

**Dokument**  
**k inžinierskemu dielu**  
Tím 24

# Obsah

<b>1</b>	<b>Úvod</b>	<b>7</b>
<b>2</b>	<b>Globálne ciele pre ZS</b>	<b>8</b>
<b>3</b>	<b>Globálne ciele pre LS</b>	<b>9</b>
<b>4</b>	<b>Celkový pohľad na systém</b>	<b>10</b>
<b>4.1</b>	<b>Architektúra</b>	<b>10</b>
4.1.1	Backend	10
4.1.1.1	Manažment BE knižníc (CakePHP)	11
4.1.1.2	Mechanizmus spracovania chýb	11
4.1.1.3	Mechanizmus vykreslenia stránok	11
4.1.1.4	Databázový model, ORM	12
4.1.1.5	NodeJS Server	13
4.1.1.6	Manažment BE knižníc (NodeJS)	13
4.1.2	Frontend	14
4.1.2.1	Manažment FE knižníc	14
4.1.2.2	Buildovanie SASS a BrowserSync	15
4.1.2.3	Minifikácia JS a CSS, optimalizácia obrázkov	15
<b>5</b>	<b>Moduly</b>	<b>16</b>
<b>5.1</b>	<b>Manažment používateľov</b>	<b>17</b>
5.1.1	Úvod	17
5.1.2	Analýza	17
5.1.3	Návrh	17
5.1.4	Implementácia	21
5.1.4.1	Úvodná stránka	21
5.1.4.1.1	Akceptačné kritéria	21
5.1.4.1.2	Validácia	21
5.1.4.1.3	Obrazovky úvodnej stránky	22
5.1.4.2	Registrowanie a aktivovanie používateľského účtu	23
5.1.4.2.1	Akceptačné kritéria	23
5.1.4.2.2	Validácia	23
5.1.4.2.3	Databázový model	24
5.1.4.2.4	Sekvenčný diagram	25
5.1.4.2.5	Obrazovka registrowania a aktivovania používateľského účtu	26
5.1.4.3	Prihlásenie a odhlásenie sa zo systému	27
5.1.4.3.1	Akceptačné kritéria	27
5.1.4.3.2	Validácia	27
5.1.4.3.3	Obrazovka pre prihlásenie sa	28
5.1.4.4	Zabudnuté a nové heslo	29
5.1.4.4.1	Akceptačné kritéria	29
5.1.4.4.2	Validácia	29
5.1.4.4.3	Sekvenčný diagram	30
5.1.4.4.4	Obrazovka pre zabudnuté heslo	31
5.1.5	Testovanie	32
<b>5.2</b>	<b>Manažment projektov</b>	<b>33</b>
5.2.1	Úvod	33
5.2.2	Analýza	33
5.2.3	Návrh	33
5.2.4	Implementácia	37
5.2.4.1	Vytvorenie projektu	37
5.2.4.1.1	Akceptačné kritéria	38
5.2.4.1.2	Validácia	38

5.2.4.1.3	Databázový model.....	38
5.2.4.1.4	Obrazovky dashboardu .....	39
5.2.4.2	Zmazanie projektu .....	40
5.2.4.2.1	Akceptačné kritéria .....	40
5.2.4.2.2	Validácia .....	40
5.2.4.2.3	Databázový model.....	41
5.2.4.2.4	Sekvenčný diagram .....	42
5.2.4.2.5	Obrazovka pre mazanie projektu .....	43
5.2.4.3	Editovanie všeobecných informácií projektu.....	44
5.2.4.3.1	Akceptačné kritéria .....	44
5.2.4.3.2	Validácia .....	45
5.2.4.3.3	Stavový diagram – projekt.....	45
5.2.4.3.4	Obrazovka pre editovanie všeobecných informácií projektu .....	46
5.2.4.4	Notifikovanie kolaborantov na projekte .....	47
5.2.5	Testovanie .....	48
<b>5.3</b>	<b>Manažment kolaborantov.....</b>	<b>49</b>
5.3.1	Úvod .....	49
5.3.2	Analýza .....	49
5.3.3	Návrh .....	49
5.3.4	Implementácia .....	53
5.3.4.1	Pridelenie kolaboranta na projekt .....	53
5.3.4.1.1	Akceptačné kritéria .....	53
5.3.4.1.2	Validácia .....	53
5.3.4.1.3	Databázový model.....	54
5.3.4.1.4	Stavový diagram pre status používateľa .....	55
5.3.4.1.5	Obrazovka pre pridelenie kolaboranta na projekt .....	56
5.3.4.2	Zmena práv kolaboranta .....	57
5.3.4.2.1	Akceptačné kritéria .....	57
5.3.4.2.2	Validácia .....	57
5.3.4.2.3	Stavový diagram pre zmenu používateľských práv .....	58
5.3.4.2.4	Obrazovka zmeny práv používateľa .....	59
5.3.4.3	Odstránenie kolaboranta z projektu .....	60
5.3.4.3.1	Akceptačné kritéria .....	60
5.3.4.3.2	Validácia .....	60
5.3.4.3.3	Databázový model.....	61
5.3.4.3.4	Obrazovka pre odstránenie kolaboranta z projektu .....	62
5.3.4.4	Blokovanie kolaboranta .....	63
5.3.4.4.1	Akceptačné kritéria .....	63
5.3.4.4.2	Validácia .....	63
5.3.4.4.3	Databázový model.....	63
5.3.4.4.4	Obrazovka pre blokovanie kolaboranta .....	64
5.3.4.5	Notifikovanie kolaboranta mailom .....	65
5.3.4.5.1	Akceptačné kritéria .....	65
5.3.4.5.2	Validácia .....	65
5.3.4.5.3	Obrazovky pre notifikovanie kolaboranta.....	66
5.3.5	Testovanie .....	67
<b>5.4</b>	<b>Manažment tagov.....</b>	<b>68</b>
5.4.1	Úvod .....	68
5.4.2	Analýza .....	68
5.4.3	Návrh .....	68
5.4.4	Implementácia .....	70
5.4.4.1	Tagovanie projektu .....	70
5.4.4.1.1	Akceptačné kritéria .....	70
5.4.4.1.2	Validácia .....	71
5.4.4.1.3	Databázový model.....	71
5.4.5	Testovanie .....	72

<b>5.5</b>	<b>Editor</b> .....	<b>73</b>
5.5.1	Úvod .....	73
5.5.2	Analýza .....	74
5.5.3	Analýza podporného nástroja pre tvorbu stránok .....	75
5.5.3.1	Diagram tried, CSS composer.....	75
5.5.3.2	Sekvenčný diagram, zmena textového komponentu.....	76
5.5.3.3	Diagram tried, manažér selektorov .....	77
5.5.3.4	Sekvenčný diagram, pridanie komponentu do kolekcie komponentov.....	78
5.5.4	Návrh .....	79
5.5.4.1	Rozšírenie návrhu editora.....	83
5.5.5	Implementácia .....	87
5.5.5.1	Editor bez kolaborácie .....	87
5.5.5.1.1	Akceptačné kritéria .....	88
5.5.5.1.2	Validácia .....	88
5.5.5.1.3	Dátový model .....	89
5.5.5.1.4	Autorizácia .....	90
5.5.5.2	Kolaboratívny editor .....	91
5.5.5.2.1	Akceptačné kritéria .....	92
5.5.5.2.2	Validácia .....	92
5.5.5.2.3	Práca s prototypmi .....	93
5.5.5.2.4	Synchronizácia zmien v prototypoch .....	97
5.5.5.2.5	Vizualizácia označených komponentov.....	106
5.5.5.2.6	Uzamykanie komponentov .....	112
5.5.6	Testovanie .....	117
<b>6</b>	<b>Záverečné testovanie</b> .....	<b>118</b>
<b>6.1</b>	<b>Testovacie scenáre</b> .....	<b>118</b>
6.1.1	Registrácia a prihlásenie .....	118
6.1.2	Správa projektov a Dashboard .....	119
6.1.3	Práca s editorom #1.....	120
6.1.4	Práca s editorom #2.....	121
6.1.5	Práca s editorom #3.....	122
<b>6.2</b>	<b>Dotazník</b> .....	<b>124</b>
<b>6.3</b>	<b>Výstup z testovania</b> .....	<b>125</b>
<b>7</b>	<b>Záver</b> .....	<b>127</b>



## Obsah obrázkov

Obrázok 1 - Celkový databázový model pre postgresql .....	10
Obrázok 2 - Tabuľky pre databázu Cassandra .....	13
Obrázok 3 - UC diagram pre manažment používateľov .....	18
Obrázok 4 - Obrázok pre úvodnú stránku .....	22
Obrázok 5 - Databázový model registráciu používateľa .....	24
Obrázok 6 - Sekvenčný diagram pre registráciu používateľského účtu .....	25
Obrázok 7 - Obrázok pre registrovanie sa do systému .....	26
Obrázok 8 - Obrázok pre prihlásenie sa do systému .....	28
Obrázok 9 - Sekvenčný diagram pre zabudnuté heslo .....	30
Obrázok 10 - Obrázok pre obnovu hesla .....	31
Obrázok 11 - UC diagram pre manažment projektov .....	34
Obrázok 12 - DB Model pre vytváranie projektu .....	38
Obrázok 13 - Obrázok pre vytvorenie projektu .....	39
Obrázok 14 - V rámci mazania projektov sa používa projects tabuľka .....	41
Obrázok 15 - Sekvenčný diagram pre mazanie projektu .....	42
Obrázok 16 - Obrázok pre mazanie projektu .....	43
Obrázok 17 - Stavový diagram pre projekt .....	45
Obrázok 18 - Obrázok pre editovanie informácií projektu .....	46
Obrázok 19 - Obrázok pre notifikovanie kolaborantov projektu .....	47
Obrázok 20 - UC diagram pre manažment kolaborantov .....	50
Obrázok 21 - DB Model pre pridelenie kolaboranta na projekt .....	54
Obrázok 22 - Stavový diagram pre status používateľa .....	55
Obrázok 23 - Obrázok pre pridelenie kolaboranta .....	56
Obrázok 24 - Stavový diagram pre zmenu používateľských práv .....	58
Obrázok 25 - Obrázok zmeny práv používateľa .....	59
Obrázok 26 - DB Model pre odstránenie kolaboranta .....	61
Obrázok 27 - Obrázok pre odstránenie kolaboranta z projektu .....	62
Obrázok 28 - Tabuľka project_relations pre blokovanie kolaboranta .....	63
Obrázok 29 - Obrázok pre blokovanie kolaboranta .....	64
Obrázok 30 - Obrázky pre notifikovanie kolaboranta .....	66
Obrázok 31 - Autentifikácia používateľa v teste .....	67
Obrázok 32 - UC diagram pre manažment tagov .....	68
Obrázok 33 - Databázový model pre manažment tagov .....	71
Obrázok 34 - Diagram tried pre CSS Composer .....	75
Obrázok 35 - Sekvenčný diagram pre zmenu textového komponentu .....	76
Obrázok 36 - Diagram tried pre manažér selektorov .....	77
Obrázok 37 - Diagram tried pre pridanie komponentu do kolekcie komponentov .....	78
Obrázok 38 - UC diagram pre editor .....	79
Obrázok 39 - UC diagram pre rozšírenie návrhu editora .....	83
Obrázok 40 - Databázový Model pre editor .....	89
Obrázok 41 - NoSQL databázový model pre editor .....	89
Obrázok 42 - Obrázok pre vytvorenie novej stránky .....	93
Obrázok 43 - Obrázok pre mazanie stránky .....	94
Obrázok 44 - Obrázok pre panel kolaborantov .....	95
Obrázok 45 - Obrázok pre panel stránok .....	96
Obrázok 46 - Posielanie zmien vykonaných v editore ostatným klientom (všeobecný prípad) .....	97
Obrázok 47 - Udalosti súvisiace s označením komponentu v editore po jeho načítaní .....	106

Obrázok 48 - Zvýraznenie komponentu, ktorý je práve označený iným kolaborantom .....	107
Obrázok 49 - Zvýraznenie označených komponentov medzi dvoma rôznymi kolaborantami súčasne.....	107
Obrázok 50 - Uzamykanie/odomykanie komponentov.....	112
Obrázok 51 - Odomknutý komponent .....	113
Obrázok 52 - Uzamknutý komponent.....	113
Obrázok 53 - Zoznam uzamknutých komponentov .....	113
Obrázok 54 - Očakávaný výstup pre scenár: Registrácia a prihlásenie .....	118
Obrázok 55 – Očakávaný výstup pre scenár: Správu projektov a Dashboard .....	119
Obrázok 56 - Očakávaný výstup pre scenár: Práca s editorom #1 .....	120
Obrázok 57 - Očakávaný výstup pre scenár: Práca s editorom #2 .....	121
Obrázok 58 - Očakávaný výstup pre scenár: Práca s editorom #3 .....	122
Obrázok 59 - Dotazník po realizácii testovania .....	124

# 1 Úvod

Prioritou nášho tímu je vytvoriť webovú aplikáciu, ktorá má v prvom rade slúžiť ľuďom, zjednodušať 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 príde zaujímavé napr. pridanie verziovania jednotlivých prototypov, ukladanie histórie zásahov do prototypu aby sme vedeli proces 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 zahŕňa registráciu a prihlásenie používateľov, vytvorenie úvodnej stránky pre nalákatie zákazníkov nato aby používali náš produkt.

Ďalej by sme sa radi venovali manažmentu projektov. Teda ak sa už používateľ registruje do systému predpokladáme, 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 vedú 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 zamýšľame 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 reálnom čase čo kolaborant robí, taktiež bude vidieť tiež zoznam kolaborantov, ktorí akurát editujú prototyp.

### 3 Globálne ciele pre LS

Ako sme už spomínali v zimnom semestri plánujeme s dokončím zjednodušenej verzie editora bez kolaborácie. V ďalšom semestri máme na pláne obohatiť editor o našu najväčšiu pridanú hodnotu a to kolaboráciu. V prípade, že sa nám podarí proces kolaborácie používateľov vyvinúť a odladiť v krátkom čase pristúpime k realizácii niektorých z nasledovných User Story (priorita klesá smerom zhora nadol):

- Možnosť vytvárania viacerých stránok v jednom projekte
- Hierarchické prelinkovanie stránok
- Interaktívny prototyp, preview
- Verziovanie projektov
- Ukladanie histórie akcií na prototype
- Chat
- Audio Chat
- Pridávanie poznámok k jednotlivým elementom
- Pridávanie hlasových poznámok k jednotlivým elementom
- Prehranie histórie (retrospektíva)
- Profil používateľa

Hlavným cieľom letného semestra je dokončiť, odladiť kolaboratívny editor a zdokumentovať technické aspekty, kvôli zjednodušeniu začiatkov budúceho vývoja nasledovného tímu.

## 4 Celkový pohľad na systém

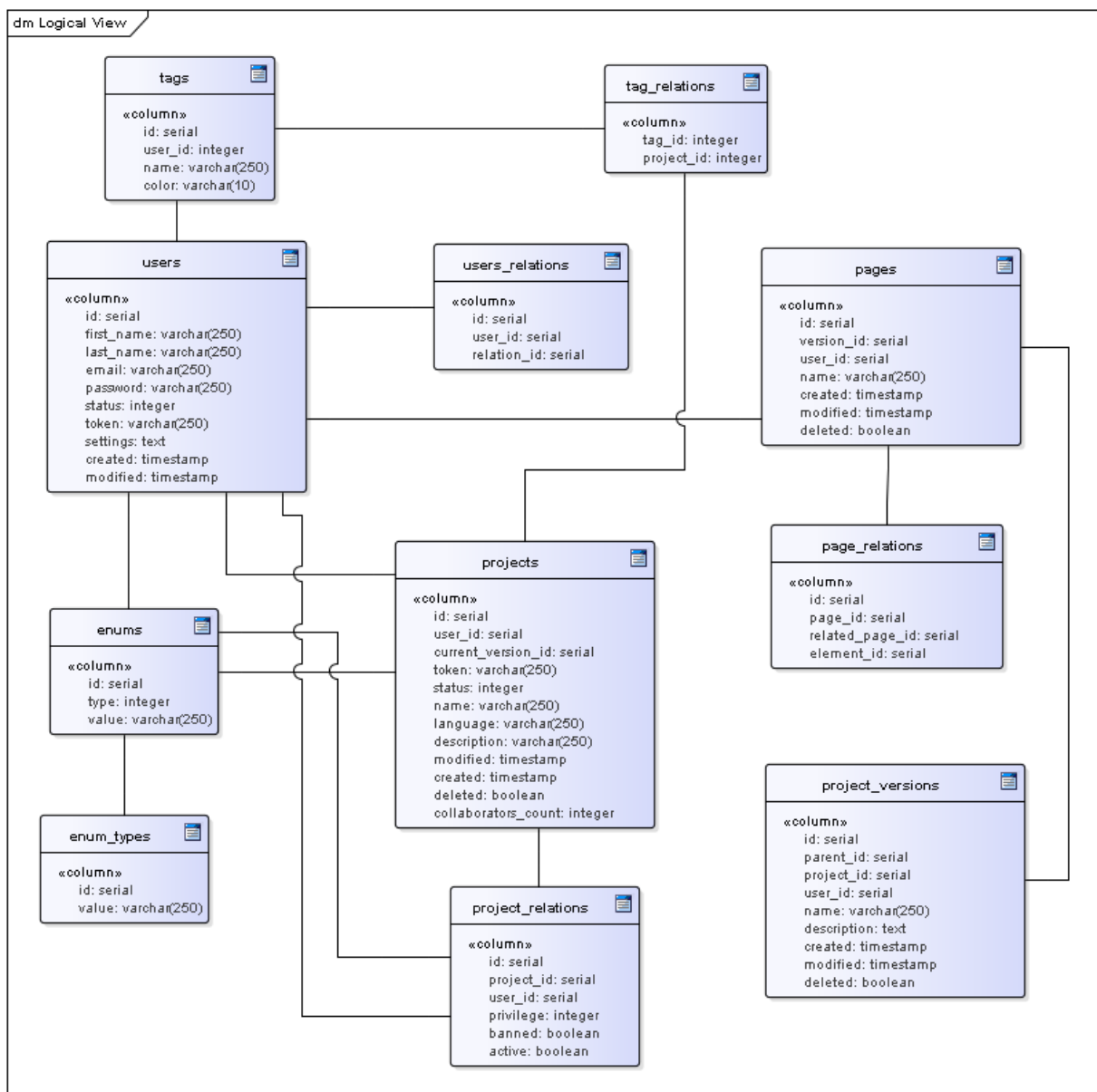
### 4.1 Architektúra

#### 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 implementácia 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 priečinkov 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
```

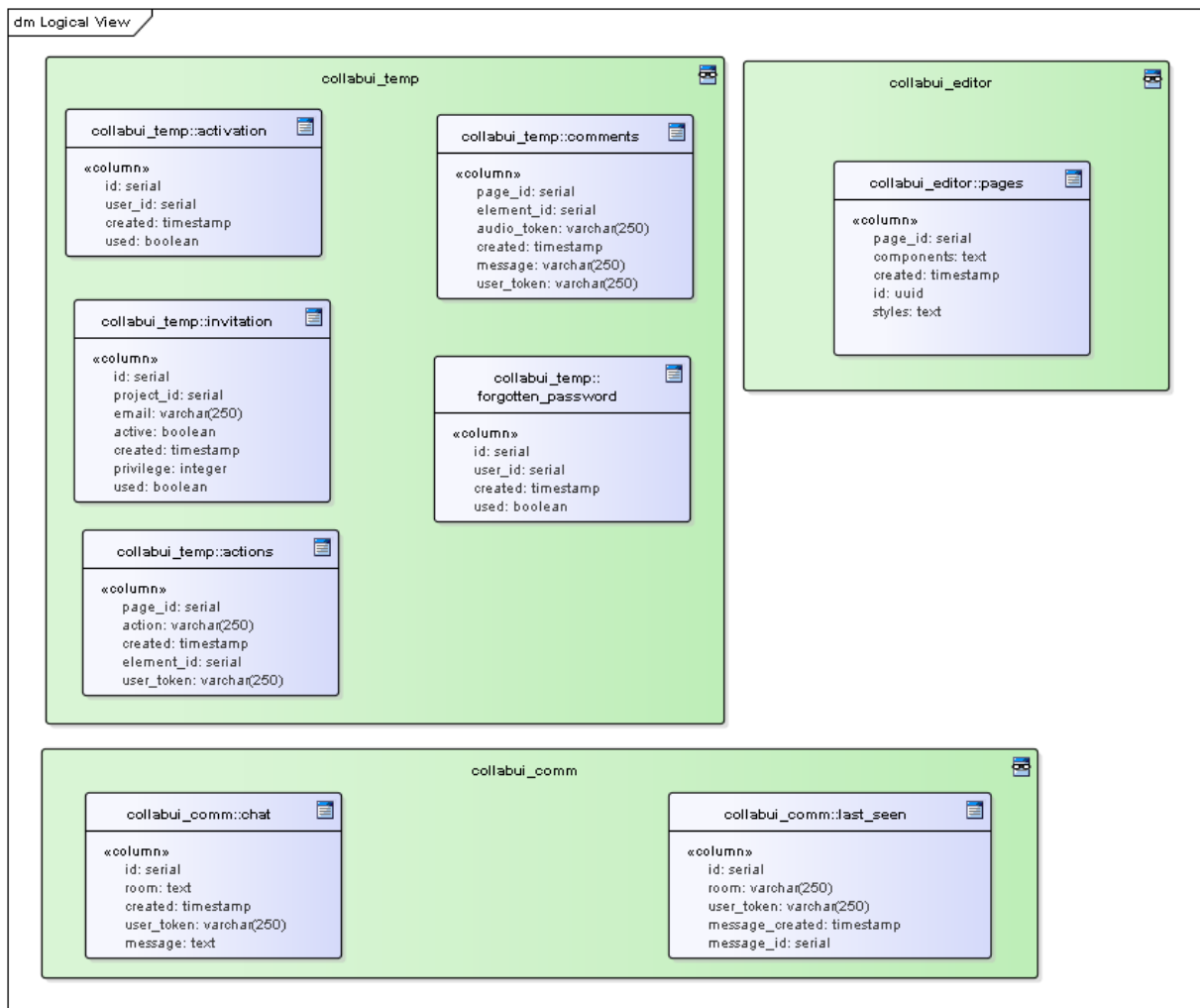
Databáza Cassandra obsahuje v súčasnosti 3 keyspace:

- `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

Priečinko 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.

#### 4.1.2 Frontend

Ako ste sa už dočítali vyššie šablóny sú v CakePHP označované 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, rozbiehajú ju na viacero menších súčiastok.

Nachádzajú sa v priečinku:

root/src/Template/Element/...

Okrem šablón patria k frontendu samozrejme aj css a js.

##### 4.1.2.1 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

Priečinko 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
```

#### **4.1.2.2 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 potrebné 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 zadanom porte a bude reagovať na zmeny v zdrojových súboroch .ctp, .js a .scss

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

#### **4.1.2.3 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
- Manažment tagov
- Editor

## 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 funkcií systému, technológie, s ktorými bol systém vytvorený, dodatočné služby poskytované v rámci jednotlivých častí projektu, kto na tomto projekte pracoval a vlastne ako dokáže sám používateľ tento systém 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 funkciami systému.

### 5.1.2 Analýza

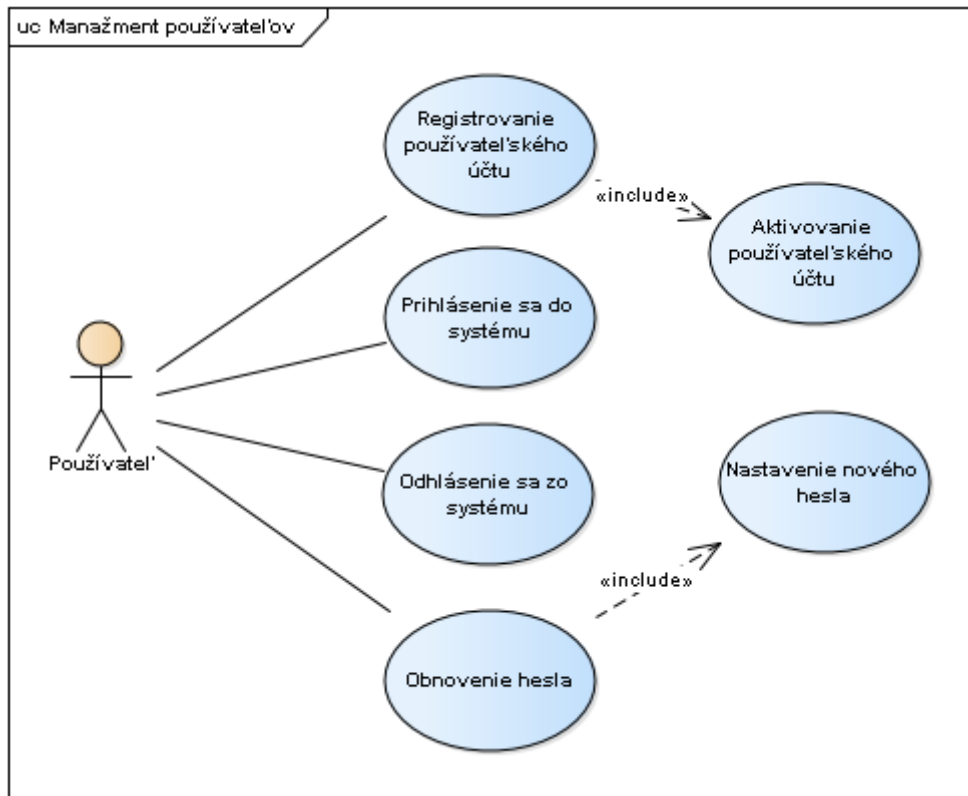
V rámci analýzy sme sa zameriavali nato kto bude náš hlavný používateľ. Náš používateľ by predovšetkým mal byť niekto kto je technicky zdatný. Primárne sa teda budeme sústreďovať na dizajnérov a programátorov popr. frontend developerov. Týchto používateľov sme vybrali práve kvôli špecifickosti systému. 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 presmerovaný na podstránku nastavenia nového hesla
3. Používateľ zadá nové heslo a potvrdí svoju akciu
4. Systém zvaliduje 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 obsahuje 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 obsahujú 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.

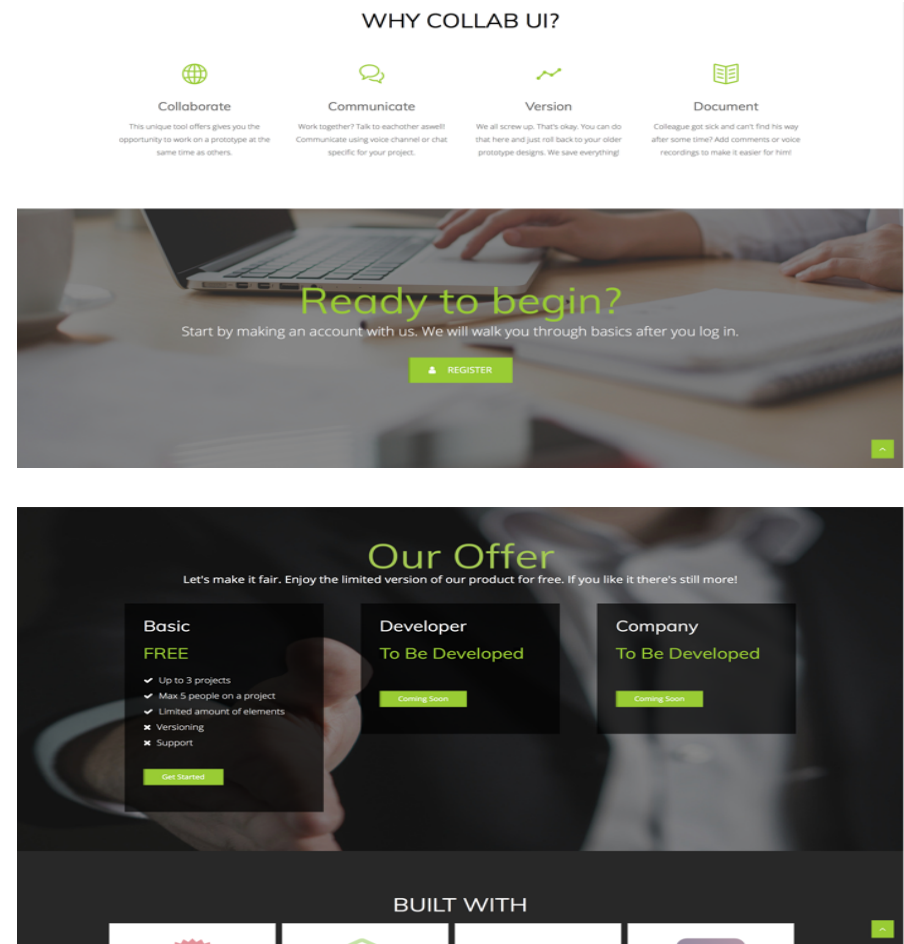
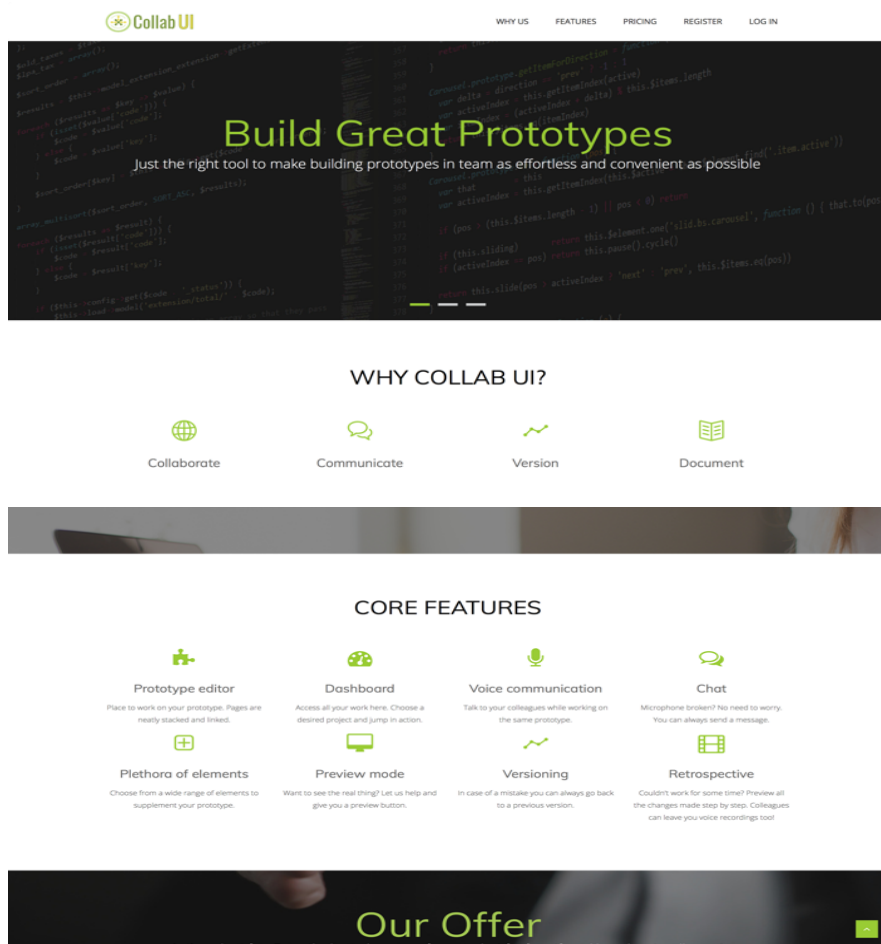
#### 5.1.4.1.1 Akceptačné kritéria

- 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 dôležitá 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 funkcionálnosť sa nachádza v utilities.js a potom sa len binduje na inputy s heslom.

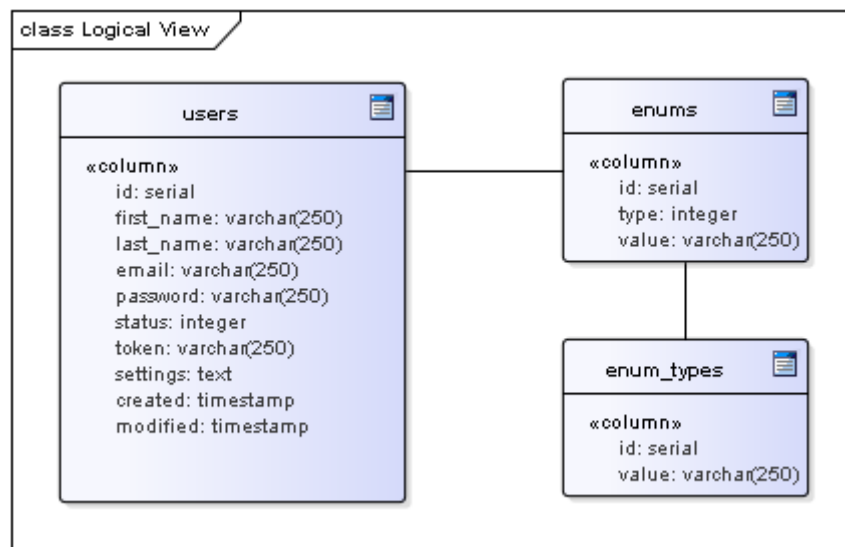
Čo sa týka backendovej validácie používateľa, tá je vykonávaná pomocou validačných funkcií v UserTable.php, kde sa pre každý typ formuláru zavolá iná validačná funkcia. Naviazanie na formulár a validačných funkcií prebieha v UserController.

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 administrátor používateľa zabanoval, pri prihlásení uvidí správu "This account has been banned."

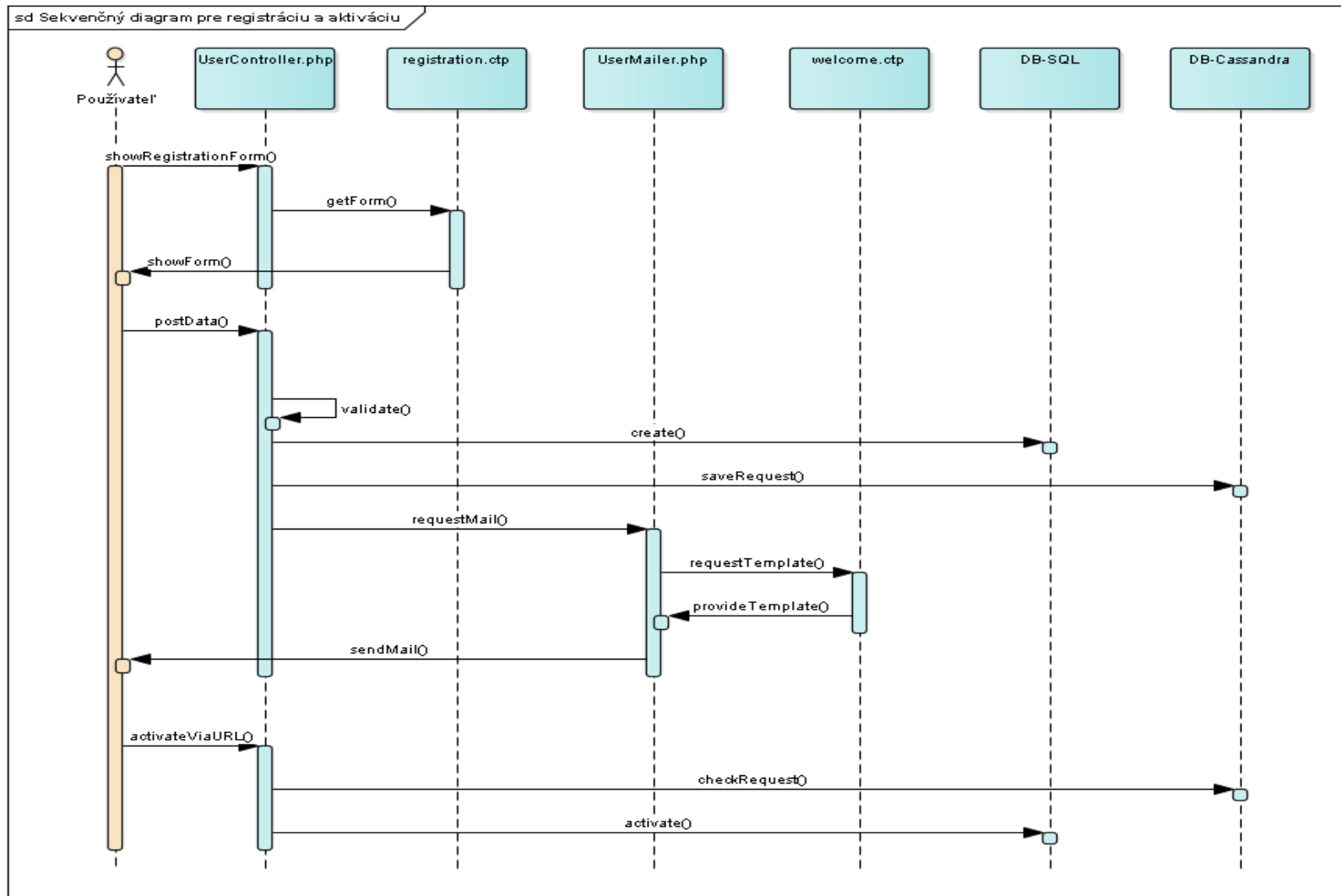
#### 5.1.4.2.3 Databázový model

Čo sa týka databázového pohľadu, sa pri registrácii a aktivácii používateľa používajú nasledovné tabuľky.



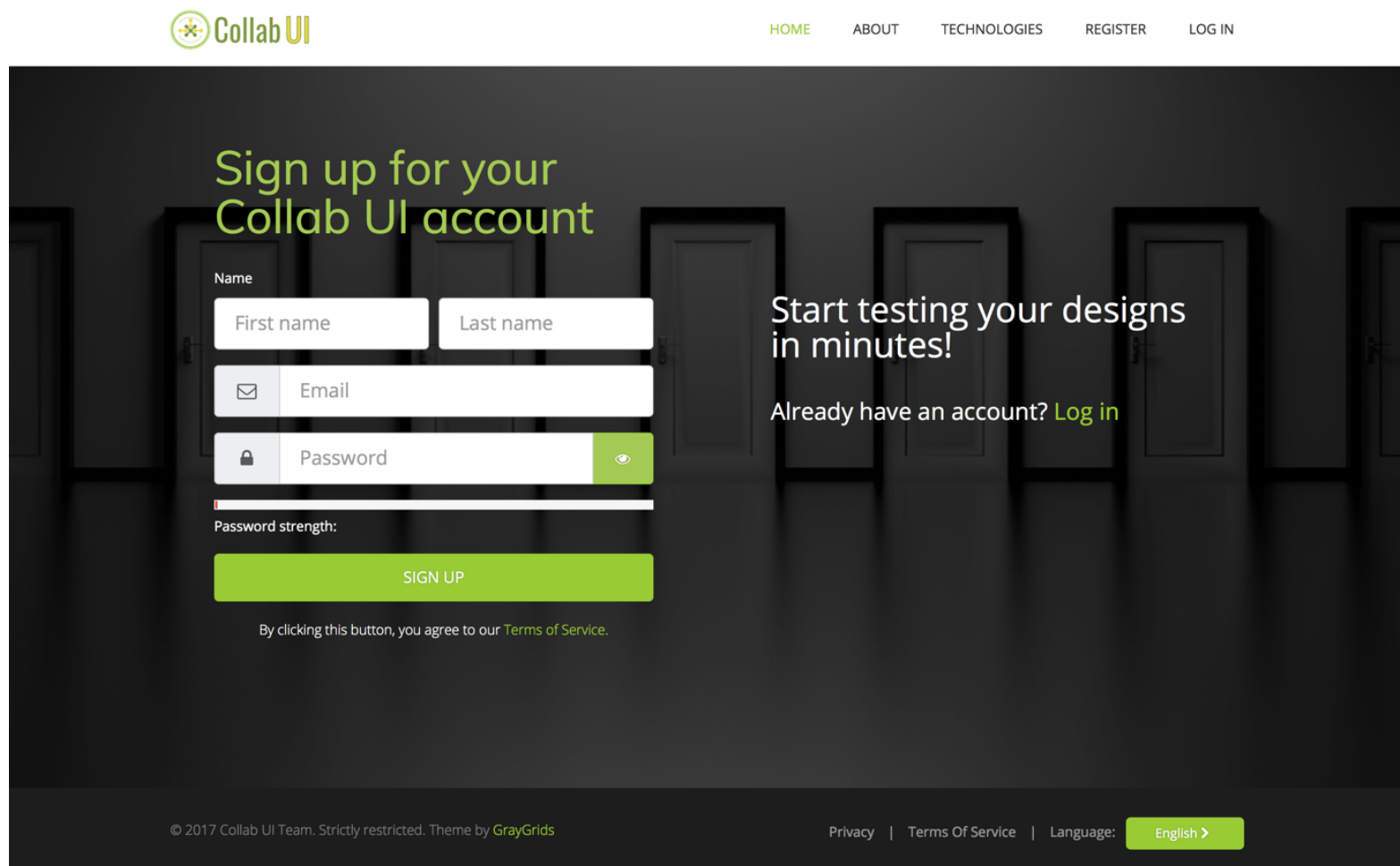
Obrázok 5 - Databázový model registráciu používateľa

### 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 odhlásenie: /log-out

Pri prihlásení používateľ taktiež vyplní 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

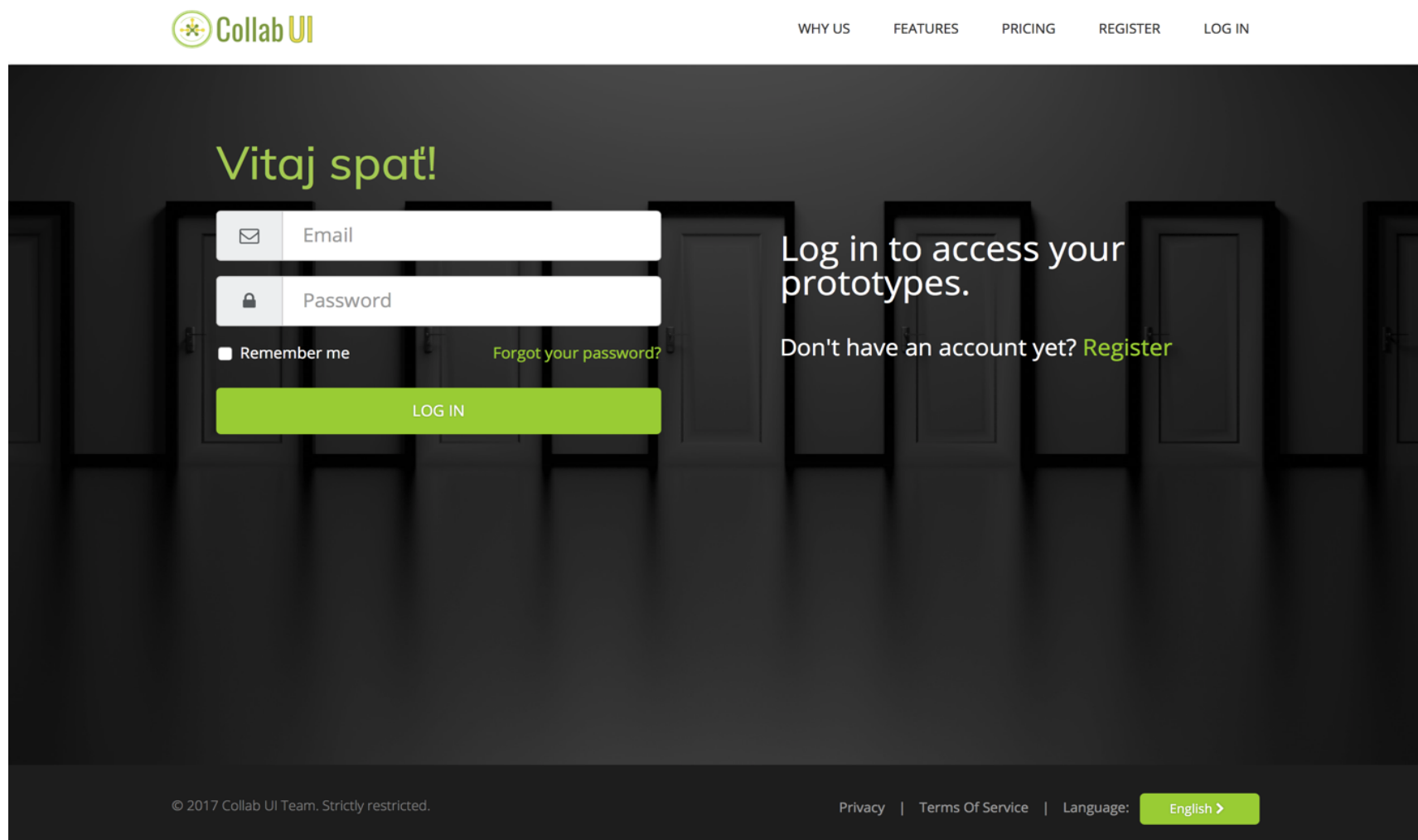
Čo sa týka frontendu, kontroluje sa či jednotlivé textové polia boli vyplnené alebo nie. V prípade, že vyplnené neboli tak sa pri príslušnom polí prostredníctvom animácie zobrazí modálne okno, ktoré používateľa upozorní o nevyplnení daného pola a vyzve ho k jeho vyplneniu.

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 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 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

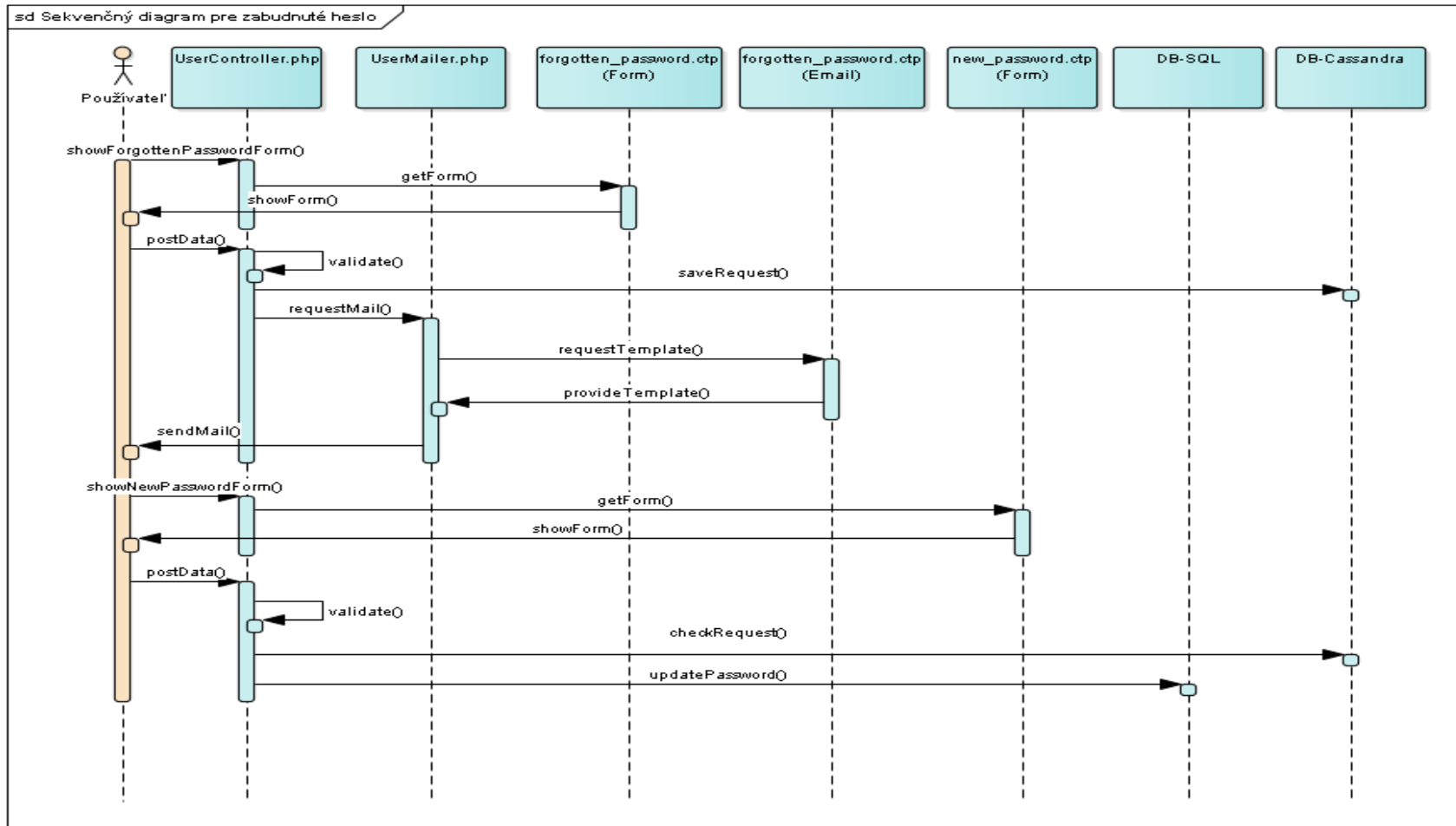
##### **5.1.4.4.1 Akceptačné kritéria**

- 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: /login s informáciou o zmene jeho hesla
- Po prihlásení s novým heslom sa používateľ dostane na URL: /dashboard

##### **5.1.4.4.2 Validácia**

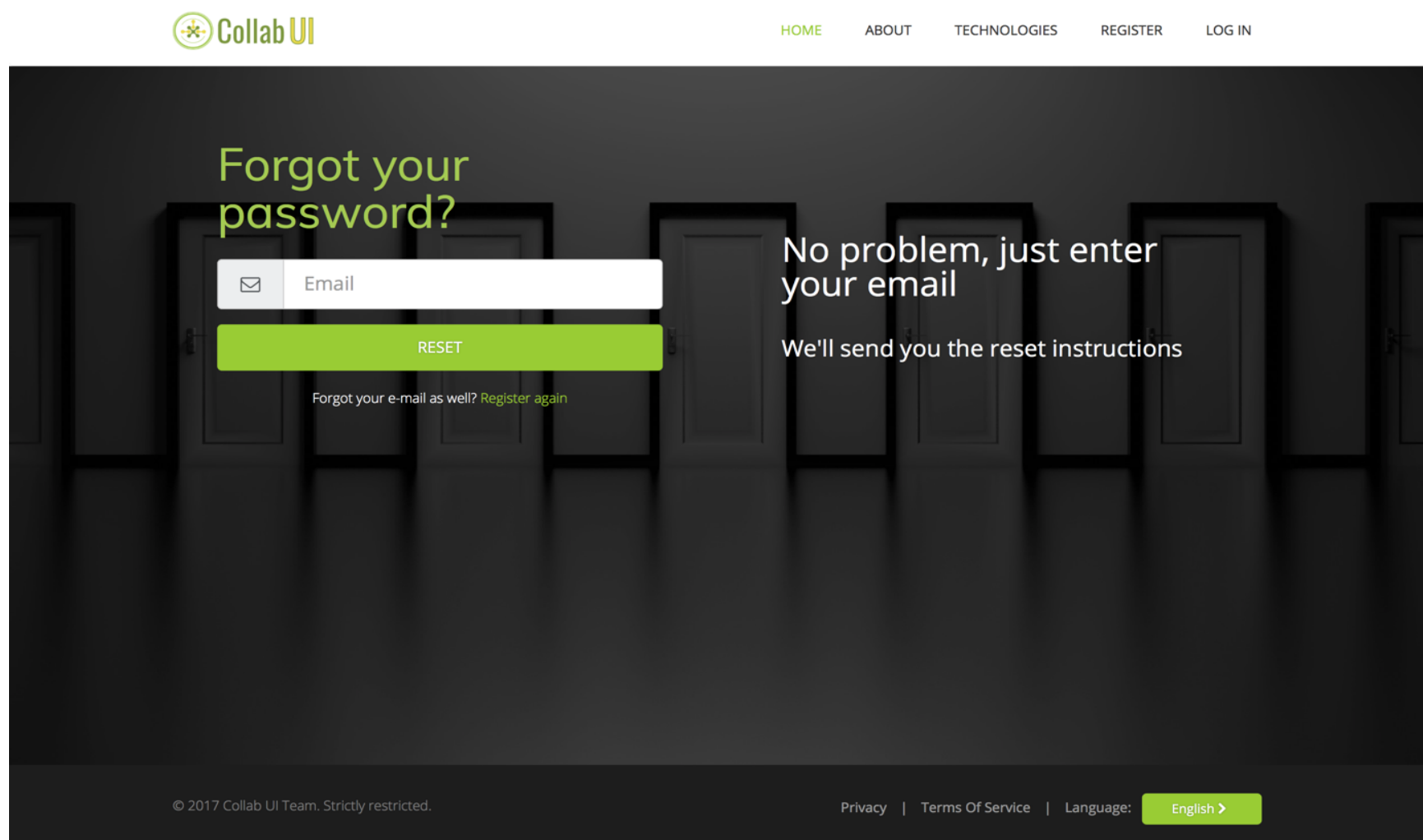
V rámci validácie zabudnutého hesla sa pri nastavovaní hesla kontroluje či je používateľ v systéme zaregistrovaný. Taktiež sa kontroluje email, z ktorého bol používateľ presmerovaný na podstránku zabudnutého hesla.

### 5.1.4.4.3 Sekvenčný diagram



Obrázok 9 - Sekvenčný diagram pre zabudnuté heslo

#### 5.1.4.4 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 scenáre:

- 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 najzákladnejších 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 možnosť prispôsobovania si daného projektu potrebuje mať k dispozícii funkciu editovania projektu, aby vedel bližšie špecifikovať názov projektu, spísať opis projektu, ktorý bude môcť dať ostatným členom tímu lepší prehľad o danom projekte. Tak isto sa bude môcť dozvedieť kedy bol projekt vytvorený, vlastníkom 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

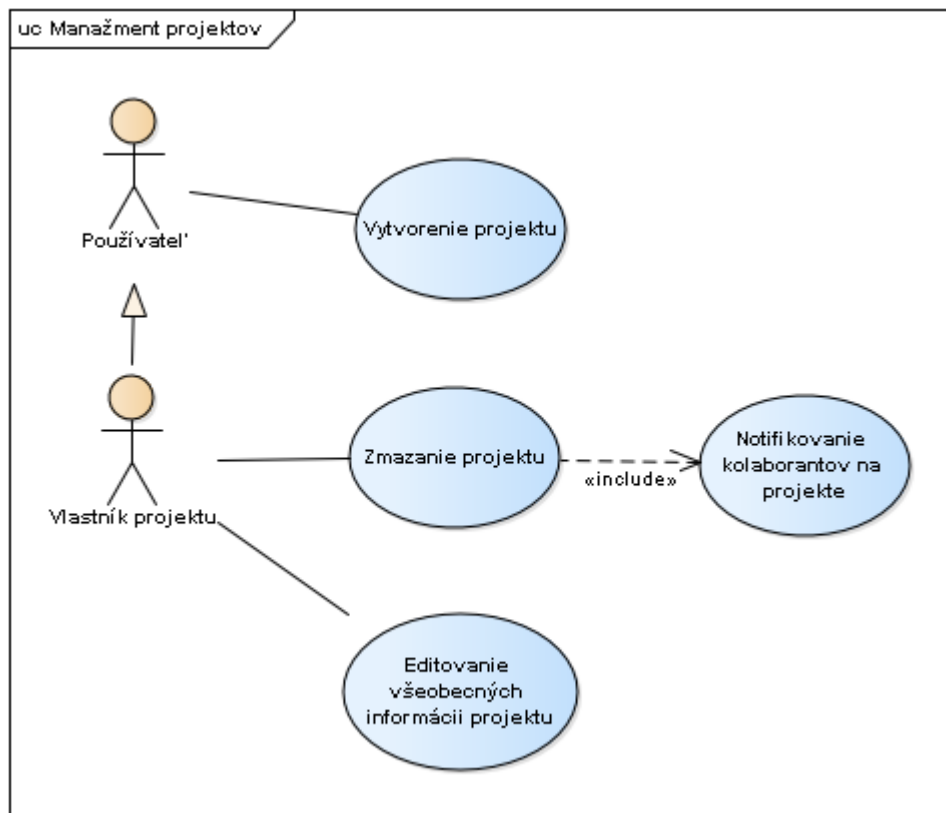
V rámci analýzy sme sa zameriavali rozvrhnutie projektu v rámci jednoduchosti používania a vytvárania projektov používateľom. Veľa krát môže byť pri webových aplikáciách problém práve v rozvrhnutí elementov tejto aplikácie. Ak používateľ nevie intuitívne nájsť riešenie na svoj aktuálny problém, dizajn aplikácie nebude zrovna najlepší. Snažili sme sa teda inšpirovať moderným dizajnom webových aplikácií s jednoduchou myšlienkou. Používateľ po prihlásení 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 chybnéj voľby nezmazal celý projekt. Okrem iného by mal byť informovaný o jednotlivom dianí v aplikácii. Toto riešime pomocou obľúbených notifikácií, 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 informácií 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. Prípád použitia končí

## **UC Editovanie všeobecných informácií 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. Prípád 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. Prípád použitia končí



## 5.2.4 Implementácia

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

- Aoweb<sup>1</sup>
- Crud plugin pre CakePHP<sup>2</sup>
- Bootstrap notify<sup>3</sup>
- Bootstrap modal<sup>4</sup>

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 plugin 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 prehľadných frontend modelov. Disponuje s veľkým množstvom užitočných funkcionalít, ktoré zjednodušujú a uľahčujú prácu programátora. AOWeb je semestrálny projekt 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)

---

<sup>1</sup> <https://github.com/Ado20/aoweb>

<sup>2</sup> <https://github.com/FriendsOfCake/crud>

<sup>3</sup> <http://bootstrap-notify.remabledesigns.com/>

<sup>4</sup> <https://v4-alpha.getbootstrap.com/components/modal/>

### 5.2.4.1.1 Akceptačné kritéria

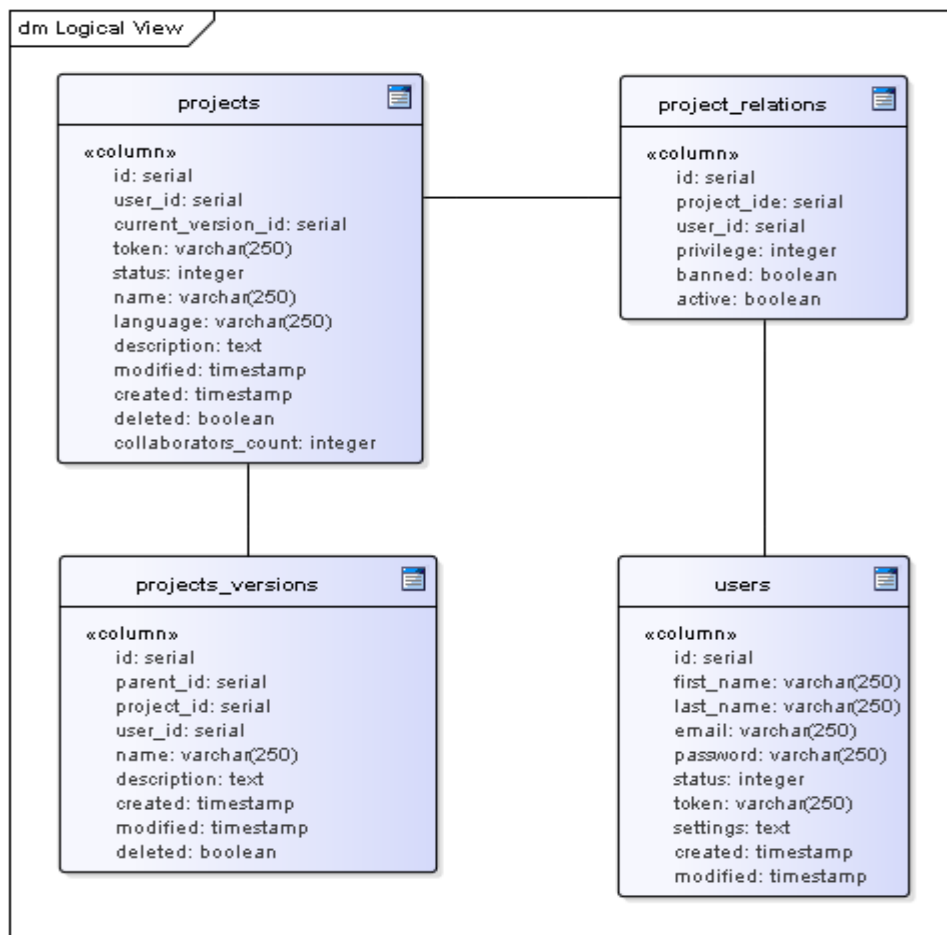
- Používateľ dokáže vytvoriť nový projekt vo verzii 1.0.0
- Po vytvorení projektu je presmerovaný na detail projektu
- Používateľ dokáže zmeniť základné informácie o projekte

### 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

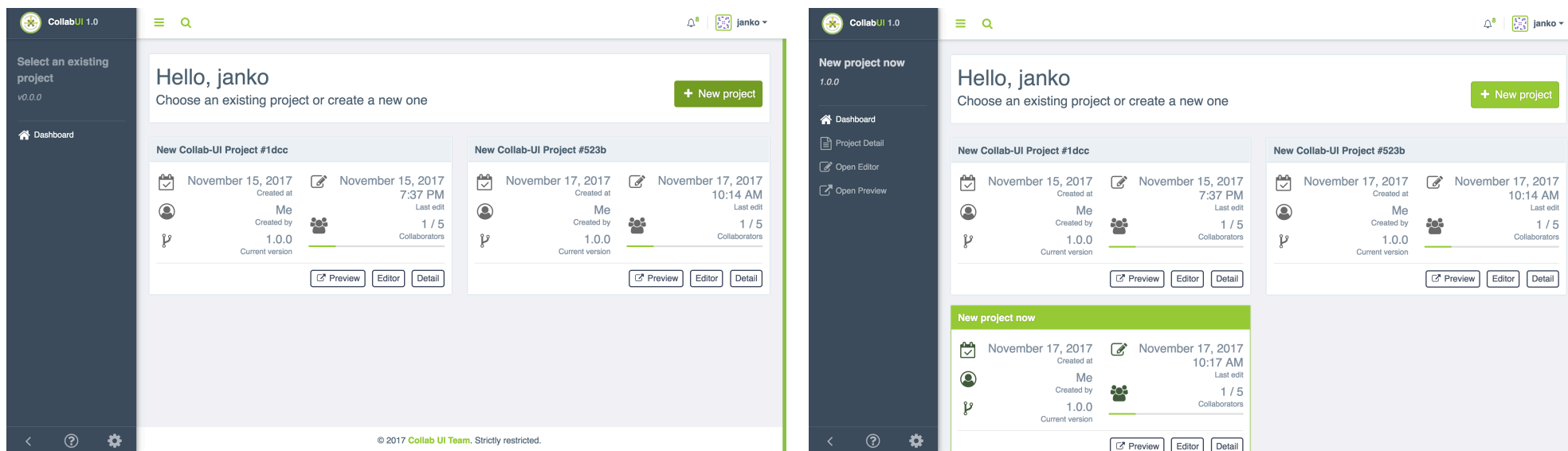
Čo sa týka databázového pohľadu, sa pri vytváraní projektu používajú nasledovné tabuľky.



Obrázok 12 - DB Model pre vytváranie projektu

Pri vytváraní projektu sa vlastník daného projektu deteguje 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 môcť byť iba projekt, ktorý bol vytvorený jeho vlastníkom. Mazanie nebude sprístupnené pre používateľov, ktorí nie sú jeho vlastníkami.

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 zmazení projekt títo používatelia notifikujú prostredníctvom emailu, ktorý im oznámi, že už ďalej nie sú súčasťou 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**

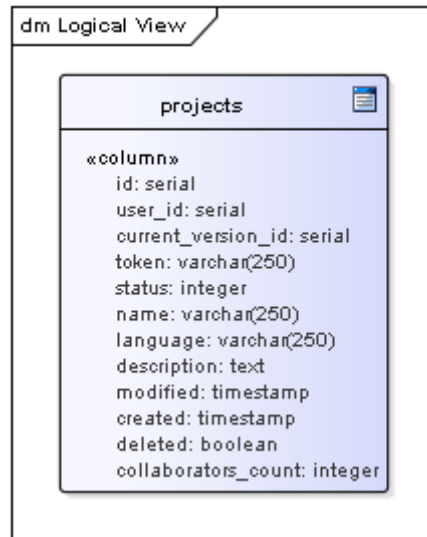
- Odstránenie projektu musím najskôr potvrdiť
- Dokáže používateľ odstrániť projekt
- Po odstránení bude presmerovaný na Dashboard a notifikovaný o úspešnej operácii
- Po vymazaní projektu sa projekt (a jeho relations) reálne nezmaže ale sa len označí v DB ako deleted.
- Po stlačení tlačidla musí byť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

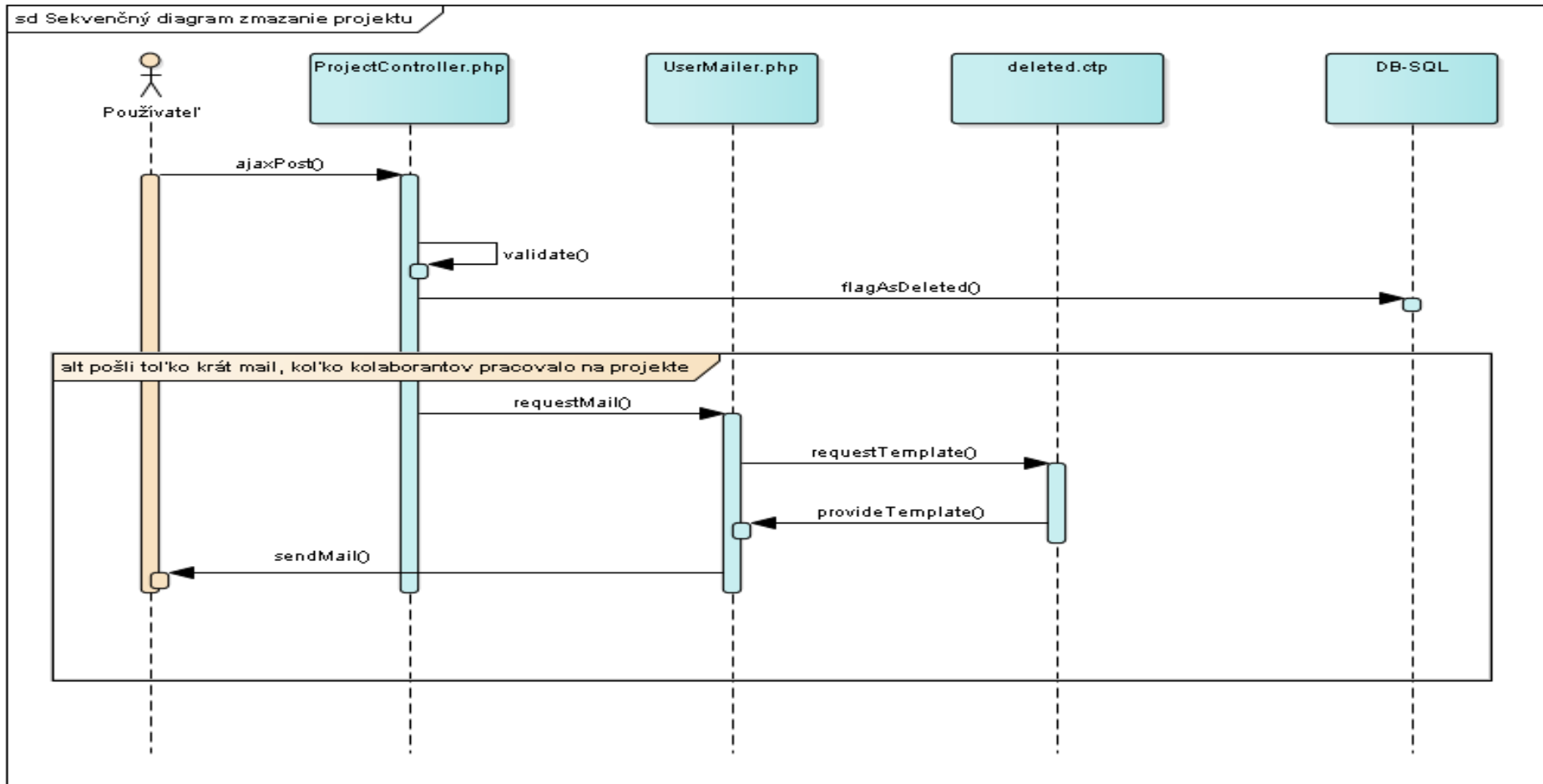
Čo sa týka databázového pohľadu, sa pri mazaní projektu používa iba tabuľka projects, v ktorej sa nastaví flag pre mazanie projektu "deleted" na true. User relation v tabuľke UserRelations sa ale nezmaže.



Obrázok 14 – V rámci mazania projektov sa používa projects tabuľka

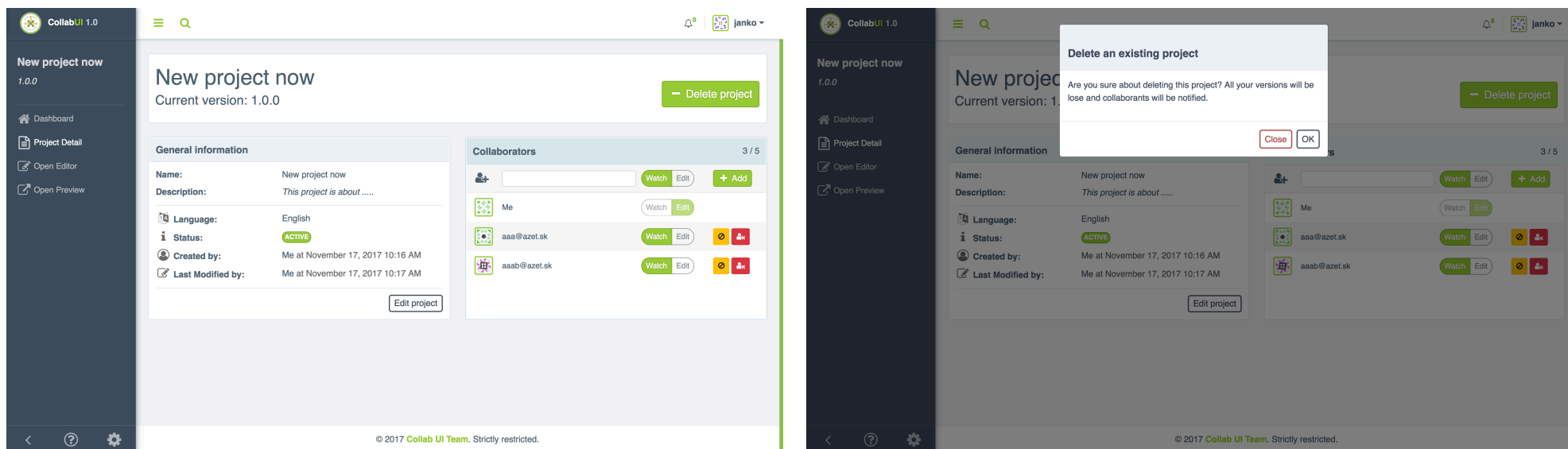
V rámci budúcej práce by sme chceli zintegrovať databázový cron, ktorý by každý deň o polnoci prešiel všetky záznamy, záznamy s flagom deleted by zarchivoval a v poslednom kroku vymazal z databázy.

### 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 zmazení projektu. Taktiež dáva používateľovi možnosť potvrdiť akciu alebo zrušiť akciu.

Obrázok 16 - Obrazovka pre mazanie projektu

### **5.2.4.3 Editovanie všeobecných informácií projektu**

Samotné editovanie projektu je najdôležitejšou časťou manažmentu projektov. Je to z toho hľadiska, 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

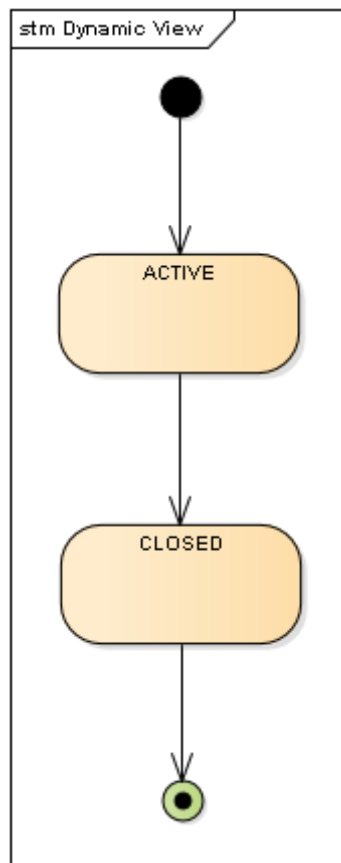


### 5.2.4.3.2 Validácia

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 informácií projektu

**General information**

**Name:** Project007

**Description:** This project is only for testing purposes.

**Language:** English

**Status:** ACTIVE CLOSED

**Created by:** Me at November 15, 2017 7:37 PM

**Last Modified by:** Me at November 15, 2017 7:37 PM

Cancel Save

**General information**

**Name:** Project007

**Description:** This project is only for testing purposes.

**Language:** English

**Status:** ACTIVE

**Created by:** Me at November 17, 2017 10:27 AM

**Last Modified by:** Me at November 17, 2017 10:27 AM

Cancel Save

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:

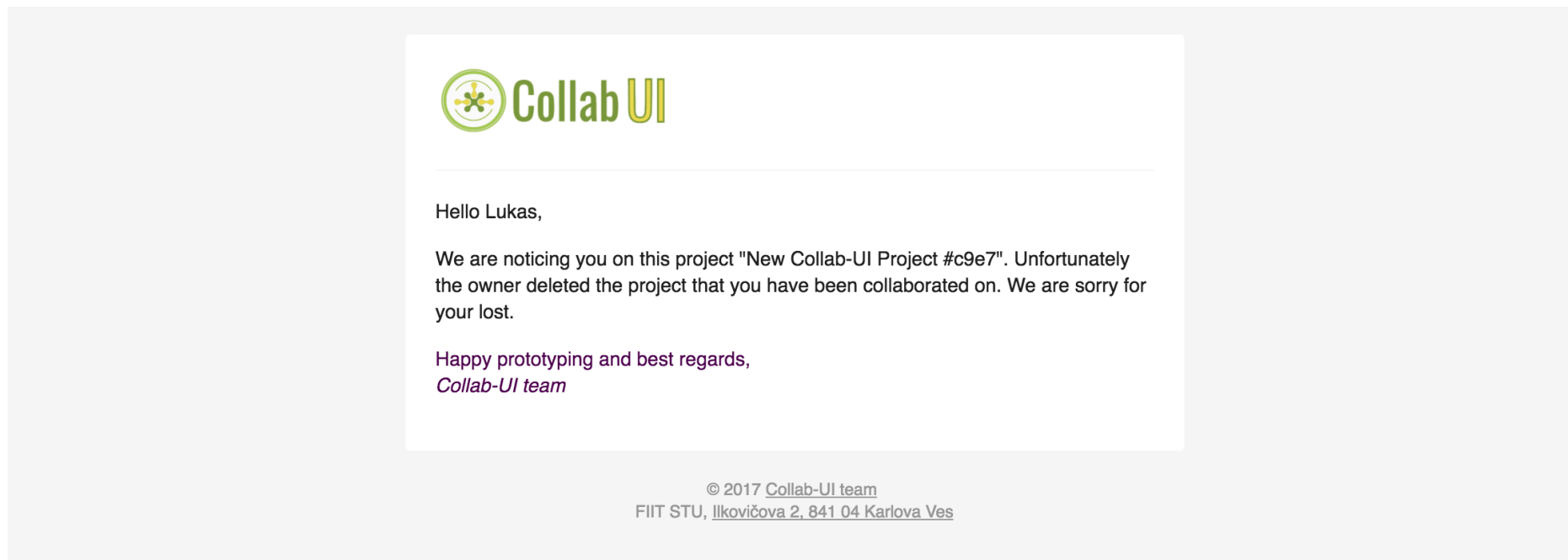
- Uložiť
- Zrušiť

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

Obrázok 18 - Obrazovka pre editovanie informácií 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 môcť prizvať kolaboranta prostredníctvom emailu, na ktorý mu príde oznámenie o jeho pozvaní do konkrétneho projektu a ďalej to už záleží na tom 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 odobrať 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

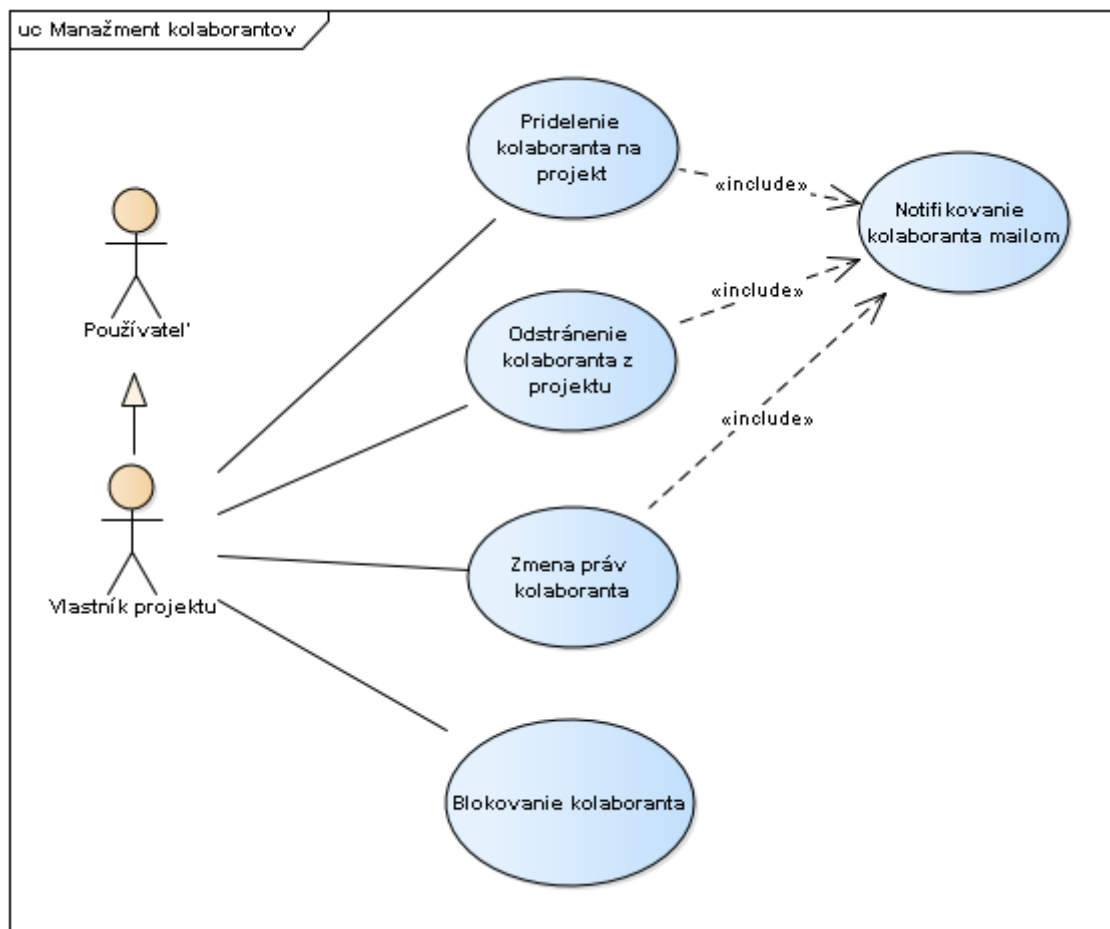
V rámci analýzy sme riešili hlavne akým štýlom bude daný kolaborant pridaný do projektu. Stále sme sa snažili dodržiavať nezložité grafické prevedenie s očakávanou funkcionalitou. Rozhodli sme sa preto rozdeliť kolaborantov a projekty do osobitných boxov. Tento fakt sme už spomínali aj pri vytváraní projektov. Tu ho spomíname znovu preto, lebo práve aj z časti kvôli vyriešeniu spôsobu pridávania kolaborantov sme učinili spomínané zmeny. V detaile projektu sa teda nachádza box pre projekt a vedľa neho ďalší box pre kolaborantov k danému projektu. Pri pridávaní kolaboranta by vlastníčkovi 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 zaimplementované 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. Prípád 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. Prípád použitia končí



## 5.3.4 Implementácia

### 5.3.4.1 Pridelenie kolaboranta na projekt

Pridelenie kolaboranta je funkcia, ktorú bude potrebovať 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.

#### 5.3.4.1.1 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

#### 5.3.4.1.2 Validácia

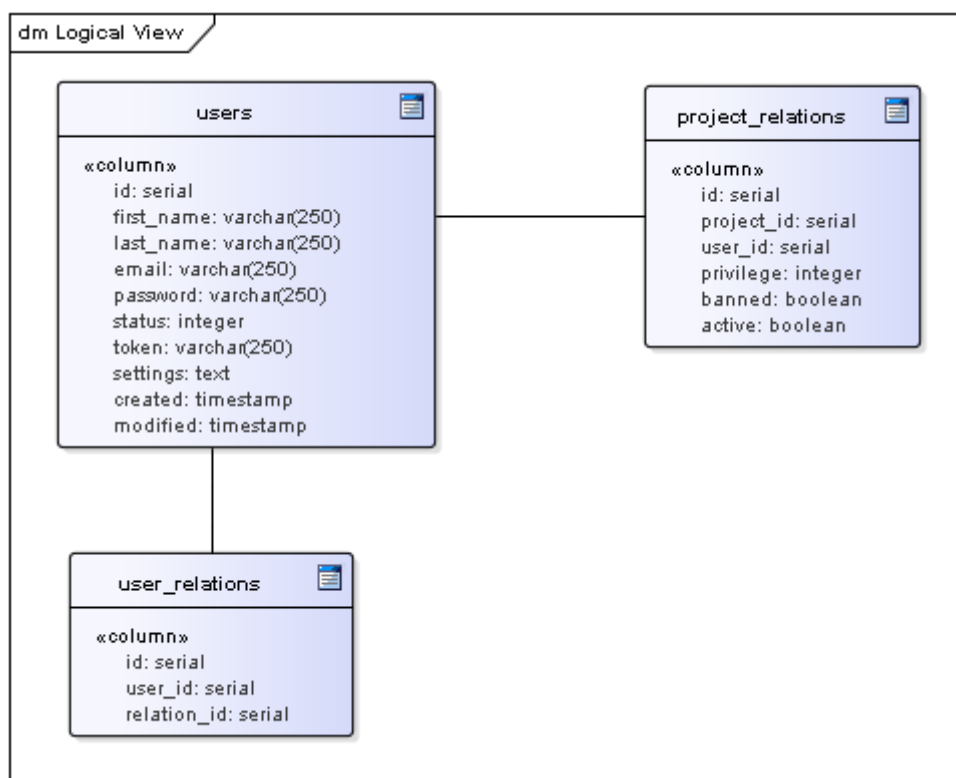
Celá validácia pridania nového kolaboranta do projektu prebieha na backende.

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 vytvorí nový používateľ. Takto vytvorený používateľ je špeciálneho typu "invited", pričom pri jeho vytvorení sa validuje len emailová adresa.

Na druhej strane, ak sa pri pridani 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 kontrol (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.

### 5.3.4.1.3 Databázový model

V rámci pridelenia kolaboranta na projekt sa využívajú nasledovné tabuľky.



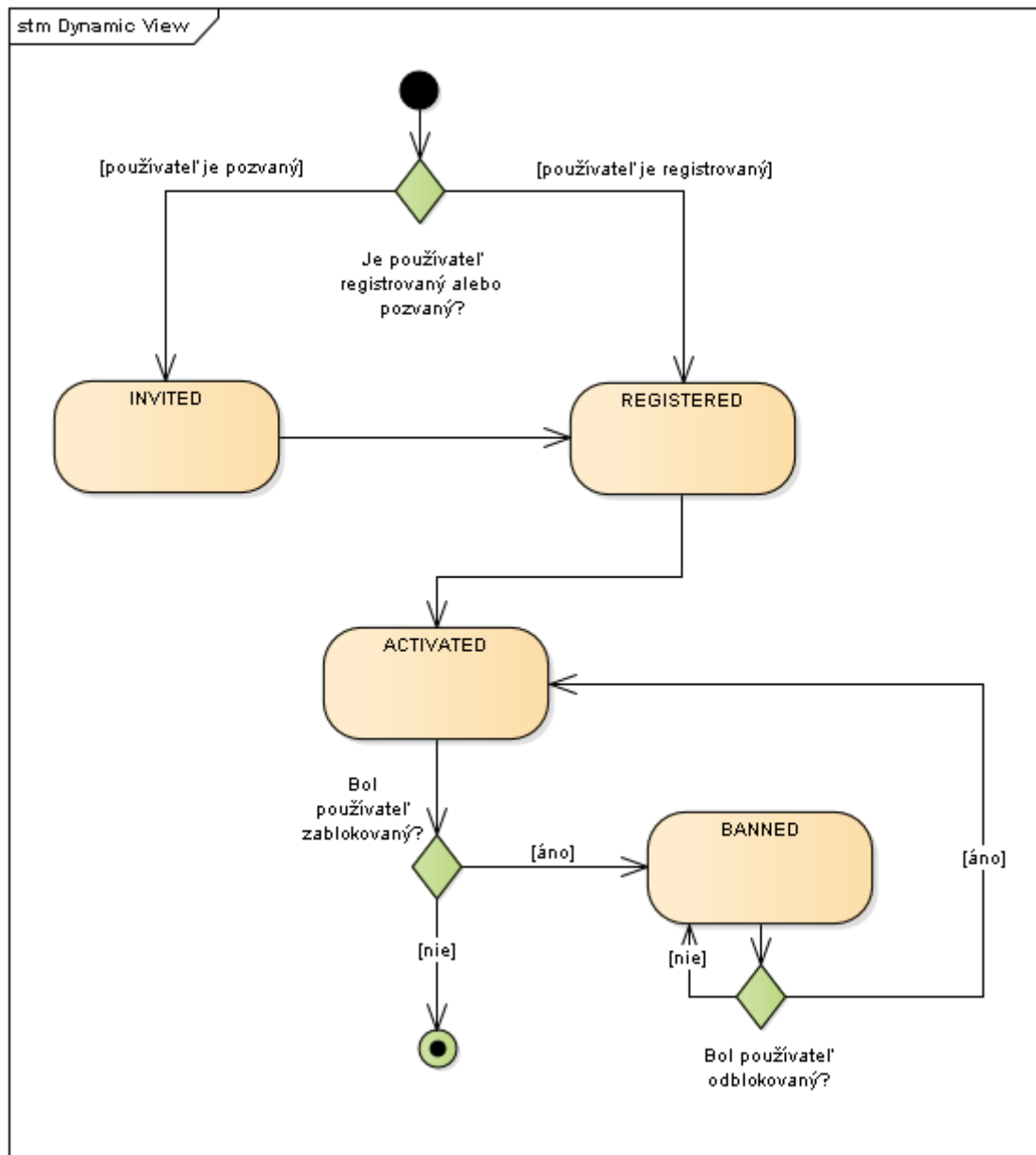
Obrázok 21 - DB Model pre pridelenie kolaboranta na projekt

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".




### 5.3.4.1.4 Stavový diagram pre status používateľa



Obrázok 22 - Stavový diagram pre status používateľa

### 5.3.4.1.5 Obrazovka pre pridelenie kolaboranta na projekt

## Collaborators 2 / 5

	<input type="text" value="aaa"/>	<span>Watch</span> <span>Edit</span>	<span>+ Add</span>
	<input type="text" value="aaab@azet.sk"/> Me	<span>Watch</span> <span>Edit</span>	
	aaa@azet.sk	<span>Watch</span> <span>Edit</span>	<span>⊘</span> <span>✕</span>

Obrázok 23 - Obrazovka pre pridelenie kolaboranta

### **5.3.4.2 Zmena práv kolaboranta**

Zmenu práv môže vykonávať 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.

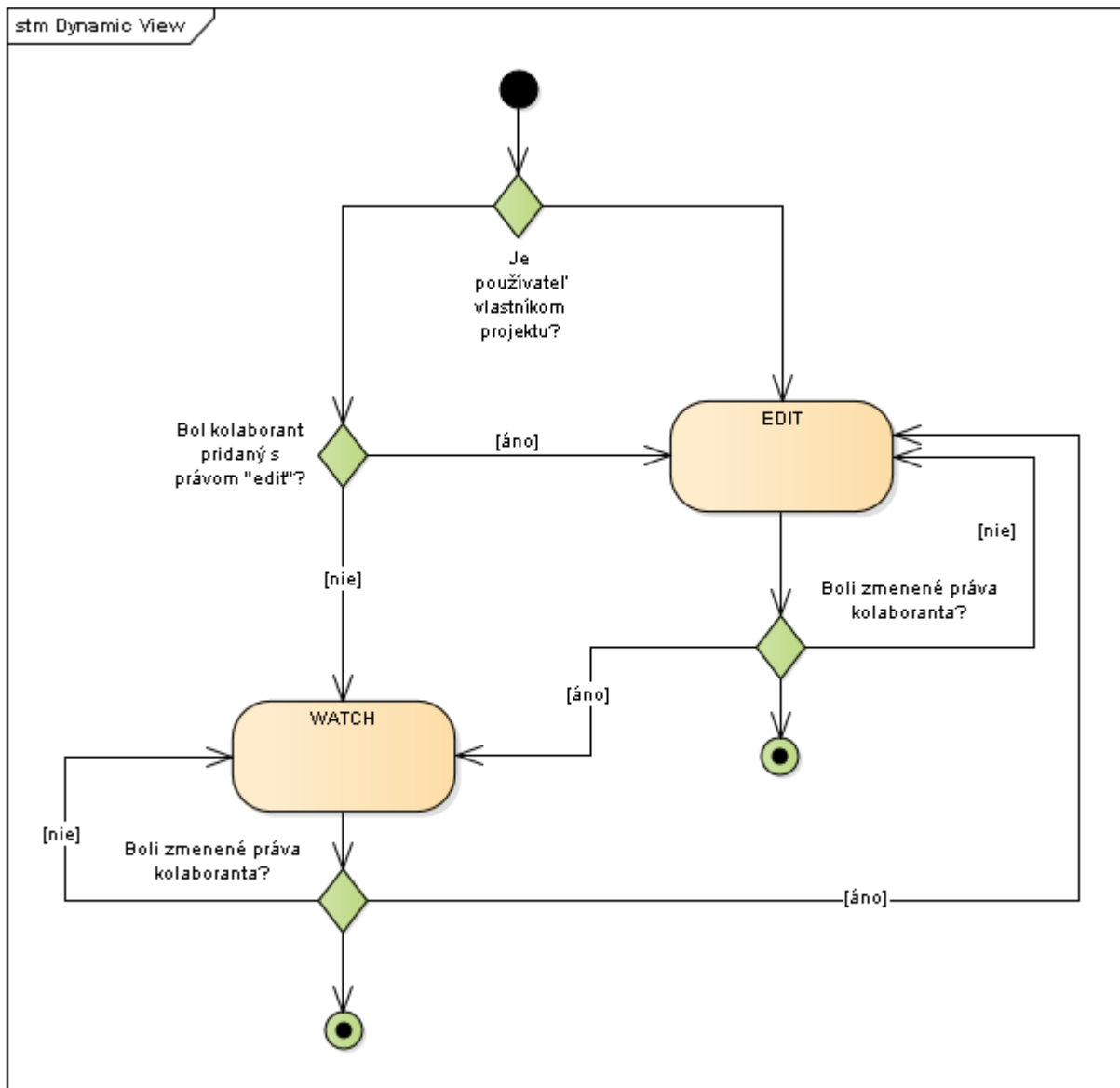
#### **5.3.4.2.1 Akceptačné kritéria**

- 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

#### **5.3.4.2.2 Validácia**

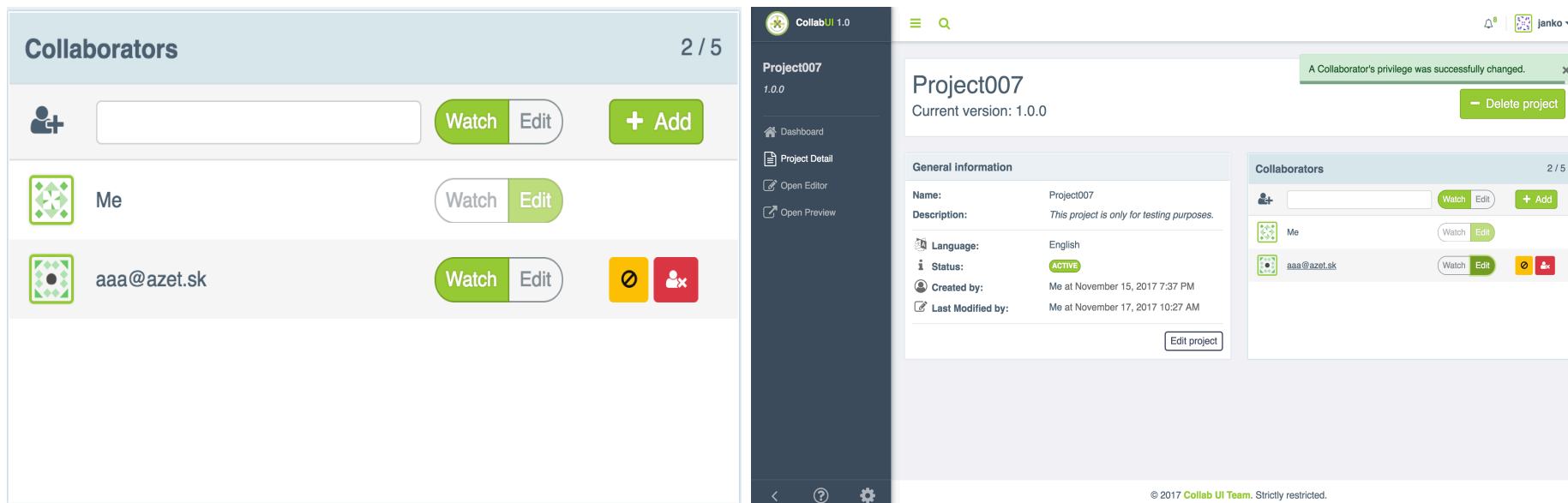
Jediná validácia, ktorá sa vykonáva pri tejto akcii je kontrola či je používateľ, ktorý chce zmeniť práva kolaboranta, naozaj aj vlastník projektu a teda či má patričné práva pre zmenu jeho práv.

### 5.3.4.2.3 Stavový diagram pre zmenu používateľských práv



Obrázok 24 - Stavový diagram pre zmenu používateľských práv

### 5.3.4.2.4 Obrazovka zmeny práv používateľa



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

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

Obrázok 25 - Obrazovka zmeny práv používateľa

### **5.3.4.3 Odstránenie kolaboranta z projektu**

Vlastník projektu má možnosť odstraňovať 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.

#### **5.3.4.3.1 Akceptačné kritéria**

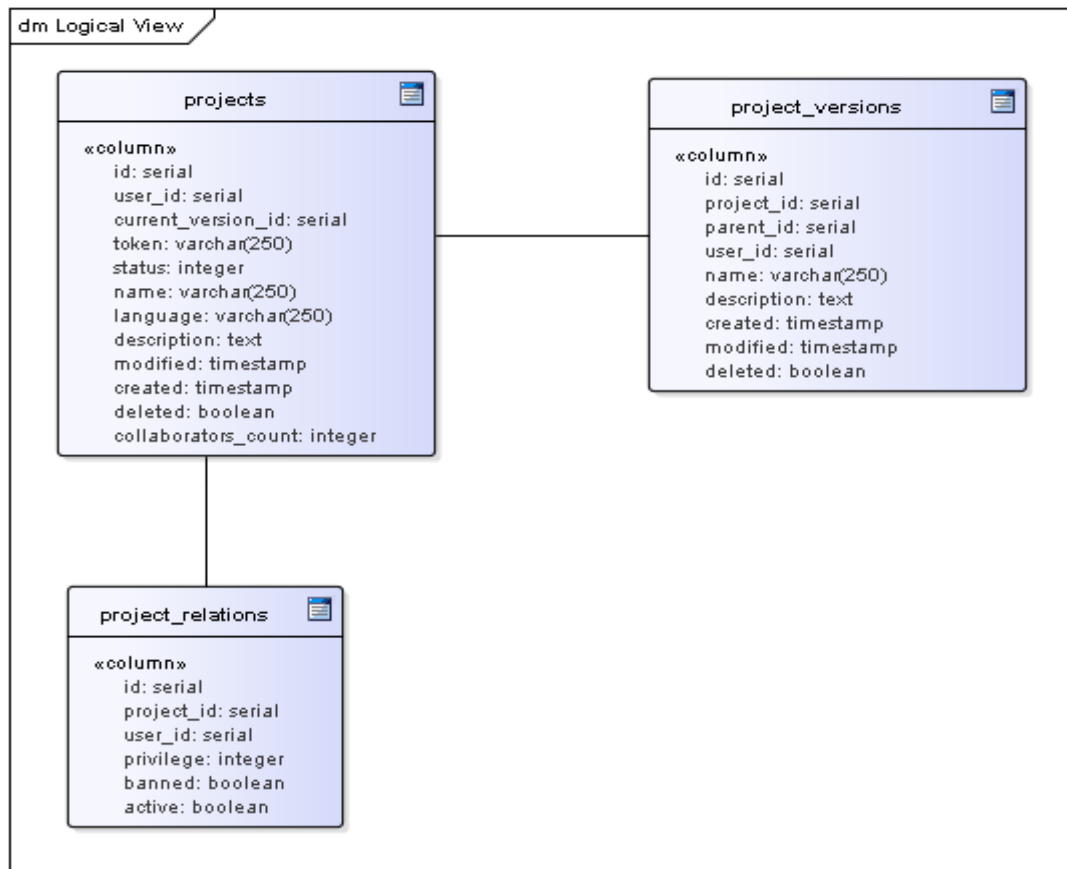
- Možnosť odstránenia kolaboranta pre vlastníka projektu
- Iba vlastník projektu môže odstraňovať kolaborantov
- Poskytnutie modálneho okna pre potvrdenie akcie vlastníka projektu
- Zníženie počtu kolaborantov pri potvrdení akcie, o konkrétneho používateľa

#### **5.3.4.3.2 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.



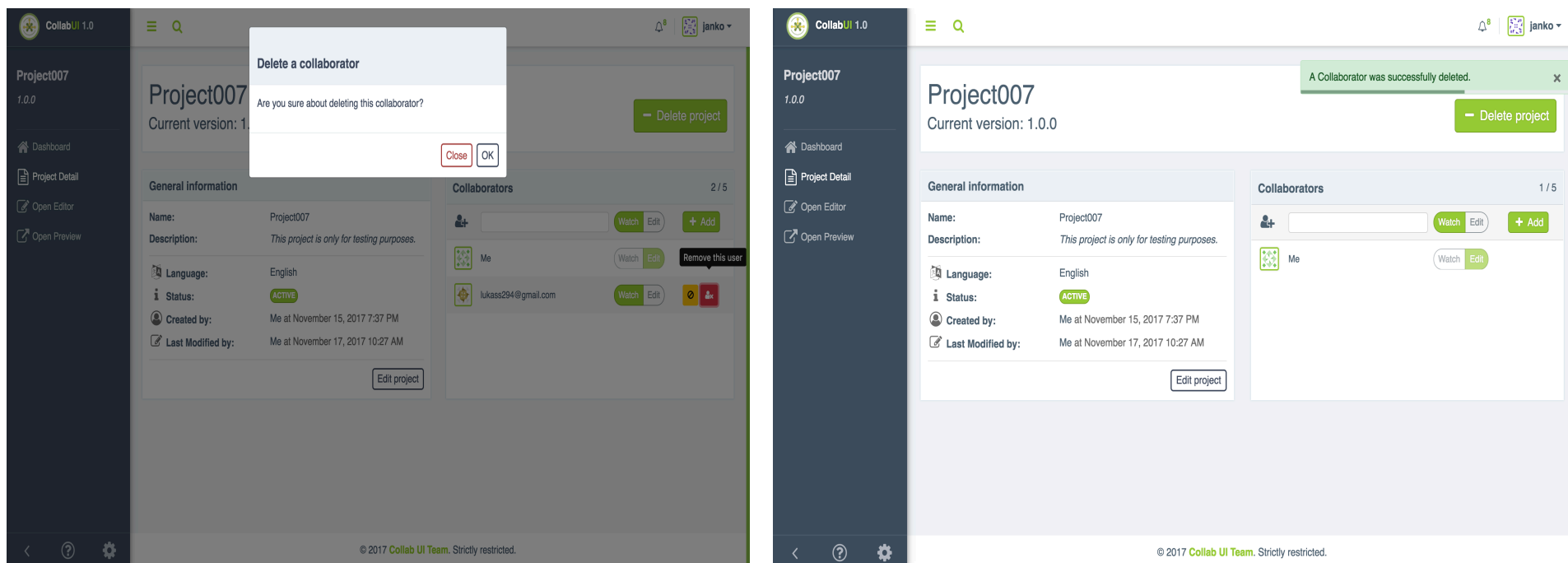
### 5.3.4.3.3 Databázový model



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 zničí počet **collaborators\_count**.

### 5.3.4.3.4 Obrazovka pre odstránenie kolaboranta z projektu



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

- Potvrdenie akcie
- Zrušenie akcie

Samozrejme po potvrdení akcie by mal byť vlastník projektu patrične notifikovaný o tom čo spravil. V tomto prípade sú to znova naše obľúbené notifikácie.

Obrázok 27 - Obrazovka pre odstránenie kolaboranta z projektu

#### 5.3.4.4 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.

##### 5.3.4.4.1 Akceptačné kritéria

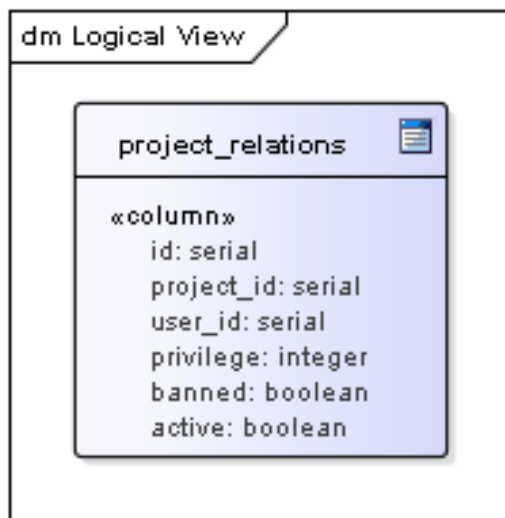
- Blokovanie môže byť vykonávané iba vlastníkom projektu
- Blokovaný kolaborant nesmie byť notifikovaný o jeho aktuálnej zmene práv
- Kolaborant nemá ďalej sprístupnený projekt, na ktorom pracoval

##### 5.3.4.4.2 Validácia

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

##### 5.3.4.4.3 Databázový model

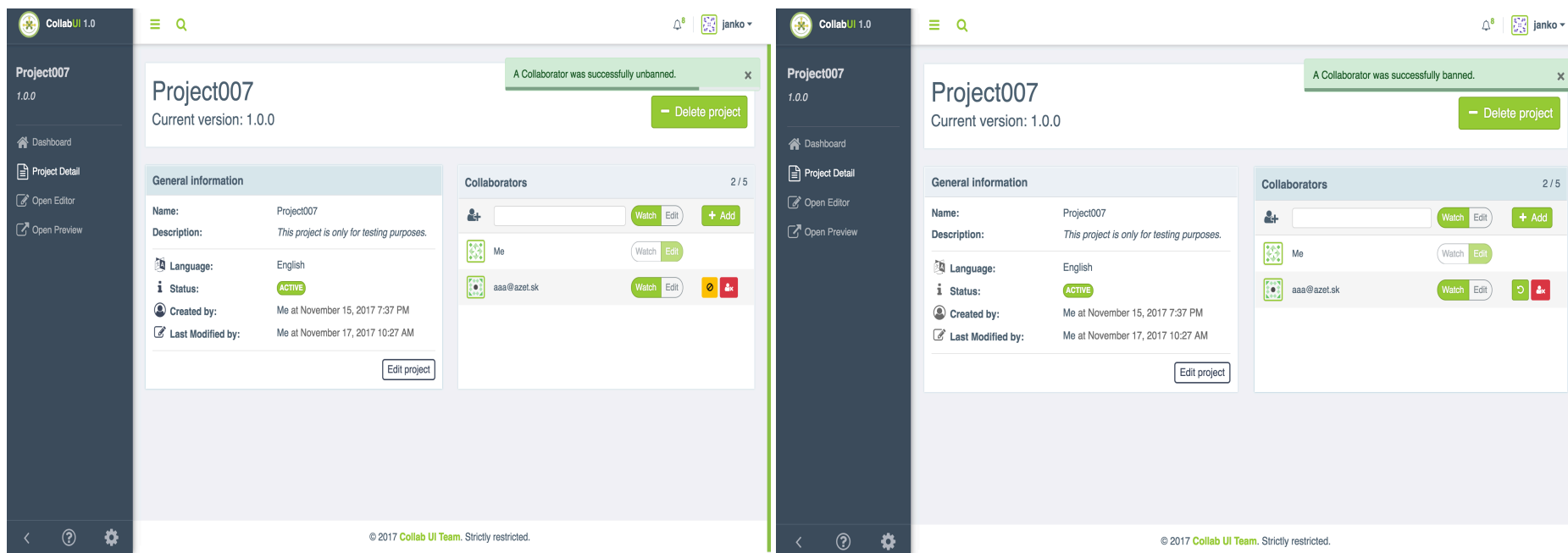
Pri tejto akcii sa využíva iba jediná tabuľka a tou je `project_relations`.



Obrázok 28 - Tabuľka `project_relations` pre blokovanie kolaboranta

Pri zablokovaní kolaboranta sa nastaví flag "banned" na true.

#### 5.3.4.4 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

#### **5.3.4.5 Notifikovanie kolaboranta mailom**

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

- Blokovanie kolaboranta

Pre notifikovanie kolaborantov používame emailové šablóny, prostredníctvom ktorých vieme poselať 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 ctg súboroch. Taktiež sa používa technológia Premailer.php, ktorá operuje na backende aplikácie.

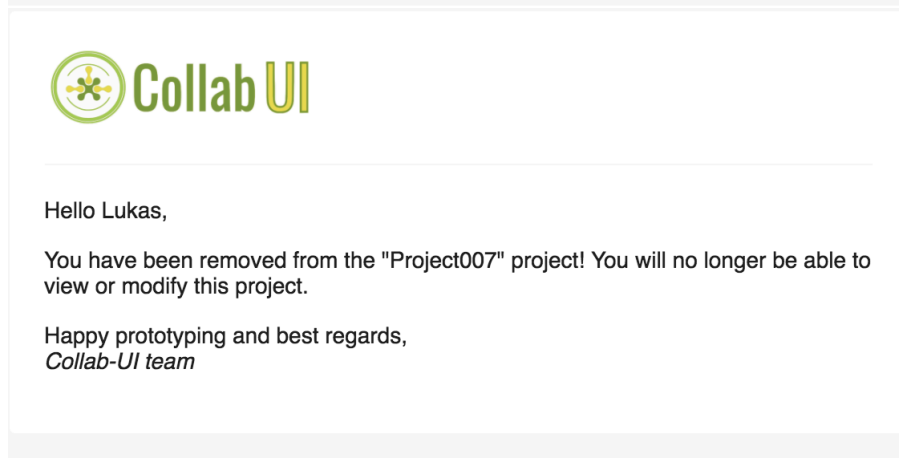
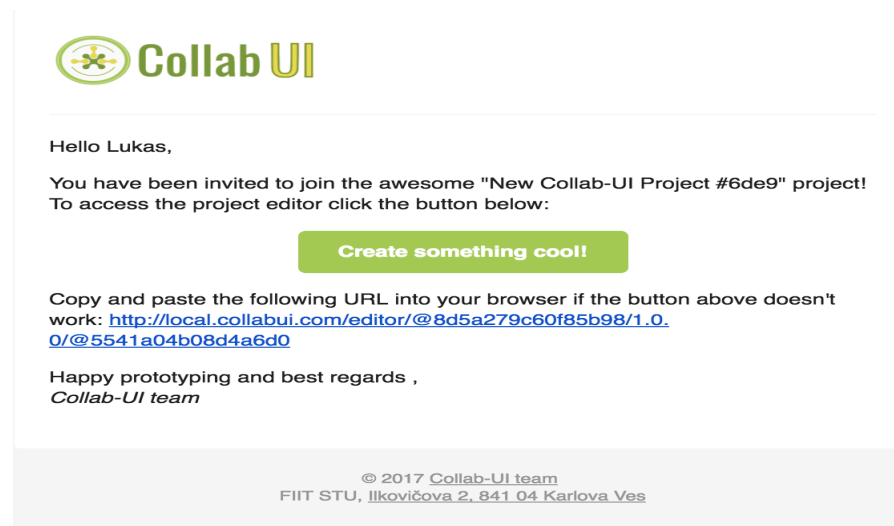
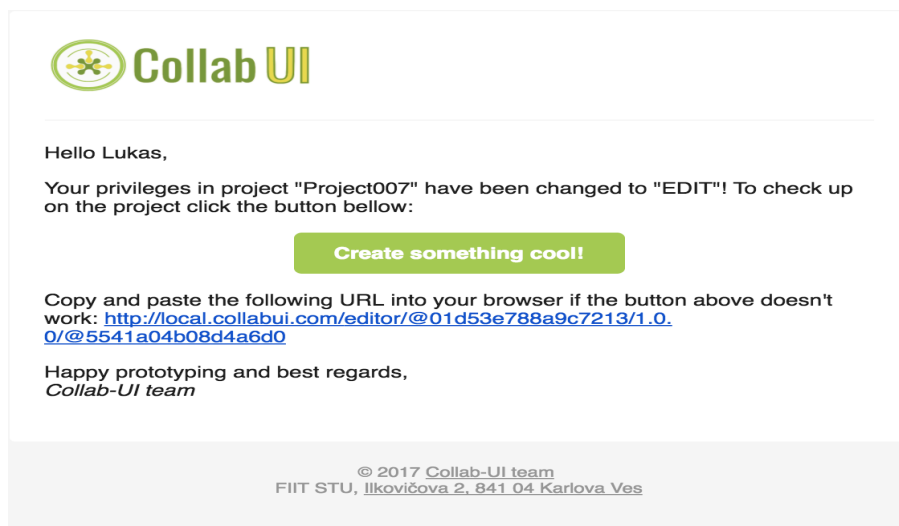
##### **5.3.4.5.1 Akceptačné kritéria**

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

##### **5.3.4.5.2 Validácia**

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

### 5.3.4.5.3 Obrazovky pre notifikovanie kolaboranta



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

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

### 5.3.5 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 scenáre:

- 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 Manažment tagov

### 5.4.1 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.

### 5.4.2 Analýza

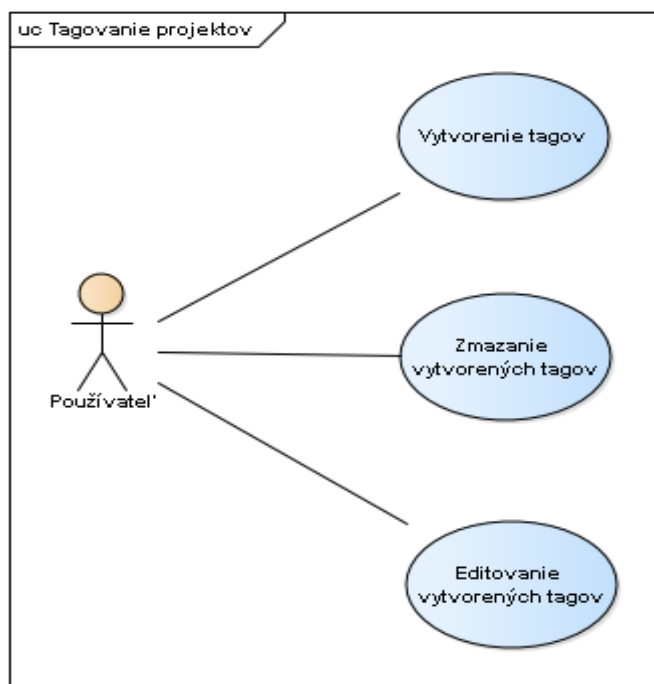
V rámci analýzy sme dospeli k záveru, že kvôli narastajúcemu počtu vytvorených projektov je potrebné dopracovať zefektívnenie vyhľadávania, ktorej realizáciu sme zamýšľali príliš jednoduchú. Rozhodli sme sa teda riešenie postaviť na otagovaní projektov.

### 5.4.3 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 32 - 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

## **5.4.4 Implementácia**

### **5.4.4.1 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
- Pridení projektu – assigned, edit alebo watch

#### **5.4.4.1.1 Akceptačné kritéria**

- Po vytvorení, resp. pridení 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

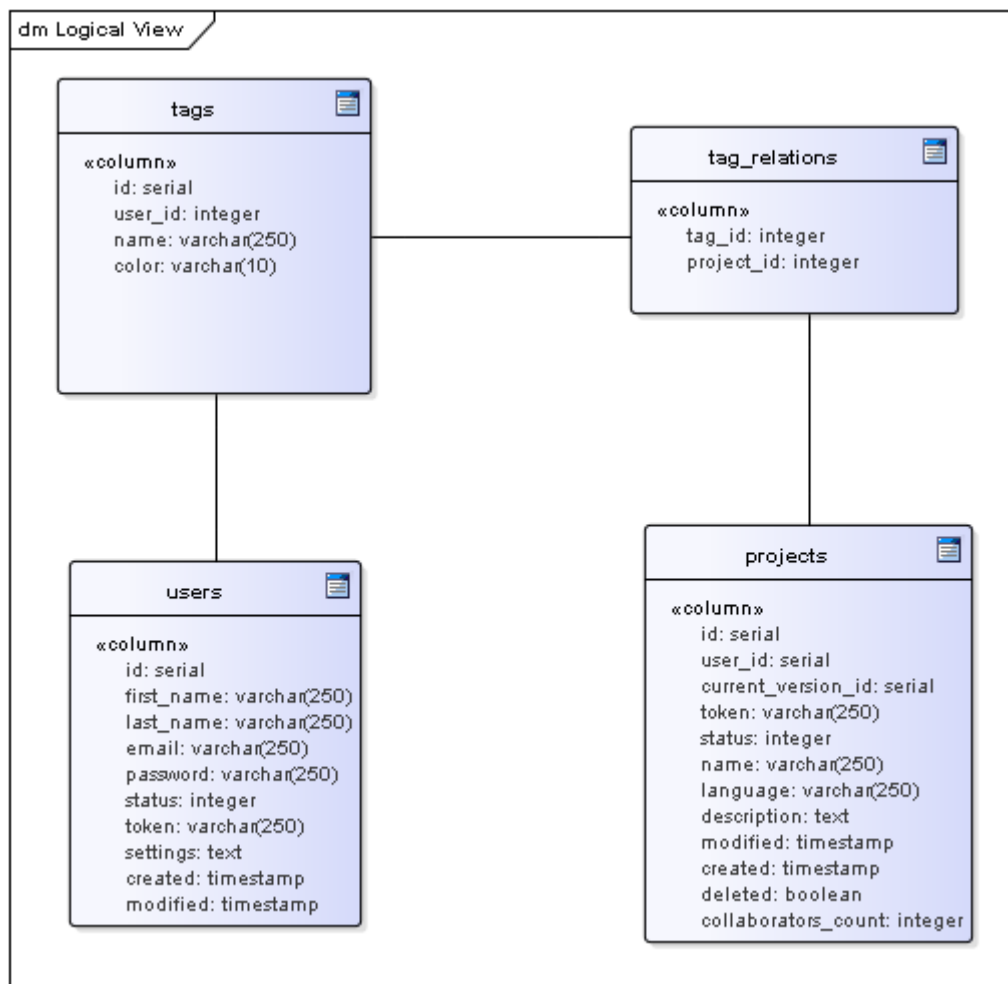
#### 5.4.4.1.2 Validácia

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

#### 5.4.4.1.3 Databázový model



Obrázok 33 - Databázový model pre manažment tagov

#### **5.4.5 Testovanie**

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

## 5.5 Editor

### 5.5.1 Úvod

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

- Editor bez kolaborácie
- 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 štylovať 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 iníciaľne oboznámenie sa s nástrojom. Tu používateľ môže vidieť akým štýlom môžu byť rôzne komponenty kombinované a taktiež ako ich meniť. Ďalšou obšírnu 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 prispôbovať ich veľkosť podľa vlastných potrieb. Ďalej sme implementovali obmedzenia pre role „watch“ a role „banned“.

Čo sa týka editora s implementáciou kolaborácie sme dorábali jednotlivé časti akými bolo preposielanie dát kolaborantov, synchronizácia zmien medzi kolaborantami a mnoho iného. Detailne nižšie.

## 5.5.2 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 funkcií 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 zavrhlí podporu mobilných zariadení v rámci editora a vytvorili novú chybovú stránku.

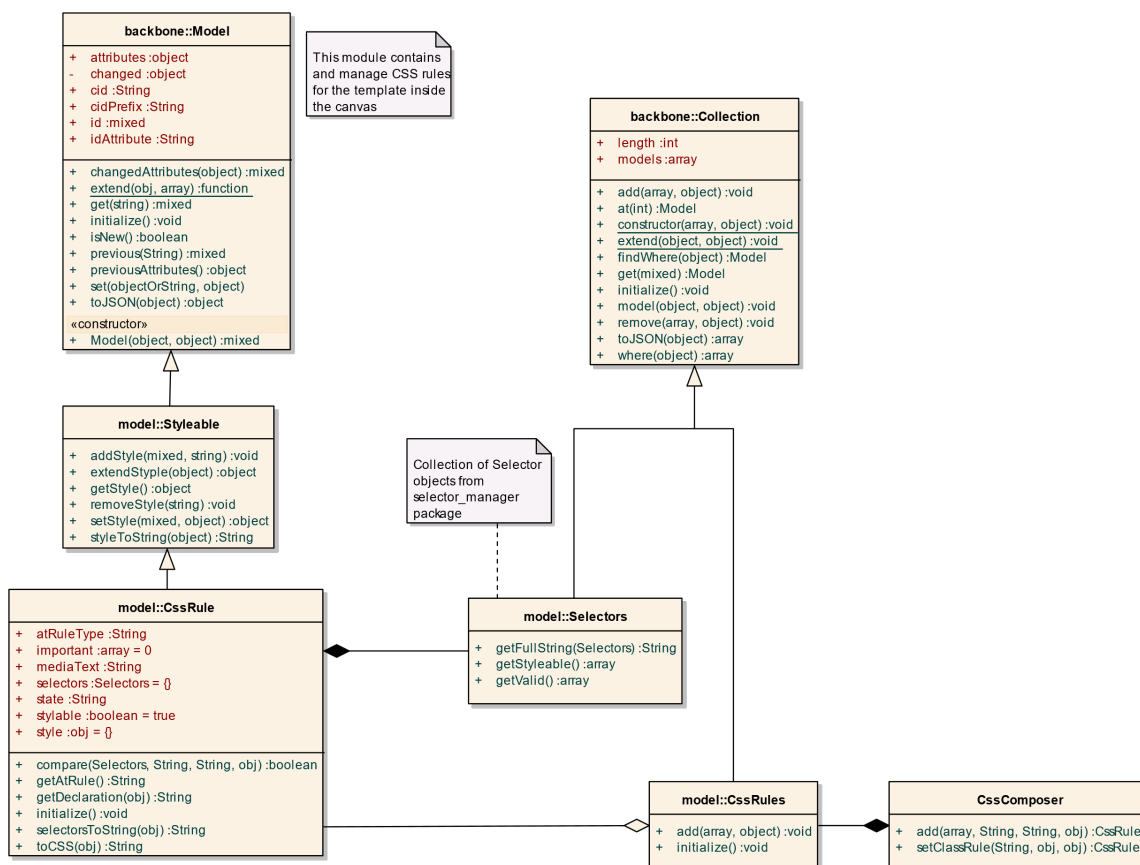
Analýza riešenia kolaborácie v reálnom čase vyžadovala hlbšie ponorenie sa do technickej štruktúry podporného nástroja pre tvorbu stránok, ktorej opis sa nachádza v nasledujúcich častiach.

### 5.5.3 Analýza podporného nástroja pre tvorbu stránok

K pochopeniu vnútorného fungovania podporného nástroja pre tvorbu stránok<sup>5</sup> bolo nutné zhotoviť UML diagramy. Znalosti, ktoré sme z diagramov nadobudli využívame pri implementácii kolaboratívnej časti projektu, nakoľko samotná kolaborácia si vyžaduje komplexnejšie úpravy GrapesJS a preto pochopenie vnútorného fungovania niektorých častí bolo nevyhnutnou podmienkou pre ich správnu implementáciu. Sústredili sme sa na označovanie komponentov, CSS tried komponentov, zmeny textového komponentu a pridanie nového komponentu.

#### 5.5.3.1 Diagram tried, CSS composer

Diagram tried CSS ovládača, ktorý zachytáva vzťahy medzi triedami Backbone.js a GrapesJS. BackboneJS je framework určený na štruktúrovanie JavaScript kódu do MVC modelu, ktorý GrapesJS využíva na reprezentáciu.

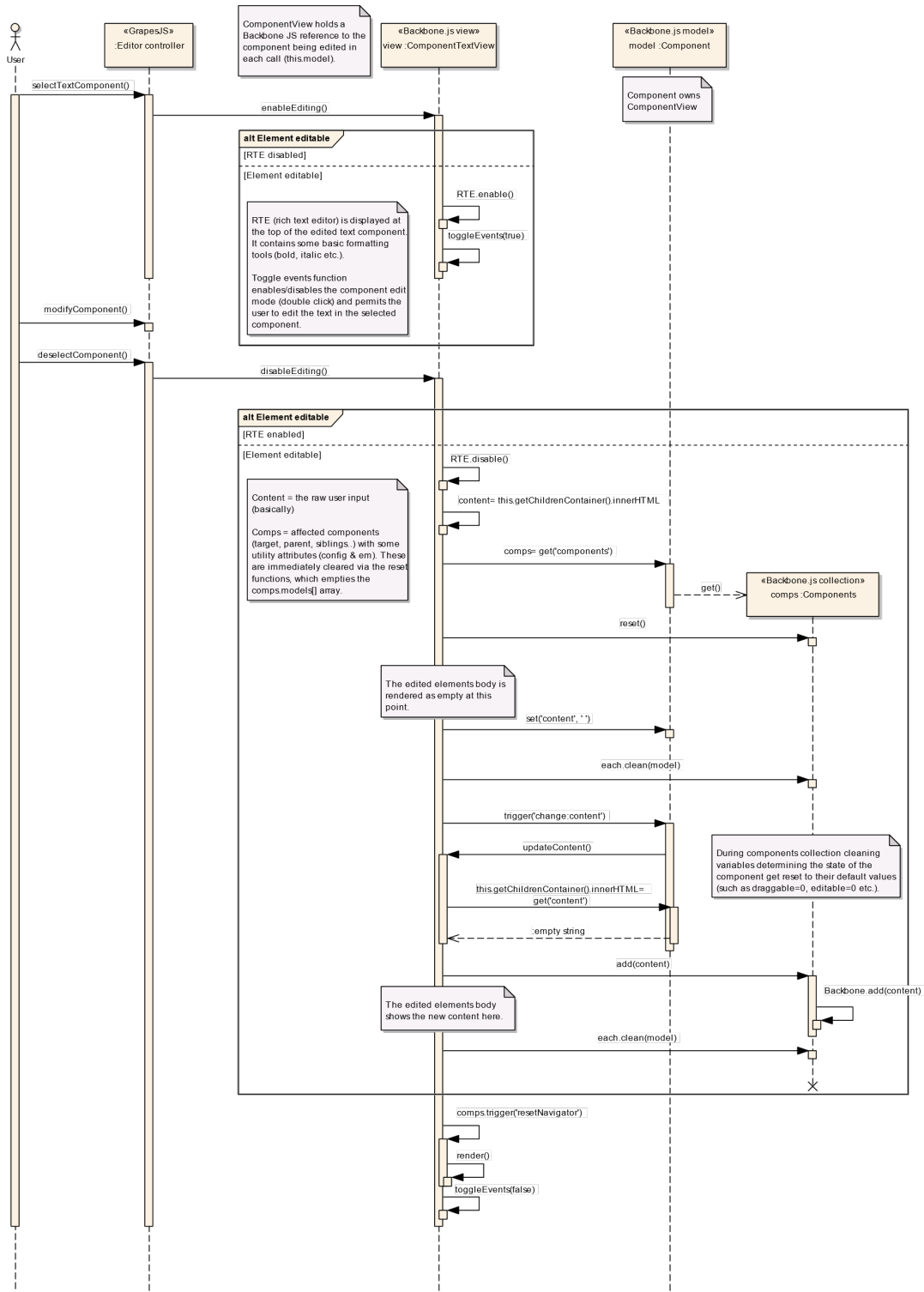


Obrázok 34 - Diagram tried pre CSS Composer

<sup>5</sup> <http://grapesjs.com/>

### 5.5.3.2 Sekvenčný diagram, zmena textového komponentu

Sekvenčný diagram, ktorý zachytáva tok udalostí pri zmene textového komponentu.

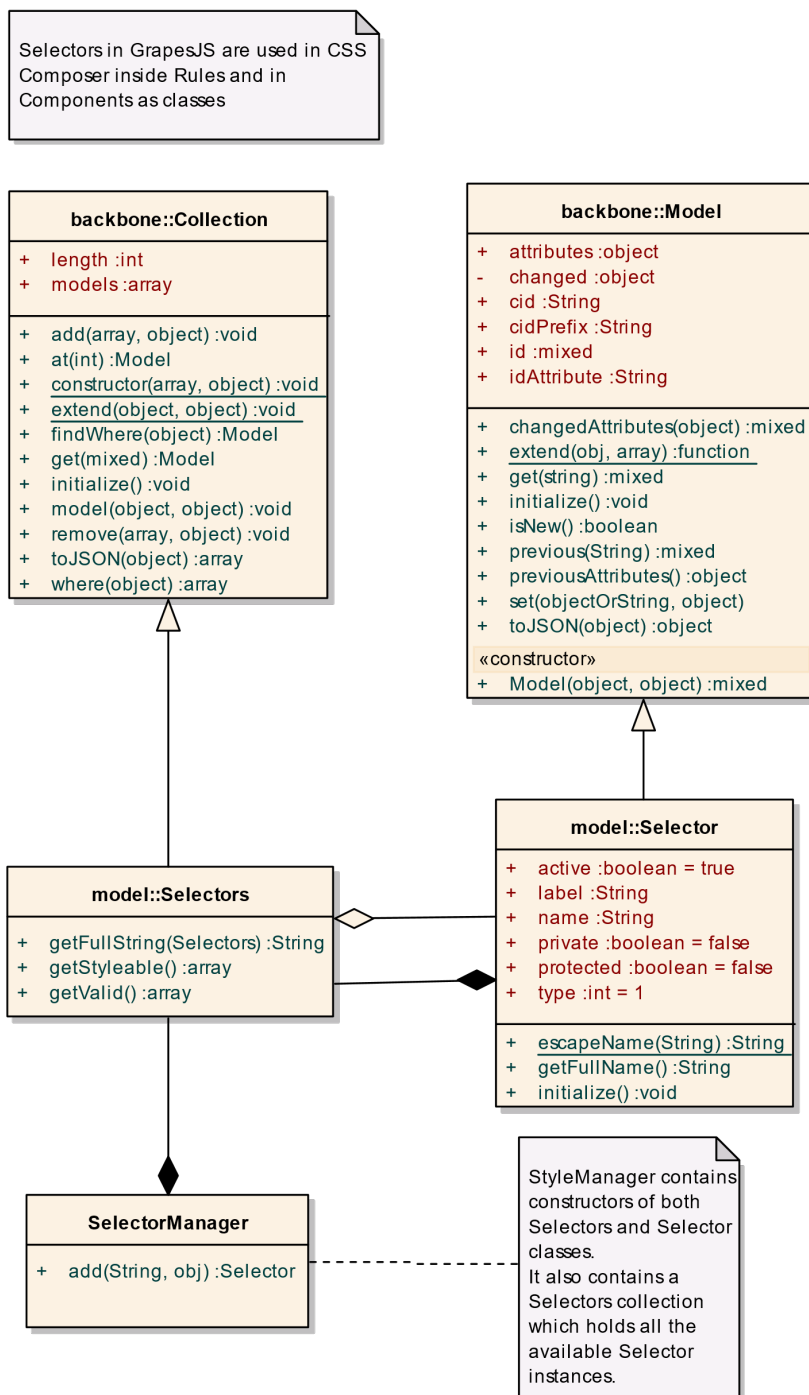


Obrázok 35 - Sekvenčný diagram pre zmenu textového komponentu



### 5.5.3.3 Diagram tried, manažér selektorov

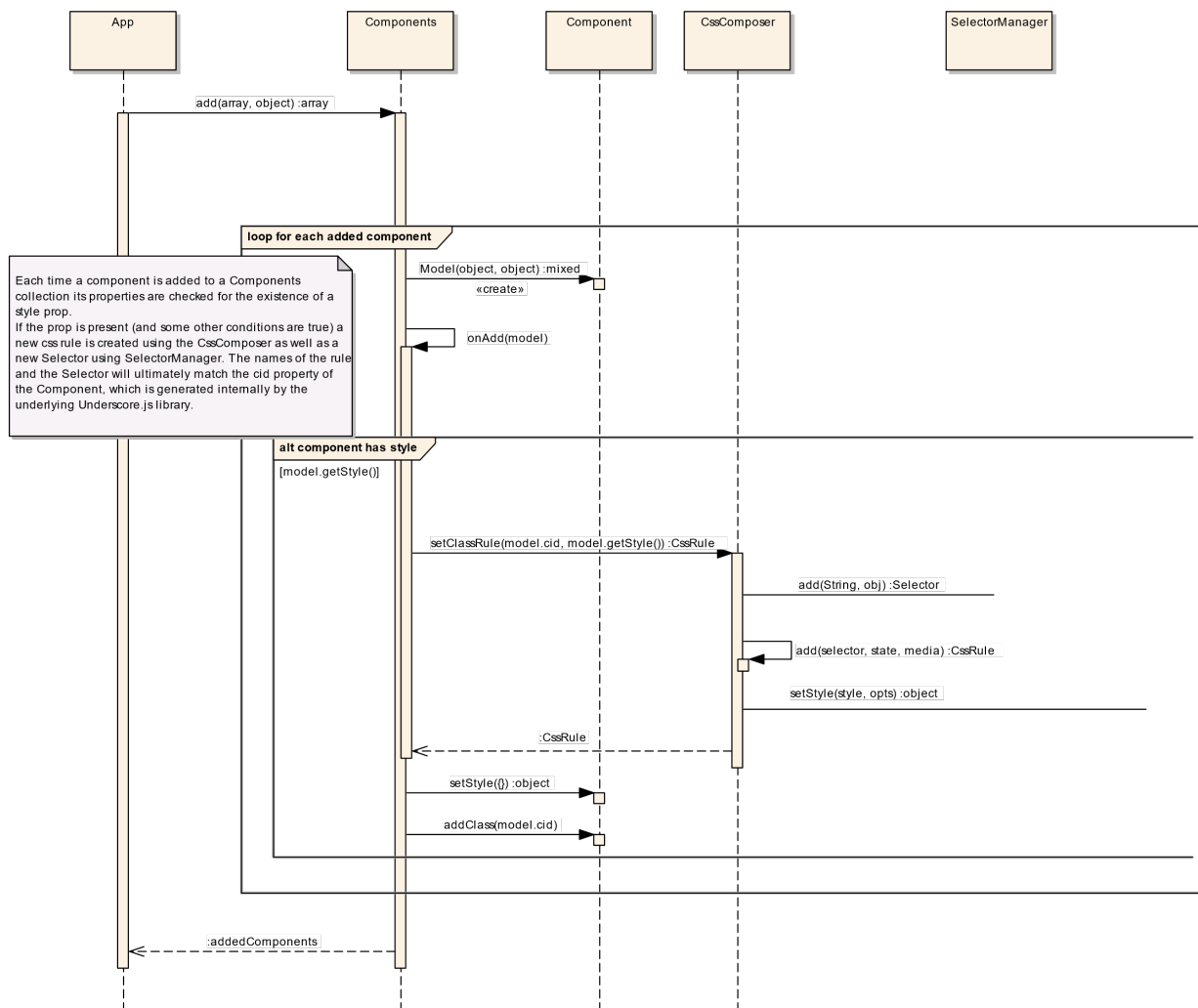
Diagram tried selektorov komponentov, ktorý zachytáva vzťah Backbone.js a GrapesJS. Rovnako ako pri CSS composer aj tu GrapesJs využíva Backbone.js na prácu s modelmi.



Obrázok 36 - Diagram tried pre manažér selektorov

### 5.5.3.4 Sekvenčný diagram, pridanie komponentu do kolekcie komponentov

Sekvenčný diagram, ktorý zachytáva tok udalostí pri pridaní komponentu do kolekcie komponentov.



Obrázok 37 - Diagram tried pre pridanie komponentu do kolekcie komponentov

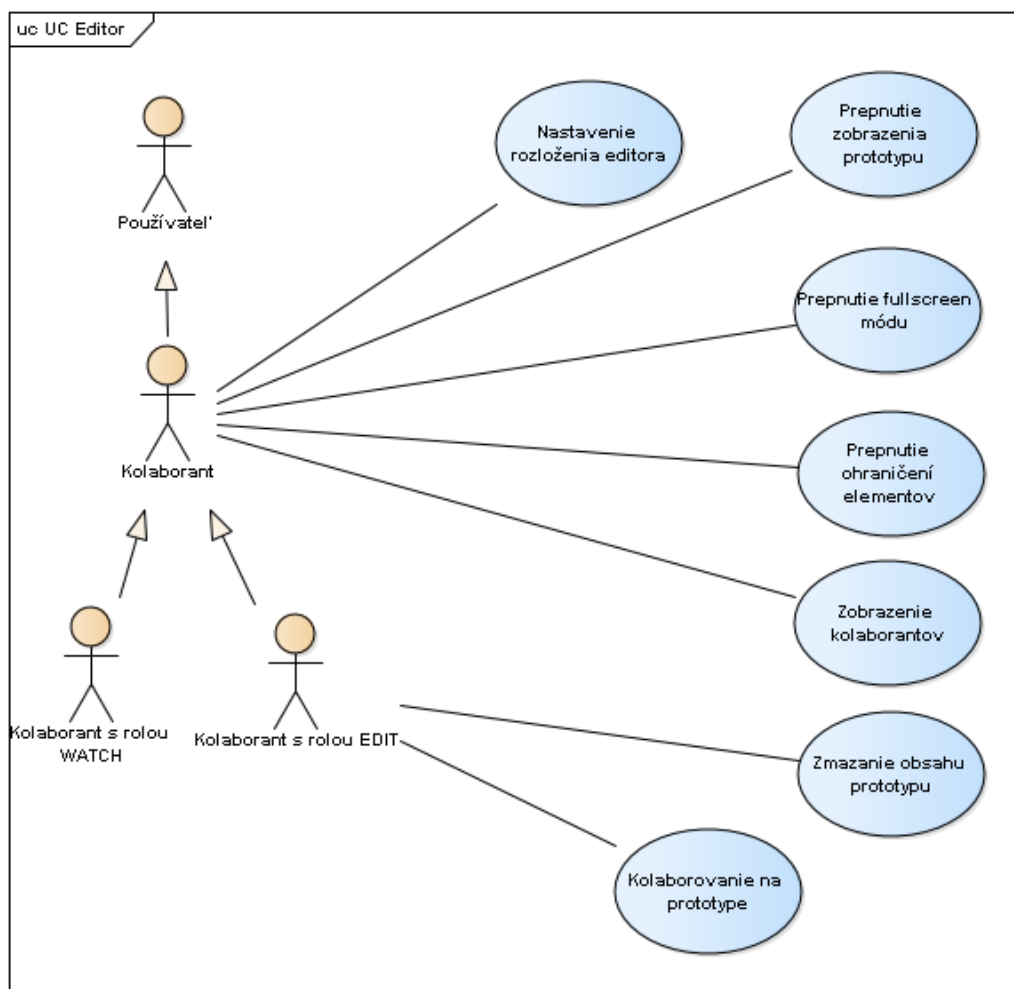
Diagramy, ktoré sme analyzovali nám v niektorých častiach pomohli k úspešnej implementácii. Treba však zdôrazniť, že samotné zakreslenie a analyzovanie diagramov je časovo veľmi náročné a preto je najlepšie zachytiť diagramom len tie najproblematickejšie časti. V rámci projektu sme chceli zakresliť ešte viacero diagramov, ktoré by nám v konečnom dôsledku mohli zrýchliť čas vývoja alebo pomôcť optimalizovať niektoré existujúce časti avšak vyššiu prioritu sme priradili kompletizácii celkovej implementácie projektu. Diagramy, ktoré odporúčame zakresliť a analyzovať do budúcnosti sú diagramy týkajúce sa zmien pozícií komponentov.

## 5.5.4 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 prototypu

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



Obrázok 38 - 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 prihlásených používateľov o príchode nového kolaboranta
3. Systém zobrazí kolaboranta v panely určenom pre evidenciu kolaborantov
4. Prípad použitia končí

## **UC Kolaborovanie na prototype**

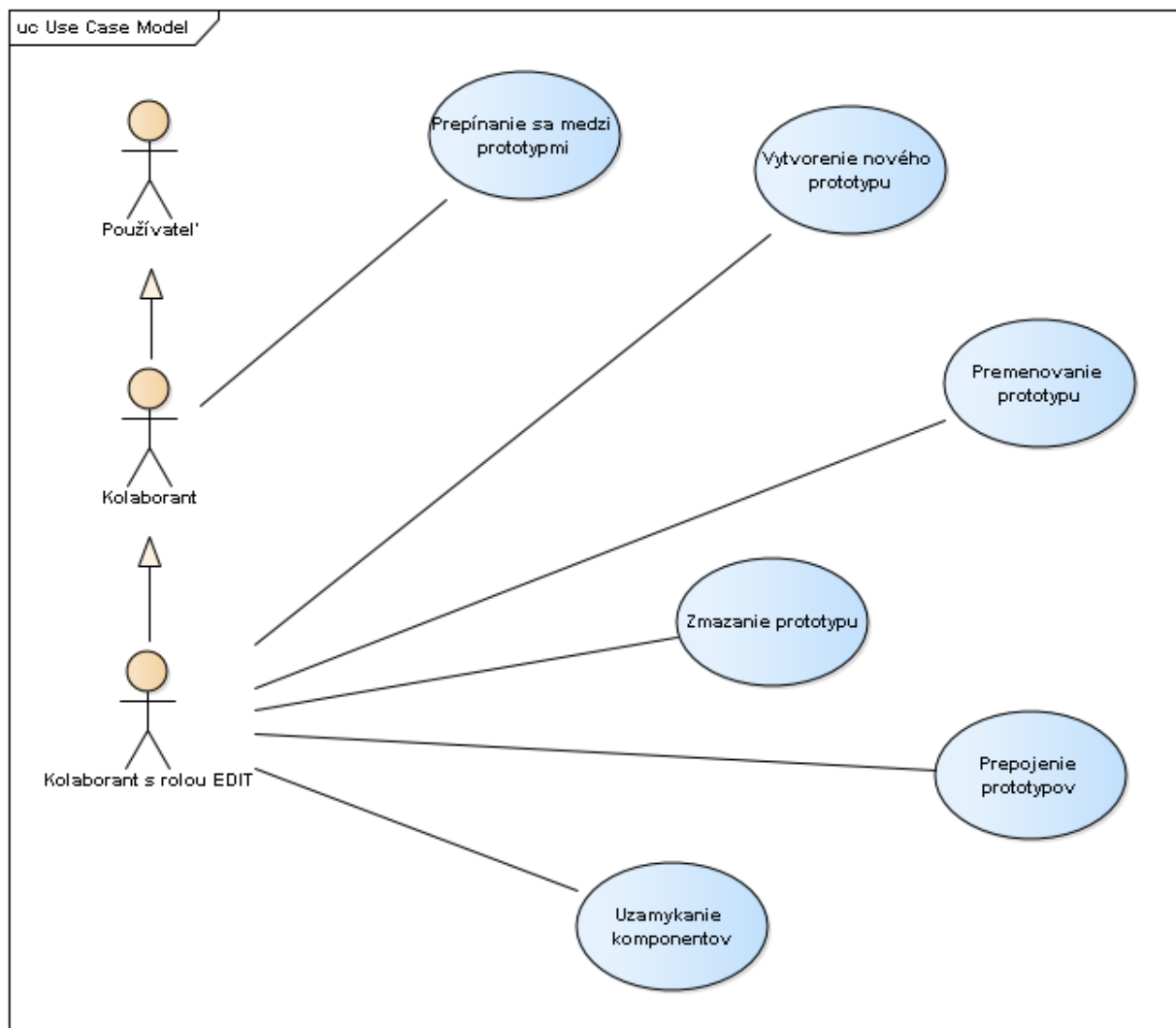
### **Hlavný tok:**

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

### 5.5.4.1 Rozšírenie návrhu editora

Samotné rozšírenie návrhu editora sa skladá z nasledujúcich UC:

- Vytvorenie nového prototypu
- Premenovanie prototypu
- Zmazanie prototypu
- Prepínanie sa medzi prototypmi
- Prepojenie prototypov
- Uzamykanie komponentov



Obrázok 39 - UC diagram pre rozšírenie návrhu editora

## **UC Vytvorenie nového prototypu**

### **Hlavný tok:**

1. Kolaborant vyberie možnosť "New" v editore
2. Systém vytvorí nový prototyp, ďalej stránku a prepne kolaboranta na novo vytvorenú stránku
3. Systém notifikuje ostatných kolaborantov aby sa im vytvorila rovnaká stránka
4. Prípad použitia končí

## **UC Zmazanie prototypu**

### **Hlavný tok:**

1. Kolaborant vyberie možnosť odpadkového koša v paneli pre stránky
2. Systém zobrazí modálne okno s potvrdením o zmazanie
3. Kolaborant potvrdí akciu zmazania stránky
4. Systém vymaže stránku a notifikuje ostatných kolaborantov aby sa im stránka rovnako zmazala
5. Prípad použitia končí

### **Alternatívny tok:**

- 3.a1 Kolaborant zruší akciu vymazania stránky
- 3.a2 Systém nevykoná žiadnu akciu
- 3.a3 Pokračuje bodom 5



## **UC Premenovanie prototypu**

### **Hlavný tok:**

1. Kolaborant vyberie možnosť ceruzky v paneli pre stránky
2. Systém zobrazí modálne okno s potvrdením o premenovaní
3. Kolaborant potvrdí akciu premenovania stránky
4. Systém premenuje stránku a notifikuje ostatných kolaborantov aby sa im stránka rovnako premenovala
5. Prípád použitia končí

### **Alternatívny tok:**

- 3.a1 Kolaborant zruší akciu premenovania stránky
- 3.a2 Systém nevykoná žiadnu akciu
- 3.a3 Pokračuje bodom 5

## **UC Prepínanie medzi prototypmi**

### **Hlavný tok:**

1. Kolaborant zvolí jednu z existujúcich stránok v editore
2. Systém získa informácie o stránke
3. Systém spracuje a zobrazí stránku v korektnej podobe
4. Systém notifikuje ostatných kolaborantov o prepnutí kolaboranta na konkrétnu stránku
5. Prípád použitia končí

## **UC Prepojanie prototypov**

### **Hlavný tok:**

1. Kolaborant vyberie komponent "anchor" z panelu preddefinovaných komponentov a potiahne ho do editora
2. Kolaborant vyberie v panely konfigurácie komponentu jeden z vytvorených prototypov, ktorý chce s otvoreným prototypom prepojiť
3. Systém ch pripojených notifikuje ostatných pripojených kolaborantov o zmene
4. Prípad použitia končí

## **UC Uzamykanie komponentov**

### **Hlavný tok:**

1. Kolaborant vyberie možnosť "uzamknutie komponentu" (zámok) na danom komponente
2. Systém uzatvorí možnosť editácie komponentu pre ostatných kolaborantov a notifikuje ich o zmene
3. Systém indikuje uzamknuté komponenty v panely vrstiev
4. Prípad použitia končí

## 5.5.5 Implementácia

### 5.5.5.1 Editor bez kolaborácie

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 akcií
- Panel pre zoznam kolaborantov
- Panel pre chat
- Panel pre uzamknuté komponenty

Taktiež sme mysleli na potrebu používateľa prispôbiť 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 prispôbí 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.

Obmedzenia pre kolaborantov s právom WATCH sú nasledovné:

- Zmazanie obsahu prototypu
- Úprava, editovanie prototypu
- 4 vstavané panely knižnice
- Vytvorenie, premenovanie, zmazanie prototypu
- Uzamykanie komponentov

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

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.

#### **5.5.5.1.1 Akceptačné kritéria**

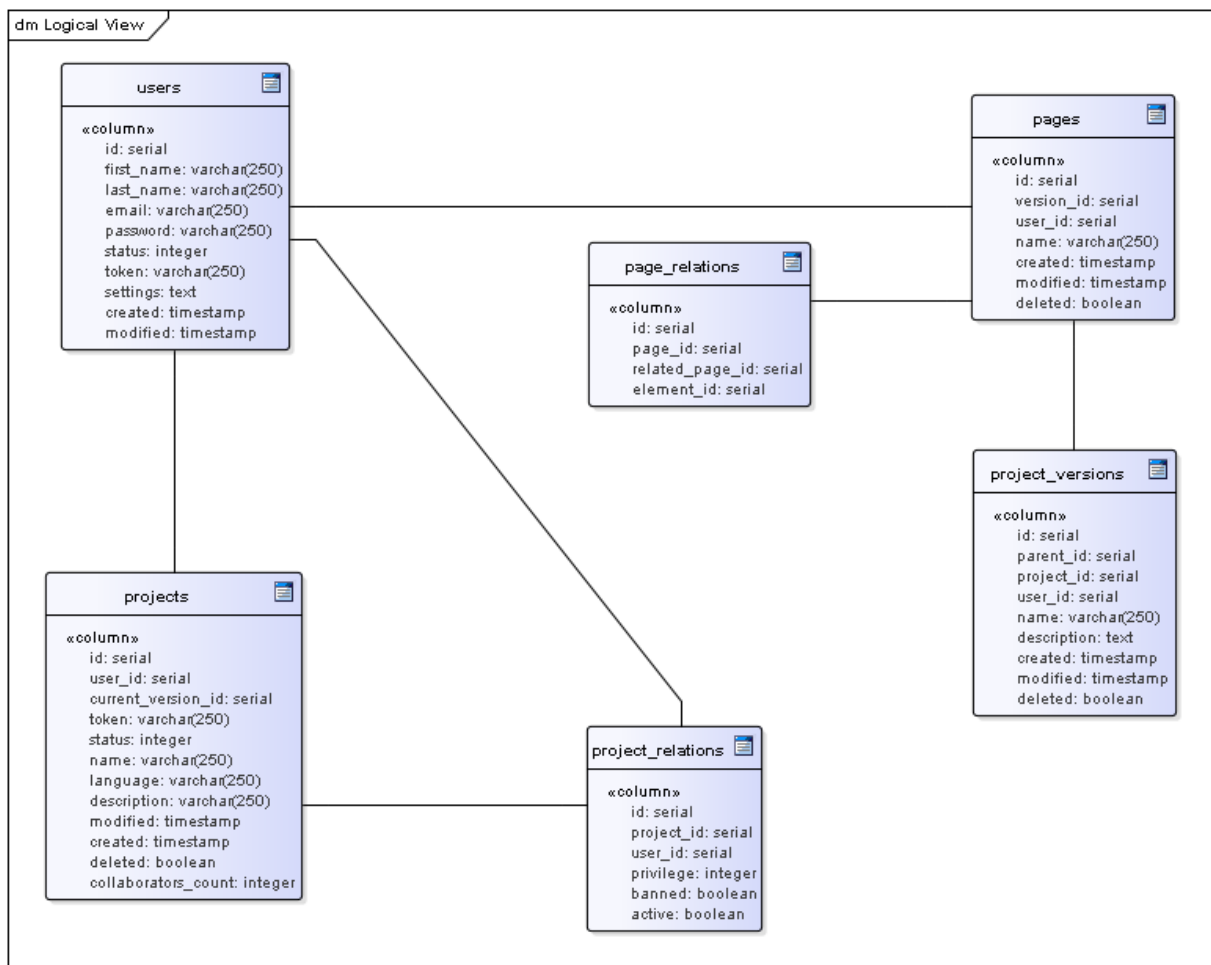
- Možnosť prihlásenia sa do editora a následná autorizácia
- Vyskladanie prototypu prostredníctvom editora
- Vytvorený prototyp je automaticky uložený po zmene
- Možnosť zmeny rozloženia panelov editora pre právo „EDIT“ a zvlášť pre právo „WATCH“
- Používateľ s právom WATCH, nedokáže editovať prototype alebo vykonať akúkoľvek zmenu, ktorú by systém zachytil a uložil
- Používateľ s právom BANNED je presmerovaný na chybovú stránku

#### **5.5.5.1.2 Validácia**

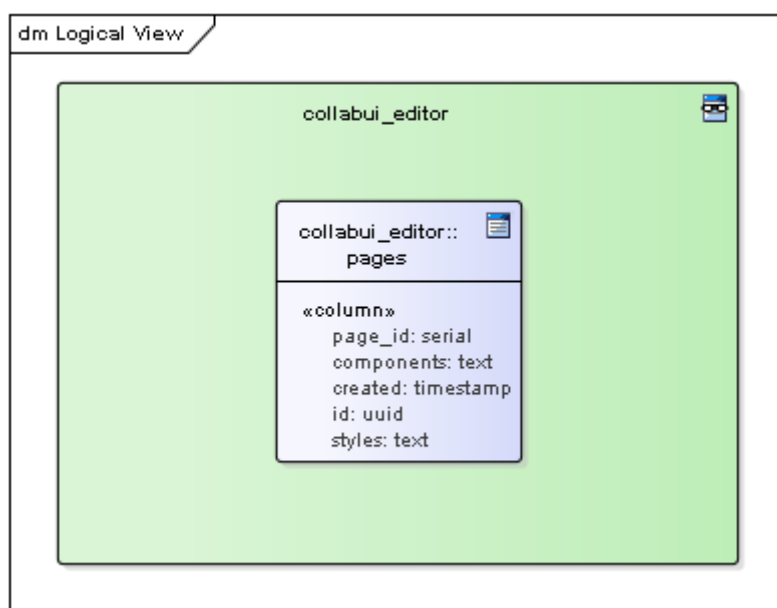
Pri editore je dôležité hlavne validovať skutočnosť, že používateľ s právom WATCH nesmie za žiadnych okolností upraviť prototyp. Validácia prebieha na strane NodeJS Servera.

### 5.5.5.1.3 Dátový model

Poznámka: Zelený štvorec v NoSQL diagrame znázorňuje keyspace.



Obrázok 40 - Databázový Model pre editor



Obrázok 41 - NoSQL databázový model pre editor

#### 5.5.5.1.4 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 nekorešponujú 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.

### **5.5.5.2 Kolaboratívny editor**

Po implementácii základnej kostry editora sme pristúpili k implementácii kolaboratívneho módu, čo je hlavným poslaním vyvíjaného systému. K implementácii zmien v prototypu 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 zapracovali podporu viacerých simultánne editovateľných prototypov a prácu s nimi. Pre každého kolaboranta udržujeme informáciu, na ktorom prototypu aktuálne pracuje, pomocou čoho dokážeme kategorizovať pripojených kolaborantov.

Ďalej sme sa sústredili na vyladenie kolaborácie a synchronizáciu zmien v prototypoch. Počas testovania v rámci implementácie sme narazili na problém s vymazanými komponentami, ktoré boli označené iným kolaborantom, preto sme začali uvažovať nad možnosťou uzamykania komponentov o čom sa dočítate nižšie.

### 5.5.5.2.1 Akceptačné kritéria

- Používateľ dokáže vytvoriť novú stránku
  - Po vytvorení stránky je používateľ presmerovaný na novo vytvorenú stránku
  - Po vytvorení stránky musia byť všetci kolaboranti notifikovaní
- Používateľ dokáže meniť obsah novej stránky bez ovplyvnenia ostatných stránok
- Používateľ dokáže premenovať stránku
  - Premenovanie stránky musím najskôr potvrdiť
  - Po premenovaní stránky musia byť všetci kolaboranti notifikovaní
- Používateľ dokáže odstrániť stránku
  - Odstránenie stránky musím najskôr potvrdiť
  - Po odstránení bude používateľ presmerovaný na stránku na ľavo od vymazanej
  - Po vymazaní stránky sa stránka (a jej vzťah) reálne nezmažú iba sa označia v DB ako deleted
  - Po vymazaní stránky musia byť všetci kolaboranti notifikovaní
- Používateľ sa dokáže prepínať medzi stránkami
  - Pri prepnutí na inú stránku sa kolaborant zobrazí v paneli kolaborantov na danej stránke
  - Pri prepnutí na inú stránku sa zobrazia dáta konkrétnej stránky
- Používateľ dokáže kolaborovať s ostatnými kolaborantami
  - 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
  -
- Používateľ dokáže prepojiť vytvorené stránky
- Používateľ dokáže uzamknúť vybraný komponent alebo skupinu komponentov

### 5.5.5.2.2 Validácia

Vytvorenie stránky sa vykoná iba v prípade, že prebehne úspešný dopyt na vytvorenie záznamu.

Premenovanie stránky sa vykoná iba v prípade, že prebehne úspešný dopyt na premenovanie záznamu.

Pri vymazaní stránky je potrebné overiť či kolaborant, ktorý sa snaží vymazať stránku má právo na túto akciu, konkrétne aby nevymazal stránku na inom projekte. Základnú index stránku nie je možné vymazať, čo je kontrolované pred dopytom do databázy.



### 5.5.5.2.3 Práca s prototypmi

Na vytváranie, mazanie, prepínanie stránok a notifikovanie o akciách sa používajú správy posielané pomocou NodeJS a pre ukladanie informácií o stránkach(komponenty a štýly) sa používa CQL(Cassandra) databáza.

Komponenty a štýly sú v CQL databáze uložené a sprístupnené cez ich ID a neexistuje vzťah medzi stránkou a projektom. Neoprávnené prístupy na modifikáciu stránky sú ošetrené na strane NodeJS servera, ktorý kontroluje na akej stránke sa nachádza kolaborant a teda či má právo modifikovať stránku.

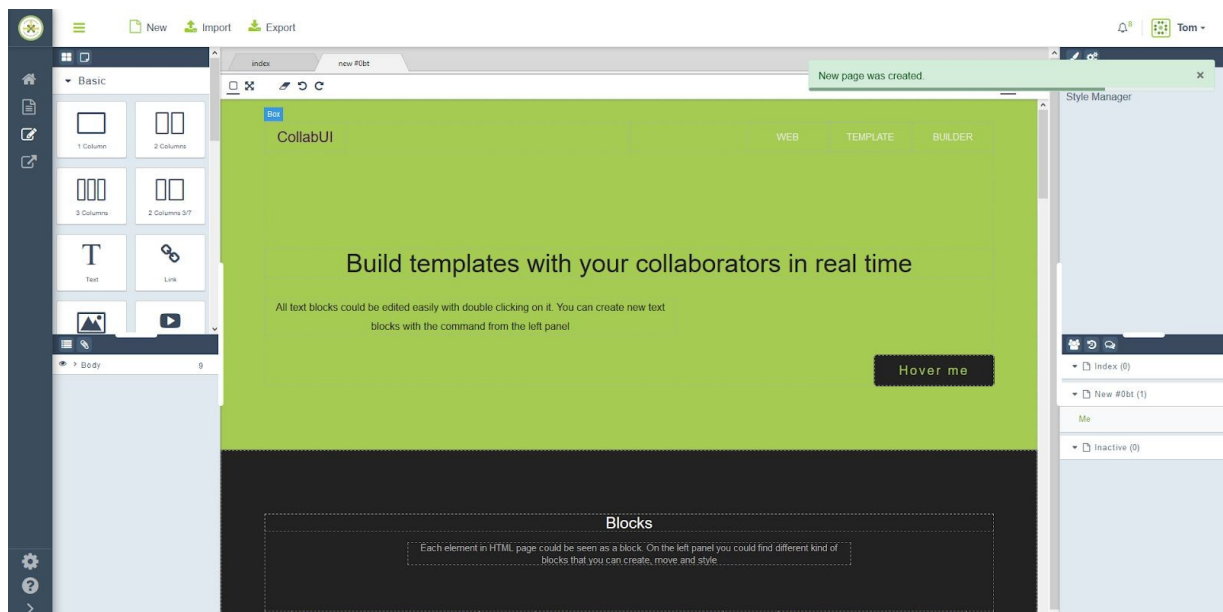
### Vytvorenie stránky

Ako prihlásený používateľ v editore chcem vedieť vytvoriť novú stránku. Vytváranie novej stránky sa vykonáva v editore v konkrétnom projekte.

Zoznam vytvorených stránok sa nachádza v SQL databáze a prístupuje sa k nim cez id.

Pri kliknutí na vytvorenie novej stránky sa otvorí modálne okno, do ktorého je potrebné zadať názov novej stránky. Potvrdenie akcie v modálnom okne zavolá dopyt do databázy a po úspešnom zápise sa vytvorí nová karta vo vrchnej časti editora a notifikujú sa ostatní kolaboranti. Tento proces sa nachádza v *panelPrototypes.js* vo funkcii *createPage*.

Po tomto procese je vytvorená stránka načítaná aj do oboch bočných panelov, konkrétne do panelu Prototypes a Collaborators.

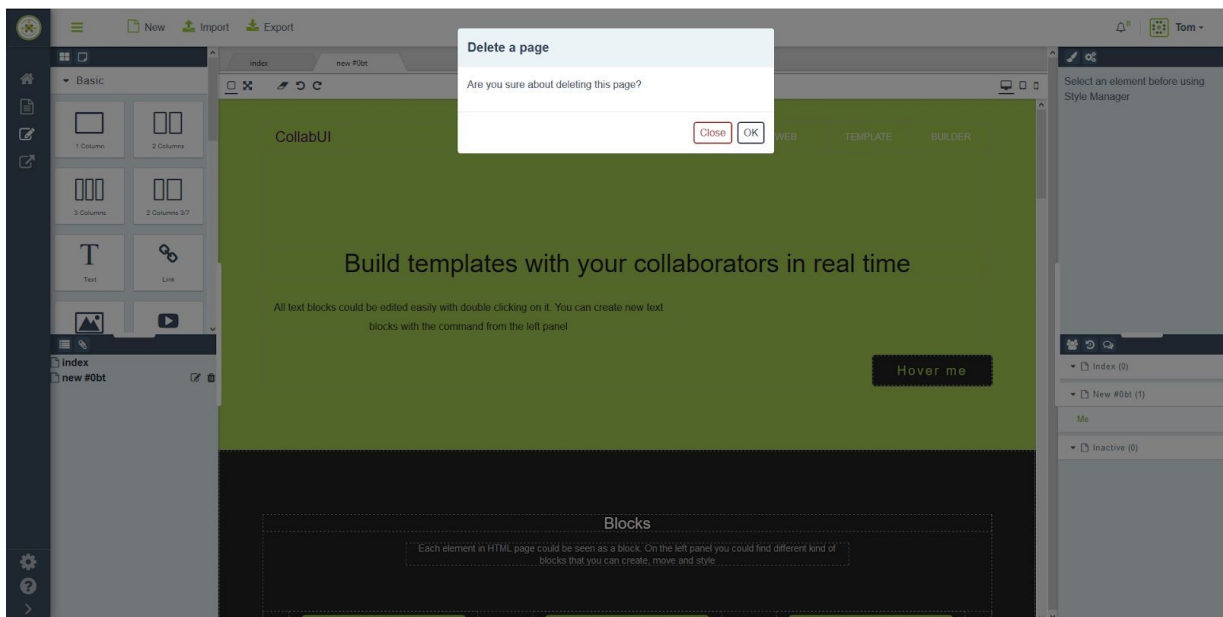


Obrázok 42 - Obrázovka pre vytvorenie novej stránky

## Zmazanie stránky

V tomto prípade použitia sa odstráni stránka u kolaboranta, ktorý túto akciu inicioval a notifikujú sa ostatní kolaboranti aby sa aj im stránka odstránila z editora.

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



Obrázok 43 - Obrázovka pre mazanie stránky

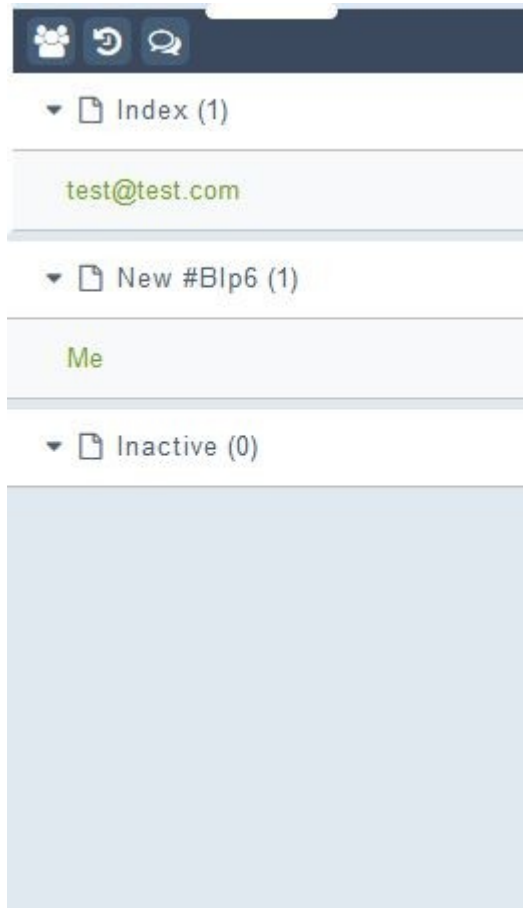
## Prepínanie medzi stránkami

Prepínanie medzi stránkami používa NodeJS server na notifikovanie a načítavanie obsahu inej stránky. Kolaboranti sú zobrazovaní na akej stránke sú momentálne aktívni, táto informácia sa nachádza v paneli kolaborantov.

Pri prepnutí na inú stránku sa z databázy získajú dáta o stránke(komponenty a štýly) a načítajú sa do editora v *EditorClient.js*. Počas načítavania sa pozastavia všetky event listenery(napr. component:add) a po načítaní sa obnovia. Pozastavenie sa vykonáva pomocou nastavenia boolean premennej *isPaused* v *CollaborationComponent.js* pričom sa táto premenná mení pomocou funkcie *pause()* a *unpause()*.

## Panel kolaborantov

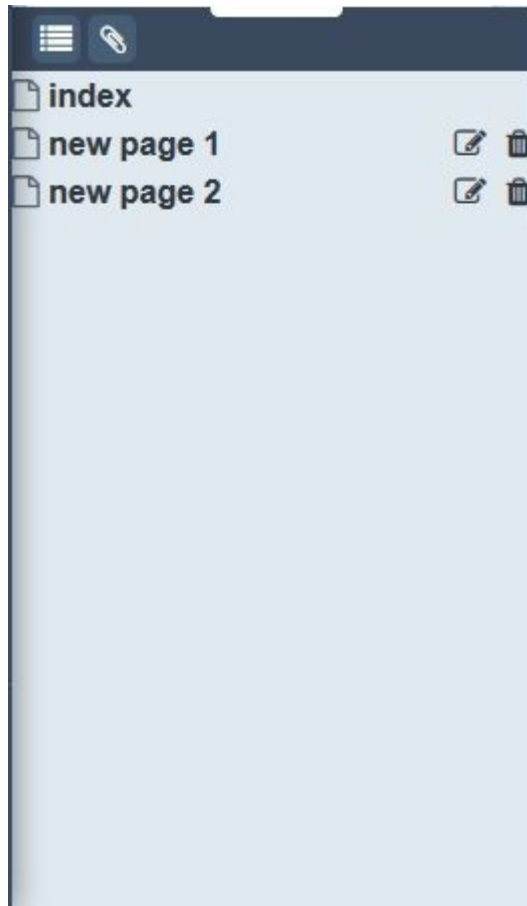
Pri prepínaní medzi stránkami sa každý kolaborant zobrazuje na momentálne aktívnej stránke alebo v inaktívnej zložke v prípade, že nie sú pripojení. V zátvorke sa zobrazuje počet kolaborantov a tieto zložky je možné otvárať alebo zatvárať.



Obrázok 44 - Obrazovka pre panel kolaborantov

## Panel stránok

V paneli stránok sa zobrazujú všetky vytvorené stránky spolu s základnou index stránkou. Tieto stránky je možné premenovať pomocou tlačidla s ikonou ceruzky a vymazať s tlačidlom s ikonou odpadkového koša.



Obrázok 45 - Obrazovka pre panel stránok

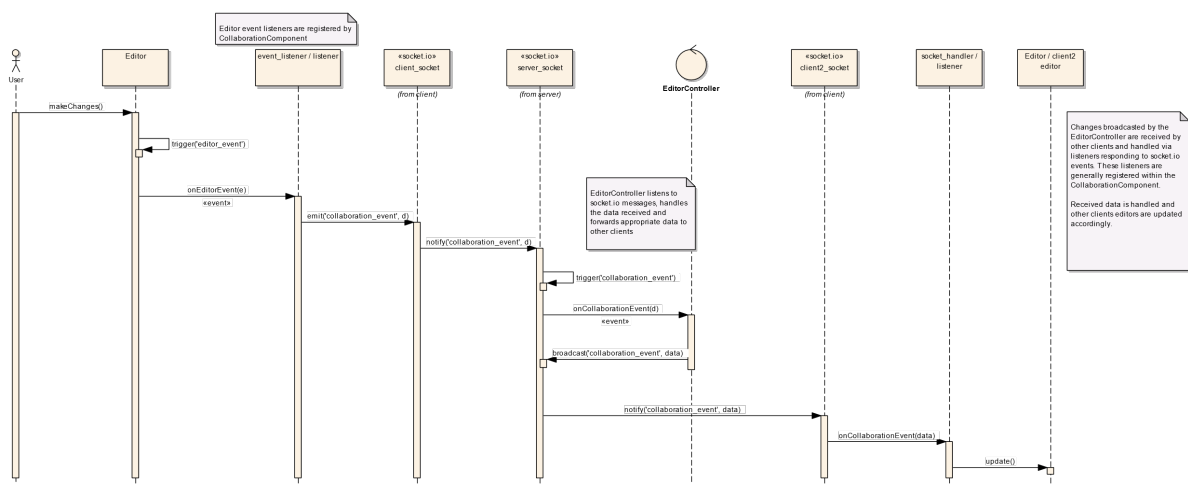
### 5.5.5.2.4 Synchronizácia zmien v prototypoch

Táto kapitola poskytuje vysokoúrovňový pohľad na zabezpečenie synchronizácie dát medzi používateľmi modifikujúcimi jeden prototyp používateľského rozhrania súčasne. Propagácia zmien vykonaných jedným klientom je zabezpečená dvoma aplikačnými komponentami:

- **CollaborationComponent:** komponent zabezpečujúci klientsku časť kolaborácie
- **EditorController:** ovládač na strane Node.js serveru zodpovedný za spracovanie dát od klientov vykonávajúcich zmeny a preposielanie spracovaných zmien ostatným klientom

Preposielanie dát po sieti je zabezpečené pomocou JavaScript knižnice socket.io. Vo všeobecnosti sú zmeny synchronizované nasledovným spôsobom:

1. Klient A vykoná zmeny v editore
2. Objekt editoru vyvolá patričnú udalosť
3. Listener zaregistrovaný CollaborationComponent inštanciou spracuje danú udalosť a potrebné dáta odošle na Node.js server pomocou socket.io
4. Server spracuje prijaté dáta a pomocou socket.io ich vyšle ostatným klientom pripojených do miestnosti stránky, na ktorej Klient A pracoval
5. Klienti B a C prijmú dáta zo serveru pomocou listenerov zaregistrovaných inštanciou CollaborationComponent na ich socket
6. Listener spracuje prijaté dáta a vykoná potrebné zmeny v editore



Obrázok 46 - Posielanie zmien vykonaných v editore ostatným klientom (všeobecný prípad)

## Klient

Kolaborácia je na strane klienta zabezpečená pomocou komponentu CollaborationComponent. Zmeny vykonané v editore sú odosielané na server pomocou knižnice socket.io

### **CollaborationComponent.js**

Komponent zabezpečujúci kolaboráciu na strane klienta. Okrem iného je zodpovedný predovšetkým za nasledovné:

- Registráciu listenerov pre udalosti, ktoré nastávajú počas kolaborácie v editore
- Registráciu socket.io listenerov, ktoré spracovávajú dáta prijaté zo serveru (socket.io namespace: *editor*)
- Spracovanie dát prijatých zo serveru
- Odosielanie lokálnych zmien server

### **initPlugin(client, editor, options)**

Funkcia pre inicializáciu komponentu. Očakávané argumenty:

- *client*: inštancia knockout klienta
- *editor*: inštancia grapes.js editoru
- *options*: objekt s nastaveniami komponentu
  - *socket*: inštancia socket.io socketu pripojeného na *editor* namespace
  - *isPaused*: boolean, pokiaľ je atribút nastavený na true, tak sa komponent inštanciu v stave, v ktorom nebude reagovať na grapes.js a socket.io udalosti

Úlohou tejto funkcie je inicializovať komponent tak, aby bol schopný zabezpečiť kolaboráciu. Zavolaním tejto funkcie prvý krát sa zaregistrujú všetky potrebné event listenery pre grapes.js a socket.io a preťaží sa istá časť grapes.js funkcionality.

```

CollaborationComponent.prototype.initPlugin = function (client, editor, options) {
  var self = this, socket = undefined;

  if (this.plugin || !options.hasOwnProperty('socket')) {
    return;
  }

  socket = options.socket;

  this.instance = editor;
  this.client = client;
  this.isPaused = options.paused || false;

  grapesjs.plugins.add(this.pluginId, function (editor, options) {
    self.overrideEditorFunctionality(editor);
    self.enableRemoteAdding(editor, socket);
    self.enableRemoteRemoving(editor, socket);
    self.enableRemoteSelecting(self.client, editor, socket);
    self.enableRemoteUpdating(editor, socket);
    self.enableRemoteMoving(editor, socket);

    socket.on('component:assignIDs', function (data) {
      var source = undefined, target = undefined;

      if (typeof data === 'string' || data instanceof String) {
        target = editor.DomComponents.getComponents().models.slice(0);
        source = JSON.parse(data);
      } else {
        target = self.findComponent(data.cid);
        source = JSON.parse(data.JSONmodel);
      }

      if (target) {
        self.assignIDsToObjects(source, target);
        self.prepareCssSelectors(target);
      }
    });

    console.log('Collaboration plugin enabled!');
  });

  this.plugin = grapesjs.plugins.get(this.pluginId);
  if (this.plugin) {
    this.plugin(editor, options);
  } else {
    console.log('Could not initialize collaboration component.');
```

## overrideEditorFunctionality(editor)

Funkcia zodpovedná za úpravu správania editoru. V súčasnosti iba zabezpečuje rozšírenie konštruktorov grapes.js objektov.

```
CollaborationComponent.prototype.overrideEditorFunctionality = function (editor) {  
    this.extendComponents(editor);  
};
```

## extendComponents(editor)

Funkcia pre rozšírenie konštruktorov objektov v inštancii grapes.js editoru. V rámci tejto funkcie sú rozšírené nasledovné konštrukty:

- kolekcia *Components*
- všetky konštrukty komponentov (objekty dediace od grapes.js triedy *Component*)

```
CollaborationComponent.prototype.extendComponents = function (editor) {  
    //Override the default Components collection to allow listeneing to Component  
    add/remove events  
    var Components = editor.DomComponents.Components, self = this;  
  
    editor.DomComponents.Components = Components.extend({  
        initialize: function (models, opts = {}) {  
            Components.prototype.initialize.apply(this, arguments);  
            this.on('add', self.onAddedToCollection);  
        }  
    });  
  
    // Get the default models of all components and extend them with custom  
    functionality  
    var componentTypes = editor.DomComponents.componentTypes, i = 0;  
    for (i; i < componentTypes.length; i++) {  
        componentTypes[i].model = this.extendDefaultComponent(editor,  
        componentTypes[i].model);  
    }  
  
    // Further extend other component models  
    var linkCmpModel = this.getComponentType(editor, 'link').model;  
    this.extendLinkComponent(editor, linkCmpModel);  
};
```

Grapes.js v súčasnosti neposkytuje vhodný spôsob pre sledovanie pohybu komponentov pri presúvaní. Z toho dôvodu musela byť funkcia *initialize* kolekcie *Components* preťažená tak, aby sa dalo odsledovať, do ktorej kolekcie bol komponent vložený.

**Poznámka:** Presun komponentu z kolekcie sa dá odsledovať počúvaním udalosti *run:tlb-move:before*, ktorú vysiela inštancia editoru.

Súvisiace časti: [enableRemoteMoving\(editor, socket\)](#)



## extendDefaultComponent(editor, model)

Funkcia pre rozšírenie komponentov grapes.js editoru. Úlohou tejto funkcie je rozšíriť komponenty tak, aby:

- v JSON reprezentácii objektu neostávali dáta používané pre zabezpečenie kolaborácie
- aby sa dali pri inicializácii aplikovať štýly uložené v internom atribúte `__style`, do ktorého sa zároveň cacheujú (aby sa mohli odoslať ostatným klientom)

```
CollaborationComponent.prototype.extendDefaultComponent = function (editor, model)
{
  return model.extend({
    toJSON: function () {
      var obj = model.prototype.toJSON.apply(this, arguments);

      delete obj.addedByAnother;
      delete obj.addedLocally;
      delete obj.removedByAnother;

      return obj;
    },
    initialize: function (attr = {}, ops = {}) {

      if (this.get('addedByAnother') === true) {
        this.set('style', this.get('__style'), {silent: true});
        this.unset('__style', {silent: true});
      } else {
        this.set('__style', this.get('style'), {silent: true});
      }

      model.prototype.initialize.apply(this, arguments);
    },
    initComponents: function () {
      // Have to add components after the init, otherwise the parent
      // is not visible
      const comps = new editor.DomComponents.Components(null, this.opt);
      comps.parent = this;
      !this.opt.avoidChildren && comps.reset(this.get('components'));
      this.set('components', comps);
      return this;
    }
  });
};
```

## findComponent(cid, [byRemote = False])

Funkcia pre hľadanie komponentov v grapes.js strome. Pokiaľ je atribút `byRemote` false, tak budú komponenty vyhľadávané pomocou atribútu `cid`, ktorý je interne generovaný grapes.js. Tento atribút identifikuje komponenty iba lokálne a môže sa u rôznych klientov líšiť a navyše je generovaný pri každom inštanciovaní komponentu – tzn., že pri každej relácii môže mať inú hodnotu.

Pokiaľ je hodnota argumentu `byRemote` true, tak budú komponenty hľadané pomocou interného atribútu `__sharedID`. Tento atribút je generovaný serverom a mal by správne identifikovať komponenty u všetkých klientov.

### **assignIDsToObjects(source, target, [idAttr = “\_\_sharedID”])**

Funkcia pre priradovanie ID generovaných serverom lokálnym komponentom. Táto funkcia je volaná napríklad v prípade, že klient A pridá nový komponent do prototypu stránky. Nový komponent (a jeho deti) sa odošlú na server, ktorý im všetkým vygeneruje unikátne id. Toto id sa uloží do atribútu `__sharedID` a upravené komponenty sa odošlú ostatným klientom. Následne sa aktualizovaný strom odošle aj klientovi A, u ktorého sa novo-vygenerované id priradia vytvoreným komponentom.

### **enableRemoteAdding(editor, socket)**

Funkcia inicializujúca synchronizáciu vytvárania nových komponentov. Táto funkcia zaregistruje listener na udalosť `component:add` grapes.js editoru, ako aj rovnomenný listener pre klientsky socket. Vyvolanie udalosti `component:add` editorom znamená, že bol pridaný nový komponent, ktorý sa musí odoslať na server. Vyvolanie udalosti `component:add` socketom znamená, že bol zo serveru prijatý nový komponent, ktorý sa musí pridať do editoru.

**Poznámka:** Presúvanie komponentu v editore z jedného miesta na druhé v súčasnosti nespustí udalosť `component:add`. Napriek tomu ale stále spustí udalosť `component:remove`.

#### **(editor, socket)**

Funkcia inicializujúca synchronizáciu vymazávania nových komponentov. Táto funkcia zaregistruje listener na udalosť `component:remove` grapes.js editoru, ako aj rovnomenný listener pre klientsky socket. Vyvolanie udalosti `component:remove` editorom znamená, že bol odstránený komponent. O tejto skutočnosti je server informovaný pomocou socket.io správy označenej `component:remove`. Prijatie udalosti `component:remove` socketom znamená, že iný klient vymazal komponent, ktorý treba odstrániť lokálne.

**Poznámka:** Presúvanie komponentu v editore z jedného miesta na druhé v súčasnosti nespustí udalosť `component:add`. Napriek tomu ale stále spustí udalosť `component:remove`.

### **enableRemoteUpdating(editor, socket)**

Funkcia inicializujúca synchronizáciu zmien vykonaných nad komponentami. Táto funkcia registruje listenery na nasledovné grapes.js udalosti:

- `component:styleUpdate` – udalosť vyvolaná pri zmene štýlu komponentu
- `component:update` – udalosť vyvolaná pri zmene atribútov komponentu

Táto funkcia registruje listenery na nasledovné socket.io udalosti:

- *component:styleUpdate* – udalosť vyvolaná pri prijatí zmien štýlu komponentu od serveru
- *component:update* – udalosť vyvolaná pri prijatí zmien atribútov komponentu od serveru

Táto funkcia ďalej registruje *add* listener na kolekciu CSS pravidiel grapes.js komponentu *CssComposer*, ktorý je súčasťou inštancie editora.

### **enableRemoteMoving(editor, socket)**

Funkcia zabezpečujúca synchronizáciu presúvania komponentov. Úlohou tejto funkcie je zaregistrovať listenery na grapes.js udalosť *run:tlb-move:before*. Táto udalosť je vyvolaná keď používateľ začne presúvať komponent v editore. Ďalej táto funkcia obsahuje listener, ktorý sa spustí pri pridaní komponentu do *Components* kolekcie. Tento stav nastane pri vytvorení nového komponentu, ale aj pri premiestnení komponentu z jedného miesta na druhé. Tieto dva stavy sa rozlíšia a informácie o presunutom komponente sa odošlú na server. Informácie o komponentoch presunutých ostatnými klientami sú prijaté pomocou socket.io socketu a správy označenej *component:moved*.

**Poznámka:** Presúvanie komponentu v editore z jedného miesta na druhé v súčasnosti nespustí udalosť *component:add*. Napriek tomu ale stále spustí udalosť *component:remove*.

**Súvisiace časti:** [extendComponents\(editor\)](#)

### **socket.io udalosti**

Táto časť obsahuje opis socket.io udalostí, ktoré môže klient obdržať v súvislosti s procesom zabezpečenia kolaborácie viacerých klientov.

#### **component:add**

Správy obsahujúce nový komponent, ktorý bol pridaný iným klientom.

#### **component:assignIDs**

Správy obsahujúce vygenerované ID, ktoré treba priradiť lokálnym komponent aby bolo možné identifikovať komponenty počas kolaborácie.

#### **component:remove**

Informácie o vymazaní komponentu iným klientom.

### **component:update**

Zmena atribútov komponentu.

### **component:styleUpdate**

Zmena štýlu komponentu.

### **component:moved**

Zmena pozície komponentu.

## **Server**

Zodpovednosť serveru v procese kolaborácie leží prioritne v preposielaní dát medzi klientami a ukladanie dát do databázy Cassandra.

### **EditorController.js**

Ovládač zodpovedný za zabezpečenie preposielania dát medzi klientami pomocou socket.io socketu pre *editor* namespace.

### **onConnect(socket, connection)**

Funkcia volaná pri úspešnom pripojení klienta na server.  
Táto funkcia plní nasledovné úlohy:

- priradí socket do miestnosti projektu
- pridá socket do miestnosti stránky projektu
- zaregistruje listenery prístupné používateľom s oprávnením *watch*
- zaregistruje listenery prístupné používateľom s oprávnením *edit*

### **joinPageRoom(socket)**

Pridanie socketu do miestnosti projektovej stránky. Pokiaľ je socket pripojený do inej miestnosti stránky, tak bude z tejto miestnosti vyradený.

### **joinProjectRoom(socket)**

Pridanie socketu do miestnosti projektu.

### **addWatchLevelListeners(socket)**

Registrácia listenerov pre udalosti prístupné používateľom s oprávnením *watch*. Napr.: zmena stránky, načítanie dát prototypu a načítanie komponentov, s ktorými pracujú ostatní kolaboranti.

## addEditLevelListeners(socket)

Pridanie listenerov pre udalosti prístupné používateľom s oprávnením *edit*:

- ukladanie dát stránky
- voľba komponentu
- zrušenie voľby komponentu
- uzamknutie/odmknutie komponentu
- pridanie, odstránenie, presunutie a úprava komponentu

## onComponentAdd(data, socket)

Funkcia pre spracovanie novo pridaných komponentov. Táto funkcia prijme dáta stromu nových komponentov, priradí im ID, odošle komponenty ostatným kolaborantom na pridanie a odošle upravený strom jeho tvorcovi pre priradenie nových ID.

```
EditorController.prototype.onComponentAdd = function (data, socket, respond) {
  var self = this;
  var page_id = socket.editorData.page_id;

  //Synchronous query to prevent counter value collision during multiple requests
  Page.findOne({page_id: page_id}, function (err, page) {
    if (err || !page) return;
    var component = JSON.parse(data.JSONmodel);
    var generated = CM.prepareComponents(component, page.component_counter,
true);

    component = generated.root;
    data.JSONmodel = JSON.stringify(component);

    console.log('Saving page', page.page_id, 'with counter',
page.component_counter);
    Page.update({page_id: socket.editorData.page_id}, {component_counter:
Long.fromInt(generated.counter)}, {}, function (err) {
      if (err) {
        console.log(err);
      } else {
        socket.emit('component:assignIDs', data);
        socket.broadcast.to(socket.pageRoom).emit(
          'component:add',
          self.withUser(data, socket)
        );
      }
    });
  });
};
```

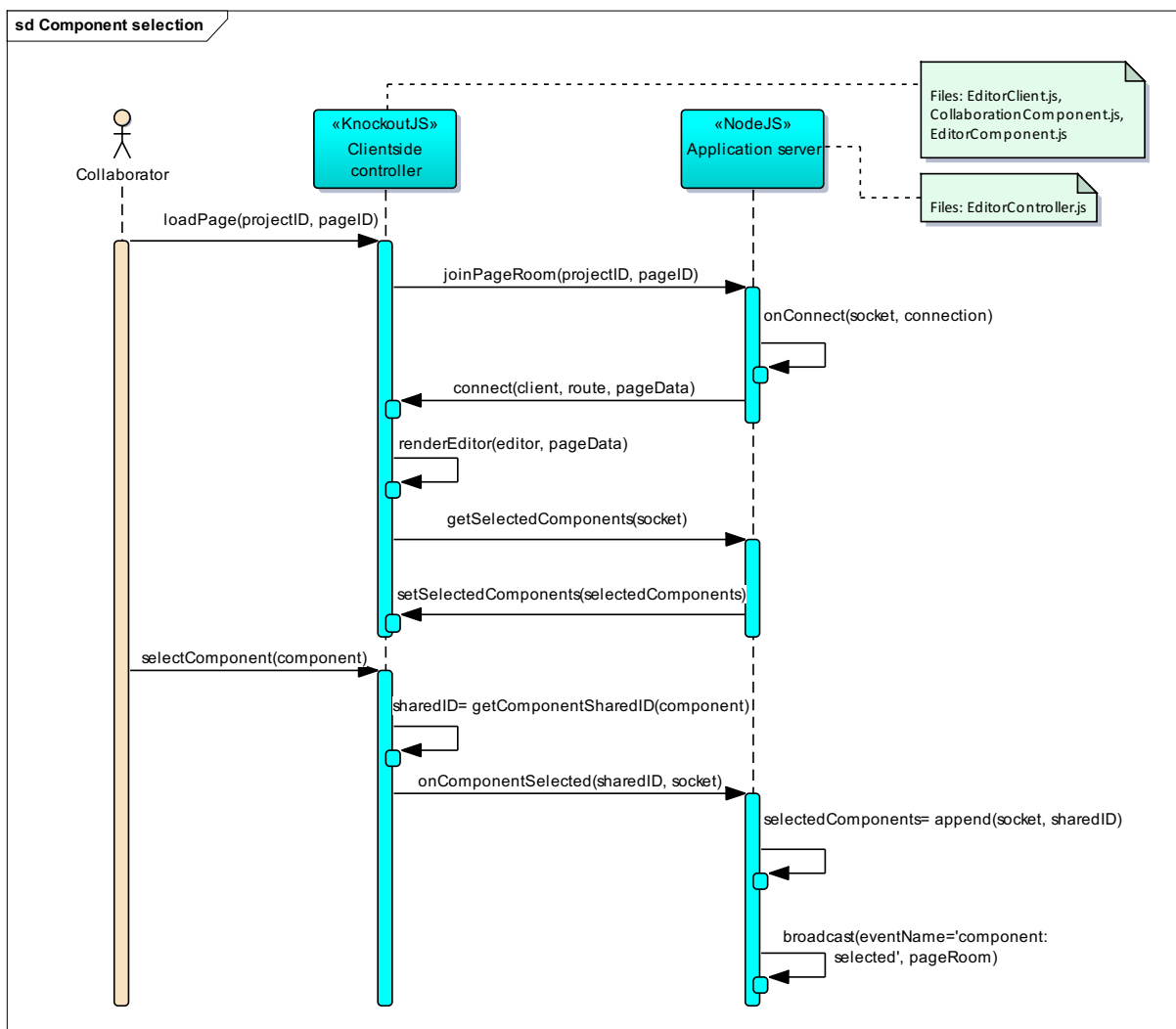
## onComponentUpdate, onComponentStyleUpdate, onComponentRemove, onComponentMoved

V kontexte kolaborácie je úlohou tejto sady funkcií prakticky iba preposlať dáta ostatným kolaborantom.

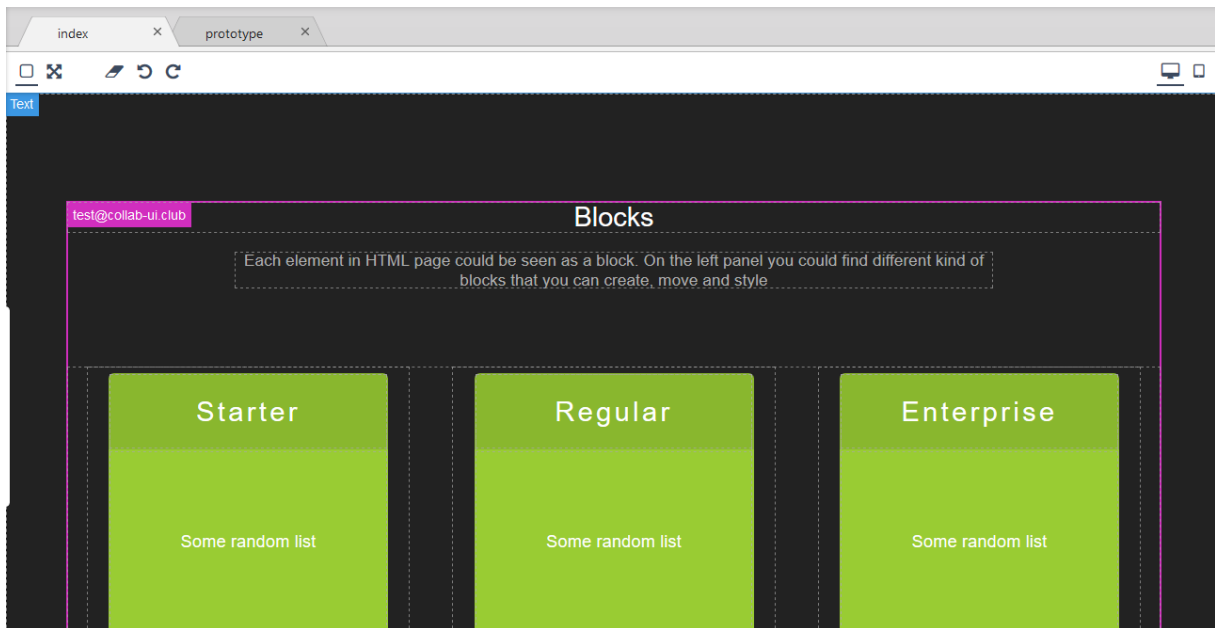
### 5.5.5.2.5 Vizualizácia označených komponentov

Zvýrazňovanie označených komponentov je pridaná funkcionlita nad interaktívnym editorom aplikácie, ktorý vychádza z knižnice GrapesJS. Vďaka nej má používateľ vždy prehľad o tom, s akými komponentami pracujú ostatní členovia tímu. Zmeny sú uložené na aplikačnom serveri a prenášajú sa vždy pri pripojení alebo odpojení kolaboranta alebo pri označení ľubovoľného komponentu. Zároveň pri zmene šírky bočných panelov editora vždy dochádza ku explicitnému prekresleniu vyznačených komponentov, nakoľko v opačnom prípade by mohlo dôjsť k ich nesprávnemu zarovnaníu.

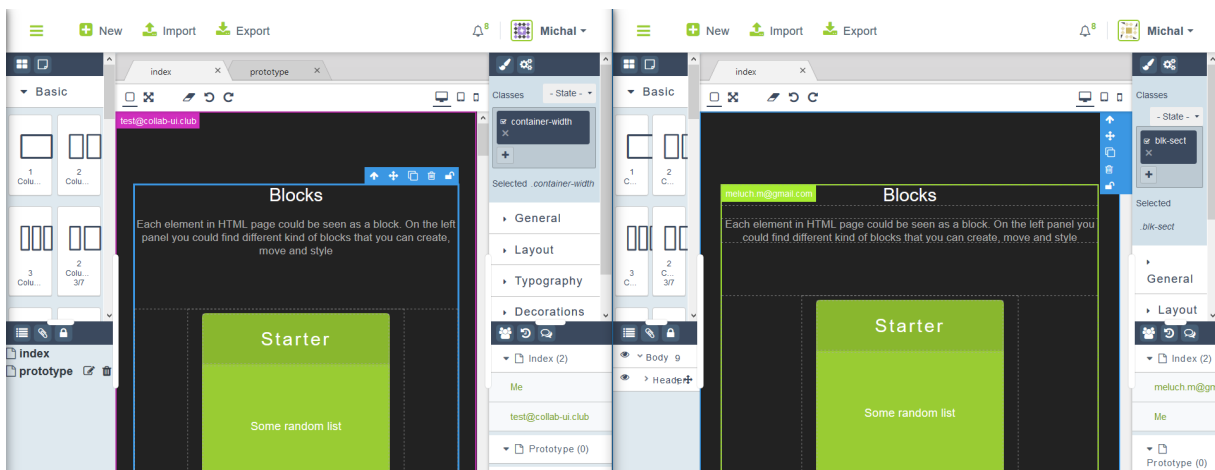
Pri zvýrazňovaní dôjde ku prepisu CSS štýlu daného komponentu (pridanie orámovania) a ku pridaniu HTML elementu obsahujúceho štítok s e-mailovou adresou daného kolaboranta. Nemodifikujeme teda funkcionlitu knižnice tretej strany, ale pridávame separátne akcie, ktoré sa vykonajú až po skončení všetkých ďalších udalostí.



Obrázok 47 - Udalosti súvisiace s označením komponentu v editore po jeho načítaní



Obrázok 48 - Zvýraznenie komponentu, ktorý je práve označený iným kolaborantom



Obrázok 49 - Zvýraznenie označených komponentov medzi dvoma rôznymi kolaborantami súčasne

Označovanie komponentov prebieha tak, ako všetky ostatné akcie pomocou websocketov, ktoré obsluhuje aplikačný server NodeJS. Po každom označení komponentu na strane klienta dôjde ku vyslaniu signálu `component:selected` a jeho obsluženiu metódou `onComponentSelected` na strane servera (`EditorController.js`).

## Odoslanie udalosti vlastného označenia komponentu

Táto časť opisuje implementáciu súvisiacu s označením komponentu na strane klienta a prenesením tejto zmeny ku ostatným kolaborantom. Server si udržiava zoznam označených komponentov v globálnej premennej `selectedComponents`. Ide o objekt, ktorý si uchováva informácie o tom, ktorý označený komponent patrí ku ktorej stránke (resp. prototypu) a kto je jeho „majiteľom“ – t. j. kto ho označil:

```
selectedComponents = {
  pageRoom: {
    userToken1: { selectedComponentID },
    userToken2: { selectedComponentID },
    ...
  }
}
```

Priebežné uchovávanie označených komponentov v serverovej premennej je dôležité pre prípad, že sa k projektu pripojí ďalší kolaborant a chceme mu poskytnúť informáciu o tom, ktoré komponenty sú na danej stránke už označené inými ľuďmi. Na novoprijatú správu o označení komponentu zareaguje tak, že si k danej stránke priradí nový záznam o označení komponentu kolaborantom. Následne rozošle informáciu o tejto udalosti všetkým ostatným kolaborantom pracujúcim na danej stránke (resp. prototypu). Informácie o pôvodcovi udalosti (`userToken`, `pageID`) berie server z objektu `socket`, ktorý sa defaultne posielajú spolu s údajmi a názvom udalosti pri akejkoľvek komunikácii:

```
socket.on('component:selected', function (data) {
  self.onComponentSelected(data, socket);
});
```

```
EditorController.prototype.onComponentSelected = function (data, socket) {
  var self = this;

  // If this is the first user selecting a component in the page room
  if (!this.selectedComponents.hasOwnProperty(socket.pageRoom)) {
    this.selectedComponents[socket.pageRoom] = {};
  }

  var component = {
    'idPrev': self.selectedComponents[socket.pageRoom][socket.userToken],
    'idNew': data
  };

  // Notify all other users in the page room about the newly selected component
  socket.broadcast.to(socket.pageRoom).emit(
    'component:selected',
    self.withUser(component, socket)
  );

  // Update the server-side selected components object
  this.selectedComponents[socket.pageRoom][socket.userToken] = data;
};
```



Implementácia odoslania udalosti označenia komponentu na strane klienta je pomerne jednoduchá a zahŕňa len vyhľadanie jeho zdieľaného ID (špeciálna vetva pre body element) a odoslanie údajov na server:

```
// Notify the server when the current user selects a component (editor listener)
editor.on('component:selected', function (components) {

    var changedComponent = components[0].changed.selectedComponent;
    var sharedID = undefined;
    self.mySelectedComponent = changedComponent;

    console.log('Selected component: ', changedComponent);

    // Special case - body (wrapper) component
    if (self.isComponentWrapper(changedComponent)) {
        sharedID = changedComponent.cid;
    }
    // All other user components
    else if (changedComponent.attributes.hasOwnProperty('__sharedID')) {
        sharedID = changedComponent.attributes['__sharedID'];
    }
    else {
        console.error('Failed to retrieve selected component ID from: ', changedComponent);
        return;
    }

    socket.emit('component:selected', sharedID);
});
```

Označenie komponentu je implementované obdobne, avšak iniciátorom udalosti je server. Dôvodom je, že udalosť označenia komponentu sa odosiela len pri odpojení kolaboranta z aktuálnej stránky prototypu (napr. vypnutie prehliadača, zmena aktuálneho prototypu..). Informáciu o tejto udalosti má k dispozícii logicky len server – konkrétne je iniciátorom tohto volania funkcia `EditorController.onDisconnect(socket)`.

```
/**
 * Broadcasts a notice that the user with defined socket has deselected a component
 *
 * @param socket
 */
EditorController.prototype.emitComponentDeselect = function (socket) {
    var self = this;

    if (this.hasUserAnyElementSelected(socket.pageRoom, socket.userToken)) {
        socket.broadcast.to(socket.pageRoom).emit(
            'component:deselected',
            {
                user: {'token': socket.userToken},
                component: {'id': self.selectedComponents[socket.pageRoom][socket.userToken]}
            }
        );

        // Also remove the component from the selected components object
        delete this.selectedComponents[socket.pageRoom][socket.userToken];
    }
};
```

## Prijatie udalosti cudzieho o(d)značenia komponentu

Táto časť opisuje implementáciu scenára, pri ktorom používateľ pracujúci na prototype obdrží informáciu o označení alebo odznačení komponentu iným kolaborantom, ktorú mu odoslal server. Okamžite po prijatí tejto informácie na strane klienta dôjde ku odznačeniu komponentu, ktorý mal auto udalosti pôvodne označený (metóda `remSelectedComponentStyle`). Následne je na základe informácie o pôvodcovi udalosti a čísla cieľového komponentu možné ho označiť:

```
// Respond to servers notification about another user (or me on other devices/tabs) selecting a component
socket.on('component:selected', function (data) {

    var userToken = data.user.token;
    var user = self.getUserByToken(client, userToken);

    // Reset any previously selected components for this user
    self.remSelectedComponentStyle(userToken);

    self.addSelectedComponentStyle(
        data.component.idNew,
        user.email,
        user.token
    );
});
```

```
// Highlight a component selected by another collaborator
CollaborationComponent.prototype.addSelectedComponentStyle = function (sharedID, userEmail, userToken) {
    var self = this;
    var component = self.findComponent(sharedID, true);

    // Ignore uninitialized components (e. g. 'uncommitted' text nodes)
    if (!component || !component.view) {
        return;
    }

    // Convert the GrapesJS component to a jQuery html DOM object
    var componentHtml = $(component.view.el);

    // Generate the user email badge element & random color to be used for this user
    var selectedComponentBadge = $('<div id="badge" + userToken.replace('@', '') + ">" + userEmail + '</div>')
    var userColor = randomColor({'seed': userToken, 'luminosity': 'bright'});
    var outlineSize = 2; // px

    // Set the user email badge dynamic styles & position
    selectedComponentBadge.addClass('gjs-badge user-selection user-selection-badge');
    selectedComponentBadge.css({ ...
    });

    // Set the selected element outline style
    componentHtml.addClass('user-selection-outline')
    componentHtml.css({ ...
    });

    $('#gjs-tools').prepend(selectedComponentBadge);

    // The canvas wrapper doesn't get a shared ID - send its base CID instead
    this.selectedComponents[userToken] = {
        'id': self.isComponentWrapper(component) ? component.cid : component.attributes['__sharedID'],
        'email': userEmail
    };
};
```

Označenie komponentu prebieha tak, že sa k jeho CSS štýlu pridá atribút `outline` a do editora sa vygeneruje nový HTML element štítku s e-mailom kolaboranta, ktorý ho označil. Farba tohto štítku, ako aj orámovanie (`outline`) cieľového komponentu je generovaná pseudonáhodne na základe e-mailu kolaboranta, ktorý ho označil.

Informácie o označených komponentoch sú po prijatí uchovávané na strane klienta (`CollaborationComponent.selectedComponents`) z dôvodu, že je nutné ich prekreslenie vždy po zmene veľkosti bočných panelov a frekvencia týchto volaní by zbytočne zaťažovala server.

Odznačenie komponentu je na strane klienta spracované obdobne ako označenie – t. j. dôjde ku odstráneniu štítku s menom kolaboranta a orámovania cieľového komponentu. Na záver sa odstráni aj záznam o tomto komponente z lokálneho objektu označených komponentov:

```
CollaborationComponent.prototype.remSelectedComponentStyle = function (userToken) {
  // Remove the generated user email badge on the unselected component
  $('#div#badge' + userToken.replace('@', '')).remove();

  // The selected component style has been already removed (I'm the one who deleted it, for instance)
  if (!this.selectedComponents[userToken]) {
    return;
  }

  // Retrieve the unselected component
  var unselectedComponent = this.findComponent(
    this.selectedComponents[userToken].id, true
  );

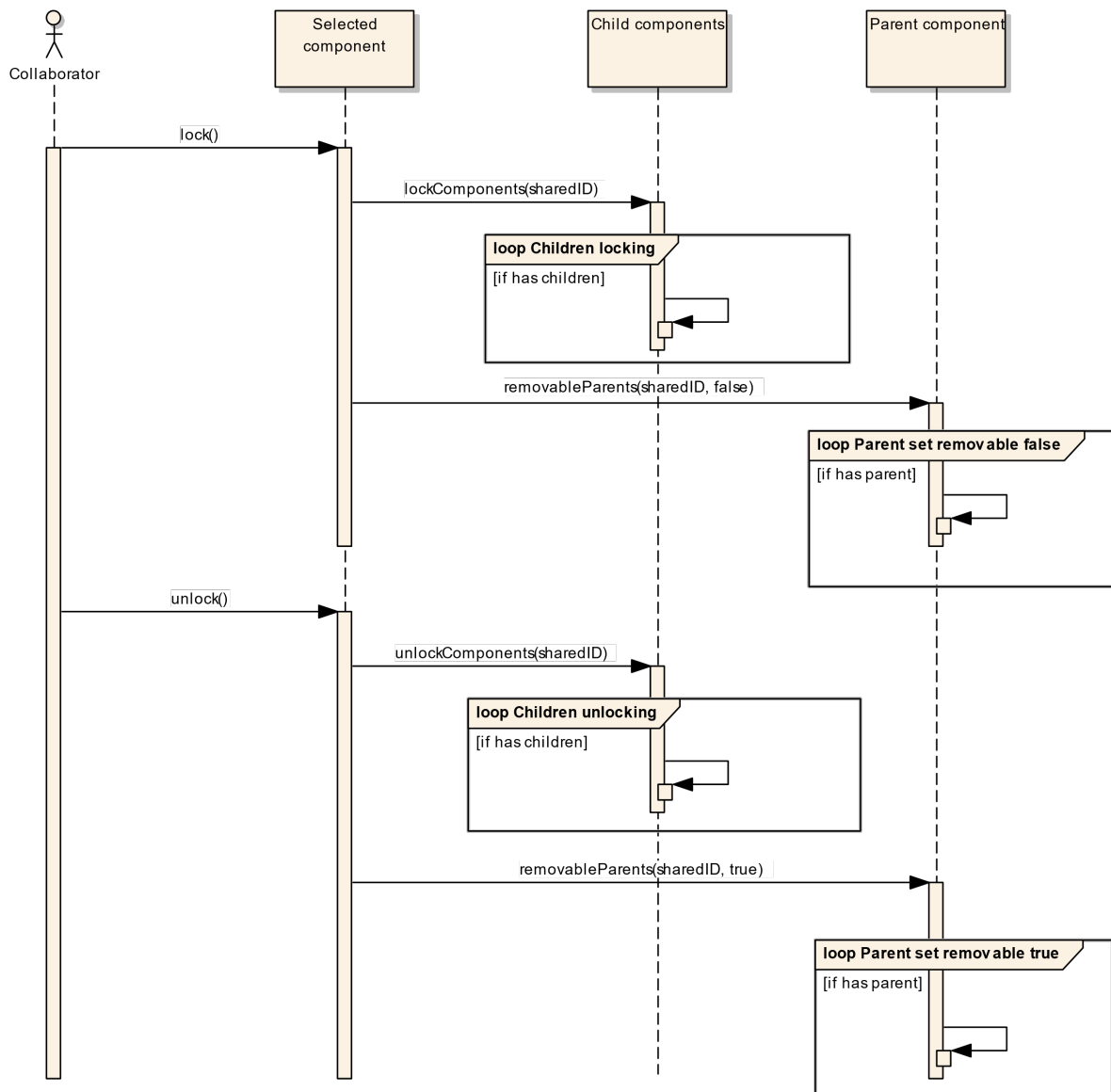
  if (unselectedComponent) {
    // Cast the unselected component HTML to a jQuery object
    var unselectedComponentHtml = unselectedComponent.view.el;
    unselectedComponentHtml = $(unselectedComponentHtml);

    // Remove the outline effect
    unselectedComponentHtml.css({
      'outline': 'initial',
      'outline-color': 'initial'
    });
  }

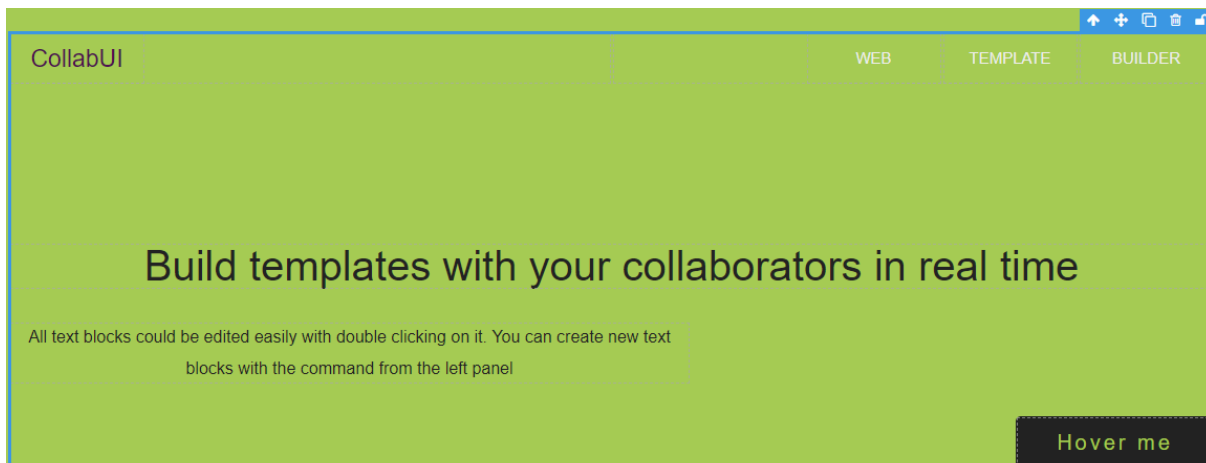
  delete this.selectedComponents[userToken];
};
```

### 5.5.5.2.6 Uzamykanie komponentov

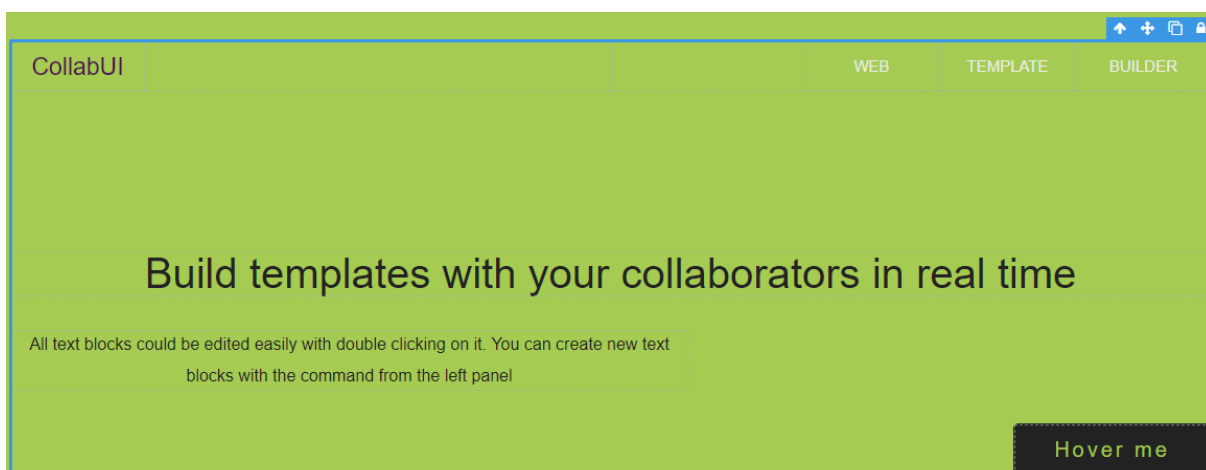
Uzamykanie komponentov je rozširujúca funkcionlita pre GrapesJS. Pomocou uzamykania si môže kolaborant uzamknúť vybraný komponent, ktorý nebudú môcť ostatní kolaboranti upravovať ani premiestňovať. Pri uzamknutí komponentu sa uzamknú aj deti vybraného komponentu. Rovnako sa musí zakázať vymazávanie pre všetky rodičovské komponenty uzamknutého komponentu. Ak by sa vymazávanie rodičovských komponentov nenastavilo, tak by sa mohlo stať, že niekto vymaže rodiča aj s uzamknutým komponentom. Uzamknúť komponent sa dá pomocou ikony zámku v tooltipe označeného komponentu vid'. obrázok 2. Odomknúť komponent sa dá pomocou ikony odomknutého zámku v tooltipe označeného komponentu vid'. obrázok 3 alebo pomocou zoznamu uzamknutých komponentov v panely vid'. obrázok 4. Do zoznamu uzamknutých komponentov sa nepridávajú komponenty uzamknuté ostatnými kolaborantmi. Taktiež sa komponenty odomknú keď kolaborant, ktorý komponenty uzamkol opustí konkrétnu stránku projektu resp. opustí socket.



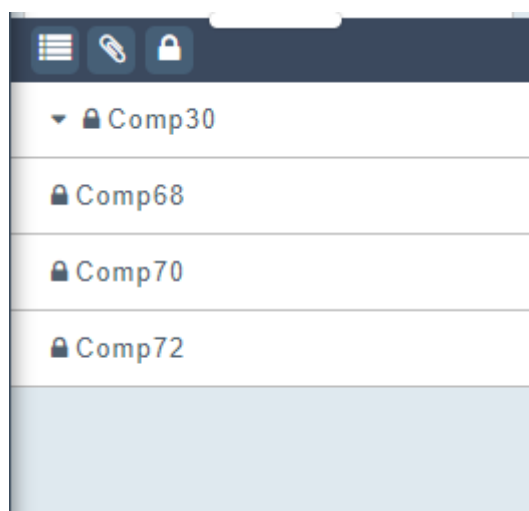
Obrázok 50 - Uzamykanie/odomykanie komponentov



Obrázok 51 - Odomknutý komponent



Obrázok 52 - Uzamknutý komponent



Obrázok 53 - Zoznam uzamknutých komponentov

Uzamykanie a odomykanie v rámci kolaborácie prebieha pomocou soketov. Na node servery (EditorController.js) prebieha spracovanie požiadavky od kolaboranta, ktorý komponent uzamkol alebo odomkol. Následne je pomocou soketov táto požiadavka rozposlaná medzi pripojených kolaborantov. Pre lepšie pochopenie implementácie je potrebné uvedomiť si, či uzamykanie/odomykanie prebieha u kolaboranta, ktorý túto požiadavku inicioval alebo, či sa jedná o kolaboranta, ktorý túto požiadavku prijal.

Prvým krokom pri implementácii je pridanie tlačidla do tooltipu označeného komponentu (CollaborationComponent.js):

```
tb.push({
  id: 'locker',
  attributes: {class: 'fa fa-lock'},
  command: 'locking'
});
```

Tlačidlo obsahuje príkaz locking, ktorý pracuje s GrapesJS pričom je potrebné tento príkaz najprv pridať aj pri inicializácii GrapesJS inštancie (EditorComponent.js):

```
commands: {
  defaults: [ {
    id: 'locking',
    run: function(editor, sender){

    },
    stop: function(editor, sender){

    },
  },
],
},
```

Ak takto definujeme príkaz pomocou GrapesJS, tak je teraz možné na tento príkaz počúvať:

```
editor.on('run:locking', function () {})
```

Každý kolaborant má uložený zoznam uzamknutých komponentov (lockedComponents v CollaborationComponent.js) s ktorým sa počas všetkých procesov uzamykania a odomykania pracuje. Taktiež na servery (lockedComponents v EditorController.js) existuje rovnaký zoznam uzamknutých komponentov ale s tým rozdielom, že obsahuje všetky uzamknuté komponenty naprieč všetkými existujúcimi projektami aby sa po pripojení na konkrétny projekt poslali všetky aktuálne uzamknuté komponenty kolaborantovi.

## Uzamykanie

Po kliknutí na ikonu zámku obrázok 2 sa spustí príkaz run:locking. Po spustení príkazu sa pridajú do zoznamu uzamknutých elementov aktuálne zvolený komponent a všetky jeho deti, funkcia lockComponents, ktorá vracia zoznam všetkých zamknutých komponentov. V rámci funkcie lockComponents sa ešte zavolá funkcia removableParents, ktorá všetkým rodičovským komponentom pozastaví možnosť vymazania.

```
var components = self.lockComponents(sharedID);  
socket.emit('component:locked', components);
```

Zoznam uzamknutých komponentov je následne poslaný cez socket na server (EditorComponent.js), ktorý počúva na tento emit.

```
socket.on('component:locked', function (data) {  
  self.onComponentLocked(data, socket);  
});
```

Po zachytení na strane server sa zavolá funkcia onComponentLocked, ktorá pridá uzamknuté komponenty do zoznamu uzamknutých komponentov na strane server a ďalej broadcastne zoznam uzamknutých komponentov ďalším kolaborantom pripojeným na ten istý projekt.

```
socket.broadcast.to(socket.pageRoom).emit(  
  'component:locked', self.withUser(data, socket, "components")  
);
```

Následne všetci ostatní kolaboranti počúvajú na tento emit (CollaborationComponent.js).

```
socket.on('component:locked', function (data) {})
```

Po zachytení tohoto emitu sa každému kolaborantovi pridajú uzamknuté komponenty to jeho vlastného zoznamu uzamknutých komponentov. Následne je treba spraviť aj to aby sa uzamknuté komponenty priamo v editore nedali upravovať alebo s nimi hýbať pomocou funkcie remoteLocking.

Pridanie existujúcich uzamknutých komponentov po príchode kolaboranta zabezpečuje odoslanie emitu zo strany server.

```
socket.emit(  
  'collaborators:set_locked_components', self.lockedComponents[socket.pageRoom]  
);
```

A na strane klienta sa tento emit zachytí.

```
socket.on('collaborators:set_locked_components', function (data) {})
```

## Odomykanie

Proces odomykania je v podstate rovnaký ako proces uzamykania. Kolaborant spustí funkciu unlockComponents (CollaborationComponent.js), čím odstráni zo svojho zoznamu odomknuté komponenty a tento zoznam odošle soketom na server.

```
var components = self.unlockComponents(sharedID);  
socket.emit('component:unlocked', components);
```

Zoznam odomknutých komponentov je následne na servery odstránený zo zoznamu uzamknutých komponentov na servery (lockedComponents v EditorController.js). Server počúva na component:unlocked pričom následne zavolá funkciu onComponentUnlocked.

```
socket.on('component:unlocked', function (data) {  
  self.onComponentUnlocked(data, socket);  
});
```

Vo funkcii onComponentUnlocked server broadcastne ostatným kolaborantom zoznam odomknutých komponentov.

```
socket.broadcast.to(socket.pageRoom).emit(  
  'component:unlocked', data  
);
```

Následne všetci ostatní kolaboranti počúvajú na tento emit (CollaborationComponent.js).

```
socket.on('component:unlocked', function (data) {})
```

Po zachytení tohoto emitu sa každému kolaborantovi odstránia uzamknuté komponenty z jeho vlastného zoznamu uzamknutých komponentov. Následne je treba spraviť aj to aby sa odomknuté komponenty priamo v editore dali upravovať pomocou funkcie remoteLocking.



### 5.5.6 Testovanie

V rámci ladenia synchronizácie editora sme pristúpili k testovaniu jednotlivých preddefinovaných komponentov, pričom vzniklo mnoho testovacích scenárov, ktoré nájdete v priloženom súbore **testovanie\_komponentov.zip**

## 6 Záverečné testovanie

Záverečné testovanie prebiehalo v spolupráci s opozitným tímom, druhým tímom vedúceho projektu. K testovaniu sme vybrali 2 spôsobilých adeptov, s ktorými sme prechádzali predpripravené testovacie scenáre.

### 6.1 Testovacie scenáre

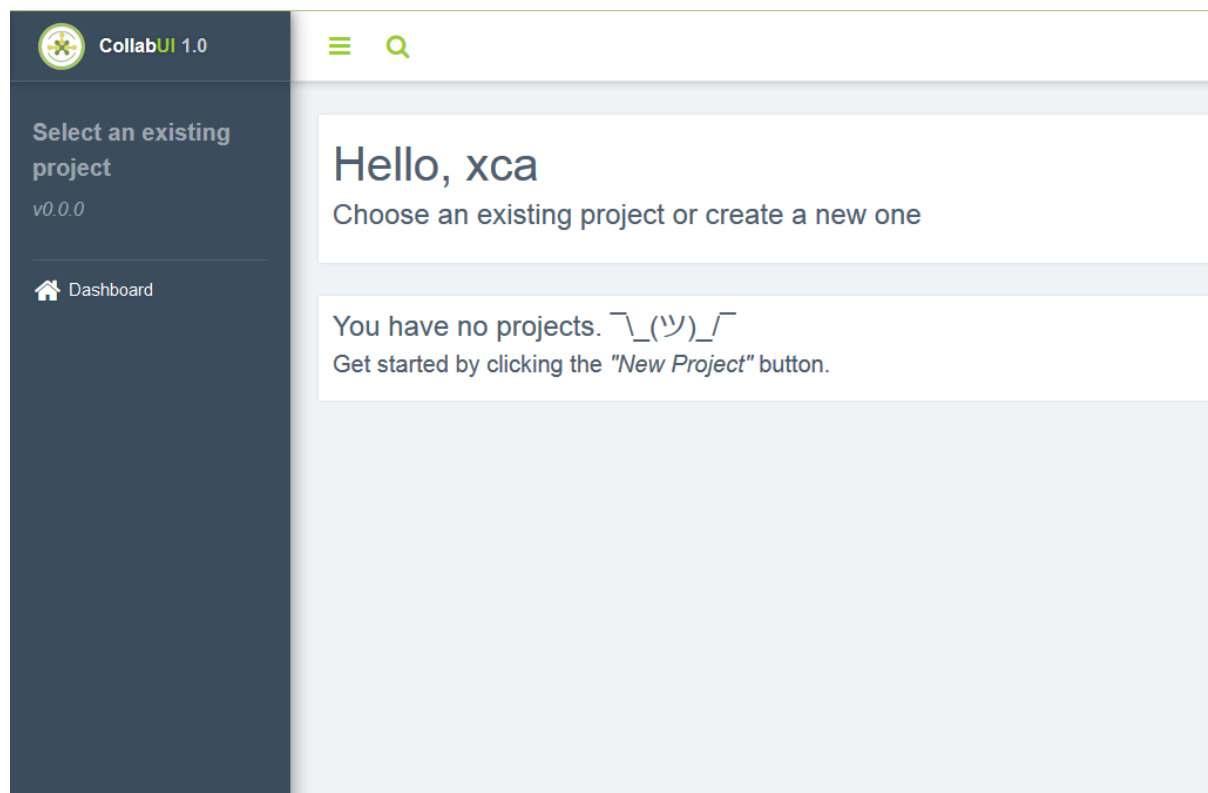
#### 6.1.1 Registrácia a prihlásenie

##### Zadanie Úlohy:

Ak chceme pracovať v tíme na našich prototypoch a mať miesto kam sa budú ukladať projekty, potrebujeme si registrovať nový účet. Cieľom tejto úlohy je úspešne si vytvoriť účet na stránke test.collabui.club a následne sa prihlásiť.

##### Testovací scenár:

1. Prejdite na stránku test.collabui.club a vyberte možnosť pre registráciu nového účtu
2. Vo formulári vyplňte potrebné údaje, pričom sila hesla bude aspoň *medium*
3. Prihláste sa pomocou novo vytvoreného účtu. Mali by ste byť presmerovaný na Dashboard.



Obrázok 54 - Očakávaný výstup pre scenár: Registrácia a prihlásenie

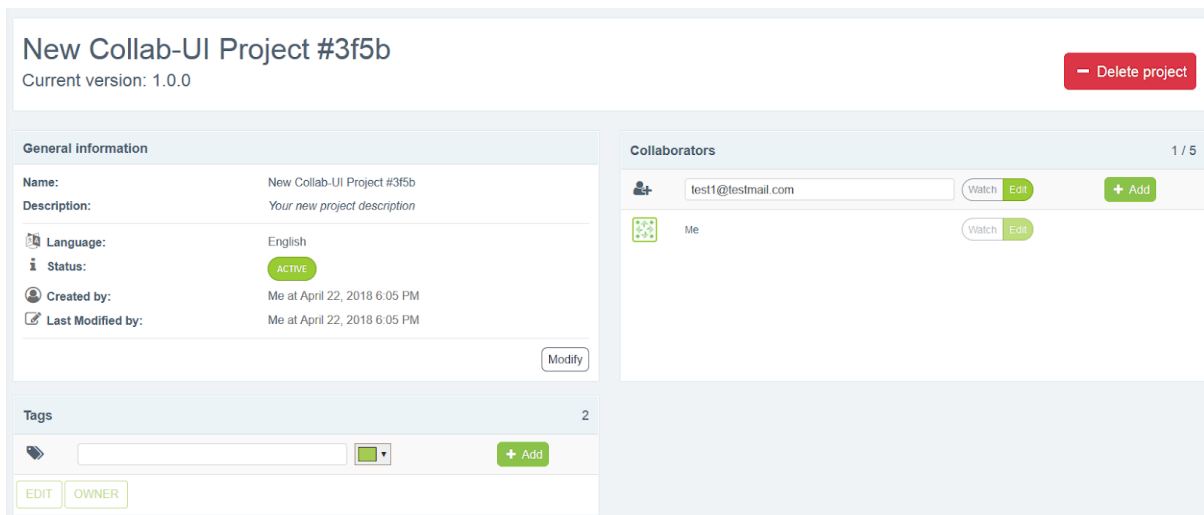
## 6.1.2 Správa projektov a Dashboard

### Zadanie Úlohy:

Po prihlásení do nášho účtu máme možnosť vytvárať nové a spravovať už vytvorené projekty. Cieľom tejto úlohy je vytvoriť nový projekt a otestovať funkcionality jeho jednotlivých prvkov. Konkrétne ide o časti *General information*, *Collaborators* a *Tags*.

### Testovací scenár:

1. Prihláste sa do svojho účtu alebo ak už ste prihlásený vytvorte nový projekt.
2. Prejdite na detail projektu
3. Zmeňte meno projektu na "Hello Collab" a priradte mu ľubovoľný opis.
4. Projektu priradte tag s názvom "test-tag" a dajte mu modrú farbu.
5. Pozvite do projektu druhého účastníka experimentu a priradte mu možnosť editovať projekt.
6. Ako posledný krok projekt vymažte.



Obrázok 55 – Očakávaný výstup pre scenár: Správu projektov a Dashboard

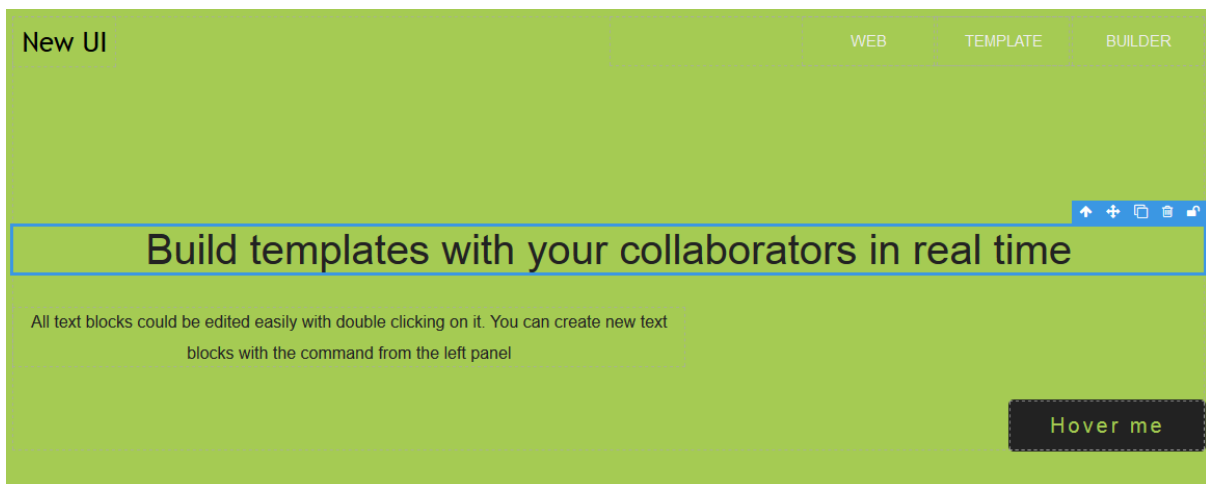
### 6.1.3 Práca s editorom #1

#### Zadanie Úlohy:

Po vytvorení nového projektu máme možnosť ísť do editora, kde prebieha tvorba prototypov webstránok. Cieľom prvej časti práce s prototypom bude zoznámiť sa s prostredím a komponentami.

#### Testovací scenár:

1. Prihláste sa do svojho účtu alebo ak už ste prihlásený vytvorte nový projekt.
2. Prejdite na editor projektu.
3. Na karte index vášho projektu zmeňte text "CollabUI" v navigačnom bare na "SoloUI". Farbu textu nastavte na modrú a zmeňte veľkosť fontu na 32px.
4. Zmenený text komponent zmažte.
5. Pridajte na jeho miesto nový text komponent.
6. Zmeňte zobrazenie na *inline-block* a zarovnajte ho doľava
7. Zmeňte farbu textu komponentu tak aby bol viditeľný.
8. Nastavte font na trebuchet MS a veľkosť fontu na 24px.



Obrázok 56 - Očakávaný výstup pre scenár: Práca s editorom #1

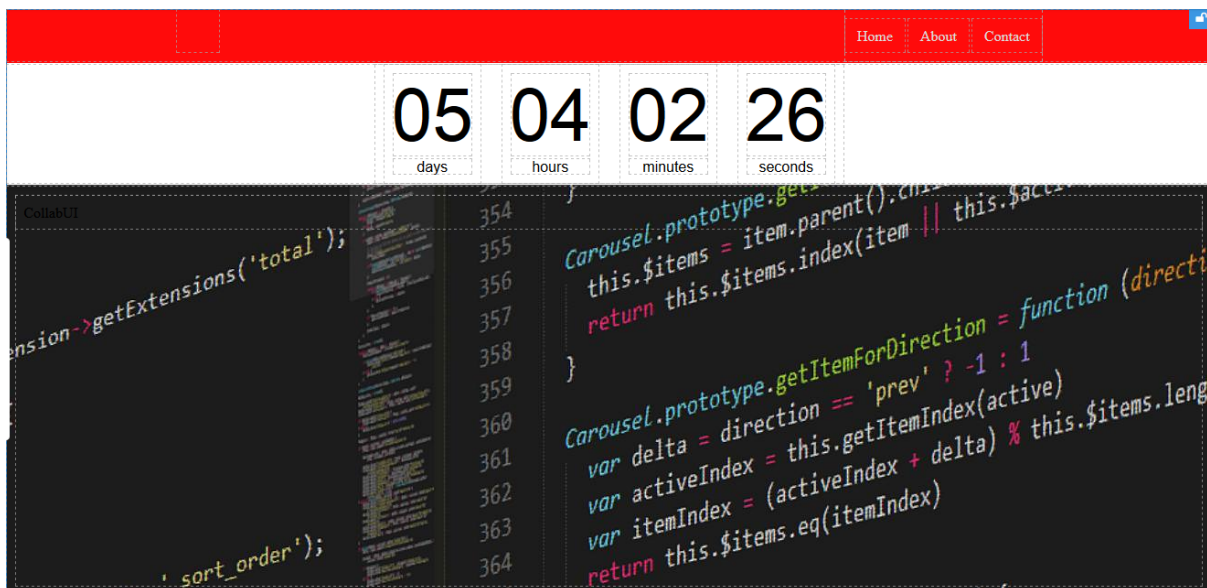
## 6.1.4 Práca s editorom #2

### Zadanie Úlohy:

Po vytvorení nového projektu máme možnosť ísť do editora, kde prebieha tvorba prototypov webstránok. Cieľom druhej časti práce s prototypom bude vyskúšať si prácu spolu s druhým účastníkom experimentu. Obaja budú pracovať na novej stránke inej ako index a v reálnom čase tvoriť jej obsah pomocou niekoľkých komponentov.

### Testovací scenár:

1. Jeden z účastníkov vytvorí nový projekt a prizve druhého s právom EDIT.
2. Vlastník aj kolaborant prejdú do editora.
3. Vlastník vytvorí novú kartu s názvom "HomeTest" a potom ju premenuje na "Home".
4. Kolaborant vytvorí novú kartu s názvom "ContactTest" a potom ju premenuje na "Contact".
5. Obaja účastníci pracujú oddelene na svojich kartách.
6. Pridajte komponent *Navbar* a zmeňte mu background color na červenú
7. Pridajte komponent *Countdown* a nastavte mu odpočítavanie do 28.4.2018
8. Pridajte komponent *1 Column* a vložte doňho text komponent.
9. Prepíšte text na CollabUI
10. Zmeňte background image komponentu 1 column a jeho výšku na 450px



Obrázok 57 - Očakávaný výstup pre scenár: Práca s editorom #2

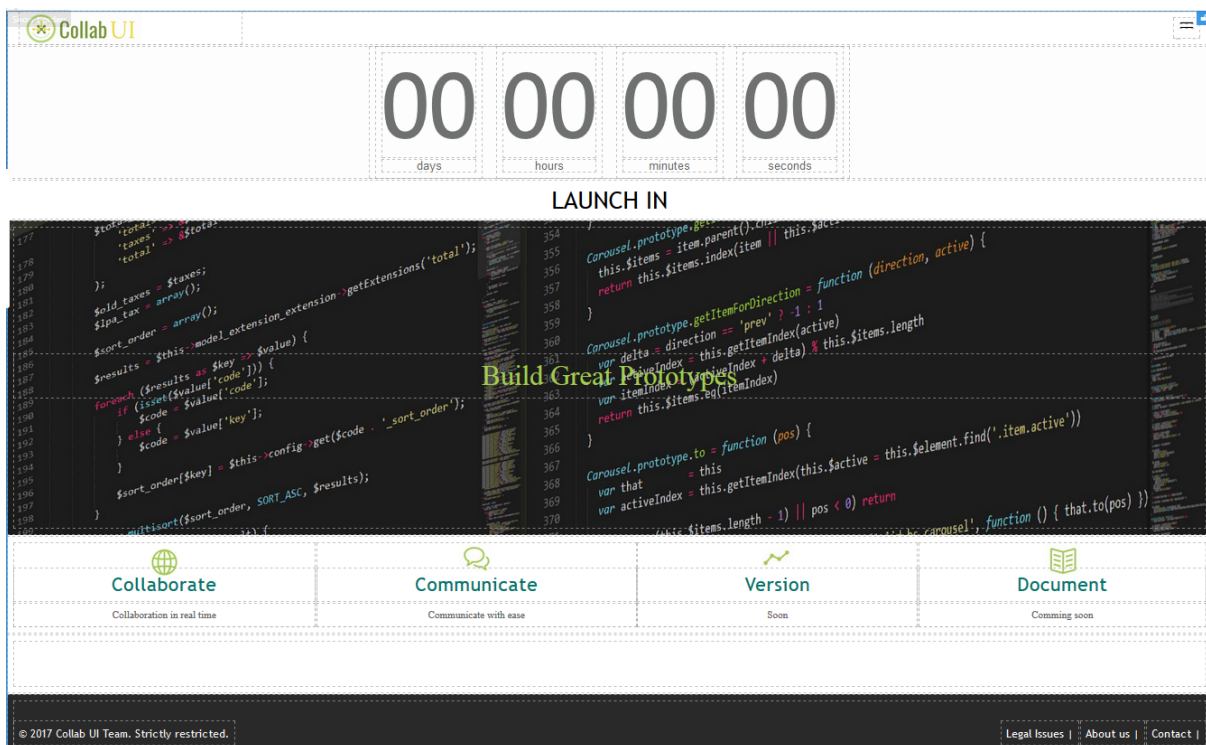
## 6.1.5 Práca s editorom #3

### Zadanie Úlohy:

Po vytvorení nového projektu máme možnosť ísť do editora, kde prebieha tvorba prototypov webstránok. Cieľom finálnej časti práce s prototypom je otestovať skombinovať znalosti z predošlých úloh a vytvoriť prototyp stránky podľa predlohy.

### Testovací scenár:

1. Jeden z účastníkov vytvorí projekt a prizve druhého s právom EDIT.
2. Vlastník projektu vytvorí novú kartu s názvom "CollabUI".
3. Obaja účastníci budú pracovať na tejto karte.
4. Podľa predlohy zobrazenej nižšie vytvoria obaja účastníci kolaboratívnym prístupom prototyp webovej stránky.



Obrázok 58 - Očakávaný výstup pre scenár: Práca s editorom #3

Pozn.: Použité komponenty v poslednej úlohe sú Navbar, Countdown, Columns a Text.

# Zdrojové médiá k vypracovaniu úloh



# Collab UI

```
177 // We have to put the totals in an array so that they pass
178 // by reference.
179 $totals = [];
180 $taxes => $totals;
181 $taxes = array();
182 $sort_order = array();
183
184 $results = $this->model_extension->getExtensions('total');
185
186 foreach ($results as $key => $value) {
187     if (isset($value['code'])) {
188         $code = $value['code'];
189     } else {
190         $code = $value['key'];
191     }
192     $sort_order[$key] = $this->config->get($code . '_sort_order');
193 }
194
195 array_multisort($sort_order, SORT_ASC, $results);
196
197 foreach ($results as $result) {
198     if (isset($result['code'])) {
199         $code = $result['code'];
200     } else {
201         $code = $result['key'];
202     }
203     if ($this->config->get($code . '_status')) {
204         $this->load->model('extension/total/' . $code);
205
206         // We have to put the totals in an array so that they pass
207         // by reference.
208         $this->{ 'model_extension_total_' . $code }->getTotal($
209             total_data);
210
211         if (empty($totals[count($totals) - 1]) && !isset($totals[
212             count($totals) - 1]['code'])) {
213             $totals[count($totals) - 1]['code'] = $code;
214         }
215
216         $tax_difference = 0;
217
218         foreach ($taxes as $tax_id => $value) {
219             if (isset($total_taxes[$tax_id])) {
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

## Build Great Prototypes

Just the right tool to make building prototypes in team as effortless and convenient as possible



## 6.2 Dotazník

Po vypracovaní testovacích scenárov sme našim respondentom poskytli dotazník, kde mali možnosť vyjadriť svoju spokojnosť.

**CollabUI feedback**

\*Povinné pole

**Aká bola odozva akcií v aplikácii? \***

1 2 3 4 5

veľmi pomalá      veľmi rýchla

**Ako by ste ohodnotili intuitívnosť rozhrania? \***

1 2 3 4 5

zložitá      intuitívne

**Myslíte si, že pri návrhu prototypov by vám kolaborácia ušetrila čas? \***

1 2 3 4 5

určite nie      určite áno

**Ako by ste hodnotili celkový dizajn aplikácie? \***

1 2 3 4 5

Nič moc      Parádica

**V čom vidíte najväčšie nedostatky aplikácie, čo by ste zlepšili?**

Vaše odpoveď

**ODESLAŤ**

Nikdy přes Formuláře Google neposílejte hesla.

Obsah není vytvořen ani schválen Googlem. Nahlásit zneužití - Smluvní podmínky služby - Další smluvní podmínky

Google Formuláře

Obrázok 59 - Dotazník po realizácii testovania

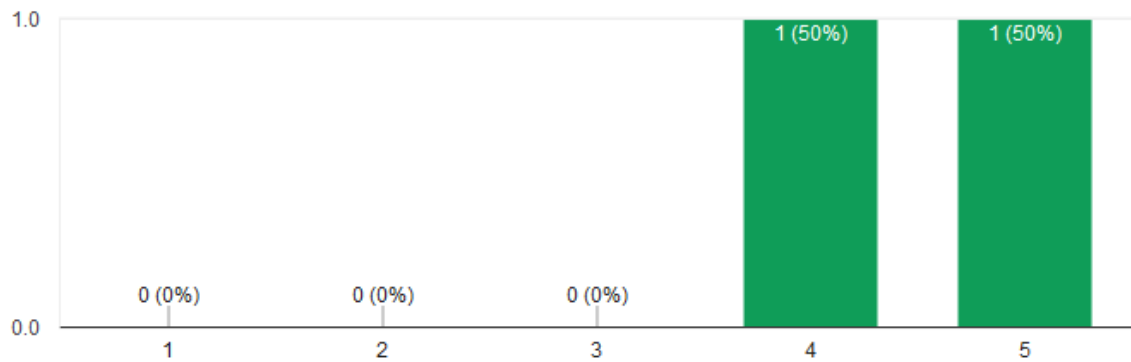


### 6.3 Výstup z testovania

Hodnotenie aspektov sa pohybuje v intervale  $<1,5>$ , pričom 1 je najhoršie a 5 najlepšie.

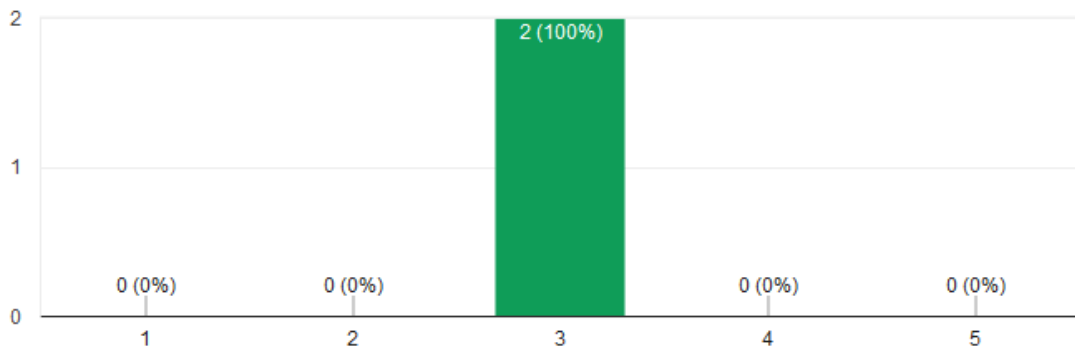
Aká bola odozva akcií v aplikácii?

2 responses



Ako by ste ohodnotili intuitívnosť rozhrania?

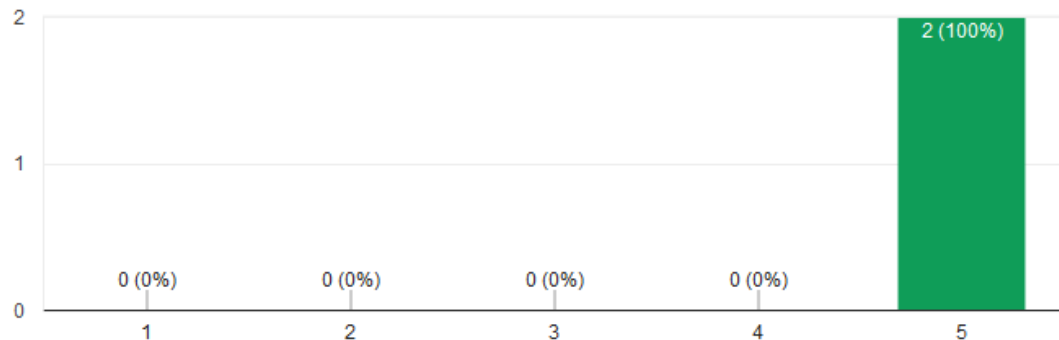
2 responses



Myslíte si, že pri návrhu prototypov by vám kolaborácia ušetrila čas?



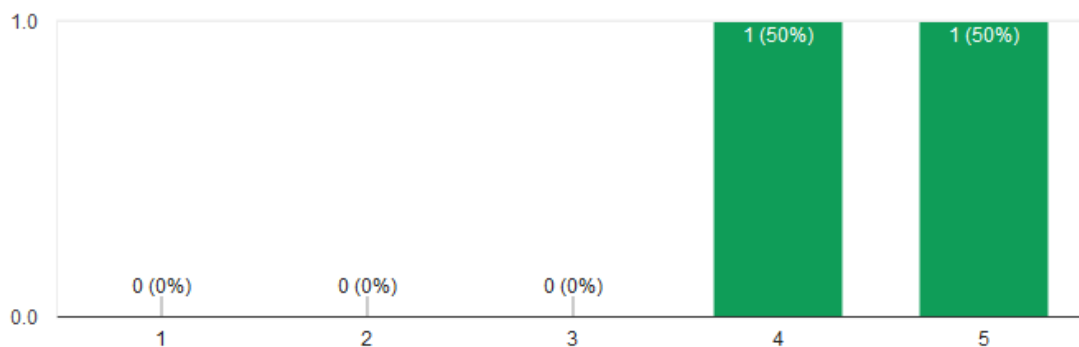
2 responses



Ako by ste hodnotili celkový dizajn aplikácie?



2 responses



## Feedback

- Možnosť meniť šírku a výšku komponentov v editore pomocou myšky
- Ak klikneme na nejaký komponent v editore, bolo by fajn aby sa zvýraznil aj vľavo v strome komponentov
- Pri niektorých CSS nastaveniach sú neviditeľné tlačidlá, konkrétne ikony „+“ pri nastavovaní background layerov
- Zvýrazniť uzamknutý element
- Chýbajú tooltips pri ikonkách v paneloch
- Stránka by sa mala dať premenovať v rámci karty
- Panel pre prototypy by mal mať inú ikonku ako spinku
- Ikony v paneloch by mali byť výraznejšie
- V toolbare na ikonách keď máme označený komponent v editore chýba tooltip

## 7 Záver

Na záver tohto dokumentu by sme poznamenali skutočnosť, že sme na projekte pracovali kontinuálne a podarilo sa nám zimplementovať 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 systému 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 zaznamenávajú a prenášajú všetky potrebné náležitosti. Editor dovoľuje taktiež prácu na viacerých prototypoch, stránkach v jednom čase prostredníctvom rôznych kolaborantov. Takto vytvorené prototypy je možné premenovať, vymazať, meniť, ba prepájať medzi sebou prostredníctvom odkazov. V prípade konfliktov medzi kolaborantmi je možné pristúpiť k uzamknutiu komponentu alebo skupiny komponentov a následnému odomknutiu. Samotná kolaborácia je odladená pre všetky preddefinované komponenty podporného nástroja.

Pôvodným cieľom projektu bolo na koniec vývojového obdobia vyvinúť webovú aplikáciu na produkčnej úrovni, pripravenú na nasadenie. Avšak po hĺbkovej analýze problémovej oblasti sme pristúpili k úprave zamýšľaného plánu a k stanovaniu nového cieľu a to dokončiť a odladiť kolaboráciu v editore. Osobne si myslíme, že sme novo vytýčený cieľ dosiahli.

Vyvinutá webová aplikácia prešla testami opozitného tímu, ktorí poskytli cenný feedback, ktorý môže byť v budúcnosti zapracovaný nasledovníkmi nášho tímu.

Náš tím sa zúčastnil i súťaže TP Cup, kde sa nám podarilo dostať do semifinále. V čase písania dokumentácie prebiehajú prípravy na ďalšie kolo, do začiatku ktorej sme sa rozhodli obohatiť editor o možnosť exportovania projektu. Samotná funkcionálna exportovania už nebude zdokumentovaná, kvôli začiatku vývoja až po odovzdaní dokumentácie.

# **Dokumentácia k riadeniu projektu**

Tím 24

# Obsah

<b>1</b>	<b>Úvod</b>	<b>4</b>
<b>2</b>	<b>Role členov tímu a podiel práce</b>	<b>5</b>
2.1	Bc. Adrián Nagy	5
2.2	Bc. Lukáš Vrba	5
2.3	Bc. Peter Písecký	5
2.4	Bc. Tomáš Mňačko	5
2.5	Bc. Ján Kleň	6
2.6	Bc. Miloslav Smetana	6
2.7	Bc. Michal Melúch	6
<b>3</b>	<b>Aplikácie manažmentov a použité metodiky</b>	<b>7</b>
3.1	Manažment vývoja	7
3.2	Manažment úloh	7
3.3	Manažment dizajnu a UX	8
3.4	Manažment verziovania	8
3.5	Manažment testovania	8
3.6	Manažment dokumentácie	8
3.7	Manažment podporných nástrojov a komunikácie	9
3.8	Manažment kvality kódu	9
3.9	Manažment webu tímu a produkčného prostredia	10
<b>4</b>	<b>Povinnosti a požiadavky členov tímu</b>	<b>11</b>
4.1	Bc. Peter Písecký	11
4.1.1	Povinnosti a požiadavky na tím	11
4.2	Bc. Michal Melúch	11
4.2.1	Povinnosti a požiadavky na tím	11
4.3	Bc. Ján Kleň	12
4.3.1	Povinnosti a požiadavky na tím	12
4.4	Bc. Lukáš Vrba	13
4.4.1	Povinnosti a požiadavky na tím	13
4.5	Bc. Tomáš Mňačko	13
4.5.1	Povinnosti a požiadavky na tím	13
4.6	Bc. Miloslav Smetana	14
4.6.1	Povinnosti a požiadavky na tím	14
4.7	Bc. Adrián Nagy	14
4.7.1	Povinnosti a požiadavky na tím	14
<b>5</b>	<b>Sumarizácie šprintov</b>	<b>16</b>
5.1	Zimný semester	16
5.1.1	Šprint1 (3.10.2017 - 18.10.2017) Birell nealko	16
5.1.2	Zhodnotenie šprintu	19
5.1.3	Šprint2 (20.10.2017 - 3.11.2017) StrongBow_Cider	20
5.1.4	Zhodnotenie šprintu	23
5.1.5	Šprint3 (3.11.2017 - 17.11.2017) Plzeň	24
5.1.6	Zhodnotenie šprintu	27
5.1.7	Šprint4 (17.11.2017 - 1.12.2017) Jack_Daniels	28
5.1.8	Zhodnotenie šprintu	31
5.1.9	Šprint5 (1.12.2017 - 15.12.2017) Jack_Daniels#2	32
5.1.10	Zhodnotenie šprintu	35
5.1.11	Medzišprint (16.12.2017 - 17.2.2018) Nešprint	36
5.2	Letný semester	37

5.2.1	Šprint1 (18.2.2018 - 4.3.2018) TatraTea_22.....	37
5.2.2	Zhodnotenie šprintu .....	39
5.2.3	Šprint2 (4.3.2018 - 18.3.2018) TatraTea_32.....	40
5.2.4	Zhodnotenie šprintu .....	42
5.2.5	Šprint3 (18.3.2018 - 2.4.2018) TatraTea_42.....	43
5.2.6	Zhodnotenie šprintu .....	45
5.2.7	Šprint4 (2.4.2018 - 16.4.2018) TatraTea_52.....	46
5.2.8	Zhodnotenie šprintu .....	48
5.2.9	Šprint5 (16.4.2018 - 30.4.2018) TatraTea_62.....	49
5.2.10	Zhodnotenie šprintu.....	51
<b>6</b>	<b>Globálna retrospektíva .....</b>	<b>52</b>
6.1	Zimný semester .....	52
6.2	Letný semester .....	53
<b>7</b>	<b>Záver .....</b>	<b>54</b>
<b>1</b>	<b>Príloha A - Metodiky .....</b>	<b>55</b>

## Obsah Obrázkov

Obrázok 1	- Burndown chart pre prvý šprint.....	17
Obrázok 2	- Retrospektíva prvého šprintu.....	18
Obrázok 3	- Logo prvého šprintu .....	19
Obrázok 4	- Burndown chart pre druhý šprint .....	21
Obrázok 5	- Retrospektíva k druhému šprintu.....	22
Obrázok 6	- Logo druhého šprintu.....	23
Obrázok 7	- Burndown chart pre tretí šprint .....	25
Obrázok 8	- Retrospektíva pre tretí šprint .....	26
Obrázok 9	- Logo tretieho šprintu .....	27
Obrázok 10	- Burndown chart pre štvrtý šprint.....	29
Obrázok 11	- Retrospektíva pre štvrtý šprint.....	30
Obrázok 12	- Logo štvrtého šprintu .....	31
Obrázok 13	- Burndown chart pre piaty šprint.....	33
Obrázok 14	- Retrospektíva pre piaty šprint.....	34
Obrázok 15	- Logo piateho šprintu .....	35
Obrázok 16	- Retrospektíva prvého šprintu v letnom semestri .....	38
Obrázok 17	- Retrospektíva druhého šprintu v letnom semestri .....	41
Obrázok 18	- Retrospektíva tretieho šprintu v letnom semestri .....	44
Obrázok 19	- Retrospektíva štvrtého šprintu v letnom semestri.....	47
Obrázok 20	- Retrospektíva piateho šprintu v letnom semestri.....	50

# 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 načrtáme 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
  - Vedúci tímu
  - Scrum master

### 2.2 Bc. Lukáš Vrba

- Zimný semester
  - Dokumentrista
    - Dokument riadenia k tímovému projektu
    - Dokument k inžinierskemu dielu
    - Moduly aplikácie
  - Dodatokový programátor
  - Plánovač
- Letný semester
  - Plánovač
  - Dokumentarista

### 2.3 Bc. Peter Písecký

- Zimný semester
  - Kvalitár
  - Code review master pre php
  - Backend programátor
- Letný semester
  - Softvérový architekt
  - Kvalitár

### 2.4 Bc. Tomáš Mňáčko

- Zimný semester
  - Hlavný tester
  - Časť dokumentácie pre testovanie



- Code review master pre javascript
- Letný semester
  - Hlavný tester
  - Hlas (vo videu)

## **2.5 Bc. Ján Kleň**

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

## **2.6 Bc. Miloslav Smetana**

- Zimný semester
  - Manažér komunikácie
  - Backend programátor
  - Validácie používateľa a projektov
- Letný semester
  - Moderátor testovania
  - Manažment integrácie a nasadenia produktu

## **2.7 Bc. Michal Melúch**

- Zimný semester
  - Frontendista (UX)
  - Dizajnér
  - Databázový master
- Letný semester
  - Manažér komunikácie
  - Dizajnér

## 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 prídelenú 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<sup>1</sup>, ktorý využívame prostredníctvom školskej licencie zdarma. Úlohy rozdelujeme pomocou 3 abstraktných úrovní:

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

ScrumDesk sme prostredníctvom webhooku premostili s tímovým nástrojom pre komunikáciu<sup>2</sup>, na ktorý prichádzajú notifikácie po každej zmene stavu, úprave zadania, resp. atribútov úlohy, či komentára. Po naplánovaní úloh sú všetky tasky prídelené 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**.

---

<sup>1</sup> ScrumDesk

<sup>2</sup> 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 dizajnu 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 dôležité 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 organizácii 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<sup>3</sup>, ktorý je prostredníctvom webhooku premostený s tímovým slackom, na ktorý prichádzajú notifikácie po každom pushnutí nového commitu. K verziovaniu používame voľne dostupnú aplikáciu<sup>4</sup>, 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 vyvíjame aplikáciu je potrebné okrem jej testovania a organizácie kvality kódu vytvoriť aj kvalitnú dokumentáciu. V rámci dokumentácie sa pri každom feature rieši

---

<sup>3</sup> Bitbucket

<sup>4</sup> SourceTree

analýza problémovej oblasti ako aj zameranie používateľa, návrh riešenia ako aj samotná implementácia v podobe technickej dokumentácie. V prípade potreby opisujeme jednotlivé implementačné 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 spísať 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 naše 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 konvencií. Je to práve z dôvodu aby sa neznižovala kvalita kódu, čitateľnosť a aby sa predišlo nezrozumiteľnostiam pri vyvíjaní ostatných členov tímu. Inakšie povedané aby sa ostatní členovia tímu nestrácali v kóde. Na zabezpečenie vyššej kvality kódu bolo nutné začať vykonávať striktnjšie 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 spísaný 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 tímu. 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 navrhnuť 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

Počiatocne 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 vid'. 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 prehľadný 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

Počiatocne 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 dozeráť 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 zmenené 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

Počiatocne 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 dohľadanie 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

Počiatocne 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

Počiatocne 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 seriózný 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 konfrontácií.

**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 kvalítárom 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 pracovať. 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 realizujeme 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 funkcií 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 technológií pre budúcu integráciu.

Celkový počet úloh v šprinte: 12

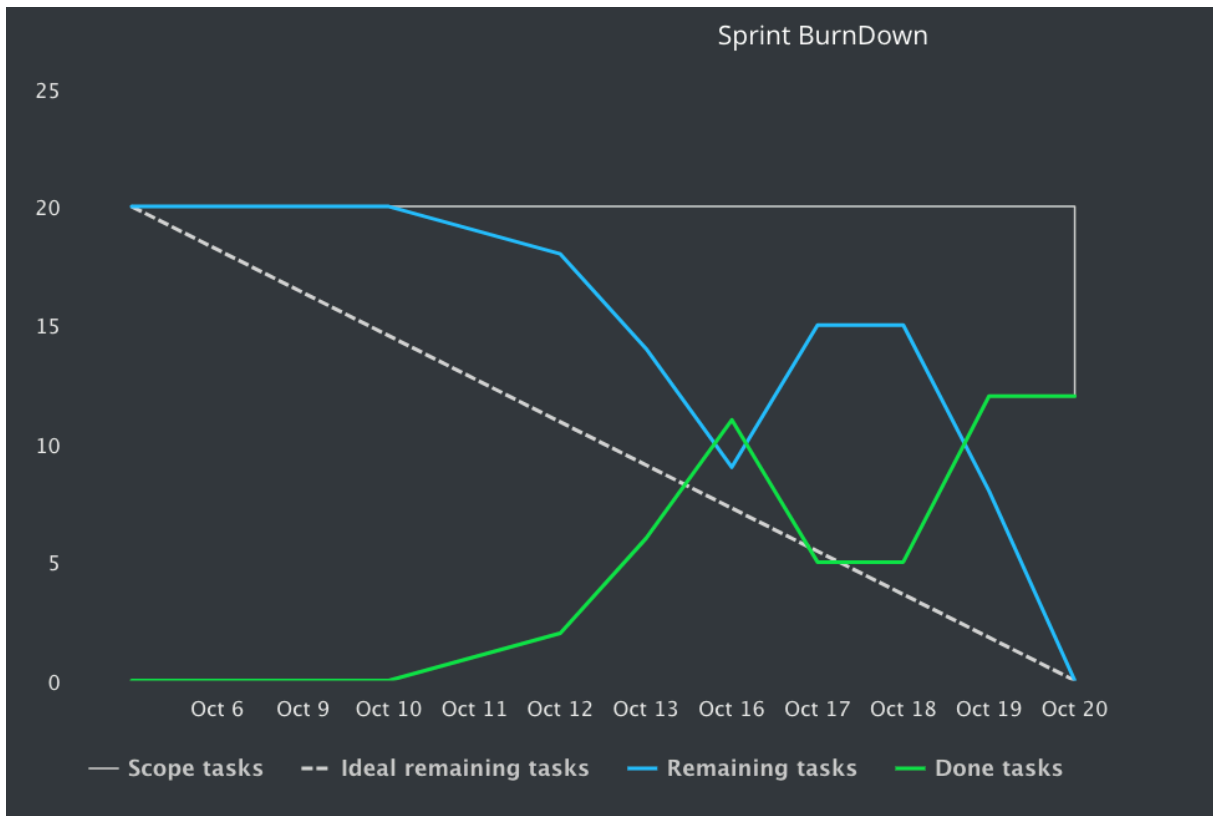
Každý task obsahuje dokumentáciu a testovanie.

##### 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

The image shows a digital Scrum retrospective board with seven sticky notes arranged in a grid. Each note has a colored header, a title, a rating (four stars), a close button (x), and a count (0). The notes are as follows:

- Top Left (Green):** Title: **glad**. Content: Pohli sme sa. Nedokončili sme síce všetko podľa plánov, ale to nevadí. Author: Birell\_Nealko. Label: ToDo.
- Top Middle (Yellow):** Title: **sad**. Content: Nemám na projekt, tolko času, koľko by si vyžadoval :/. Author: Birell\_Nealko. Label: ToDo.
- Top Right (Pink):** Title: **mad**. Content: Domyslenie vecí, viac komunikácie. Author: Birell\_Nealko. Label: ToDo.
- Middle Left (Green):** Title: **glad**. Content: -celkom rychly nastup na scrum  
-dobra komunikacia v time  
-kazdy robil co si potiahol. Author: Birell\_Nealko. Label: ToDo.
- Middle Middle (Yellow):** Title: **sad**. Content: Presnejšie definovanie úloh(taskov). Čo a ako to urobiť. Author: Birell\_Nealko. Label: ToDo.
- Middle Right (Pink):** Title: **BabyRage**. Content: -lepšie popísať jednotlivé tasky (niekedy nechápem že čo)  
-byť konzistentnejší v robení UI, potom to x-krát prerábame  
-trochu lepšie popísať testovanie, nech to človek nemusí študovať deň  
-zadefinovať nejaké mantinely v UI, spravíme niečo ale ak sa tam niečo nezdá treba to prerobiť, chce to diskusu, napr urobiť svetlejšie bannery = zlá viditeľnosť na text  
-lepšie si rozdeliť vykonávanie úloh, robí tam reálne 5 ľudí. Author: Birell\_Nealko. Label: ToDo.
- Bottom Middle (Yellow):** Title: **sad**. Content: Skúsif rozumnejšie rozbranchovať projekt - backendové veci spolu, frontendové veci spolu.. aby sme nemuseli tolko cherrypickovať  
Nahodnotiť stories na 5-6 ľudí namiesto 7  
Pri tvorbe obrazoviek najprv poriadne doladiť jednu a potom na jej obraz robif ostatné. Author: Birell\_Nealko. Label: ToDo.

Obrázok 2 - Retrospektíva prvého šprintu

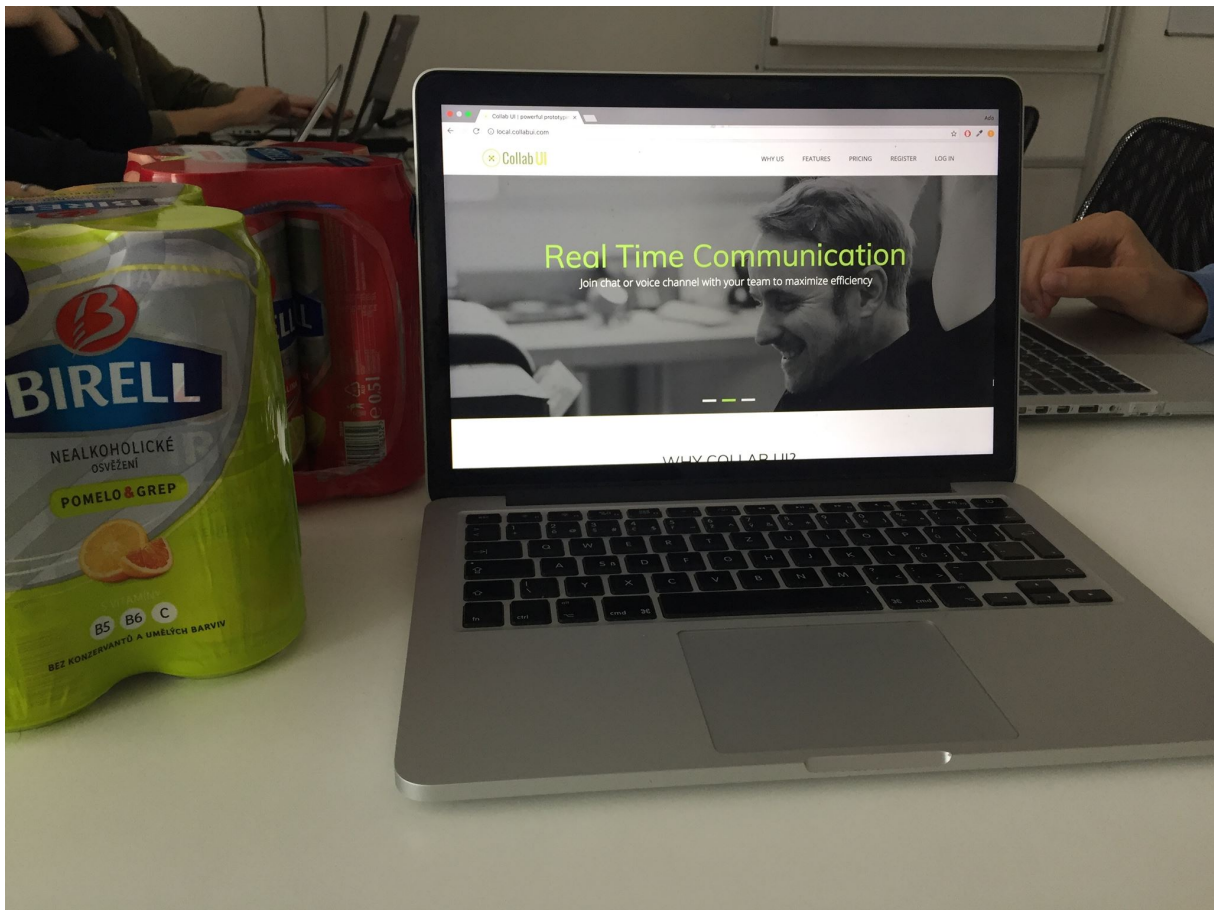
## 5.1.2 Zhodnotenie šprintu

Ako sme už z burndown chartu mohli zistiť zo začiatku nám chvíľu trvalo kým sme sa zosúladi. 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, navrhnuť a samozrejme aj implementovať spôsob akým používateľ bude môcť vytvárať vlastné projekty a taktiež akým štýlom bude môcť pridávať 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:

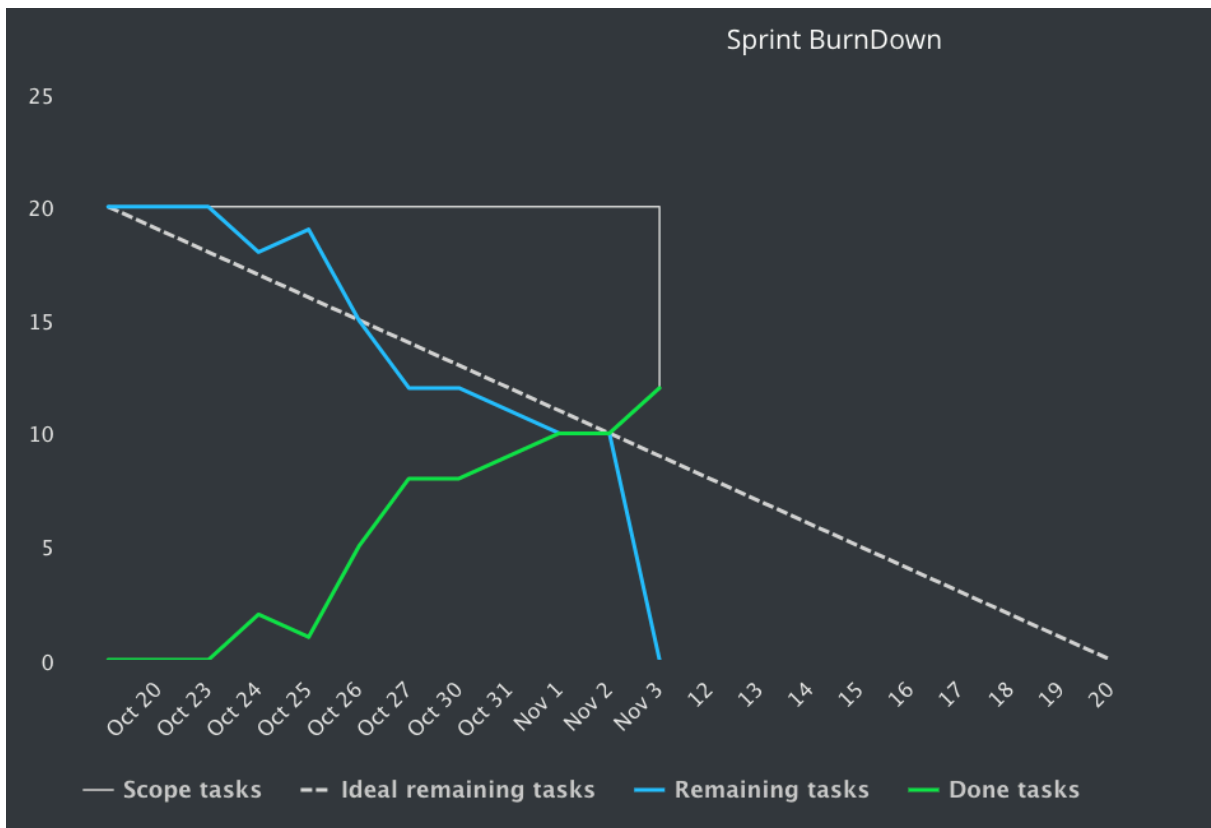
- Zabudnuté a nové heslo
- Vytváranie projektov a pridávanie kolaborantov

#### 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

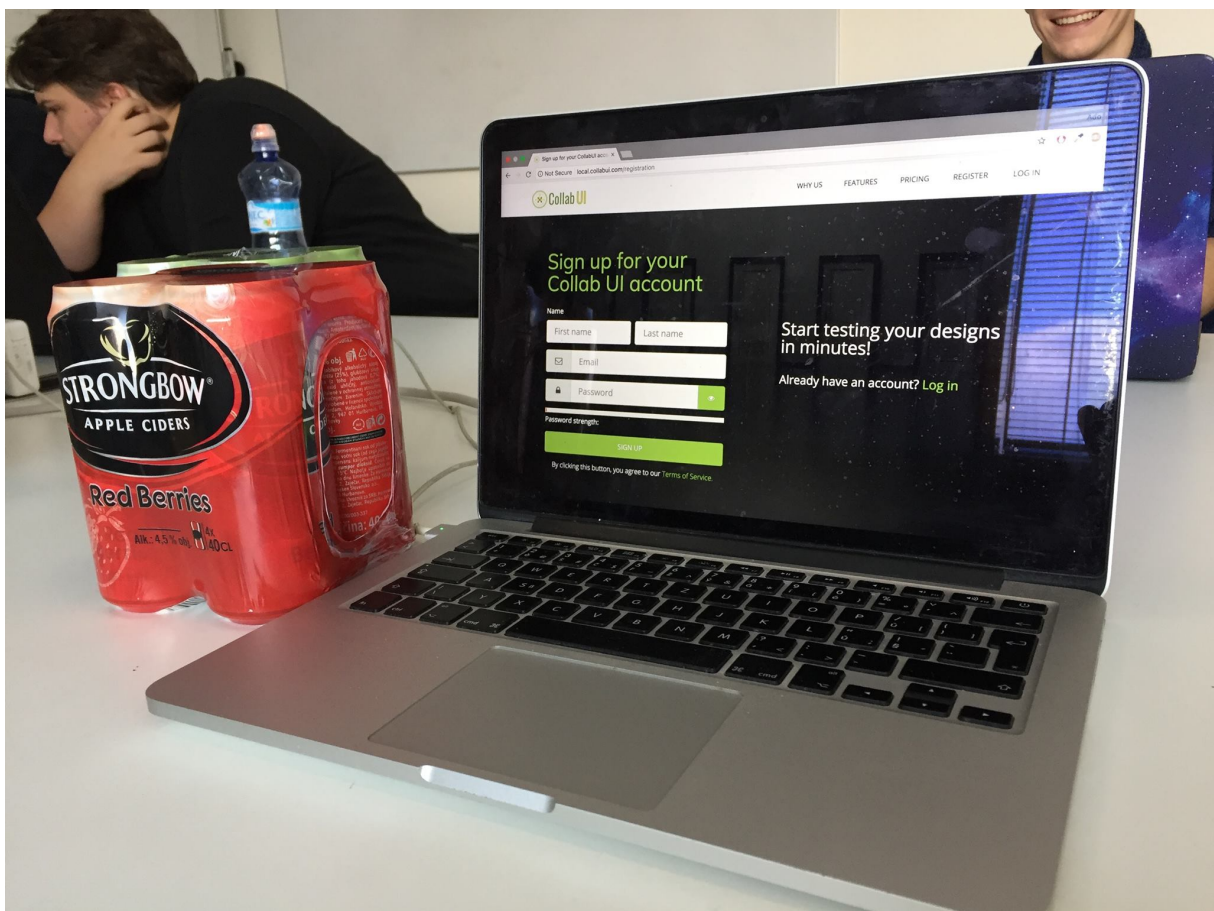
<p><b>glad</b> ★★★★★ ✕ 0</p> <p>občas to ide no</p> <p>StrongBow_Cider    ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 0</p> <p>3 poslali dokumentáciu k tomu čo robili</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Kódimе štyria Na Slacku sa bavíme štyria</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Zas sme nestihli k čomu sme sa commitli ;(</p> <p>StrongBow_Cider    ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>Nerovnomerné prerozdelenie úloh v tíme</p> <p>StrongBow_Cider    ToDo</p>
<p><b>glad</b> ★★★★★ ✕ 3</p> <p>Čo je hotové je spravené celkom dobre a použiteľne - netreba sa k tomu toľko vracat ako minulý šprint</p> <p>StrongBow_Cider    ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 0</p> <p>Snažíme sa</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Ešte by sme mali popracovať na špecifikácii jednotlivých taskov. Častokrát obsahujú len kľúčové slová, alebo slovné spojenia a pokiaľ človek nebol na šprinte, alebo si nepamätá, tak sa ťažko určuje čo a ako.</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Nízka úroveň motivácie v tíme.</p> <p>StrongBow_Cider    ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 4</p> <p>- ScumDesk</p> <p>StrongBow_Cider    ToDo</p>
<p><b>glad</b> ★★★★★ ✕ 4</p> <p>stránka vyzerá zatiaľ super</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 3</p> <p>Stále sa nevyznám v projekte tak ako by som chcel.</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Vedúci tímu je tvrdohlavý</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 3</p> <p>-chalani to potiahli za nás -je mi ľúto, že som sa k tomu viacej nedostal -mierna demotivácia</p> <p>StrongBow_Cider    ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 3</p> <p>EASY</p> <p>StrongBow_Cider    ToDo</p>
<p><b>glad</b> ★★★★★ ✕ 0</p> <p>Peto a Mišo makajú ako fretky !</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 6</p> <p>veľa papierikov k retrospektíve, nezmesť sa do dokumentácie :(</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 1</p> <p>Niekedy strácame veľa času na drobnostiach.</p> <p>StrongBow_Cider    ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Použitie nových pluginov, nevieme do čoho ideme, obmedzenia</p> <p>StrongBow_Cider    ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>Niekedy ponocujem kvôli niečomu, čo si ostatní vyše týždňa ani nepullnú lebo projekt proste nesledujú</p> <p>StrongBow_Cider    ToDo</p>

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 riešiť ďalšie úlohy v poradí. Na druhej 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 zintegrovať 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

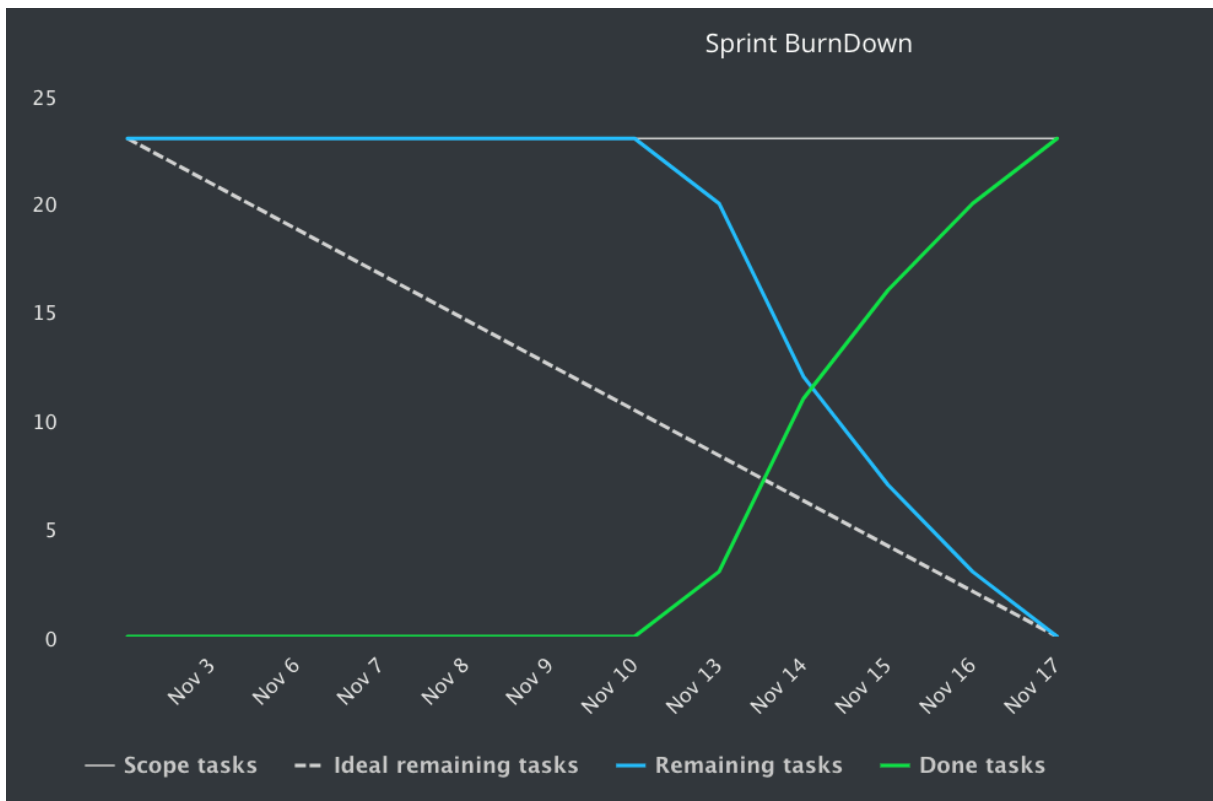
Každý task obsahuje dokumentáciu a testovanie.

#### 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

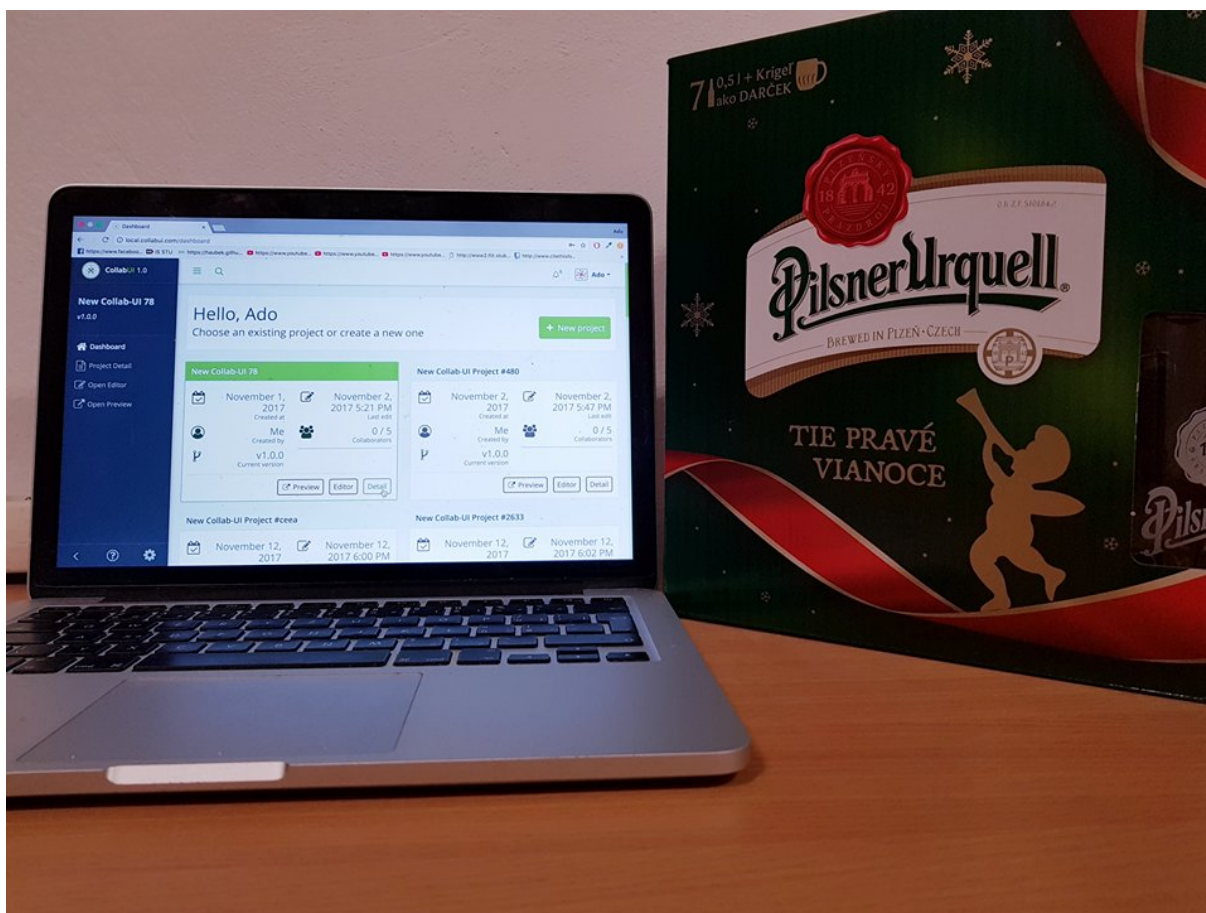
<b>glad</b> ★★★★★ ✖ 1 Posledné dni tohto sprintu sme poprvykrát pracovali ako tím Plzeň      ToDo	<b>glad</b> ★★★★★ ✖ 0 KnockoutJS má dobrú dokumentáciu Plzeň      ToDo	<b>sad</b> ★★★★★ ✖ 0 Trochu chýba povedomie o projekte - "ved už máme skoro všetko done". Kto na tom robil vie, že zďaleka nemáme. Plzeň      ToDo
<b>glad</b> ★★★★★ ✖ 0 Aďo nestráca trpezlivosť Plzeň      ToDo	<b>glad</b> ★★★★★ ✖ 0 Všetci členovia tímu sú ochotní pomáhať slabším členom Plzeň      ToDo	<b>sad</b> ★★★★★ ✖ 0 Ak by sme robili celý čas tak ako v posledné dni, už sme mohli byť skoro hotoví Plzeň      ToDo
<b>glad</b> ★★★★★ ✖ 0 Prezentačná vrstva aplikácie vyzerá viac než dobre Plzeň      ToDo	<b>glad</b> ★★★★★ ✖ 0 FE kód je prehľadný a ľahko pochopiteľný Plzeň      ToDo	<b>sad</b> ★★★★★ ✖ 0 BE kód vyžaduje refaktoring, zjednotenie Plzeň      ToDo
<b>glad</b> ★★★★★ ✖ 0 Každý priložil ruku k dielu, výsledok je na úrovni! Plzeň      ToDo	<b>glad</b> ★★★★★ ✖ 0 Pekná vizuálna prezentácia, taktiež loader Plzeň      ToDo	
<b>glad</b> ★★★★★ ✖ 0 pomoc pri vytváraní dokumentácie od ostatných členov pre budúce odovzdanie Plzeň      ToDo	<b>glad</b> ★★★★★ ✖ 0 aoweb Plzeň      ToDo	
<b>glad</b> ★★★★★ ✖ 0 Už napredujeme Plzeň      ToDo		

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 počtu 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 "Pizeň".



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

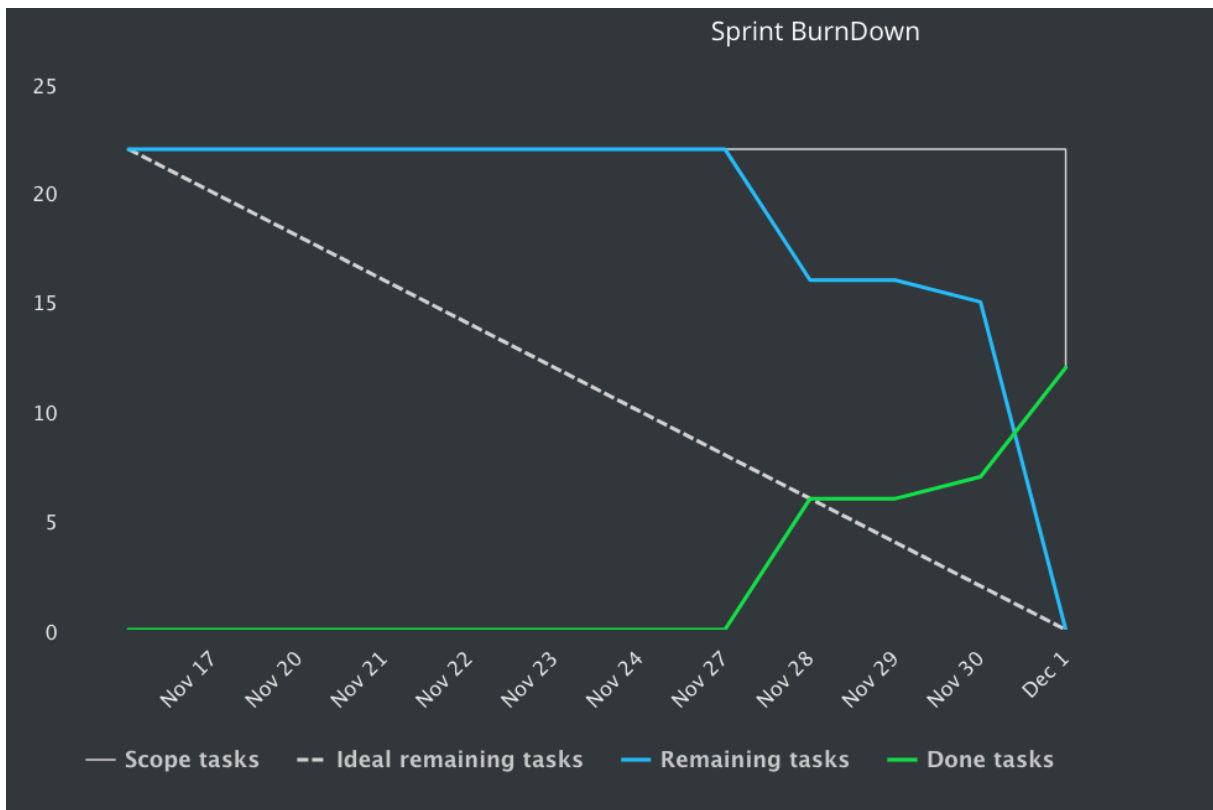
Každá feature obsahuje dokumentáciu a testovanie.

#### 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

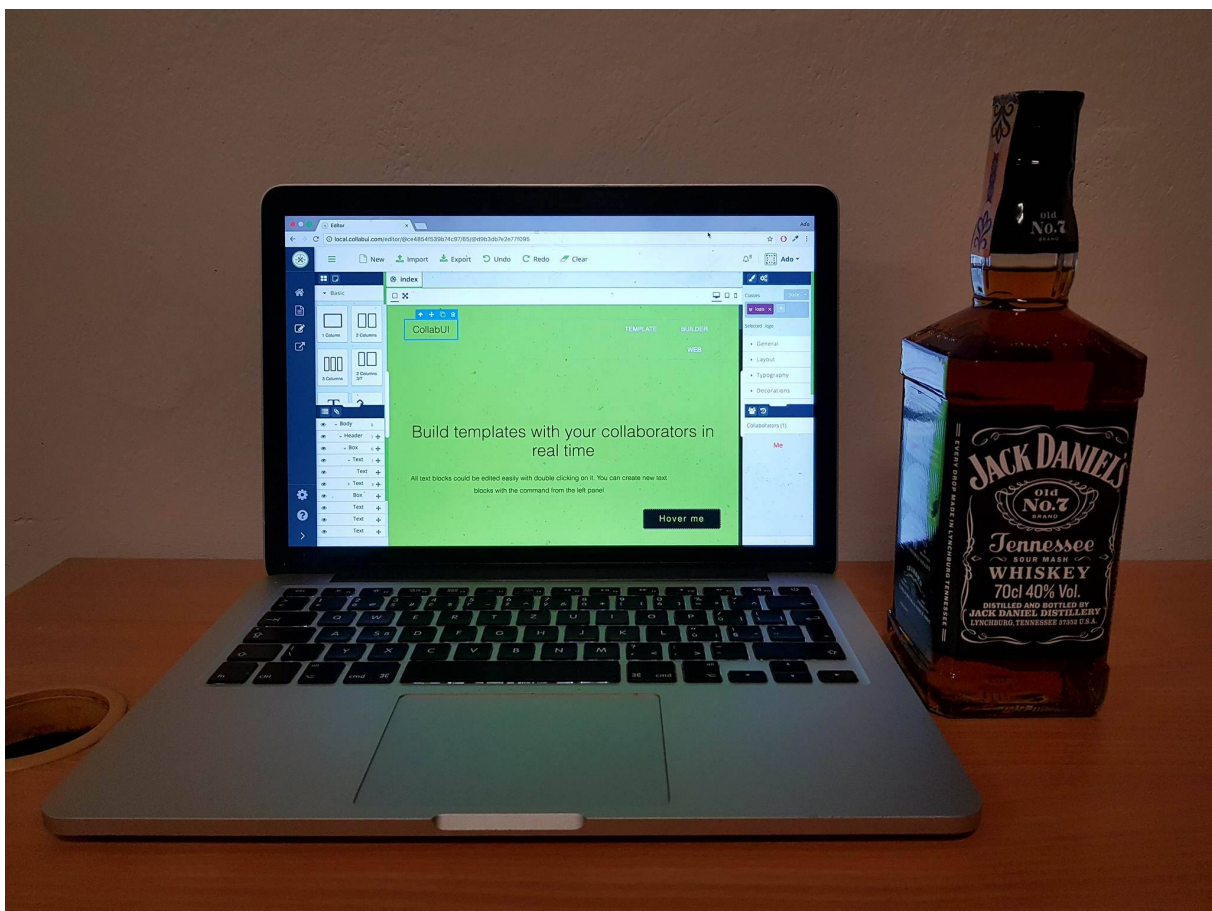
<p><b>glad</b> ★★☆☆ ✕ 0</p> <p>organizacia panelov, oproti povodnemu planu</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>BabyRage</b> ★★☆☆ ✕ 3</p> <p>rozbehali sme node a socket.io</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 4</p> <p>GrapesJS s nami nestiha</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 9</p> <p>Tlačia ma ďalšie zadania Aj mňa +1</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>mad</b> ★☆☆☆ ✕ 4</p> <p>GrapesJS API a jeho celková zrozumiteľnosť je nanič</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>mad</b> ★☆☆☆ ✕ 0</p> <p>nie vzdy dostupna miestnosť</p> <p>Jack_Daniels <span>ToDo</span></p>
<p><b>glad</b> ★★☆☆ ✕ 4</p> <p>Zvianočnieva sa.</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>glad</b> ★★☆☆ ✕ 0</p> <p>Node Server vyzerá byť fajn</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 2</p> <p>GrapesJS má vo future backlogu funkcionality, ktoré by sme potrebovali :/</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 3</p> <p>Okrem kódovania fičúr nie je čas na nič iné - testy, CI, code review, refactoring..</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>mad</b> ★☆☆☆ ✕ 6</p> <p>Závislosť od Grapesjs vývojárov nás spomaľuje, hlavne ich neschopnosť odpovedať na otázky.</p> <p>Jack_Daniels <span>ToDo</span></p>	
<p><b>glad</b> ★★☆☆ ✕ 0</p> <p>Pomalicky sa dostávame k pevnému základu a tým pádom sa budú dať brať menšie úlohy a lepšie rozdeliť práca</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>thank you</b> ★★☆☆ ✕ 0</p> <p>lubi sa mi ako sa shapuje editor dik ze to s nami vydrzis ado</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 3</p> <p>GrapesJS API je dost slabé</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>sad</b> ★☆☆☆ ✕ 3</p> <p>Miloš sa dnes nevedel pochváliť so svojou prácou na Node-e :/</p> <p>Jack_Daniels <span>ToDo</span></p>	<p><b>emotive</b> ★☆☆☆ ✕ 4</p> <p>som smutny ze som nic nespravil ale na chalanov som hrdy #bipolar</p> <p>Jack_Daniels <span>ToDo</span></p>	

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 funkcií 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 funkcií 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 ďalší 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

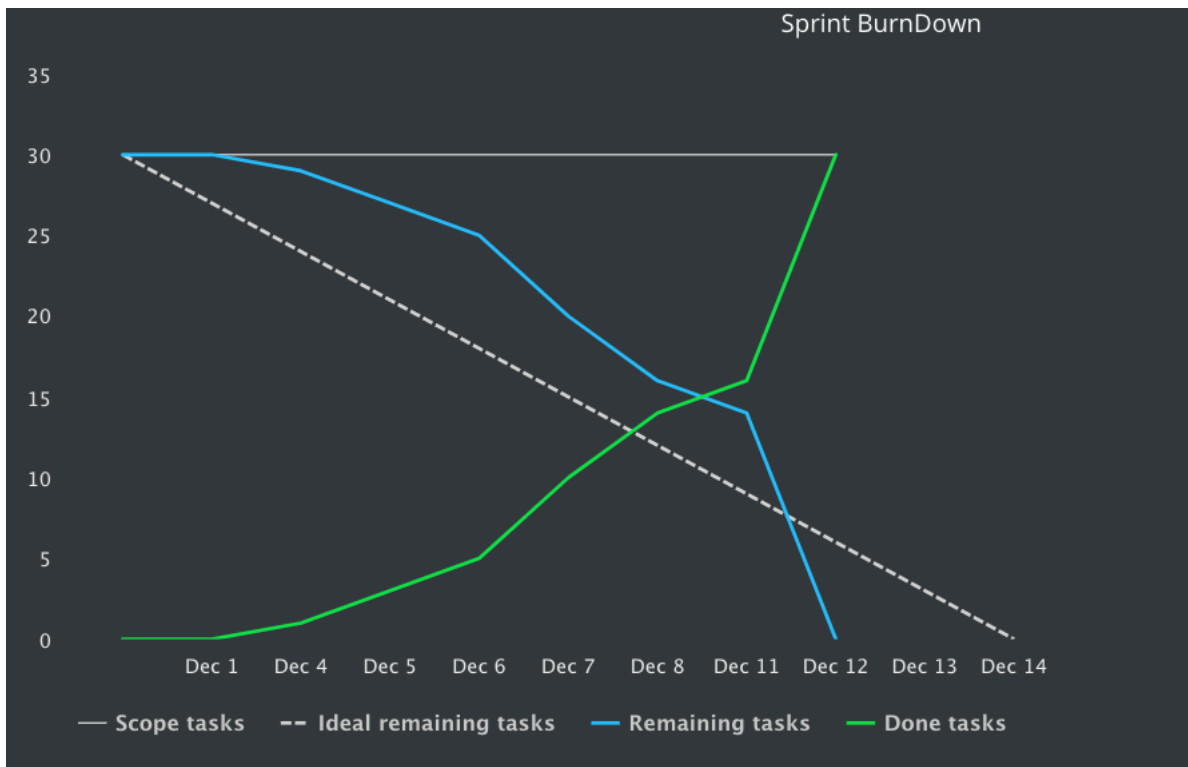
Každá feature obsahuje dokumentáciu a testovanie.

#### 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

<b>glad</b> ★★★★★ ✕ 0 Vrámcí semestra sa podarilo oveľa viac ako som čakal Jack_Daniels#2    ToDo	<b>glad</b> ★★★★★ ✕ 0 Každým týždňom je projekt kvalitnejší Jack_Daniels#2    ToDo	<b>sad</b> ★★★★★ ✕ 0 Projekt je žrút času a stále je tam čo doladovať Jack_Daniels#2    ToDo	<b>suicide watch</b> ★★★★★ ✕ 0 všetky zadania, prezentácie a testy v posledný tyzden Jack_Daniels#2    ToDo
<b>glad</b> ★★★★★ ✕ 0 PO je šťastný muž Jack_Daniels#2    ToDo	<b>glad</b> ★★★★★ ✕ 0 Základ editora je hotový. Začína nadstavba. Jack_Daniels#2    ToDo	<b>a lot of wisdom</b> ★★★★★ ✕ 0 komunikácia tímu grapesjs nie je ideálna (dead discord) Jack_Daniels#2    ToDo	<b>mad</b> ★★★★★ ✕ 0 Nechodí tímový mail -_ Jack_Daniels#2    ToDo
<b>glad</b> ★★★★★ ✕ 0 PO sa teší z rozloženia editora a skrývajúcích sa panelov Jack_Daniels#2    ToDo	<b>glad</b> ★★★★★ ✕ 0 Máme parádny editor Jack_Daniels#2    ToDo	<b>sad</b> ★★★★★ ✕ 2 Nemám sa teraz na čo sťažovať Jack_Daniels#2    ToDo	
<b>glad</b> ★★★★★ ✕ 0 Peťove a Milošove backendy sú top notch Jack_Daniels#2    ToDo	<b>impressive</b> ★★★★★ ✕ 0 hrozne sa mi paci editor Jack_Daniels#2    ToDo		
★★★★★ ✕ 0 Response time členov tímu je neuveriteľný Jack_Daniels#2    ToDo	<b>a privilege</b> ★★★★★ ✕ 0 to work with these fine gentlemen Jack_Daniels#2    ToDo		

Obrázok 14 - Retrospektíva pre piaty šprint

### 5.1.10 Zhodnotenie šprintu

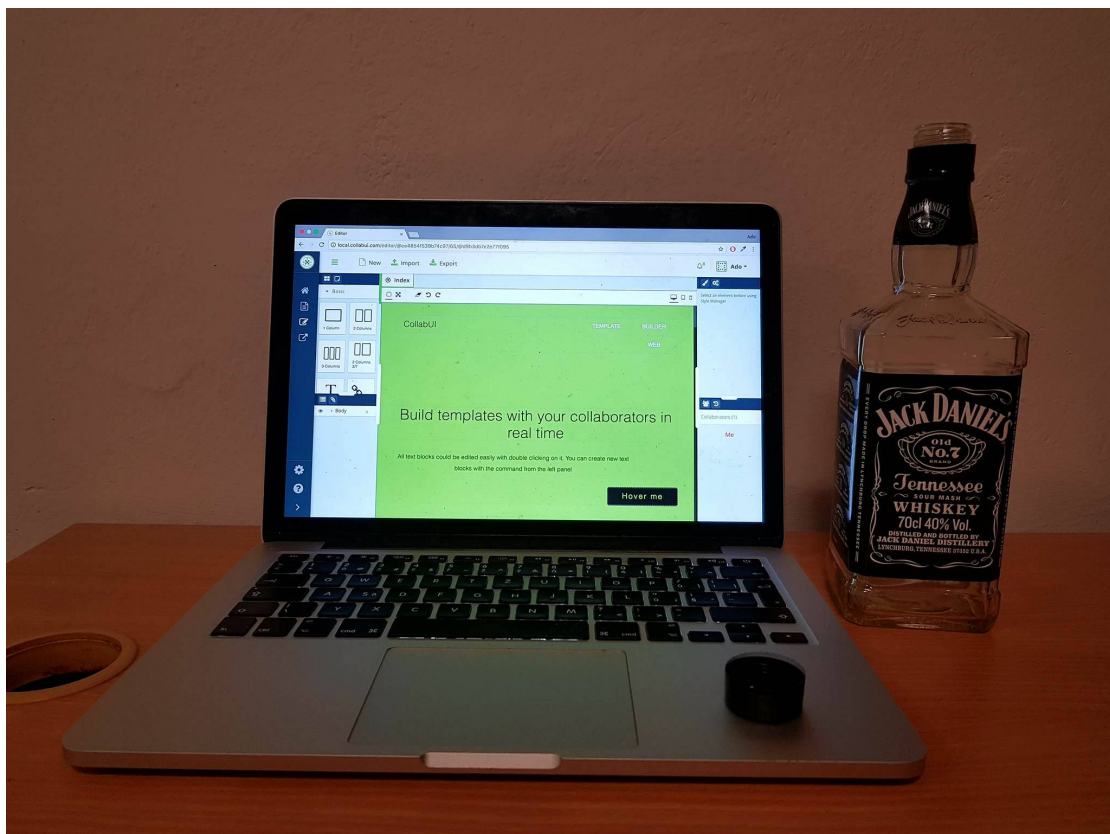
Keďže išlo o posledný šprint v rámci semestra všetky otvorené úlohy boli dokončené.

Ako vidno z burndown chartu zo začiatku sme mali implementačné problémy. Keďže nám veľakrát nastala situácia kedy bolo viac členov tímu závislých 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 závislí 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

### **5.1.11 Medzišprint (16.12.2017 - 17.2.2018) Nešprint**

Po skončení prvého semestra sme sa rozhodli v čase medzi semestrami vychytať chyby, na ktoré počas prvého semestra nezostal čas. Počas skúškového obdobia a sviatkov sme riešili nasledovné celky:

- Odladenie manažmentu tagov
- Zavedenie cronov do projektu
- Riešenie problémov s testovacím prostredím

Hlavným implementačným problémom tohto šprintu bolo generovanie a posielanie mailov. Generovanie mailu trvalo pri niektorých akciách nehorázne dlho, kvôli komplexnosti mailovej šablóny. Prejavovalo sa to hlavne pri generovaní skupinových mailov. Rozhodli sme sa preto maily posielat' asynchrónne prostredníctvom zavedenia cronov, ktorí sa spúšťajú každú minútu.

Ďalším problémom, ktorý sme museli vyriešiť bol nespoľahlivý server, na ktorom beží naše testovacie prostredie. Štatisticky 3x denne server zamrzol a nebolo možné sa do našej webovej aplikácie ani prihlásiť.

## 5.2 Letný semester

Na letný semester sme sa rozhodli sústrediť sa na samotnú implementáciu, kvôli časovému sklzu v naplánovaných „user stories“. Z tohto dôvodu sme vypustili z nášho workflowu pár pravidiel a náležitosti riešili operatívne.

### 5.2.1 Šprint1 (18.2.2018 - 4.3.2018) TatraTea\_22

Letný semester začal kostrbato a tento šprint sa radí medzi najmenej efektívne šprinty za celkové pôsobenie. Počas tohto šprintu testovali doposiaľ opravené funkcionality a aktualizovali podpornú knižnicu pre tvorbu stránok, čo prinieslo mnoho problémov. Kvôli problémom sme sa rozhodli sústrediť naše sily na implementáciu kolaboratívneho editora a všetko ostatné sme tým pádom vyradili z backlogu.

Celkový počet úloh v šprinte: 12

Každá feature obsahuje dokumentáciu a testovanie.

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Real-Time prenos udalostí nad Canvasom	7	13	Adrián Nagy
			Michal Melúch
			Miloslav Smetana
			Peter Písecký
SPLIT Rest z prvého semestra	5	5	Tomáš Mňačko
			Ján Kleň
			Lukáš Vrba



### 5.2.1.1 Retrospektíva

Mood	Item	Rating	Count
glad	osekanie uloh - vyzera to realnejsie	★★★★	0
glad	Kolaborácia sa hýbe	★★★★	0
sad	Budú mi chýbať frontendové veci	★★★★	0
sad	Mohli sme si prehodnotením úloh ušetriť robotu už skôr a nenechávať to na Peťa	★★★★	0
glad	Zamerali sme sa na to, čo je pre projekt skutočne dôležité	★★★★	0
change, elimi...	Vela menej podstatnych taskov sa odstranilo a ostalo tak viac casu na tie hlavne.	★★★★	0
sad	Na Peťa je veeeeeľa roboty	★★★★	0
a lot of wisdom	boolean expression vracia 1 alebo 0 nie farbu	★★★★	0
glad	Peto a Mišo = MVP	★★★★	0
sad	Lenivý začiatok, ja som lenivý. aj ja	★★★★	0
glad	Máme menej roboty, nemusíme sa ponáhľať	★★★★	0

Obrázok 16 - Retrospektíva prvého šprintu v letnom semestri

### **5.2.2 Zhodnotenie šprintu**

Vzhľadom k tomu, že tento šprint bol prvý v letnom semestri, efektivita práce členov tímu nebola nastavená dostatočne vysoko. Ako vidno z retrospektívy členovia tímu boli nadšení z vyradenia zvyšných naplánovaných úloh keďže sa všetci tešili na riešenie samotnej kolaborácie. Šprint sme rozdelili na dve skupiny. Prvá skupina riešila zvyškové úlohy z medzišprintu, zatiaľčo druhá skupina sa sústredila na prenos udalostí nad canvasom v reálnom čase. Vzhľadom k už spomínanej efektívite práce sme úspešne dokončili všetky zvyškové úlohy z medzišprintu. V rámci prenosu udalostí nad canvasom sa nám nepodarilo doriešiť všetky úlohy a teda sme boli nútení preniesť ho do ďalšieho šprintu.

### 5.2.3 Šprint2 (4.3.2018 - 18.3.2018) TatraTea\_32

V rámci tohto šprintu sme začali s manipuláciou prototypov. Najhlavnejšími časťami tohto šprintu boli nasledovné úlohy:

- Vytvorenie novej stránky
- Zmazanie stránky
- Prepínanie medzi stránkami
- Premenovanie stránky

Okrem vyššie spomínaných úloh sme ešte vyladzovali „bugy“ vzniknuté počas implementácie úloh, či už tohto šprintu alebo minulého.

Celkový počet úloh v šprinte: 16

Každá feature obsahuje dokumentáciu a testovanie.

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
Prepojenie stránok (anchors)	9	8	Adrián Nagy
			Lukáš Vrba
			Ján Kleň
			Tomáš Mňačko
SPLIT Real-Time prenos udalostí nad Canvasom	7	13	Peter Písecký
			Michal Melúch
			Miloš Smetana

### 5.2.3.1 Retrospektíva

<b>glad</b> ★★★★★ × 0 lepšie podelené úlohy, robíme progress TatraTea_32 ToDo	<b>glad</b> ★★★★★ × 0 esteze sockety su dost rychle TatraTea_32 ToDo	<b>sad</b> ★★★★★ × 0 Na projekt nemám zďaleka toľko času ako v zime TatraTea_32 ToDo	<b>sad</b> ★★★★★ × 0 viacero členov sa nezapájalo TatraTea_32 ToDo	<b>mad</b> ★★★★★ × 0 Skúmať GrapesJS je tímák sám o sebe TatraTea_32 ToDo
★★★★★ × 0 Celkom vela sa spravilo od minuleho stretnutia. TatraTea_32 ToDo	<b>glad</b> ★★★★★ × 0 Vyzerá, že budúci týždeň by mohli byť "spraviteľnejšie" tasky (menšie, viac) TatraTea_32 ToDo	<b>sad</b> ★★★★★ × 0 neviem ako funguju grapesjs komponenty :/ TatraTea_32 ToDo	<b>suicide watch</b> ★★★★★ × 0 skoro ostala len polka teamu24 TatraTea_32 ToDo	<b>mad</b> ★★★★★ × 0 Niektore tasky sa začínajú komplikovať TatraTea_32 ToDo
<b>glad</b> ★★★★★ × 0 ľudia su ochotní pomôcť TatraTea_32 ToDo	<b>glad</b> ★★★★★ × 0 Skoro sa všetko dokončilo TatraTea_32 ToDo	<b>sad</b> ★★★★★ × 0 Stále to ide moc pomaly TatraTea_32 ToDo	<b>sad</b> ★★★★★ × 0 Peto je chory TatraTea_32 ToDo	
<b>glad</b> ★★★★★ × 0 Takmer všetci členovia posledný týždeň makali (ja moc nie:D) TatraTea_32 ToDo	<b>glad</b> ★★★★★ × 0 Poniektoľ cítia deficit práce... výborne :D TatraTea_32 ToDo			

Obrázok 17 - Retrospektíva druhého šprintu v letnom semestri

#### 5.2.4 Zhodnotenie šprintu

Po už spomínanom prvom šprinte, ktorý predstavoval rozbiehanie tímového ducha do letného semestra, sa efektivita práce členov tímu zvýšila. Pôvodná skupina, ktorá pracovala na Feature: **Real-Time prenos udalostí nad Canvasom**, bola vystriedaná druhou skupinou, ktorá predtým tento problém neriešila. Tento manéver bol z dôvodu vystriedania a zároveň začlenenia členov tímu do danej problematiky. Takýmto štýlom sme vedeli lepšie rozložiť nápor problematiky na celú skupinu. Na druhej strane zaangažovanie druhej skupiny do tejto problematiky trvalo dlhší čas a tým pádom sme boli nutení znova presunúť tento Feature na ďalší šprint. Podobným spôsobom prebiehalo riešenie druhého Feature: **Prepojenie stránok (anchors)**.

### 5.2.5 Šprint3 (18.3.2018 - 2.4.2018) TatraTea\_42

Po náročnom druhom šprinte počas, ktorého sme riešili interné zdieľanie know-how sme sa plný elánu prehupli do ďalšieho šprintu. Riešili sme hlavne problémy s prenosom komponentov medzi kolaborantami, pričom sme dospeli k zisteniu, že bude potrebné vyhotovenie testovacích scenárov pre každý separátne preddefinovaný komponent. Čoraz hlbšie sme sa vnorili do tajov pomocného nástroja na tvorbu stránok, čo vyžadovalo taktiež dlhú analýzu.

Celkový počet úloh v šprinte: 16

Každá feature obsahuje dokumentáciu a testovanie.

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
SPLIT Prepojenie stránok (anchors)	13	8	Adrián Nagy
			Lukáš Vrba
			Ján Kleň
			Miloslav Smetana
			Michal Melúch
SPLIT SPLIT Real-Time prenos udalostí nad Canvasom	3	13	Peter Písecký
			Tomáš Mňačko

### 5.2.5.1 Retrospektíva

The image displays a Kanban board with retrospective cards. The cards are arranged in a grid-like structure with columns and rows. Each card has a header with a title, a star rating, and a status icon. The main body of the card contains a text description. At the bottom of each card, the author's name 'TatraTea\_42' and a 'ToDo' label are visible.

Card Title	Star Rating	Status Icon	Description	Author	Label
glad	★★★★	✖ 0	stránky sú super prepojené a informácie sa skvelo real-time prenášajú nad canvasom	TatraTea_42	ToDo
glad	★★★★	✖ 0	Už len doladiť :)	TatraTea_42	ToDo
mad	★★★★	✖ 0	Mali sme mať alkohol.. worst deal ever! sad!	TatraTea_42	ToDo
glad	★★★★	✖ 0	Appka zvládla stress test veľmi dobre	TatraTea_42	ToDo
glad	★★★★	✖ 1	Na projekt máme kopec voľného času	TatraTea_42	ToDo
glad	★★★★	✖ 0	Spoločné tričká	TatraTea_42	ToDo
glad	★★★★	✖ 0	Real time kolaborácia funguje parádne	TatraTea_42	ToDo
glad	★★★★	✖ 1	sme v cieľovej rovinke a tím maká	TatraTea_42	ToDo
horrible	★★★★	✖ 3	Kde je môj tatranský?	TatraTea_42	ToDo
glad	★★★★	✖ 0	o mesiac končí semester	TatraTea_42	ToDo
glad	★★★★	✖ 0	skvelý nástroj, kolaborácia funguje ako švajčiarske hodinky	TatraTea_42	ToDo
glad	★★★★	✖ 3	Dorabanie poslednych veci a fixovanie vyzera to, ze to naozaj stihneme a pravdepodobne vyhrame ked som trafil ten kelimok do kosa.	TatraTea_42	ToDo
glad	★★★★	✖ 0	Príprava na IIT.SRC	TatraTea_42	ToDo
sad	★★★★	✖ 0	Veľa sme toho nestihli, ale v konečnom dôsledku to vôbec nevaďí...	TatraTea_42	ToDo

Obrázok 18 - Retrospektíva tretieho šprintu v letnom semestri

### **5.2.6 Zhodnotenie šprintu**

Šprint 3 sa niesol v znamení pozitívnej nálady. Testovacie scenáre nám pomohli v detegovaní problémov a tím zrýchlila proces vývoja. Táto operatíva však zabrala mnoho času, no bola nevyhnutá k efektívnemu pokračovaniu. Pozitívna nálada pretrvala až do konca šprintu, kde sme už začali vidieť svetlo na konci tunela. Zároveň sa blížili prípravy na súťaž IIT.SRC 2018, ktorá bola pre nás dôležitou súčasťou semestra. Táto skutočnosť nás motivovala počas celého šprintu a tím bol nadšený, že môžeme byť súčasťou takejto pocty.



## 5.2.7 Šprint4 (2.4.2018 - 16.4.2018) TatraTea\_52

V rámci tohto šprintu sme sa sústredili na postup v už spomínanej súťaži IIT.SRC 2018. Súčasťou úpravy bolo zhotovenie plagátu, pri ktorom sme sa sústredili na výhru. Bohužiaľ po zistení nečakaných zvrátov, oznamujúcich našu nemožnosť zúčastnenia na súťaži plagátov v rámci súťaži IIT.SRC, sme nemohli aplikovať naši pripravovanú výhru. Taktiež sme nakúpili tímové košeľe a pripravili scenáre pre prezentáciu. Ďalej sme sa v šprinte sústredili na nasledujúce časti projektu:

- Uzamykanie komponentov

Okrem tejto časti sme vo veľkom naďalej odľadňovali prípadné vzniknuté chyby.

Celkový počet úloh v šprinte: 17

Každá feature obsahuje dokumentáciu a testovanie.

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
SPLIT SPLIT Prepojenie stránok (anchors)	4	2	Peter Písecký
			Miloslav Smetana
			Ján Kleň
SPLIT SPLIT SPLIT Real-Time prenos udalostí nad Canvasom	8	5	Ján Kleň
			Tomáš Mňačko
IIT.SRC 2018	5	5	Michal Melúch
			Lukáš Vrba
			Adrián Nagy
			Ján Kleň

## 5.2.7.1 Retrospektíva

<p><b>glad</b> ★★★★★ ✕ 0</p> <p>Niektoré komponenty príjemne prekvapili (countdown atd.)</p> <p>TatraTea_52      ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 0</p> <p>Vyhráme to lebo Miloš trafil kelímok do koša a teda je to osud!</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Projekt sa začína sám rozšypávať</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Comic San je super font, nerozumiem aký s ním máte problém ;{</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>Vytvorili sme neovládateľnú chiméru vzájomných závislostí medzi knižnicami. Toto už nikto nikdy nerozmotá. Dali nám priveľa moci. Boh nás bude súdiť.</p> <p>"I am become death, the destroyer of source." - Bower Update, 2018</p> <p>TatraTea_52      ToDo</p>
<p>★★★★ ✕ 0</p> <p>Máme pekný návrh (bez diagramov)</p> <p>TatraTea_52      ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 0</p> <p>Budeme mať spoločné tričká, budeme vyzerať ako siámske opice :))</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>furt straší parkinson</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Nie, Comic San nie je super font na IITSRC!</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>mišo si so mnou nedal tatrasky</p> <p>TatraTea_52      ToDo</p>
<p><b>glad</b> ★★★ ✕ 8</p> <p>zadusím sa v konskej hlave</p> <p>TatraTea_52      ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 1</p> <p>Dotiahli sme to ďaleko</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>Praca sa ku koncu semestra kopi</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>Nesúhlasím s niektorými členmi tímu</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>A za tebou okno !!!</p> <p>TatraTea_52      ToDo</p>
<p><b>glad</b> ★★★★★ ✕ 3</p> <p>Žiadne väčšie problémy pri fixovaní. S veľkou pravdepodobnosťou postupíme. Celú výhru investujem do krypta.</p> <p>TatraTea_52      ToDo</p>	<p><b>glad</b> ★★★★★ ✕ 0</p> <p>táto retrospektíva je super</p> <p>TatraTea_52      ToDo</p>	<p><b>sad</b> ★★★★★ ✕ 0</p> <p>grapesjs tam má pár vecí kvalitne rozjebaných, mohli by spraviť kolaboráciu</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 0</p> <p>Tam su dvere.</p> <p>TatraTea_52      ToDo</p>	<p><b>mad</b> ★★★★★ ✕ 3</p> <p>Nehádajte sa!!!</p> <p>TatraTea_52      ToDo</p>

Obrázok 19 - Retrospektíva štvrtého šprintu v letnom semestri

### 5.2.8 Zhodnotenie šprintu

Ako vidno z retrospektívy v tíme prevládla odviazaná atmosféra. Každý člen tímu je spokojný z doposiaľ odvedenou prácou. Každý našiel v projekte časť, ktorá ho naplňala. Konečne sa tímu podarilo odladiť Feature: **Prepojenie stránok**. Do ďalšieho šprintu sme si prenisli iba druhú Feature a to z dôvodu zavedenia pridanej hodnoty do kolaborácie vo forme riešenia konfliktov medzi kolaborantmi. V rámci tohto šprintu sme sa zúčastnili súťaže IIT.SRC, na ktorej sa nám úspešne podarilo postúpiť súverénne medzi prvými dvoma tímami. Okrem postupu sa o vytvorený prototyp zaujímalo veľa ľudí. Medzi nich patrili aj niektoré firmy, ktoré si dokonca vypýtali kontakt pre možnú budúcu spoluprácu.

### 5.2.9 Šprint5 (16.4.2018 - 30.4.2018) TatraTea\_62

V poslednom šprinte sme si hlavne plánovali úlohy k dokumentáciám. Zamerali sme sa na technický opis problémovej oblasti a snažili podchytiť nami vyvinuté náležitosti a riešenie. Dôležitou časťou šprintu bola príprava na odovzdávku projektu nasledujúcemu tímu, ktorí bude zrejme potrebovať zo začiatku jasný a stručný technický opis riešenia. Zároveň sme si naplánovali posledný balík chýb, ktoré vyplávali z minulého šprintu. Naplánovali sme taktiež záverečné testovanie s dvoma adeptami z opozitného tímu.

Celkový počet úloh v šprinte: 10

Každá feature obsahuje dokumentáciu a testovanie.

Feature	Počet úloh v rámci feature	Odhadované úsilie	Zodpovední členovia tímu
SPLIT SPLIT SPLIT SPLIT Real-Time prenos udalostí nad Canvasom	10	5	Ján Kleň
			Miloslav Smetana
			Michal Melúch
			Adrián Nagy
			Lukáš Vrba
			Peter Písecký
			Tomáš Mňačko

### 5.2.9.1 Retrospektíva

<b>glad</b> ★★★★★ x 3 konečne finishujeme TatraTea_62 ToDo	<b>glad</b> ★★★★★ x 3 nemam este prezentáciu, Milos pojde rychlo domov ^_^ TatraTea_62 ToDo	<b>mad</b> ★★★★★ x 0 tesne pred diplomovkov treba robit dokumentáciu, ktorá už mohla byť hotová TatraTea_62 ToDo	<b>glad</b> ★★★★★ x 6 Postúpili sme do semifinále !!! TatraTea_62 ToDo
<b>glad</b> ★★★★★ x 0 Keď doladíme bugy a urobíme export tak to bude dobre prezentovateľné TatraTea_62 ToDo	<b>sad</b> ★★★★★ x 0 Posledné týždne na projekt nemám absolútne žiadny čas kvôli diplomovke TatraTea_62 ToDo	<b>mad</b> ★★★★★ x 1 toto bol omyl TatraTea_62 ToDo	<b>glad</b> ★★★★★ x 0 Ak zvíťazíme ideme na víkend na chatu !!! TatraTea_62 ToDo
<b>glad</b> ★★★★★ x 3 Peto je hotovy TatraTea_62 ToDo	<b>sad</b> ★★★★★ x 0 mam alergiu a stipu ma oci a mam zapchaty nos TatraTea_62 ToDo	<b>glad</b> ★★★★★ x 0 ucastnici testu dali aj nejaky usefult feedback TatraTea_62 ToDo	
<b>glad</b> ★★★★★ x 0 Participantni ocenili nas projekt a v skutku sa im pacil TatraTea_62 ToDo	<b>sad</b> ★★★★★ x 0 Vela dokumentacie za tyzden ved to bude mat cez 800MB cela dokumentacia... TatraTea_62 ToDo		

Obrázok 20 - Retrospektíva piateho šprintu v letnom semestri

### **5.2.10 Zhodnotenie šprintu**

I keď sme naplánovali prípravu dokumentácii v tomto šprinte sme sa k nim nedostali kvôli snahe finalizácie odladenia kolaboratívneho editora, kvôli záverečnému testovaniu. Testovanie v skratke prebehlo úspešne, výborne ba až skvele. Respondenti hodnotili webovú aplikáciu za použiteľnú a hodnú na kvalitatívne použitie v praxi. Zároveň poskytli cenný feedback predlohou budúceho vývoja.

## 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 šprint 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. preštýľovanie 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 refactoring, code review a testovanie

#### 5.šprint

- Opraviť tímový mail ☹

## 6.2 Letný semester

Ako už je vo zvykom aj tento semester sme na konci v rámci tímu spravili globálnu retrospektívu, na ktorej sme zhodnotili všetky šprinty a celkovo sme si povedali ako sa nám práca v tíme páčila, nepáčila, hnevala, či tešila.

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

### 1.šprint

- Odťažit' preťažených jednotlivcov
- Motivovať zlenivelých členov

### 2.šprint

- Rozbit' komplexné úlohy na menšie
- Zdieľať technické problémy a nájsť riešenie spoločne
- Nájsť si viac času na projekt (školské povinnosti)
- Plánovať viac času na analýzu

### 3.šprint

- Zarobiť si na zaužívané tímové občerstvenie

### 4.šprint

- Nepoužívať Comic San
- Sústrediť sa na kolaboratívny editor a odladiť ho

### 5.šprint

- Do budúca dokumentovať priebežne

### 6.šprint (aktuálne prebieha)

### 7.šprint (budúci, posledný šprint)



## 7 Záver

Na záver dokumentu riadenia nemôžeme nechať opomenúť fakt, že sme skoro 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 boli odhodlaní pracovať aj počas skúškového obdobia a sviatkov. Ako sa hovorí: Bez koláča nie sú práce.

Na skúškové obdobie a sviatky sme si naplánovali nasledovné funkcionality:

- Vytváranie viacerých prototypov v jednom projekte
- Prepojenie prototypov, stránok
- Vytvorenie prototypu na zahodenie – hlasový chat
- Verziovanie projektov
- Chat

Vyššie stanovené ciele sa nám však nepodarilo naplniť, kvôli problémom popisovaným vyššie a ďalší semester nezačal podľa našich predstáv, bez plánovaného mierneho náskoku. Každým ďalším šprintom sme sa ponárali hlbšie a hlbšie do problémov, čo spôsobilo vylúčenie funkcionalít z backlogu a ich posunutie do ďalšieho vývoja.

Nadmernú časť práce letného semestra vyplňala analýza vnútornej štruktúry podporného nástroja pre vytváranie stránok, odchyťovanie vstavaných udalostí a ich použitie v riešení. Túto analýzu bolo nemožné odhadnúť a naplánovať, z tohto dôvodu to môže v niektorých prípadoch vyzeráť ako stratený čas, avšak bez nej by sme sa nedokázali posunúť ďalej. Z tohto dôvodu vyhlasujeme burndown charts pre letný semester ako irelevantné a neuvažujeme ich v dokumentácii. Napriek vyššie uvedeným skutočnostiam sa nám podarilo na termín prezentácie IIT.SRC 2018 vyvinúť prezentovateľnú aplikáciu, ktorú sa nám vďaka našim vylepšeným komunikačným zručnostiam podarilo „predať“ a dostať sa ako jeden z najlepších tímov do semifinále.

V čase písania dokumentácie sa pripravujeme na spomínané semifinále, na ktorú pripravujeme novú funkčnosť, exportovanie projektov.

# 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 akcií ako aj znižuje neprehľad členov tímu o vykonávaných skutočnostiach. V rámci nášho tímu uvádzame 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 funkcionálna časť projektu, ktorá sa skladá z niekoľkých taskov

**Task** - úloha v rámci jedného story

**Product owner** - PO - komunikuje so zákazníkom a zbiera od neho požiadavky, slúži ako mediátor pre komunikáciu s tímom a zákazníkom

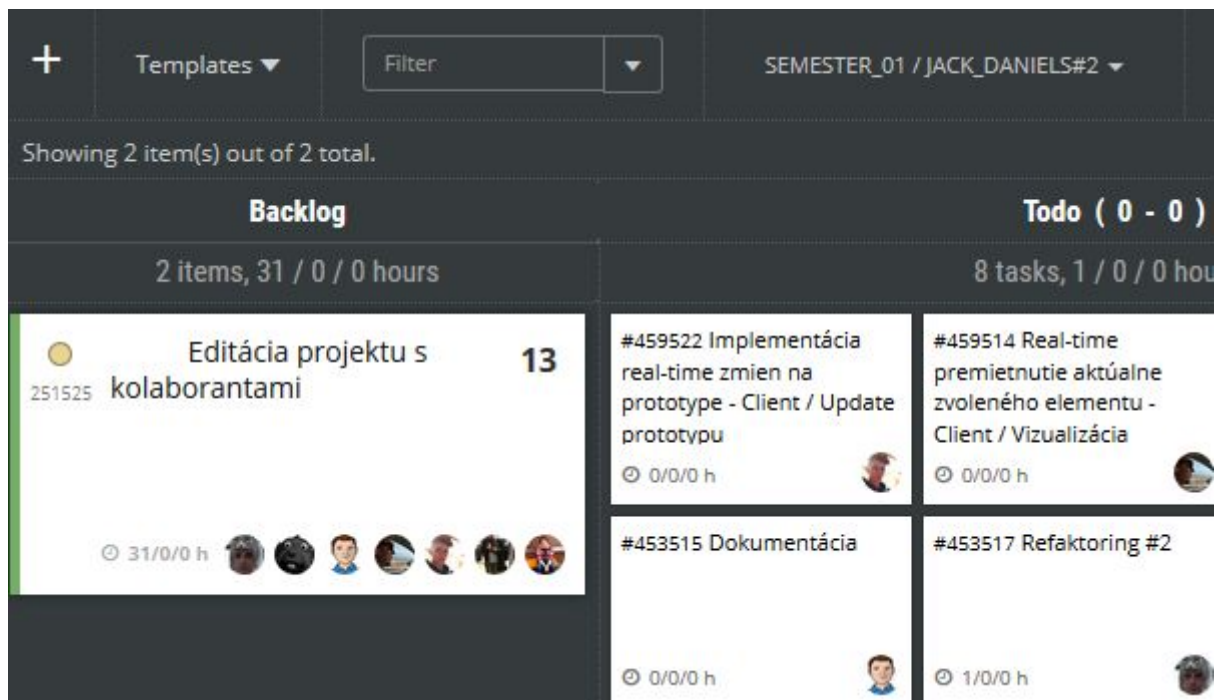
**Definition of done** - DOD - požiadavky, ktoré musia byť splnené pri odovzdaní časti projektu či už ide o task alebo story...

**Effort** - celkový čas strávený nad jednou US

## 3. Pravidlá a postupy

### 3.1 Konvencie názvov

Všetky názvy pre jednotlivé Epicy, features 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 Povinné 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.

Epic/Feature **Editácia a kolaborácia (editor)** Theme ▾

Due date **Not set** **13**

## Editácia projektu s kolaborantami

MOSCOW VALUE ▾ RISK KANO

11 / 19 31/0/0 h

Release **SEMESTER\_01** ▾ Sprint **SEMESTER\_01 / Jack\_Daniels#2** ▾

Lead time: 9.89 days Cycle time: 10.58 days

Ako používateľ (aj neregistrovaný) chcem vedieť prístup k editovaniu projektu a zároveň kolaborovať (vidieť zmeny) s ostatnými používateľmi, kt. pracujú s prototypom v čase.

Obsahom úlohy je zabezpečiť kolaboráciu v reálnom čase prostredníctvom socket.io a NodeJS Servera

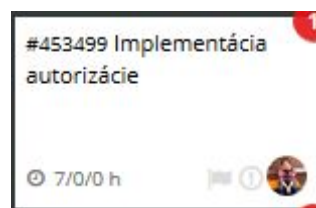
**ACCEPTANCE CRITERIA**

- Dokážem otvoriť editor aj bez prihlásenia cez URL
- Ak mám iba právo WATCH, neviem editovať prototyp, teda ani uložiť
- Vidím akcie kolaborantov a oni moje v reálnom čase

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 stretnutie, kde sa diskutuje a značí na tabuľu čo sa bude ďalej robiť a s tým spojené vytvorenie features. Features s

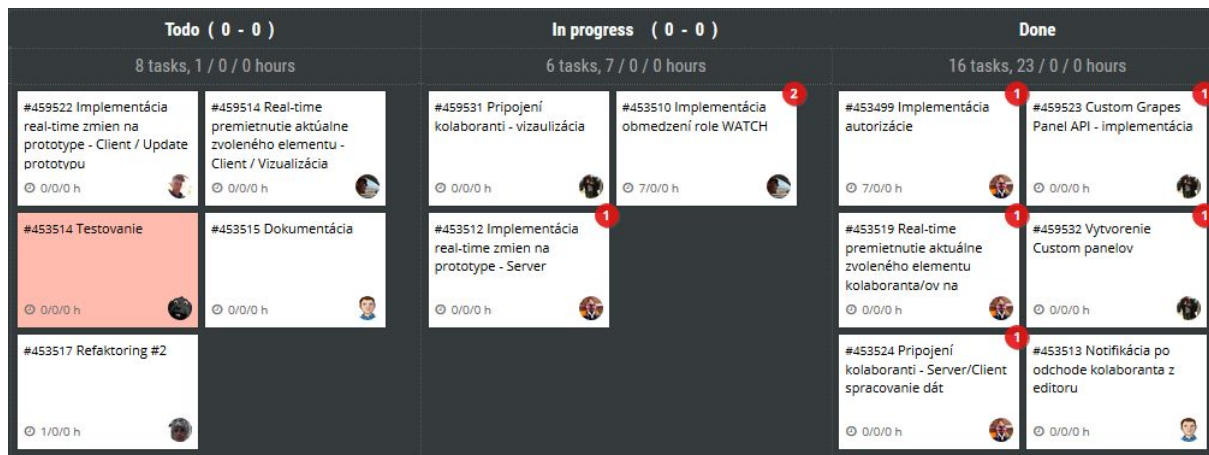
taskami sú neskôr v ten deň nahodené a dôkladne popísané aj s validačnými kritériami pridané do nástroja ScrumDesk vedúcim tímu po diskusii.

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 kritérií sa prijme kompromis, ktorý vyhovuje všetkým členom tímu.

Každý člen tímu si prideliť 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 keď 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.

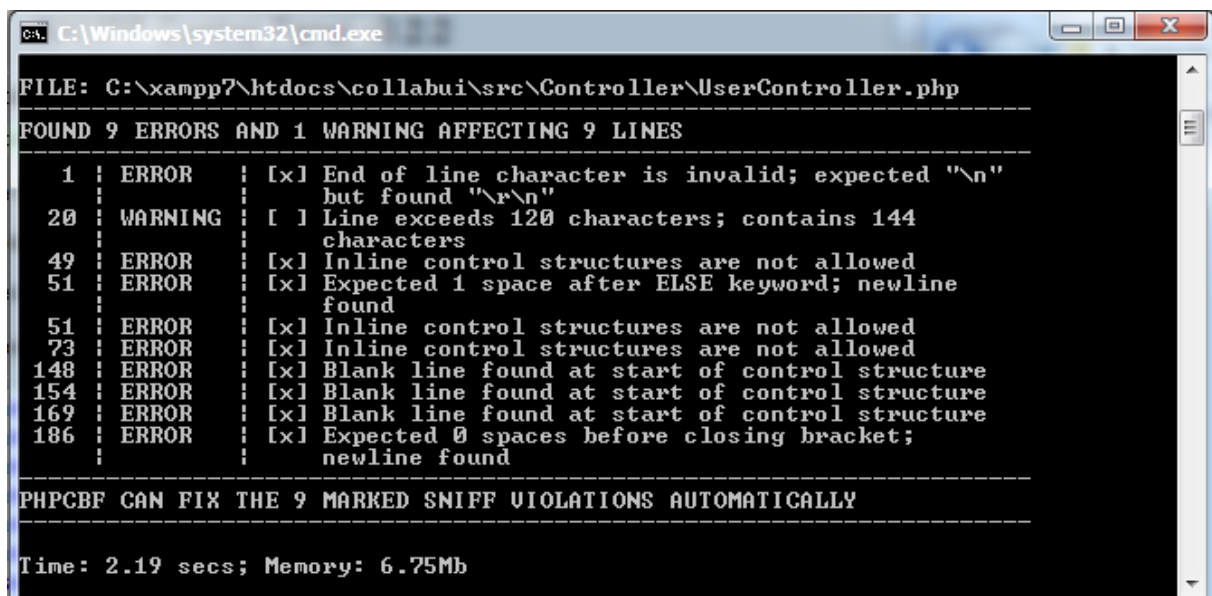
### Inštalácia:

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

### Použitie:

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

### Príklad výstupu:



```
C:\Windows\system32\cmd.exe
FILE: C:\xampp7\htdocs\collabui\src\Controller\UserController.php
-----
FOUND 9 ERRORS AND 1 WARNING AFFECTING 9 LINES
-----
  1 | ERROR   | [x] End of line character is invalid; expected "\n"
    |         |     but found "\r\n"
 20 | WARNING | [ ] Line exceeds 120 characters; contains 144
    |         |     characters
 49 | ERROR   | [x] Inline control structures are not allowed
 51 | ERROR   | [x] Expected 1 space after ELSE keyword; newline
    |         |     found
 51 | ERROR   | [x] Inline control structures are not allowed
 73 | ERROR   | [x] Inline control structures are not allowed
148 | ERROR   | [x] Blank line found at start of control structure
154 | ERROR   | [x] Blank line found at start of control structure
169 | ERROR   | [x] Blank line found at start of control structure
186 | ERROR   | [x] Expected 0 spaces before closing bracket;
    |         |     newline found
-----
PHPCBF CAN FIX THE 9 MARKED SNIFF VIOLATIONS AUTOMATICALLY
-----

Time: 2.19 secs; Memory: 6.75Mb
```

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 najstriktnnejšie. 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
```

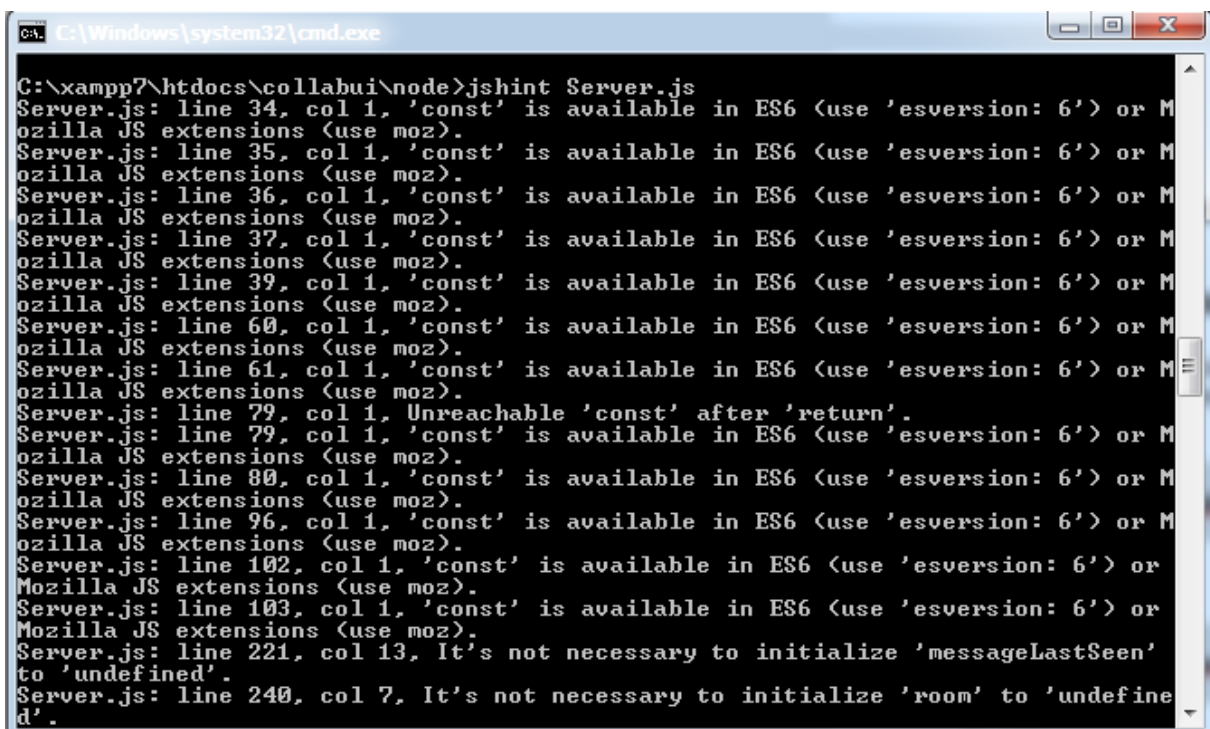
alebo:

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

Použitie:

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

Príklad výstupu:



```
C:\Windows\system32\cmd.exe
C:\xampp7\htdocs\collabui\node>jshint Server.js
Server.js: line 34, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 35, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 36, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 37, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 39, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 60, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 61, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 79, col 1, Unreachable 'const' after 'return'.
Server.js: line 79, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 80, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 96, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 102, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 103, col 1, 'const' is available in ES6 (use 'esversion: 6') or Mozilla JS extensions (use moz).
Server.js: line 221, col 13, It's not necessary to initialize 'messageLastSeen' to 'undefined'.
Server.js: line 240, col 7, It's not necessary to initialize 'room' to 'undefined'.
```

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. Integrované 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*.

```
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

```

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();

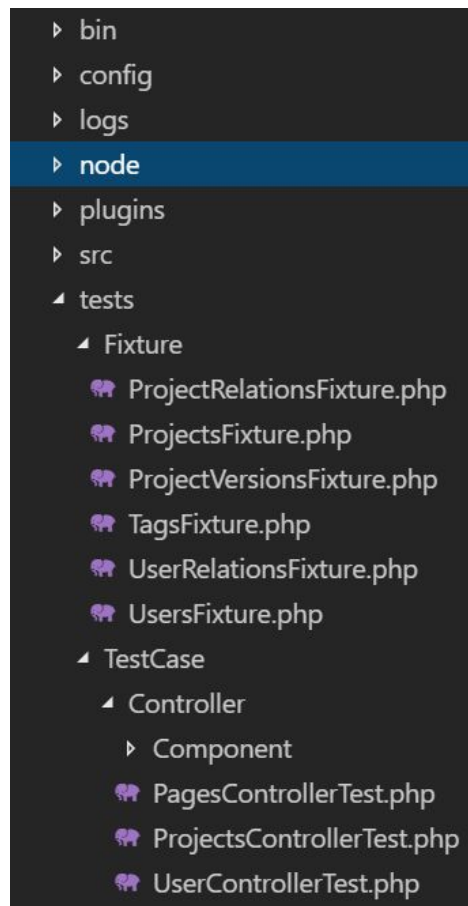
    $this->assertRedirect('/');
}

```

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 rozdeľuje 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 názov súboru.test.js v opačnom prípade nebude vedieť skript nájsť 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);

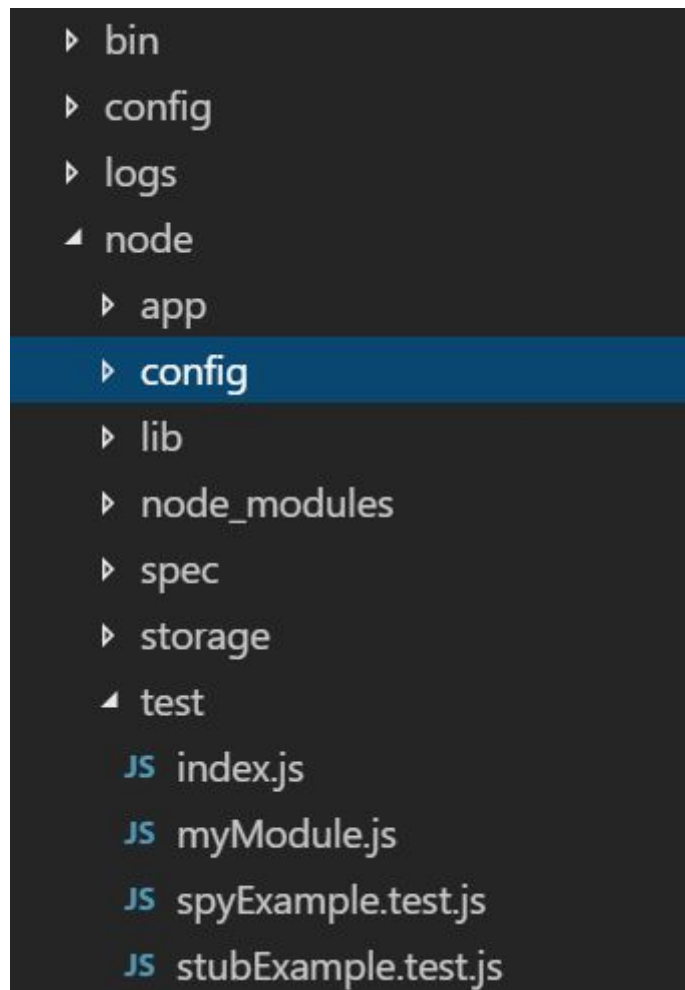
    //when
    proxy();
    proxy();

    //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 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.

JS testy sa spúšťajú 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 funkcionálna jednotka projektu, ktorá sa skladá z niekoľkých taskov
- **Task** - úloha v rámci jedného story
- **Product owner** - 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](https://bitbucket.org/agileuiteam/uiagile_repo)
  - URL pre klonovanie: [https://BB-NICKNAME@bitbucket.org/agileuiteam/uiagile\\_repo.git](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:

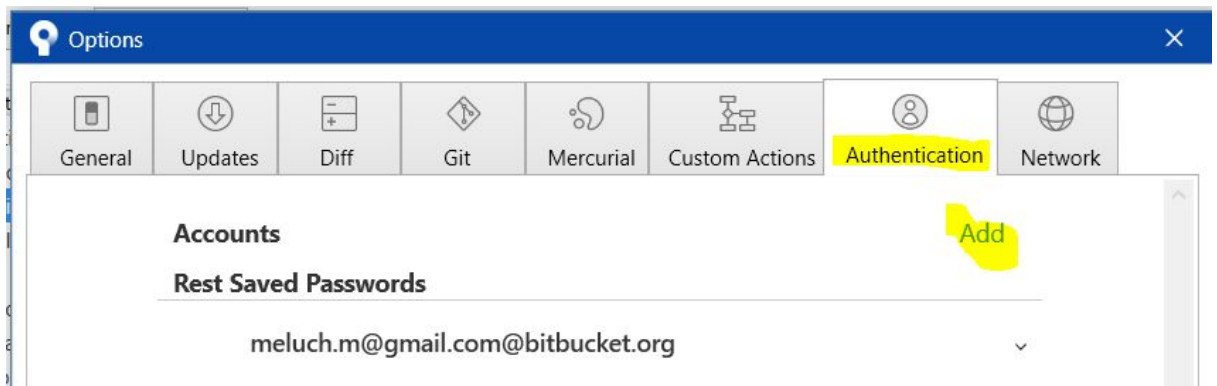
1. Systém riadenia revízií **Git** (stiahnuť [TU](#))
2. Grafický nástroj pre správu verzií **Sourcetree** od spoločnosti *Atlassian* (stiahnuť [TU](#))

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

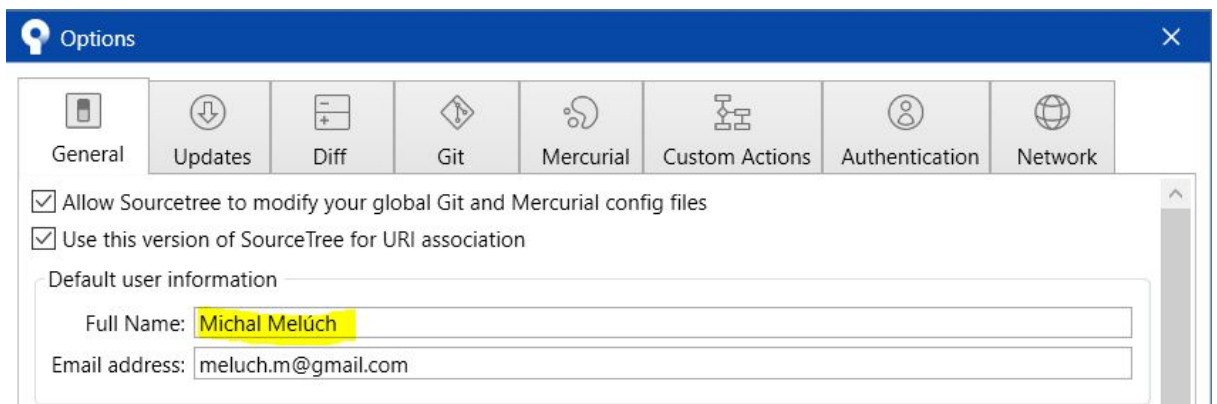
Po nainštalovaní uvedeného softvéru je nutné prepojiť aplikáciu *Sourcetree* so vzdialeným repozitárom projektu a Atlassian kontom vývojára.

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*):



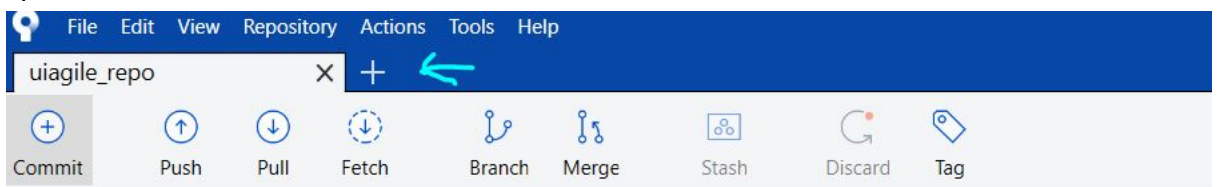
3. Zadáme svoje prístupové údaje, pomocou ktorých sa prihlasujeme aj na doménu <https://bitbucket.org/>.
4. V prvej záložke pup-up okna vyplníme svoje celé meno a priezvisko:



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

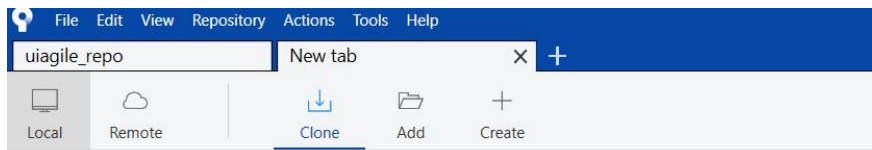
Postup pripojenia sa k vzdialenému repozitáru aplikácie je nasledovný:

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](#)).



## Clone

Cloning is even easier if you set up a remote account

Repository Type: This is a Git repository

Local Folder:

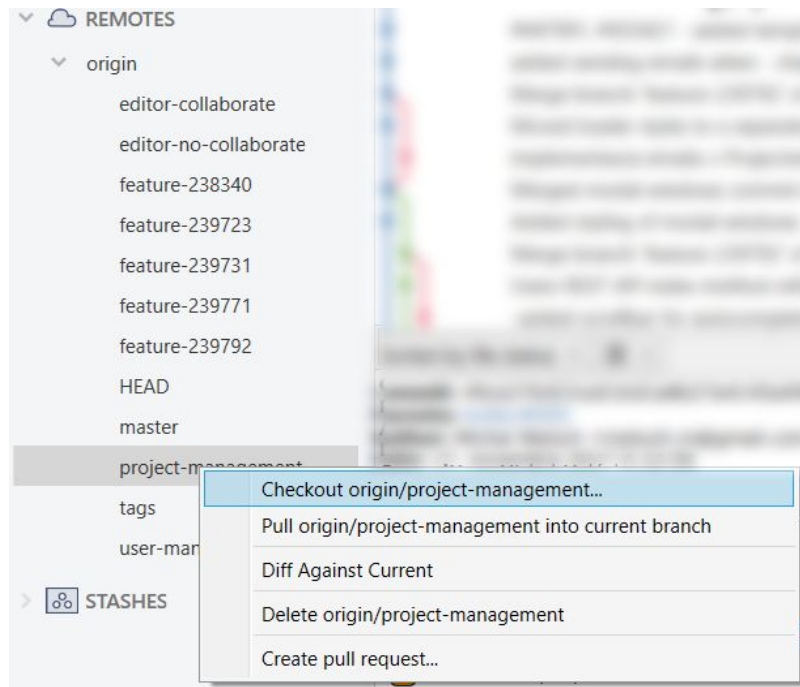
Advanced Options

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)

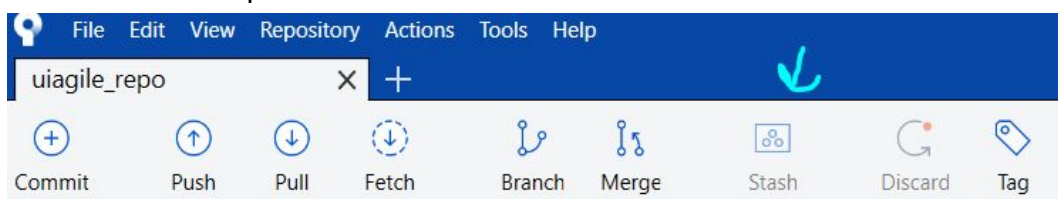
V prípade konfliktov medzi súbormi po príkaze *Push* je nutné tieto konflikty vyriešiť tak, aby zostali zachované zmeny obidvoch autorov. Následne je možné tieto súbory opakovane commitnúť a budú systémom *Git* akceptované.

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

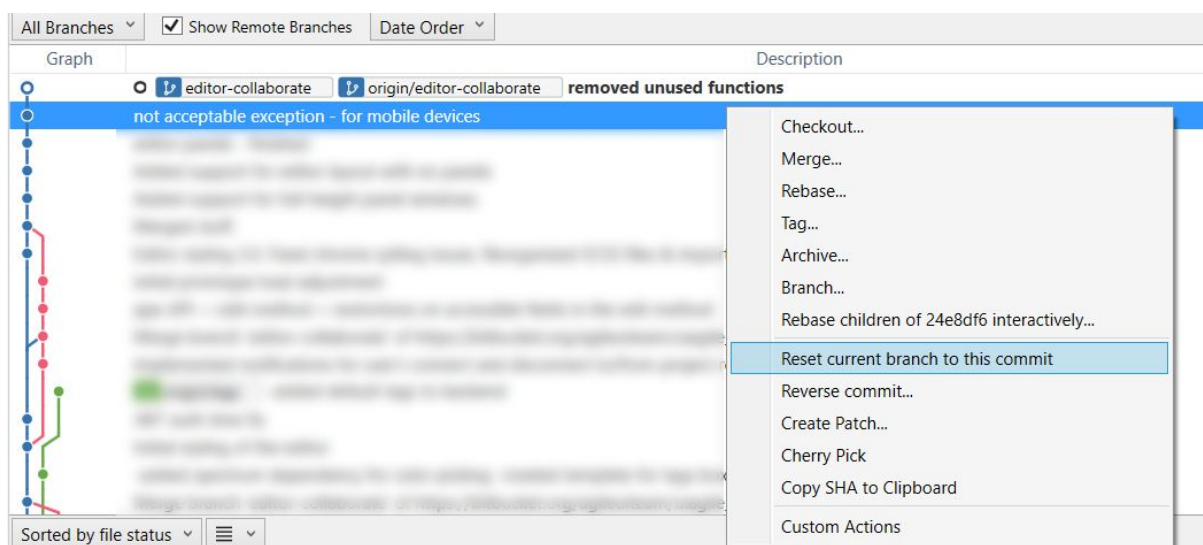
## 4.4 Návrat ku starším verziám projektu

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:

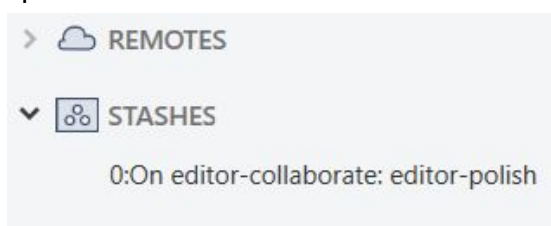
1. Lokálne zmeny si v prípade potreby uložíme pomocou príkazu *Stash* z horného menu aplikácie:



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 commitu. 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 postupuje 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 [dokumentácii](#), v časti poznámky k inštalácii 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. Príkaz 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 'POUZI VATELSKY 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](#)
- [PostgreSQL + 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 a prejsť do priečinka, v ktorom sa bude do aktuálneho priečinka pomocou gitu:

```
git clone git @bitbucket.org:agileteam/ui-agile-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)

```

229 'Datasources' => [
230     'default' => [
231         'className' => 'Cake\Database\Connection',
232         'driver' => 'Cake\Database\Driver\Postgres',
233         'persistent' => false,
234         'host' => 'localhost',
235         'port' => '5432',
236         'username' => 'Iceweex',
237         'password' => '',
238         'database' => 'test',
239         'encoding' => 'utf8',
240         'timezone' => 'UTC',
241         'flags' => [],
242         'cacheMetadata' => true,
243         'log' => false,
244         'quoteIdentifiers' => false,
245         'url' => env('DATABASE_URL', null),
246     ],

```

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

```

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

```

A pre Cassandra pomocou príkazom:

```

cqlsh -f 'Root directory'/src/Dat abase/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 skopírovaním súboru `gulp.default.js` v priečinku `/node/` do súboru `gulp.js` a nastaviť údaje (viď. Obrázok nižšie).

```

2 //***** CUSTOME SETTINGS *****/
3 var server = {
4   base: '/Users/Icweex/Sites/uiagile_repo', //ROOT FOLDER
5   port: 80
6 };
7
8 var browser = {
9   proxy: 'http://uiagile.com/',
10  port: 3000
11 };

```

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`.

```

1 ClientComponent = function () {
2
3   aoweb.component.call(this);
4
5   this.host = 'http://localhost:8082/';
6
7   this.loaderActive = undefined;
8   this.socket = undefined;
9 };

```

Na spustenie Node servera z priečinka `/node/` použijeme nasledovný príkaz, ktorý ostane 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:POUZI VATELSKE_MENO/tea_m_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:

- Nahratie 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`