

Dokument
k inžinierskemu dielu
Tím 24

Obsah

1	Úvod	7
2	Globálne ciele pre ZS	8
3	Globálne ciele pre LS	9
4	Celkový pohľad na systém	10
4.1	Architektúra	10
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
5	Moduly	16
5.1	Manažment používateľov	17
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
5.2	Manažment projektov	33
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
5.3	Manažment kolaborantov.....	49
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
5.4	Manažment tagov.....	68
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

5.5	Editor	73
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
6	Záverečné testovanie	118
6.1	Testovacie scenáre	118
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
6.2	Dotazník	124
6.3	Výstup z testovania	125
7	Záver	127

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

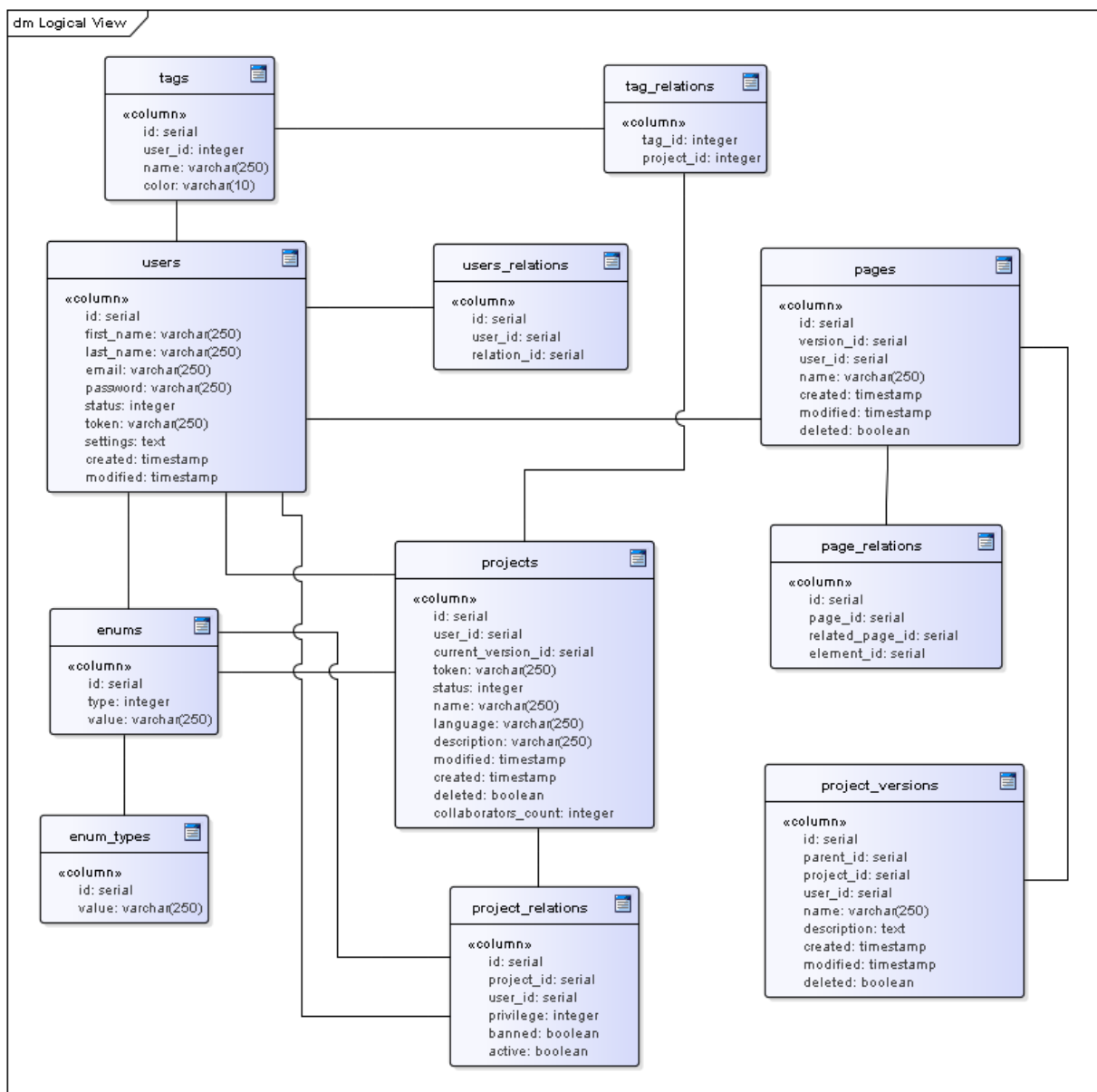
4.1 Architektúra

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 „upiect“, teda vytvoriť PHP Model, prostredníctvom príkazu z root adresára:

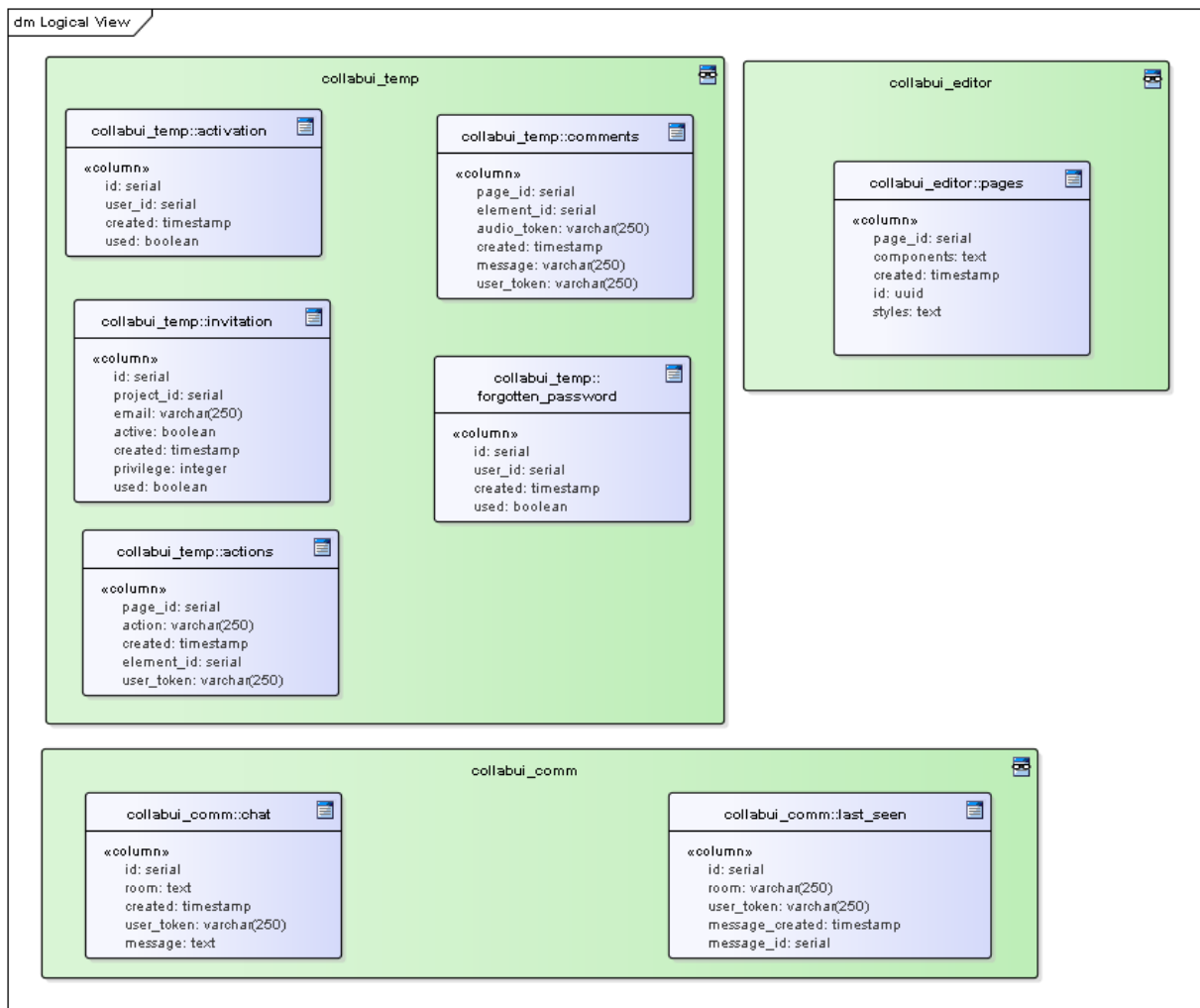
```
./bin/cake bake model nazov_tabulky
```

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

Priečinkom 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

Priečinkom 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

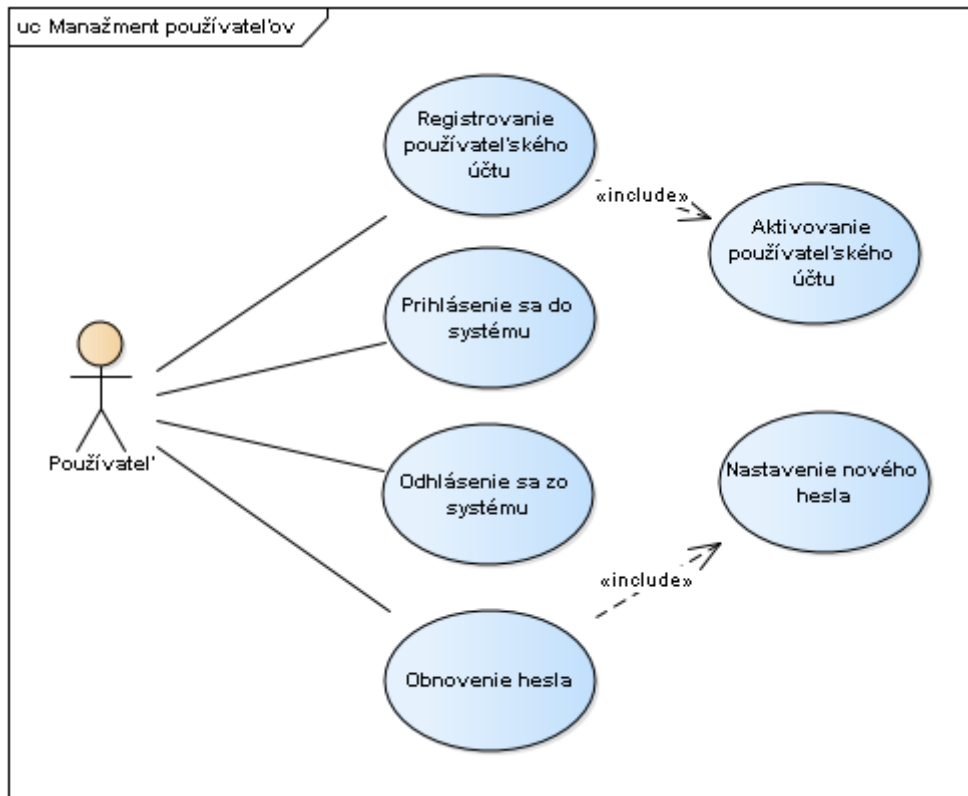
V rámci analýzy sme sa zameriavali nato kto bude náš hlavný používateľ. Náš používateľ by predovšetkým mal byť niekto kto je technicky zdatný. Primárne sa teda budeme sústreďovať na dizajnérov a programátorov popr. frontend developerov. Týchto používateľov sme vybrali práve kvôli špecifickosti systému. 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.

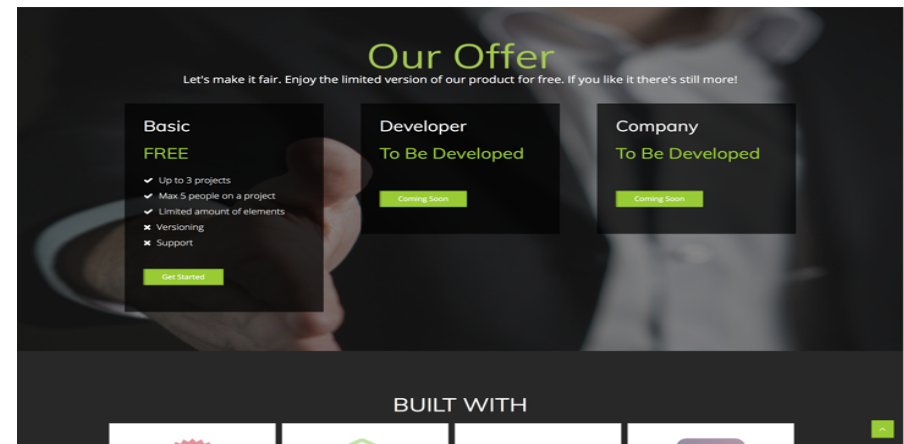
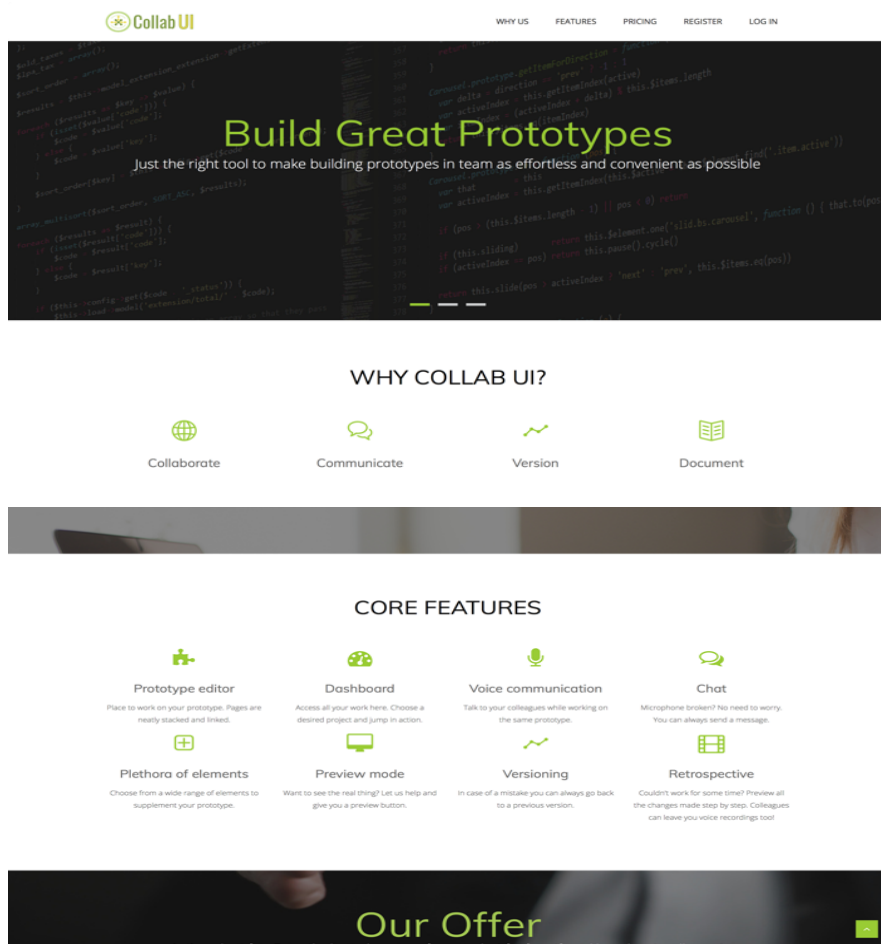
5.1.4.1.1 Akceptačné kritéria

- 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.

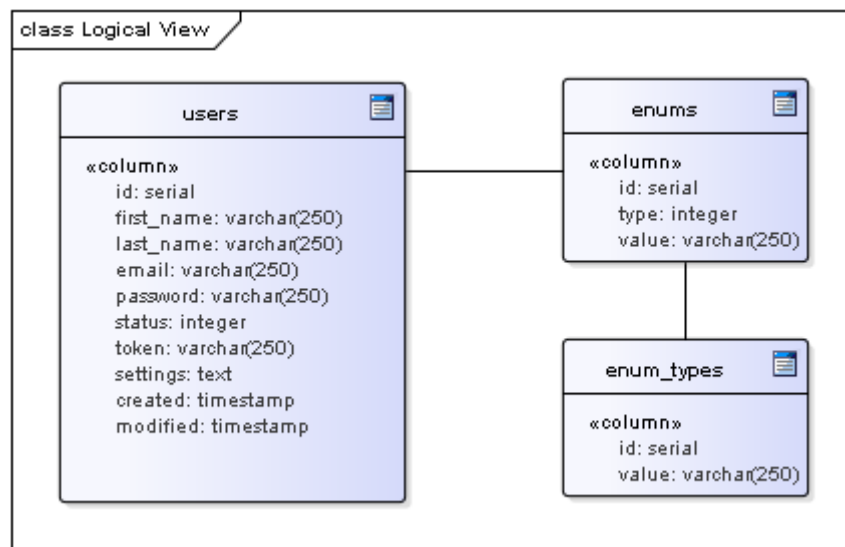
Čo sa týka backendovej validácie používateľa, tá je vykonávaná pomocou validačných funkcií v UserTable.php, kde sa pre každý typ formuláru zavolá iná validačná funkcia. Naviazanie na formulár a validačných funkcií prebieha v UserController.

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."

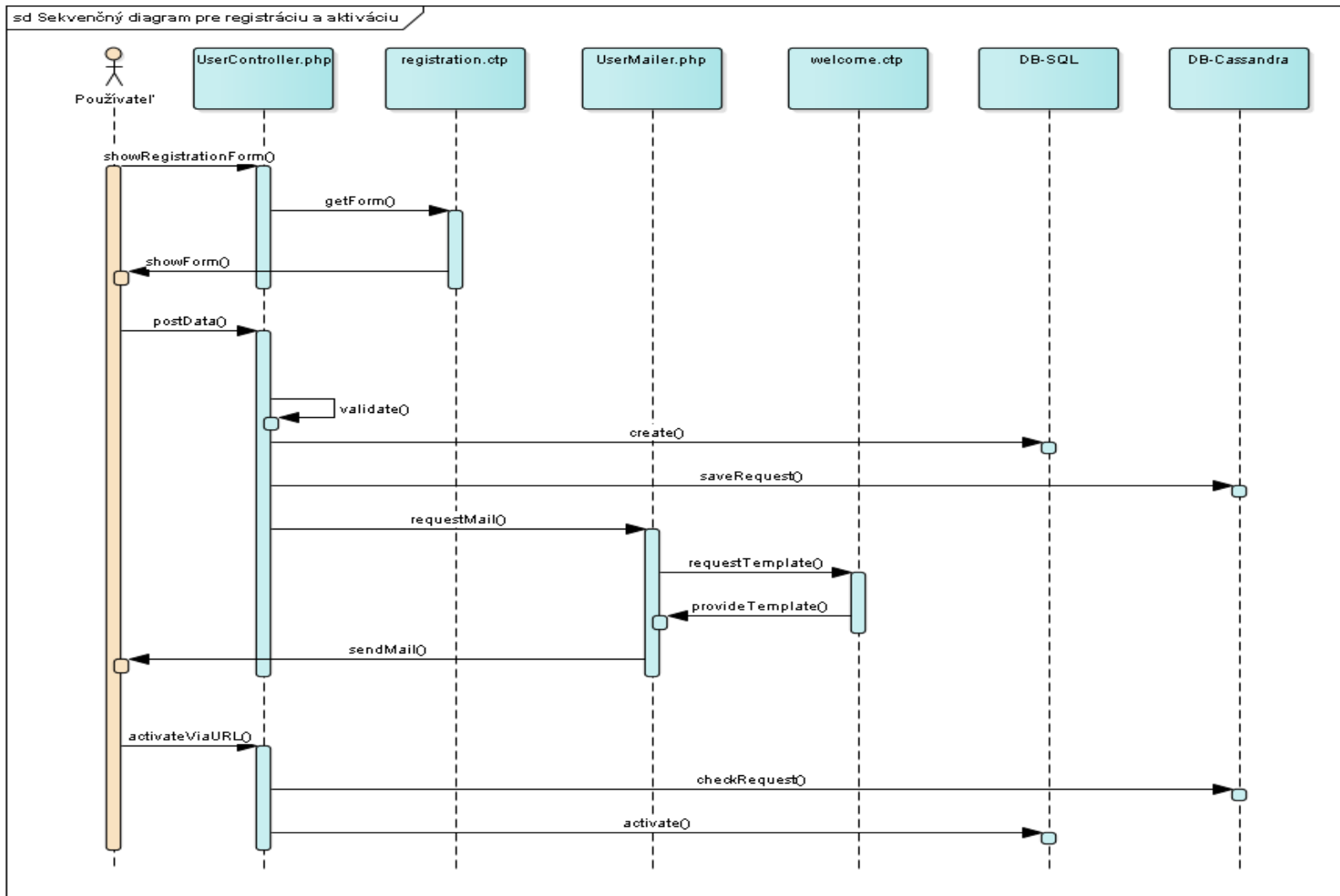
5.1.4.2.3 Databázový model

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



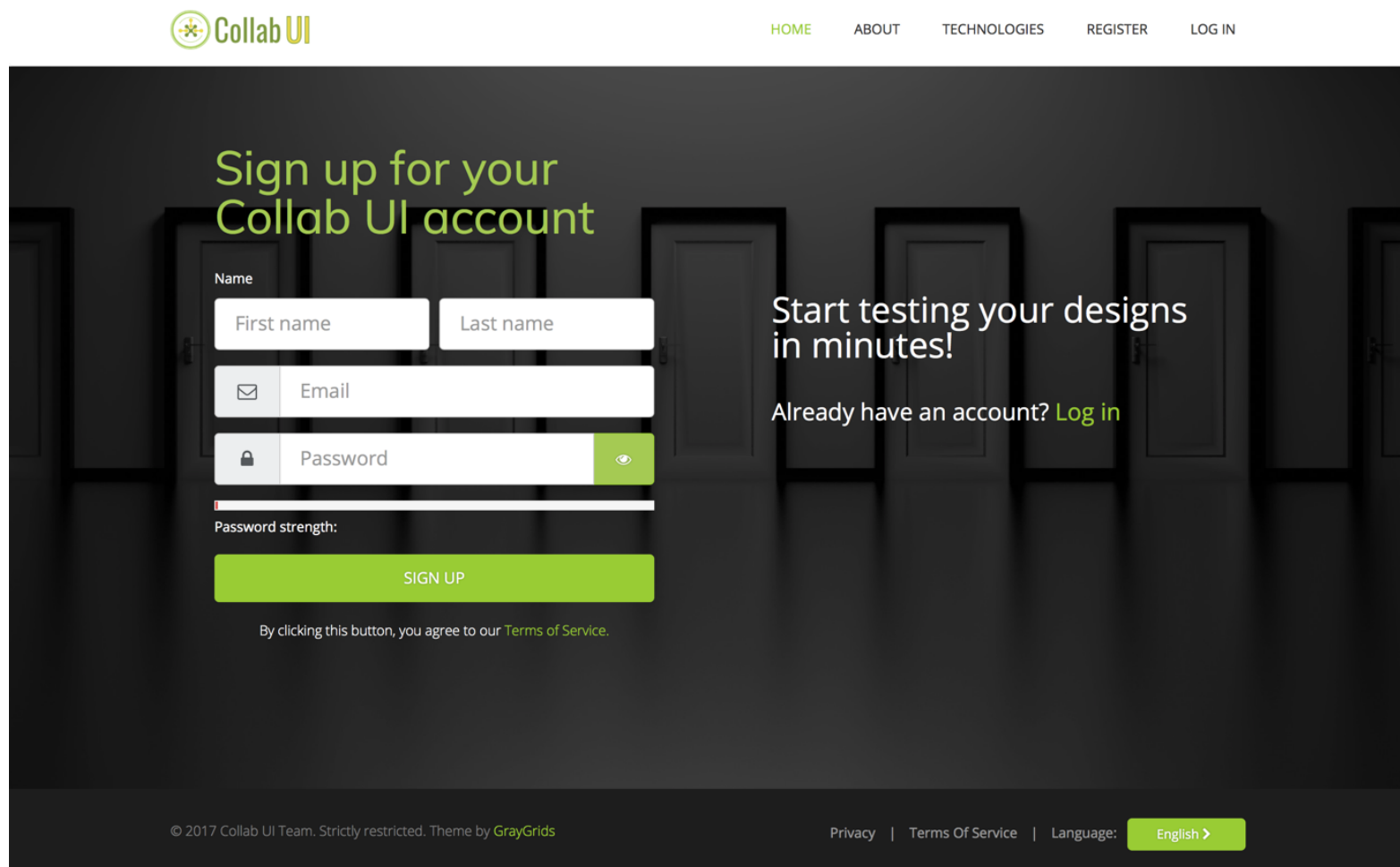
Obrázok 5 - Databázový model registráciu používateľa

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

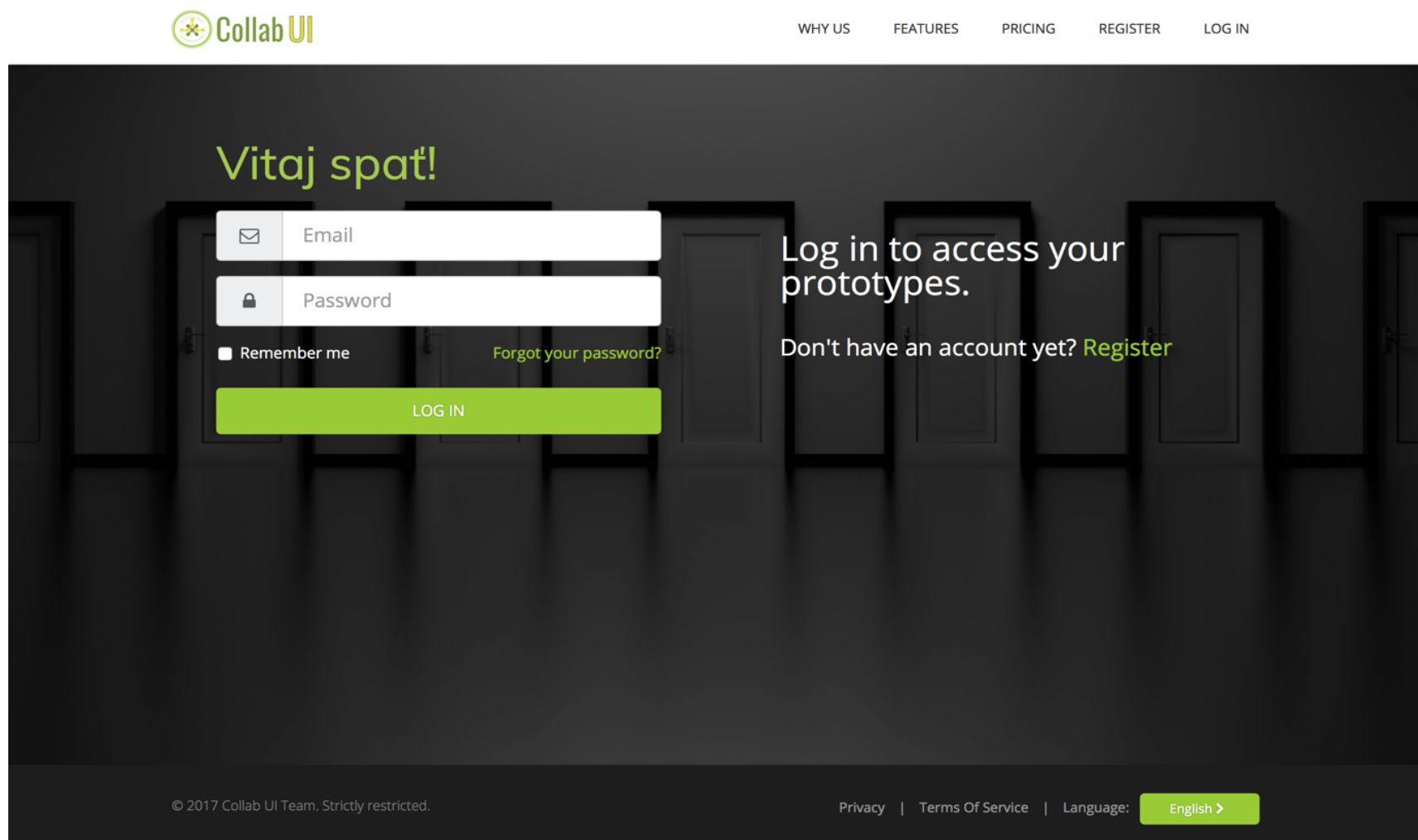
Čo sa týka frontendu, kontroluje sa či jednotlivé textové polia boli vyplnené alebo nie. V prípade, že vyplnené neboli tak sa pri príslušnom polí prostredníctvom animácie zobrazí modálne okno, ktoré používateľa upozorní o nevyplnení daného pola a vyzve ho k jeho vyplneniu.

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

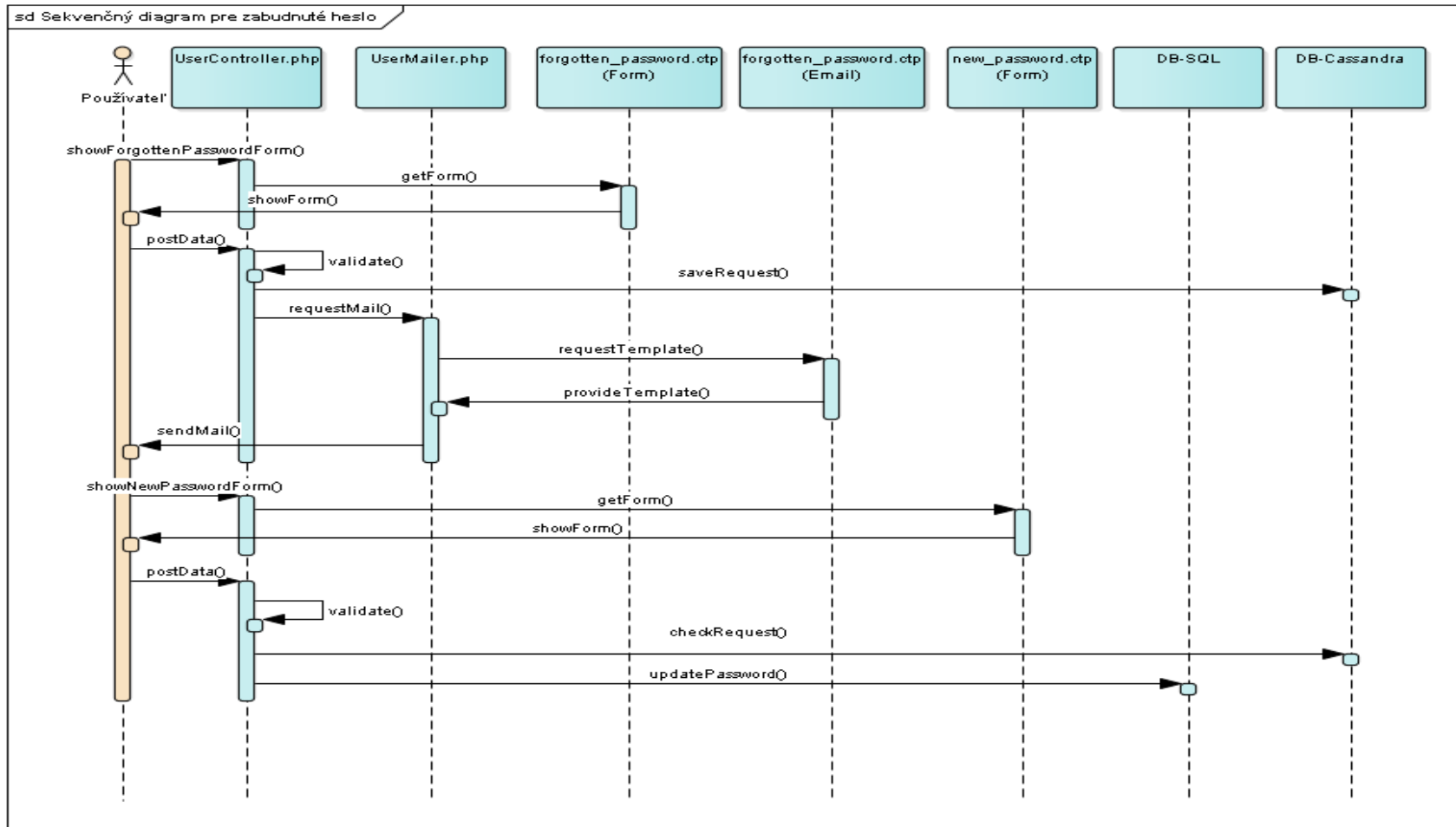
5.1.4.4.1 Akceptačné kritéria

- 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

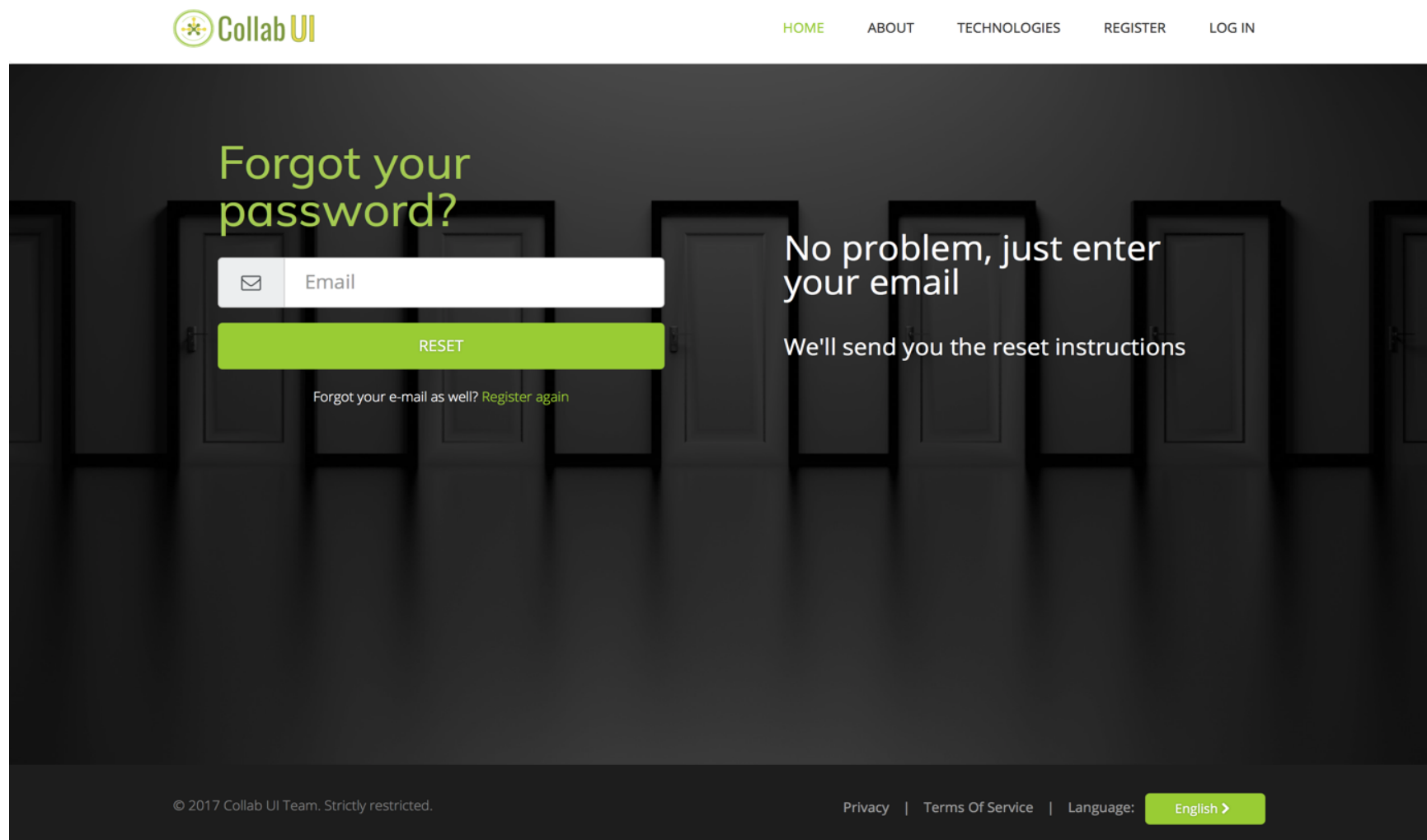
V rámci validácie zabudnutého hesla sa pri nastavovaní hesla kontroluje či je používateľ v systéme zaregistrovaný. Taktiež sa kontroluje email, z ktorého bol používateľ presmerovaný na podstránku zabudnutého hesla.

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

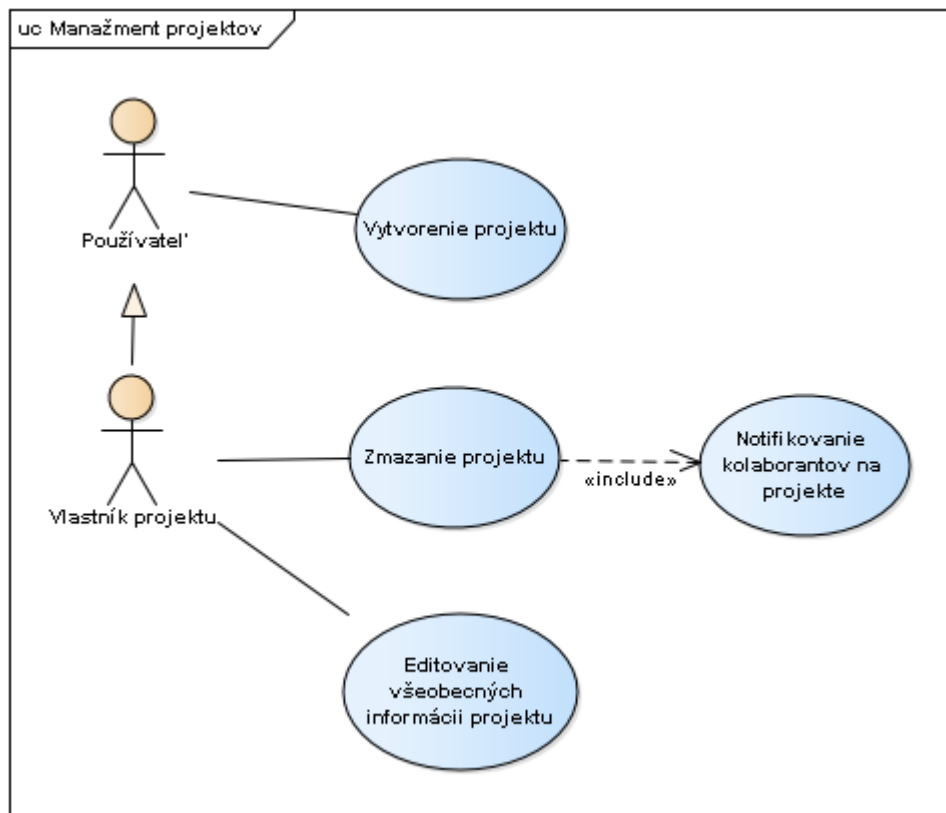
V rámci analýzy sme sa zameriavali rozvrhnutie projektu v rámci jednoduchosti používania a vytvárania projektov používateľom. Veľa krát môže byť pri webových aplikáciách problém práve v rozvrhnutí elementov tejto aplikácie. Ak používateľ nevie intuitívne nájsť riešenie na svoj aktuálny problém, dizajn 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 zvyknutí 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:

- Aoweb¹
- Crud plugin pre CakePHP²
- Bootstrap notify³
- Bootstrap modal⁴

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

AOWeb je aspektovo-orientovaný rámec, implementovaný v jazyku JavaScript, ktorý pomáha v oddelení záležitostí a písaní 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)

¹ <https://github.com/Ado20/aoweb>

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

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

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

5.2.4.1.1 Akceptačné kritéria

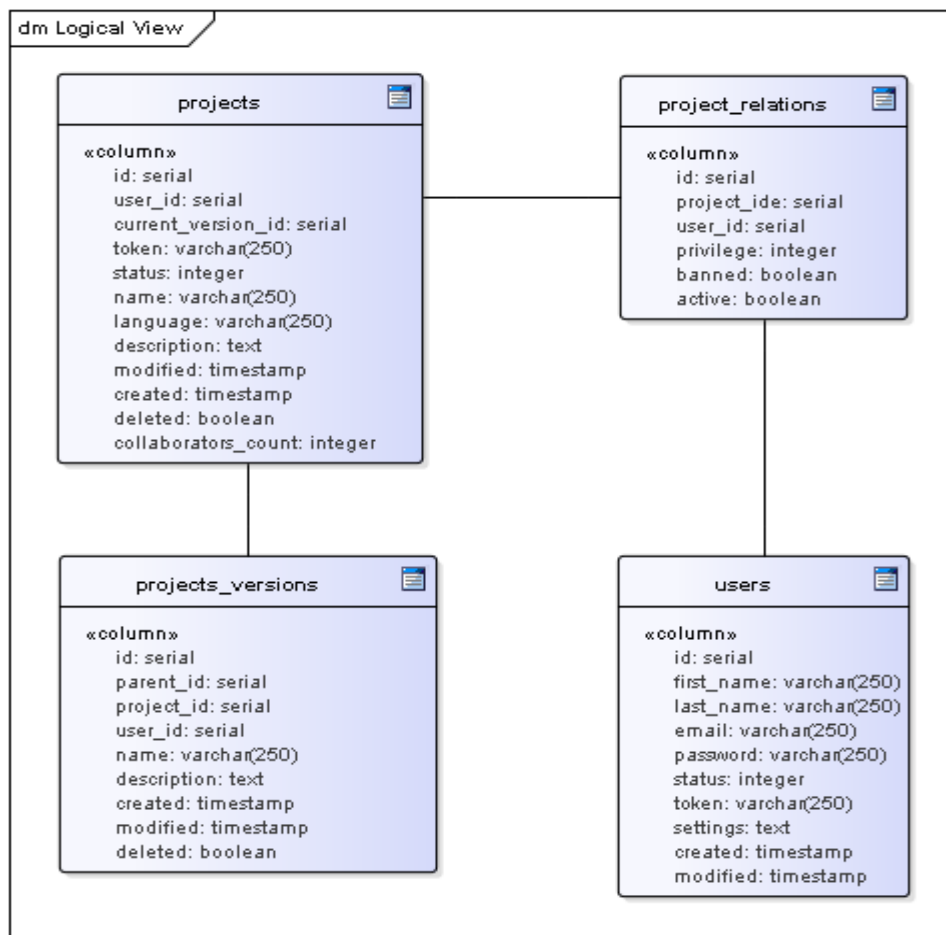
- Používateľ dokáže vytvoriť nový projekt vo verzii 1.0.0
- Po vytvorení projektu je presmerovaný na detail projektu
- Používateľ dokáže zmeniť základné informácie o projekte

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

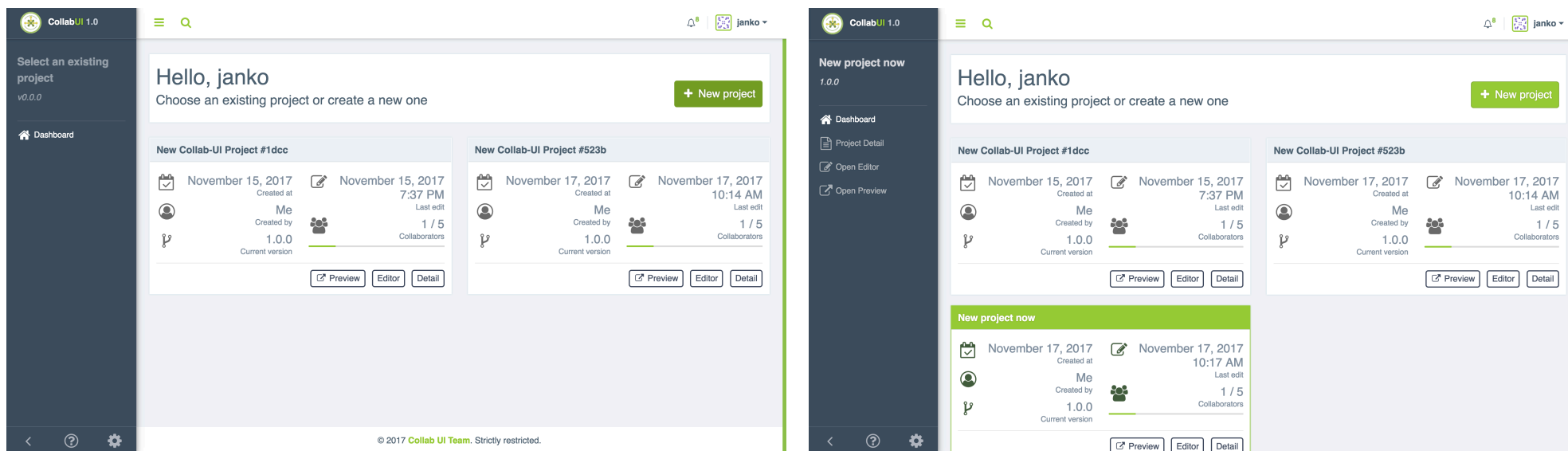
Čo sa týka databázového pohľadu, sa pri vytváraní projektu používajú nasledovné tabuľky.



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

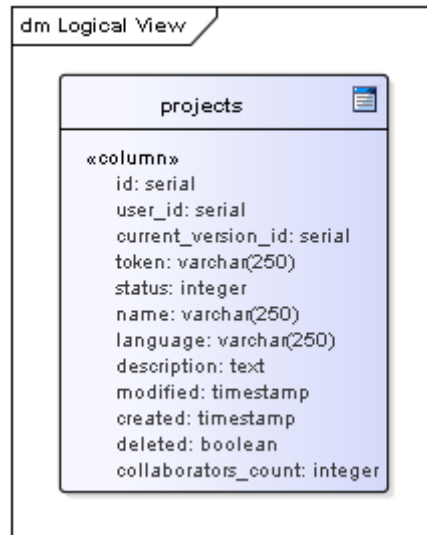
- Odstránenie projektu musím najskôr potvrdiť
- Dokáže používateľ odstrániť projekt
- Po odstránení bude presmerovaný na Dashboard a notifikovaný o úspešnej operácii
- Po vymazaní projektu sa projekt (a jeho relations) reálne nezmaže ale sa len označí v DB ako deleted.
- Po stlačení tlačidla musí byť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

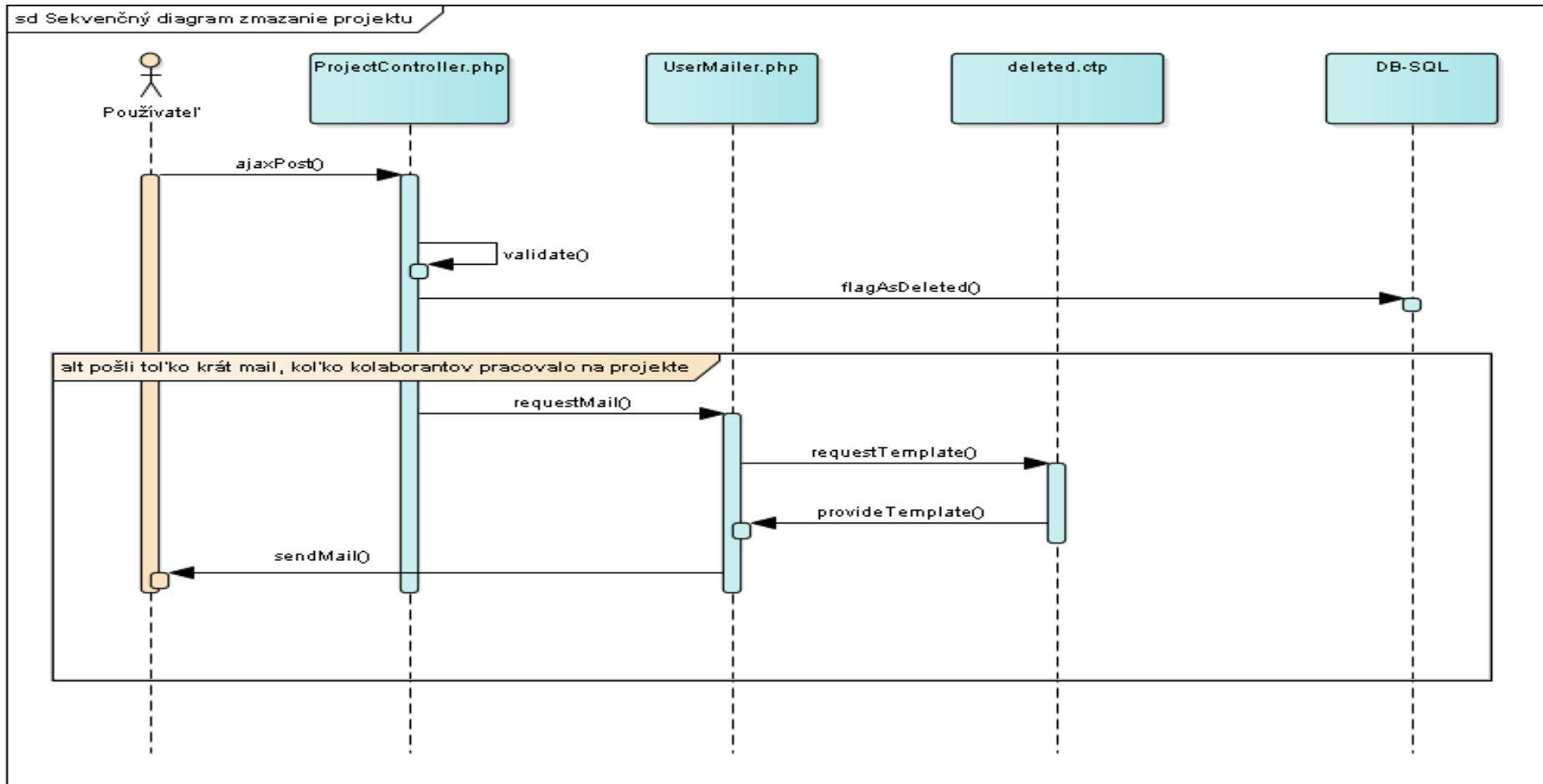
Čo sa týka databázového pohľadu, sa pri mazaní projektu používa iba tabuľka projects, v ktorej sa nastaví flag pre mazanie projektu "deleted" na true. User relation v tabuľke UserRelations sa ale nezmaže.



Obrázok 14 – V rámci mazania projektov sa používa projects tabuľka

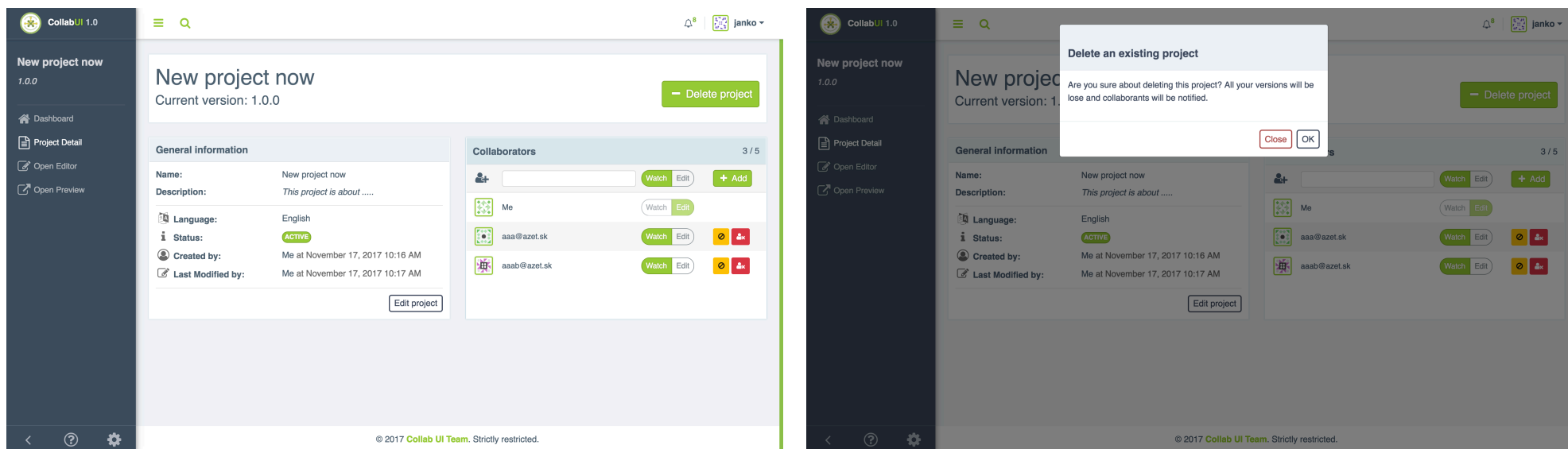
V rámci budúcej práce by sme chceli zintegrovať databázový cron, ktorý by každý deň o polnoci prešiel všetky záznamy, záznamy s flagom deleted by zarchivoval a v poslednom kroku vymazal z databázy.

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 späť 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

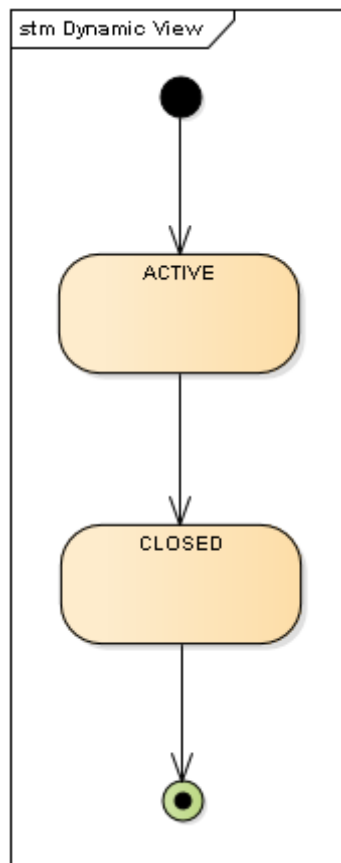
- Kontrola emailu a hesla pri vyplnení jednotlivých polí používateľom
- Možnosť nastavenia statusu na aktívny alebo zatvorený
- Nastavenie viacerých jazykov
- Rozdelenie projektu do samostatného boxu
- Uloženie zmien v projekte po potvrdení akcie
- Pri zrušení akcie neuložiť vyplnené polia ale ponechať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

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:

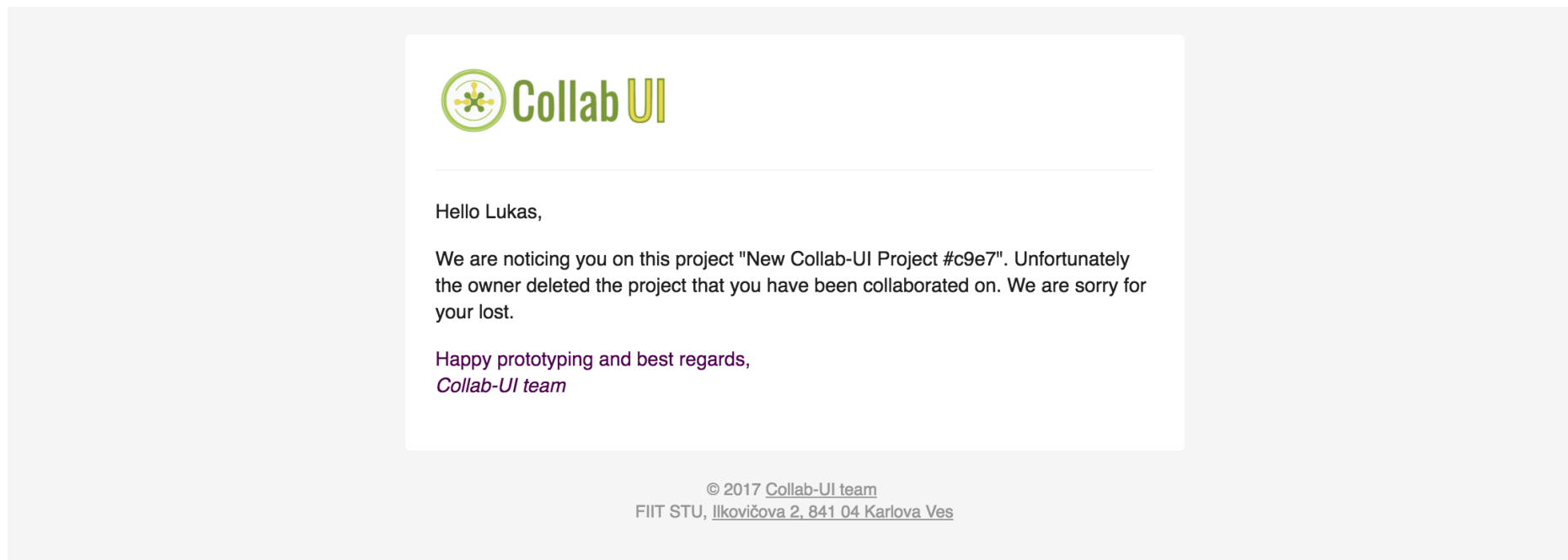
- Uložiť
- Zrušiť

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

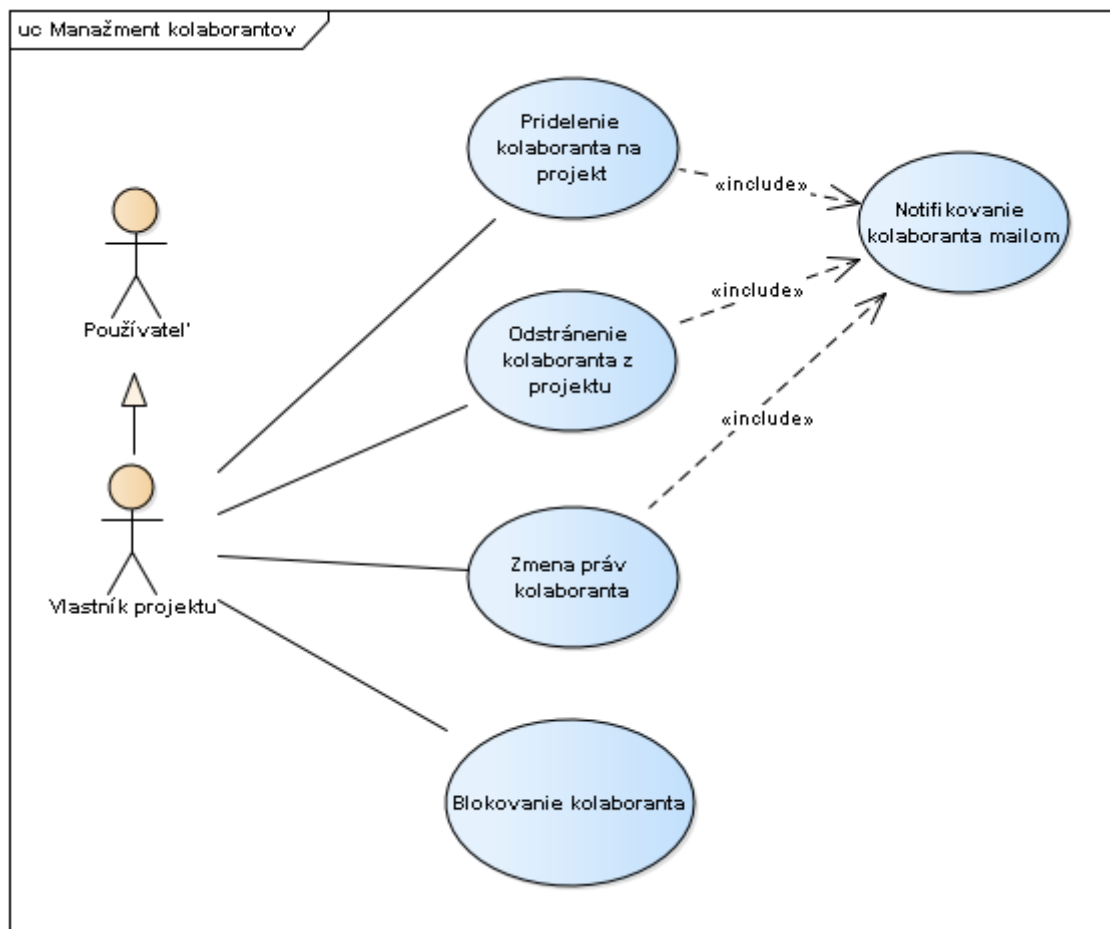
V rámci analýzy sme riešili hlavne akým štýlom bude daný kolaborant pridaný do projektu. Stále sme sa snažili dodržiavať nezložité grafické prevedenie s očakávanou funkcionalitou. Rozhodli sme sa preto rozdeliť kolaborantov a projekty do osobitných boxov. Tento fakt sme už spomínali aj pri vytváraní 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 vlastníkovi projektu mohlo ponúkať členov tímu, ktorých už niekedy pridával. Všetky akcie ktoré sa týkajú správy kolaborantov by sa mohli vykonávať v spomínanom boxe. Teda aj zabanovanie aj odstránenie by sa mohlo nachádzať pri každom kolaborantovi aby používateľ mohol rýchlo rozhodnúť o jeho budúcej aktivite. Taktiež používanie modálnych okien a notifikácií 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. Prípad použitia končí

Alternatívny tok:

- 1.a1 Vlastník projektu zvolí možnosť odblokovania kolaboranta
- 1.a2 Systém odblokuje kolaboranta
- 1.a3 Pokračuje s bodom 3

UC Notifikovanie kolaboranta mailom

Hlavný tok:

1. Systém notifikuje vybraného kolaboranta prostredníctvom emailu
2. Prípad použitia končí

5.3.4 Implementácia

5.3.4.1 Pridelenie kolaboranta na projekt

Pridelenie kolaboranta je funkcia, ktorú bude 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

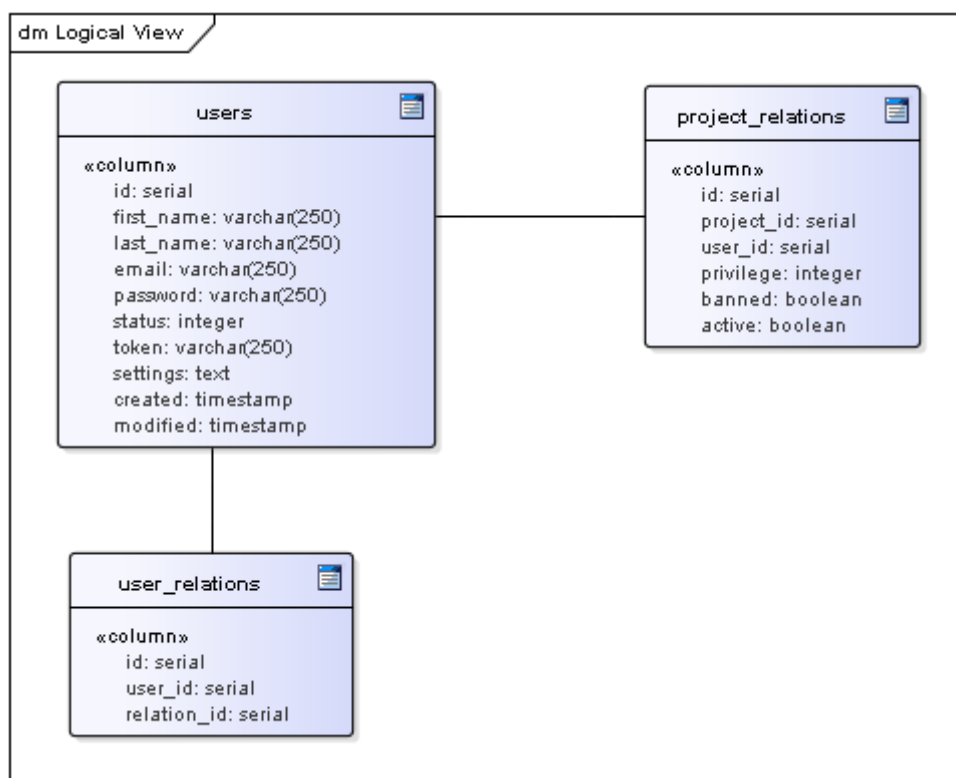
Celá validácia pridania nového kolaboranta do projektu prebieha na backende.

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.



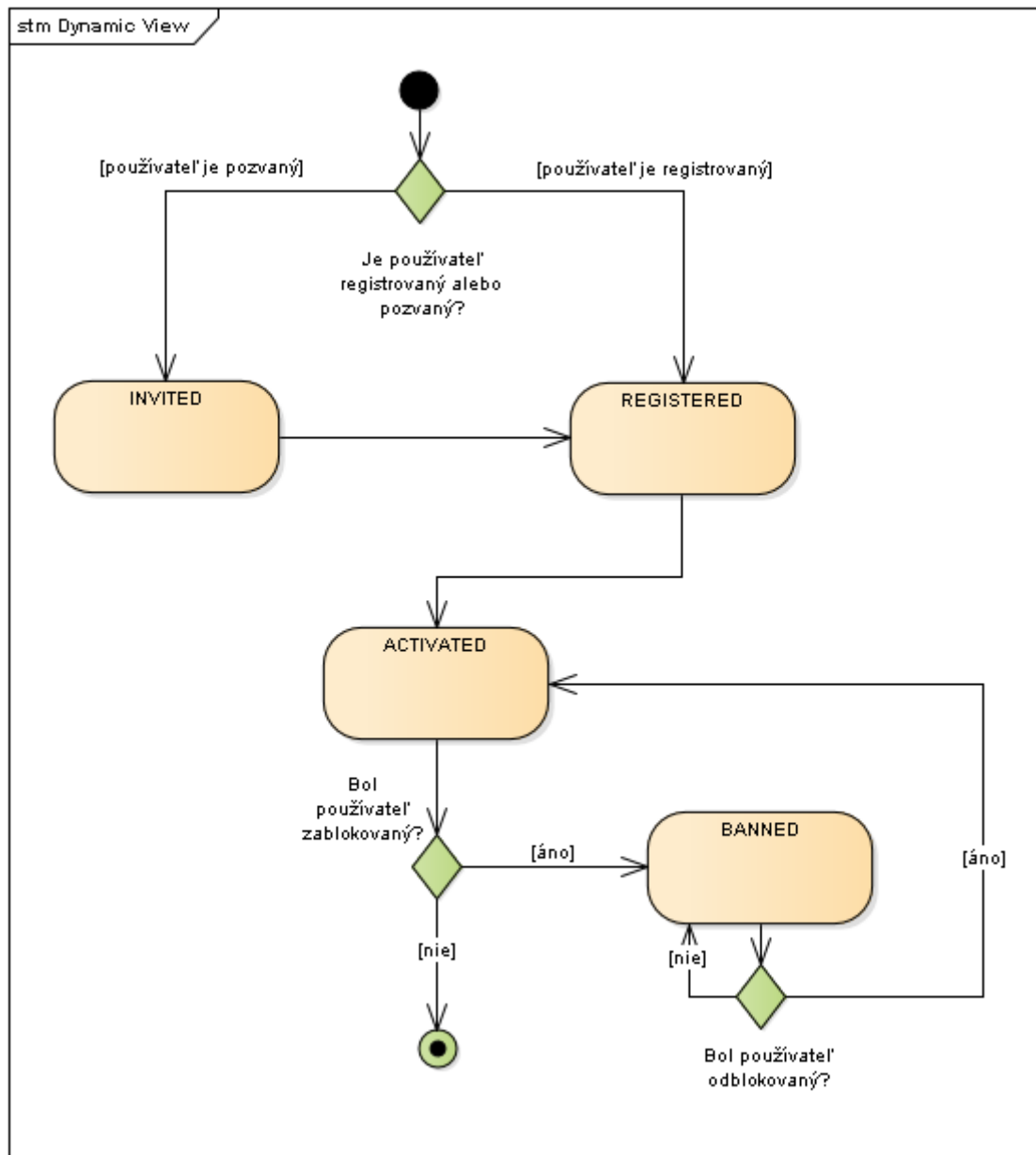
Obrázok 21 - DB Model pre pridelenie kolaboranta na projekt

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

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.

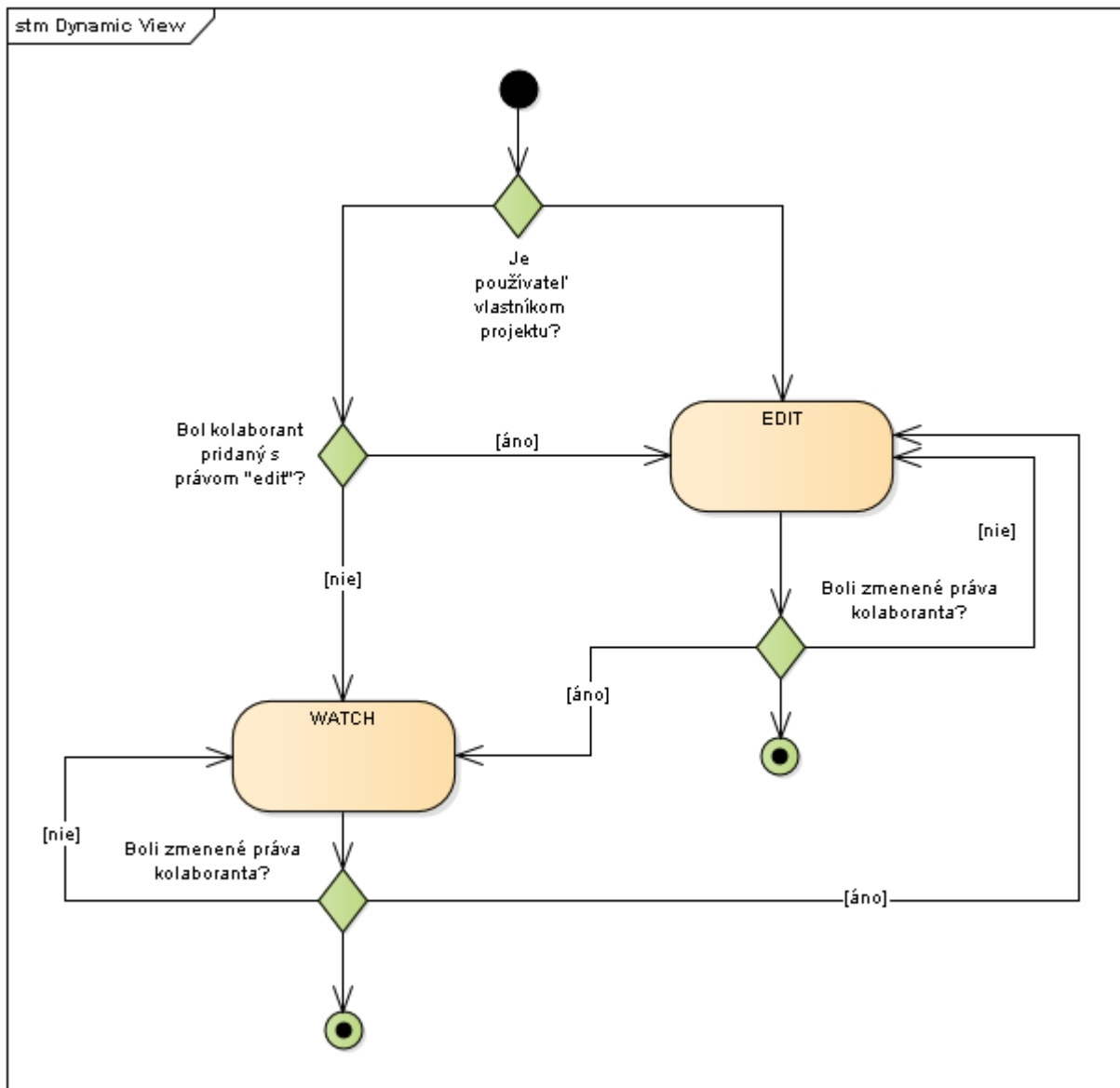
5.3.4.2.1 Akceptačné kritéria

- 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

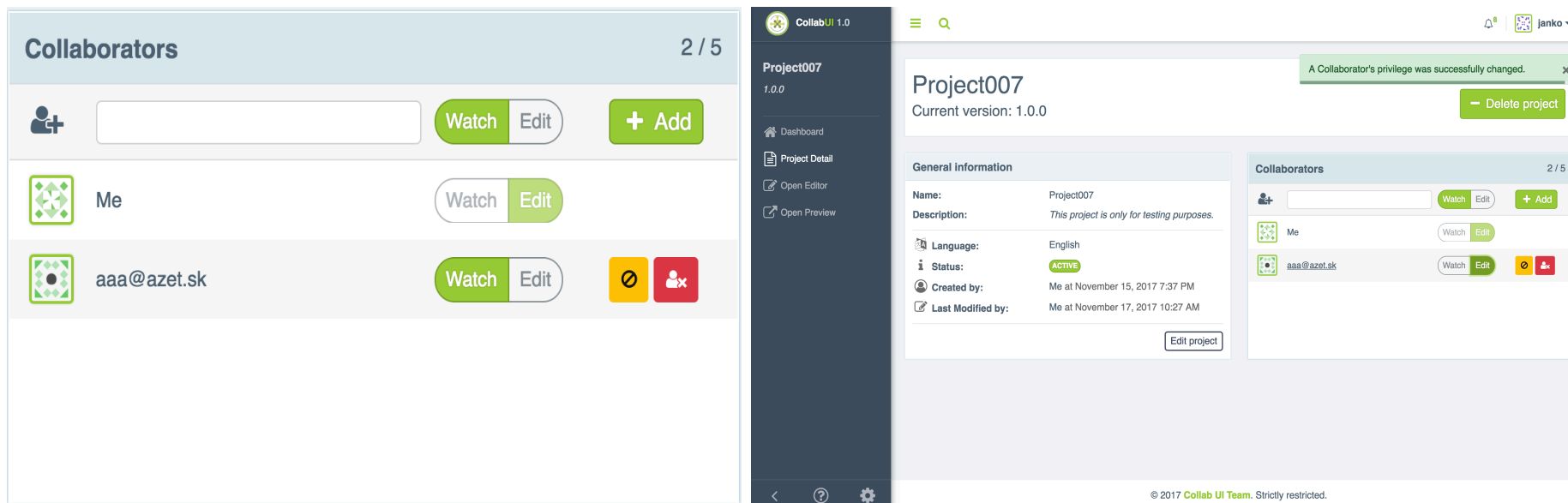
Jediná validácia, ktorá sa vykonáva pri tejto akcii je kontrola či je používateľ, ktorý chce zmeniť práva kolaboranta, naozaj aj vlastník projektu a teda či má patričné práva pre zmenu jeho práv.

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 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.

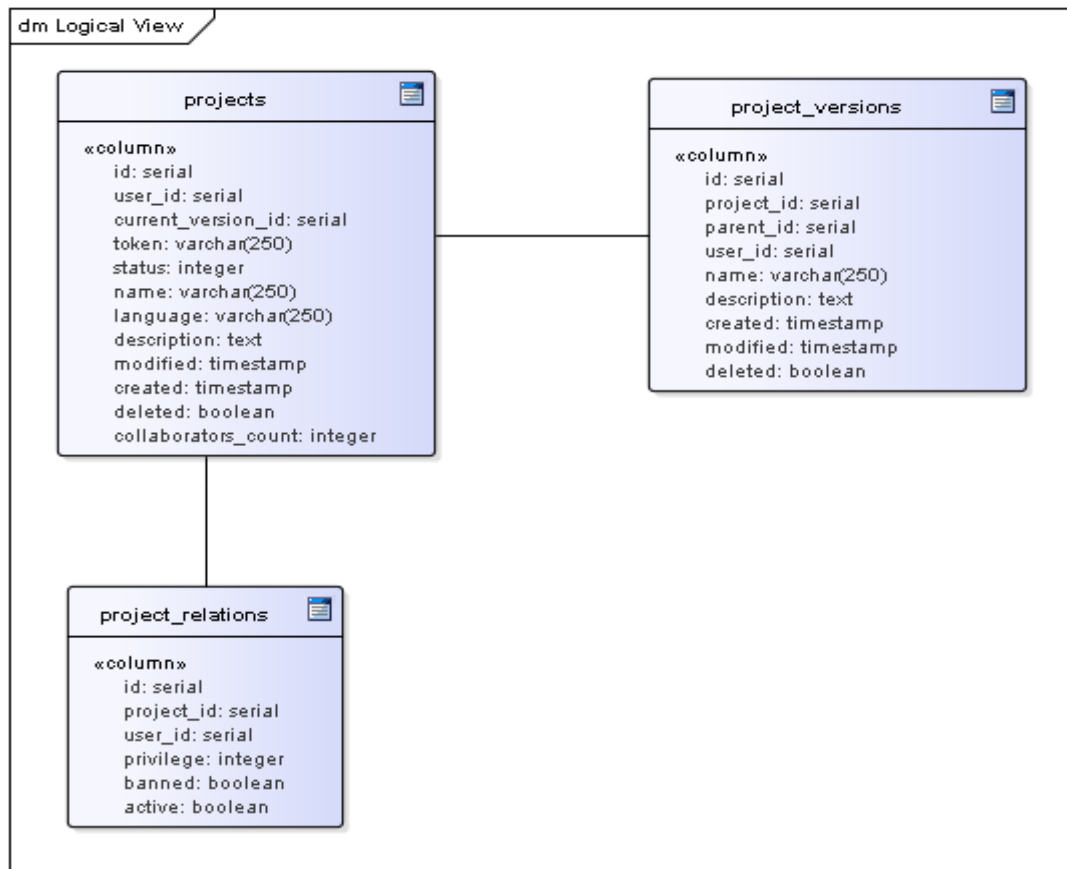
5.3.4.3.1 Akceptačné kritéria

- 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.

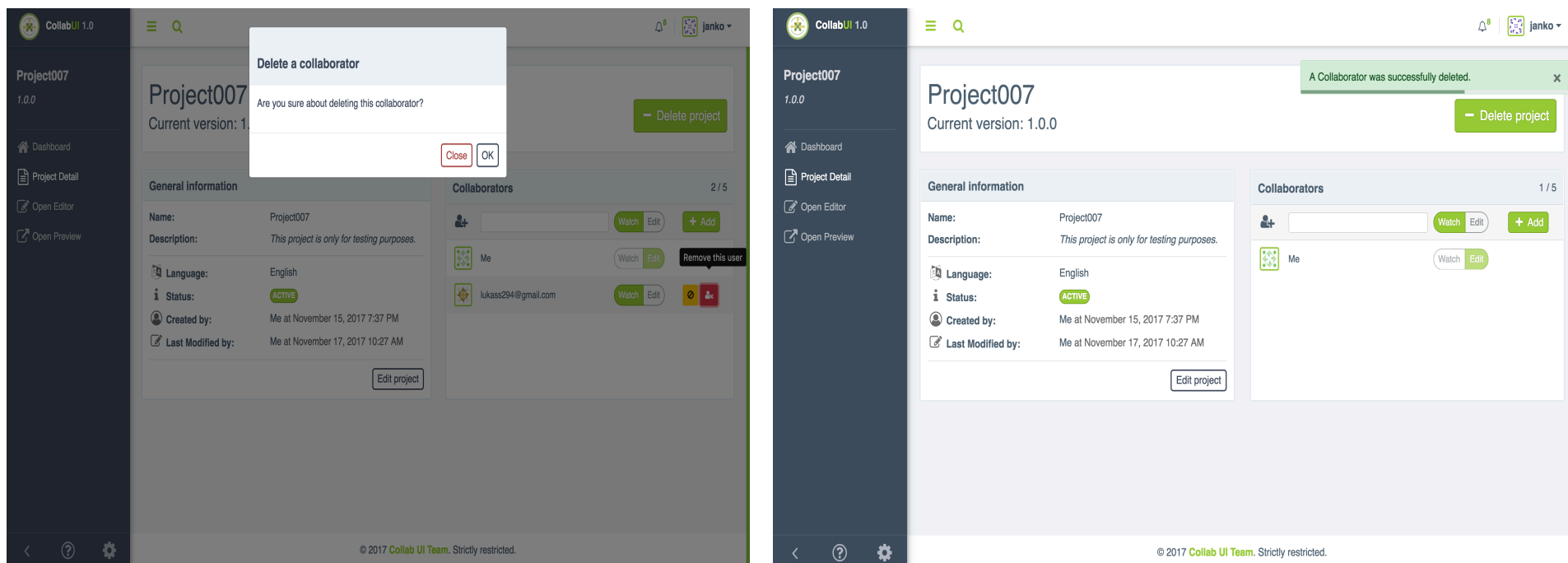
5.3.4.3.3 Databázový model



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

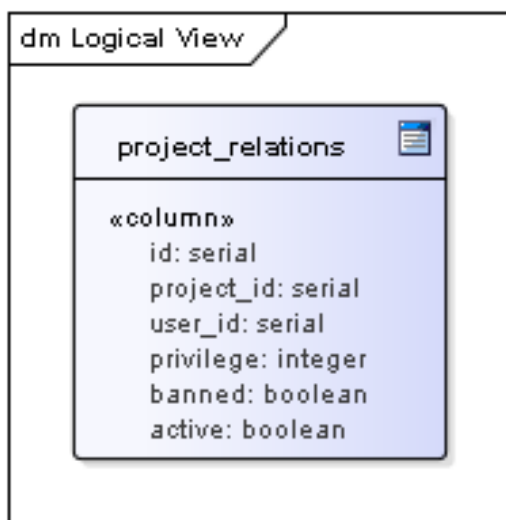
- Blokovanie môže byť vykonávané iba vlastníkom projektu
- Blokovaný kolaborant nesmie byť notifikovaný o jeho aktuálnej zmene práv
- Kolaborant nemá ďalej sprístupnený projekt, na ktorom pracoval

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

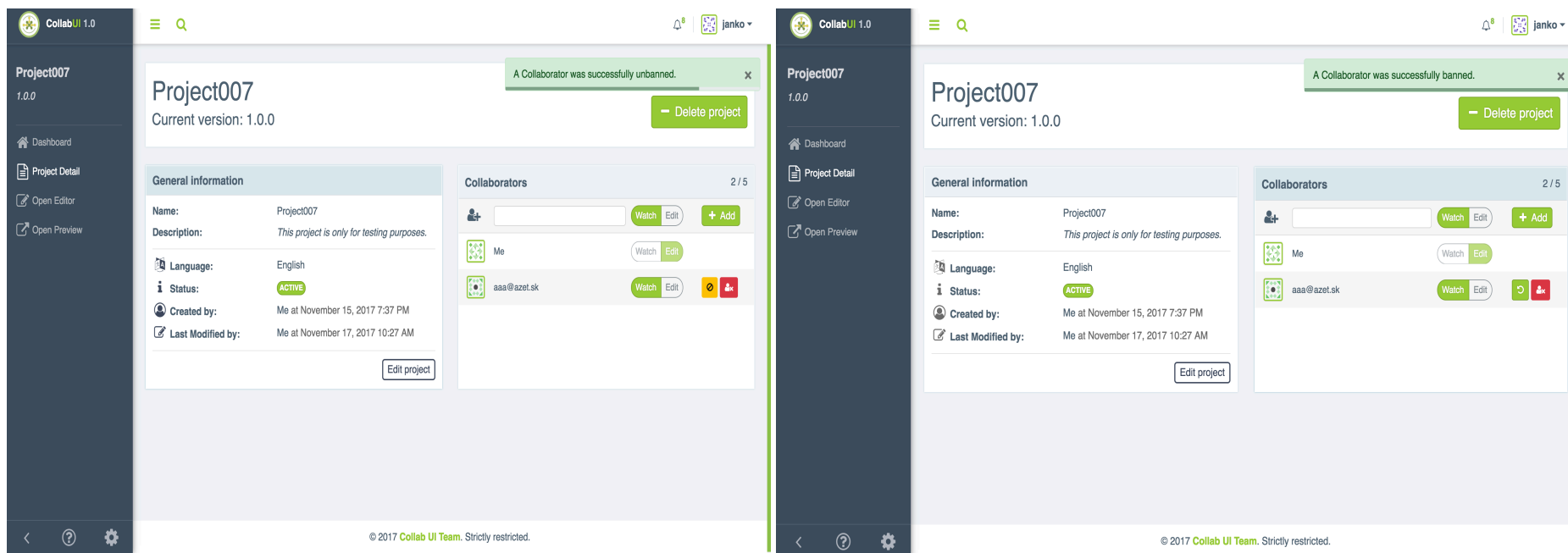
Pri tejto akcii sa využíva iba jediná tabuľka a tou je `project_relations`.



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.

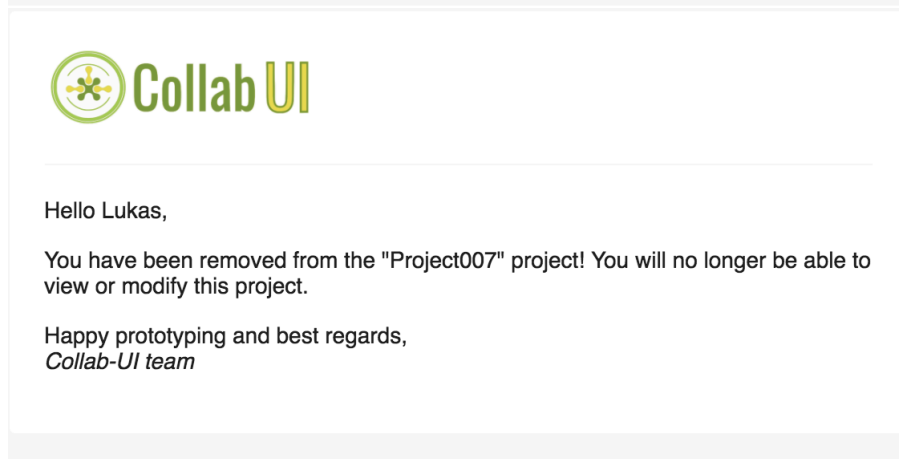
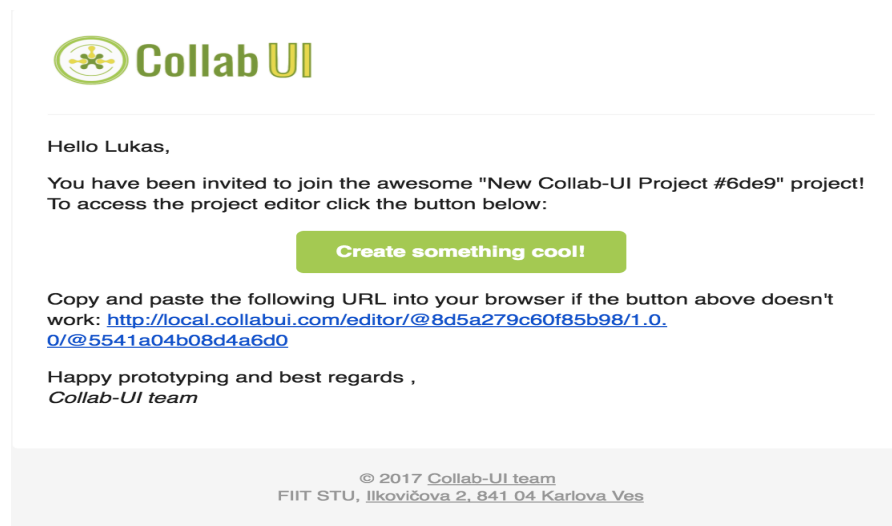
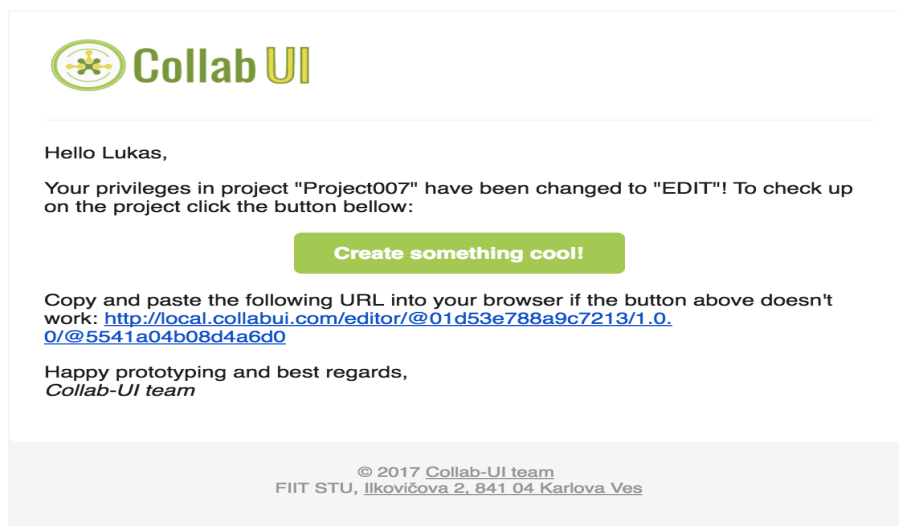
5.3.4.5.1 Akceptačné kritéria

- 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

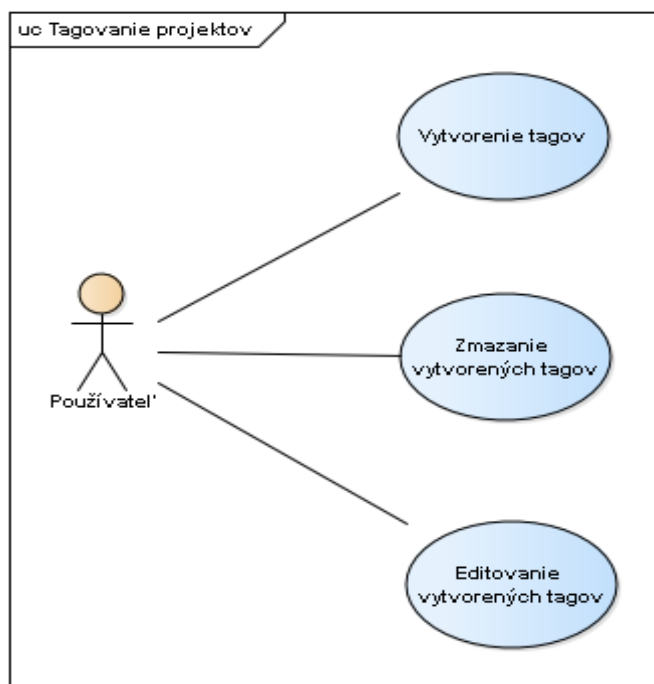
V rámci analýzy sme dospeli k záveru, že kvôli narastajúcemu počtu vytvorených projektov je potrebné dopracovať zefektívnenie vyhľadávania, ktorej realizáciu sme zamýšľali príliš jednoduchú. Rozhodli sme sa teda riešenie postaviť na otagovaní projektov.

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

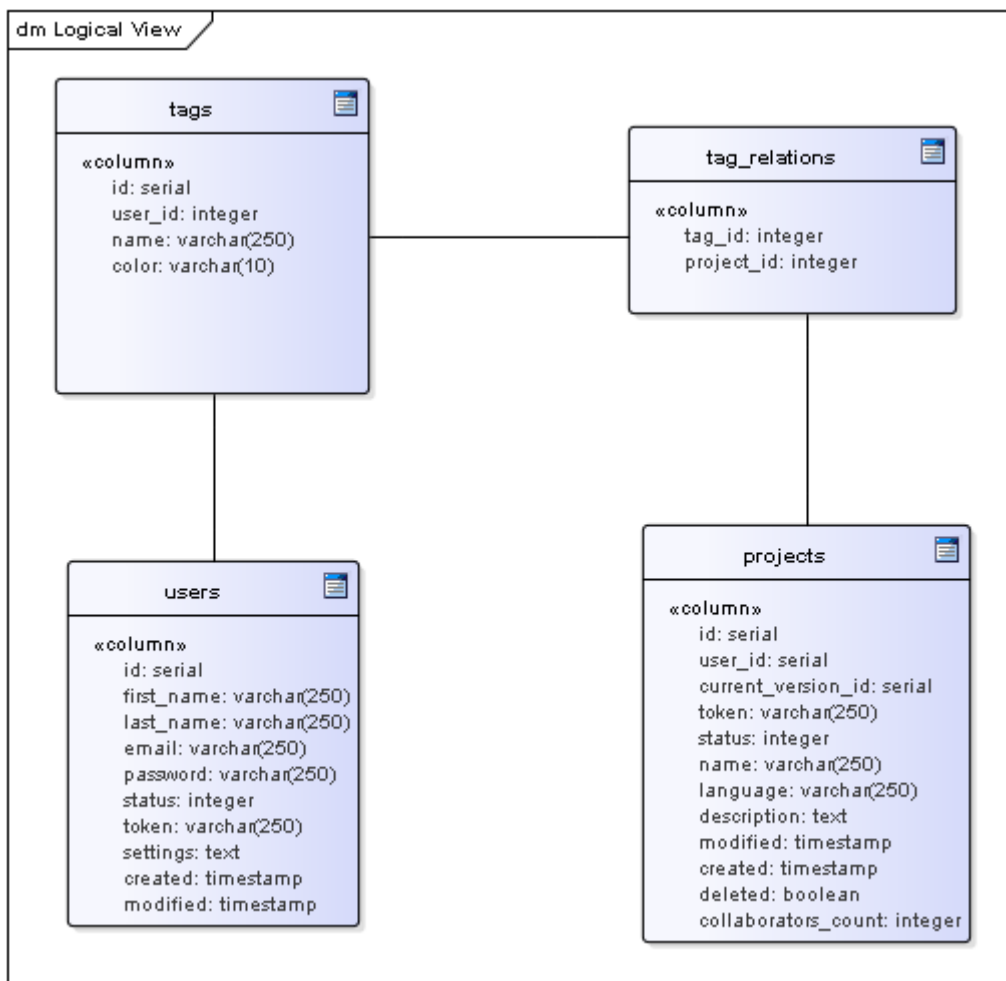
5.4.4.1.2 Validácia

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 ukladania a práce 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.

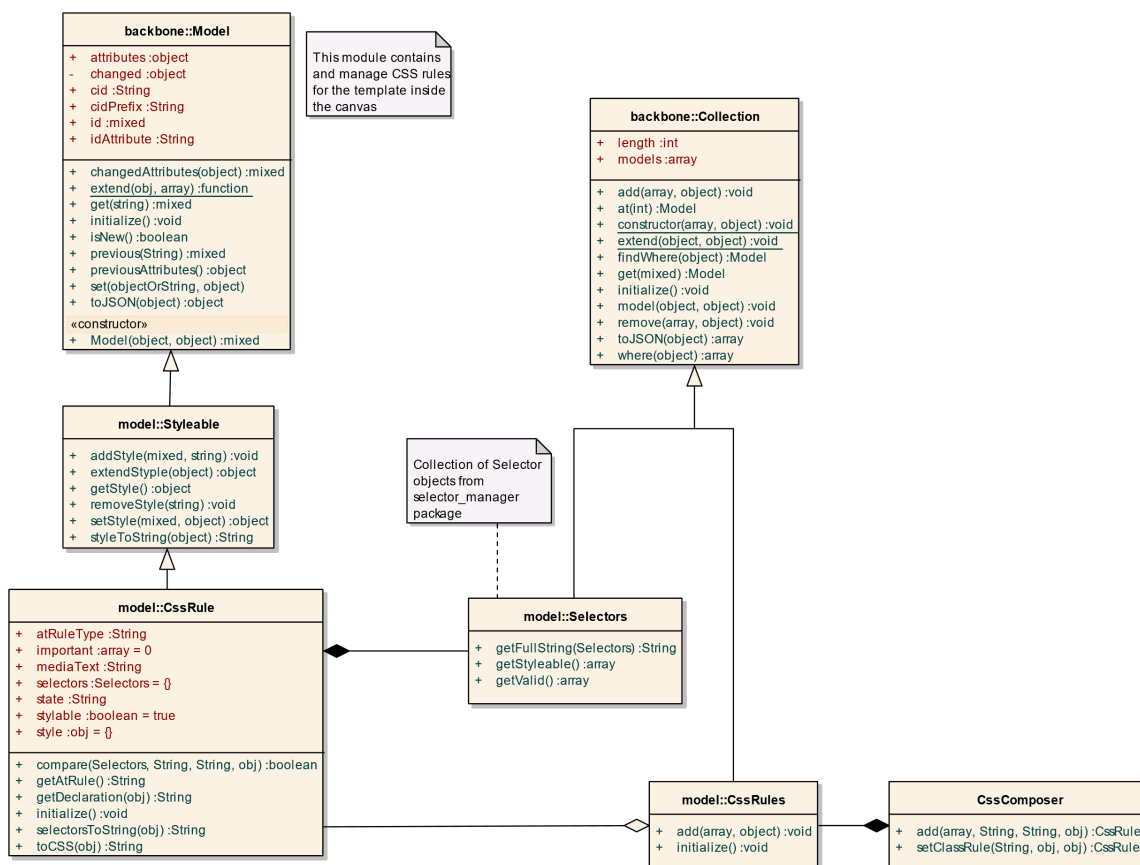
Analýza riešenia kolaborácie v reálnom čase vyžadovala hlbšie ponorenie sa do technickej štruktúry podporného nástroja pre tvorbu stránok, ktorej opis sa nachádza v nasledujúcich častiach.

5.5.3 Analýza podporného nástroja pre tvorbu stránok

K pochopeniu vnútorného fungovania podporného nástroja pre tvorbu stránok⁵ 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.

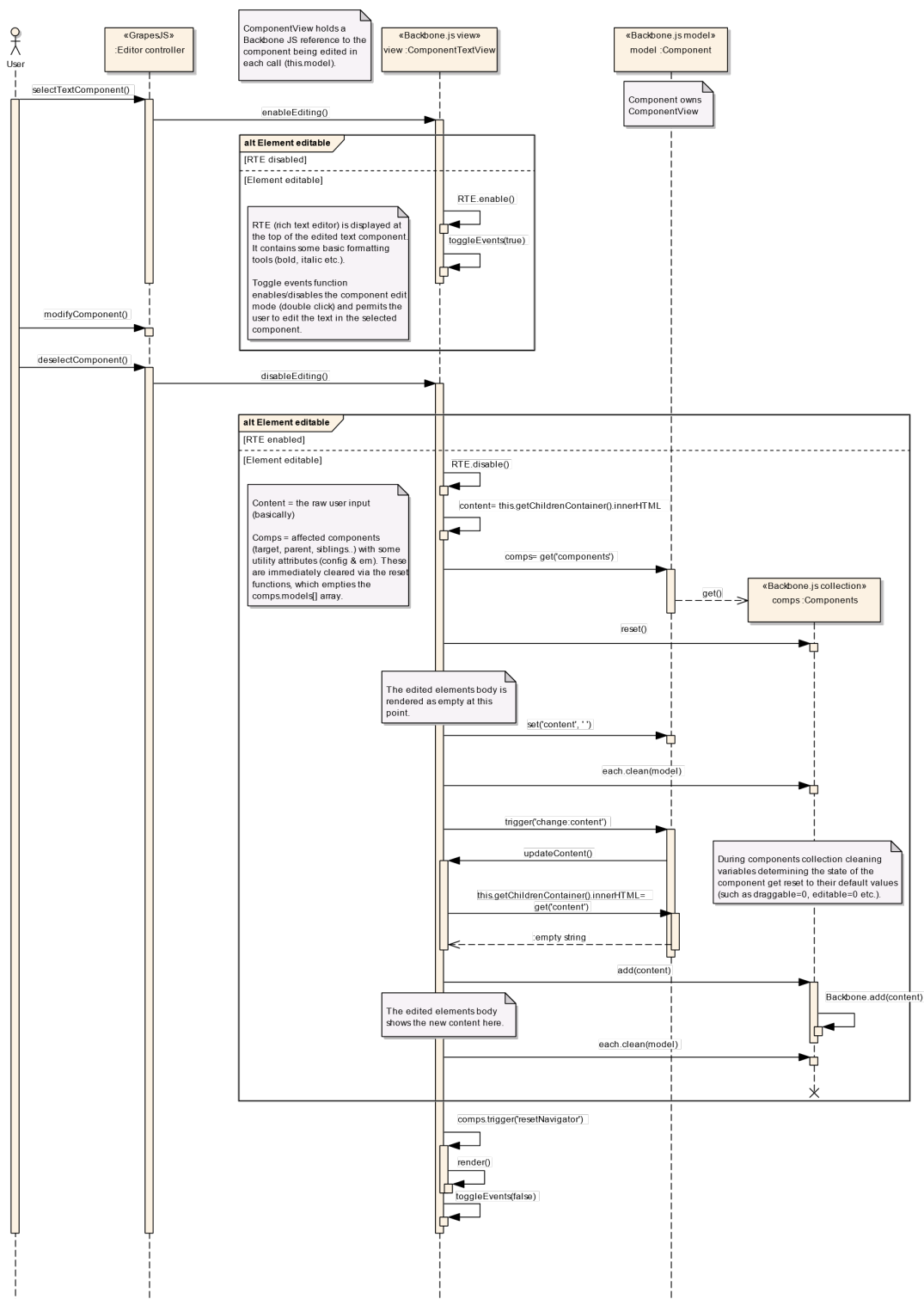


Obrázok 34 - Diagram tried pre CSS Composer

⁵ <http://grapesjs.com/>

5.5.3.2 Sekvenčný diagram, zmena textového komponentu

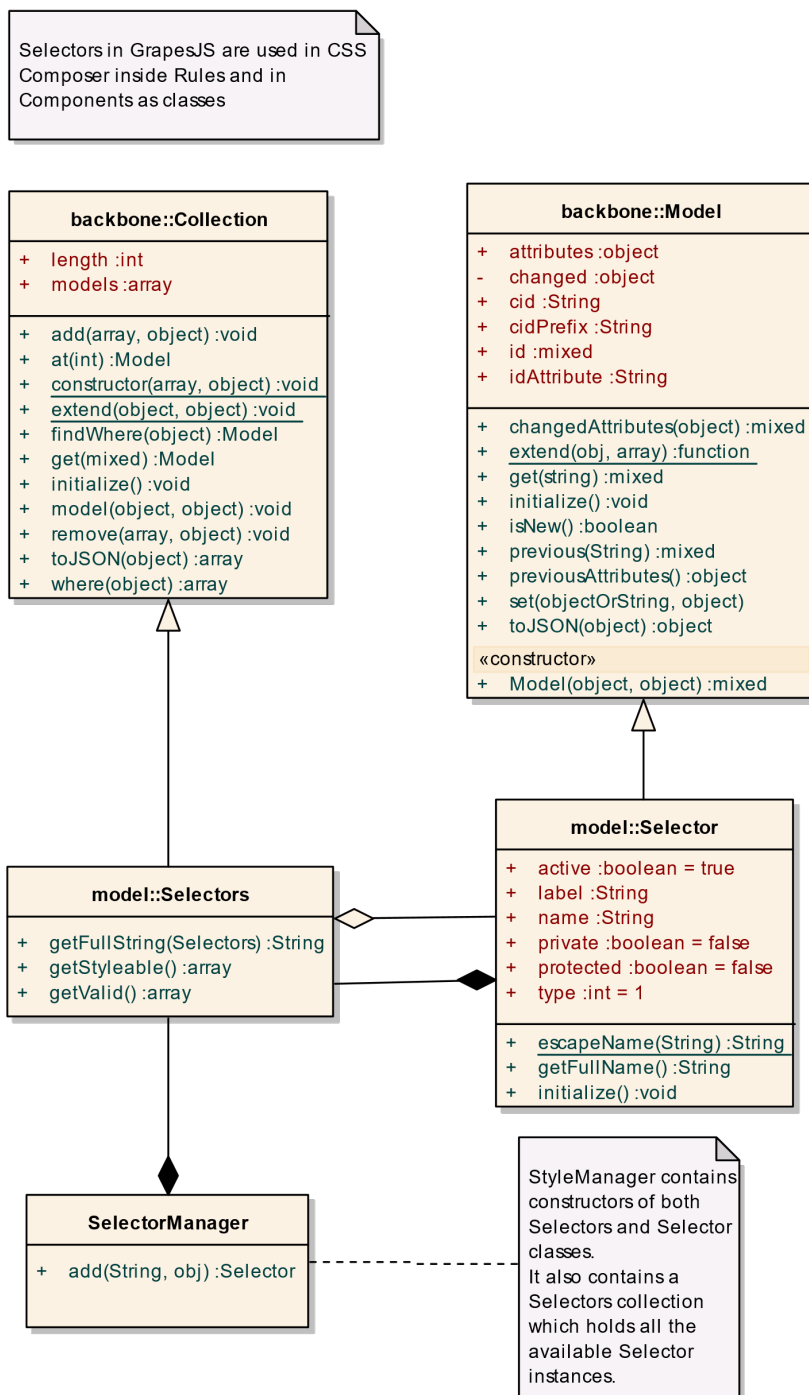
Sekvenčný diagram, ktorý zachytáva tok udalostí pri zmene textového komponentu.



Obrázok 35 - Sekvenčný diagram pre zmenu textového komponentu

5.5.3.3 Diagram tried, manažér selektorov

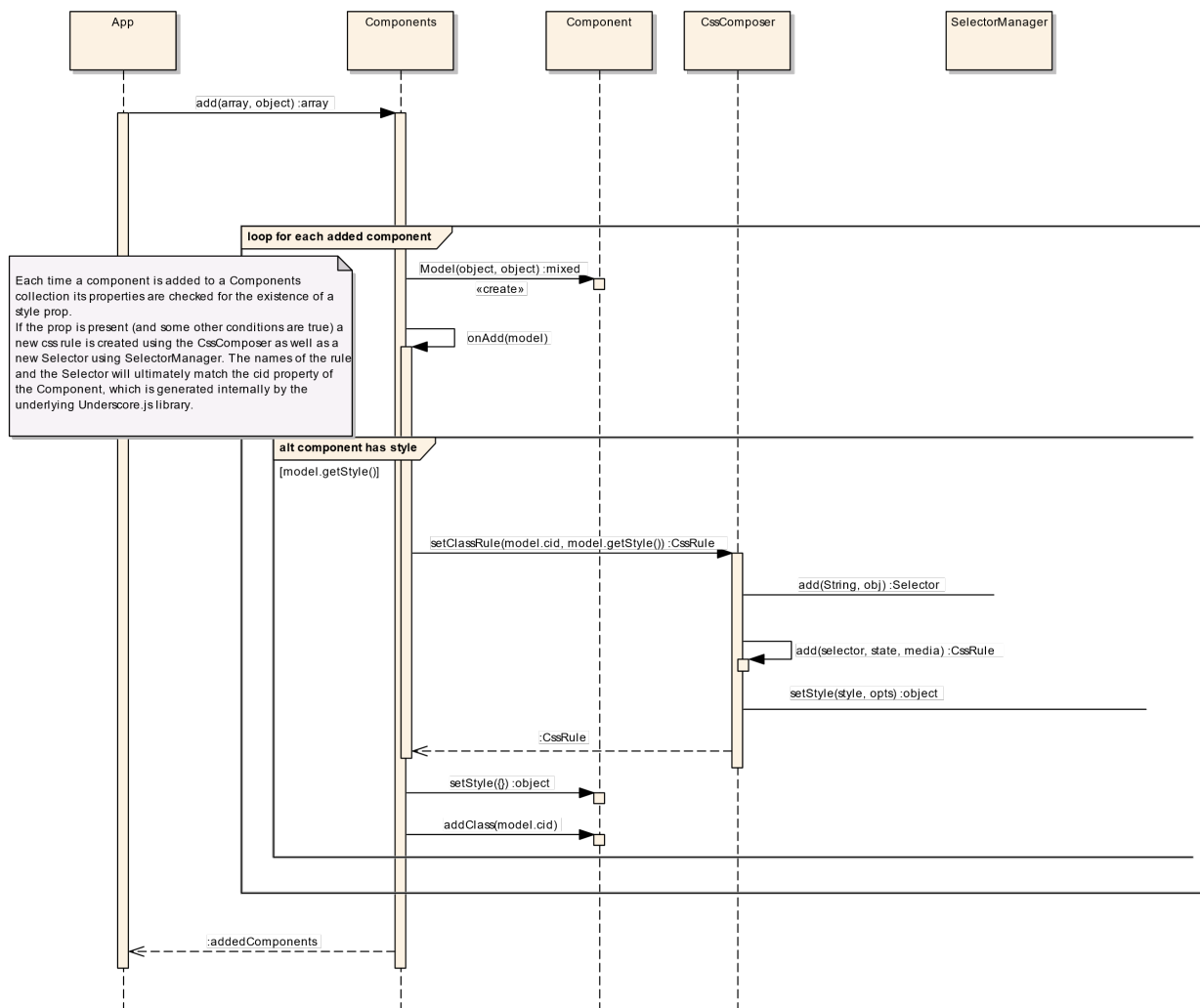
Diagram tried selektorov komponentov, ktorý zachytáva vzťah Backbone.js a GrapesJS. Rovnako ako pri CSS composer aj tu GrapesJs využíva Backbone.js na prácu s modelmi.



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

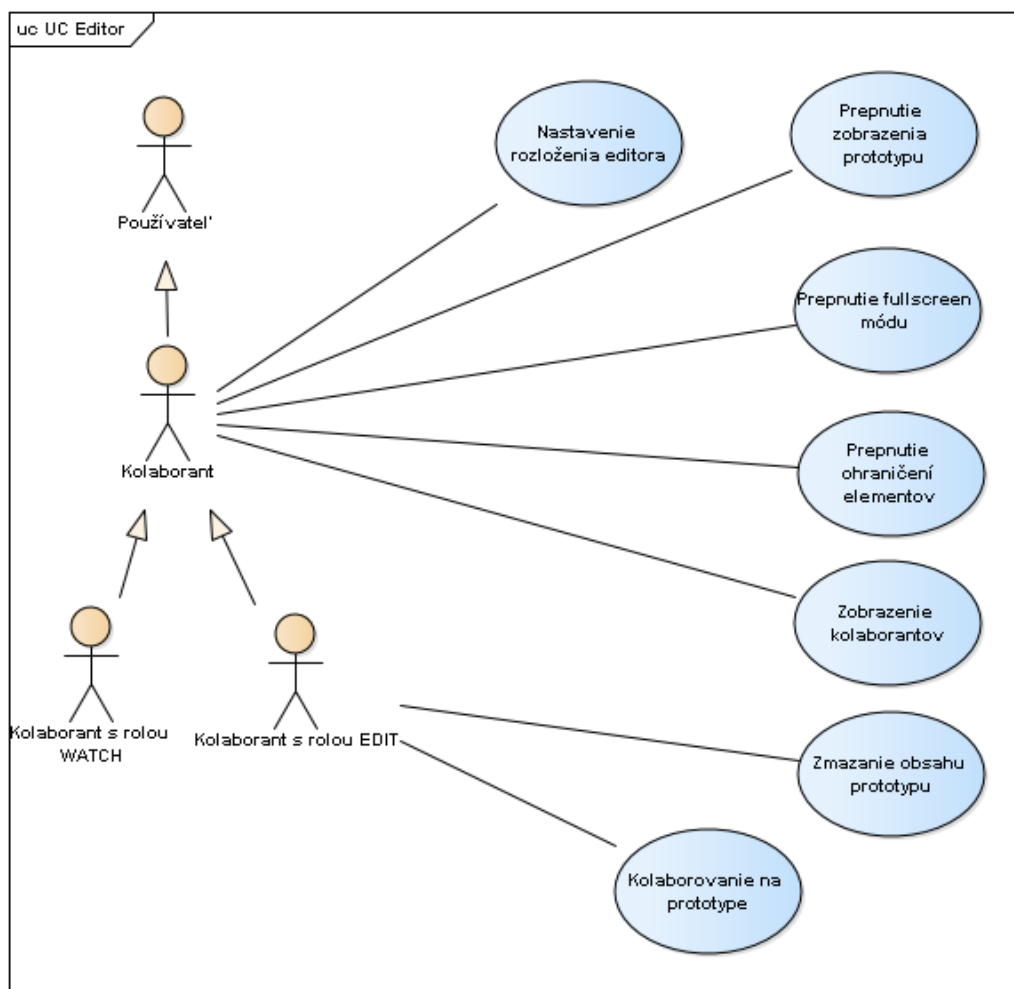
Diagramy, ktoré sme analyzovali nám v niektorých častiach pomohli k úspešnej implementácii. Treba však zdôrazniť, že samotné zakreslenie a analyzovanie diagramov je časovo veľmi náročné a preto je najlepšie zachytiť diagramom len tie najproblematickejšie časti. V rámci projektu sme chceli zakresliť ešte 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 ohraňení 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

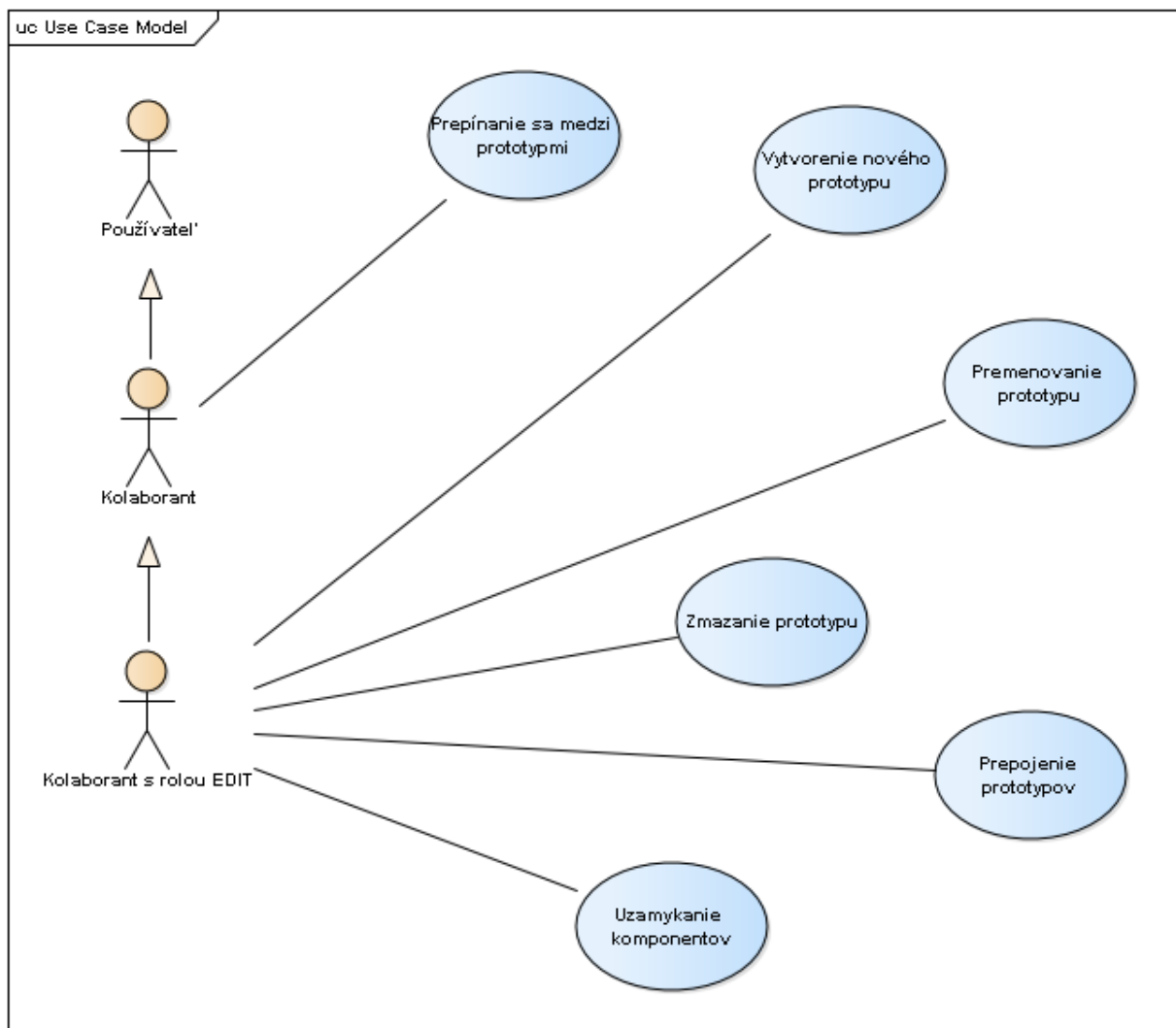
Hlavný tok:

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

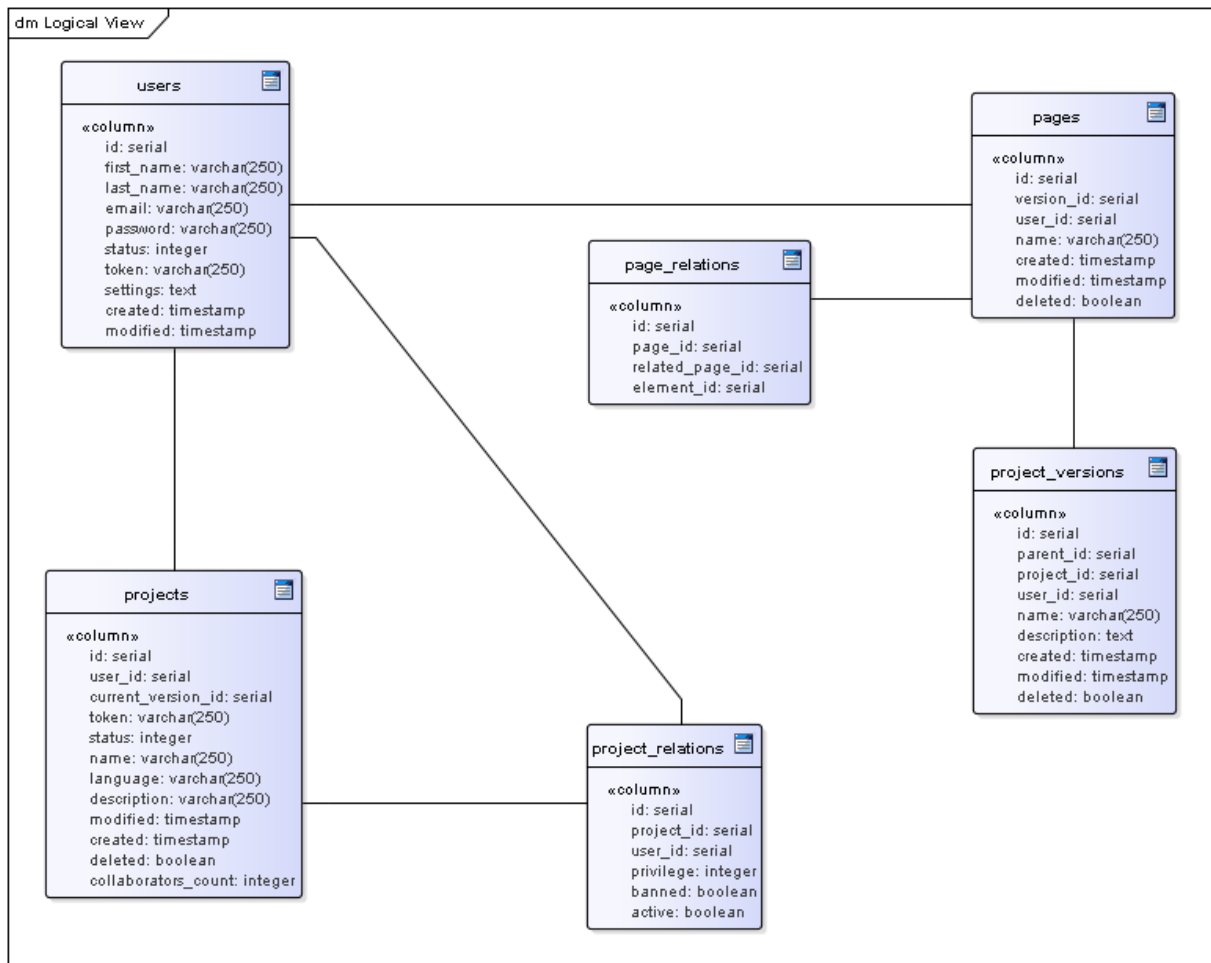
- Možnosť prihlásenia sa do editora a následná autorizácia
- Vyskladanie prototypu prostredníctvom editora
- Vytvorený prototyp je automaticky uložený po zmene
- Možnosť zmeny rozloženia panelov editora pre právo „EDIT“ a zvlášť pre právo „WATCH“
- Používateľ s právom WATCH, nedokáže editovať prototype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

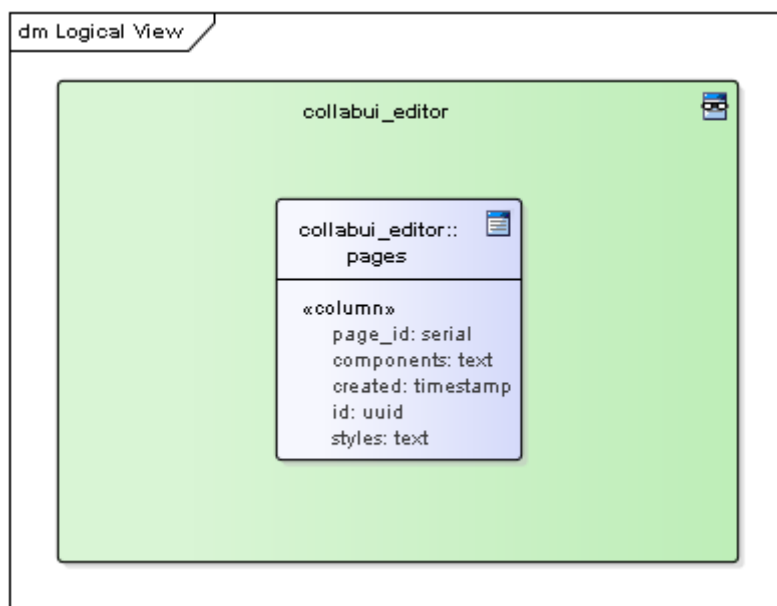
Pri editore je dôležité hlavne validovať skutočnosť, že používateľ s právom WATCH nesmie za žiadnych okolností upraviť prototyp. Validácia prebieha na strane NodeJS Servera.

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.

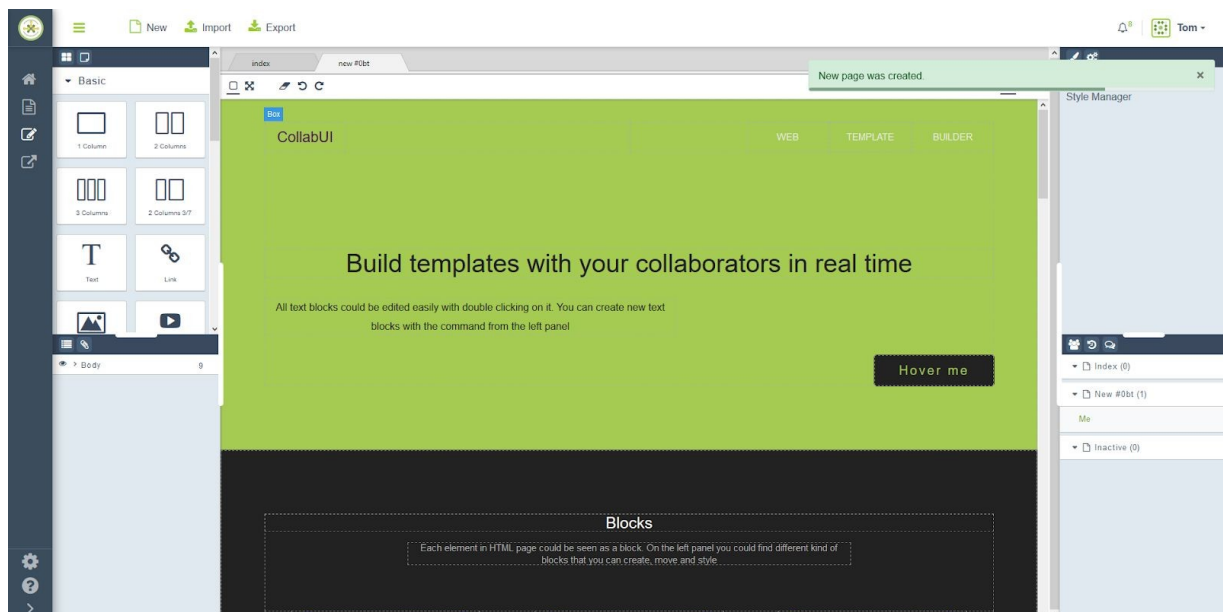
Vytvorenie stránky

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.

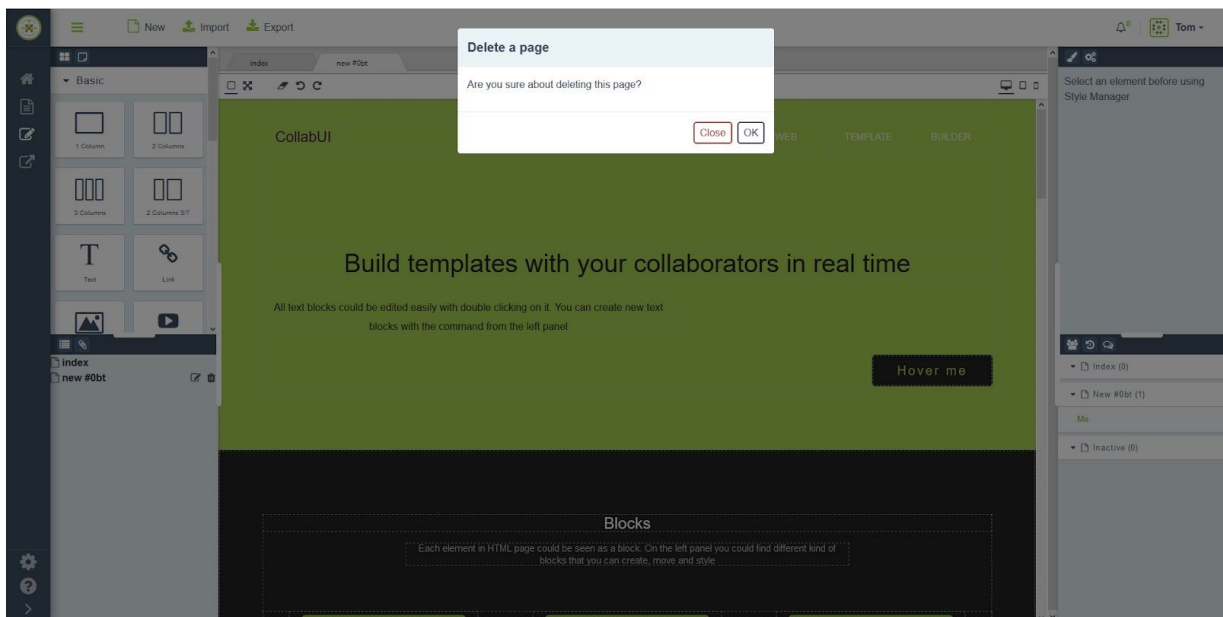


Obrázok 42 - Obrázovka pre vytvorenie novej stránky

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



Obrázok 43 - Obrázovka pre mazanie stránky

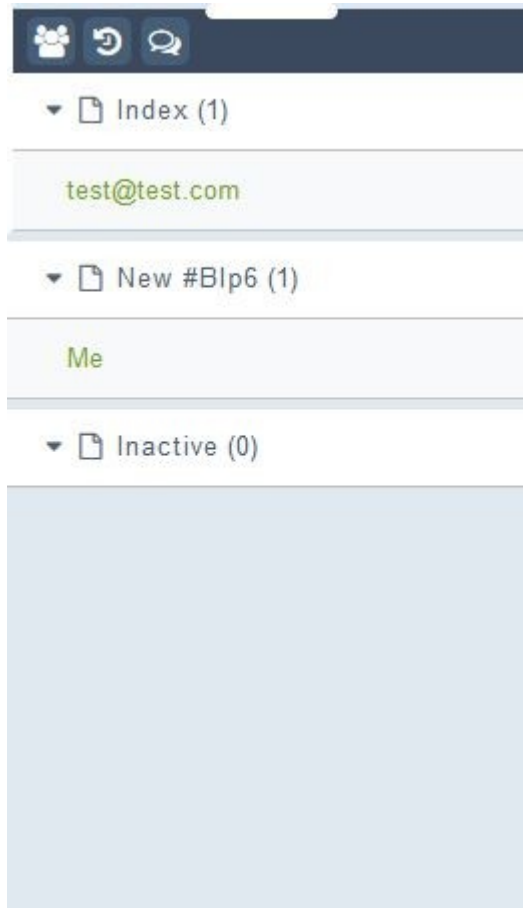
Prepínanie medzi stránkami

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

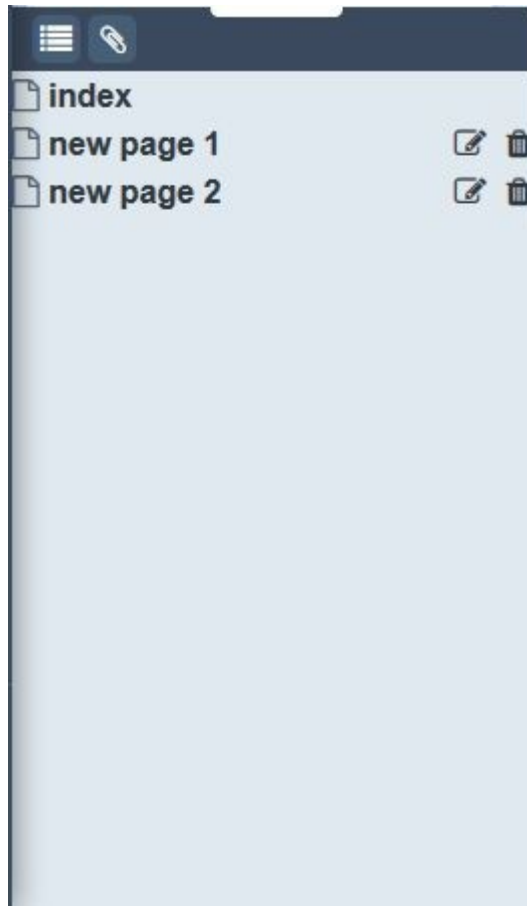
Pri prepínaní medzi stránkami sa každý kolaborant zobrazuje na momentálne aktívnej stránke alebo v inaktívnej zložke v prípade, že nie sú pripojení. V zátvorke sa zobrazuje počet kolaborantov a tieto zložky je možné otvárať alebo zatvárať.



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

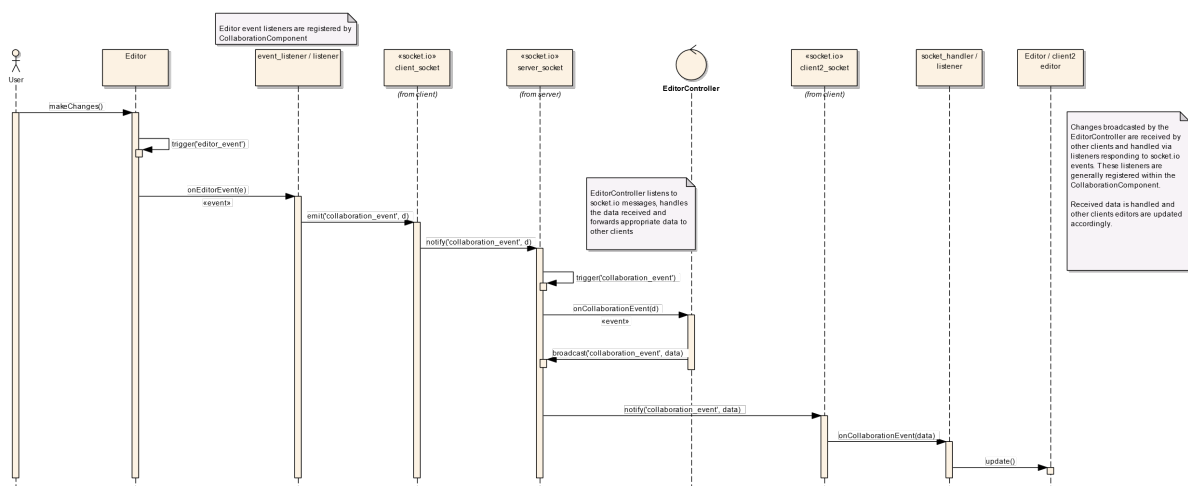
5.5.5.2.4 Synchronizácia zmien v prototypoch

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 konštruktry:

- kolekcia *Components*
- všetky konštruktry 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/odomyknutie 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)
        );
      }
    });
  });
};
```

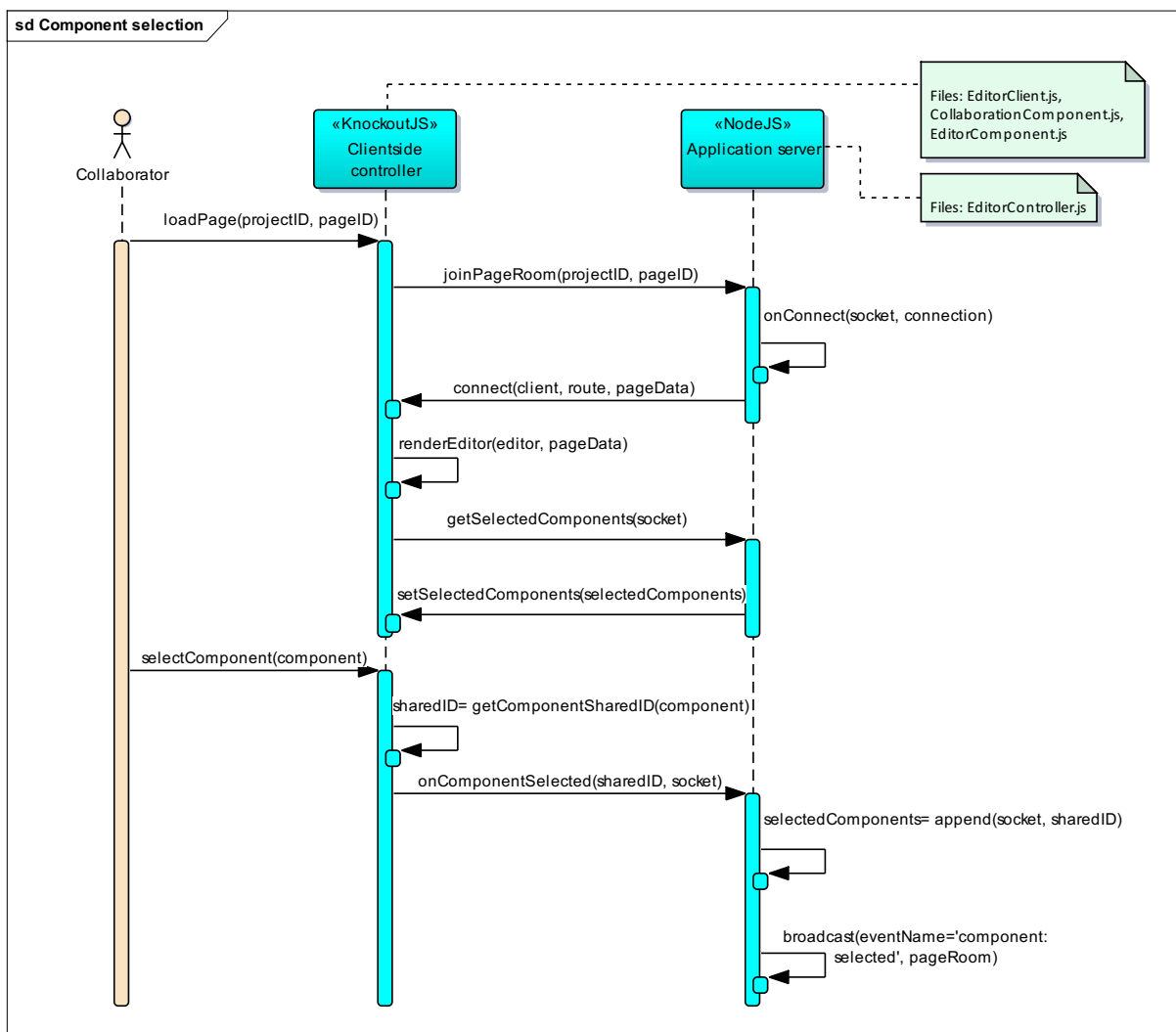
onComponentUpdate, onComponentStyleUpdate, onComponentRemove, onComponentMoved

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

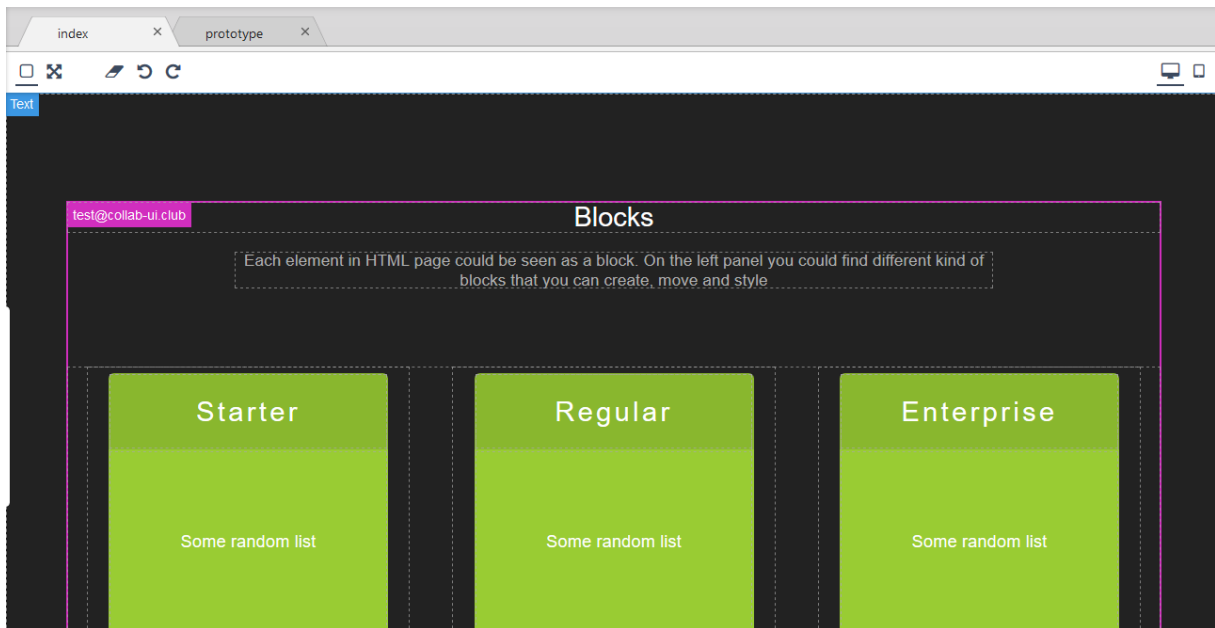
5.5.5.2.5 Vizualizácia označených komponentov

Zvýrazňovanie označených komponentov je pridaná funkcionálna 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 zarovnaniu.

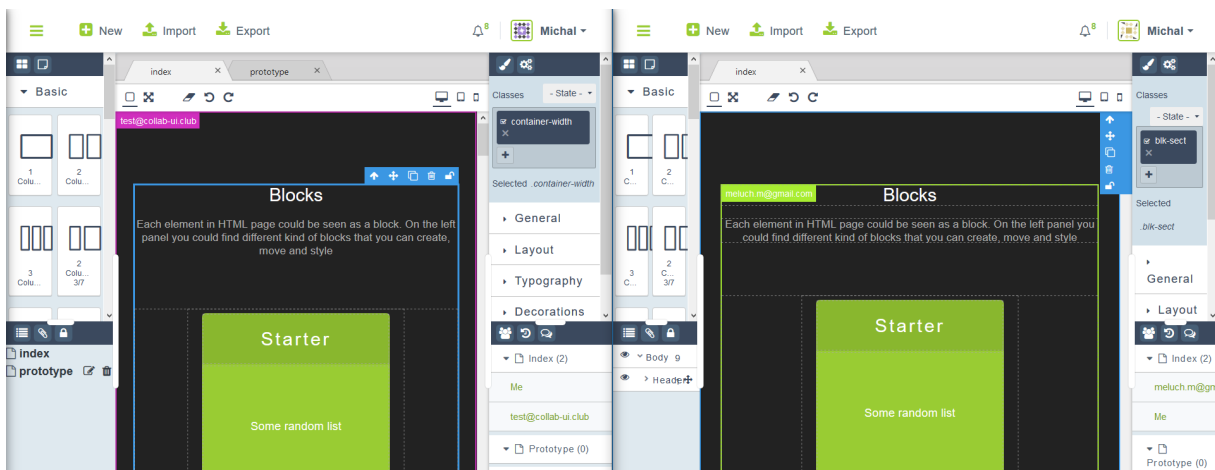
Pri zvýrazňovaní dôjde ku prepisu CSS štýlu daného komponentu (pridanie orámovania) a ku pridaniu HTML elementu obsahujúceho štítok s e-mailovou adresou daného kolaboranta. Nemodifikujeme teda funkcionálnosť knižnice tretej strany, ale pridávame separátne akcie, ktoré sa vykonajú až po skončení všetkých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

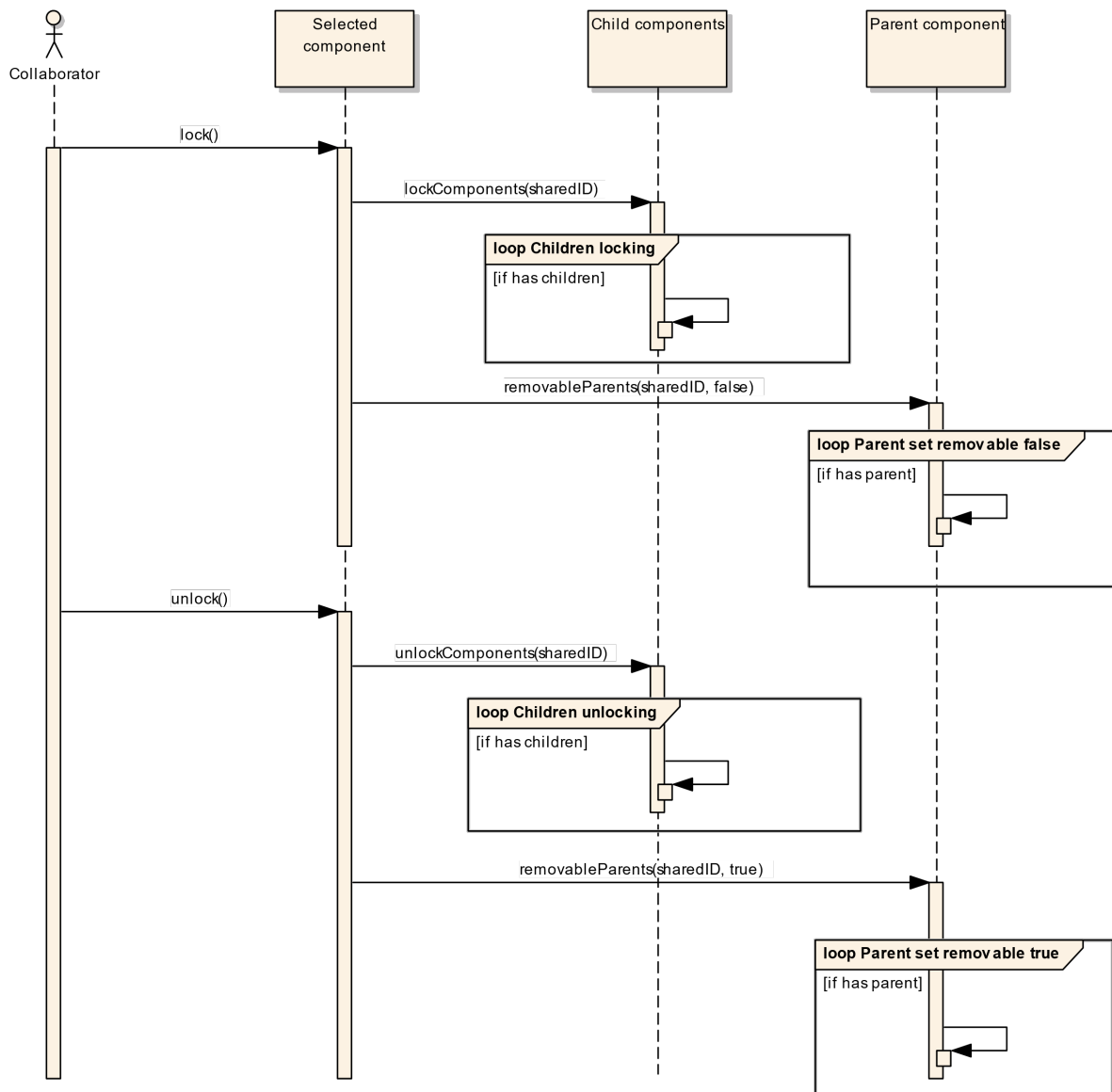
  if (unselectedComponent) {
    // Cast the unselected component HTML to a jQuery object
    var unselectedComponentHtml = unselectedComponent.view.el;
    unselectedComponentHtml = $(unselectedComponentHtml);

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

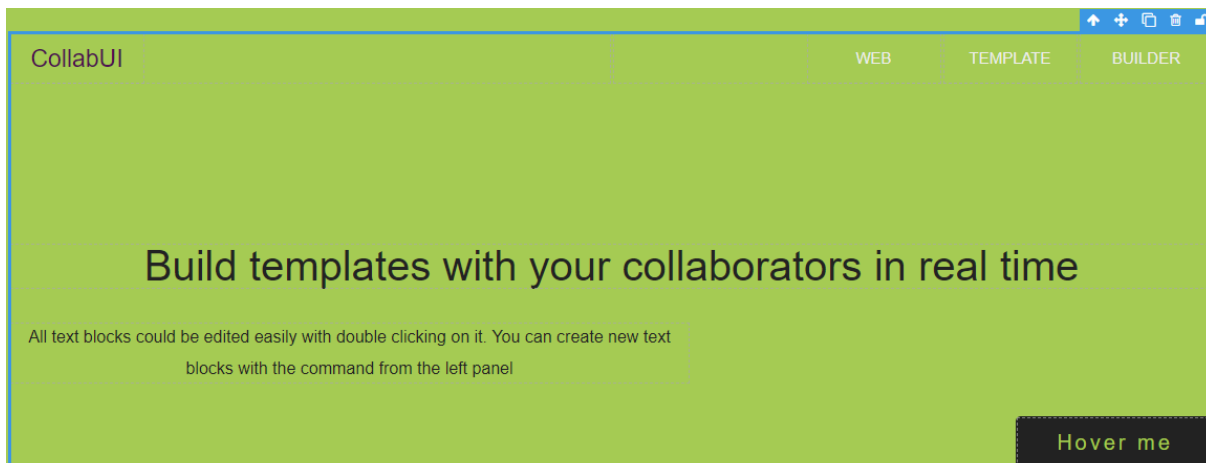
  delete this.selectedComponents[userToken];
};
```

5.5.5.2.6 Uzamykanie komponentov

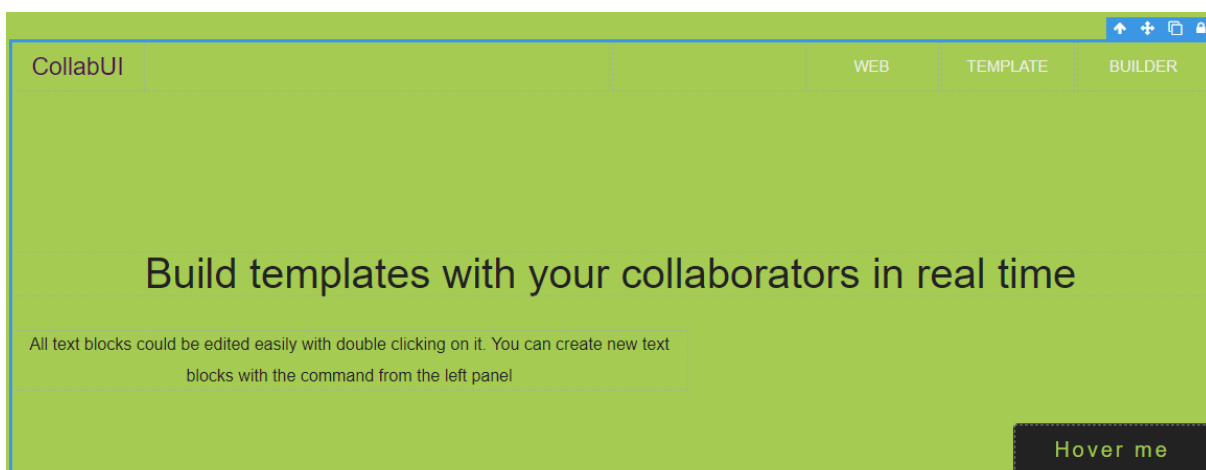
Uzamykanie komponentov je rozširujúca funkcionálna pre GrapesJS. Pomocou uzamykania si môže kolaborant uzamknúť vybraný komponent, ktorý nebudú môcť ostatní kolaboranti upravovať ani premiestňovať. Pri uzamknutí komponentu sa uzamknú aj deti vybraného komponentu. Rovnako sa musí zakázať vymazávanie pre 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. Odomyknúť komponent sa dá pomocou ikony odomyknutého 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 odomyknú keď kolaborant, ktorý komponenty uzamkol opustí konkrétnu stránku projektu resp. opustí socket.



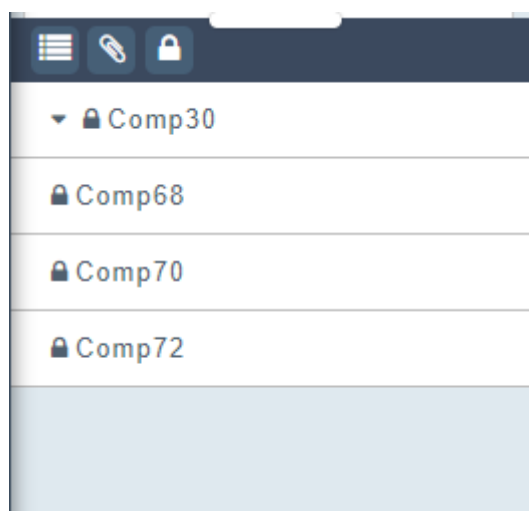
Obrázok 50 - Uzamykanie/odomykanie komponentov



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

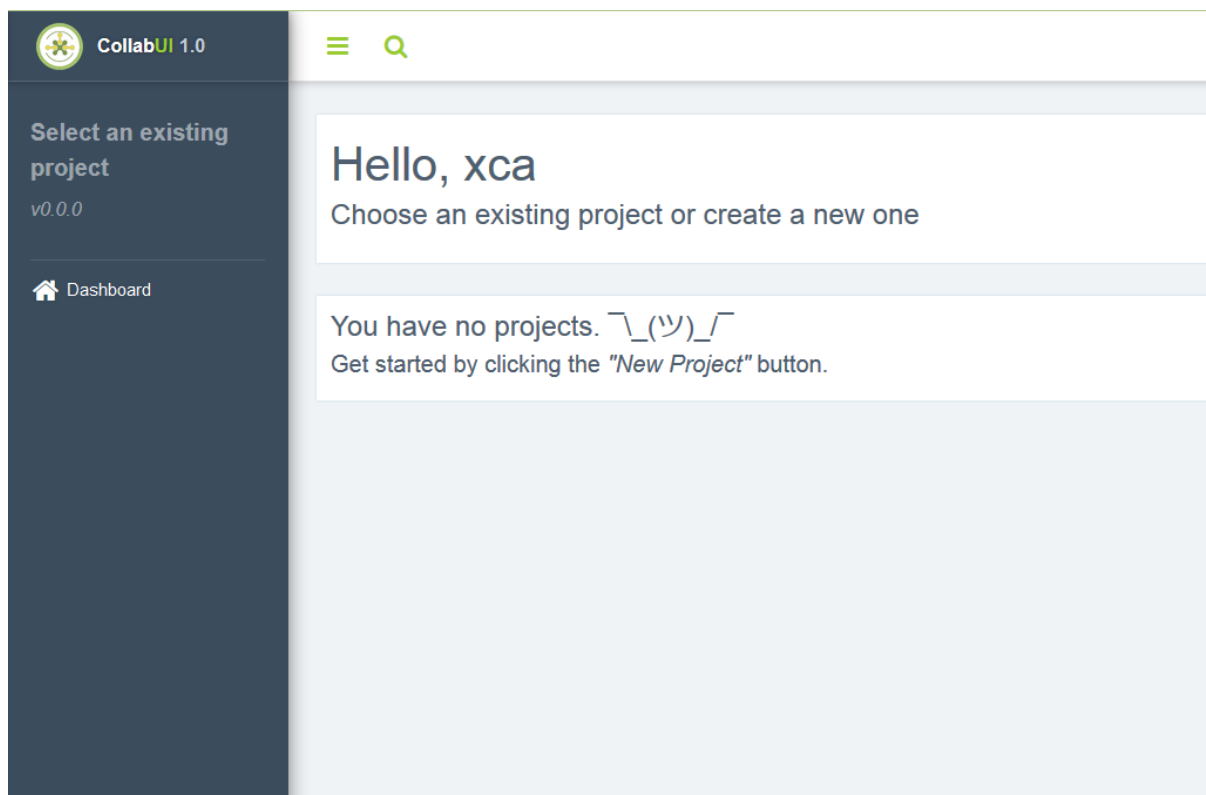
6.1.1 Registrácia a prihlásenie

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

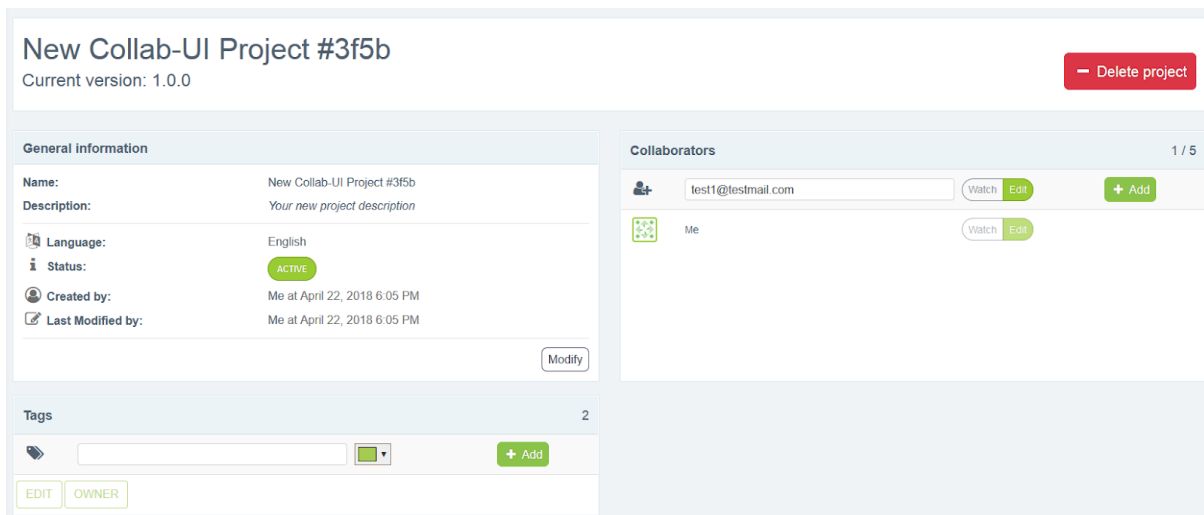
6.1.2 Správa projektov a Dashboard

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

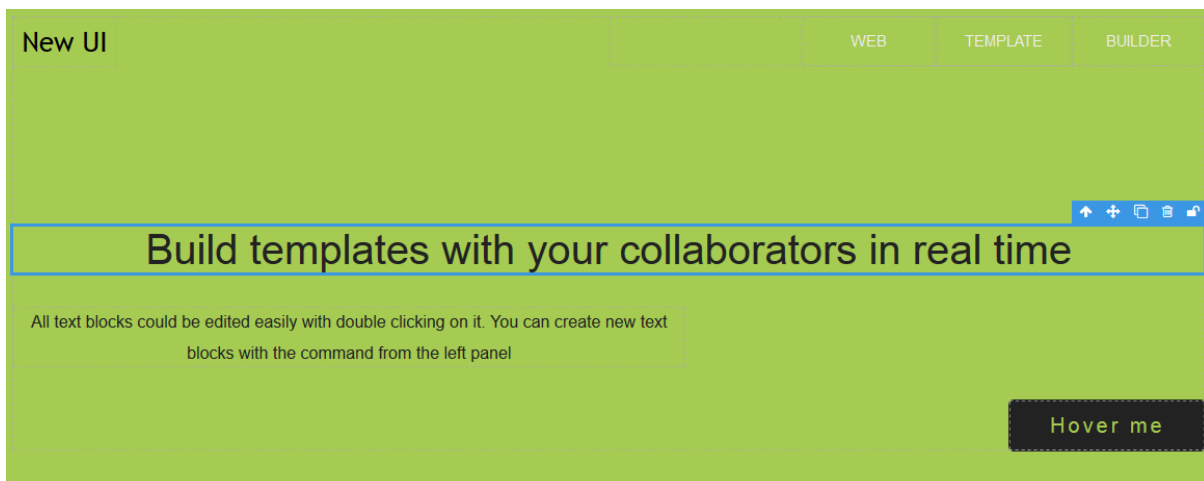
6.1.3 Práca s editorom #1

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

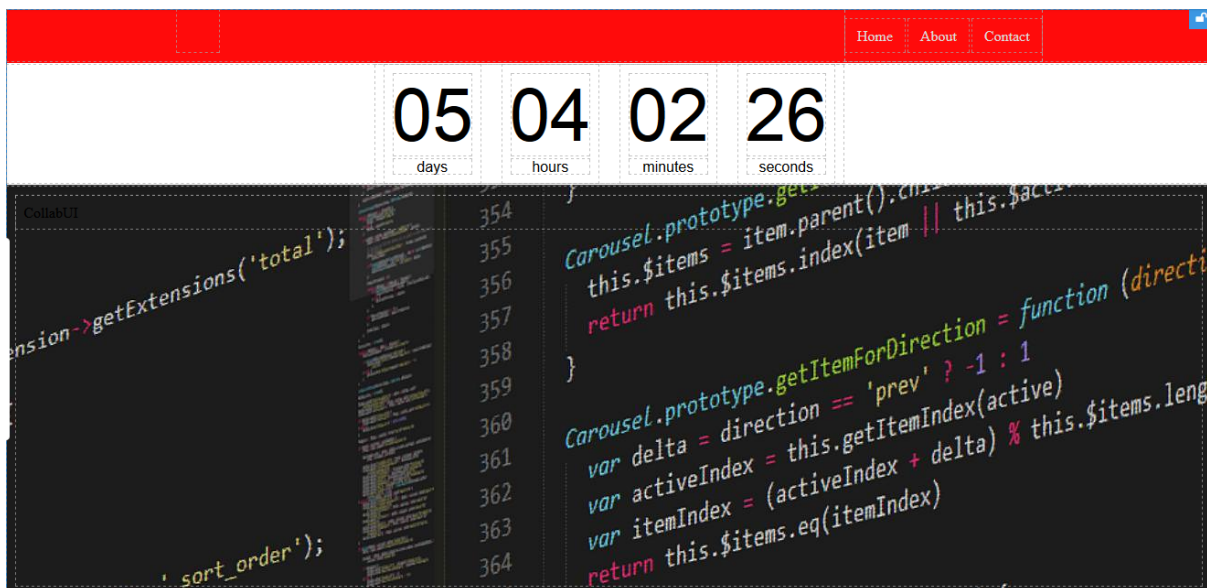
6.1.4 Práca s editorom #2

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

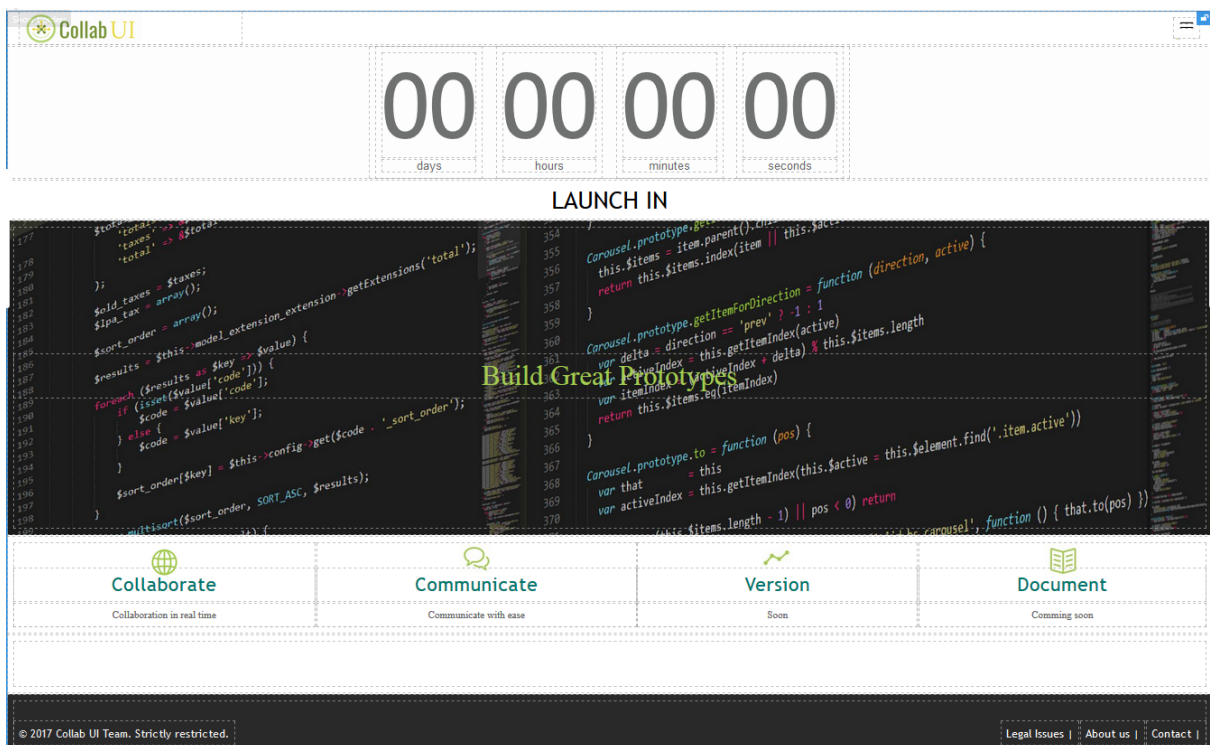
6.1.5 Práca s editorom #3

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

```
Build Great Prototypes
Just the right tool to make building prototypes in team as effortless and convenient as possible
```

```
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
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
```



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 Parádíčka

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

Vaše odpoveď

ODESLAT

Nikdy přes Formuláře Google neposílejte hesla.

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

Google Formuláře

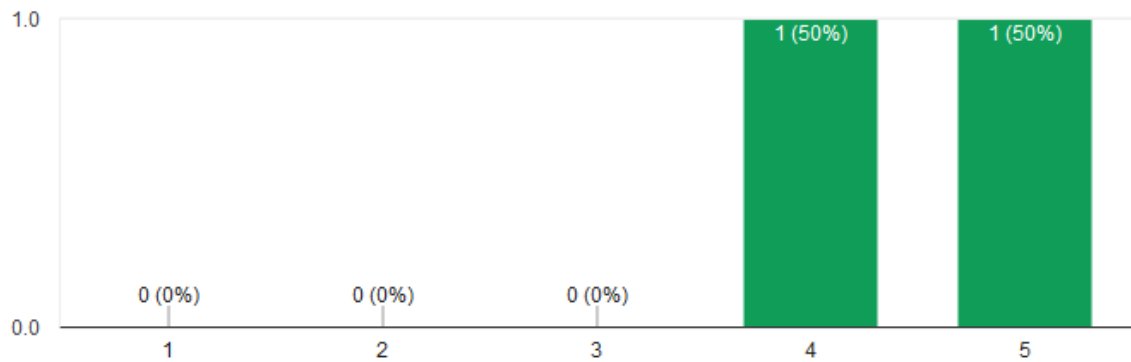
Obrázok 59 - Dotazník po realizácii testovania

6.3 Výstup z testovania

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

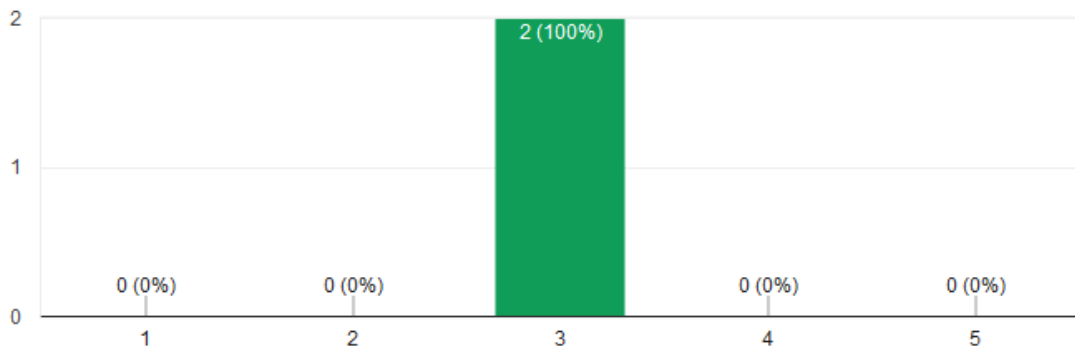
Aká bola odozva akcií v aplikácii?

2 responses



Ako by ste ohodnotili intuitívnosť rozhrania?

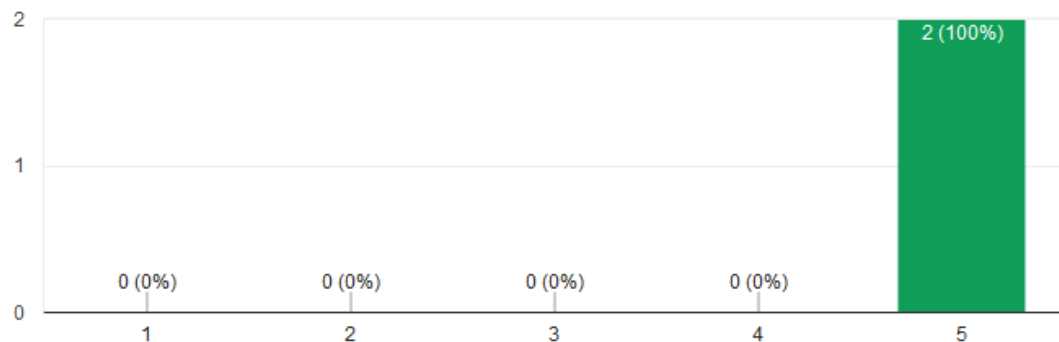
2 responses



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



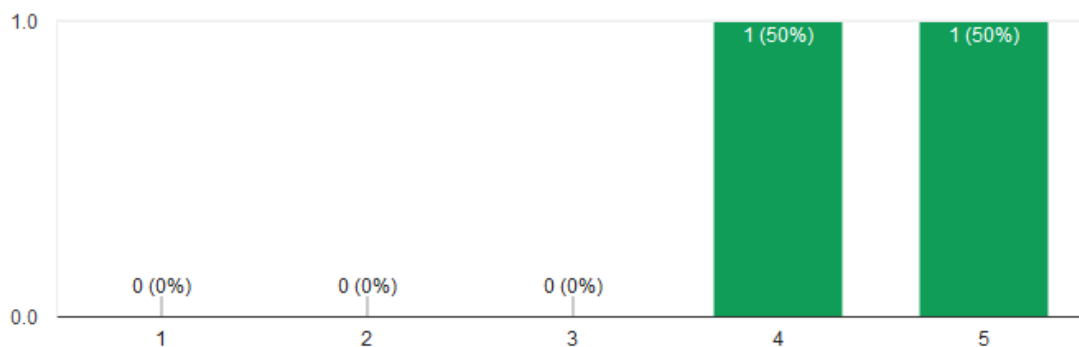
2 responses



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.