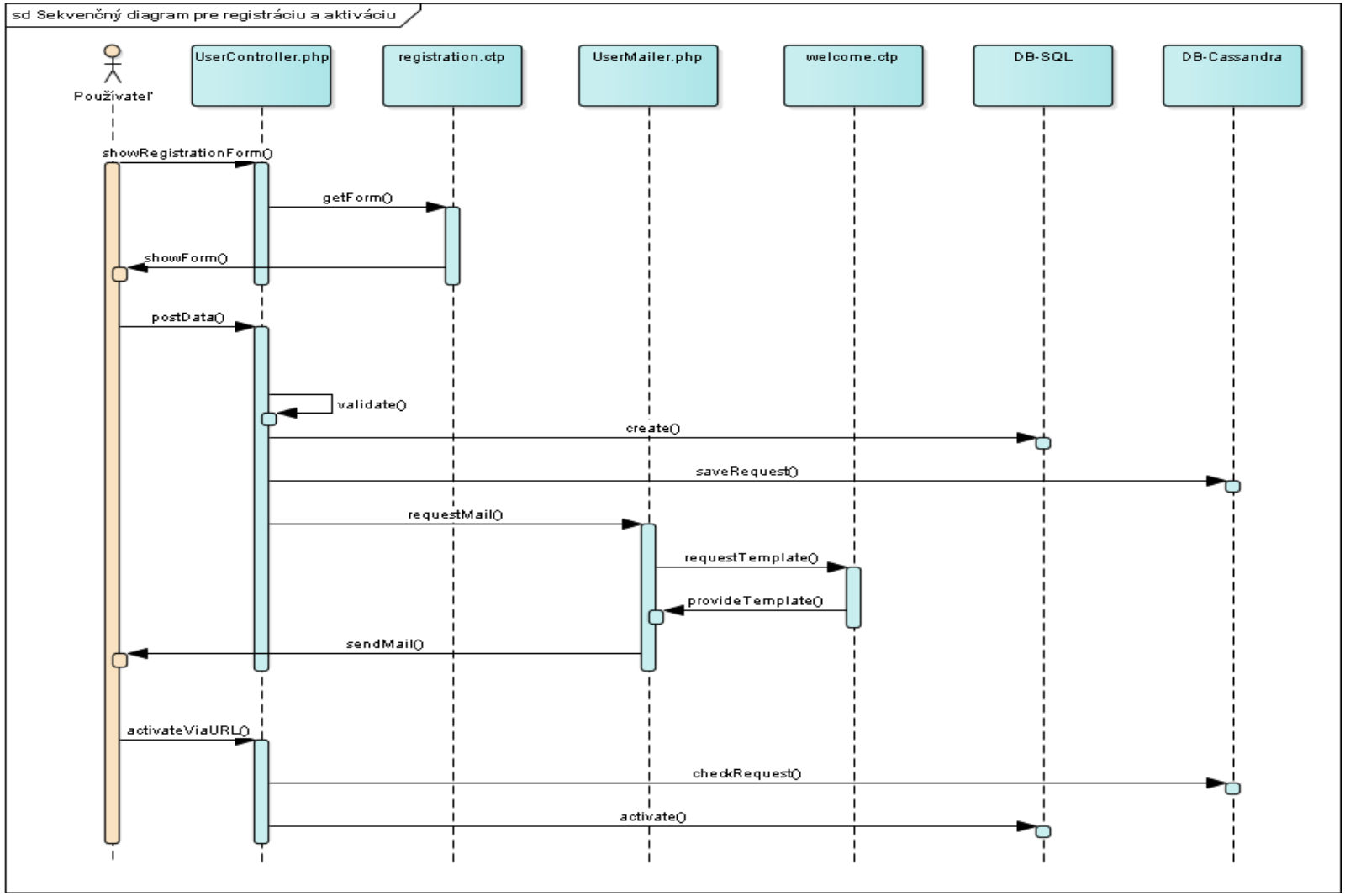
5.1.4.2.3 Databázový model

Čo sa týka databázového pohľadu, sa pri registrácií a aktivácií používateľa používajú nasledovné tabuľky.



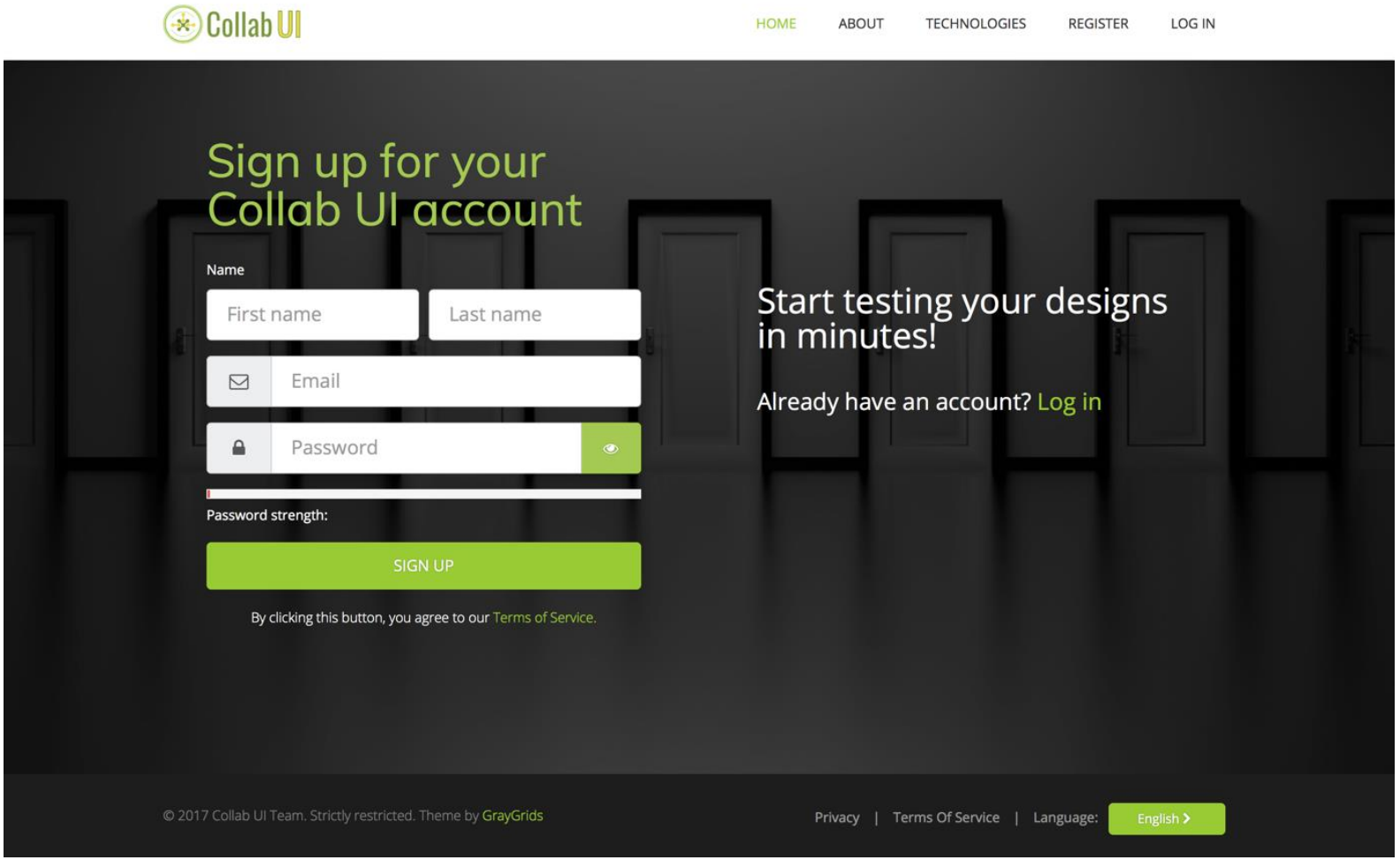
Obrázok 5 - Databázový model registráciu používate

5.1.4.2.4 Sekvenčný diagram



Obrázok 6 - Sekvenčný diagram pre registráciu používateľského účtu

5.1.4.2.5 Obrazovka registrovania a aktivovania používateľského účtu



Obrázok 7 - Obrazovka pre registrovanie sa do systému

5.1.4.3 Prihlásenie a odhlásenie sa zo systému

Samozrejme v rámci prezentačnej stránky riešime aj používateľské prihlásenie a odhlásenie sa. Ako neprihlásený používateľ sa chcem prihlásiť do systému aby som získal prístup k manažovaniu svojich vytvorených, resp. pridelených projektov. Zároveň sa dokážem zo svojho účtu odhlásiť. Samotné prihlásenie a odhlásenie sa, sa odohráva na samostatnej podstránke, ktorej prislúcha konkrétna URL.

Pre prihlásenie: /log-in
Pre odlhásenie: /log-out

Pri prihlásení používateľ taktiež vyplňa krátky formulár, ktorý obsahuje nasledovné polia:

- Email
- Heslo

Pri odhlásení sa vymaže konkrétny session používateľa z cake php.

5.1.4.3.1 Akceptačné kritéria

- Kontrola emailu a hesla pri vyplnení jednotlivých polí používateľom
- Používateľ sa dokáže prihlásiť na účet na URL: /log-in
- Používateľ sa dokáže odhlásiť z účtu pomocou URL: /log-out
- Po odhlásení je používateľ presmerovaný na úvodnú stránku

5.1.4.3.2 Validácia

Validácia používateľa sa v tomto prípade vykonáva na nasledujúcich častiach projektu:

- Frontend
- Backend

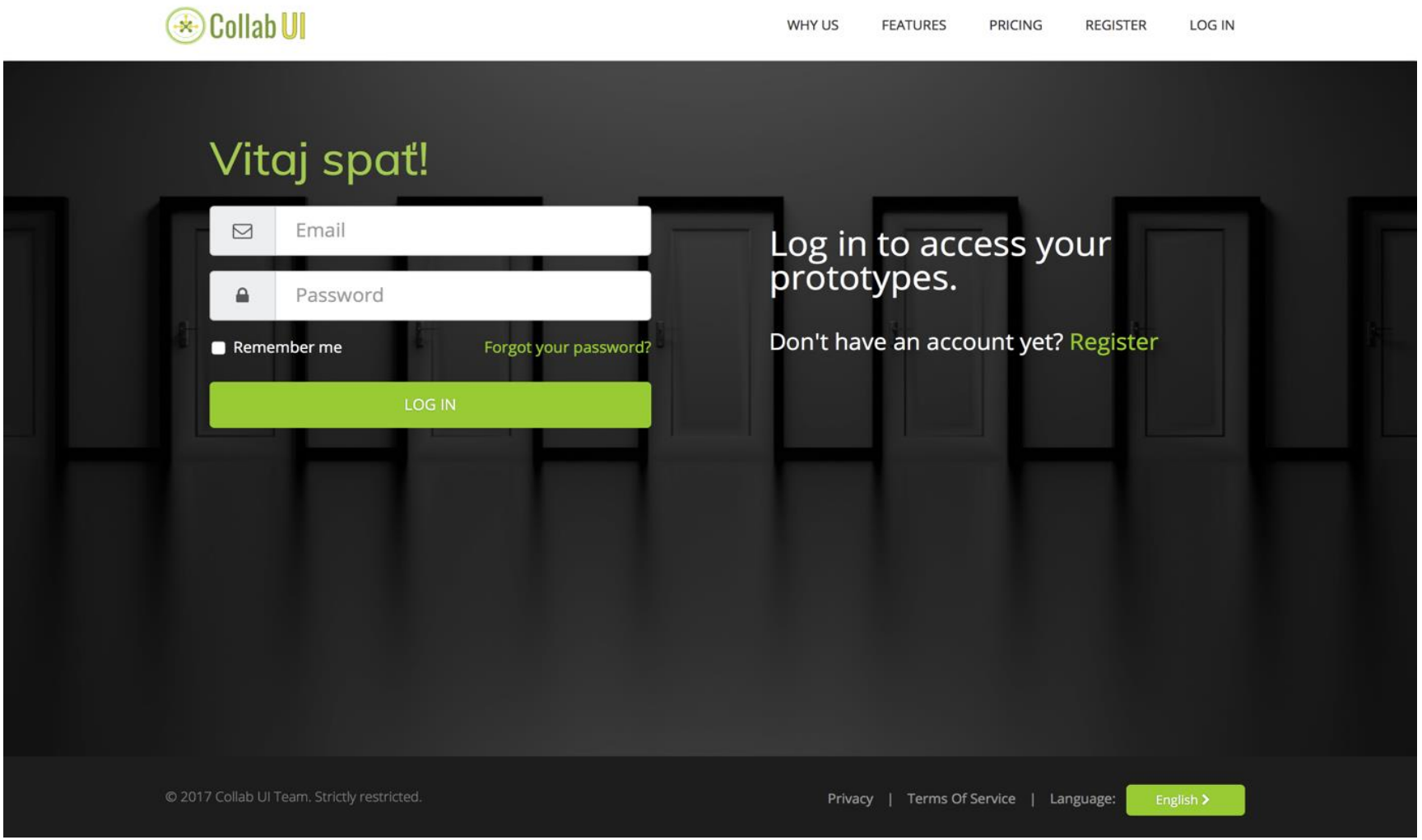
Čo sa týka frontendu, kontroluje sa či jednotlivé textové polia boli vyplnené alebo nie. V prípade, že vyplnené neboli tak sa pri príslušnom polí prostredníctvom animácie zobrazí modálne okno, ktoré používateľa upozorní o nevyplnení daného pola a vyzve ho k jeho vyplneniu.

Čo sa týka backendu, tak sa kontrolujú údaje vložené do jednotlivých polí. Polia sú nasledovné:

- Email
- Heslo

Po vyplnení nasledovných polí a používateľovej akcie v podobe požiadavky na prihlásenie sa sa odošle request, porovnajú sa údaje, ktoré používateľ vpísal do textových polí sa s jeho registračnými údajmi uloženými v databáze. Ak sa údaje zhodujú, tak je používateľ presmerovaný na URL: /dashboard

5.1.4.3.3 Obrazovka pre prihlásenie sa



Obrázok 8 - Obrazovka pre prihlásenie sa do systému

5.1.4.4 Zabudnuté a nové heslo

Ako neprihlásený používateľ chcem mať možnosť v prípade zabudnutia svojho hesla obnoviť prístup do systému aby som mohol pokračovať v manažovaní svojich projektov. K tejto udalosti prislúcha konkrétna URL.

Pre zabudnuté heslo: /forgotten-password
Pre nové heslo: /new-password/HASH

Formulár pre "Zabudnuté heslo" pozostáva z pola:

- Email

ktoré je nutné vyplniť nielen kvôli validácii používateľa ale aj kvôli zaslaniu emailu.
Formulár pre "Nové heslo" pozostáva z pola:

- Heslo

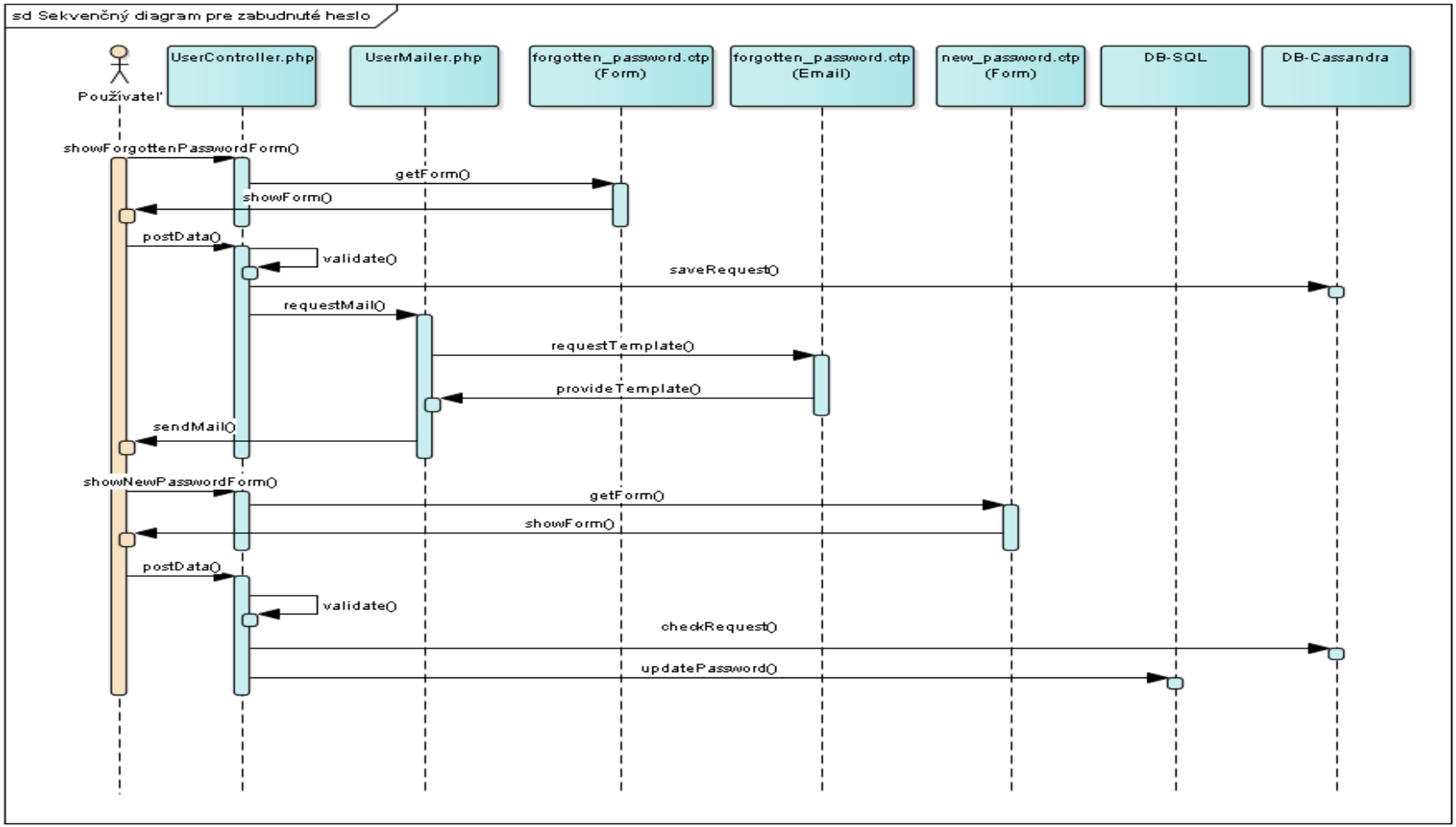
5.1.4.4.1 Akceptačné kritéria

- Po odoslaní formulára "Zabudnuté heslo" dostane používateľ email s URL
- URL presmeruje používateľa na /new-password/HASH
- Po odoslaní formulára "Nové heslo" je používateľ presmerovaný na URL: /log-in s informáciou o zmene jeho hesla
- Po prihlásení s novým heslom sa používateľ dostane na URL: /dashboard

5.1.4.4.2 Validácia

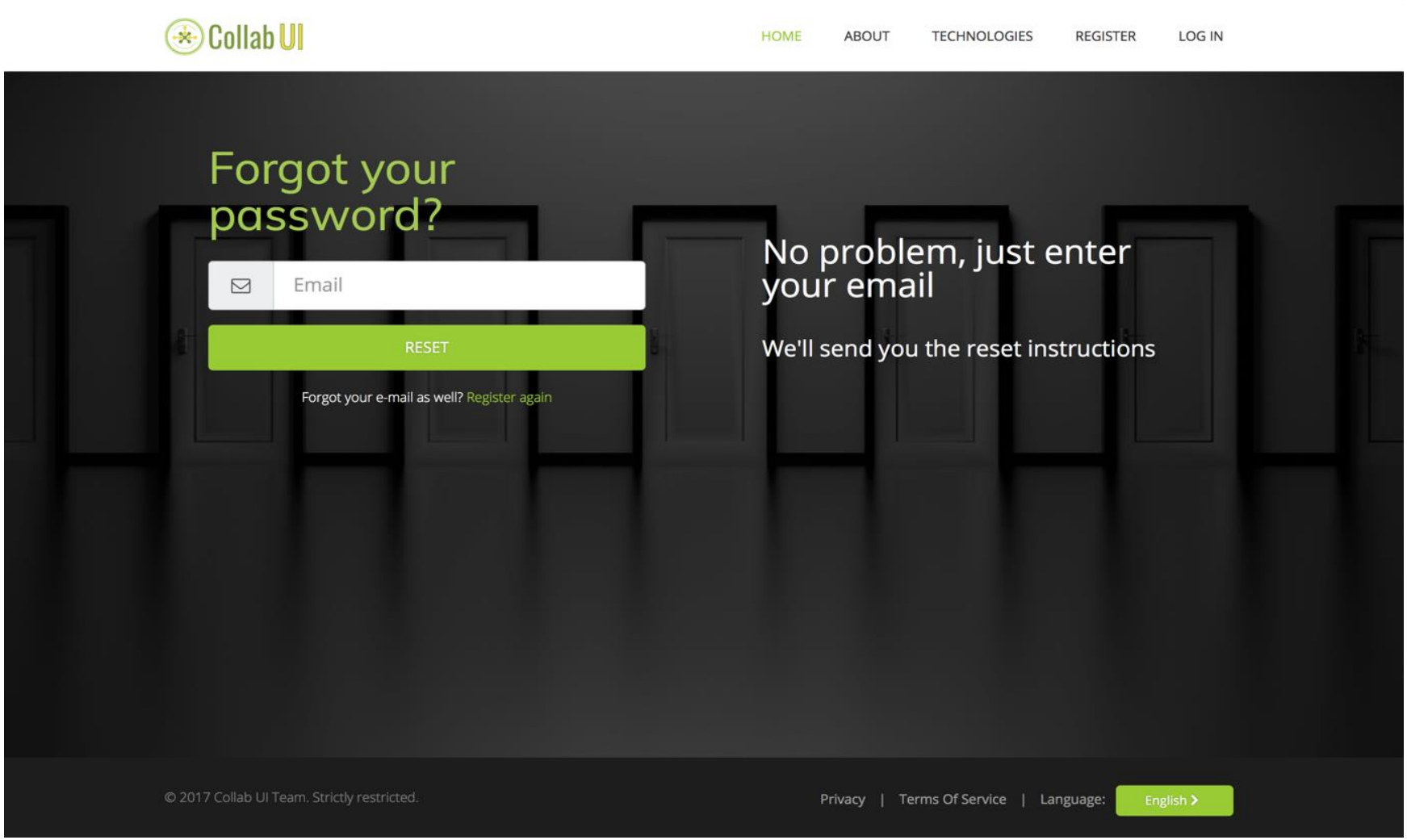
V rámci validácie zabudnutého hesla sa pri nastavovaní hesla kontroluje či je používateľ v systéme zaregistrovaný. Taktiež sa kontroluje email, z ktorého bol používateľ presmerovaný na podstránku zabudnutého hesla.

5.1.4.4.3 Sekvenčný diagram



Obrázok 9 - Sekvenčný diagram pre zabudnuté heslo

5.1.4.4.4 Obrazovka pre zabudnuté heslo



Obrázok 10 - Obrazovka pre obnovu hesla

5.1.5 Testovanie

V tomto šprinte sme testovali manažment používateľov. Testy pre manažment používateľov sa nachádzajú v zložke \tests\TestCase\Controller\UserControllerTest.php

Testovali sa nasledujúce scénare:

- Test presmerovania vo viacerých prípadoch ako presmerovanie pri odoslanie zabudnutého hesla z /password-reset na úvodnú stránku

- Test neexistujúceho emailu v prípade zabudnutého hesla

5.2 Manažment projektov

5.2.1 Úvod

Druhý bod, ktorý sme naplánovali v rámci projektu Collab-ui je samotný manažment projektov. V rámci takéhoto manažmentu sme si prešli vytváraním, editáciou a mazaním projektu. Táto časť je jedna z najzákladnejší častí celkového projektu Collab-ui. Ak sa používateľ rozhodne pre naše kolaboratívne prostredie, bude potrebovať tím, ktorý s ním bude kolaborovať a taktiež projekt, na ktorom budú pracovať. Nato aby vôbec používateľ mohol pozvať svoj tím, potrebuje mať vytvorený projekt, do ktorého bude môcť pozvať jednotlivých členov tímu. Taktiež aby mal používateľ väčšiu mošnosť prispôsobovania si daného projektu potrebuje mať k dispozícii funkciu editovania projektu, aby vedel bližšie špecifikovať názov projektu, spísať opis projektu, ktorý bude mocť dať ostatným členom tímu lepší prehľad o danom projekte. Tak isto sa bude môcť dozvedieť kedy bol projekt vytvorený, vlastník projektu bude môcť nastavovať projekt na aktívny, keď bude v procese vývoja alebo na zavretý keď sú už všetky skutočnosti projektu uzatvorené. Okrem iného bude môcť nastavovať jazyk projektu.

5.2.2 Analýza

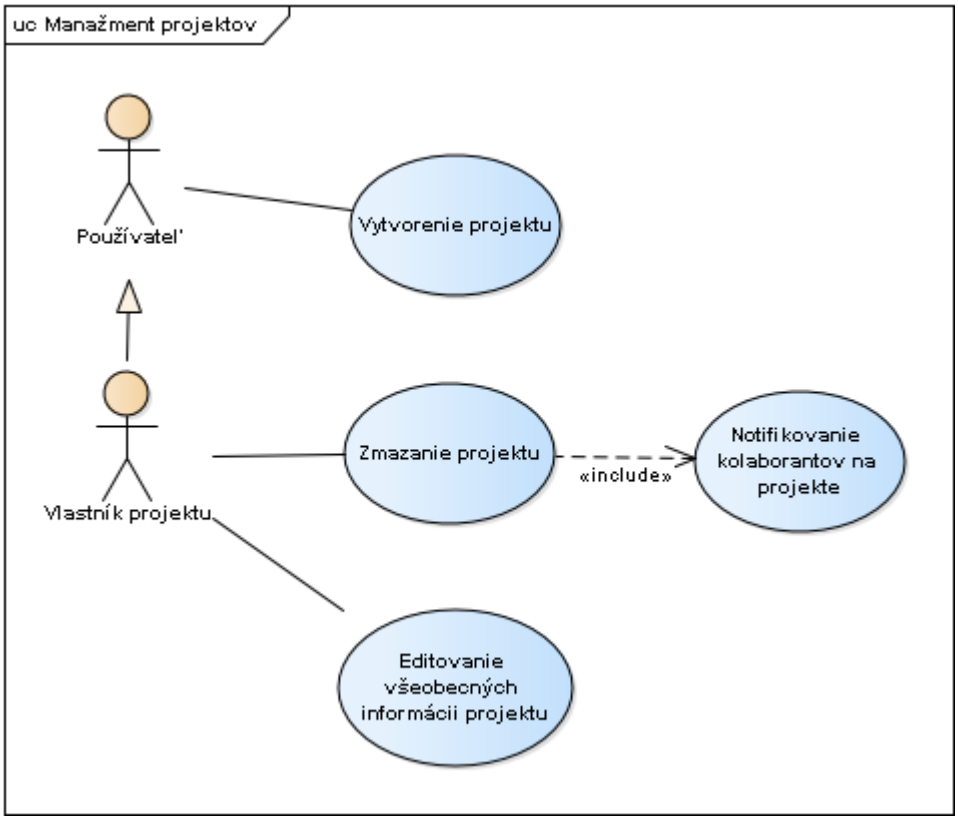
V rámci analýzi sme sa zameriavali rozvrhnutie projektu v rámci jednoduchosti používania a vytvárania projektov používateľom. Veľa krát môže byť pri webových aplikáciach problem práve v rozvrhnutí elementov tejto aplikácie. Ak používateľ nevie intuitivne nájsť riešenie na svoj aktuálny problem, dizajn aplikácie nebude zrovna najlepší. Snažili sme sa teda inšpirovať moderným dizajnom webových aplikácií s jednoduchou myšlienkou. Používateľ po príhlásení hneď uvidí možnosť vytvorenia projektu práve z toho dôvodu, lebo toto by podľa nás mala byť jeho prvá aktivita a teda môžeme očakávať, že tak bude aj konať. Hneď ako vytvorí projekt je presmerovaný na jeho detail, v ktorom by ho mohol upravovať alebo aj vymazať. Každá akcia by mala byť pokrytá istými potvrdeniami v podobe modálnych okien, práve z toho dôvodu aby sa v prípade používateľovej chybnej voľby nezmazal celý projekt. Okrem iného by mal byť informovaný o jednotlivom dianí v aplikácii. Toto riešime pomocou oblúbených notifikácii, na ktoré sú už používatelia v dnešnej dobe zvyknutý a prídu im sympatické častokrát.

5.2.3 Návrh

Manažment projektov sa skladá z nasledujúcich UC:

- Vytvorenie projektu
- Zmazanie projektu
- Editovanie všeobecných informácii projektu
- Notifikovanie kolaborantov na projekte

Nasledujúci use case diagram znázorňuje základný sled akcií používateľa pri vytváraní, mazaní alebo editovaní projektu, ktorý môže vlastniť alebo na ňom kolaborovať.



Obrázok 11 - UC diagram pre manažment projektov

UC Vytvorenie projektu

Hlavný tok:

1. Používateľ navštívil dashboard aplikácie
3. Používateľ zvolil možnosť vytvorenia projektu
4. Systém vytvorí pre používateľa projekt
5. Systém presmeruje používateľa na detail vytvoreného projektu
6. Systém notifikuje používateľa o úspešnej akcii
7. Prípad použitia končí

UC Editovanie všeobecných informácii projektu

Hlavný tok:

1. Vlastník projektu zvolí možnosť “Edit project”
2. Systém prenastaví fixné polia projektu na editovateľné
3. Vlastník projektu upraví jednotlivé polia projektu podľa vlastného uváženia
4. Potvrdí akciu editovania projektu jeho uložením
5. Systém notifikuje používateľa o úspešnej akcii
5. Prípad použitia končí

Alternatívny tok:

4.a1 Vlastník projektu zruší akciu editovania projektu

4.a2 Systém prenastaví informácie o projekte na posledné uložené, teda neuloží vykonané zmeny

4.a3 Pokračuje bodom 5

UC Zmazanie projektu

Hlavný tok:

1. Vlastník projektu zvolí možnosť zmazania projektu (musí sa nachádzať na detaile projektu)

2. Systém ho upozorní o určitosti jeho akcie prostredníctvom modálneho okna
3. Vlastník projektu potvrdí akciu zmazania projektu
4. Systém zmaže všetky verzie projektu
5. Vlastník projektu je presmerovaný na základný dashboard aplikácie
6. Pokračuje s UC Notifikovanie kolaborantov na projekte

Alternatívny tok:

3.a1 Vlastník projektu zruší akciu zmazania projektu

3.a2 Systém zatvorí modálne okno, čím zruší akciu zmazania

3.a3 Pokračuje s bodom 6

UC Notifikovanie kolaborantov na projekte

Hlavný tok:

1. Systém notifikuje jednotlivých kolaborantov, ktorí pracovali na projekte, prostredníctvom mailu, v ktorom ich oboznámi so skutočnosťou zmazania projektu

2. Prípad použitia končí

5.2.4 Implementácia

V rámci manažmentu projektov sme sa rozhodli využiť niektoré ďalšie technológie. Jednými z nich je:

- Aoweb[1]
- Crud plugin pre CakePHP[2]
- Bootstrap notify[3]
- Bootstrap modal[4]

Princíp "scaffolding" je technika, ktorá umožňuje vývojárom definovať a vytvoriť základnú aplikáciu, ktorá dokáže vytvárať, načítať, aktualizovať a mazať objekty. V CakePHP tiež umožňuje vývojárom definovať, ako sú jednotlivé objekty navzájom prepojené a vytvárať ako aj prerušovať tieto prepojenia. Táto technika sa používa práve v Crud plugine pre CakePHP. Rozhodli sme sa teda použiť tento plugin práve kvôli viac flexibilnejšiemu prístupu a rýchlejšiemu vytváraniu prototypov s rovnakým základom.

AOWeb je aspektovo-orientovaný rámec, implementovaný v jazyku JavaScript, ktorý pomáha v oddelení záležitostí a písaní prehladných frontend modelov. Disponuje s veľkým množstvom užitočných funkcionalít, ktoré zjednodušujú a uľahčujú prácu programatora. AOWeb je semestrálny project AOVS a bol implementovaný jedným členom nášho tímu.

Bootstrap používame už od samého začiatku, práve preto sme si povedali, že budeme využívať aj jeho funkcie:

- Notify
- Modal

Práve tieto funkcie nám budú pomáhať upozorňovať používateľa na rôzne diania aplikácie prostredníctvom notifikácii ako aj pomáhať potvrdiť používateľove akcie cez modálne okná.

5.2.4.1 Vytvorenie projektu

Ako prihlásený používateľ chcem vedieť vytvoriť nový projekt vo verzii 1.0.0
Vytváranie nového projektu sa vykonáva na nasledovnej URL: /dashboard

Zoznam vytvorených a pridelených projektov sa nachádza na nasledovnej URL: /api/project/(project-id)

---

[1] https://github.com/Ado20/aoweb
[2] https://github.com/FriendsOfCake/crud
[3] http://bootstrap-notify.remabledesigns.com/
[4] https://v4-alpha.getbootstrap.com/components/modal/

5.2.4.1.1 Akceptačné kritéria

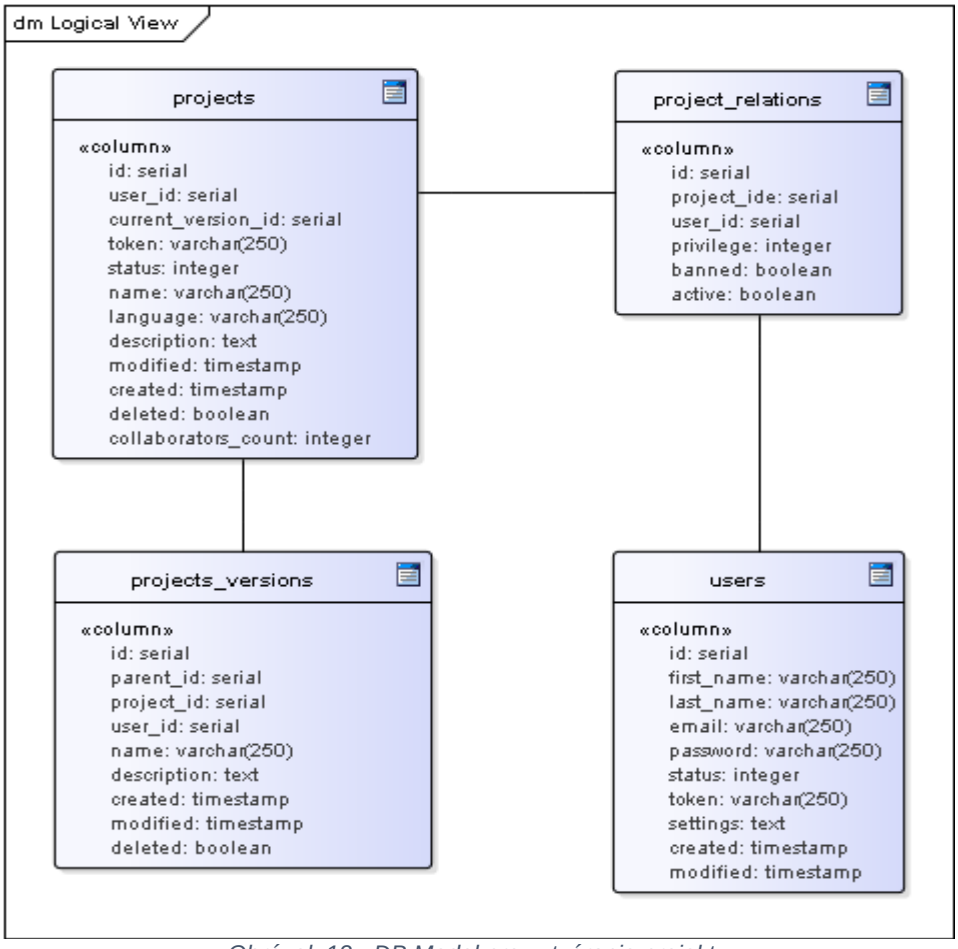
- Používateľ dokáža vytvoriť nový projekt vo verzii 1.0.0
- Po vytvorení projektu je presmerovaný na detail projektu
- Používateľ dokáže zmeniť základné informácie o projekte

5.2.4.1.2 Validácia

Vytvorenie projektu kontroluje len základné dáta a ich správnosť. Taktiež kontroluje, či sú tieto dáta poslané v požiadavke.

5.2.4.1.3 Databázový model

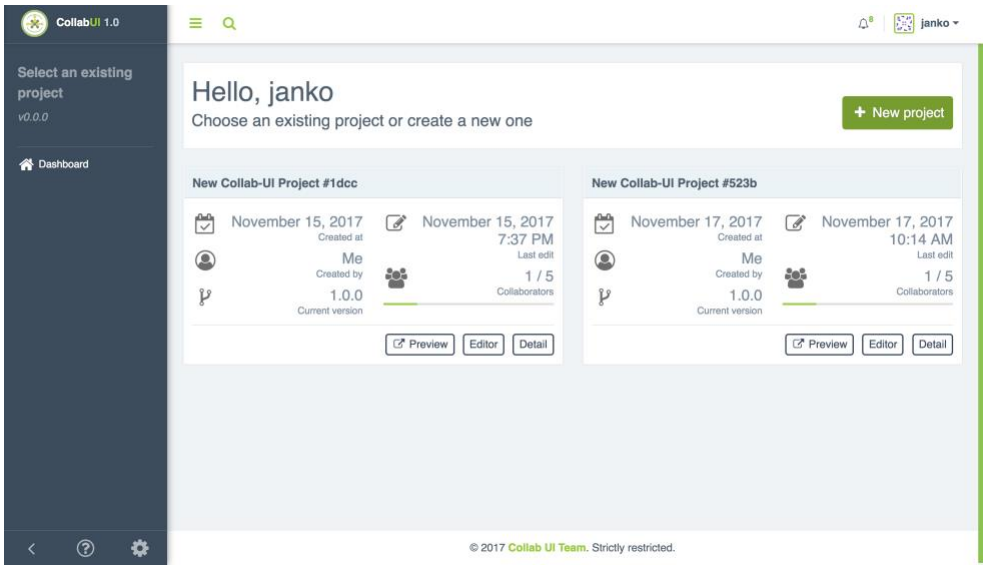
Čo sa týka databázového pohľadu, sa pri vytváraní projektu používajú nasledovné tabuľky.

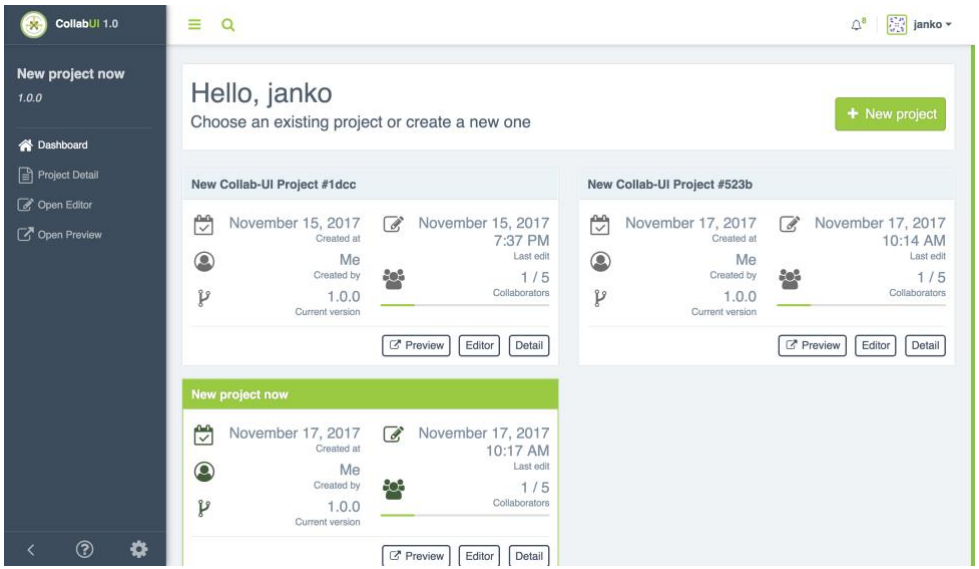


Obrázok 12 - DB Model pre vytváranie projektu

Pri vytváraní projektu sa vlastník daného projektu detekuje pomocou user_id.

5.2.4.1.4 Obrazovky dashboardu





Na dashboarde môžeme vidieť náhľad do dashboardu aplikácie. V tomto konkrétnom príklade máme vytvorené 2 projekty. Chceme vytvoriť ďalší tak prejdeme na možnosť “New project”.

Po zvolení tejto možnosti sa nám vytvorí nový projekt so základnou šablónou, v ktorej je prednastavený:

- dátum vytvorenia projektu
- používateľ, ktorý projekt vytvoril
- verzia projektu
- kedy bol projekt naposledy upravovaný
- počet kolaborantov

Obrázok 13 - Obrazovka pre vytvorenie projektu

5.2.4.2 Zmazanie projektu

Cieľom tohto user story bolo vytvorenie možnosti zmazania už vytvoreného projektu jeho vlastníkom. Zmazaný bude mocť byť iba projekt, ktorý bol vytvorený jeho vlastníkom. Mazanie nebude sprístupnené pre používateľov, ktorí niesú jeho vlastníkmi.

Taktiež ak príde k situácii, že vlastník projektu zmaže ním vytvorený projekt, aby sme predišli jednotlivým nedorozumeniam v rámci jeho tímu a kolaborantov, ktorí na projekte pracovali, sa po zmazaní projekt títo používatelia notifikujú prostredníctvom emailu, ktorý im oznámi, že už ďalej niesú sučasťou daného projektu z dôvodu jeho zmazania.

Po vymazaní projektu sa projekt (a jeho relations) reálne nezmaže ale sa len označí v DB ako deleted pritom User relation zostáva.

URL pre zmazaný projekt: /dashboard/deleted-project

5.2.4.2.1 Akceptačné kritéria

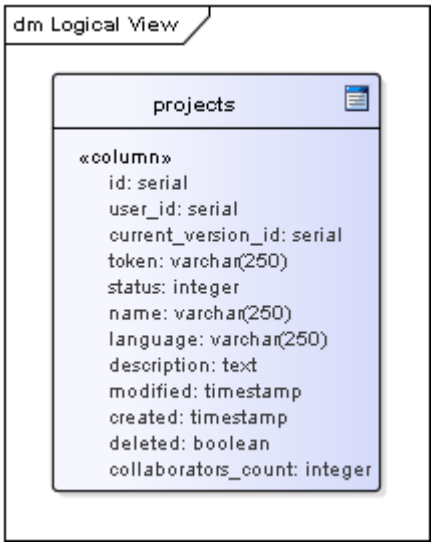
- Odstránenie projektu musím najskôr potvrdiť
- Dokáže používateľ odstrániť projekt
- Po odstránení bude presmerovaný na Dashboard a notifikovaný o úspešnej operácii
- Po vymazaní projektu sa projekt (a jeho relations) reálne nezmaže ale sa len označí v DB ako deleted.
- Po stlačení tlačidla musí byť používateľ notifikovaný o tom, že bude projekt definitívne zmazaný a že moji koaboratni stratia prístup k projektu (modálne okno).

5.2.4.2.2 Validácia

Pri vymazaní projektu sa okrem základných validácií nachádza aj validácia používateľa, ktorý chce vymazanie vykonať, pretože na takúto požiadavku je potrebné, aby bol používateľ zároveň aj vlastníkom projektu.

5.2.4.2.3 Databázový model

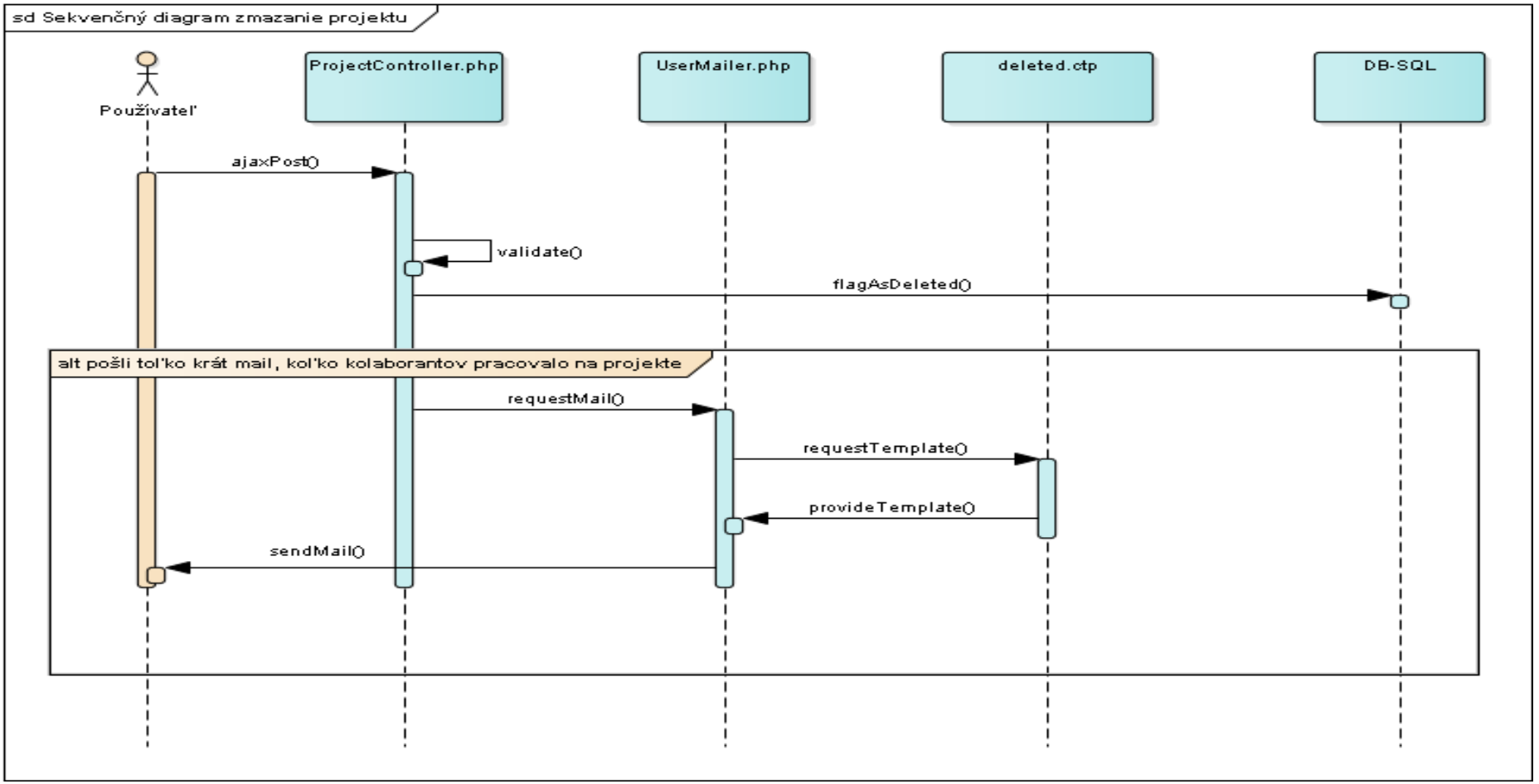
Čo sa týka databázového pohľadu, sa pri mazaní projektu používa iba tabuľka projects, v ktorej sa nastaví flag pre mazanie projektu “deleted” na true. User relation v tabuľke UserRelations sa ale nezmaže.



Obrázok 14 – V rámci mazania projektov sa používa projects tabuľka

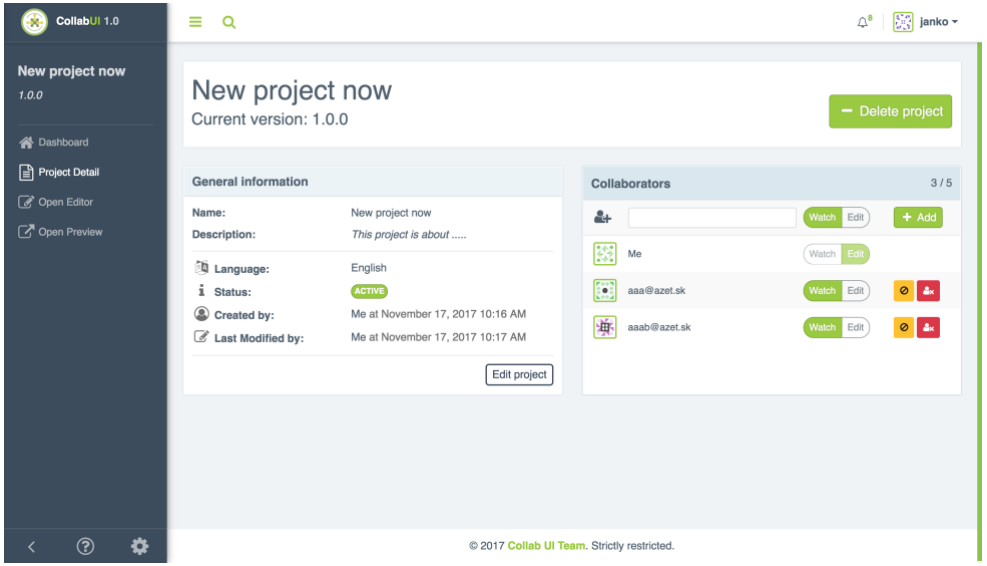
V rámci budúcej práce by sme chceli zaintegrovať databázový cron, ktorý by každý deň o polnoci prešiel všetky záznamy, záznamy s flagom deleted by zarchivoval a v poslednom kroku vymazal z databázy.

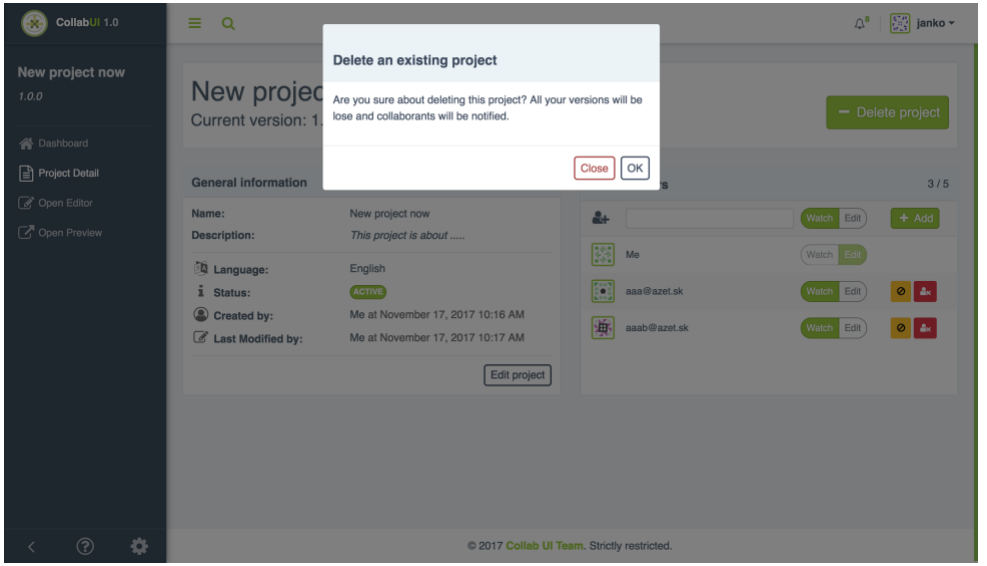
5.2.4.2.4 Sekvenčný diagram



Obrázok 15 - Sekvenčný diagram pre mazanie projektu

5.2.4.2.5 Obrazovka pre mazanie projektu





Na konkrétnej obrazovke sa používateľ nachádza na detaile projektu, na ktorom môže vidieť jednotlivé informácie o projekte a môže vykonávať jednotlivé akcie späté s manažmentom projektu:

- Zmazanie projektu
- Editovanie projektu

Po zvolení možnosti zmazania projektu nám vyskočí modálne okno s otázkou či sme si naozaj istý, že chceme odstrániť daný projekt, lebo s ním budú odstránené aj všetky jeho verzie. Taktiež informuje o notifikovaní kolaborantov o zmazaní projektu. Taktiež dáva používateľovi možnosť potvrdiť akciu alebo zrušiť akciu.

Obrázok 16 - Obrazovka pre mazanie projektu

5.2.4.3 Editovanie všeobecných informácii projektu

Samotné editovanie projektu je najdôležitejšou časťou manažmentu projektov. Je to z toho hladiska, že ak používateľ vytvorí projekt, automaticky sa nastaví na základnú šablónu. Kebyže projekt nemôžeme editovať tak by boli všetky projekty so základnou šablónou a používatelia by sa náročne orientovali medzi nimi. V rámci editovania projektu je možné meniť nasledovné informácie o projekte:

- Názov
- Popis
- Jazyk
- Stav

Čo sa týka jazykov zatiaľ sú k dispozícii:

- Slovenčina
- Angličtina
- Maďarčina
- Nemčina
- Čeština

Status môže nadobúdať nasledovné stavy:

- Aktívny
- Zatvorený

V rámci editovania sme rozdelili detail projektu na jednotlivé boxy:

- Box pre projekt
- Box pre manažment kolaborantov

V rámci tohto rozdelenia sme použili už spomínaný framework aoweb, ktorý použitím funkcií “separation of concerns” slúži na rozdelenie kontextu a na komunikáciu jednotlivých submodulov.

5.2.4.3.1 Akceptačné kritéria

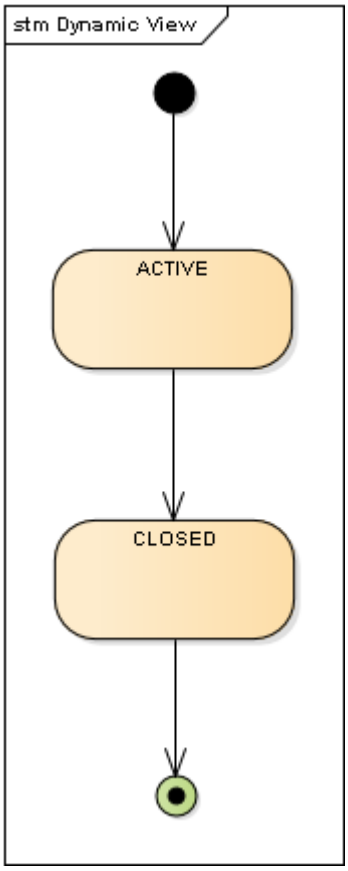
- Kontrola emailu a hesla pri vyplnení jednotlivých polí používateľom
- Možnosť nastavenia statusu na aktívny alebo zatvorený
- Nastavenie viacerých jazykov
- Rozdelenie projektu do samostatného boxu
- Uloženie zmien v projekte po potvrdení akcie
- Pri zrušení akcie neuložiť vyplnené polia ale ponechať pôvodné informácie

5.2.4.3.2 Validácia

V rámci validácie používateľa je v tomto prípade iba validácia vyplnenia polí na detaile projektu. Teda ak používateľ zabudol vyplniť niektoré z textových polí, tak ho systém pomocou notifikácie upozorní na túto skutočnosť.

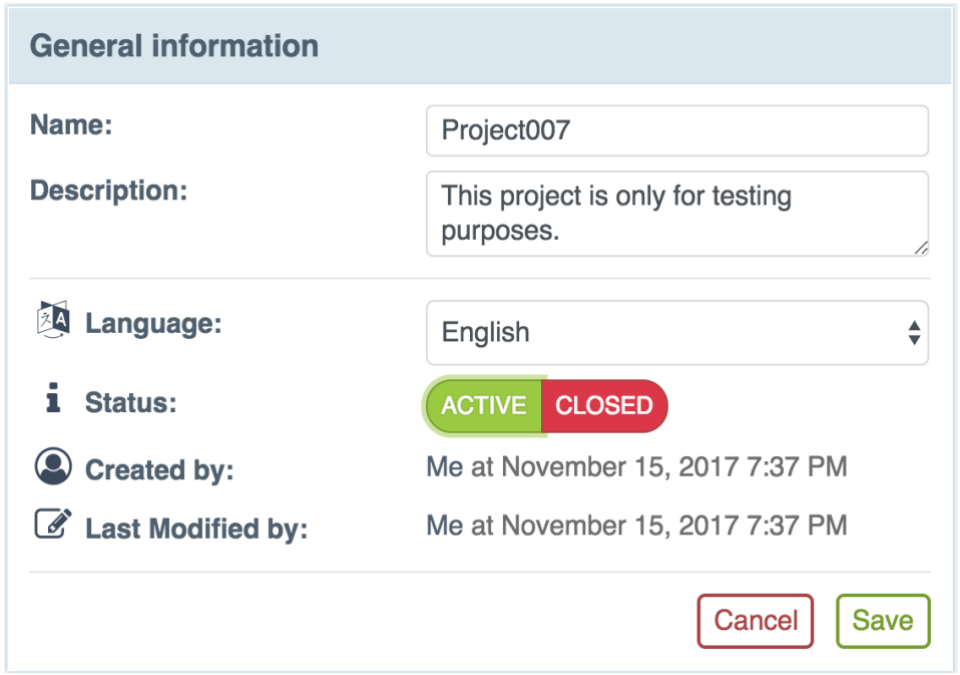
Taktiež sa okrem základných validácií nachádza aj validácia používateľa, ktorý chce úpravu vykonať, pretože na takúto požiadavku je potrebné, aby bol používateľ zároveň aj vlastníkom projektu.

5.2.4.3.3 Stavový diagram – projekt



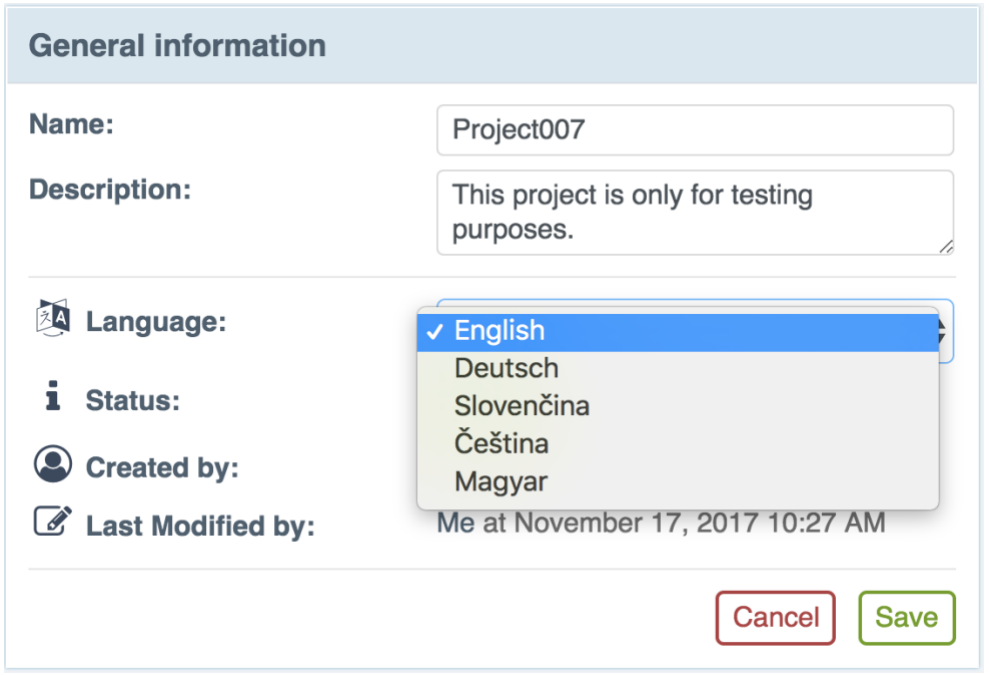
Obrázok 17 - Stavový diagram pre projekt

5.2.4.3.4 Obrazovka pre editovanie všeobecných informácii projektu



V rámci boxu pre samotný projekt máme už spomínané jednotlivé informácie. Taktiež môžeme nastavovať status, jazyk a upravovať textové polia. Akciami, ktorými používateľ disponuje v tomto prípade sú akcie:

- Uložiť
- Zrušiť

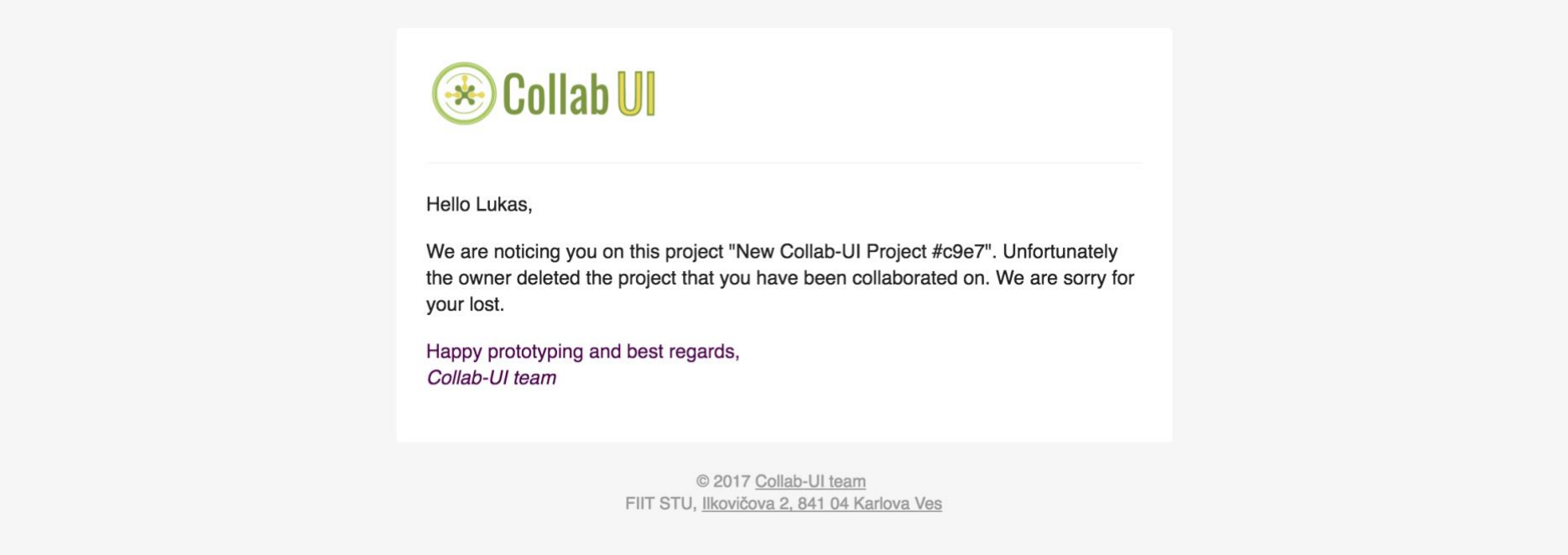


V tejto ukážke môžeme vidieť vyrolované okno pre jazyky v ktorom si používateľ môže príslušný jazyk nastaviť.

Obrázok 18 - Obrazovka pre editovanie informácii projektu

5.2.4.4 Notifikovanie kolaborantov na projekte

V rámci notifikácie kolaborantov sme spomenuli jednotlivé skutočnosti už viac krát aj v rámci vytvárania projektov a mazania projektov. Práve preto prikladáme ešte dodatočnú obrazovku pre ukážku šablóny emailu pre notifikácie.



Obrázok 19 – Obrazovka prenotifikovanie kolaborantov projektu

5.2.5 Testovanie

Testovacie v rámci manažmentu projektov zahŕňa:

- API TEST pre vytvorenie projektu
- API TEST pre zmazanie projektu
- API TEST pre editovanie projektu

5.3 Manažment kolaborantov

5.3.1 Úvod

Tretí bod, ktorý sme naplánovali v rámci projektu Collab-ui je manažment kolaborantov projektu. Samozrejme keď používateľ vytvorí projekt, väčšinou na ňom nechce robiť sám. Práve preto potrebujeme istým spôsobom ponúknuť používateľovi manažovať kolaborantov, ktorých by chcel aby pracovali na danom projekte. Používateľ si teda bude mocť prizvať kolaboranta prostredníctvom emailu, na ktorý mu príde oznámenie o jeho pozvaní do konkrétneho projektu a ďalej to už záleží natom či kolaborant prijme pozvanie alebo nie. Taktiež by sme mali vedieť spravovať kolaborantov, ktorí môžu istým škodlivým spôsobom prispievať do daného projektu. V takom prípade má vlastník projektu možnosť kolaboranta zabanovať. Ten síce o tom nebude vedieť lebo emailová notifikácia mu nepríde, zato môže sa to dozvedieť tým, že sa mu môžu zmeniť práva k spomínanému projektu. Teda už nebude môcť ďalej do projektu zasahovať. Okrem iného kolaboranti sa časom môžu meniť, preto by vlastník projektu mal mať okrem pridania kolaboranta, možnosť aj odobrat kolaboranta, ktorý už na projekte pracuje. Môže sa stať, že skončil vo firme alebo bol prehodený na iný projekt a tým pádom by nedávalo zmysel aby ďalej zostával na danom projekte.

5.3.2 Analýza

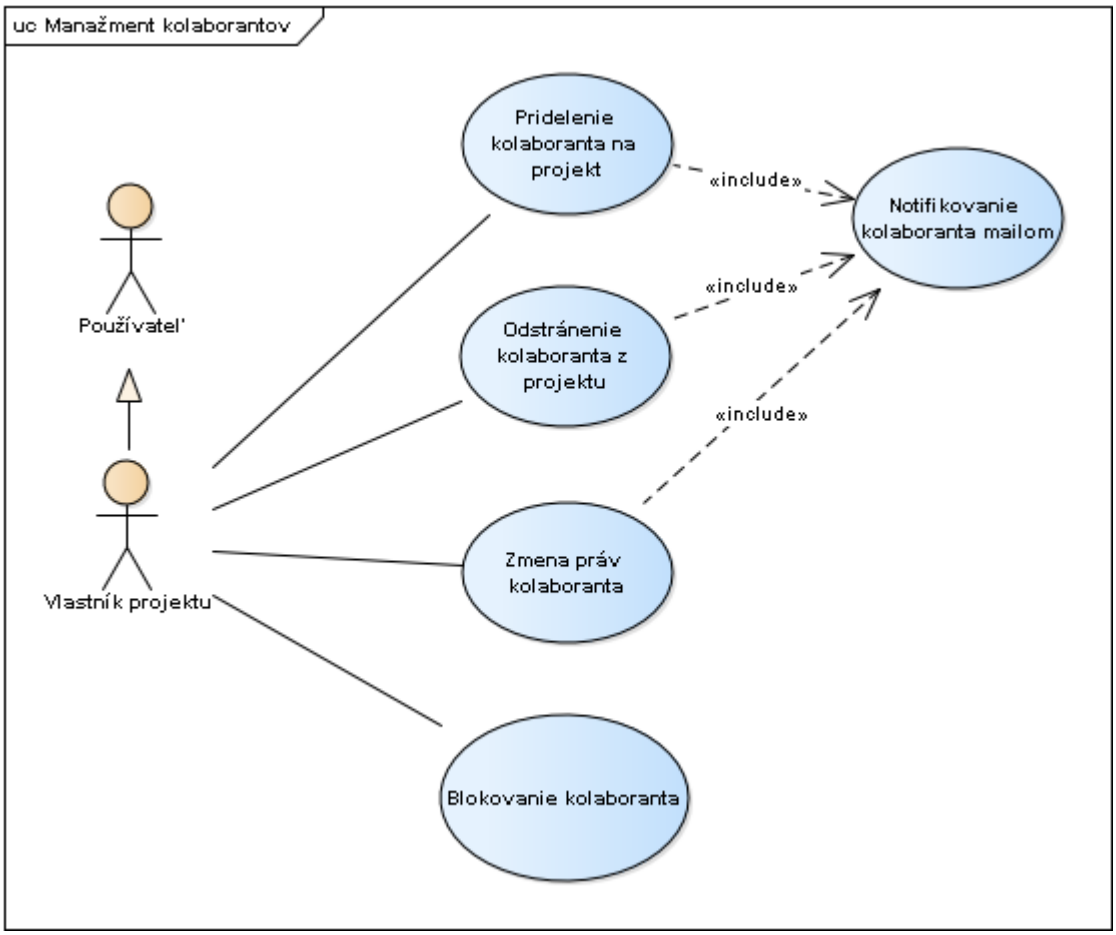
V rámci analýzi sme riešili hlavne akým štýlom bude daný kolaborant pridaný do projektu. Stále sme sa snažili dodržiavať nezložité grafické prevedenie s očakávajúcou funkcionalitou. Rozhodli sme sa preto rozdeliť kolaborantov a projekty do osobitných boxov. Tento fakt sme už spomínali aj pri vytváraní projektov. Tu ho spomíname znovu preto, lebo práve aj z časti kvôli vyriešeniu spôsobu pridávania kolaborantov sme učinili spomínane zmeny. V detaile projektu sa teda nachádza box pre projekt a vedľa neho ďalší box pre kolaborantov k danému projektu. Pri pridávaní kolaboranta by vlastníkovi projektu mohlo ponúkať členov tímu, ktorých už niekedy pridával. Všetky akcie ktoré sa týkajú správy kolaborantov by sa mohli vykonávať v spomínanom boxe. Teda aj zabanovanie aj odstránenie by sa mohlo nachádzať pri každom kolaborantovi aby používateľ mohol rýchlo rozhodnúť o jeho budúcej aktivite. Taktiež používanie modálnych okien a notifikácii ako aj pri vytváraní projektov bude zaiplmenetované aj v tejto časti.

5.3.3 Návrh

Manažment kolaborantov sa skladá z nasledujúcich UC:

- Pridelenie kolaboranta na projekt
- Odstránenie kolaboranta z projektu
- Zmena práv kolaboranta
- Blokovanie kolaboranta
- Notifikovanie kolaboranta mailom

Nasledujúci use case diagram znázorňuje základný sled akcií vlastníka projektu pri správe jednotlivých kolaborantov pracujúcich na projekte.



Obrázok 20 - UC diagram pre manažment kolaborantov

UC Pridelenie kolaboranta na projekt

Hlavný tok:

1. V boxe pre kolaborantov vlastník projektu vyhľadá kolaboranta podľa emailovej adresy

2. Systém mu ponúkne po zadaní prvých 3 písmen zoznam možných kolaborantov pre pridanie do projektu

3. Vlastník projektu vyberie zo zoznamu konkrétneho kolaboranta

4. Nastaví kolaborantovi práva na iba sledovanie alebo aj editovanie projektu

5. Potvrdí akciu pridania kolaboranta
6. Systém ho pridá do zoznamu spolupracujúcich kolaborantov na danom projekte
7. Systém notifikuje vlastníka projektu o úspešnej operácii
8. Pokračuje s UC Notifikovanie kolaboranta

UC Odstránenie kolaboranta z projektu

Hlavný tok:

1. Vlastník projektu zvolí akciu odstránenia kolaboranta z projektu
2. Systém sa prostredníctvom modálneho okna uistí o jeho akcii
3. Vlastník projektu potvrdí akciu odstránenia kolaboranta
4. Systém notifikuje vlastníka projektu o úspešnej operácii
5. Pokračuje s UC Notifikovanie kolaboranta

UC Zmena práv kolaboranta

Hlavný tok:

1. Vlastník projektu iniciuje zmenu práv kolaboranta
2. Systém zmení práva kolaborantovi
3. Systém notifikuje vlastníka o úspešnej operácii
4. Pokračuje s UC Notifikovanie kolaboranta

UC Blokovanie kolaboranta

Hlavný tok:

1. Vlastník projektu zvolí možnosť zablokovania kolaboranta
2. Systém zablokuje kolaboranta

3. Systém notifikuje vlastníka projektu o úspešnej akcii a zmení grafický prvok blokovania používateľa

4. Prípad použitia končí

Alternatívny tok:

1.a1 Vlastník projektu zvolí možnosť odblokovania kolaboranta

1.a2 Systém odblokuje kolaboranta

1.a3 Pokračuje s bodom 3

UC Notifikovanie kolaboranta mailom

Hlavný tok:

1. Systém notifikuje vybraného kolaboranta prostredníctvom emailu

2. Prípad použitia končí

5.3.4 Implementácia

5.3.4.1 Pridelenie kolaboranta na projekt

Pridelenie kolaboranta je funkcia, ktorú bude potrebovavť každý vlastník projektu, pokiaľ nechce pracovať sám. Kolaborant môže byť

- registrovaný
- neregistrovaný

do projektu.

Ak je používateľ neregistrovaný pri jeho pridaní sa pridá do databázy tabuľky users so statusom “invited”. Taktiež je notifikovaný emailom, v ktorom nájde URL prostredníctvom, ktorej dokáže pristupovať do editora a teda nepotrebuje mať vôbec vytvorený účet u nás.

Na pridelenie kolaboranta do projektu využívame jQuery UI Autocomplete s kombináciou technológie KnockoutJS Custom Bindings. Prostredníctvom ajaxových volaní sa klient dopytuje na server, kde si po zadaní prvých 3 znakov, pýta používateľov. Zároveň sú používatelia zoradení podľa ich vzťahu s vlastníkom projektu.

Ak vlastník projektu už niekedy pridával používateľa do niektorého z jeho projektov tak sa mu tento používateľ ukáže na prvých pozíciách. V prípade, že ho ešte nikdy do žiadneho projektu nepridával, tak takýto používateľ sa ukáže v zozname až po tých používateľoch, ktorých už vlastník projektu niekedy pridával do projektov.

5.3.4.2 Akceptačné kritéria

- Možnosť vyhľadania používateľského emailu
- Používateľ dokáže pridať kolaboranta
- Kolaborant sa po pridaní zobrazí v zozname a zároveň mu príde e-mail s URL k projektu
- Kolaborant vidí po prihlásení do systému pridelený projekt

5.3.4.3 Validácia

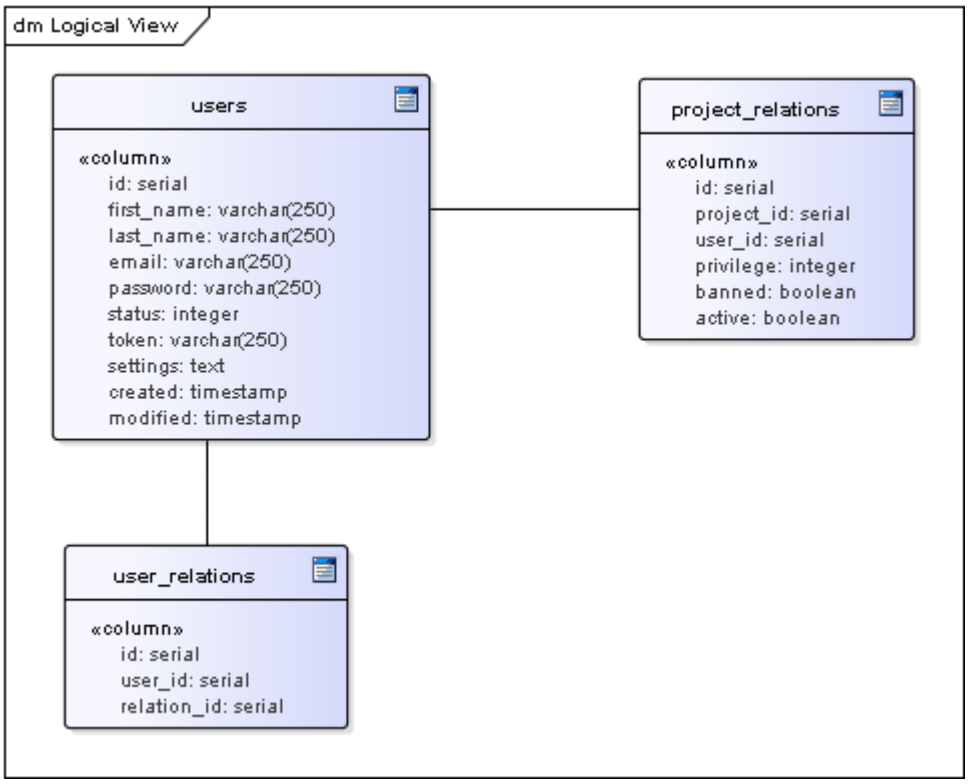
Celá validácia pridania nového kolaboranta do projektu prebieha na backende.

Ako prvé sa kontroluje existencia zadaného emailu v systéme, či sa jedná o registrovaného používateľa alebo nie, ak sa používateľ zo zadaným emailom nenájde, tak sa v systéme vytvorý nový používateľ. Takto vytvorený používateľ je špeciálneho typu "invited", pričom pri jeho vytvorení sa validuje len emailová adresa.

Na druhej strane, ak sa pri pridaní kolaboranta zadaný email nachádza v systéme, tak sa pokračuje až validáciou v ProjectrelationsTable.php, ktorá validuje vytváraný vzťah medzi používateľom a projektom. V tejto časti validácie sa nachádzajú okrem základných kontrôl (existencie potrebných parametrov) aj dve ďalšie validácie a to kontrola, či zadaný používateľ sa v projekte už nenachádza a či sa na projekte nepodieľa maximálny počet kolaborantov.

5.3.4.4 Databázový model

V rámci pridelenia kolaboranta na projekt sa využívajú nasledovné tabuľky.



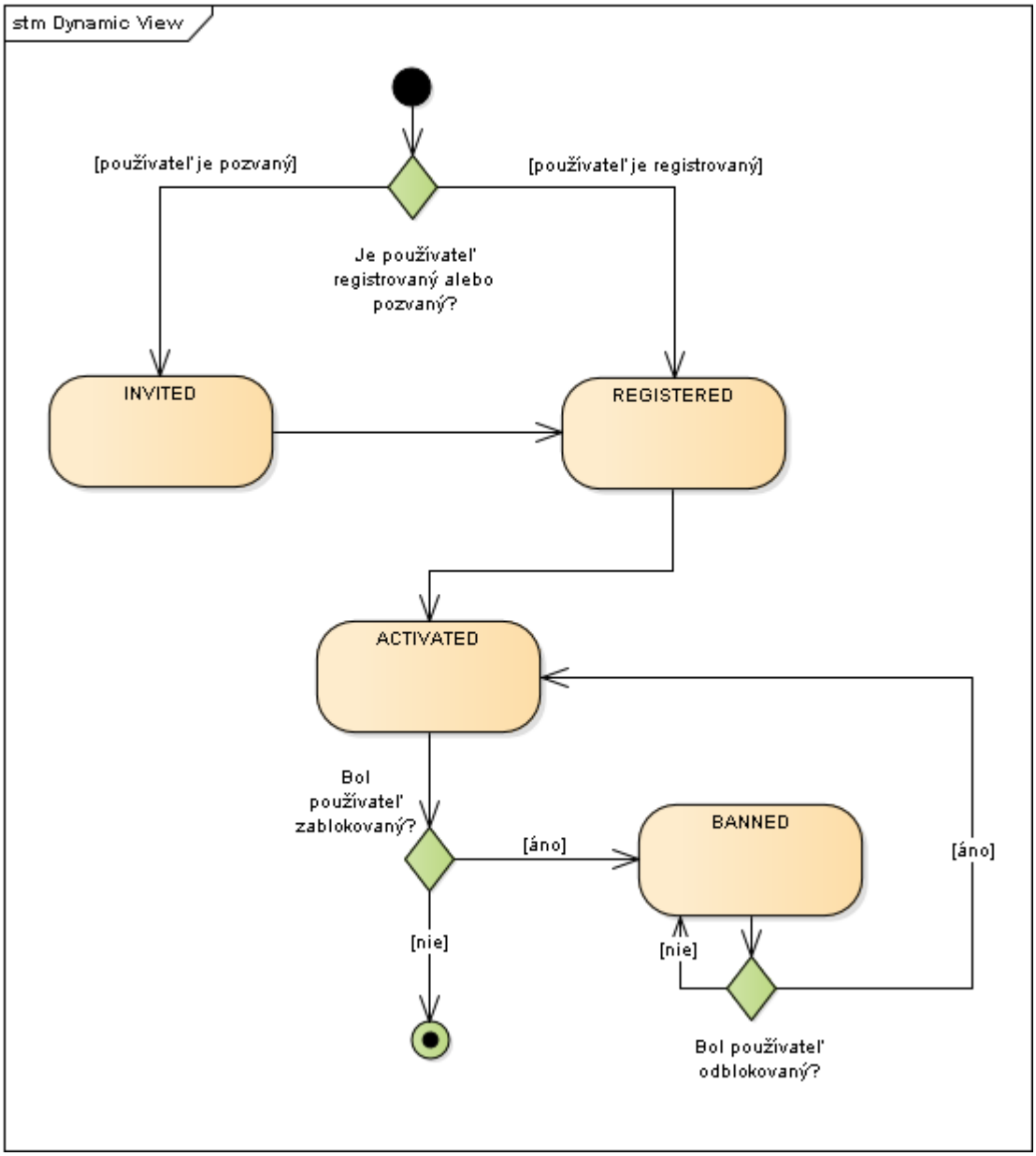
Obrázok 21 - DB Model pre pridelenie kolaboranta na projekt

V tomto prípade nám tabuľka user_relations slúži na uprednostňovanie jednotlivých používateľov vo funkcii automatického dopĺňania.

Project_relations slúži nato aby používateľ vedel nájsť svoje pridelené projekty, v ktorých ma rolu kolaboranta.

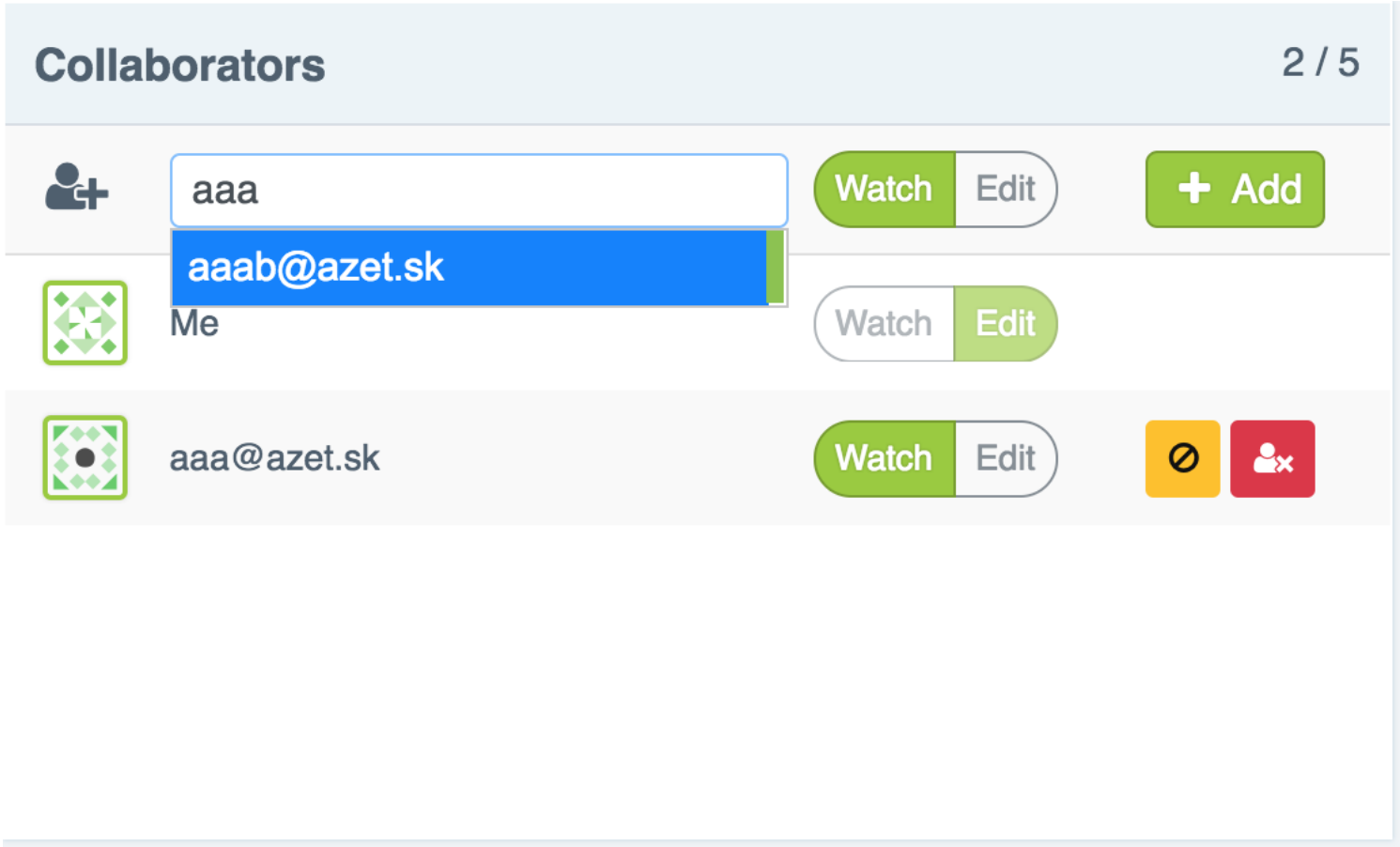
Ak vlastník projektu pridá používateľa, ktorý ešte nemá v systéme zaregistrovaný email, nastaví sa pre používateľa v tabuľke users status na “invited”.

5.3.4.5 Stavový diagram pre status používateľa



Obrázok 22 - Stavový diagram pre status používateľa

5.3.4.6 Obrazovka pre pridelenie kolaboranta na projekt



Obrázok 23 - Obrazovka pre pridelenie kolaboranta

5.3.5 Zmena práv kolaboranta

Zmenu práv môže výkonávať vlastník projektu. Môže kolaborantom nastaviť nasledujúce stavy:

- Watch
- Edit

Keď je používateľ v stave “Watch” tak môže projekt iba sledovať, teda nemôže sa svojvoľne zapájať do diania čo sa týka projektu. Pokiaľ dostane práva “Edit” tak môže upravovať projekt tak isto ako ktorýkoľvek iný kolaborant.

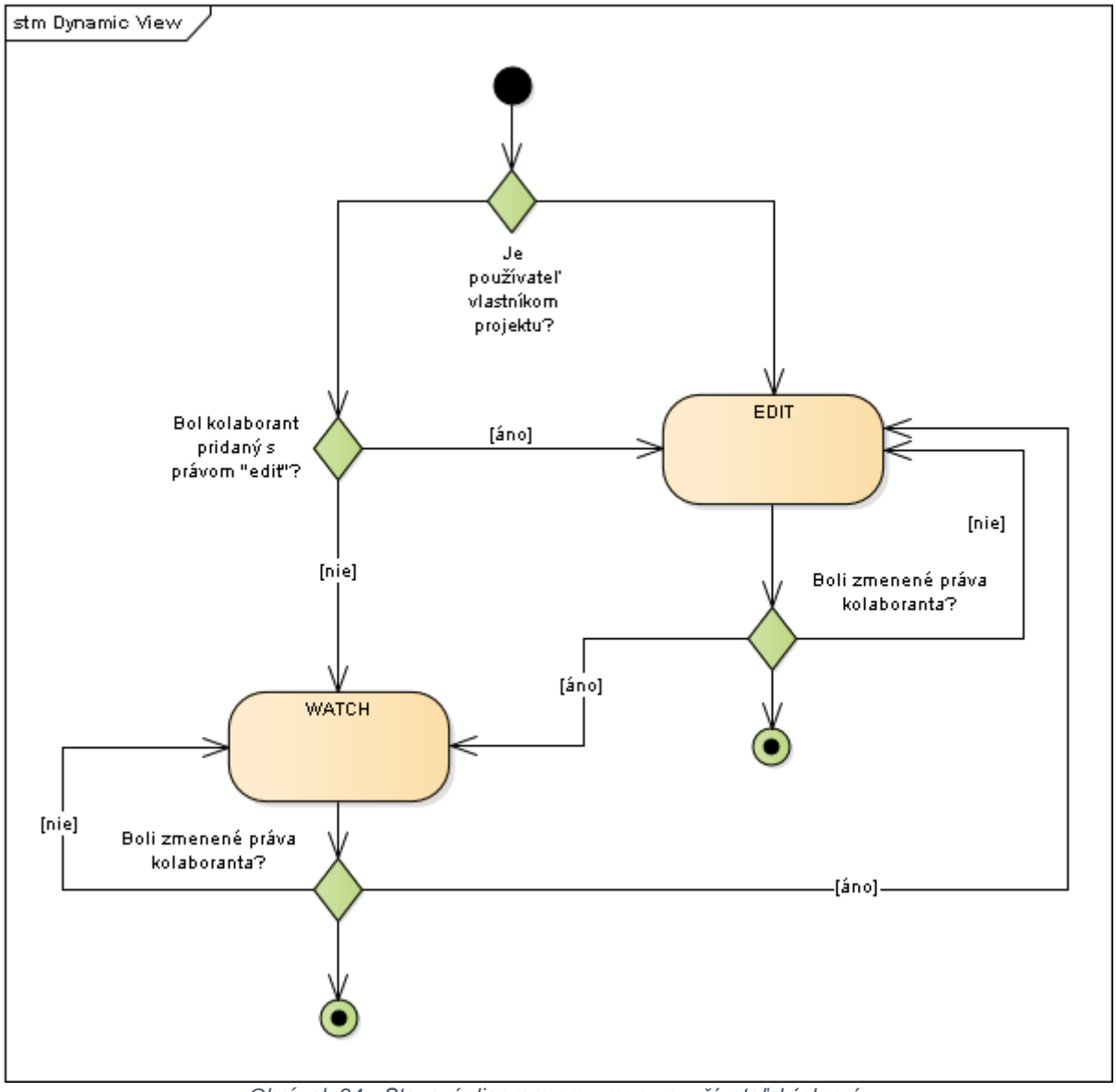
5.3.5.1 Akceptačné kritéria

- Možnosť odobrať editovacie práva kolaborantovi, teda nastaviť mu práva na sledovanie
- Možnosť priradiť editovacie práva používateľovi, teda odobrať mu práva na sledovanie
- Pridelenie základného práva pri pridávaní kolaboranta do projektu
- Vlastník projektu má automaticky editovacie práva
- Notifikovanie vlastníka projektu o zmene práv kolaboranta

5.3.5.2 Validácia

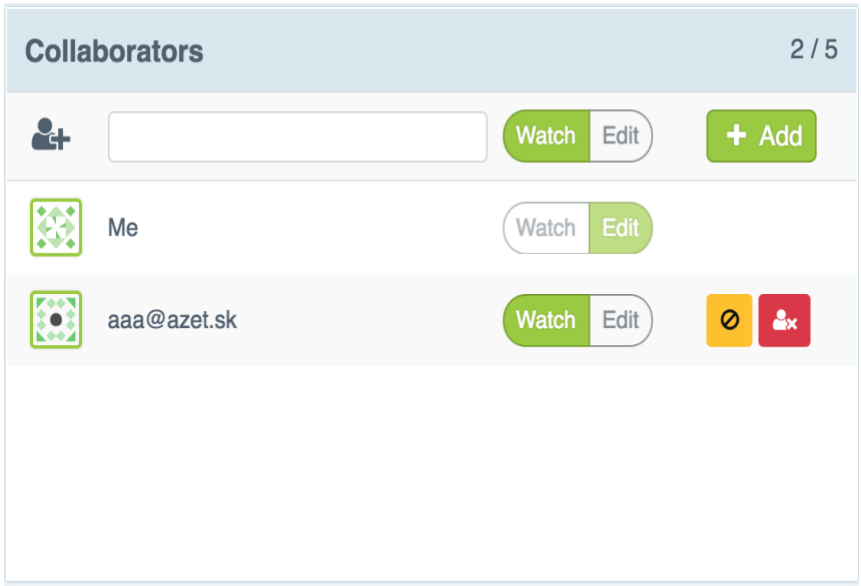
Jediná validácia, ktorá sa vykonáva pri tejto akcií je kontrola či je používateľ, ktorý chce zmeniť práva kolaboranta, naozaj aj vlastník projektu a teda či má patričné práva pre zmenu jeho práv.

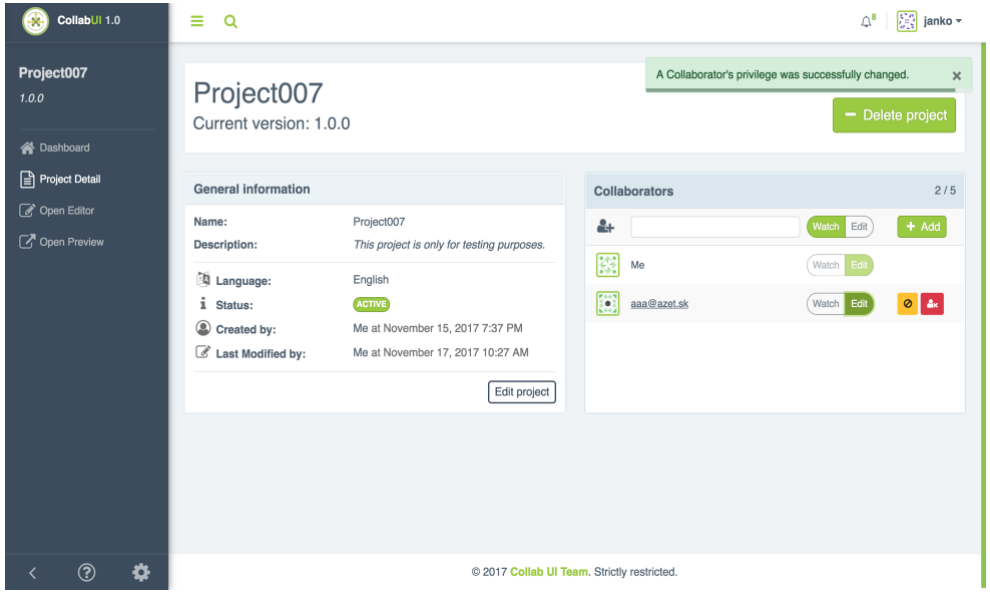
5.3.5.3 Stavový diagram pre zmenu používateľských práv



Obrázok 24 - Stavový diagram pre zmenu používateľských práv

5.3.5.4 *Obrazovka zmeny práv používateľa*





Nasledujúca obrazovka ukazuje box kolaborátorov, v ktorom je možné nastaviť jednotlivé práva. Práva sú vizuálne prezentované prostredníctvom switch tlačidla, teda je jasné, že kolaborátor môže mať vždy len 1 z ponúkaných práv.

Po potvrdení akcie zmeny práva pre konkrétneho kolaboranta je vlastník projektu notifikovaný o tejto zmene prostredníctvom Boostrap notify notifikácie.

Obrázok 25 - Obrazovka zmeny práv používateľa

5.3.6 Odstránenie kolaboranta z projektu

Vlatník projektu má možnosť odstrňovať jednotlivých kolaborantov. Táto akcia sa odohráva v boxe pre kolaborantov k príslušnému projektu. Ako aj ostatné akcie je reprezentovaná príslušnou ikonou. Ak chce vlastník projektu odstrániť konkrétneho kolaboranta musí túto akciu zvoliť pri jeho emaily. Následne sa zobrazí modálne okno, na ktoré využívame bootstrap knižnicu Modal. Prostredníctvom modálneho okna sa uistíme či je používateľova akcia naozaj opodstatnená. Pokiaľ áno tak sa používateľ odstráni zo zoznamu kolaborantov. Pokiaľ nie tak sa nevykonajú žiadne zmeny.

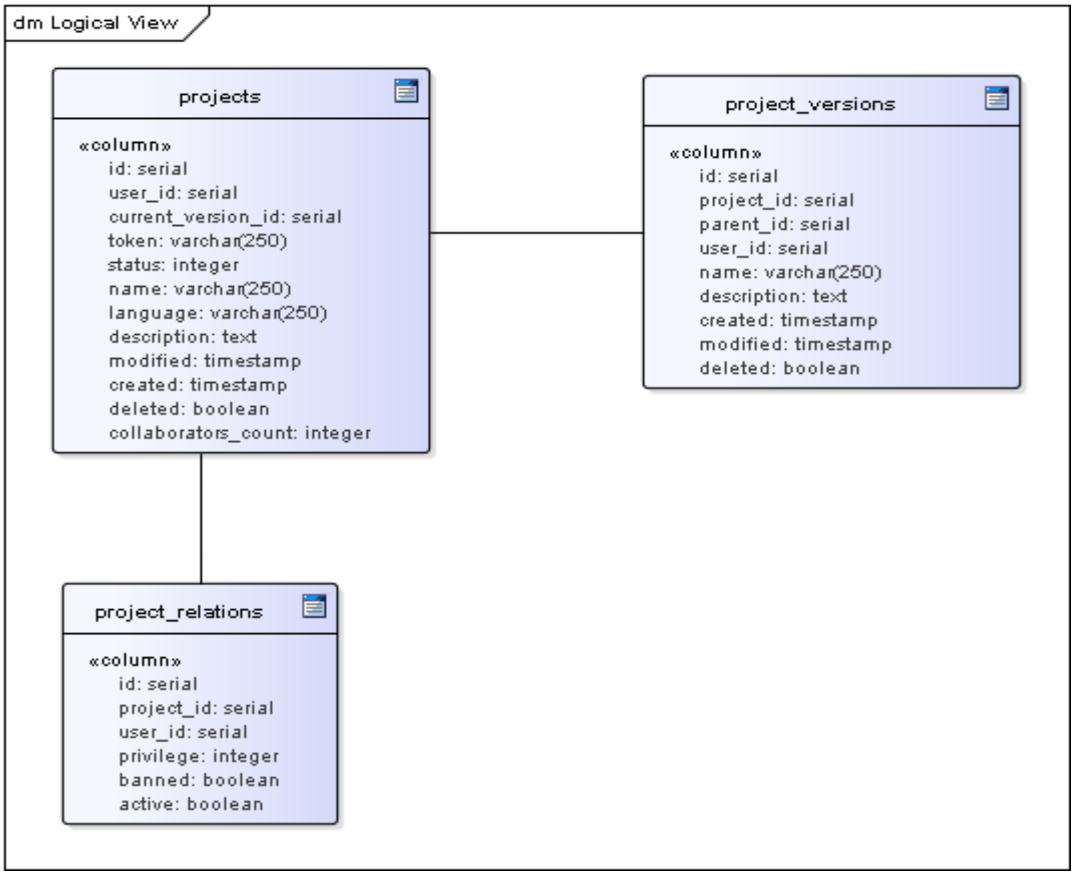
5.3.6.1 Akceptačné kritéria

- Možnosť odstránenia kolaboranta pre vlastníka projektu
- Iba vlastník projektu môže odstraňovať kolaborantov
- Poskytnutie modálneho okna pre potvrdenie akcie vlastníka projektu
- Zníženie počtu kolaborantov pri potvrdení akcie, o konkrétneho používateľa

5.3.6.2 Validácia

Jediná validácia, ktorá sa vykonáva pri tejto akcií je kontrola či je používateľ, ktorý chce kolaboranta zablokovať naozaj aj vlastník projektu a teda či má patričné práva pre zablokovanie iného člena tímu.

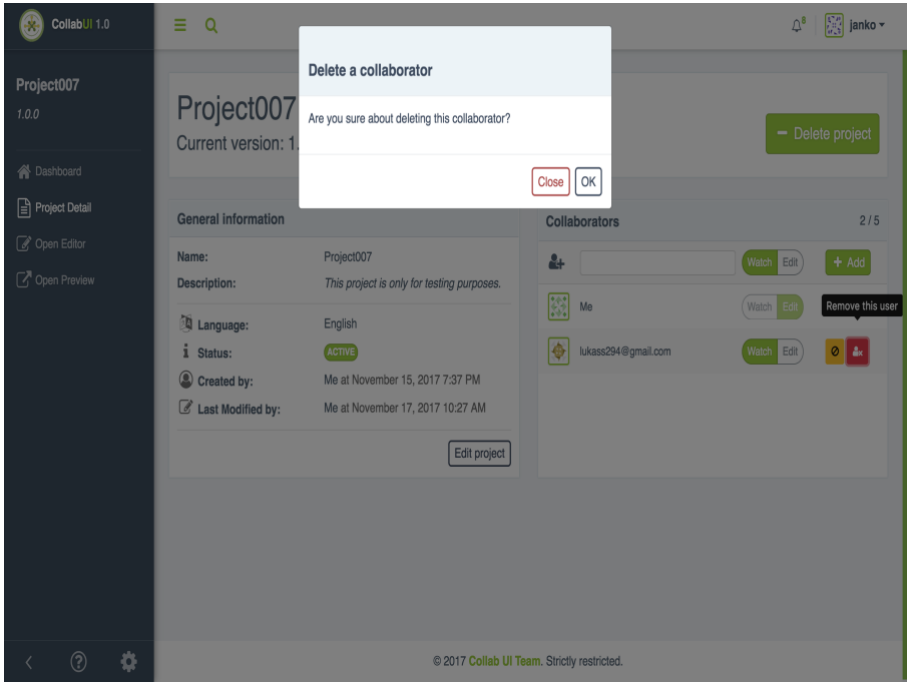
5.3.6.3 Databázový model

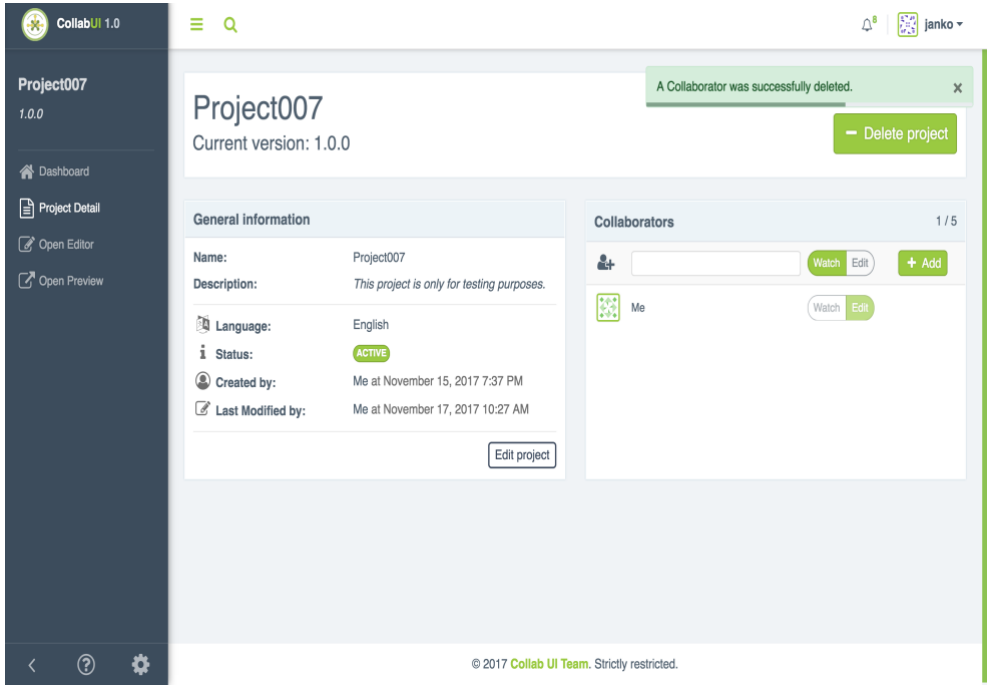


Obrázok 26 - DB Model pre odstránenie kolaboranta

Pri odstránení kolaboranta z projektu sa využíva najmä tabuľka projects, ktorá obsahuje atribút počet kolaborantov, ktorý definuje koľko kolaborantov daný projekt aktuálne má. V prípade odstránenia kolaboranta z projektu sa skontroluje v project_relations kolaborant podľa user_id a následne sa zníči počet collaborators_count.

5.3.6.4 Obrazovka pre odstránenie kolaboranta z projektu





Na nasledujúcej obrazovke vidíme psomínané modálne okno, ktoré čaká na potvrdenie akcie vlastníka projektu.
Akcie, ktoré sú na výber sú:

- Potvrdenie akcie
- Zrušenie akcie

Samozrejme po potvrdenej akcií by mal byť vlastník projektu patrične notifikovaný o tom čo spravil. V tomto prípade sú to znova naše oblúbené notifikácie.

Obrázok 27 - Obrazovka pre odstránenie kolaboranta z projektu

5.3.7 Blokovanie kolaboranta

Blokovanie kolaboranta je vcelku jednoduchá akcia kedy vlastník projektu môže zablokovať niektorého z kolaborantov. Túto akciu môže robiť neustále dokola. Taktiež sa vždy mení grafický element reprezentujúci zablokovanie alebo odblokovanie kolaboranta.

5.3.7.1 Akceptačné kritéria

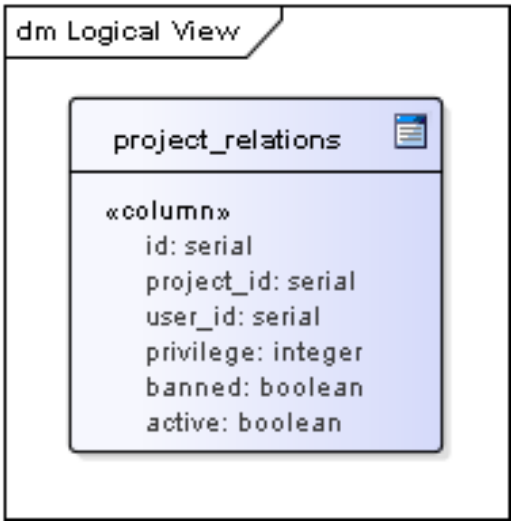
- Blokovanie môže byť vykonávané iba vlastníkom projektu
- Blokovaný kolaborant nesmie byť notifikovaný o jeho aktuálnej zmene práv
- Kolaborant nemá ďalej sprístupnený projekt, na ktorom pracoval

5.3.7.2 Validácia

Jediná validácia, ktorá sa vykonáva pri tejto akcií je kontrola či je používateľ, ktorý chce kolaboranta zablokovať naozaj aj vlastník projektu a teda či má patričné práva pre zablokovanie iného člena tímu.

5.3.7.3 Databázový model

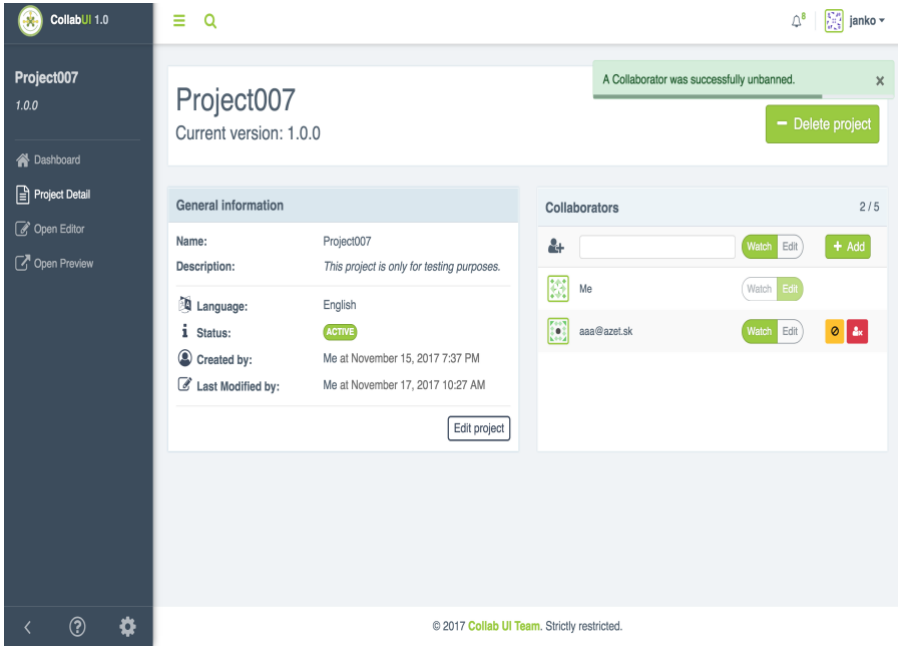
Pri tejto akcií sa využíva iba jediná tabuľka a tou je project_relations.

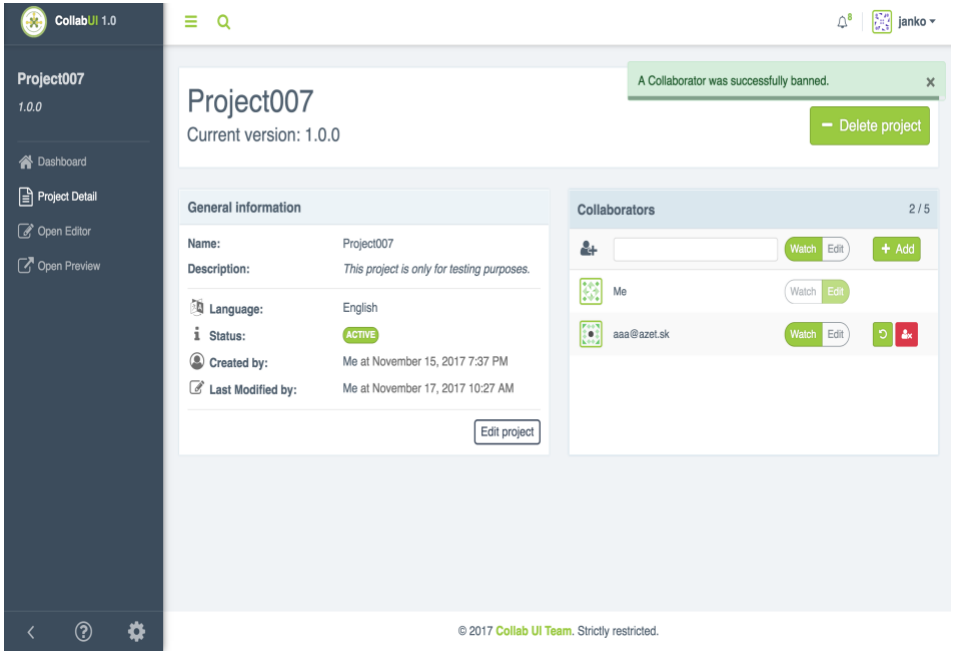


Obrázok 28 - Tabuľka project_relations pre blokovanie kolaboranta

Pri zablokovaní kolaboranta sa nastaví flag “banned” na true.

5.3.7.4 Obrazovka pre blokovanie kolaboranta





Na nasledujúcej obrazovke môžeme vidieť akciu odblokovania používateľa, ktorá okrem funkcionality zahŕňa aj notifikáciu a zmenu grafického prvku.

Pri zablokovaní používateľa sa vykonáva taktiež zmena grafického prvku, spolu s notifikáciou a zmenou práv kolaboranta.

Obrázok 29 - Obrazovka pre blokovanie kolaboranta

5.3.8 Notifikovanie kolaboranta mailom

Táto akcia je následkom jednotlivých akcii používateľov. Skoro vždy keď vlastník projektu vykoná isté zmeny, su o tom ostatní kolaboranti notifikovaní. Toto neplatí pri nasledujúcej akcii:

- Blokovanie kolaboranta

Pre notifikovanie kolaborantov používame emailové šablóny, prostredníctvom ktorých vieme posielať aj URL pre akcie používateľa akými môže byť otvorenie projektu, do ktorého bol čerstvo pozvaný. Ináč slúžia na informačné účely. Emailové šablóny sú v jednotlivých ctp súboroch. Taktiež sa používa technológia Premailer.php, ktorá operuje na backende aplikácie.

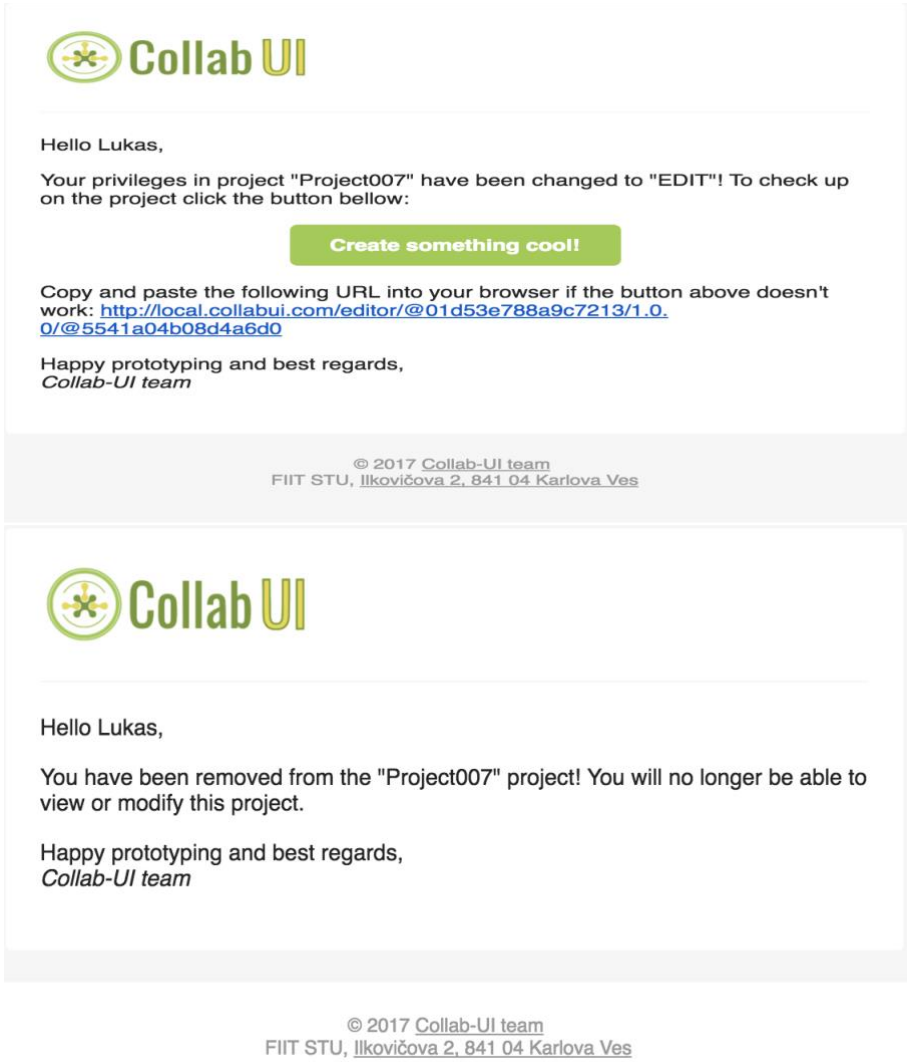
5.3.8.1 Akceptačné kritéria

- Notifikovanie kolaboranta pri každej zmene vykonanej vlastníkom projektu
- Nenotifikovať kolaboranta pri jeho zablokovaní

5.3.8.2 Validácia

Jediná validácia, ktorá sa vykonáva pri tejto akcií je kontrola zoznamu kolaborantov a ich emailov.

5.3.8.3 Obrazovky pre notifikovanie kolaboranta



Nasledujúce obrazovky reprezentujú notifikovania kolaboranta o rôznych akciách. Akcie sú nasledovné:

- Zmena práv kolaboranta
- Priedelenie kolaboranta na projekt
- Odstránenie kolaboranta z projektu

Obrázok 30 - Obrazovky pre notifikovanie kolaboranta

5.3.9 Testovanie

Pri testovaní v tomto bode sme potrebovali použiť autentifikáciu pri dopyte na ajaxové volania. Použili sme helper metódu na zaručenie autentifikácie.

Testovali sme nasledujúce scénare:

- Pridanie kolaboranta na projekt
- Pridanie rovnakého kolaboranta na projekt
- Pridanie viac ako 5 kolaborantov na jeden projekt
- Zmena práv pridaných kolaborantov
- Blokovanie pridaných kolaborantov
- Odstraňovanie pridaných kolaborantov

```
$user = $this->Users->find()->first();
$this->session([
    'Auth' => [
        'User' => [
            'id' => $user->id,
            'email' => $user->email,
        ]
    ]
]);
```

Obrázok 31 - Autentifikácia používateľa v teste

5.4 Editor

5.4.1 Úvod

Ďalší bod, ktorým sme sa zaoberali v rámci projektu CollabUI je samotný editor. Rozdelili sme ho na 2 časti a tými sú:

- Editor
- Kolaboratívny editor

V rámci editora oslobodeného od kolaborácie bolo nutné vyriešiť niekoľko problémových oblastí. Jednou s nich je konfigurácia podporného nástroja pre tvorbu stránok, ktorý predstavuje jadro pre úpravu a vytváranie jednotlivých stránok. Pomocou neho bude môcť používateľ lepšie štýlovať stránky bez nutnosti ovládania programátorských zručností. Elementy na stránke sú reprezentované ako komponenty a práve preto sme museli vyriešiť aj ukladanie týchto komponentov aby sme neskôr vedeli uchovávať prototyp, na ktorom vlastník projektu alebo kolaborant pracoval. Pri príchode na projekt sa používateľovi zobrazí základný prototyp rozloženia stránky, ktorý slúži na iniciálne oboznámenie sa s nástrojom. Tu používateľ môže vidieť akým štýlom môžu byť rôzne komponenty kombinované a taktiež ako ich meniť. V tomto bode sme si pred pripravili aj samotnú kolaboráciu účastníkov pracujúcich na konkrétnom projekte. V tomto prípade využívame knižnicu NodeJS na nadviazanie spojenia medzi používateľmi. Taktiež bude zabezpečovať viditeľnosť zmien v reálnom čase.

Čo sa týka editora s implementáciou kolaborácie sme dorábali jednotlivé časti akými bolo preposielanie dát kolaborantov. Konkrétne sa jedná o to, ktorí kolaboranti sú v miestnosti na danom projekte pripojení. Taktiež sa zobrazujú v bočnom panely. Pri príchode nového kolaboranta ako aj pri jeho odchode z projektu (resp. Opustenie vytvoreného socketu) dostanú zvyšní kolaboranti notifikácie o týchto zmenách. Ďalej sme implementovali obmedzenia pre role „watch“ a role „banned“. Najväčšiu časť Editora tvorí samotná kolaborácia resp. premietnutie zmien vykonávaných v editori ostatným kolaborantom v reálnom čase. Ďalšou obšírnou časťou editora tvoria bočné panely nástroja. Používateľ bude mať možnosť si v nastaveniach vybrať, ktoré panely sa budú nachádzať v pravej alebo ľavej časti editora. Taktiež si môže prispôsobovať ich veľkosť podľa vlastných potrieb.

5.4.2 Analýza

V rámci analýzy sme riešili najmä funkcionality knižnice pre tvorbu stránok. Zistili sme, že keďže je táto knižnica ešte relatívne nová, tak neponúka veľa funkcii čo sa týka ukladaní a prácou s komponentami. Pri ukladaní sa používa vstavaný objekt storageManager. Tento objekt ponúka funkcie akými sú auto ukladanie, získavanie predchádzajúcich krokov pri zmene komponentov, získavanie samotného komponentu ako aj jeho pridávanie. Taktiež sme zistili, že tento objekt pri ukladaní komponentov vždy pracuje so všetkými komponentami, ktoré sa v editore nachádzajú. Táto skutočnosť nie je pre nás najvýhodnejšia, keďže potrebujeme narábať často krát len s jedným konkrétnym komponentom. Riešenie by mohlo spočívať v pre-iterovaní všetkých komponentov pri posielaní do editora a následnou selekciou tých špecifických komponentov, s ktorými používateľ potrebuje pracovať.

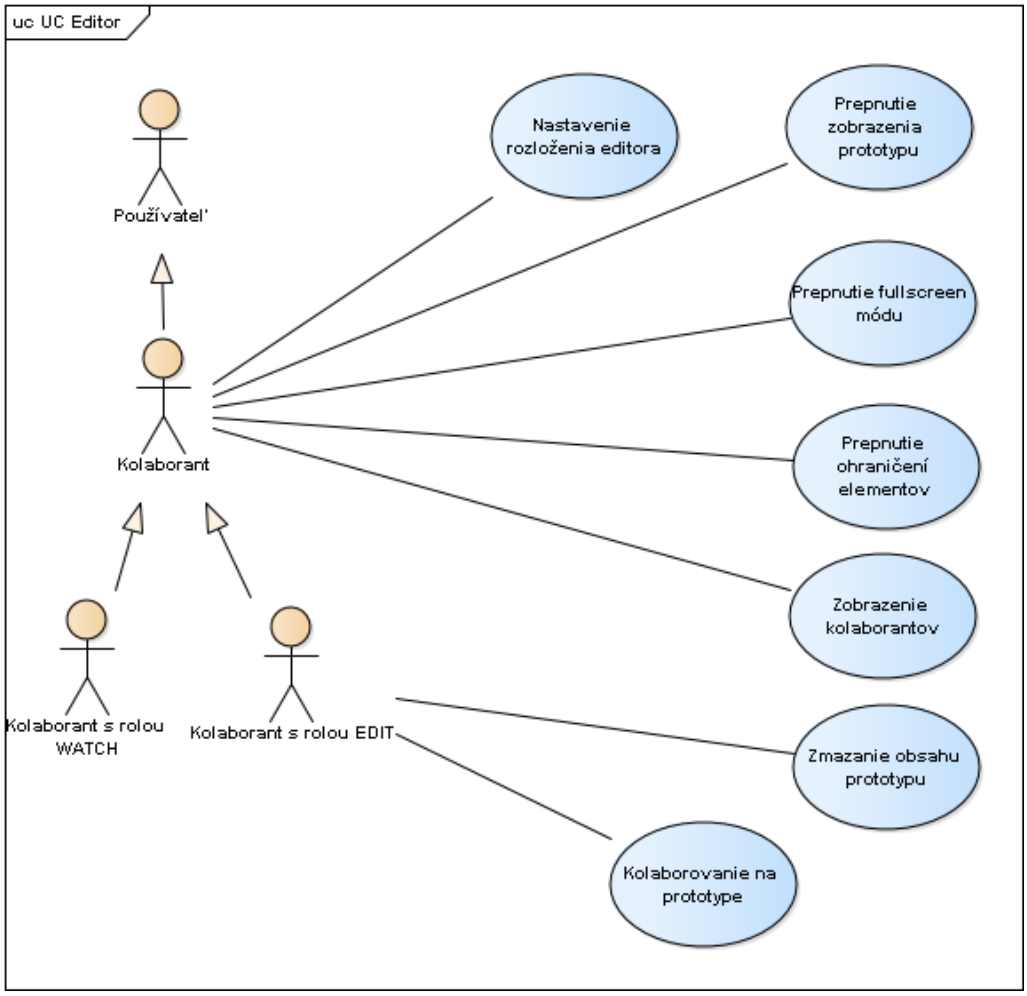
Taktiež sme identifikovali skutočnosť, že na mobilných zariadeniach nebude možné efektívne pracovať s editorom kvôli nízkemu rozlíšeniu a komplexnosti riešenia, práve z tohoto dôvodu sme zavrhli podporu mobilných zariadení v rámci editora a vytvorili novú chybovú stránku.

5.4.3 Návrh

Samotný editor sa skladá z nasledujúcich UC:

- Nastavenie rozloženia editora
- Prepnutie zobrazenia prototypu
- Prepnutie fullscreen módu
- Prepnutie ohraničení elementov
- Zmazanie obsahu prototypu
- Zobrazenie kolaborantov
- Kolaborovanie na prototype

Nasledujúci use case diagram znázorňuje akcie používateľa pri práci s editorom.



Obrázok 32 - UC diagram pre editor

UC Nastavenie rozloženia editora

Hlavný tok:

1. Používateľ zvolí možnosť “Prispôsobenie nastavení”
2. Používateľ nastaví umiestnenie jednotlivých nástrojov
3. Používateľ zvolí možnosť “Aktualizovať”
4. Systém zreviduje zmeny a podľa zvolených umiestnení nástrojov ich aktualizuje v editore
5. Systém notifikuje používateľa o vykonaných zmenách.
6. Prípad použitia končí.

UC Prepnutie zobrazenia prototypu

Hlavný tok:

1. Používateľ zvolí možnosť “Prepnutie zobrazenia” z 3 možností:
    - Mobil
    - Tablet
    - Desktop
2. Systém prepne zobrazenie prototypu podľa zvolenej možnosti
3. Prípad použitia končí

UC Prepnutie fullscreen módu

Hlavný tok:

1. Používateľ zvolí možnosť “Prepnutie fullscreen”
2. Systém prepne prototyp do režimu celého okna
3. Prípad použitia končí

UC Prepnutie ohraničení elementov

Hlavný tok:

1. Používateľ zvolí možnosť “Ohraničenie elementov”
2. Systém zobrazí ohraničenia elementov
3. Prípad použitia končí

UC Zmazanie obsahu prototypu

Hlavný tok:

1. Používateľ zvolí možnosť “Vyčistiť obsah prototypu”
2. Systém overí akciu používateľa prostredníctvom modálneho okna
3. Používateľ potvrdí akciu
4. Systém vymaže komponenty nastavené pre konkrétny projekt a uloží aktuálny stav
5. Prípad použitia končí

UC Zobrazenie kolaborantov

Hlavný tok:

1. Systém pri inicializácii editora prihlási používateľa
2. Systém notifikuje ostatných prihlasených používateľov o príchode nového kolaboranta
3. Systém zobrazí kolaboranta v panely určenom pre evidenciu kolaborantov
4. Prípad použitia končí

UC Kolaborovanie na prototype

Hlavný tok:

1. Systém pri inicializácii editora prihlási používateľa

2. Používateľ zvolí element na prototype alebo potiahne element z panelu preddefinovaných blokov do prototypu

3. Systém uloží vykonanú zmenu na prototype

4. Systém odošle vykonanú zmenu všetkým prihláseným kolaborantom a aktualizuje ich lokálny prototyp v reálnom čase

5. Prípad použitia končí

5.4.4 Implementácia

5.4.5 Editor

Základným stavebným kameňom celého projektu je editor. V samotnom editore bude môcť používateľ vykonávať zmeny v rámci konkrétneho prototypu. V súčasnej implementácii neuvažujeme o simultánnom vytváraní viacerých prototypov ani o verziovaní. Základ editora tvorí podporná knižnica pre vytváranie prototypov vo webovom rozhraní, ktorej funkcionality využívame. Knižnica disponuje 4 vstavanými panelmi:

- Panel pre pridanie preddefinovaných elementov
- Panel pre úpravu štýlov
- Panel vrstiev
- Panel konfigurácie komponentov

Naraz však dokáže knižnica pracovať iba s jedným panelom. Túto skutočnosť sme vyriešili vlastnou implementáciou, ktorá dovoľuje pridať ďalšie panely. Kvôli budúceho rozširovania editora sme pred pripravili nasledujúce panely:

- Panel pre zoznam prototypov (stránok)
- Panel pre zoznam poznámok
- Panel pre históriu akcii
- Panel pre zoznam kolaborantov

Taktiež sme mysleli na potrebu používateľa prispôsobiť rozloženie editora a z tohoto dôvodu sme pridali možnosť nastavenia umiestnenia jednotlivých panelov a to:

- Vpravo hore
- Vpravo dole
- Vľavo hore
- Vľavo dole

Panely taktiež umožňujú zmeniť veľkosť a v prípade, že používateľ nevložil do niektorého umiestnenia ani jeden panel, dané umiestnenie sa skryje a editor sa prispôsobí k tejto zmene. Rozloženie editora je možné uložiť pre právo „WATCH“ a zvlášť pre „EDIT“, totiž pri práve „WATCH“ je používateľ obmedzený len na 4 panely, vstavané panely knižnice sa v tomto prípade nevyužívajú.

Ukladanie prototypu prebieha automaticky po každej zmene na strane NodeJS Servera, pričom dáta sa kvôli veľkosti ukladajú do rýchlej nerelačnej databázy. K ukladaniu prototypu sme využili JSON reprezentáciu, tak ako ju definuje podporná knižnica.

Funkcionality ako prepnutie zobrazenia prototypu, fullscreen, či ohraničenia elementov sme prepoužili z podpornej knižnice.

5.4.5.1 Akceptačné kritéria

- Možnosť prihlásenia sa do editora a následná autorizácia
- Vyskladanie prototypu prostredníctvom editora
- Vytvorený prototyp je automaticky uložený po zmene
- Možnosť zmeny rozloženia panelov editora pre právo „EDIT“ a zvlášť pre právo „WATCH“

5.4.5.2 Autorizácia

Autorizácia v editore prebieha prostredníctvom vygenerovaných token, s ktorými disponuje každý používateľ a každý projekt.

Príklad URL: editor/@922b9fcf0ef4ab53/41/@4637f2ca7eed4f83

Tokeny začínajú znakom @.

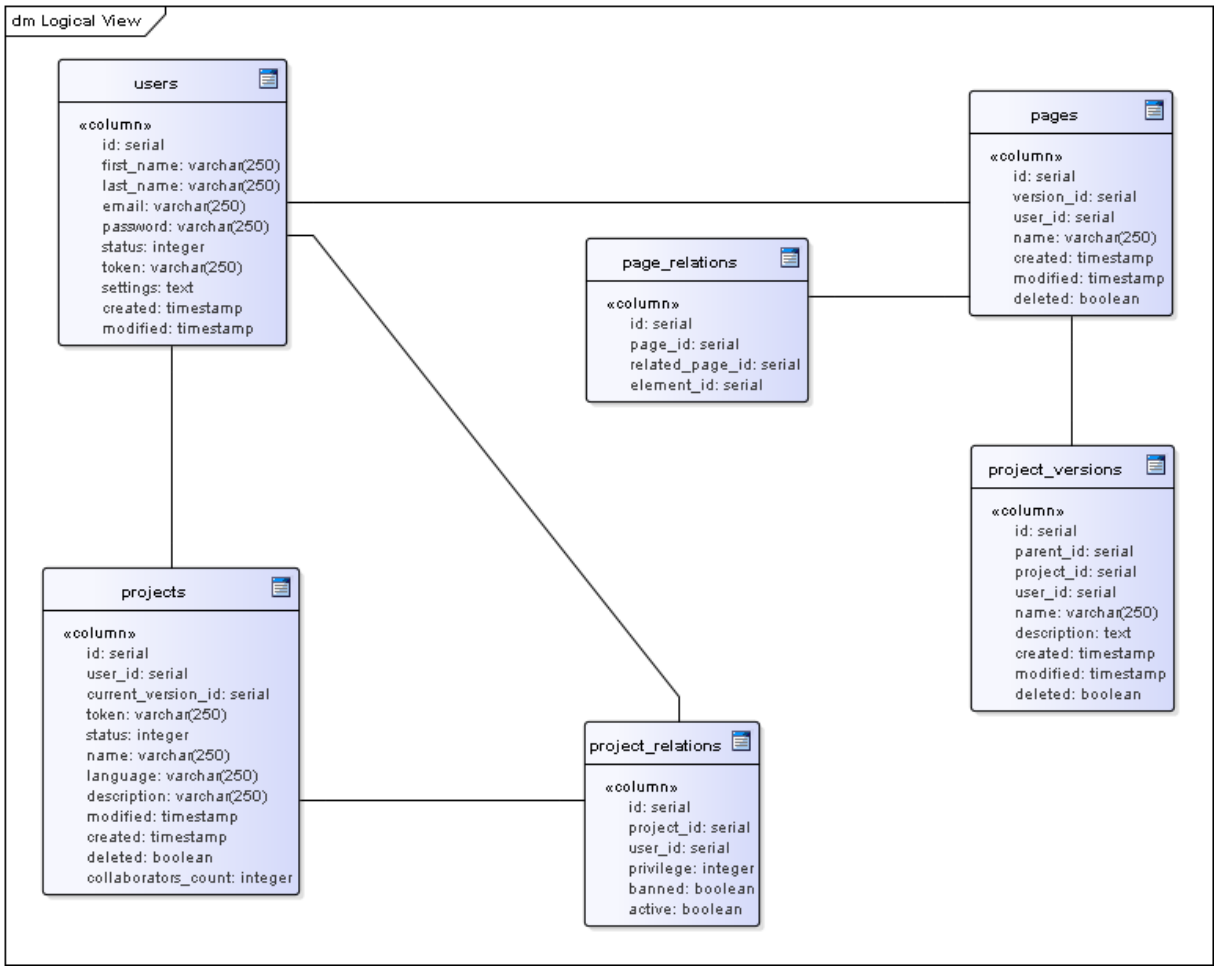
- Token projektu
- ID verzie projektu
- Token používateľa

Pri prístupe na editor s danou URL sa identifikuje projekt a na základe uloženého vzťahu projekt-používateľ sa identifikujú používatelia, ktorí majú prístup. Potom na základe tokenu používateľa vyberie zo zoznamu povolených používateľov konkrétny s právom „EDIT“ alebo „WATCH“. Ak tokeny nekorešpondujú hore uvedenej skutočnosti, prístup sa vyhlási za neautorizovaný a používateľovi je zobrazená chybová stránka.

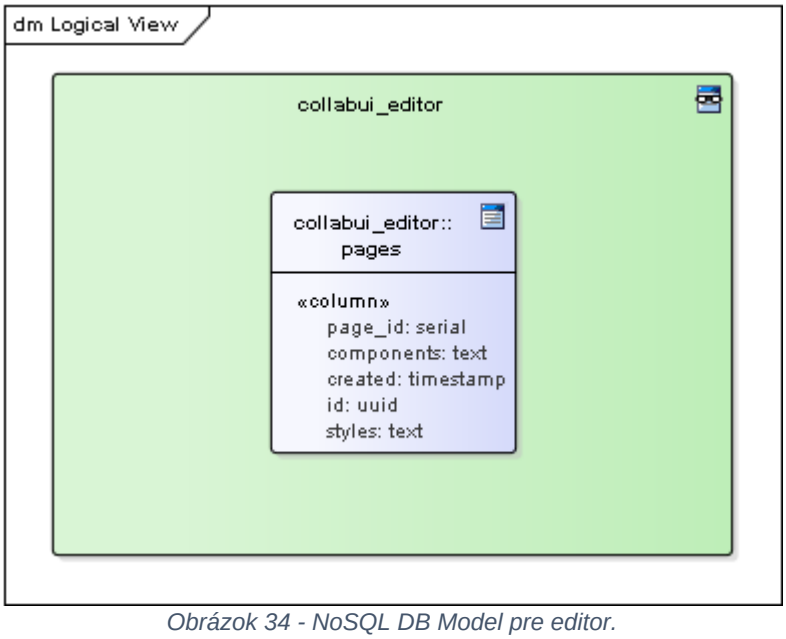
Ak prvou úrovňou autorizácie používateľ prešiel úspešne zobrazíme mu editor. Po inicializácii editora nadviaže systém komunikáciu s NodeJS Serverom kde prebieha druhá úroveň autorizácie rovnakým princípom.

5.4.5.3 Dátový model

Poznámka: Zelený štvorec v NoSQL diagrame znázorňuje keyspace

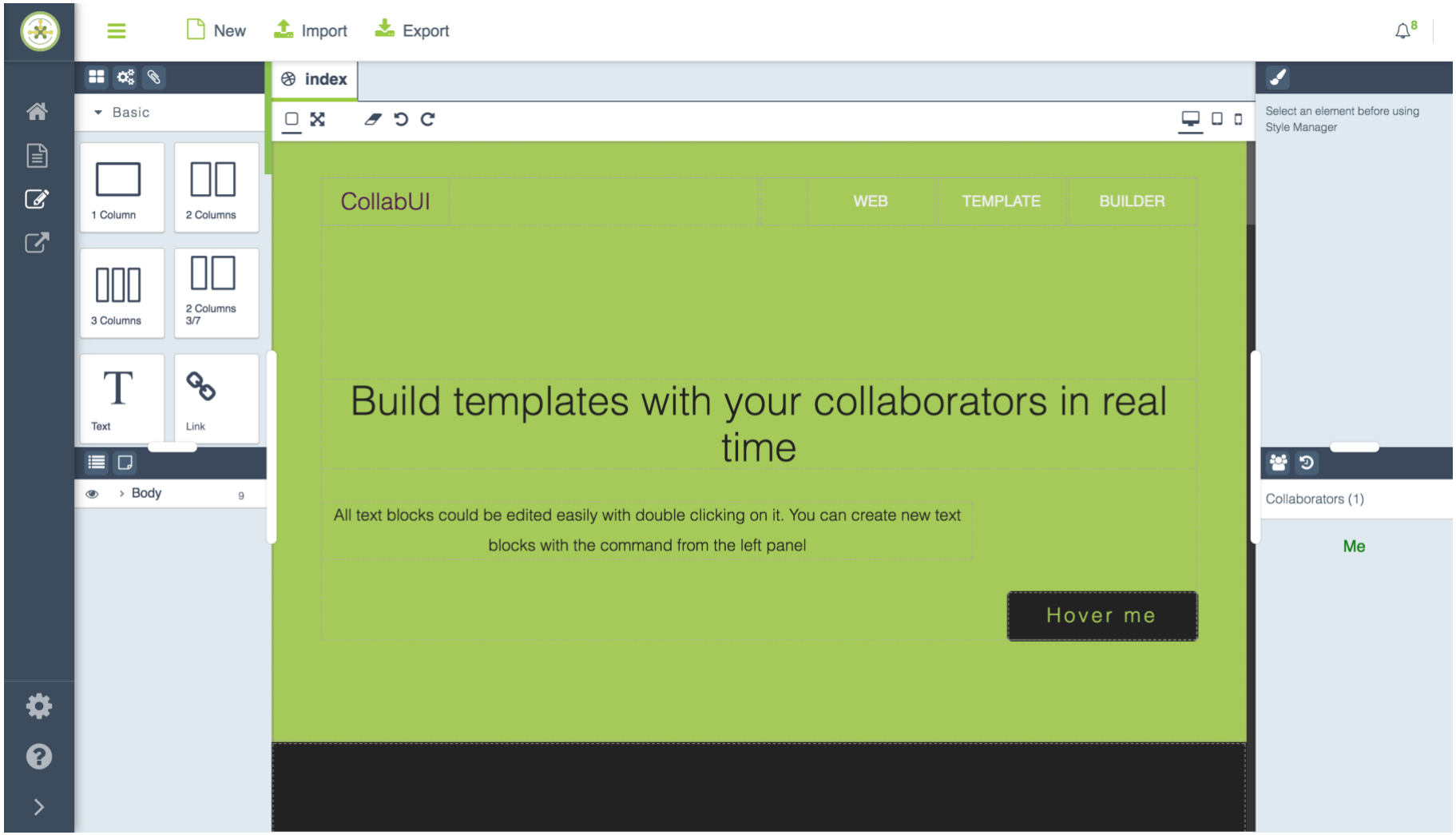


Obrázok 33 - DB Model pre editor



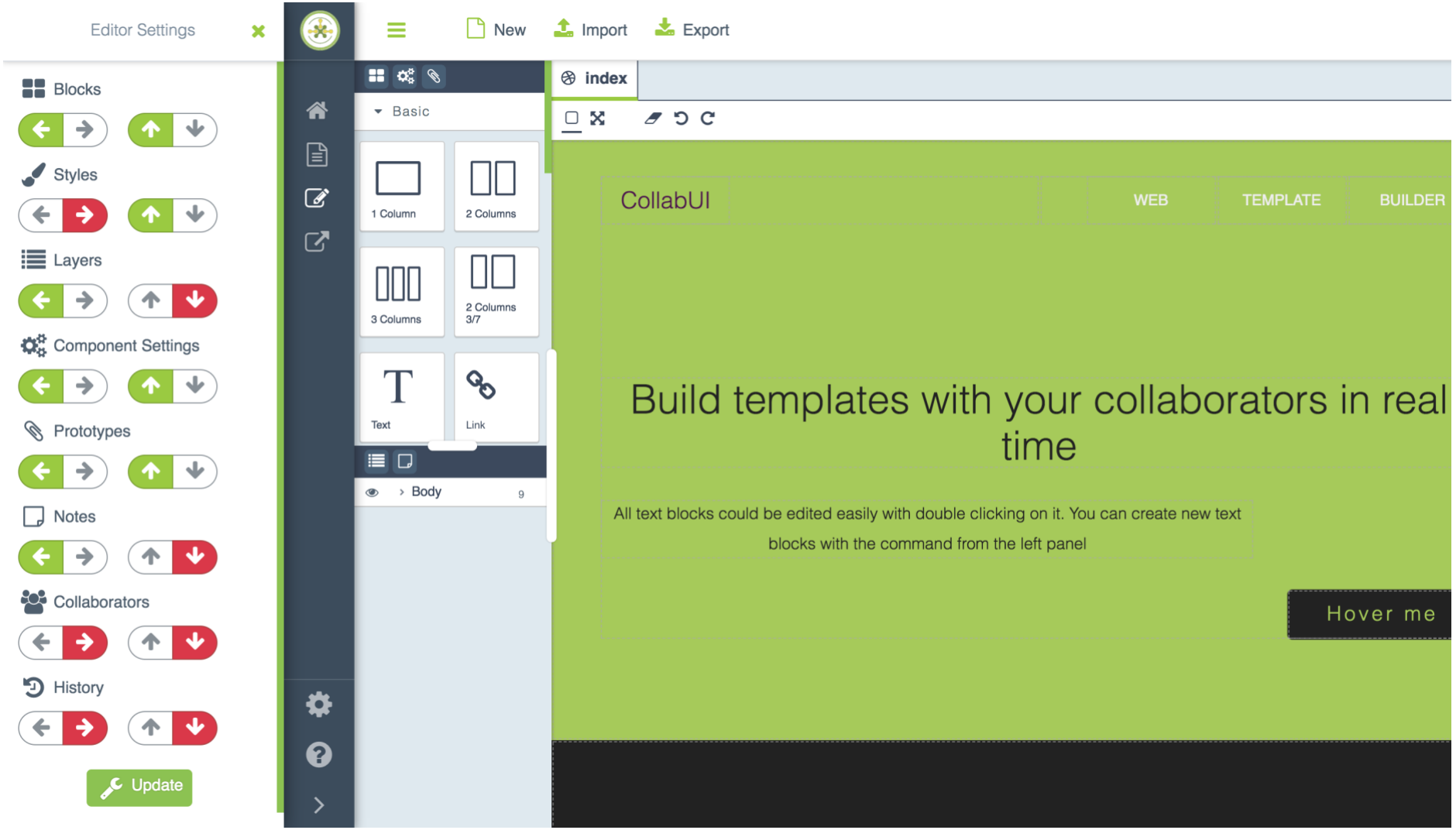
Obrázok 34 - NoSQL DB Model pre editor.

5.4.5.4 Obrazovka editora



Obrázok 35 - Obrazovka pre editor

5.4.5.5 Obrazovka editora (nastavenie rozloženia panelov)



Obrázok 36 - Nastavenie rozloženia panelov v editore

5.4.6 Kolaboratívny editor

Po implementácii základnej kostry editora sme pristúpili k implementácii kolaboratívneho módu, čo je hlavným poslaním vyvíjaného systému. K implementácii zmien v prototype v reálnom čase sme využili NodeJS Server s kombináciou so socket.io knižnicou prostredníctvom, ktorej posielame upravené elementy na server, kde sa dáta rozposielajú cez broadcast ostatným pripojeným kolaborantom, hneď po uložení zmien v rýchlej nerelačnej databáze Cassandra. Taktiež zachytávame udalosť označenia elementu používateľom, ktorú rozposielame cez broadcast kolaborantom, pričom sa im daný element vizuálne ohraničí.

Po prihlásení alebo odhlásení používateľa z editora, taktiež notifikujeme ostatných kolaborantov formou notifikácie, zároveň aktualizujeme panel pre zoznam kolaborantov, kde vizuálne odlišujeme pripojených od nepripojených kolaborantov. Taktiež sme pripravili podporu viacerých simultánne editovateľných prototypov v už spomínanom panely. Pre každého kolaboranta udržujeme informáciu, na ktorom prototype aktuálne pracuje, pomocou čoho dokážeme kategorizovať pripojených kolaborantov.

Okrem už spomínaného sme implementovali obmedzenia pre kolaborantov s právom WATCH, ktorí majú okrem znemožnené nasledovné aktivity:

- Zmazanie obsahu prototypu
- Úprava, editovanie prototypu
- 4 vstavané panely knižnice

Ďalším obmedzením je stav BANNED. Používateľ s týmto obmedzením je po prístupe na editor presmerovaný na chybovú stránku.

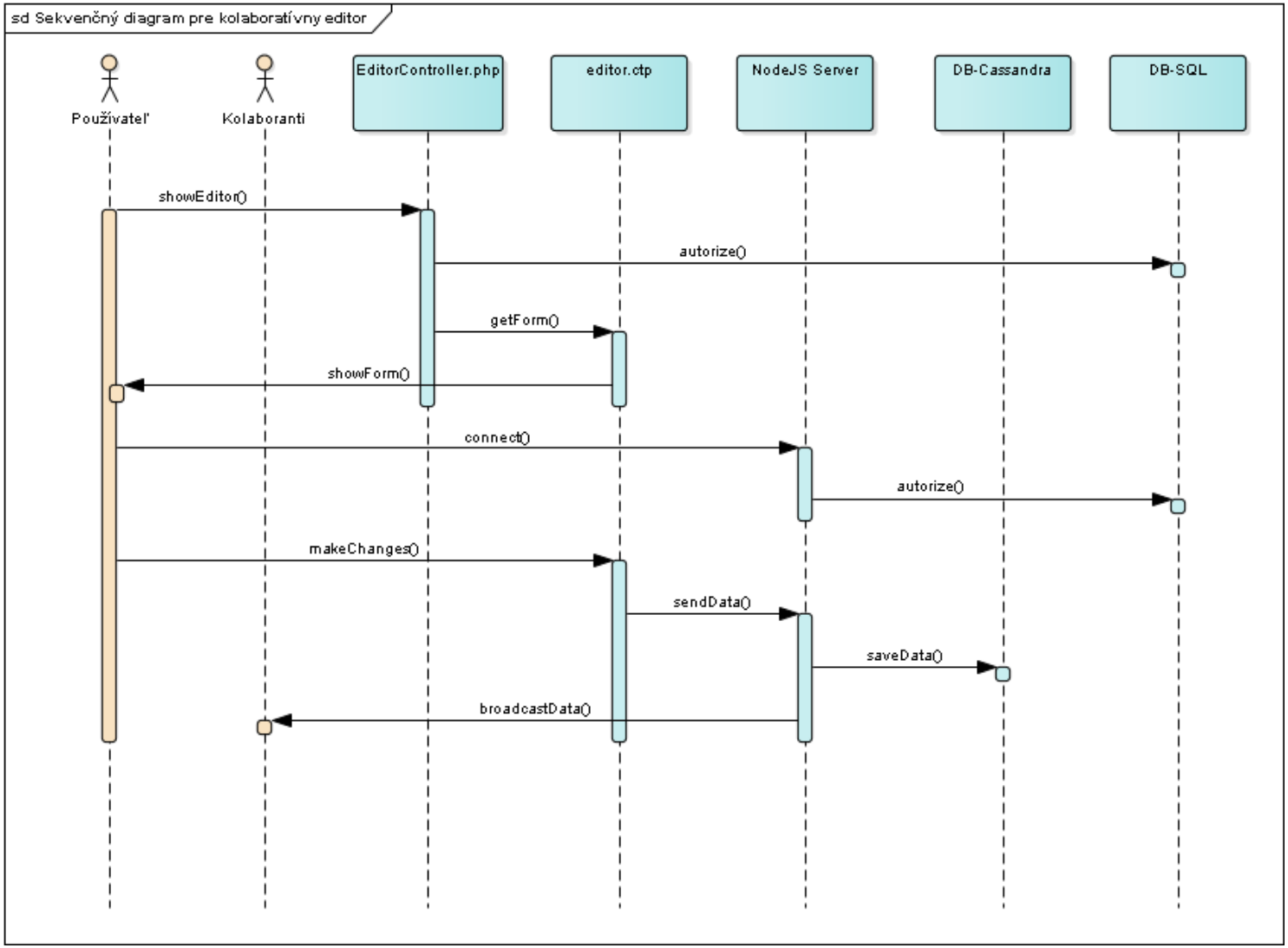
5.4.6.1 Akceptačné kritéria

- Používateľ s právom WATCH, nedokáže editovať prototype alebo vykonať akúkoľvek zmenu, ktorú by systém zachytil a uložil
- Používateľ s právom BANNED je presmerovaný na chybovú stránku
- Používateľ vidí v reálnom čase vykonané zmeny na prototype inými kolaborantami
- Používateľ vidí aktuálne označené elementy prototypu ostatných kolaborantov
- Používateľ je notifikovaný pri príchode a odchode kolaboranta
- Používateľ má prístup k zoznamu aktuálne prihlásených kolaborantov v editore

5.4.6.2 Validácia

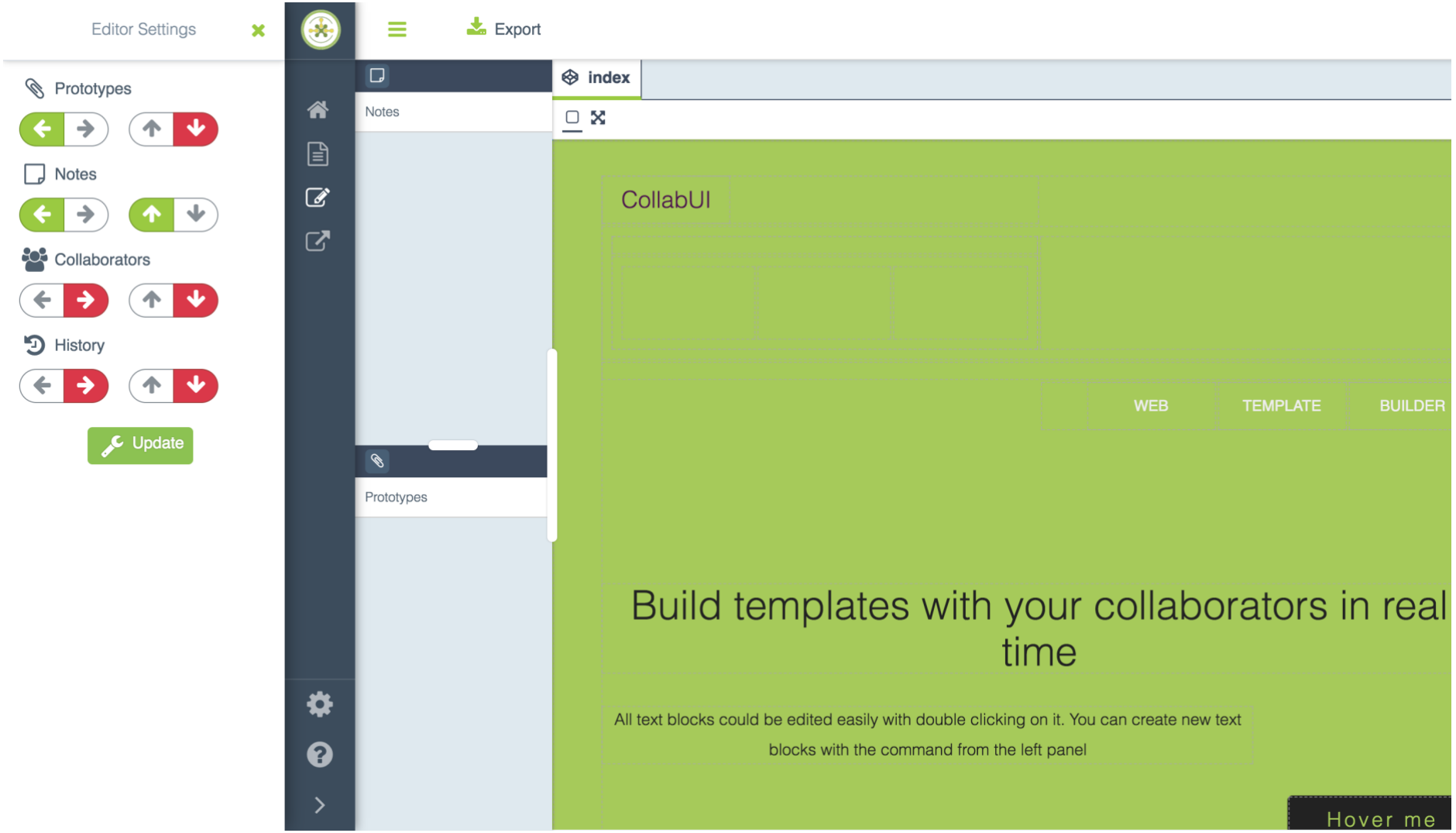
Pri kolaboratívnom editore je dôležité hlavne validovať skutočnosť, že používateľ s právom WATCH nesmie za žiadnych okolností upraviť prototyp. Validácia prebieha na strane NodeJS Servera pred uložením do db a rozposlaním dát cez broadcast ostatným kolaborantom.

5.4.6.3 Sekvenčný diagram



Obrázok 37 - Sekvenčný diagram pre kolaboratívny editor

5.4.6.4 Obrazovka editora v obmedzenom režime WATCH



Obrázok 38 - Obrazovka editora v obmedzenom režime WATCH

5.4.7 Testovanie

Kvôli charakteru naplánovaných funkcionalít a komplexnosti riešenia sme sa k implementácii automatizovaných testov nedostali. Budú predmetom budúcej integrácie.

5.4.8 Úvod

Bonusovým bodom, ktorý sme si v rámci posledného šprintu dodatočne naplánovali je manažment tagov. Táto časť bude dôležitá najmä pri budúcom zefektívnení vyhľadávania projektov používateľmi na dashboarde. Tagovanie je nesmierne dôležité kvôli orientácii používateľa medzi vytvorenými alebo pridelenými projektmi.

5.4.9 Analýza

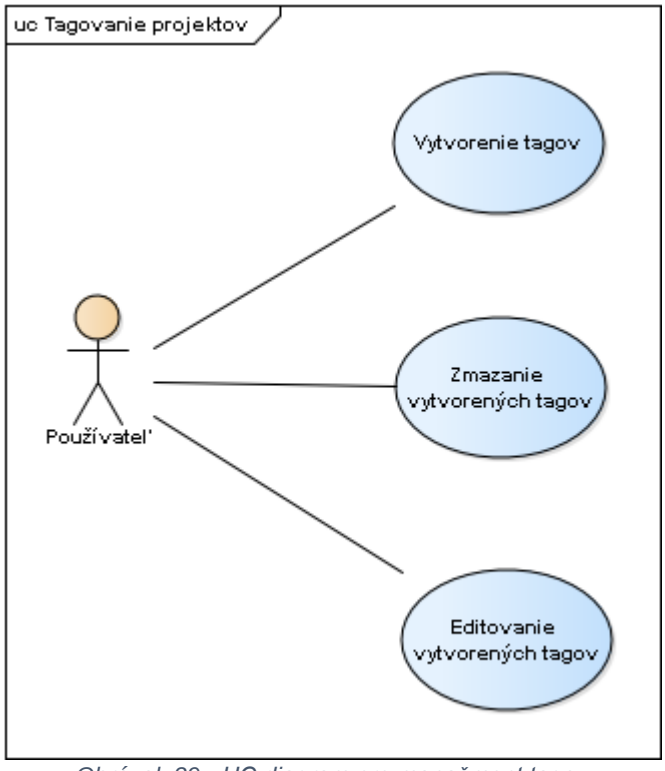
V rámci analýzy sme dospeli k záveru, že kvôli narastajúcemu počtu vytvorených projektov je potrebné dopracovať zefektívnenie vyhľadávania, ktorej realizáciu sme zamýšľali príliš jednoduchú. Rozhodli sme sa teda riešenie postaviť na otagovaní projektov.

5.4.10 Návrh

Manažment tagov sa skladá z nasledujúcich UC:

- Vytvorenie tagov
- Zmazanie vytvorených tagov
- Editovanie vytvorených tagov

Nasledujúci use case diagram znázorňuje základný sled akcií používateľa pri vytváraní používateľského účtu.



Obrázok 39 - UC diagram pre manažment tagov

UC Vytvorenie tagov

Hlavný tok:

1. Používateľ otvoril detail projektu, ktorý chce otagovať
2. Používateľ vyplní pole pre názov tagu
3. Používateľ zvolí farbu pre tag
4. Používateľ iniciuje vytvorenie tagu
5. Systém vytvorí tag
6. Systém pridelí tag na projekt
7. Prípad použitia končí

Alternatívny tok:

2. a1. Po 3 znakoch ponúkne systém používateľovi zoznam korešpondujúcich tagov, ktoré už vytvoril v minulosti
2. a2. Používateľ vyberie tag zo zoznamu
2. a3. Systém predvyplní farbu
2. a4. Používateľ iniciuje pridanie tagu na projekt
2. a5. Prípad použitia pokračuje krokom 6

UC Zmazanie vytvorených tagov

Hlavný tok:

1. Používateľ otvoril detail projektu, z ktorého chce odobrať tag
2. Používateľ zvolil možnosť „Zmazanie tagu“
3. Systém oddelil tag od projektu
4. Prípad použitia končí

UC Editovanie vytvorených tagov

Hlavný tok:

1. Používateľ otvoril detail projektu, ktorého tag chce editovať
2. Používateľ zeditoval tag
3. Systém uložil zmeny, ktoré sa prejavia na každom projekte disponujúci daným tagom
4. Prípad použitia končí

5.4.11 Implementácia

5.4.11.1 Tagovanie projektu

K implementácii tagovania projektov sme využili už použité princípy, práve z tohto dôvodu sme dokázali realizovať funkčnosť v extrémne krátkom čase.

K úpravám došlo na obrazovkách:

- Dashboard – Zoznam tagov k mini detailu projektu
- Detail projektu – Box pre manažment tagov

Predvolené tagy sa vytvárajú pri:

- Vytvorení projektu – owner, edit
- Pridelení projektu – assigned, edit alebo watch

5.4.11.2 Akceptačné kritéria

- Po vytvorení, resp. pridelení projektu sa vytvoria nové tagy
- Vytvorené tagy používateľ vidí na dashboarde
- Používateľ dokáže vytvoriť nový tag
- Používateľ dokáže upraviť vytvorený tag
- Názvy predvolených tagov nie je povolené meniť, ani tagy mazať
- Auto dopĺňanie v minulosti vytvorených tagov

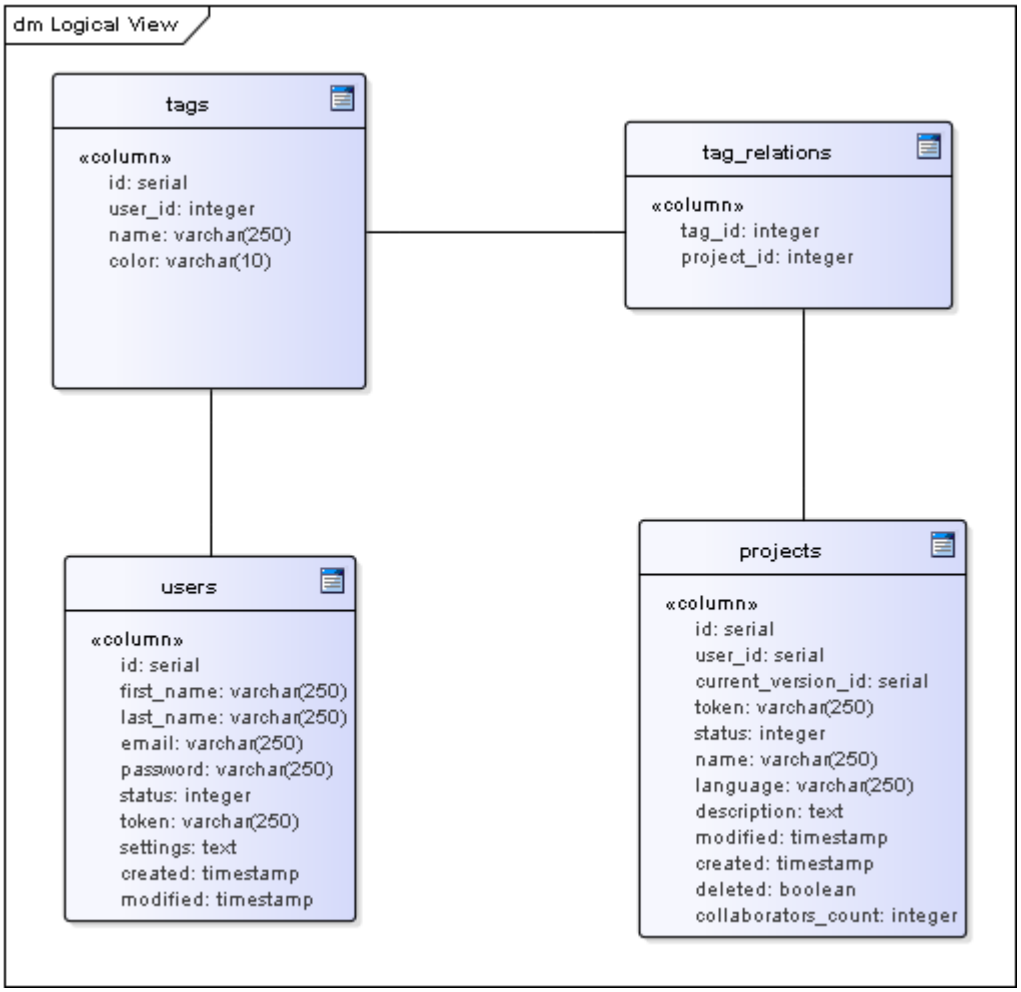
5.4.11.3 Validácia

V rámci tagovania projektov validujeme skutočnosť či už tag bol vytvorený prihláseným používateľom. Ak áno tag nevytvárame. Pri pridaní či editácii tagu validujeme existenciu nasledovných polí:

- Názov tagu
- Farba tagu

Pri zmazaní a editácii validujeme či bol tag vytvorený systémom. Ak áno požiadavku nezrelizujeme.

5.4.11.4 Databázový model



Obrázok 40 - Databázový model pre manažment tagov

5.4.12 Testovanie

Kvôli charakteru naplánovaných funkcionalít a komplexnosti riešenia sme sa k implementácii automatizovaných testov nedostali. Budú predmetom budúcej integrácie.

6 Záver

Na záver tohto dokumentu by sme poznamenali skutočnosť, že sme na projekte pracovali kontinuálne a podarilo sa nám zaimplementovať slušný základ systému určeného pre kolaboratívne prototypovanie v reálnom čase. Poskytnutú infraštruktúru sme zrefaktorovali a obohatili novými technológiami ako napríklad:

- Automatizácia procesov prostredníctvom Gulp
- CRUD plugin pre CakePHP
- AOWeb
- Sass

V rámci celkového pohľadu na projekt môžeme konštatovať, že sme vyriešili nutné podmienky ku kvalitatívnemu používaniu systému. V systéme je možné založiť nový účet, ktorý je možný aktivovať a prihlásiť sa ak používateľ zabudne svoje heslo, môže požiadať o možnosť vytvorenie nového hesla. Prihlásený používateľ dokáže vytvoriť nový projekt a editovať jeho atribúty či ho zmazať. Ďalej dokáže na projekt prideliť kolaborantov i takých, ktorí u nás ešte nemajú zaregistrovaný účet. Pridaným kolaborantom má možnosť meniť práva „EDIT“ alebo „WATCH“, dočasne ich zabanovať, či úplne odstrániť z projektu. A jednotlivé projekty môže pridávať tagy, na základe ktorých sa potom na dashboarde dokáže lepšie orientovať. Neodmysliteľnou súčasťou systém pre kolaboratívne prototypovanie je samozrejme editor. Súčasná implementácia dovoľuje upraviť rozloženie panelov editora podľa vlastného uváženia. Používateľ dokáže svoj vytvorený prototyp editovať v spolupráci s pozvanými kolaborantami v reálnom čase pričom sa zaznamenáva aj označenie elementu inými kolaborantami čo sa používateľovi reflektuje na svojej lokálnej kópii.

Na editore bude náš tím naďalej pracovať aj počas skúškového obdobia. Na tento časový interval máme naplánované ďalšie funkcionality, ktoré máme v pláne do začiatku ďalšieho semestra zaimplementovať i zdokumentovať.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Dokumentácia k riadeniu projektu

Tím 24

Študijný program: Inteligentné softvérové systémy, Internetové technológie
Ročník: 1
Predmet: Tímový projekt
Cvičiaci: Ing. Eduard Kuric, PhD.

Obsah

Obsah Obrázkov

1 Úvod

V rámci projektovej dokumentácie k riadeniu projektu sa snažíme zachytávať spôsob akým celkový tím funguje, opis samotného tímu ako aj celkové riadenie tímu v rámci predmetu Tímový projekt. V dokumente sa zaoberáme jednotlivými rôznymi manažérskymi činnosťami, ktoré sa snažíme používať na dosahovanie stanovených cieľov. Taktiež rozoberáme používané metodiky, ktoré sme si ešte na začiatku definovali ako aj postupy pri odovzdávaní vypracovaných úloh. Okrem iného popisujeme spôsob komunikácie medzi jednotlivými členmi tímu. V kapitole 2 opisujeme jednotlivé roly členov nášho tímu a k jednotlivým manažérskym činnostiam uvádzame tiež zodpovedné osoby. V rámci rolí tímu uvádzame aj povinnosti členov tímu a taktiež požiadavky na samotný tím. V kapitole 3 opisujeme jednotlivé manažérske činnosti bližšie špecifikujeme a popisujeme ich praktické aplikovanie v procese riadenia projektu v rámci aplikácii manažmentov. Taktiež z časti načŕtame používame metodiky, ktoré ale neskôr bližšie opisujeme v prílohe A. V kapitole 4 sa nachádza sumarizácia šprintov. Uvádzame podrobné výsledky, ku ktorým sme sa dopracovali počas jednotlivých dvojtýždenných šprintov, pričom k lepšiemu prehľadu o úspešnosti prispievajú retrospektívy vypracované na konci každého šprintu, grafy z dokončených úloh ako aj celkové zhrnutie šprintu. V posledných kapitolách sa nachádza globálna retrospektíva s celkovým zhrnutím všetkých dovtedy absolvovaných šprintov.

2 Role členov tímu a podiel práce

Každý člen tímu má zodpovednosť za určitú oblasť potrebnú pri vývoji aplikácie alebo pracuje ako podpora pre niektorý z manažmentov.

2.1 Bc. Adrián Nagy

- Zimný semester
  - Vedúci tímu
  - Manažment integrácie a nasadenia produktu
  - Softvérový architekt
  - Scrum máster

- Letný semester

2.2 Bc. Lukáš Vrba

- Zimný semester
  - Dokumentrista
    - Dokument riadenia k tímovému projektu
    - Dokument k inžinierskemu dielu
    - Moduly aplikácie
  - Dodatkový programátor
  - Plánovač

- Letný semester

2.3 Bc. Peter Písecký

- Zimný semester
  - Kvalitár
  - Code review master pre php
  - Backend programátor

- Letný semester

2.4 Bc. Tomáš Mňačko

- Zimný semester
  - Hlavný tester
  - Časť dokumentácie pre testovanie
  - Code review master pre javascript

- Letný semester

2.5 Bc. Ján Kleň

- Zimný semester
    - Frontendista (UI)
    - Dizajnér
    - Code review master pre css
    - Vývojár

- Letný semester

2.6 Bc. Miloslav Smetana

- Zimný semester
    - Manažér komunikácie
    - Backend programátor
    - Validácie používateľa a projektov

- Letný semester

2.7 Bc. Michal Melúch

- Zimný semester
    - Frontendista (UX)
    - Dizajnér
    - Databázový master

- Letný semester

3 Aplikácie manažmentov a použité metodiky

Každý manažment, ktorý je spomenutý v predchádzajúcej kapitole, má svoje opodstatnenie a dôvod, prečo niektorý člen tímu má na starosti danú oblasť. Veľmi stručný popis jednotlivých manažmentov je obsahom tejto kapitoly. Jednotlivé metodiky sú súčasťou prílohy A.

3.1 Manažment vývoja

Pri vývoji aplikácie je dôležité mať osobu, ktorá je zodpovedná za celkový priebeh vývoja. V našom prípade je za celkový tok jednotlivých úloh samozrejme zodpovedný tím líder. Okrem tím lídra je v rámci každého feature, ktorý je akurát v procese vývoja zodpovedný každý člen, ktorý má pridelnú istú úlohu v rámci vývoja. Riešia sa teda veci týkajúce sa kódu, programov potrebných pre vývoj a konzistentné používanie verzií. Zodpovedná osoba teda tím líder (v tomto prípade Bc. Adrián Nagy) má skúsenosti s jazykom a prostredím, v ktorom sa aplikácia vyvíja, a teda vie poskytnúť podporu v prípade potreby. Taktiež predpripravuje jednotlivé feature do každého šprintu ako aj tasky v rámci feature, ktoré sú odkonzultované s product ownerom.

3.2 Manažment úloh

K manažovaniu úloh používame nástroj pre manažment projektov[1], ktorý využívame prostredníctvom školskej licencie zdarma. Úlohy rozdelujeme pomocou 3 abstraktných úrovní:

- Epic
    - Feature
        - Technická úloha
        - Dokumentácia
        - Testovanie
        - Bug

ScrumDesk sme prostredníctvom webhooku premostili s tímovým nástrojom pre komunikáciu[2], na ktorý prichádzajú notifikácie po každej zmene stavu, úprave zadania, resp. atribútov úlohy, či komentára. Po naplánovaní úloh sú všetky tasky pridelené istému členu tímu. Ak člen tímu začne robiť na úlohe je povinný zmeniť stav úlohy na “In Progress” a po dokončení úlohy do stavu “Done”. Tímový líder jednotlivé dokončené úlohy zosumarizuje a v prípade uváženia má právo vrátiť úlohu do stavu “Todo”, pričom do komentárov uvedie problémové časti. Podrobnejší popis sa nachádza v **metodike spracovania úloh**.

[1] ScrumDesk
[2] Slack

3.3 Manažment dizajnu a UX

Dizajn častokrát predáva produkt. Preto je vhodné mať v tíme človeka, ktorý sa do dizajnu rozumie a ovláda, aspoň základné, pravidlá pre dobrý dizajn. Vie to totiž významne ovplyvniť úspech a budúcnosť produktu. Nevyhnutnosťou je úzka spolupráca s vývojom, nakoľko je jednoduchšie dorábať postupne detaily do existujúceho dizajnu, ako ho neskôr celý prerábať. V našom tíme majú dizajn na starosti Bc. Ján Kleň a Bc. Michal Melúch. Naši UX a UI špecialisti vždy spolupracujú s návrhom riešenej úlohy s tímovým lídrom. Na stretnutiach sa rieši okrem samotného diazjnu aj predpríprava wireframov pre nasledujúce feature, v prípade potreby. Dizajn bol zostavený na základe konceptu a farebnej palety product ownera.

3.4 Manažment verziovania

V rámci vytvárania projektu je doležité samotné verziovanie projektu. V našom prípade má každý šprint osobitnú branch-u, v ktorej vieme odsledovať commity členov tímu. Takýto prístup k spôsobu verziovaniu využívame práve kvoli organízácii projektu, aby sme mali oddelené jednotlivé vyvíjané časti samostatne. Na konci šprintu po odsúhlasení product ownerom, niektorí z členov tímu zmergujú jednotlivé vyvíjané funkcionality, aby sme mali ucelené časti projektu po kope. Dovyvíjané a zároveň odsúhlasené features sú namergované do branche master, ktorá je nasadená na doméne http://test.collabui.club pre testovacie účely product ownera.

Repozitár a jeho jednotlivé verzie projektu sú uložené na verziovacom systéme[3], ktorý je prostredníctvom webhooku premostený s tímovým slackom, na ktorý prichádzajú notifikácie po každom pushnutí nového commitu. K verziovaniu používame voľne dostupnú aplikáciu[4], ktorej použitie popisujeme bližšie v **metodike verziovania**.

3.5 Manažment testovania

V rámci nášho produktu sa riadime spôsobom TDD (Test Driven Development). Od začiatku je teda potrebné myslieť na dostatočné testy s čo najvyšším možným pokrytím. V rámci testovania sa vykonávajú najmä unit testy pre php a javascript. Každá úloha je pred odsúhlasením v dostatočnej miere pokrytá testami, pokiaľ si to jej character vyžaduje. Náš špecialista na testovanie Bc. Tomáš Mňačko dohliada na uvedené skutočnosti. Je nutné pred každým pushnutím implementovaných zmien otestovať doposiaľ realizované funkčnosti, čo medzi inými popisujeme v **metodike testovania**.

3.6 Manažment dokumentácie

Keďže vývíjame aplikáciu je potrebné okrem jej testovania a organizácie kvality kódu vytvoriť aj kvalitnú dokumentáciu. V rámci dokumentácie sa pri každom feature rieši

[3] Bitbucket
[4] SourceTree

analýza problémovej oblasti ako aj zameranie používateľa, návrh riešenia ako aj samotná implemetácia v podobe technickej dokumentácie. V prípade potreby opisujeme jednotlivé implemetačné časti prostredníctvom diagramov. Na tvorbu diagramov používame nástroj Enterprise architekt. Pre inšpiráciu správnej tvorby databázových modelov používame návrhy diagramov z nástroja DBeaver prepísanú do logického modelu v EA.

Okrem technickej dokumentácie je potrebné spisovať aj dokument riadenia tímového projektu. V tomto dokumente využívame skutočnosti spísané z jednotlivých zápisníc. Taktiež na zhodnotenie šprintu využívame burndown grafy z nástroja scrumdesk, v ktorom riešime organizáciu tímov a jednotlivých úloh. Taktiež využívame tento nástroj na tvorbu retrospektívy, keďže v ňom spisujeme náše pocity a názory pre konkrétny šprint. Celkový manažment dokumentácie má na starosti Bc. Lukáš Vrba.

3.7 Manažment podporných nástrojov a komunikácie

V rámci manažmentu podporných nástrojov ako už bolo spomínané používame nástroj Enterprise Architekt na dokumentáciu. Pre ukladanie jednotlivých dokumentov používame okrem školského servera teda našej tímovej stránky aj Google Drive.

Na komunikáciu navzájom využívame vyššie spomínanú aplikáciu Slack, pričom sme vytvorili 10 vlákien. Nasledovné vlákna sú:

- bitbucket: webhook z bitbucketu
- scrumdesk: webhook zo scrumdesku
- documents: vlákno pre šablóny, zápisky a dodatočné časti dokumentácie
- from_ed: vlákno pre product ownera
- from_leader: vlákno pre tím lídra
- ux_ui: vlákno pre riešenie dizajnových problémových oblastí
- help_me: vlákno pre požiadanie o pomoc od tímu
- problems: privátne vlákno tímu pre riešenie problémov
- general a random: defaultné vlákna

3.8 Manažment kvality kódu

Jednou z dôležitých vecí týkajúcich sa správy kódu je dodržiavanie istých konvencii. Je to práve z dôvodu aby sa nezničovala kvalita kódu, čitatelnosť a aby sa predišlo nezrozumitelnostiam pri vyvíjaní ostatných členov tímu. Inakšie povedané aby sa ostatní členovia tímu nestrácali v kóde. Na zabezpečenie vyššej kvality kódu bolo nutné začať vykonávať striktnejšie code review, pri ktorých sa nekontrolovala len funkčnosť ale aj dodržiavanie konvencií písania kódu. Počas šprintu sme vykonávali jednotlivé code review v rámci php, css a javascriptu. Tieto review majú na starosti Bc. Peter Písecký pre php, Bc. Tomáš Mňačko pre javascript a Bc. Ján Kleň pre css. Problémové oblasti boli akútne riešené a rovno implementované. Proces odhalovania problémových oblastí je spisaný v **metodike písania kvalitného kódu**.

3.9 Manažment webu tímu a produkčného prostredia

Manažment webovej prezentácie má na starosti Bc. Miloslav Smetana, ktorý v spolupráci s Bc. Lukášom Vrbom aktualizujú najnovší obsah tímového projektu. V rámci webovej stránky sme sprístupnili okrem samotných dokumentov aj informácie o nás ako tíme a najmä o projekte, na ktorom pracujeme. Po každom šprinte vykonávame aktualizáciu tímovej stránky, medzi inými pridanie dokumentov, čo popisujeme v **metodike nasadzovania**.

4 Povinnosti a požiadavky členov tímu

V rámci jednotlivých podkapitol uvádzame technické ako aj manažerské role členov tím. Každý člen tímu spísal povinnosti, ktoré si definoval, ako aj požiadavky, ktoré očakáva od ostatných členov.

4.1 Bc. Peter Písecký

Počiatočne stanovené role v tíme: kvalitár, code review master PHP

4.1.1 Povinnosti a požiadavky na tím

Kvalitár – Povinnosti: Ako kvalitár bude súčasťou náplne mojej práce dohliadať na to, že kvalita výstupu nášho projektu bude v súlade s požiadavkami vlastníka produktu, ako aj na to, že bude na úrovni zodpovedajúcej nášho stupňa vzdelania.

Kvalitár – Požiadavky na tím: Od ostatných členov tímu čakám, že budú ku svojej práci pristupovať zodpovedne, preukážu snahu naplniť požiadavky vlastníka produktu, budú nasledovať osvedčené postupy, ako aj metodiky ostatných členov tímu zodpovedných za jednotlivé časti implementovaného riešenia.

Code review master PHP – Povinnosti: Ako vedúci kontroly kódu v jazyku PHP bude mojou povinnosťou zabezpečiť pravidelnú kontrolu tých častí produktu, ktoré sú implementované v jazyku PHP.

Code review master PHP – Požiadavky na tím: Od ostatných členov tímu očakávam, že výstupom ich snahy bude implementácia v jazyku PHP zodpovedajúca ich schopnostiam, ako aj požiadavkám vlastníka produktu. V rámci projektu by sa členovia tímu mali snažiť dodržiavať programovacie konvencie zavedené skupinou Framework Interoperability Group. Predovšetkým sa budeme riadiť štandardom PSR-2, ale aj dodatočnými pravidlami, ktoré sú uvedené v dokumentácii CakePHP. Pre validáciu dodržiavania konvencií budeme používať nástroj CakePHP Code Sniffer.

4.2 Bc. Michal Melúch

Počiatočne stanovené role v tíme: správca databázy, UX dizajnér

4.2.1 Povinnosti a požiadavky na tím

Správca databázy – Povinnosti: Ako správca databázy je mojou úlohou dohliadať na bezproblémový chod všetkých databázových systémov použitých v projekte. Rovnako je mojou úlohou dbať na ich správnu konfiguráciu, efektivitu a bezpečnosť. Za účelom zachovania prehľadnosti by som chcel poprosiť členov tímu aby k novým tabuľkám a stĺpcom (ne)relačných databáz pripájali aspoň krátke komentáre (viď súbor collabui.sql).

Správca databázy – Požiadavky na tím: Od každého člena tímu očakávam zodpovedný prístup k databázam a údajom, s ktorými budú v rámci nich narábať.

UX dizajnér – Povinnosti: Úlohou UX dizajnéra je navrhnúť dizajn aplikácie tak, aby priviedol používateľa k vopred stanovenému cieľu bez akéhokoľvek problému. Návrh musí byť nielen estetický ale aj funkčný a použiteľný – preto budem často v priamom kontakte s UI dizajnérom.

UX dizajnér – Požiadavky na tím: Každý člen tímu, ktorý pracuje na grafickom používateľskom rozhraní projektu by mal byť aspoň zbežne oboznámený s jeho UX (a UI) guidelines a styleguides. Tiež by som chcel poprosiť každého člena nech ma kontaktuje v prípade, že si všimne akýkoľvek nedostatok týkajúci sa UX.

4.3 Bc. Ján Kleň

Počiatočne stanovené role v tíme: UI(frontend), CSS code review

4.3.1 Povinnosti a požiadavky na tím

UI(frontend) – Povinnosti: Ako frontendista mám za úlohu starať sa o to aby boli dodržiavané určité konvencie čo sa týka celkového štýlu, používateľského rozhrania a farebnej palety aplikovanej na stránke. Taktiež mám povinnosť aktívne komunikovať spolu s Michalom Melúchom ohľadne UX.

UI(frontend) – Požiadavky na tím: Od každého člena tímu očakávam, že bude dodržiavať zadaný style guide viď. Obr.1. Pričom primárne farby sú zelené najmä svetlejšia zelená. Logá su orientované na otiene zelených a žltej farby. Bolo by teda fajn orientovať sa na tieto farby poprípade použiť čiernu/šedú/bielu ako univerzálne farby.

Technológie: Bootstrap 4, SASS

SCSS code review – Povinnosti: Mojou úlohou je dbať na dobre štrukturovaný a priehľadný kód.

SCSS code review – Požiadavky na tím: Očakávam, že ten kód budeme písať ako ľudia nech sa v tom vieme orientovať. Tiež by som ocenil ku každej relevantnej časti kód v angličtine nech vieme čo, ako a prečo.

4.4 Bc. Lukáš Vrba

Počiatočne stanovené role v tíme: dokumentarista, plánovač

4.4.1 Povinnosti a požiadavky na tím

Dokumentarista – Povinnosti: V rámci role dokumentarista, budem vytvárať projektovú dokumentáciu, motivačný dokument a predpripravovať jednotlivé šablóny dokumentácii pre tím. Samotná projektová dokumentácia sa bude skladať z inžinierskej časti a z časti riadenia vývoja. Samozrejme dokument bude obsahovať analýzu, návrh, implementáciu a testovanie. Okrem iného budem dozerať na aktualizáciu tímovej stránky. Jednotlivé časti budem dokumentovať v programe Enterprise architekt.

Dokumentarista – Požiadavky na tím: V rámci tímu by som chcel požiadať jednotlivých členov, ktorí budú programovať konkrétne časti projektu, aby bol kód riadne zdokumentovaný, teda aby bolo možné časť dokumentácie vytvárať aj priamo z kódu. Taktiež by som poprosil každého člena po vykonaní istých zmien, presne popísať jednotlivé menené časti, poprípade aj zámer, ktorý spôsobil tieto zmeny. Okrem týchto požiadaviek by som mal ešte na záver jednu a teda v prípade nezrozumiteľnosti, by som požadoval jednotlivé konzultácie k riešenému problému s členom, ktorý vykonával zmeny.

Plánovač – Povinnosti: - Ako projektový plánovač budem zaznamenávať jednotlivé úlohy v tíme a motivovať tím k dodržiavaniu termínov pre dodanie požiadaviek ako aj notifikovať jednotlivých členov tímu k svojim povinnostiam, v prípade, že by na ne zabúdali.

Plánovač – Požiadavky na tím: Najdôležitejšou požiadavkou pre každého člena je hlavne dodržiavanie termínov svojich povinností.

4.5 Bc. Tomáš Mňačko

Počiatočne stanovené role v tíme: hlavný tester, code review JS

4.5.1 Povinnosti a požiadavky na tím

Hlavný tester – Povinnosti: Vytvorenie príkladu použitia testov a dohliadanie na správnu tvorbu testov ku každej časti kódu.

Hlavný tester – Požiadavky na tím: Zodpovedný prístup ku písaniu zmysluplných testov. Dodržanie aspoň 75% pokrytie testami v celom projekte. Ako písať testy je uvedené v súbore Testovanie.

Code review JS – Povinnosti: Založenie základných konvencií pri písaní kódu ako odsádzanie, používanie komentárov a podobne. Kontrola JS kódov, teda či sú dodržané pravidlá a prehľadnosť kódu, či je kód riadne otestovaný.

Code review JS – Požiadavky na tím: Dodržiavanie konvencií a písanie prehľadného kódu(ideálne dobre štruktúrovaný kód bez potreby komentárov) podľa vopred zadefinovaných pravidiel.

4.6 Bc. Miloslav Smetana

Počiatočne stanovené role v tíme: manažér komunikácie

4.6.1 Povinnosti a požiadavky na tím

Manažér komunikácie – Povinnosti: Ako manažér komunikácie tímu mám za úlohu zisťovať všetky potrebné informácie, ktoré sú nutné na splnenie všetkých požiadaviek, ktoré sú kladené na celkový vývoj tímového projektu. Taktiež mám za úlohu korigovať celú komunikáciu v tíme a riešiť prípadné problémy, ktoré môžu nastať v rámci nedostatočnej alebo zlej komunikácie. V neposlednom rade musím dohliadať na vybavovanie všetkých potrebných náležitostí, ktoré môžu ovplyvniť vývoj aplikácie, ako napríklad vybavenie licencií pre určitý softvérový nástroj.

Manažér komunikácie – Požiadavky na tím: Od každého člena tímu očakávam komunikatívnosť, nakoľko práve komunikatívnosť je jedným z hlavných aspektov agilného vývoja softvéru a taktiež je nutné v prípade, akýchkoľvek problémov komunikovať v dostatočnom predstihu. Ďalej požadujem od každého člena tímu, aby mi poskytoval všetky potrebné informácie, ktoré budem v priebehu vývoja od každého požadovať pri vybavovaní rôznych požiadaviek.

4.7 Bc. Adrián Nagy

Počiatočne stanovené role v tíme: vedúci tímu, scrum master, softvérový architekt

4.7.1 Povinnosti a požiadavky na tím

Vedúci tímu – Povinnosti: Ako vedúci tímu mám za úlohu koordinovať tím, riešiť prípadné problémy, ktoré nastanú pri plánovaní alebo implementácii. Mojou úlohou je taktiež komunikácia s tzv. Product Owner-om a naplnenie požiadaviek v rámci možností.

Vedúci tímu – Požiadavky na tím: Od každého člena tímu očakávam seriózny prístup a hlavne plnenie svojich povinností, ktoré sú definované v dokumentoch *meno_priezvisko_metodika.pdf*.

Scrum master – Povinnosti: Základnou povinnosťou scrum master-a je dbať na pravidelné dodržiavanie tímových rituálov a sledovanie Burn-Down Chart-u. Okrem týchto aktivít si beriem na starosť moderovanie stretnutia RETRO.

Scrum master – Požiadavky na tím: Chcel by som poprosiť každého člena aby prichádzali na tímové stretnutia pozitívne naladení, svoje obavy včas vyslovili a nebáli sa konfontrácií.

Softvérový architekt – Povinnosti: Ako softvérový architekt mám za úlohu nastoliť počiatočnú architektúru, pevný základ s vhodnými technológiami, na ktoré neskôr vybudujeme webové riešenie produkčnej kvality. Túto skutočnosť a ďalšie implementačné pravidlá, praktiky nájdete v zaheslovanom dokumente *architektura.pdf*. Architektúra sa samozrejme bude meniť a prispôsobovať počas implementácie, pričom vzniknú ďalšie verzie dokumentu, určené najmä pre členov tímu.

Softvérový architekt – Požiadavky na tím: Od každého člena tímu vyžadujem aby dodržiavali skutočnosti uvedené vo vyššie uvedenom dokumente. Komunikovať budem hlavne s kvalitárom a s členmi vykonávajúcimi code review, ktorí dohliadnu na dodržiavanie implementačných pravidiel a praktík.

5 Sumarizácie šprintov

V tejto kapitole sú opísané jednotlivé šprinty. Obsahujú informácie o feature a v rámci nich jednotlivých taskov, ktoré sme v riešili, na čo nové sme v rámci tímu prišli, resp. na čom sme sa od minulého šprintu snažili procovať. V časti retrospektíva sú pri každom šprinte vymenované všetky záležitosti, ktoré sme počas šprintu objavili a chceme s nimi niečo robiť alebo ich istým spôsobom modifikovať. Retrospektívu relizujeme technikou “Sad-Glad-Mad”.

5.1 Zimný semester

5.1.1 Šprint1 (3.10.2017 - 18.10.2017) Birell nealko

Hlavnou úlohou šprintu bolo zmapovať predpripravený základ projektu. V jednotlivých prípadoch boli vykonávané aj úpravy backendu, aby sa v rámci projektu lepšie orientovalo a takisto aby bola možná lepšia a rýchlejšia integrácia nových funkcii alebo modulov.

V tomto šprinte sme riešili taktiež jednotlivé organizačné úlohy v rámci tímu ako monitoring času stráveného na projekte alebo spôsob práce s taskami vo webovej aplikácii Scrumdesk.

V rámci prípravy procesov team leader Bc. Adrián Nagy pripravil základ aplikácie. Príprava pozostávala hlavne z úpravy backendu a naplánovania jednotlivých technológie pre budúcu integráciu.

Celkový počet úloh v šprinte: 12

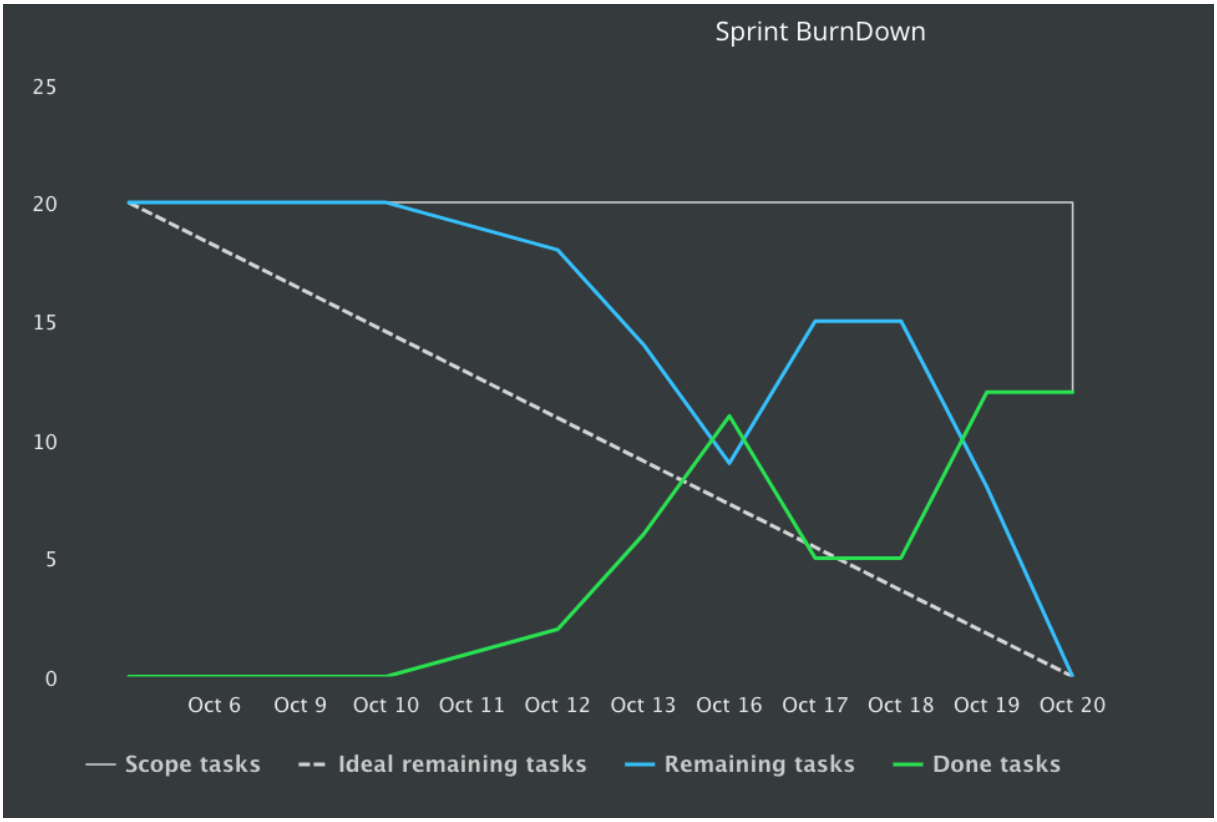
Každý task obsahuje dokumentáciu a testovanie.

5.1.1.1 Prehľad práce členov tímu

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Úvodná stránka	3	2	Ján Kleň
			Michal Melúch
			Adrián Nagy
			Lukáš Vrba
Registrácia a aktivácia účtu	3	5	Adrián Nagy
			Peter Písecký
			Miloslav Smetana
Prihlásenie a odhlásenie sa zo systému	2	2	Adrián Nagy
			Peter Písecký
			Miloslav Smetana
			Peter Písecký
Zabudnuté a nové heslo	4	2	Peter Písecký
			Michal Melúch

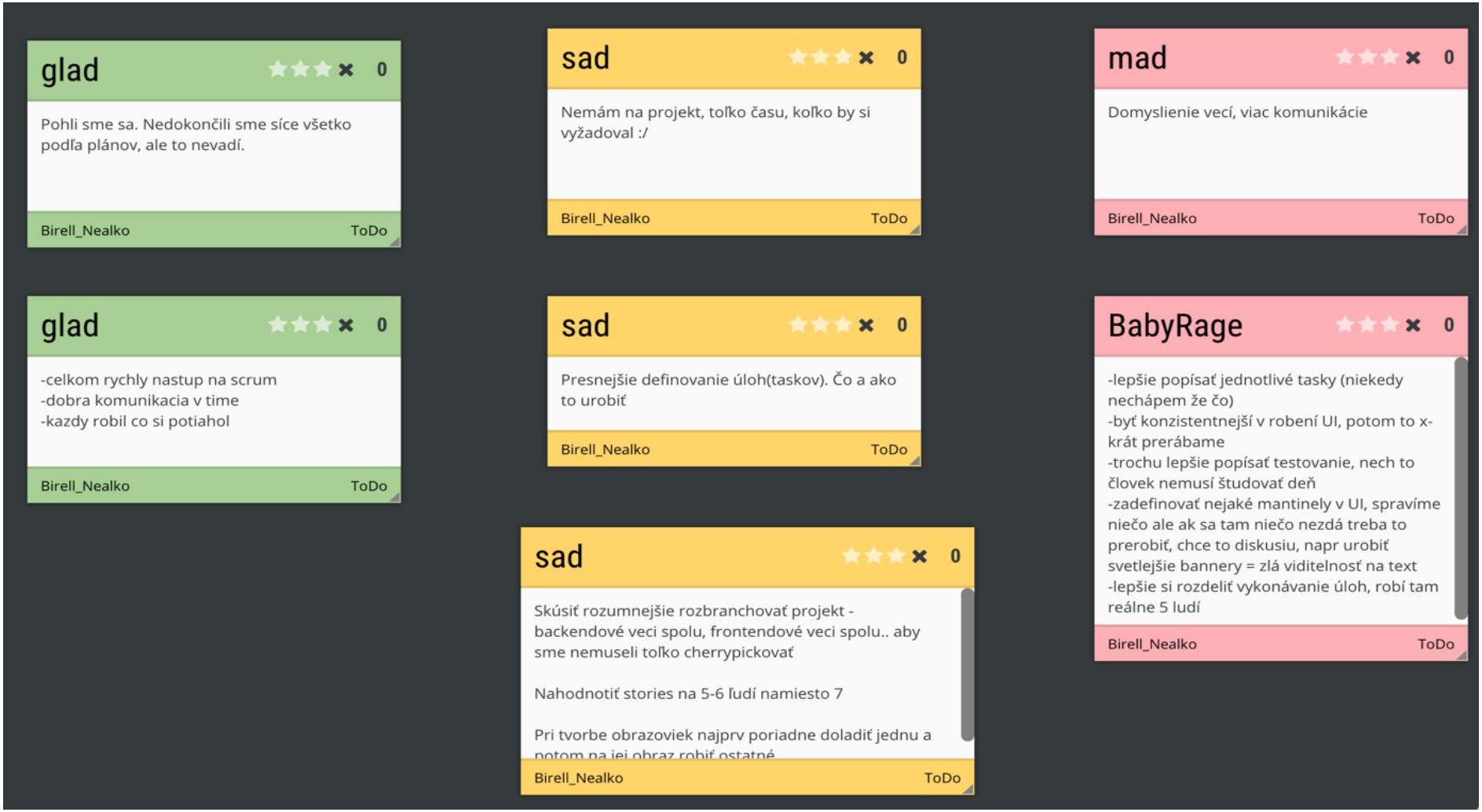
5.1.1.2 Zhrnutie

V rámci zhrnutia uvádzame burndown chart pre uvedený šprint.



Obrázok 1 - Burndown chart pre prvý šprint

5.1.1.3 Retrospektíva



Obrázok 2 - Retrospektíva prvého šprintu

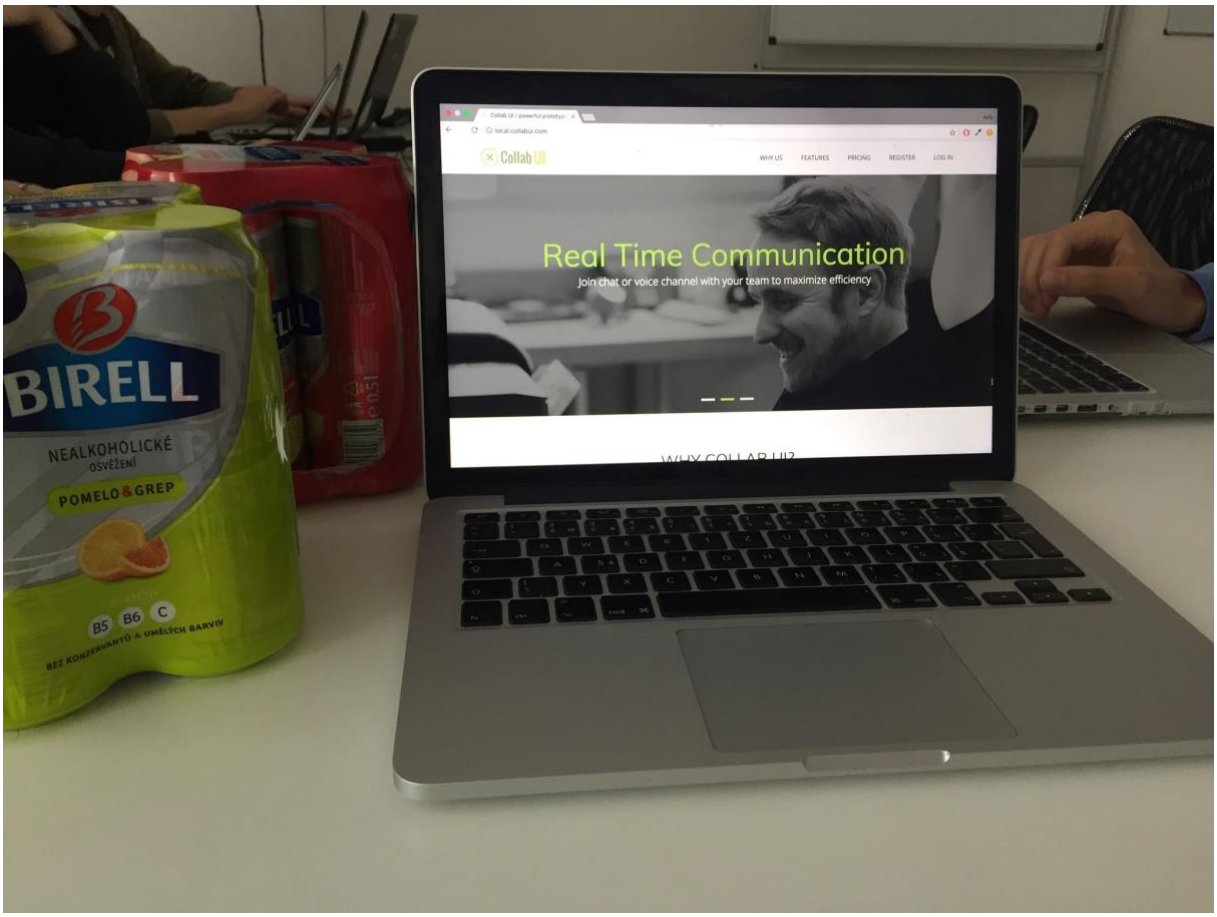
5.1.2 Zhodnotenie šprintu

Ako sme už z burndown chartu mohli zistiť zo začiatku nám chvíľu trvalo kým sme sa zosúladili. Keďže jednotliví členovia tímu sa zoznamovali s predpripraveným základom tak sa backlog itemy začali hýbať až koncom šprintu resp. vtedy sa podarilo úplne dorobiť nasledujúce backlog itemy:

- Úvodná stránka

Čo sa týka ostatných itemov, tie sme museli preniesť do ďalšieho šprintu. Tieto itemy boli už načaté a aj jednotlivé časti boli dokončené, no napriek tomu sme museli v rámci projektu prehodnotiť niekoľko skutočností tesne pred koncom šprintu.

Ďalej uvádzame tímový obrázok pre daný šprint reprezentovaný metodikou “Birell nealko”.



Obrázok 3 - Logo prvého šprintu

5.1.3 Šprint2 (20.10.2017 - 3.11.2017) StrongBow_Cider

V tomto šprinte sme sa sústredili už viac špecifickejšie na časť manažmentu projektov. Keďže v minulom šprinte nás vcelku držala úprava backendu a samotné predpripravenie projektu pre jednotlivé náročnejšie časti rôznych budúcich funkcionalít ako aj zaintegrovanie nových technológií, sme boli nútení preniesť niektoré feature aj do tohto šprintu. Spomínané feature sú nasledovné:

- Registrácia a aktivácia účtu
- Prihlásenie a odhlásenie sa zo systému
- Zabudnuté a nové heslo

Samozrejme sme sa začali zaoberať už samotným rozhraním aplikácie, kde bolo nutné vymyslieť, navrhnúť a samozrejme aj implementovať spôsob akým používateľ bude môcť vytvárať vlastné projekty a taktiež akým štýlom bude môcť prídávať iných používateľov do vlastného projektu a teda z nich spraviť kolaborantov pre konkrétny projekt.

Celkový počet úloh v šprinte: 12

Všetky tasky obsahujú dokumentáciu až na nasledovné:

- Vytváranie projektov a pridávanie kolaborantov

Všetky tasky obsahujú testovacie scenáre až na nasledovné:

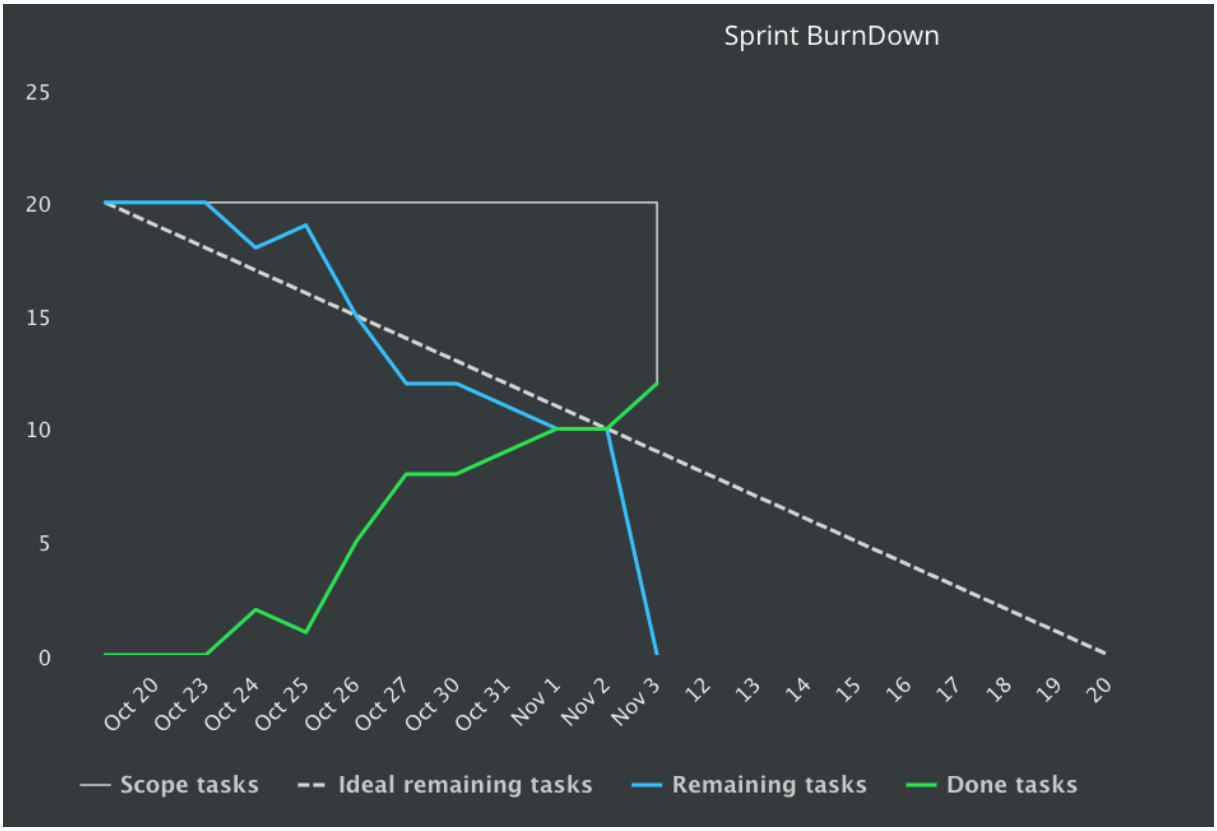
- Zabudnuté a nové heslo
- Vytváranie projektov a pridávanie kolaborantov

5.1.3.1 Prehľad práce členov tímu

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Vytváranie projektov a pridelenie kolaborantov	4	8	Adrián Nagy
			Michal Melúch
			Peter Písecký
Registrácia a aktivácia účtu	3	5	Tomáš Mňačko
			Peter Písecký
			Lukáš Vrba
Zabudnuté a nové heslo	2	2	Peter Písecký
			Lukáš Vrba
Prihlásenie a odhlásenie sa zo systému	3	2	Peter Písecký
			Lukáš Vrba
			Ján Kleň

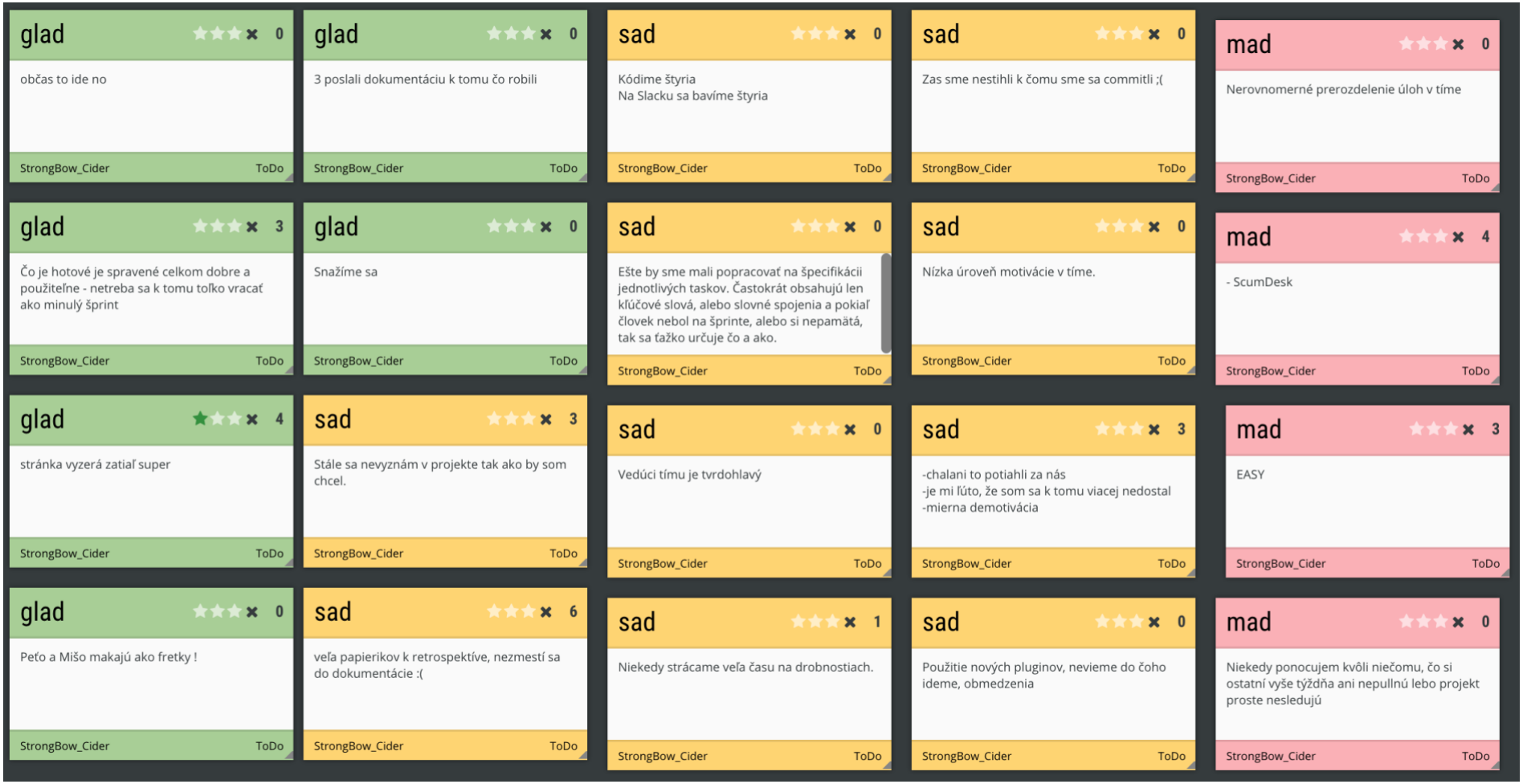
5.1.3.2 Zhrnutie

V rámci zhrnutia uvádzame burndown chart pre uvedený šprint.



Obrázok 4 - Burndown chart pre druhý šprint

5.1.3.3 Retrospektíva

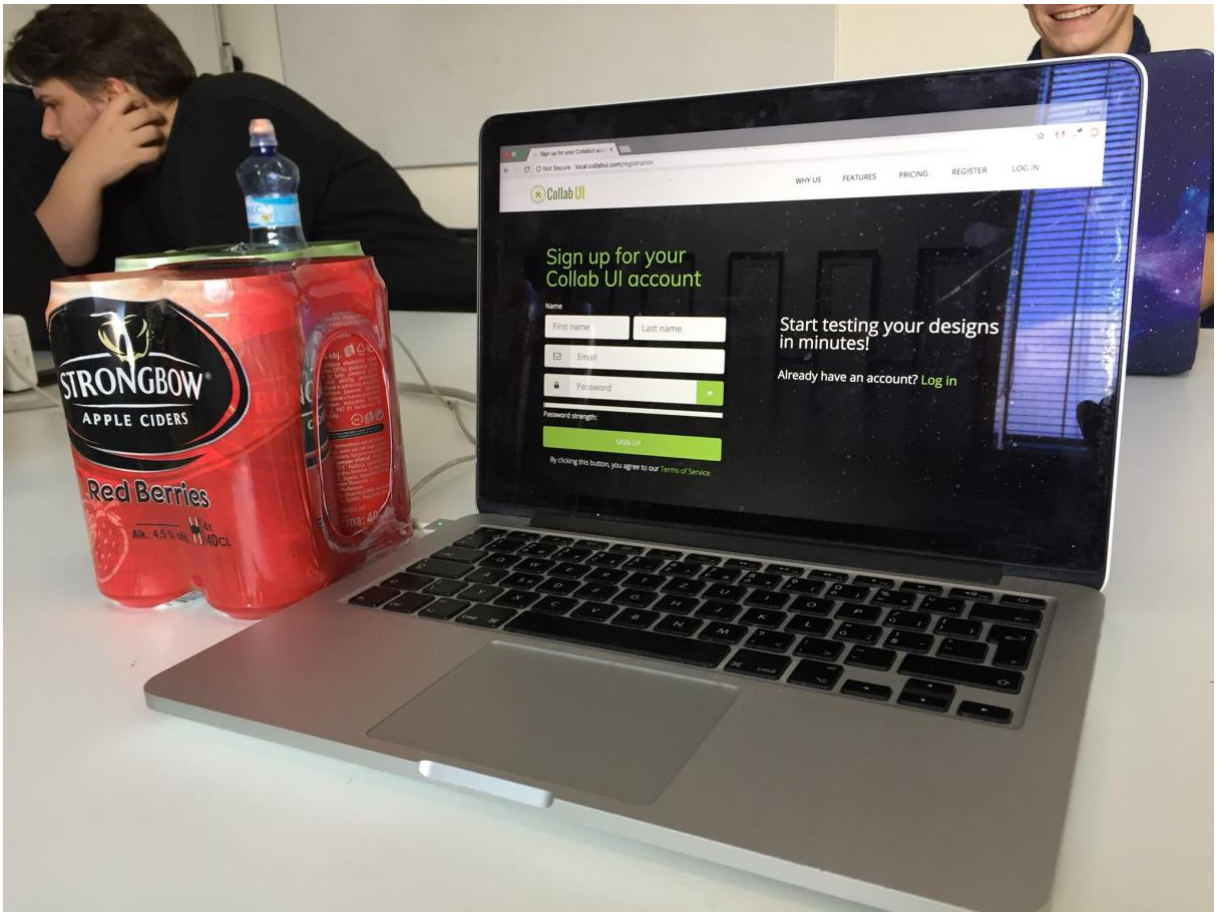


Obrázok 5 - Retrospektíva k druhému šprintu

5.1.4 Zhodnotenie šprintu

Ako už môže byť jasné z uvádzaného burndown chartu, už zo začiatku sa nám podarilo spraviť niekoľko úloh, väčšinu úloh sme riešili práve v polovici a na konci šprintu. Rozhodne môžeme povedať, že oproti minulému šprintu sa nám dokončenie jednotlivých úloh podarilo lepšie rozvrhnúť. Na druhej strane malo za následok aj fakt, že sme z prvého šprintu niektoré úlohy preniesli. Tieto úlohy už boli načrnuté alebo skoro hotové teda náročnosť ich dokončenia nebula až tak vysoká. Na jednej strane sme radi, že tieto úlohy boli rýchlo dokončené a mohli sme ísť riešit ďalšie úlohy v poradí. Na druhej strane strane zvyšok kompetentných úloh z danému šprintu sa nám podarilo dorobiť až na konci tohto šprintu, teda rezerva nám znova neostala žiadna, no napriek tomu bol šprint úspešnejší keďže sme už z prvého šprintu mali predpripravený základ. Najnáročnejšiu úlohu tohto šprintu sme ale museli znova rozdeliť a časť preniesť do ďalšieho šprintu.

Ďalej uvádzame tímový obrázok pre daný šprint reprezentovaný metodikou “StrongBow_Cider”.



Obrázok 6 - Logo druhého šprintu

5.1.5 Šprint3 (3.11.2017 - 17.11.2017) Plzeň

V rámci tohto šprintu sme sa sústreďovali na samotný manažment kolaborantov pracujúcich na projekte. Z minulého šprintu nám ale zvýšila nasledujúci feature:

- Vytváranie projektov a pridelenie kolaborantov

Taktiež sme sa snažili zaintegrovať nové technológie, spraviť refactoring a úpravu backendu ako aj rozčleniť manažment projektov a kolaborantov do samostatných boxov pomocou frameworku zo semestrálnej práce Aoweb. Táto technológia bola namapovaná na projekt už v druhom šprinte ale zakomponovanie jednotlivých úloh a jej využívanie sa týkalo aj tohto šprintu.

Okrem manažmentu kolaborantov sme ešte riešili notifikovanie kolaborantov pri zmazaní projektu, jeho zmene práv ako aj autodopĺňanie zoznamu používateľov pri pozvaní do projektu.

Celkový počet úloh v šprinte: 23

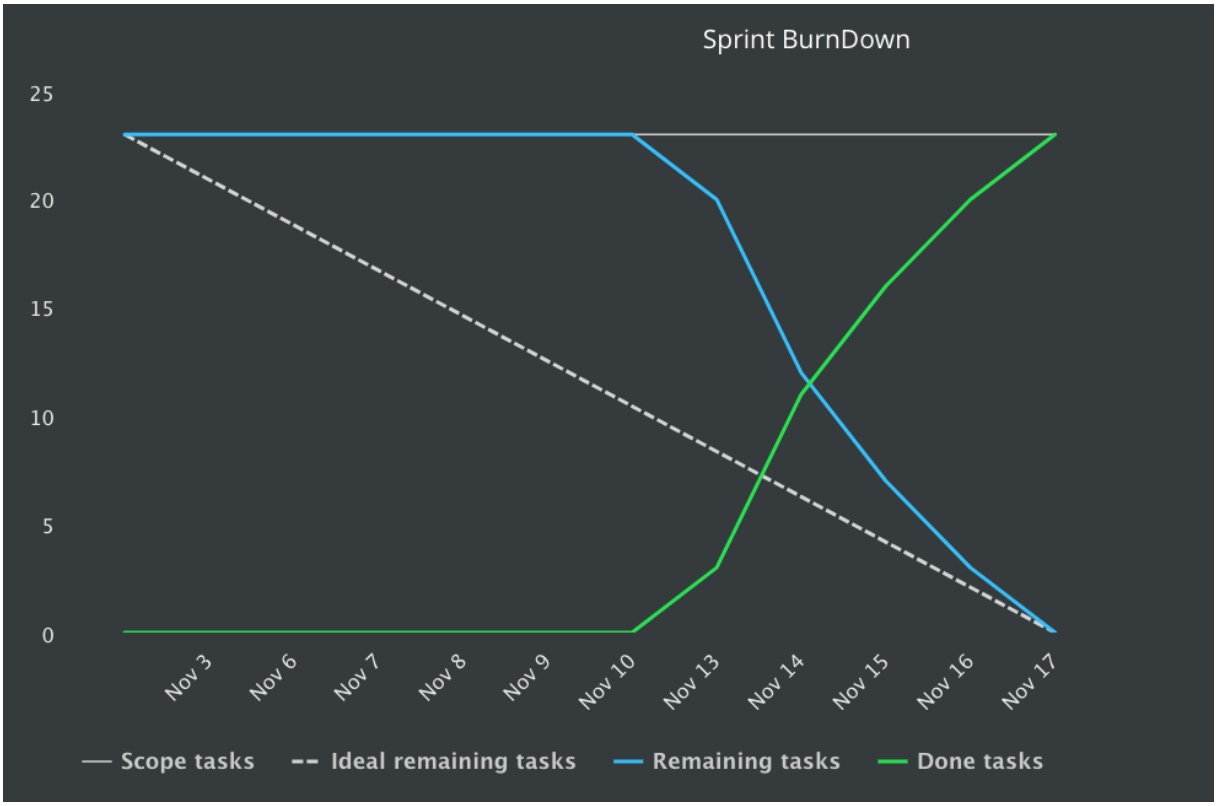
Každý task obsahuje dokumentáciu a testovanie.

5.1.5.1 Prehľad práce členov tímu

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Vytváranie projektov a pridelenie kolaborantov	9	5	Adrián Nagy
			Michal Melúch
			Lukáš Vrba
			Miloslav Smetana
			Ján Kleň
			Tomáš Mňačko
			Peter Písecký
Manažment kolaborantov na projekte	9	5	Adrián Nagy
			Peter Písecký
			Michal Melúch
			Tomáš Mňačko
			Lukáš Vrba
Mazanie projektov	5	3	Peter Písecký
			Adrián Nagy
			Lukáš Vrba
			Ján Kleň

5.1.5.2 Zhrnutie

V rámci zhrnutia uvádzame burndown chart pre uvedený šprint.



Obrázok 7 - Burndown chart pre tretí šprint

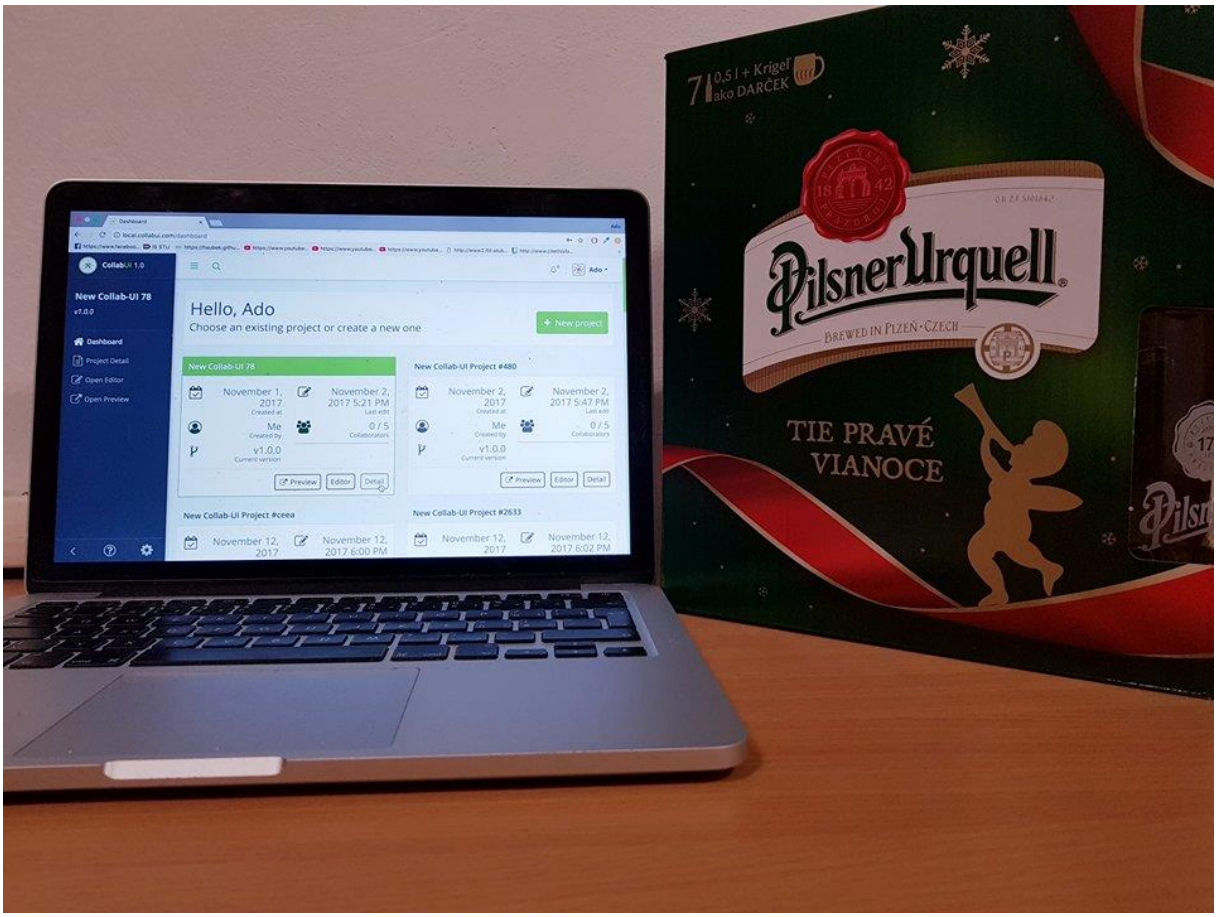
5.1.5.3 Retrospektíva

Obrázok 8 - Retrospektíva pre tretí šprint

5.1.6 Zhodnotenie šprintu

V rámci tohto šprintu sa nám podarilo kompletne uzatvoriť zmenu práv kolaboranta ako aj jeho pridelenie a odstránenie. Kompletne sa nám podarilo uzatvoriť základný manažment projektov. Na začiatku šprintu sa polovica tímu zaoberala ešte dotváraním jednotlivých funkcií pridelenia kolaborantov ešte z minulého šprintu. Vo veľkom sa dokončovalo autodoplňovanie zoznamu kolaborantov. Taktiež sa ešte dotvárali jednotlivé funkcie v rámci oddelenia kolaborantov od projektov v samostatných boxoch. Taktiež sme implementovali notifikácie prostredníctvom emailu a dotvorili notifikáciu zmazania projektu. Ako je vidno na burndown charte vzhľadom k omnoho väčšiemu poštu taskov oproti ostatným šprintom sme sa vrátili k rovnakým výsledkom z prvého šprintu. Teda väčšina taskov sa podarilo dokončiť až koncom šprintu. Na rozdiel od ostatných šprintov sa nám v tomto aj napriek skoro dvojnásobnému počtu taskov podarilo všetky dokončiť a teda žiaden feature nebolo treba rozdeľovať a prenášať do ďalšieho šprintu, čo nás veľmi teší.

Ďalej uvádzame tímový obrázok pre daný šprint reprezentovaný metodikou “Plzeň”.



Obrázok 9 - Logo tretieho šprintu

5.1.7 Šprint4 (17.11.2017 - 1.12.2017) Jack_Daniels

V rámci tohto šprintu sme sa sústreďovali na prototyp editora bez kolaborantov. Čo sa týka minulého šprintu nepreniesli sme ani jeden feature, teda sme sa mohli naplno sústrediť na šprint Jack Daniels a jednotlivé tasky v rámci neho. Najväčšia časť šprintu tvorila najmä konfigurácia a implementácia panelov obsahujúcich komponenty a nastavenia pre vytváranie resp. frontendových častí webových stránok, pri ktorej sme zistili niektoré obmedzené funkcie. Práve aj vďaka novo objaveným problémom v rámci už spomínanej implementácie sme boli nútení spraviť menšie zmeny.

Okrem konfigurácie sme riešili aj vytvorenie spojenia medzi klientom a serverom, ktorý má slúžiť ako základ pre budúce integrácie nových funkcionalít v rámci vizuálnej prezentácie zmien projektov v reálnom čase.

Ďalším aspektom, ktorým sme sa zaoberali je základná šablóna pri prvotnom zobrazení prototypu editora. Používateľ má takýmto štýlom možnosť vidieť základné komponenty pre vytváranie a upravovanie projektu.

Celkový počet úloh v šprinte: 12

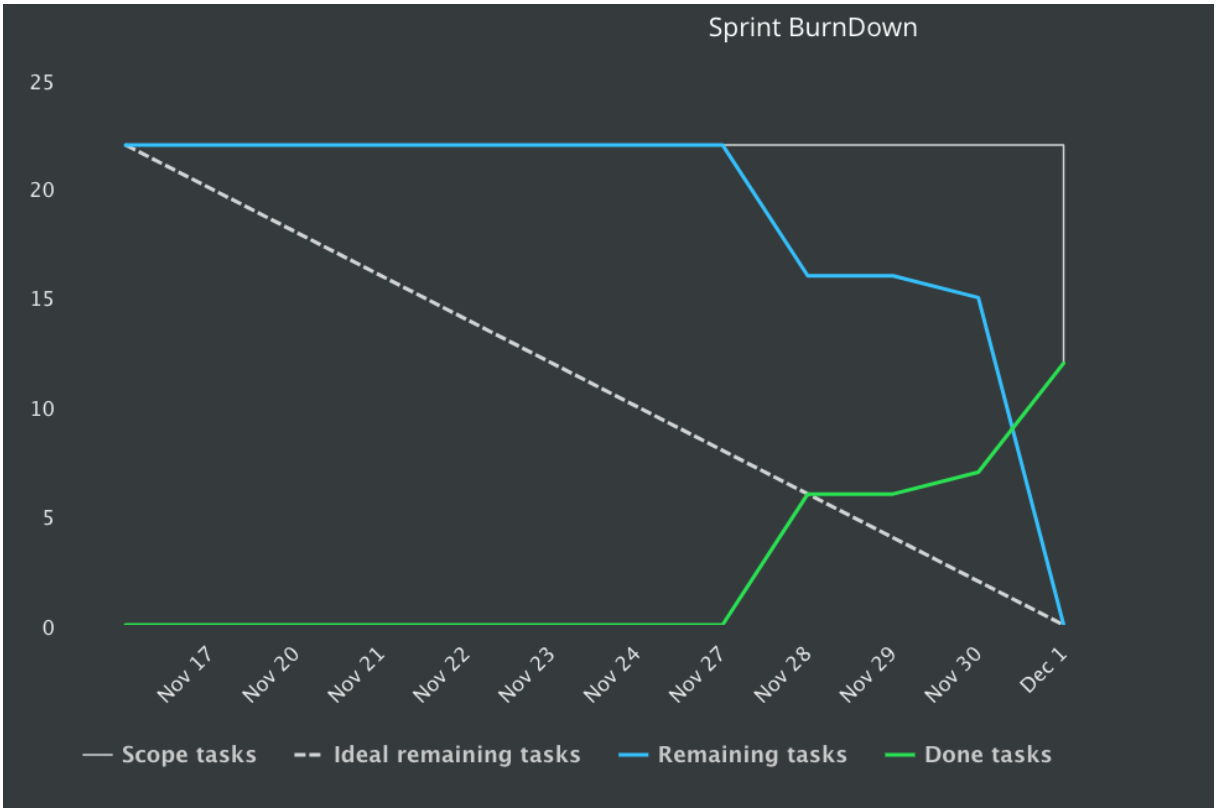
Každá feature obsahuje dokumentáciu a testovanie.

5.1.7.1 Prehľad práce členov tímu

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Editácia projektu bez kolaborantov	12	13	Adrián Nagy
			Michal Melúch
			Lukáš Vrba
			Miloslav Smetana
			Ján Kleň
			Tomáš Mňačko
			Peter Písecký

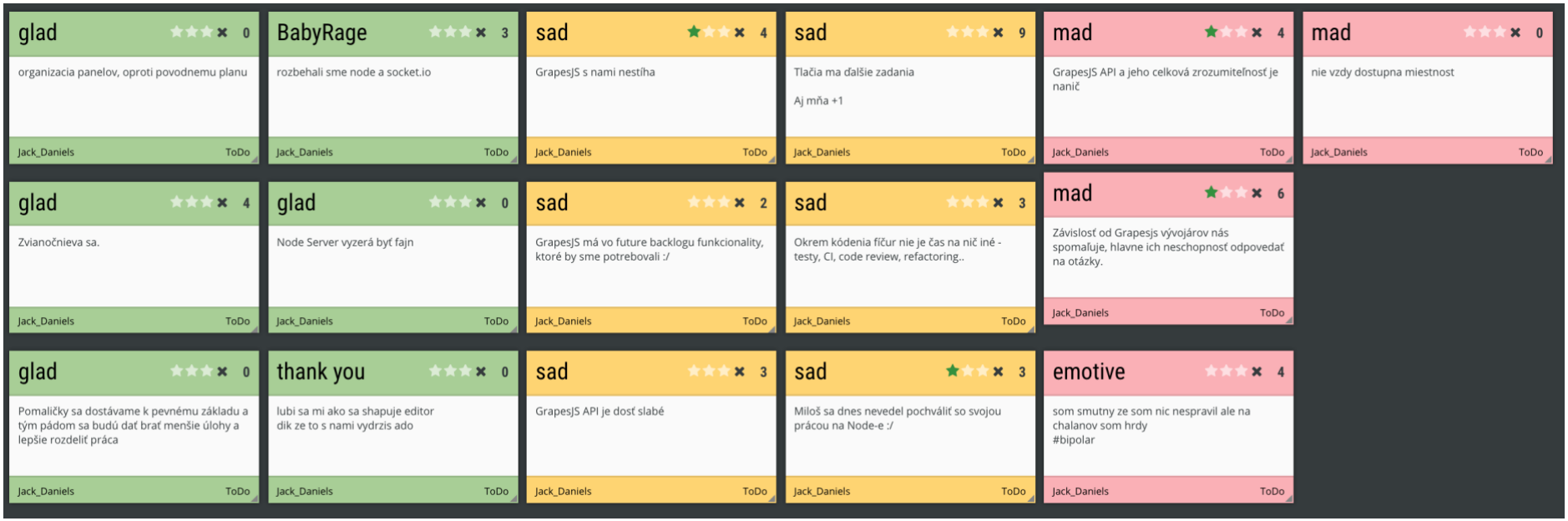
5.1.7.2 Zhrnutie

V rámci zhrnutia uvádzame burndown chart pre uvedený šprint.



Obrázok 10 - Burndown chart pre štvrtý šprint

5.1.7.3 Retrospektíva

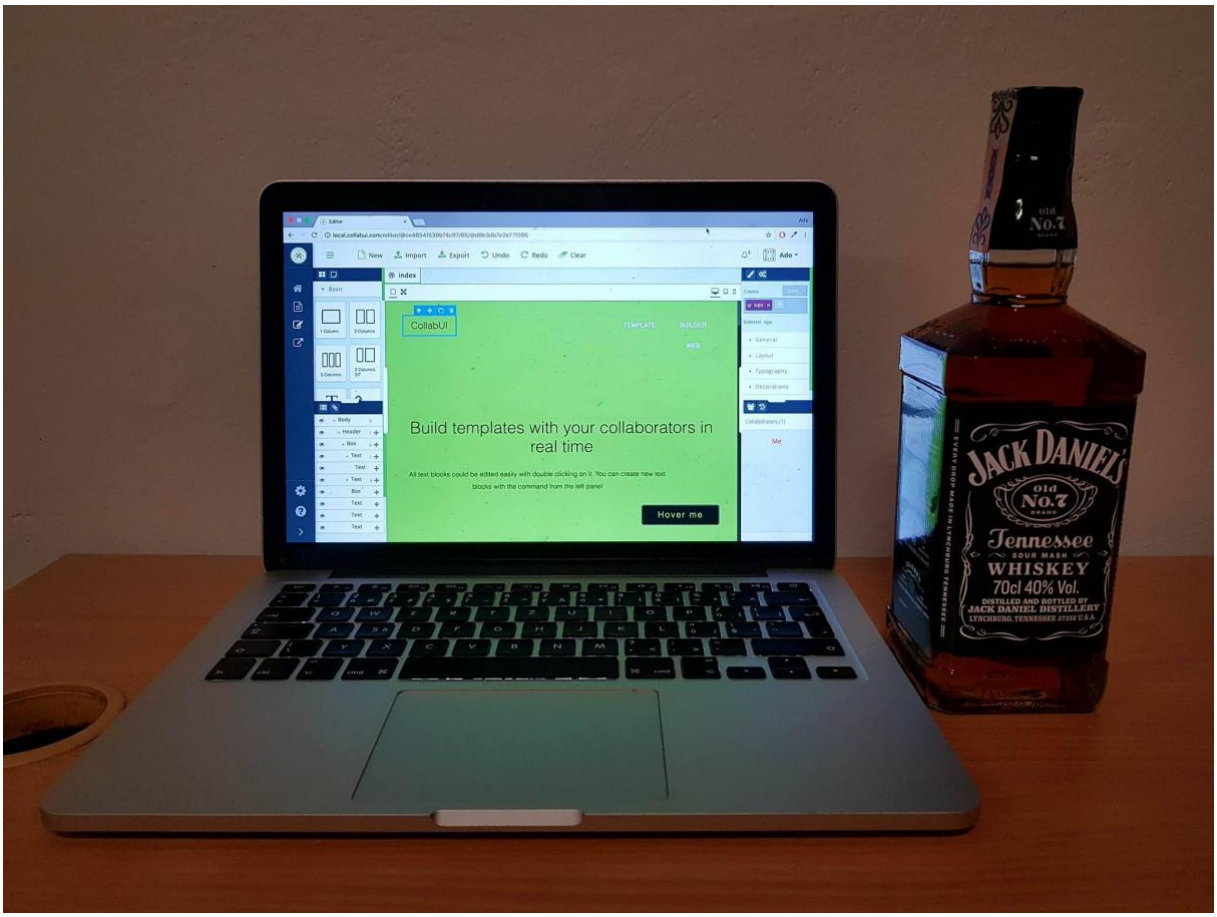


Obrázok 11 - Retrospektíva pre štvrtý šprint

5.1.8 Zhodnotenie šprintu

V rámci tohto šprintu sa nám podarilo uzatvoriť editor bez kolaborantov. Inakšie povedané základy pre editor, ktoré bolo nutné implementovať, sme všetky stihli v rámci tohto šprintu. Napriek tomu sme mali trochu problém pri konfigurácii a implementácii bočných panelov ako aj pri poskytovaní podporných funkcii pri vytváraní alebo navrhovaní stránky. Problémy boli najmä so selektovaním konkrétnych komponentov v rámci stránky. Táto skutočnosť nás zdržala o väčší časový úsek ako sme pôvodne plánovali. Okrem spomínaných funkcii sme si v rámci tohto šprintu predpripravili základ pre prepojenie viacerých používateľov. Prepojenie používateľov a ich pripájanie na editor pomocou soketov sme pôvodne tiež chceli riešiť tento šprint, ale keďže nás jednotlivé už vyššie spomínané skutočnosti zdržali, rozhodli sme sa prehodnotiť pripojiteľnosť používateľov a nakoniec sme zhodnotili, že túto časť necháme na ďalši šprint. V každom prípade sme si v rámci tohto šprintu predpripravili základy pre neskôr dorábané funkcie editora. Čo sa týka burndown chartu opodstatnenie pre dlhú dobu v rámci nedokončených úloh môžeme znova oprieť o skutočnosť zdržania pri základoch editora a problémoch pri implementácii komponentov, ich selektovaní a ukladaní.

Ďalej uvádzame tímový obrázok pre daný šprint reprezentovaný metodikou “Jack_Daniels”.



Obrázok 12 - Logo štvrtého šprintu

5.1.9 Šprint5 (1.12.2017 - 15.12.2017) Jack_Daniels#2

V rámci posledného šprintu za zimný semester sme si zobrali za úlohu dokončiť všetky otvorené úlohy. Tieto úlohy sa týkali najmä editora samotného, ktorého základy sme načali v šprinte „Jack Daniels“. V tomto šprinte sme sa sústredili najmä na dáta kolaborantov. Tieto dáta taktiež zobrazujeme. V rámci používateľov sme implementovali jednotlivé obmedzenia akými sú role „watch“ alebo „banned“. Okrem obmedzení používateľov sme implementovali aj errorové stránky ako aj errorovú stránku pre mobilné zariadenia, ktoré momentálne nepodporujeme. Ďalej sme v rámci manažmentu kolaborantov v editore implementovali ich autorizáciu.

Najväčšiu časť tohto šprintu zabrala implementácia a konfigurácia bočných panelov pre nástroj na návrh stránok, ktorý sme načali v minulom šprinte. V tomto sme ešte doimplementovali kolaboráciu v reálnom čase ako aj rozloženie panelov nástrojov, spolu s ich nastaveniami v rámci editora. Teraz si teda používateľ môže zvoliť rozloženie panelov a nástrojov. Do tohto šprintu sme si preniesli nasledovný user story:

- Editácia projektu s kolaborantmi

Ako ďalší user story sme si zobrali nasledujúce:

- Tagovanie projektov

Celkový počet úloh v šprinte: 30

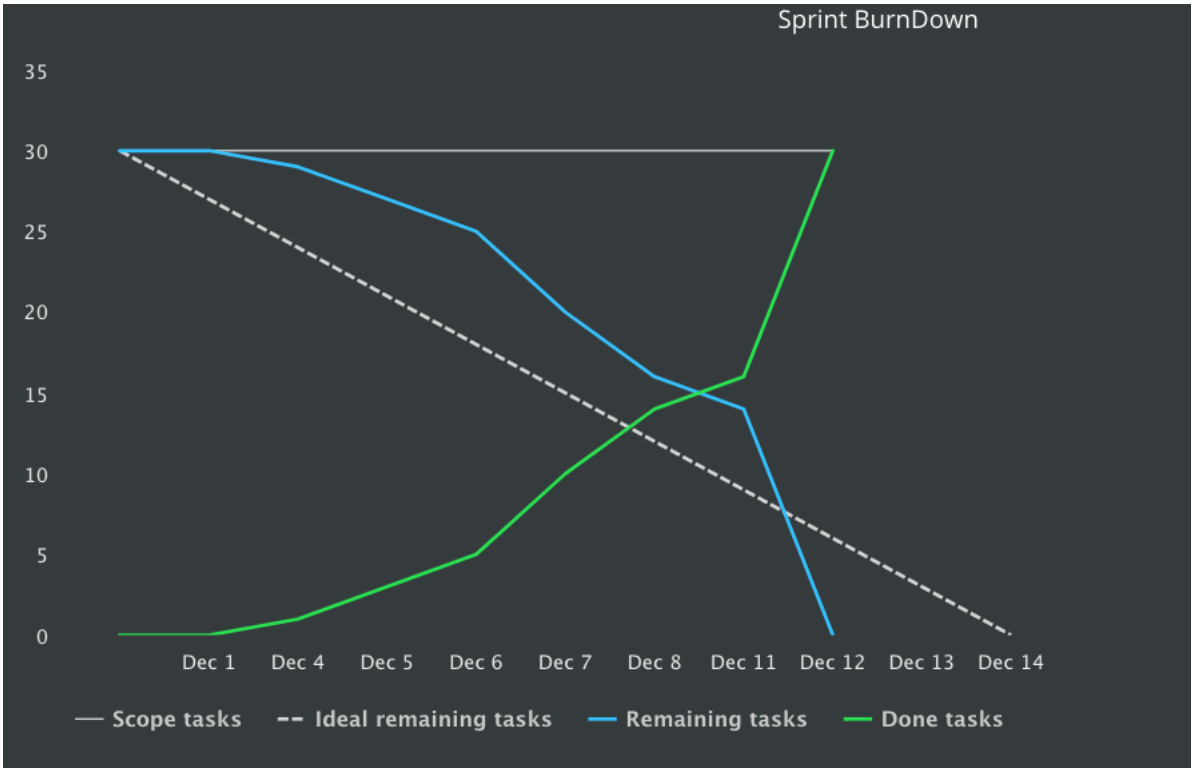
Každá feature obsahuje dokumentáciu a testovanie.

5.1.9.1 Prehľad práce členov tímu

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Editácia projektu s kolaborantmi	19	13	Adrián Nagy
			Michal Melúch
			Lukáš Vrba
			Miloslav Smetana
			Ján Kleň
			Tomáš Mňačko
			Peter Písecký
Tagovanie projektov	11	8	Tomáš Mňačko
			Ján Kleň
			Lukáš Vrba

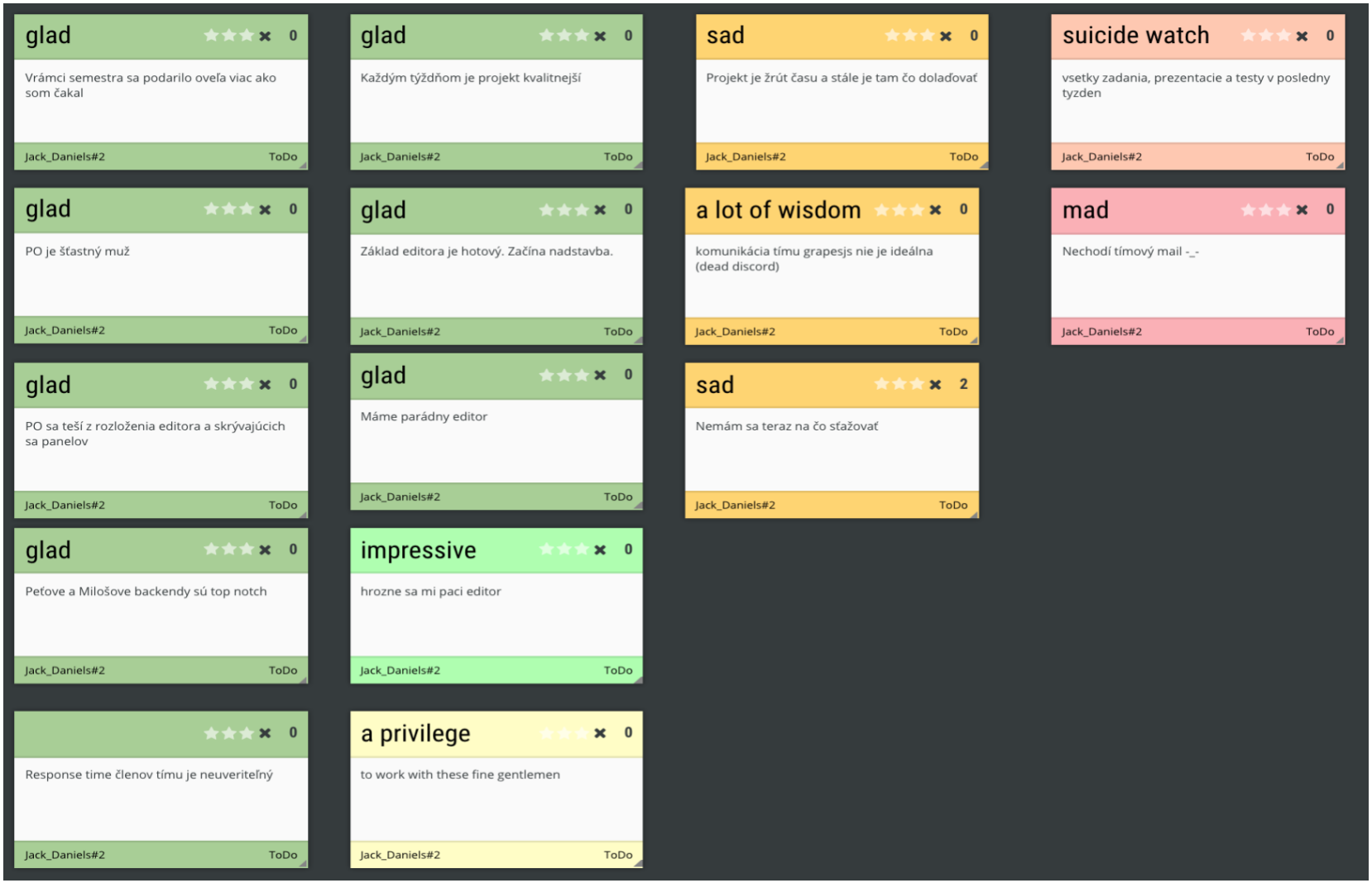
5.1.9.2 Zhrnutie

V rámci zhrnutia uvádzame burndown chart pre uvedený šprint.



Obrázok 13 - Burndown chart pre piaty šprint

5.1.9.3 Retrospektíva



Obrázok 14 - Retrospektíva pre piaty šprint

5.1.10 Zhodnotenie šprintu

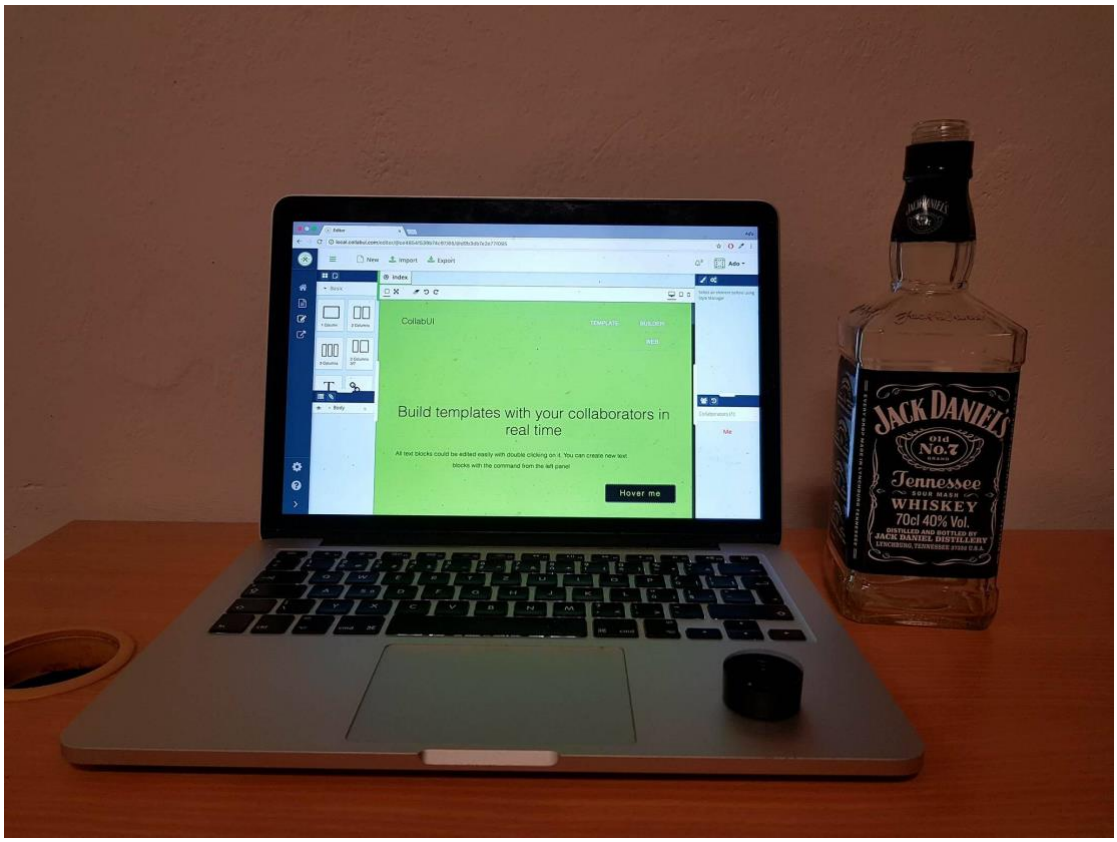
Keďže išlo o posledný šprint v rámci semestra všetky otvorené úlohy boli dokončené.

Ako vidno z burndown chartu zo začiatku sme mali implementačné problémy. Keďže nám veľakrát nastala situácia kedy bolo viac členov tímu zavislých na práci niekoho iného, tento krát sme rozdelili šprint na 2 časti. Jedna časť tvorila samotný editor a druhá časť tvorila tagy pre projekty v rámci collabui. Túto stratégiu sme zvolili práve kvôli tomu aby neboli viacerí členovia tímu priamo závislý na práci niekoho iného a teda aby časť z nás mohla efektívne paralelne vyvíjať istú časť projektu a teda tagovanie projektov. Samozrejme obe časti sme konzultovali všetci spoločne.

Zo začiatku sme teda polovica tímu riešila najmä editor. Práve preto sa nám viacero úloh podarilo dorobiť v prvej polovici šprintu. Druhá polovica zatiaľ efektívne vyvíjala časť pre tagovanie projektov. Keďže sme si uvedomili, že nám stretnutia nevychádzajú do konca semestra boli sme nútení prehodnotiť ďalší týždeň a premiestniť stretnutie z piatka na utorok. Do utorka sme doriešili všetky úlohy či už v rámci editora alebo tagovania projektov a odkonzultovali sme ich s product ownerom.

Tento šprint berieme ako najviac úspešný, keďže sme mali oproti minulým šprintom omnoho viac úloh, ktoré sa nám podarili dokončiť a úspešne uzavrieť naplánované časti projektu pre zimný semester.

Ďalej uvádzame tímový obrázok pre daný šprint reprezentovaný metodikou “Jack_Daniels#2”.



Obrázok 15 - Logo piateho šprintu

6 Globálna retrospektíva

6.1 Zimný semester

Na konci semestra sme si v rámci tímu spravili globálnu retrospektívu, na ktorej sme zhodnotili všetky šprinty a celkovo sme si povedali v akých princípoch budeme pokračovať a v akých nie.

V rámci zistených problémov v jednotlivých šprintoch sme identifikovali pre každý špint nasledujúce riešenia:

1.šprint

- Zlepšiť komunikáciu v tíme
- Striktnejšia definícia jednotlivých úloh, potreba wireframov
- Napísať metodiku pre testovanie
- Lepšie prerozdeliť úlohy v rámci tímu
- Sofistikovanejšie branchovanie projektu

2.šprint

- Motivovať tím
- Nezaoberať sa s drobnosťami, napr. preštýľovávanie elementov dookola
- Pokúsiť sa lepšie odhadnúť jednotlivé feature
- Lepšie prerozdeliť úlohy v rámci tímu
- Naučiť sa pracovať s nástrojom ScrumDesk

3.šprint

- Zrefaktorovať BE
- Budúci šprint začať implementáciu už na začiatku

4.šprint

- Zrýchliť proces implementácie
- Nájsť si čas pre refaktoring, code review a testovanie

5.šprint

- Opraviť tímový mail ☹

7 Záver

Na záver dokumentu riadenia nemôžeme nechať opomenúť fakt, že sme každý šprint pracovali s plným nasadením, o čom svedčí skutočnosť, že sme dokázali na koniec každého šprintu prispieť do branche master novou otestovanou a akceptovanou funkcionalitou. Každý z nás nadobudol nové skúsenosti, ktoré s určitosťou využijeme v praxi. Mnohí z nás sa naučili pracovať s novými technológiami a taktiež vylepšili svoje komunikačné schopnosti.

Téma tímového projektu nás natoľko nadchla, že sme sa prihlásili na TP Cup. Kvôli tejto skutočnosti sme odhodlaní pracovať aj počas skúškového obdobia. Ako sa hovorí: Bez koláča nie sú práce.

Ná skúškové obdobie máme naplánované:

- Vytváranie viacerých prototypov v jednom projekte a ich prepojenie
- Vytvorenie prototypu na zahodenie – hlasový chat
- Verziovanie projektov
- Chat

Dúfame, že sa nám podarí uskutočniť vyššie stanovené cieľe a ďalší semester začneme s mierným náskokom.

1 Príloha A - Metodiky

V rámci správneho fungovania tímu je dôležité mať definované jednotlivé zvyklosti, ktoré sa v tíme snažíme dodržiavať. Urýchluje to čas pri tvorbe stereotypných akcii ako aj zmenšuje neprehľad členov tímu o vykonávaných skutočnostiach. V rámci nášho tímu úvádzame jednotlivé metodiky, ktoré sme si pre tento projekt definovali.

- Metodika spracovania úloh
- Metodika písania kvalitného kódu
- Metodika testovania
- Metodika verziovania
- Metodika nasadzovania

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie

[CollabUI]

Metodika spracovania úloh

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1. Úvod

Priebeh spracovania úloh od návrhu až po realizáciu prebieha v našom tíme v niekoľkých krokoch. Je veľmi dôležité zdokumentovať ako tento proces prebieha, pretože má veľký vplyv na celkový priebeh projektu.

Používame podporný nástroj ScrumDesk, ktorý je uspôsobený pre agile development pričom ostatné záležitosti sú prerokované na stretnutí. Tento dokument popisuje základný spôsob práce s nástrojom Scrumdesk.

Touto metodikou sa riadia všetci členovia tímu č. 24 CollabUI v rámci predmetu tímový projekt v akademickom roku 2017/2018.

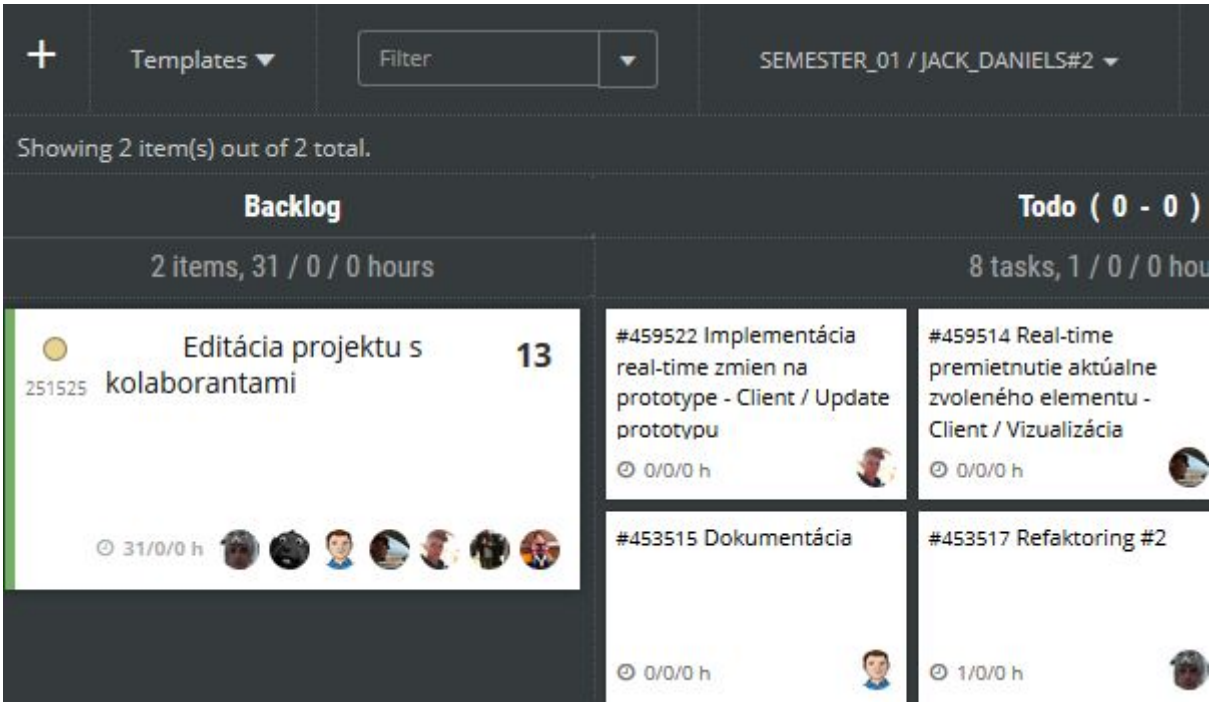
2. Pojmy a skratky

Sprint - iteračné obdobie projektu, v našom prípade 2. týždne
Epic - konkrétna oblasť projektu, dekomponovaná na menšie oblasti
User Story/Feature - US - jedna funkcionalita projektu, ktorá sa skladá z niekoľkých taskov
Task - úloha v rámci jedneho story
Product owner - PO - komunikuje so zákazníkom a zbiera od neho požiadavky, slúži ako mediátor pre komunikáciu s tímom a zákazníkom
Definition of done - DOD - požiadavky, ktoré musia byť splnené pri odovzdaní časti projektu či už ide o task alebo story…
Effort - celkový čas strávený nad jednou US

3. Pravidlá a postupy

3.1 Konvencie názvov

Všetky názvy pre jednotlivé Epicy, featury a tasky sú napísané buď v angličtine alebo slovenčine. Preferovaný jazyk je slovenčina. Sú stručné a popisujú podstatu svojho prvku.



Obr. 1 - Názvy US a taskov

3.1 Poviné atribúty

- story pointy
- opis + wireframe ak je potrebný
- akceptačné kritéria

Všetky prvky majú podrobný opis v slovenčine, kde sú napísané aj validačné kritéria pre DoD. Každá feature má nastavený effort, ktorý je výsledkom planning pokeru pri plánovaní ďalšieho šprintu. Planning poker je skupinové ohodnotenie jednotlivých features hodnotou, ktorá reprezentuje počet hodín strávených na projekte a náročnosť. V našom projekte je jednotkou pre effort

pracovný deň t.j. 8 hodín. US ohodnotený 13 bodmi teda bude vyžadovať cca 104 hodín.

Obr. 2 - Opis US

3.2 Sledovanie času

Každú člen tímu pracujúci na tasku si značí koľko času na ňom strávil.

Obr. 3 - čas

3.3 Priebeh šprintu a definovanie úloh

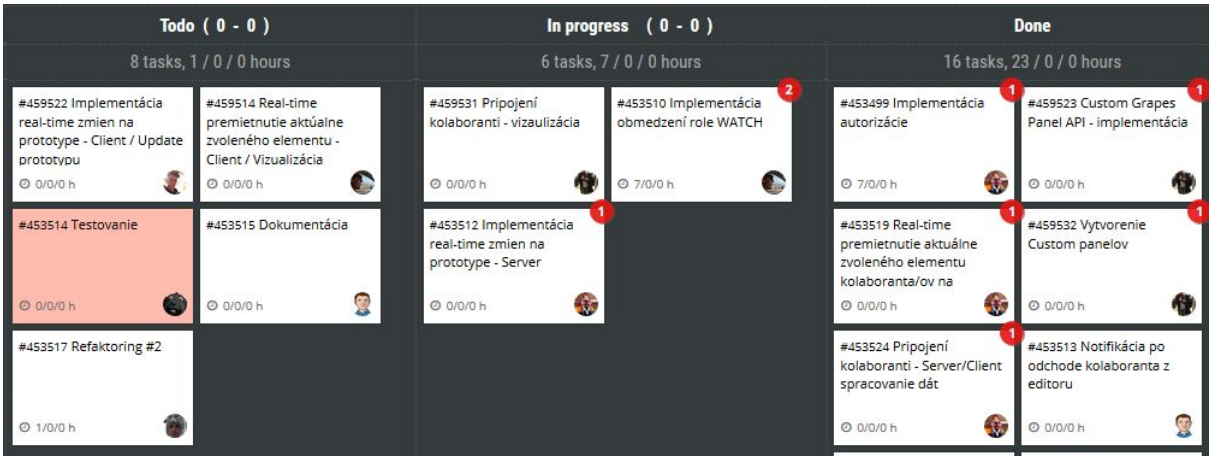
Pred začatím nového šprintu prebehne tímové strenutie, kde sa diskutuje a značí na tabuľu čo sa bude ďalej robiť a s tým spojené vytvorenie features. Features s

taskami sú neskôr v ten deň nahodené a dôkladne popísané aj s validačnými kritériami pridané do nástroja ScrumDesk vedúcim tímu po diskusii.

Ďalej prebieha planning poker pričom sa označí náročnosť jednotlivých features a diskutuje sa prečo si jednotlivý členovia zvolili takú a takú náročnosť. Po zhodnotení všetkých kritérii sa prijme kompromis, ktorý vyhovuje všetkým členom tímu.

Každý člen tímu si pridelí tasky, na ktorých bude potenciálne robiť. Ak nie je task v in progress a teda na ňom nerobí, môže mu byť odobratý iným členom tímu. Životný cyklus tasku prebieha nasledovne:

- ToDo - task nie je rozpracovaný, hociktorý člen tímu si ho môže zobrať
- In progress - člen tímu práve pracuje na tasku
- Done - task spĺňa validačné kritéria a je hotový



Obr. 4 - Životný cyklus taskov

Pri definovaní taskov sa prihliada aj na testovanie a code review, vždy ked sa jedná o implementačné časti. Pri uzatvorení šprintu musia byť všetky tasky v poli Done a features schválené PO. Inak sú prenesené do ďalšieho šprintu.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie
[CollabUI]
Metodika písania kvalitného kódu

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1. Úvod

Metodika obsiahnutá v tomto dokumente predpisuje pravidlá pre písanie kódu v jazykoch PHP a JavaScript. Cieľom tejto metodiky je zjednotiť štýly písania kódu medzi jednotlivými členmi tímu tak, aby vyprodukovaný kód bol prehľadný a nasledoval konvencie písania kódu v jazykoch PHP a JavaScript, ako aj konvencie technológií CakePHP a Node.js.

2. Dôležité odkazy

PSR-2: Coding Style Guide - http://www.php-fig.org/psr/psr-2/
CakePHP: Coding Standards - https://book.cakephp.org/3.0/en/contributing/cakephp-coding-conventions.html
CakePHP Code Sniffer - https://github.com/cakephp/cakephp-codesniffer
JSHint - http://jshint.com/

3. Pravidlá písania kódu v jazyku PHP

Kód napísaný v jazyku PHP musí nasledovať štandard PSR-2, ako aj dodatočné pravidlá, ktoré sú uvedené v dokumentácii CakePHP. Nižšie uvedený zoznam poskytuje prehľad najdôležitejších pravidiel pre písanie kódu v jazyku PHP.

Odsadenie
Pre odsadenie PHP kódu sa používajú 4 medzery.

Maximálna dĺžka riadku
Maximálna dĺžka riadku je 100-120 znakov.

Porovnávanie
Porovnávanie hodnôt by malo byť vždy čo najstriktnejšie. Hodnota, voči ktorej sa porovnáva by mala byť vždy umiestnená na pravej strane.

Pomenovanie metód
Názov metódy musí byť vždy napísaný vo formáte camelBack.

Volanie funkcií
Volanie funkcií by nemalo obsahovať medzeru medzi názvom funkcie a zátvorkami. Medzi každým argumentom volanej funkcie by sa mala nachádzať práve jedna medzera.

Definovanie funkcií
Parametre s predpísanou hodnotou musia byť umiestnené na konci zoznamu parametrov definovanej funkcie. Každá funkcia by *mala* niečo vracať.

Pomenovanie tried
Názvy tried musia byť napísané vo formáte CamelCase.

Pomenovanie premenných
Všetky názvy premenných musia byť vo formáte camelBack.

Definovanie konštánt
Názvy konštánt by vždy mali byť definované pomocou veľkých písmen.

Komentovanie a nápovedy pri písaní
Každá funkcia by mala byť okomentovaná tak, aby jej správanie bolo jednoducho pochopiteľné. Verejné funkcie musia obsahovať prvky umožňujúce poskytovanie nápovede pri písaní kódu v IDE.

Zreťazenie metód
Zreťazené volania metód by mali obsahovať každé volanie na samostatnom riadku odsadené 4 medzerami.

PHP značky
Vždy musí byť využití dlhší formát PHP značiek (`<?php … ?>`).

Pomenovanie súborov
Názvy súborov neobsahujúcich triedy musia byť napísané malým písmom a predelené podtržníkom: `dlhy_nazov_suboru.php`

3.1. Kontrola napísaného kódu

Framework CakePHP poskytuje nástroj CakePHP Code Sniffer pre kontrolu správneho nasledovania vyššie uvedených štandardov.

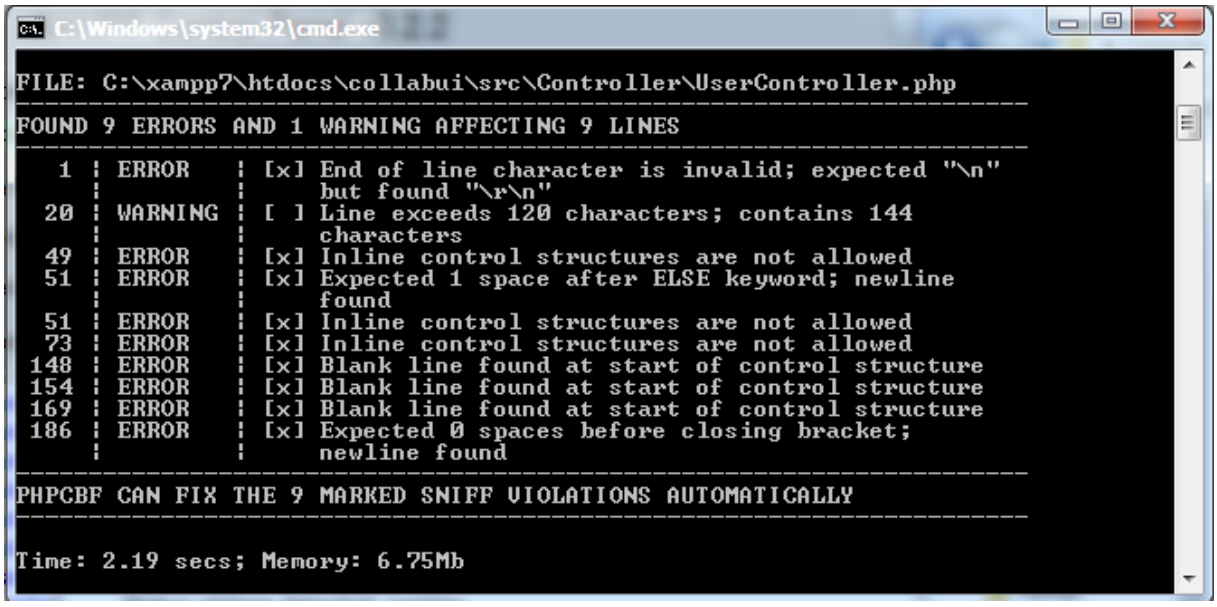
Inštalácia:

```
composer require --dev "cakephp/cakephp-codesniffer"
```

Použitie:

```
vendor/bin/phpcs --standard=CakePHP /path/to/code
```

Príklad výstupu:



Obrázok 1: Príklad výstupu nástroja CakePHP Code Sniffer

4. Pravidlá písania kódu v jazyku JavaScript

Technológia Node.js nemá predpísaný štýl písania kódu. Nižšie uvedený zoznam predpisuje nami stanovené pravidlá pre písanie kódu v jazyku JavaScript.

ECMAScript
JavaScript kód napísaný pre Node.js časť aplikácie môže využívať nové konštrukty špecifikácie ES6.

Odsadenie
Pre odsadenie JavaScript kódu sa používajú 2 medzery.

Maximálna dĺžka riadku
Maximálna dĺžka riadku je 100-120 znakov.

Porovnávanie
Porovnávanie hodnôt by malo byť vždy čo najstriktnejšie. Hodnota, voči ktorej sa porovnáva by mala byť vždy umiestnená na pravej strane.

Pomenovanie funkcií
Názov funkcie musí byť napísaný vo formáte camelBack. Výnimka: Funkcie slúžiace ako konštruktory (prototyp) musia byť napísané v CamelCase.

Volanie funkcií
Volanie funkcií by nemalo obsahovať medzeru medzi názvom funkcie a zátvorkami. Medzi každým argumentom volanej funkcie by sa mala nachádzať práve jedna medzera.

Definovanie funkcií
Parametre s predpísanou hodnotou musia byť umiestnené na konci zoznamu parametrov definovanej funkcie. Každá funkcia by *mala* niečo vracať.

Pomenovanie premenných a atribútov objektov
Všetky názvy premenných a atribútov objektov musia byť vo formáte camelBack.

Definovanie konštánt
Názvy konštánt by vždy mali byť definované pomocou veľkých písmen. Výnimka: Konštanty obsahujúce Node.js moduly volané pomocou funkcie *require.*

Komentovanie a nápovedy pri písaní
Každá funkcia by mala byť okomentovaná tak, aby jej správanie bolo jednoducho pochopiteľné. Verejné funkcie musia obsahovať prvky umožňujúce poskytovanie nápovede pri písaní kódu v IDE.

Zreťazenie metód
Zreťazené volania metód by sa mali (ale nemusia) nachádzať na samostatnom riadku odsadené 2 medzerami.

4.1. Kontrola napísaného kódu

Pre kontrolu správnosti kódu napísaného v jazyku JavaScript je možné využiť správne nakonfigurovaný nástroj JSHint.

Inštalácia:

```
npm install -g jshint
```

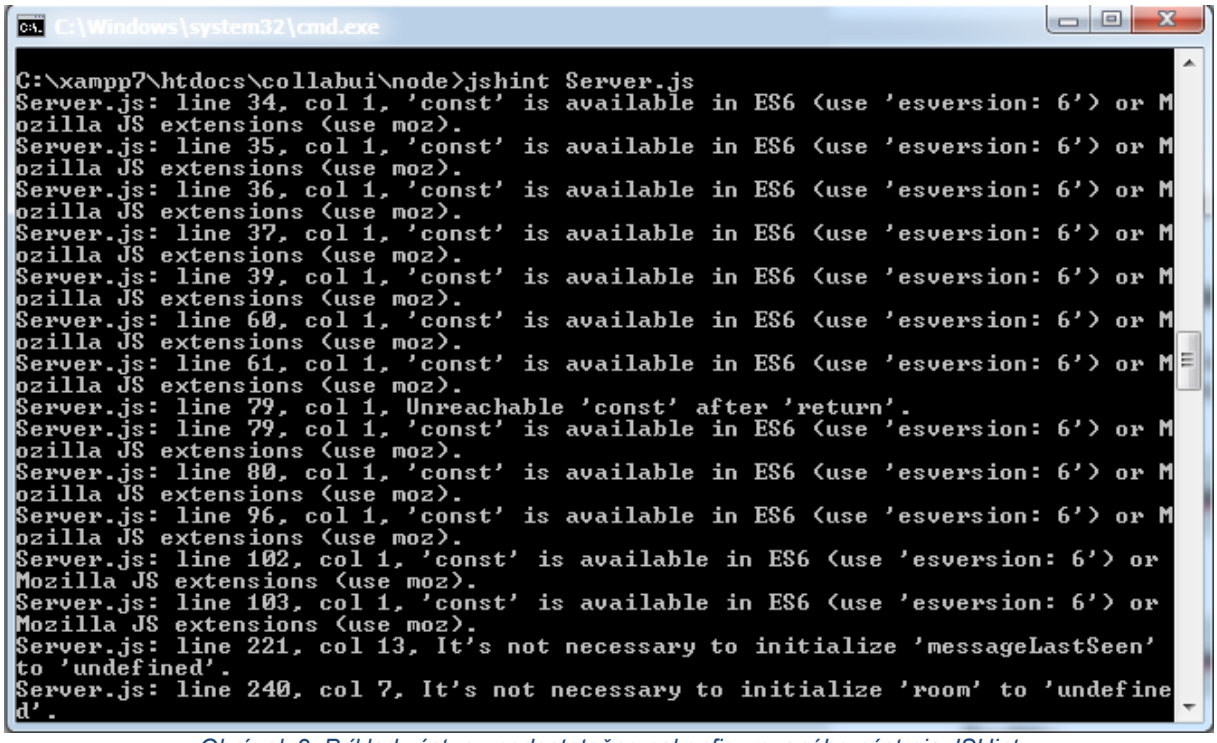
alebo:

```
npm install --save-dev jshint
```

Použitie:

```
jshint /path/to/code
```

Príklad výstupu:



Obrázok 2: Príklad výstupu nedostatočne nakonfigurovaného nástroja JSHint

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie

[CollabUI]

Metodika testovania

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1 Úvod

Táto metodika je určená pre členov tímu číslo 24 v akademickom roku 2016/2017 a obsahuje postupy a pravidlá pre testovanie webovej aplikácie na kolaboratívne prototypovanie. Metodika pozostáva z opisu oboch jazykov, ktoré sú použité v tomto projekte spolu s prikázanými konvenciami a návodmi ako spúšťať a ukladať testy.

2 Skratky a značky

Mock	simulovaný objekt, ktorý napodobňuje správanie skutočného objektu
Fixture	mockovací objekt používaný knižnicou PhpUnit
JS	JavaScript
Commit	zmeny, ktoré vykonal člen tímu v projekte a označil ich komentárom

3 Postupy

Táto kapitola predpisuje spôsob, ktorým sa pri testovaní riadia všetci členovia tímu, konkrétne upresňuje spôsob ukladania testov, ich spúšťanie a pravidlá písania testov. Testy sú vykonávané v dvoch jazykoch a to: Javascript a Php. Každý jazyk má svoje knižnice a pravidlá, rovnako ako aj miesto uloženia. Tieto informácie sú uvedené v podkapitolách nižšie.

3.1 Jazyk PHP

PhpUnit je knižnica, ktorá sa musí používať v prípade ak je potrebné vytvoriť testy pre jazyk PHP. Integračné testy je nutné vytvárať práve pomocou tejto knižnice, rovnako ako testy týkajúce sa databázy.

3.1.1 Konvencie

Na testovanie databázy je vždy potrebné vytvoriť mocky jednotlivých tabuliek pomocou objektov nazývaných Fixture v knižnici PhpUnit. Súbory, ktoré sú použité ako Fixture musia byť nazvané vo formáte NázovtabulkyFixture.php, napríklad UsersFixture.php. Na obrázku 1 je príklad použitia Fixture v jazyku PHP.
Každý test v PhpUnit musí byť definovaný ako funkcia, ktorej názov začína kľúčovým slovom test a potom opisom toho, čo testuje, napríklad *testLogInValid().* Na obrázku 2 je uvedený príklad testu *testLogInValid*.

```php
class UsersFixture extends TestFixture
{
    // Import table definition: specify its name and connection to be used (see app.php -> connections)
    public $import = ['table' => 'users', 'records' => false, 'connection' => 'default'];

    public function init()
    {
        $this->records = [
            [
                'first_name' => 'Test',
                'last_name' => 'Test-Surname',
                'email' => 'testuser@email.com',
                'password' => (new DefaultPasswordHasher)->hash(
                    '12345'
                ),
                'token' => 'password',
                'settings' => null,
                'status' => 2
            ]
        ];
        parent::init();
    }

}
```

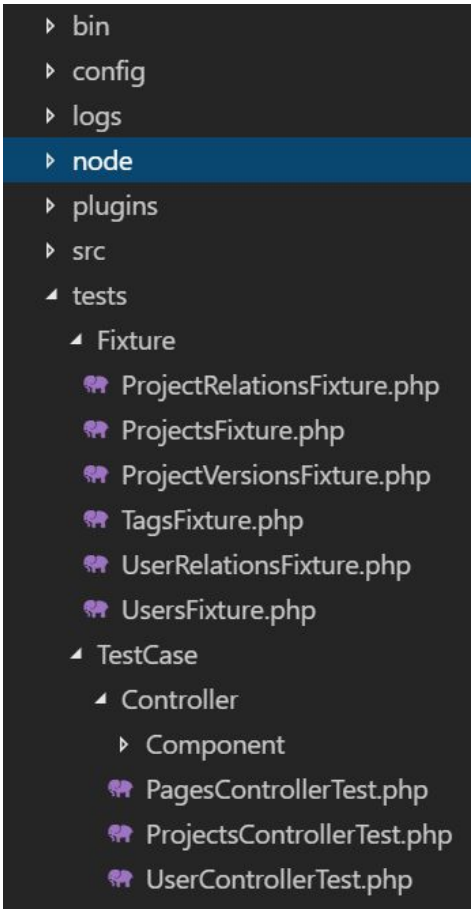
Obr. 1 - príklad mockovania tabuľky Users

```php
public function testLogInValid()
{
     // Load a user from the fixture dummy data
    $user = $this->Users->find('all')->first();
        var_dump($user);
    $data = [
        'email' => 'testuser@email.com',
        'password' => '12345'
    ];
    // Perform the password reset request
    $this->post('/log-in', $data);

    // Look for a 2xx/3xx response code
    $this->assertResponseSuccess();

    // Check the redirect to homepage after a successful request
    $this->assertRedirect('/dashboard');

    $this->get('/log-out');

    $this->assertResponseSuccess();

    $this->assertRedirect('/');
}
```

Obr. 2 - test správneho prihlásenia a presmerovania

3.1.2 Úložisko testov

Testy určené pre jazyk PHP sa musia ukladať do zložky tests\TestCase. A mocky, ktoré reprezentujú tabuľky musia byť umiestnené v zložke tests\Fixture. Na obrázku 3 je ukázaný príklad umiestnenia testov a mockov v jazyku PHP.



Obr. 3 - miesto uloženia PHP testov

3.1.3 Spúšťanie testov

Testy sa musia spúšťať pred každým commitom do ktorejkoľvek vetvy. Každý člen tímu je zodpovedný za to, aby testy po jeho práci boli úspešné.
PHP testy sa spúšťajú pomocou zadania príkazu *phpunit* do konzoly v root adresári, v prípade, že je potrebné vykonať konkrétny test, ako prvý argument sa zadá cesta ku testu, napríklad *phpunit tests\TestCase\UserControllerTest.php*

3.2 Jazyk JavaScript

Pre jazyk JavaScript boli určené knižnice Mocha na vytváranie a overovanie JS testov a knižnica Sinon, ktorá je použitá na overovanie správnych volaní funkcií a ich argumentov.

3.2.1 Konvencie

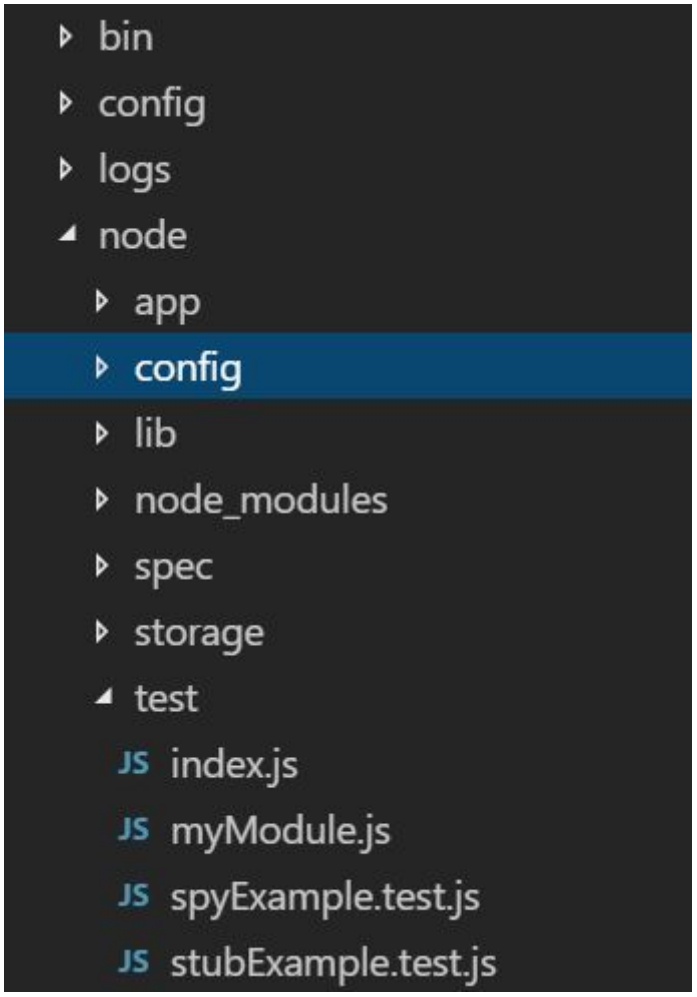
Pri vytváraní testov v jazyku JS sa rozďeluje test na 3 časti a to: given, when, then. V časti given sa inicializujú dáta a premenné, v časti when sa vykonáva logika, ktorá má byť otestovaná, konkrétne sa volajú sa funkcie a v poslednej časti then sa overuje korektné správanie funkcií.
Testy sa vždy nazývajú názovsúboru.test.js v opačnom prípade nebude vedieť skript nájst a vykonať testy.
Príklad testu a konvencií je uvedený na obrázku 4.

```
const sinon = require('sinon');
const myModule = require('./myModule');
const assert = require('assert');

// spy allows us to get information like number of calls,
// what arguments they received and so on
describe('spy example', function () {

it('calls the original function once', function() {
    // given
    var callback = sinon.spy()
    var proxy = myModule.once(callback);

    //when
    proxy();
    proxy();

    //then
    assert(callback.calledOnce);
});
```

Obr. 4 - príklad konvencií JS testov

3.2.2 Úložisko testov JavaScript

Testy pre JS musia byť uložené v zložke node\test a nazvané podľa konvencie uvedenej v časti 3.2.1. Testy, ktoré sú umiestnené v inej zložke nebudú vykonané. Príklad umiestnenia testov je na obrázku 5.



Obr. 5 - príklad umiestnenia JS testov

3.2.3 Spúštanie testov

Testy sa musia spúštať pred každým commitom do ktorejkoľvek vetvy. Každý člen tímu je zodpovedný za to, aby testy po jeho práci boli úspešné.
JS testy sa spúštajú zo zložky node zadaním príkazu *npm run test* do konzoly

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie

[CollabUI]

Metodika verziovania

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1. Úvod

Cieľom tejto metodiky je definovať základné postupy odovzdávania zdrojového kódu a udržiavania verzií v projekte Collab-UI. Metodika opisuje prácu s nástrojom na správu verzií Git spolu s použitou grafickou nadstavbou Sourcetree. Repozitár celej vyvíjanej aplikácie je uložený na portáli Bitbucket.

Touto metodikou sa riadia všetci členovia tímu č. 24 Collab-UI v rámci predmetu tímový projekt v akademickom roku 2017/2018.

2. Pojmy a skratky

- **Sprint** - iteračné obdobie projektu, v našom prípade 2. týždne
- **Epic** - konkrétna oblasť projektu, dekomponovaná na menšie oblasti
- **User Story/Backlog** - US - jedna funkcionalita projektu, ktorá sa skladá z niekoľkých taskov
- **Task** - úloha v rámci jedneho story
- **Product owne**r - PO - komunikuje so zákazníkom a zbiera od neho požiadavky, slúži ako mediátor pre komunikáciu s tímom a zákazníkom
- **Git** - distribuovaný systém riadenia revízií
- **Branch** - vetva - vývojová vetva projektu, v ktorej sa realizuje ucelená časť funkcionality.
- **Master branch** - hlavná vetva projektu, v ktorej je nasadená implementácia, ktorá úspešne prešla testovaním a zároveň ju akceptoval aj product owner.
- **Commit** - príspevok (najčastejšie implementačný) do vybranej vetvy projektu

3. Dôležité odkazy

- Repozitár projektu: https://bitbucket.org/agileuiteam/uiagile_repo
    - URL pre klonovanie: https://BB-NICKNAME@bitbucket.org/agileuiteam/uiagile_repo.git
- Systém riadenia revízií Git: https://git-scm.com/
- Nástroj pre pokročilú správu revízií: https://www.sourcetreeapp.com/

4. Pravidlá a postupy

Pri práci s verziami projektu je nutné riadiť sa nasledovnými pravidlami:

- Na správu verzií sa používa výhradne systém *Git* v spojení s hostingovou službou *Bitbucket*
- Na všetky úkony sa používa GUI rozhranie nástroja *Sourcetree*. Predchádza sa tak chybám, ktoré vznikajú pri používaní systému *Git* pomocou príkazového riadku. Využitie príkazového riadku je povolené len v špeciálnych prípadoch, ktoré nemôžu byť vyriešené pomocou nástroja *Sourcetree*.
- Závažné konflikty medzi jednotlivými príspevkami (t. j. commitmi) sa vždy riešia manuálne po konzultácii s autorom konfliktného kódu.
- Hlavná (angl. master) vetva projektu je v správe scrum mastera, ktorý do nej zlučuje zmeny až po ich úspešnom otestovaní akceptácií product ownerom. **Do master vetvy sa v priebehu šprintu necommituje žiadny kód.**

4.1 Základy práce s nástrojmi na správu verzií

Aby bolo možné pracovať s vzdialeným repozitárom projektu Collab-UI, je nutné mať nainštalovaný nasledujúci softvér:

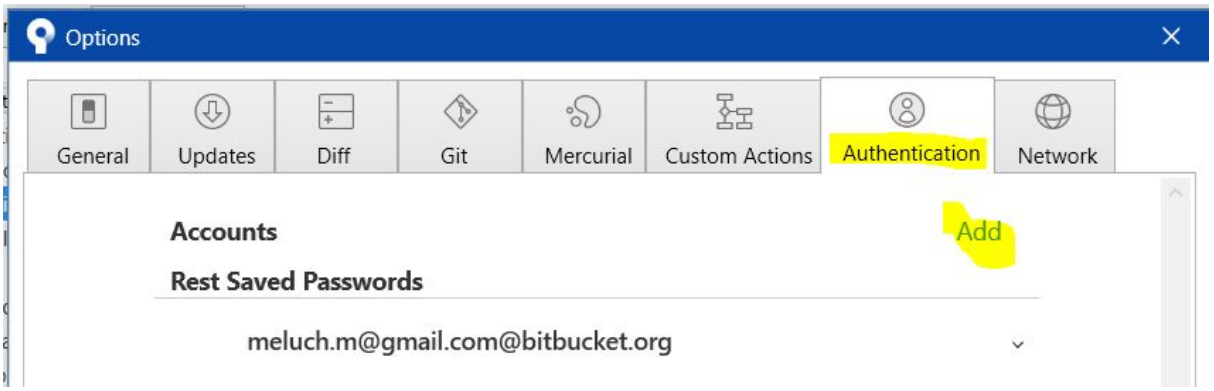
1. Systém riadenia revízií **Git** (stiahnuť TU)
2. Grafický nástroj pre správu verzií **Sourcetree** od spoločnosti *Atlassian* (stiahnuť TU)

Aby bolo možné pracovať so vzdialeným repozitárom uloženým na serveri Bitbucket, **je nutné mať vytvorené Atlassian konto**. Konto je možné si vytvoriť TU.

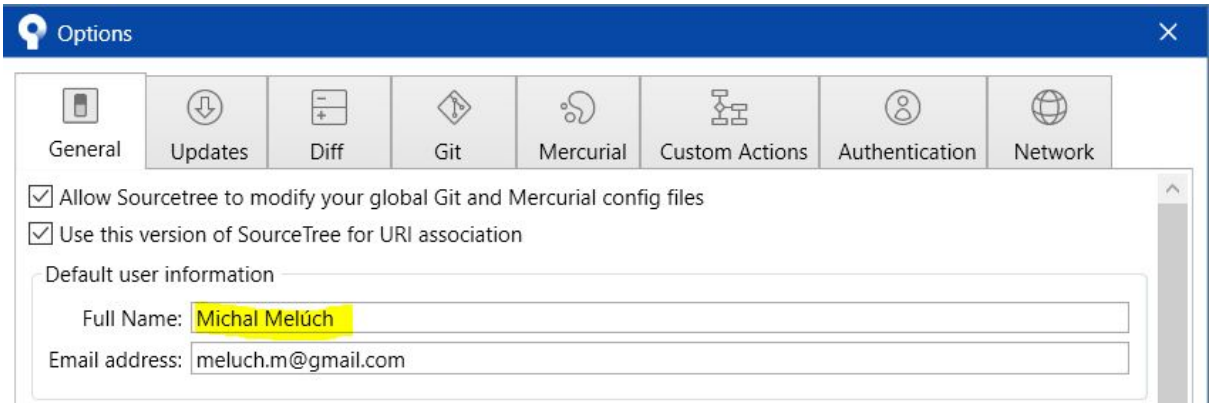
Po nainštalovaní uvedeného softvéru je nutné prepojiť aplikáciu *Sourcetree* so vzdialeným repozitárom projektu a Atlassian kontom vývojára.

Postup prihlásenia sa do svojho Atlassian konta v aplikácii Sourcetree je nasledovný:

1. Z hornej lišty zvolíme možnosť *Tools* a následne *Options.*
2. V pop-up okne, ktoré sa zobrazilo vyberieme predposlednú záložku s názvom *Authentication* a vyberieme možnosť pridať nové konto (*Add*):



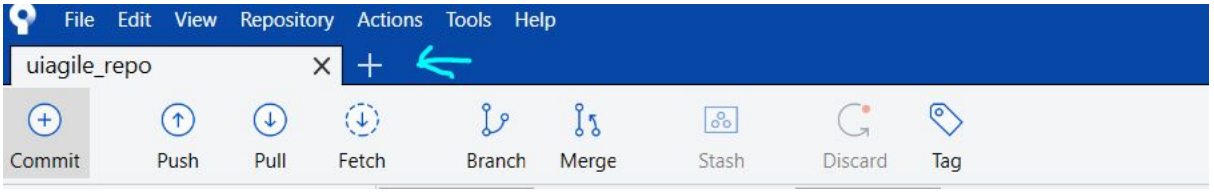
3. Zadáme svoje prístupové údaje, pomocou ktorých sa prihlasujeme aj na doménu https://bitbucket.org/.
4. V prvej záložke pup-up okna vyplníme svoje celé meno a priezvisko:



5. Aplikácia *Sourcetree* je teraz korektne prepojená s naším *Bitbucket* kontom.

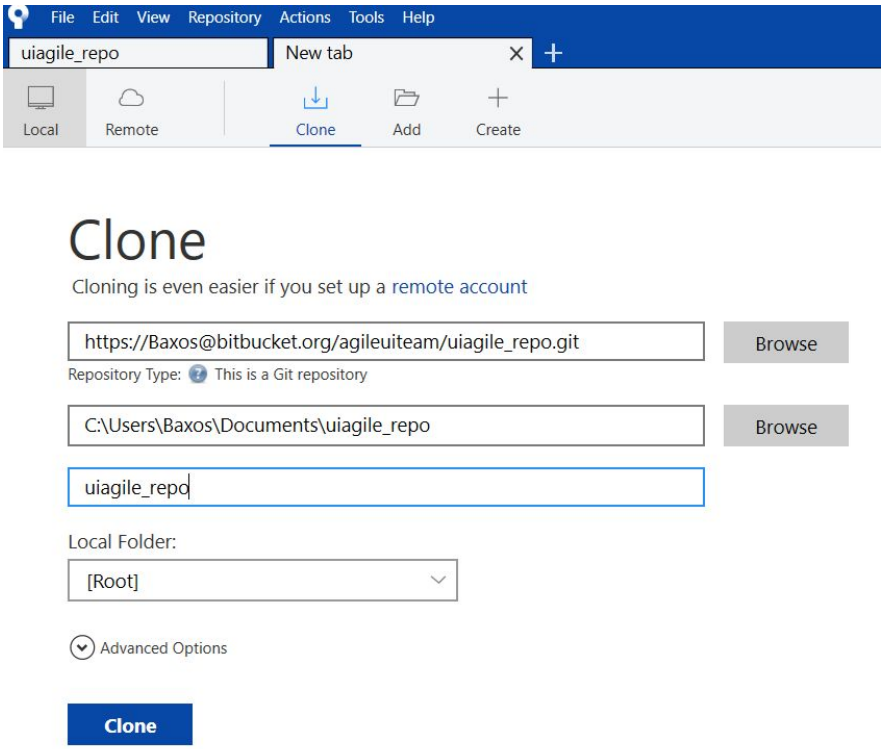
Postup pripojenia sa k vzdialenému repozitáru aplikácie je nasledovný:

1. Ak nás aplikácia sama nevyzve na klonovanie existujúceho repozitára, túto možnosť spustíme pomocou pridania novej záložky pod hornou lištou aplikácie:



2. Z horného menu pod názvom záložky vyberieme možnosť *Clone* a vložíme HTTPS URL Bitbucket repozitára, ktorá je uvedená v prehľade (*Overview*)

na stránke Bitbucket (odkaz TU).

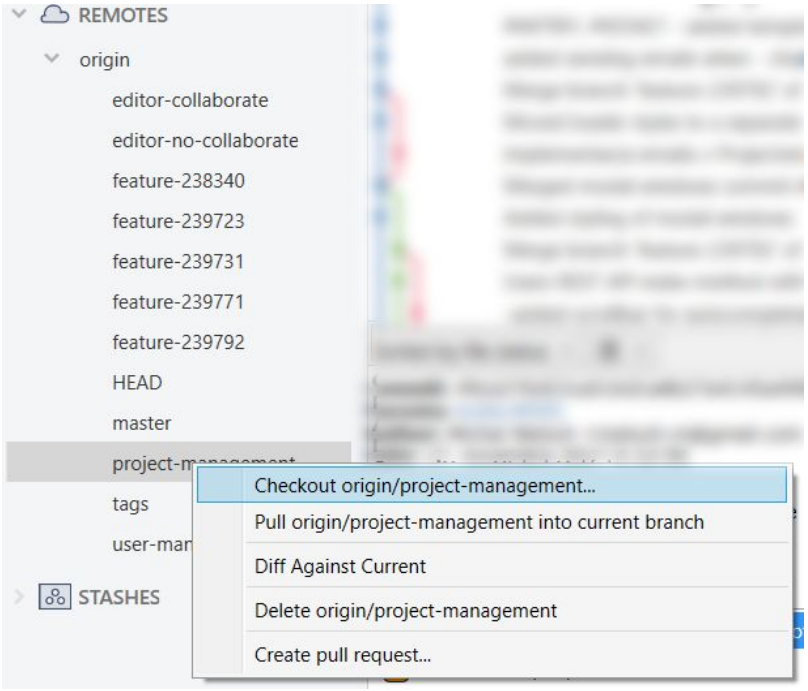


3. Proces ukončíme tlačidlom *Clone* a počkáme kým sa repozitár naklonuje na nami zvolené lokálne úložisko.

4.2 Práca s vetvami projektu

Vetvy projektu sú vytvárané pomocou zoskupovania aktuálne riešených úloh (t. j. podľa scrum stories). Názvy vetiev musia byť čitateľné a zodpovedať charakteru riešených úloh - napr. *user-management* alebo *project-management*.

Rozlišujeme lokálne (tzv. *local*) a vzdialené (tzv. *remote*) vetvy. Vytvárať nové lokálne vetvy nie je pri bežnom vývoji potrebné - všetky vetvy vytvárame ako remotes (t. j. vzdialené) a ostatní členovia tímu si ich následne stiahnu do svojich počítačov pomocou možnosti *checkout* nachádzajúcej sa v bočnom menu pod záložkou *Remotes*:



Po checkoutnutí vzdialenej vetvy sa vytvorí jej lokálna kópia, ktorú môžeme vidieť v bočnom menu pod záložkou *Branches*. Prepínanie medzi takto vytvorenými lokálnymi verziami funguje taktiež pomocou príkazu *checkout* avšak vykonávame ho pod záložkou *Branches* (t. j. medzi lokálnymi vetvami)

Postup vytvárania novej vzdialenej vetvy (napr. pri začiatku nového šprintu) je podrobne opísaný v dokumentácii nástroja *Sourcetree* (odkaz TU).

4.3 Zásady prispievania do vetiev

Bežné prispievanie do vetiev projektu je realizované pomocou nasledovných *Git* príkazov dostupných v prostredí aplikácie *Sourcetree*:

- **Fetch** (automaticky po zapnutí aplikácie, manuálne v prípade potreby)
- **Pull** (pred začatím práce na projekte)
- **Commit** (po skončení menšej časti práce)
- **Push** (po skončení ucelenej časti práce)

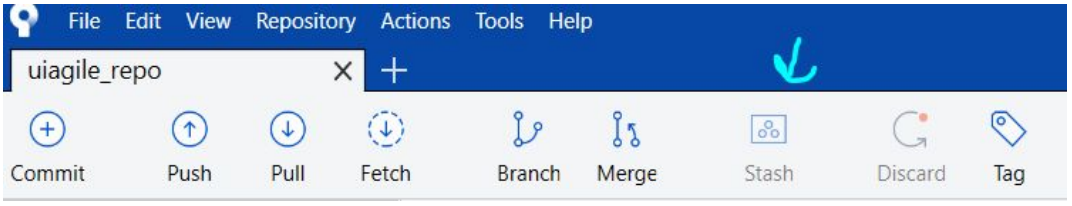
V prípade konfliktov medzi súbormi po príkaze *Push* je nutné tieto konflikty vyriešiť tak, aby zostali zachované zmeny obidvoch autorov. Následne je možné tieto súbory opakovane commitnúť a budú systémom *Git* akceptované.

Správy opisujúce jednotlivé commity musia byť stručné a vecné ale zároveň by mali dostatočne výstižne opisovať problém, ktorý bol v rámci daného commitu riešený.

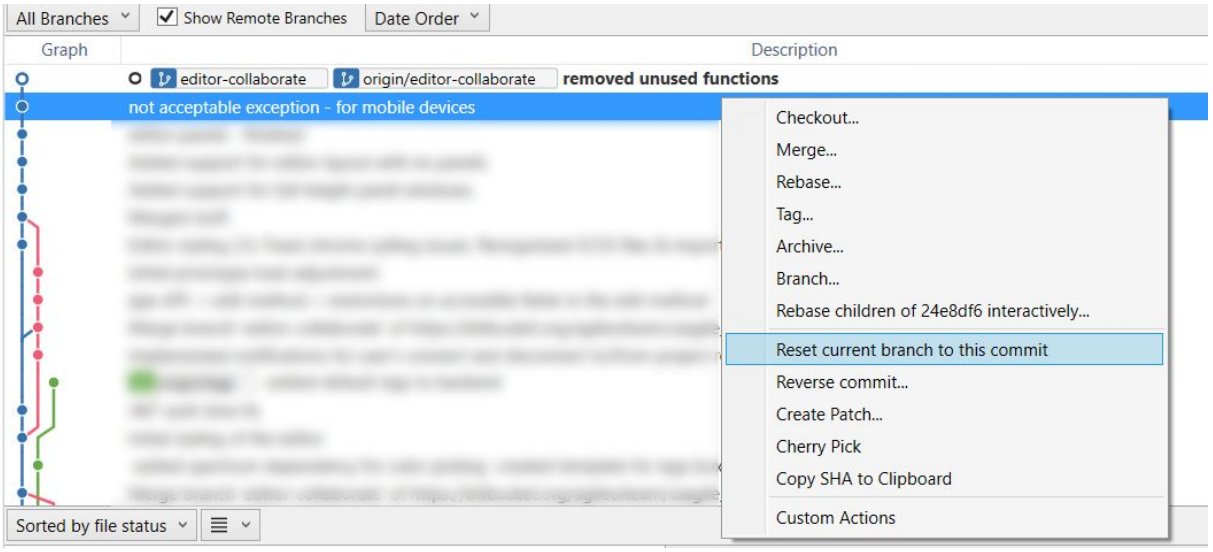
4.4 Návrat ku starším verziám projektu

V prípade potreby návratu ku staršej verzii projektu (napr. v prípade zavedenia závažných chýb novšou verziou) je možné sa vrátiť ku staršiemu commitu pomocou nasledovných krokov:

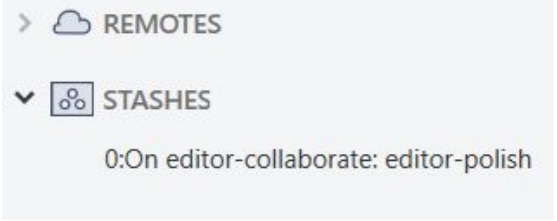
1. Lokálne zmeny si v prípade potreby uložíme pomocou príkazu *Stash* z horného menu aplikácie:



2. Zvolíme želaný starší commit v hlavnom prehľade a v jeho kontextovom menu zvolíme možnosť *Reset current branch to this commit:*



3. V nasledujúcom pop-up okne zvolíme mód *Hard* - t. j. nahradenie lokálnych zmien - a potvrdíme voľbu.
4. Stav projektu sa teraz nachádza v stave, v akom bol pri danom commite. Lokálne zmeny, ktoré sme predtým vykonali sú stále dostupné v *Stashi*, ktorý sme vytvorili v prvom kroku tohto procesu. Zoznam dostupných stashov môžeme vidieť pod záložkou *Stashes* v ľavom bočnom menu aplikácie:



Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Kolaboratívne prototypovanie [CollabUI]

Metodika nasadzovania

Tím: 24. CollabUI
Vedúci tímu: Ing. Eduard Kuric, PhD.
Členovia tímu: Adrián Nagy, Lukáš Vrba, Ján Kleň, Michal Melúch, Tomáš Mňačko, Peter Písecký, Miloslav Smetana
Študijný program: Inteligentné softvérové systémy
Ročník: 2017/2018

1. Úvod

Metodika nasadzovania definuje postupy pri nasadzovaní produkčnej verzie projektu, webovej stránky tímového projektu a aktualizácií dokumentov na tímovej stránke. Metodika sa viaže k nasadzovaniu na server s operačným systémom **Ubuntu 17.04**.

Touto metodikou sa riadia všetci členovia tímu č. 24 Collab-UI v rámci predmetu tímový projekt v akademickom roku 2017/2018.

2. Pojmy a skratky

SSH (angl. Secure shell) – zabezpečí prístup k príkazovému riadku
VNC (angl. Virtual network computing) – grafický program, ktorý umožňuje vzdialené pripojenie ku grafickému používateľskému rozhraniu pomocou počítačovej siete
DNS (angl. Domain name system) – systém, ktorý ukladá prístup k informácii o názve stroja a názve domény
HTML – hypertextový značkovací jazyk
PHP – skriptovací jazyk
Git – distribuovaný systém riadenia verzií
Virtual host – spôsob hostovania viacerých domén na jednom servery
Root directory – hlavný adresár hierarchie adresovej štruktúry
DB – databáza
JS - JavaScript
Package manager – kolekcia softvérových nástrojov, ktoré automatizujú proces inštalácie

3. Dôležité odkazy

3.1 Konfigurácia VNC

http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/html.html - časť poznámky k inštalácii servera
https://help.ubuntu.com/lts/serverguide/openssh-server.html - inštalácia SSH
https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-16-04 - inštalácia Apache
https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-14-04 - inštalácia Git
https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-16-04 - nastavenie Virtual host

3.2 Nasadenie produkčnej verzie

https://book.cakephp.org/3.0/en/installation.html - inštalácia CakePHP
https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions - inštalácia NodeJS
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-16-04 - inštalácia PostgreSQL
https://stackoverflow.com/questions/38975974/how-to-install-bower-on-ubuntu-16-04-lts - inštalácia Bower
https://github.com/datastax/php-driver/blob/master/ext/README.md - PHP driver pre PHP rozšírenia
https://github.com/datastax/php-driver - Cassandra PHP driver

4 .Pravidlá a postupy

Pri nasadzovaní je potrebné striktné dodržanie postupu krokov, nakoľko jednotlivé kroky majú medzi sebou rôzne závislosti. Pred samotným nasadením projektu a webovej stránky je dobré vedieť, ako sa potupuje pri úplne novom VNC, ktorý ešte nie je nakonfigurovaný, aby nebol problém ani pri zmene poskytovateľa serverových služieb.

4.1 Konfigurácia VNC

Na prihlásenie do VNC je potrebný program, ktorý s VNC dokáže pracovať, napríklad program RealVNC. Prihlásenie na virtuálny stroj je v tvare teamXX-17.studenti.fiit.stuba.sk, podľa informácií uvedených v tejto dokumentácií, v časti poznámky k inštalácií servera.

Pred samotnou konfiguráciou je potrebné definovať, ako budeme k virtuálnemu stroju v neskorších fázach pristupovať. V našom prípade sme zvolili možnosť prístupu cez SSH. Prikaz na inštaláciu SSH na server:

```
sudo apt install openssh-server
```

V nasledujúcich krokoch budeme vždy na server pristupovať nasledovne:

```
ssh PRIHLASOVACIE_MENO@teamXX-17.studenti.fiit.stuba.sk
```

Aby sme vedeli zabezpečiť nasadenie projektu a webovej stránky je potrebný webový server napríklad Apache. Inštalácia Apache je možná nasledujúcim príkazom:

```
sudo apt-get update
sudo apt-get install apache2
```

Pred inštaláciou akéhokoľvek rozšírenia je dobré najprv aktualizovať správcu inštalácie balíkov pomocou príkazu *sudo apt-get update*.

Aby sme nemuseli pri nasadzovaní ručne kopírovať súbory, tak na server nainštalujeme git nasledujúcim príkazom:

```
sudo apt-get install git
```

A prihlásime sa do gitu, z ktorého budeme získavať všetky dáta na nasadenie a aktualizáciu nasledovne:

```
git config --global user.name ''POUZIVATELSKE_MENO''
```

```
git config --global user.email ''POUZIVATELSKY EMAIL'
```

4.2 Nasadenie produkčnej verzie projektu

Pred nasadením produkčnej verzie projektu na server je potrebné nainštalovať požiadavky, ktoré sú nutnou súčasťou správneho fungovania projektu.

4.2.1 Konfigurácia virtual hosta

Nakoľko sa na servery nachádza viac adries, ktoré sa používajú je potrebné nastaviť virtual hosta, aby pri zadaní domény, na ktorej sa má nachádzať projekt smeroval do adresára s projektom. V našom prípade sme potrebovali umiestniť projekt, tak aby bol na doméne prvej úrovne, ale podľa zadania sa na tejto adrese musí nachádzať práve webová stránka tímu. Tým pádom sme museli kúpiť doménu, ktorej DNS záznamy sme presmerovali na náš server. Nastavenie virtual hosta prebieha v troch krokoch:

- Vytvorenie priečinka, v ktorom sa bude nachádzať projekt
- Vytvorenie nového virtual hosta v priečinku /etc/apache2/sites-available, podľa tohto dokumentu.
- Nastavenie nového virtual hosta, pričom root directory bude smerovať do novo vytvoreného priečinka z prvého bodu

4.2.2 Inštalácia požiadaviek na server

Na zistenie požiadaviek je najprv potrebné overiť, aké technológie boli na projekt použité a či je potrebné doinštalovanie požiadaviek na prácu s nimi. V našom prípade sa jedná o nasledujúce technológie, ktoré si vyžadujú inštaláciu dodatočných požiadaviek:

- CakePHP
- Cassandra
- PostgresSQL + vytvorenie DB
- NodeJS
- Bower

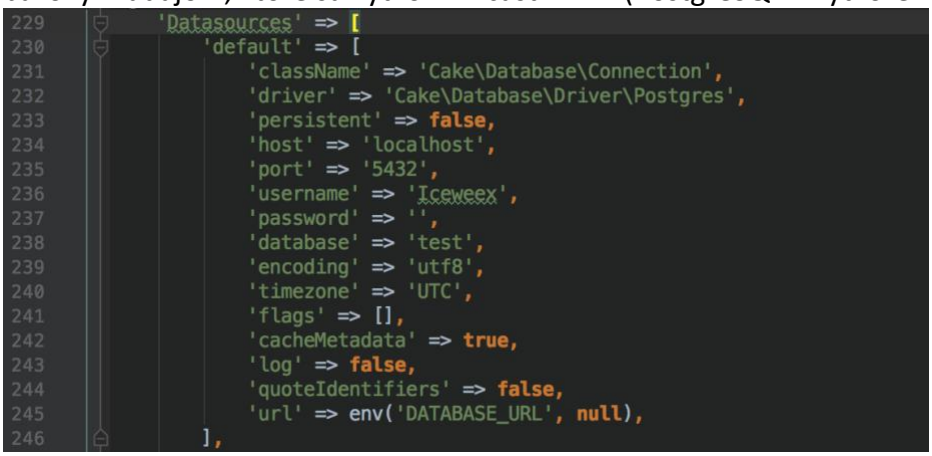
4.2.3 Spustenie projektu na servery

Po nastavení servera pomocou inštalácie potrebných požiadaviek je treba postupovať podľa nasledujúcich inštrukcií. Ako prvé je potrebné prihlásiť na server aa prejsť do priečinka, v ktorom sa bude do aktuálneho priečinka pomocou gitu:

```
git clone git@bitbucket.org:agileuiteam/uiagile_repo.git
git pull
```

Po skopírovaní repozitára je potrebné nastavenie projektu v niekoľkých krokoch:

- V priečinku */config/* sa nachádza súbor *app.default.php*, ktorý treba skopírovať do súboru *app.php*
- V súbore *app.php*, treba nastaviť údaje (viď. obrázok nižšie), zodpovedajúce databázovým údajom, ktoré sa vytvorili v časti 4.2.2 (PostgreSQL + vytvorenie DB)



Po nastavení pripojenia na relačnú databázu sa môžu spustiť migrácie oboch databáz pomocou príkazou pre PostgreSQL:

```
sudo -i u postgres
psql -U postgres -d postgres -a -f ''Root directory'/src/Database/collabui.sql
```

A pre Cassandru pomocou príkazu:

```
cqlsh -f ''Root directory'/src/Database/collabui.cql
```

Po migrácií databáz sa môžu aktualizovať PHP a JS package managers. PHP package manager Composer sa spustí príkazom z root directory:

```
composer install
```

A JS package manager Bower príkazom z root directory:

```
bower install
```

Nakoniec NPM package manager z priečinka /node/

```
npm install
```

Pre kompilovanie js a scss súborov je potrebné nastaviť Gulp a to skopríovaním súboru gulp.default.js v priečinku /node/ do súboru gulp.js a nastaviť údaje (viď. Obrázok nižšie).

```
//****************** CUSTOME SETTINGS ******************
var server = {
    base: '/Users/Iceweex/Sites/uiagile_repo', //ROOT FOLDER
    port: 80
};

var browser = {
    proxy: 'http://uiagile.com/',
    port: 3000
};
```

Následne sa môže spustiť kompilácia príkazom:

```
gulp watch
```

Ako posledné nastavenie je Socket.io, ktorému je potrebné prepísať adresu hosta na strane klienta (viď. Obrázok nižšie) v súbore */webroot/js/aoweb/components/ClientComponent.js*.

```
ClientComponent = function () {

  aoweb.component.call(this);

  this.host = 'http://localhost:8082/';

  this.loaderActive = undefined;
  this.socket = undefined;
};
```

Na spustenie Node servera z priečina */node/* použijeme nasledovný príkaz, ktorý ostanne pracovať aj po odhlásení používateľa zo servera.

```
nohup node Server.js &
```

4.3 Nasadenie webovej stránky

Na rozdiel od nasadenia produkčnej verzie webová stránka nemá žiadnu požiadavku na inštaláciu nakoľko sa jedná len o HTML web. Pri nasadení stránky do root directory */var/www/html* stačí len použiť nasledujúci príkaz Gitu:

```
Git clone git@bitbucket.org:POUZIVATELSKE_MENO/team_website.git
```

Pričom je potrebné mať najprv vytvorený repozitár s požadovanou webovou stránkou

4.4 Aktualizácia dokumentácie

Aktualizácia prebieha najprv úpravou stránky lokálne a následne sa nahrá s aktualizovanými súbormi do repozitára.

Pridanie nových dokumentov prebieha v štyroch krokoch:

- Nahranie nového dokumentu do adresára stránky, v našom prípade do priečinku */assets/docs/*
- Pridanie nového li elementu do index.html, pričom odkaz na dokument bude smerovať na novú adresu dokumentu z bodu 1

```
<li>
  <a class="btn-pdf btn btn-lg btn-common" href="assets/docs/zapisnice/Zapisnica5.pdf" target="_blank" style="width:100%;">
    <i class="fa fa-file-pdf-o"></i>
    27.10.2017
  </a>
  <p class="description hide"><b>Popis</b>: Pokračovanie 2.šprintu</p>
</li>
```

- Publikovanie aktualizácií do repozitára
- Stiahnutie aktualizácií na servery z priečinka */var/www/html* pomocou príkazu *git pull*