Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Webové rozhranie pre distribuované výpočty
BOINC@FIIT
Dokumentácia k inžinierskemu dielu

Vedúci tímu: Ing. Peter Lacko, PhD.
Členovia tímu: Bc. Pavol Čurilla, Bc. Patrik Gallik, Bc. Martin Kaššay,
Bc. Matej Kloska, Bc. Juraj Kochjar, Bc. Roman Roštár
Akademický rok: 2014/2015

História zmien dokumentu

Dátum	Verzia	Zhrnutie zmien	Autor
06.11	1.0	Vytvorenie dokumentu, definovanie štýlov a formátu, vytvorenie štruktúry (kapitol)	Matej Kloska
08.11	1.1	Doplnenie úvodu a cieľov pre zimný semester	Roman Roštár, Patrik Gallik
10.11	2.0	Prvý šprint: Ballantine’s	Celý tím
13.11	3.0	Druhý šprint: Tullamore Dew	Celý tím
17.11	4.0	Tretí šprint: Jameson	Celý tím
18.11	4.1	Kontrola a doplnenie chýbajúcich častí	Celý tím
19.11	4.2	Finalizácia dokumentu	Juraj Kochjar
08.12	4.3	Štvrtý šprint: Four Roses	Martin Kaššay
11.12	4.4	Finalizácia a reorganizácia dokumentu pre tlač	Matej Kloska, Juraj Kochar, Martin Kaššay
10.05	5.0	Štvrtý šprint: Four Roses	Celý tím
11.05	5.1	Šiesty šprint: The Wild Geese	Celý tím
13.05	5.2	Siedmy šprint: Dalmore	Celý tím
15.05	5.3	ôsmy šprint: Glenturret	Celý tím
15.05	5.4	Deviaty šprint: Midleton	Celý tím
18.05	5.5	Doplnenie chýbajúcich častí	Celý tím
20.05	5.6	Finalizácia dokumentu	Celý tím

Obsah

1 Úvod

Tento dokument slúži ako dokumentácia k inžinierskemu dielu vytváraná v rámci predmetu tímový projekt. Úlohou nášho tímu je vytvoriť webové rozhranie pre zadávanie distribuovaných výpočtových úloh do fakultného projektu BOINC@FIIT. Tento projekt využíva platformu BOINC navrhnutú pre manažment vytvárania a distribuovania výpočtovo náročných úloh slúžiacich prevažne v prospech ľubovoľných výskumných oblastí. Hlavným cieľom nášho projektu je vytvoriť všestranne použiteľné a používateľsky prívetivé rozhranie, ktoré umožní jednoduchý prístup k zadávaniu a monitorovaniu spustených aplikácií v rámci fakultného projektu BOINC@FIIT.

Rozšírením povedomia o projekte BOINC@FIIT a vytvorením spomínaného webového rozhrania chceme robiť osvetu o medzinárodne používanej platforme BOINC na našom území. Zapojením čo najväčšieho počtu dobrovoľníkov chceme navýšiť spoločne zdieľaný výpočtový výkon celého projektu a tým ho spraviť ešte zaujímavejším pre výskumníkov, ktorý ho budú používať pri počítaní svojich výpočtovo náročných úloh.

Cieľom, ktorý chceme dosiahnuť v prvom semestri je priblížiť funkčnosť nášho produktu na úroveň webového rozhrania, ktoré je ponúkané s oficiálnou serverovou inštaláciou platformy BOINC. V optimálnom stave by sme chceli rozšíriť dané rozhranie o pokročilé štatistiky a pripraviť backend produktu pre prácu v druhom semestri.

V druhom semestri chceme v systéme ponúknuť možnosť zadania úlohy a jej následnej kontroly priamo pomocou nami vytvoreného webového rozhrania.

1.1 Zoznam skratiek

- REST (*Representational state transfer*) – architektúra rozhraní navrhnutá pre distribuované prostredie, od roku 2000 orientovaná na dáta na rozdiel od SOAP resp. XML-RPC, ktoré sú orientované procedurálne
- SPA (*Single-page application*) – webová aplikácia alebo webové sídlo, ktoré sa skladá z jednej webovej stránky poskytujúcej plynulé jednotné používateľské rozhranie podobné desktopovým aplikáciám
- MVC (*model-view-controller*) – návrhový vzor, ktorý rozdeľuje dátový model na: (Model), používateľské rozhrania (View) a riadiacu logiku (Controller)
- RoR (*Ruby on Rails*) – webový rámec napísaný v programovacom jazyku Ruby
- VCS (*Version control systém*) – systém spravujúci verziovanie softvéru
- CSRF (*Cross-site request foreign*) – forma útoku do internetovej aplikácie
- SQL (*Structured Query Language*) – štruktúrovaný dopytovací jazyk
- URL *(Uniform Resource Locator)* – jednotný ukazovateľ prostriedku zdroja

2 Globálne ciele na zimný semester

Hlavnými cieľmi na zimný semester sú:

- preniknúť do architektúry BOINC:
  - pochopiť fundamentálne princípy architektúry,
  - osvojiť si základy administrácie BOINC servera,
- vytvoriť rozhranie pokrývajúce funkcionalitu ponúkanú v štandardne dostupnom rozhraní, ako napr.:
  - registrácia,
  - prihlásenie,
  - nastavenie účtu,
  - prehľad používateľského účtu,
  - základné štatistiky používateľského účtu,
  - štatistiky:
    - globálne (na úrovni participanta),
    - projektu,
    - aplikácií,
  - načrtnúť rozhranie pre zadávanie výpočtových úloh.

Rámcové ciele z pohľadu šprintov:

1. Šprint: Ballantine's – oboznámenie sa s platformou BOINC
2. Šprint: Tullamore Dew – úvodná stránka BOINC@FIIT, prihlasovanie a registrácia
3. Šprint: Jameson – dátový model a nastavenia účtu
4. Šprint: Four Roses – štatistické rozhranie
5. Šprint: Chivas Regal – úprava vnútornej štruktúry zdrojového kódu a záverečné testovanie funkčnosti produktu

3 Globálne ciele na letný semester

Hlavnými cieľmi na letný semester sú:

- pokrytie procesu zadávania úlohy na výpočet:
  - základné rozhranie,
  - expertné rozhranie,
- prepracovanie podpory pre dostupnosť výsledkov výpočtov,
- rozšírenie štatistických pohľadov na aplikácie,
- podpora autentifikácie pomocou univerzitnej identity,
- automatizácia kontroly vkladaných úloh pomocou antivírusového riešenia.

Rámcové ciele z pohľadu šprintov:

1. Šprint: Johnnie Walker – refaktoring kódu, úprava GIT repozitárov, deployment a automatizácia,
2. Šprint: The Wild Geese – redesign, používateľská zóna,
3. Šprint: Dalmore – vkladanie úlohy, JAVA úlohy,
4. Šprint: Glenturret – REST komunikácia refaktoring vkaldanai úlohy
5. Šprint: Middletonl – finalizácia funkcionalít, dolaďovanie, príprava na odovzdanie.

4 Celkový pohľad po prvom kontrolnom bode

V priebehu prvých troch šprintov sa nám podarilo pochopiť princípy fungovania BOINC architektúry a vytvoriť prototyp rozhrania poskytujúceho nasledujúce funkcionality:

1. Verejne dostupný informačný portál BOINC@FIIT: verejne dostupné webové sídlo informujúce o projekte spolu s odkazmi na oficiálny BOINC projekt.
2. Registrácia: registrácia nového používateľa pomocou vlastných prihlasovacích údajov. Neskôr počítame s automatickou registráciou pomocou AIS.
3. Prihlásenie: prihlásenie do systému pomocou registrovaných údajov
4. Nastavenie účtu: možnosť základného nastavenia svojho účtu spolu so zmenou prihlasovacích údajov
5. Prehľad účtu: základné štatistiky používania

5 Celkový pohľad po druhom kontrolnom bode

V priebehu štvrtého šprintu sa nám podarilo rozšíriť nastavenia používateľského účtu a načrtnúť štatistické rozhranie. Za jednoznačný nedostatok štvrtého šprintu môžeme považovať nedokončenie štatistického rozhrania. Implementované funkcionality a podporné subsystémy:

1. Autodeployment Capistrano skript pre CI službu Codeshop.io[1].
2. Vytvorenie technickej prezentácie reportujúcej aktuálny stav projektu pre kolegov akademickej obce.
3. End-to-end testovanie pre SPA časť aplikácie.
4. Rozšírené nastavenia účtu: možnosť nastaviť parameter pre rôzne profily (lokácie) klientov ako napr. domov, práca, škola a základný.

V porovnaní implementovanej funkcionality z pohľadu prvého a druhého kontrolného bodu sme jednoznačne poľavili na tempe, čo sa negatívne prejavilo na postupe prác. Piaty naplánovaný šprint v momente finalizácie tejto dokumentácie nie je ešte ukončený. Z tohto dôvodu nie je zahrnutý ani do tejto dokumentácie.

[1] Oficiálna stránka projektu: https://codeship.com/

6 Použité technológie

Navrhnuté riešenie používa sadu technológií, ktoré ponúkajú možnosť efektívneho vývoja, vďaka čomu sa môžeme sústrediť na vývoj produktu a nemusíme sa zapodievať procesmi, ktoré priamo nesúvisia s naším produktom.

6.1 Technológie použité v serverovej časti aplikácie:

BOINC
Oficiálny web: https://boinc.berkeley.edu/
Verzia: 7.2.42
Popis: BOINC: Berkley Open Infrastructure for Network Computing - opensource softvér pre dobrovoľnícke počítanie a grid computing.
Dôvod výberu: jadro BOINC infraštruktúry, ktoré nie je možné vymeniť a je potrebné pre chod celého systému.

MySQL
Oficiálny web: https://www.mysql.com/
Verzia: 5.5.40
Popis: relačná databáza, ktorá sa v prípade nášho projektu používa pre ukladanie dát o používateľoch a výpočtoch.
Dôvod výberu: MySQL bol zvolený hlavným vývojovým tímom BOINC a nie je možné daný databázový server vymeniť za iný aj keby na základe preferencií náš tím chcel zvoliť iný.

Ruby
Oficiálny web: https://www.ruby-lang.org/en/
Verzia: 2.1.5
Popis: vysokoúrovňový objektovo orientovaný interpretovaný programovací jazyk
Dôvod výberu: niektorí členovia tímu majú s daným programovacím jazykov skúsenosti a ostatní členovia sa chcú naučiť novým technológiám. Taktiež je dôležité spomenúť silnú komunitu, ktorá dokáže rýchlo pomôcť v prípade problému.

Ruby On Rails
Oficiálny web: http://rubyonrails.org/
Verzia: 4.1.5
Popis: opensource webový rámec pre programovací jazyk Ruby
Dôvod výberu: webový rámec, ktorý ponúka všetko potrebné pre jednoduchý vývoj webových aplikácií na strane servera.

6.2 Technológie použité v klientskej časti aplikácie:

JavaScript
Oficiálny web: http://www.w3schools.com/js/
Verzia: ECMA-262, revízia 5 (V8 engine)

Popis: interpretovaný webový programovací jazyk, ktorý sa primárne používa vo webových prehliadačoch, no svoje miesto má i na strane servera.

Dôvod výberu: potreba vykonávania aplikačnej logiky na strane webového prehliadača, platformová nezávislosť kódu

AngularJS
Oficiálny web: https://angularjs.org/
Verzia: 1.3.0
Popis: webový rámec naprogramovaný v JavaScripte pre SPA aplikácie
Dôvod výberu: jednoduchosť vývoja SPA aplikácii. AngularJS má taktiež silnú komunitu, ktorá je výhodou.

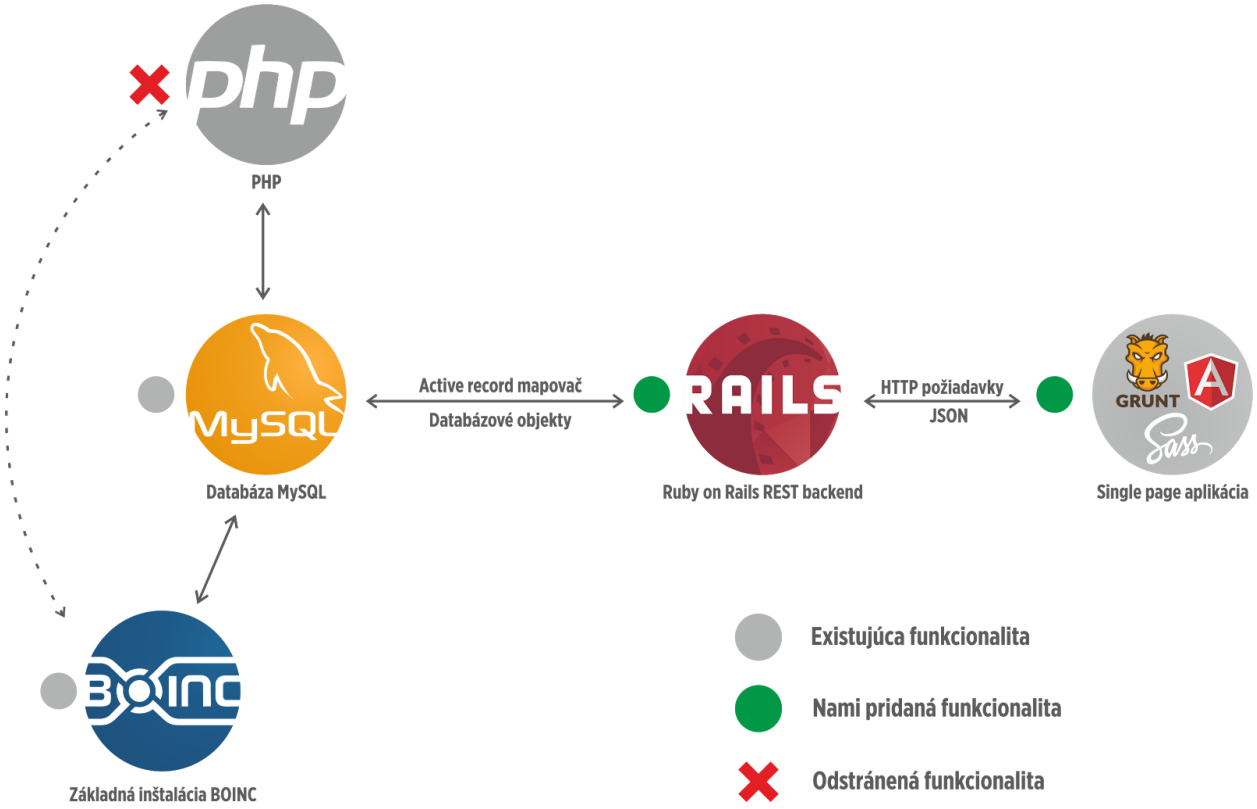
CSS3
Oficiálny web: http://www.w3schools.com/css/css3_intro.asp
Verzia: 3.0
Popis: štandard pre zápis kaskádových štýlov pre webové stránky.

SASS
Oficiálny web: http://sass-lang.com/
Verzia: 3.2.19
Popis: rozšírenie štandardného CSS zápisu štýlov. SASS štýly sa v procese prekladu pomocou SASS kompilátora stávajú štandardnými CSS štýlmi.
Dôvod výberu: potreba štruktúrovaného písania štýlov pre webovú aplikáciu

Grunt
Oficiálny web: http://gruntjs.com/
Verzia: 0.1.13
Popis: nástroj pre automatizáciu opakujúcich sa procesov v priebehu vývoja softvéru, ktorý je naprogramovaný v JavaScripte a spolupracuje s NodeJS jadrom.
Dôvod výberu: potreba automatizácie procesov vývoja SPA časti aplikácie. GruntJS má silnú podporu pre AngularJS.

7 Architektúra

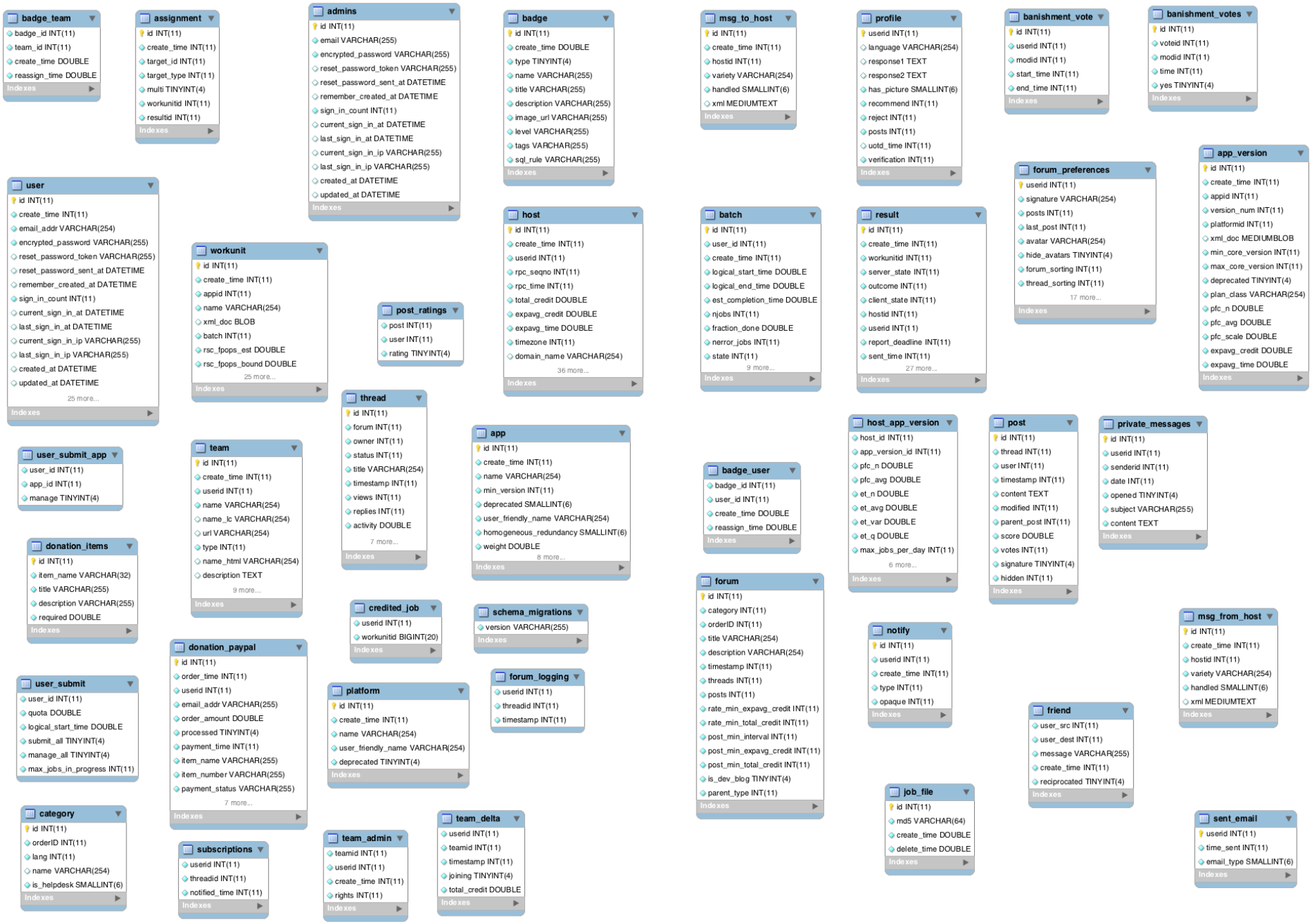
Architektonický návrh aplikácie bol ovplyvnený existujúcim riešením rozhrania, ktoré je štandardne dostupné. Pre obmedzenie šírenia komplikácií pri prechode na novšiu verziu BOINC Servera sme sa rozhodli rozdeliť aplikáciu na REST a SPA moduly, vďaka čomu SPA modul je nezávislý od implementácie BOINC servera. Samotný REST modul ďalej uplatňuje štandardný návrhový vzor MVC, vďaka čomu dokážeme ešte ďalej izolovať potenciálne problémy na úrovni dátového modelu pomocou modelov.



Obr. 1.: Návrh architektúry riešenia

7.1 Dátový model

Existujúca implementácia BOINC servera poskytuje dátový model, ktorý obsahuje entity pokrývajúce celý systém. Problém tohto modelu spočíva v chýbajúcej dokumentácii, vďaka čomu je obtiažne v tejto fáze zostrojiť kompletný dátový model produktu. Model bude v priebehu štvrtého a piateho šprintu inkrementálne pribúdať, čo v konečnom dôsledku poskytne korektný dátový model pri odovzdaní druhého kontrolného bodu.



Obr. 2.: Pôvodný dátový model obohatený o nami pridané vybrané hodnoty

Dátový model, ktorý je na obrázku *Obr. 2* je generovaný nástrojom *MySQL Workbench*, nakoľko sme štruktúru databázy nenavrhovali my, ale bola daná platformou BOINC. Ako je možné vidieť na obrázku, medzi tabuľkami neexistujú žiadne prepojenia. Databáza obsahuje iba niektoré druhy kľúčov, a to primárne kľúče a unikátne kľúče, čo však nespĺňa kvalitatívne požiadavky, pretože dáta nie sú chránené proti zmazaniu alebo prepísaniu. V databáze chýba zabezpečenie integrity dát pomocou cudzích kľúčov. Nakoľko nevieme, ako interne funguje platforma BOINC, to znamená, aké sa vykonávajú dátové dopyty, nemôžeme si dovoliť tieto cudzie kľúče definovať. Ďalšou veľkou nevýhodou je, že k databáze platformy BOINC neexistuje žiadna oficiálna dokumentácia, teda sa môžeme len domnievať, kde by sa mohli nejaké kľúče nachádzať.

8 Šprint 1 - Ballantine's

8.1 BOINC Inštalácia

8.1.1 Úloha

Nainštalovať BOINC server z oficiálnych inštalačných súborov pre možnosť oboznámenia sa so základnými princípmi fungovania platformy BOINC. Osvojiť sú základné PHP webové rozhranie nainštalovaného systému spolu s jeho architektúrou a funkcionalitou.

8.1.2 Implementácia

Inštalácia BOINC servera sa v našom prípade odvíjala od operačného systému, ktorý je v našom prípade Ubuntu Server 14.04. Oficiálna dokumentácia inštalácie[2] predpokladá použitie UNIX operačného systému pre serverovú časť. Vďaka výberu Ubuntu Server prebehla inštalácia relatívne jednoducho. Celý postup môžeme zhrnúť do nasledujúcich krokov:

- inštalácia MySQL5.5 databázového servera spolu s knižnicami potrebnými pre ďalší vývoj v MySQL,
- inštalácia potrebných softvérových knižníc súvisiacich s jadrom BOINC servera a webového rozhrania:
    - libtool-2.2.10-3.fc14.i686 (for libtoolize),
    - gcc-c++-4.5.1-4.fc14.i686 (for g++),
    - libstdc++-static-4.5.1-4.fc14.i686 (for libstdc++.a, needed by sample apps),
    - MySQL-python-1.2.3-0.5.c1.fc14.i686 (for MySQLdb),
    - php-mysql,
    - php-gd,
- naklonovanie oficiálneho repozitára BOINC servera,
- spustenie štandardného procesu:
    - configure,
    - make,
    - make check,
    - make install,
- doinštalovanie Python modulov potrebných pre vygenerovanie prázdneho projektu,
- nastavenie MySQL databázy spolu so správnymi právami používateľa,

[2] Postup inštalácie BOINC https://boinc.berkeley.edu/trac/wiki/ServerIntro#general

- nastavenie CRON záznamov pre daný projekt,
- nakonfigurovanie Apache Servera pre daný projekt,
- nastavenie správnych práv pre adresár projektu z pohľadu Apache,
- vytvorenie používateľa spolu s heslom pre administračné rozhranie projektu.

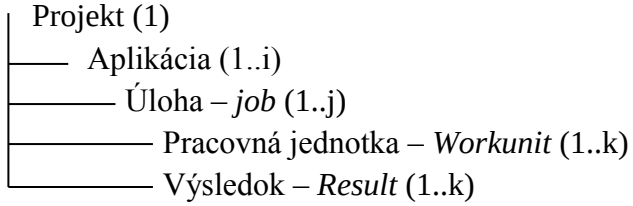
8.2 BOINC Analýza

8.2.1 Úloha

Podrobná analýza fungovania platformy pre distribuované výpočty BOINC a pochopenie jej technického konceptu, je nevyhnutné pre korektné vytvorenie webového rozhrania na vkladanie a správu výpočtových úloh.

8.2.2 Implementácia

Základnú serverovú inštaláciu BOINC, spustenú lokálne na Ubuntu Server 14.04, sme podrobne analyzovali s cieľom získať nevyhnutné informácie o spôsobe fungovania celej platformy, pričom sme sa zamerali hlavne na agendu okolo vytvárania, zadávania a sledovania úloh a ich následnej distribúcie medzi koncových participantov projektu. Nutnou predpokladom pre participovanie na akomkoľvek distribuovanom výpočtovom projekte vytvorenom v rámci fakultnej inštalácie BOINC@FIIT je mať nainštalovanú a spustenú klientsku aplikáciu platformy BOINC, voľne dostupnú na oficiálnych stránkach BOINC[3]. Logická architektúra distribuovanej výpočtovej platformy BOINC je nasledovná:



Projekt je jasne identifikovaný svojim jedinečným identifikátorom *(master URL)*, ktorý reprezentuje adresu domovskej stránky projektu. V rámci projektu je možné vytvárať a spravovať niekoľko paralelne bežiacich (logicky nezávislých) aplikácii. Každá z jednotlivých aplikácii sa skladá z ľubovoľného počtu úloh. Tie pozostávajú z pracovných jednotiek, ktoré sú ako malé a logicky nezávislé výpočtové celky odosielané koncovým používateľom na vypočítanie. Ukončením výpočtu na danej pracovnej jednotke vzniká jej výsledok. Spojením výsledkov všetkých pracovných jednotiek v rámci úlohy a v rámci všetkých aplikácii vzniká výsledok projektu.

[3] http://boinc.berkeley.edu/download.php

8.3 Základná kostra SPA aplikácie

8.3.1 Úloha

Vytvoriť prostredie pre vývoj klientskej časti SPA aplikácie. Zadefinovať adresárovú štruktúru, nástroje na podporu vývoja.

8.3.2 Návrh

Rozhodli sme sa, že aplikácia bude postavená na rámci *AngularJS*[4]. Tento rámec je momentálne pre tento účel najpoužívanejší a má veľkú komunitu, preto nebude problém s podporou a ani s prípadným ďalším vývojom iným tímom. Pre rýchle prototypovanie nových obrazoviek použijeme rámec *Bootstrap*[5]. Na automatizáciu úloh použijeme jeden z nástrojov Grunt/Gulp.

8.3.3 Implementácia

Na inicializáciu projektu sme použili generátor nástroja *Yeoman*[6], konkrétne generátor *generator-angular*[7]. Použitím tohto generátora sme podstatne urýchlili štart projektu. Na automatizáciu úloh sme použili nástroj Grunt. Medzi úlohy, ktoré sme zautomatizovali, patrí: spájanie a zmenšovanie JavaScript a CSS súborov, kontrola JavaScript súborov pomocou nástroja *JsHint*[8], spúšťanie vývojového servera, automatické znovu načítanie prehliadača pri zmene zdrojových súborov. Pomocou nástroja Grunt sme nakonfigurovali spúšťanie unit testov nad JavaScript súbormi. Dohodli sme sa použití testovacieho rámca *Jasmine* (s tým, že je to možné v budúcnosti prispôsobiť). Na správu knižníc tretích strán sme použili nástroj *Bower*[9].

Množstvo úloh a dohodnutých postupov je pomerne veľké, preto sme vypracovali samostatnú množinu metodík ako používať prostredie.

8.4 Základná kostra REST aplikácie

8.4.1 Úloha

Vytvoriť prostredie pre vývoj klientskej časti REST aplikácie. Zadefinovať adresárovú štruktúru, nástroje na podporu vývoja.

[4] AngularJS: https://angularjs.org/
[5] Bootstrap: http://getbootstrap.com/
[6] Yeoman: http://yeoman.io/
[7] Angular Generator: https://github.com/yeoman/generator-angular
[8] JS Hint: http://www.jshint.com/
[9] Bower: http://bower.io/

8.4.2 Návrh

Aplikácia bude postavená na rámci *RoR*[10], ktorý vedie k produktívnemu a rýchlemu vývoju webových aplikácií. Výhodou je relatívne veľká komunita, čo značne prispieva ku kvalitnej dokumentácii, podpore a rýchlemu riešeniu prípadných problémov tak, ako v prípade rámca AngularJS. RoR má tiež bohatý ekosystém knižníc a rozšírení (*gemov*), ktoré uľahčujú vývoj a pomáhajú zamerať sa na jadro aplikačnej logiky.

8.4.3 Implementácia a testovanie

Vytvorili sme kostru projektu postaveného na rámci RoR pomocou generátora, ktorý tento rámec poskytuje.

```
rails new backend
```

Pridali sme niekoľko užitočných gemov, spomeňme najvýznamnejšie: gemy *RSpec* a *Capybara* na písanie testov, gem *simplecov*[11], ktorý generuje štatistiky pokrytí kódu testami a gem *better_errors* pre prehľadnejšie zobrazovanie chybových upozornení. Vytvorenú kostru projektu sme umiestnili do príslušného GIT repozitára v službe Bitbucket.org a podľa príslušnej metodiky sme vytvorili hlavnú a vývojovú vetvu v rámci repozitára.

8.5 Zhodnotenie šprintu

Prvý šprint bol z pohľadu plánovania náročný a v konečnom dôsledku testovací. Tím si na ňom odskúšal základné princípy fungovania spoločnej práce. Hlavným cieľom bolo oboznámenie sa s platformou BOINC na globálnej úrovni. Tento cieľ sa nám podarilo dosiahnuť. Sekundárne ciele, ktoré sa nám podarilo dosiahnuť boli závislé od primárneho. Menovite sa jedná o navrhnutie architektúry riešenia spolu so základnými rámcami riešení konkrétnych častí architektúry: SPA a REST.

8.5.1 Retrospektíva šprintu

Okruh záujmu	**Negatívum**	**Pozitívum**
Analýza a návrh	nedostatočné načasovanie postupov stále veľmi nejasné a abstraktné, čo vlastne budeme robiť	grafické návrhy rozhraní aplikácie

[10] Ruby On Rails: http://rubyonrails.org/
[11] Oficiálna stránka ruby gemu: https://rubygems.org/gems/simplecov

	nejasné, ako automatizovať procesy nedostatočná analýza	
Konfigurácia	N/A	inštalácia a konfigurácia nutných prostredí a samotnej BOINC platformy relatívne rýchle vyriešenie podporných systémov, virtuálnych operačných systémov a nutných prerekvizít
Plánovanie	časový sklz	N/A
Tím a stretnutia	iba 1 Teambuilding veľmi slabá komunikácia pri slabej navyše často neefektívna komunikácia nedostatočná kvalita spoločných stretnutí zlé vyhodnotenie agilných metód vývoja – hlavne priebeh a vedenie scrum metodiky	morálka tímu a tímové zloženie
Programovanie	očakávaná vyššia produktivita a výstupy tímu veľmi veľa administratívy brzdí pokrok na projekte málo času venovaného programovaniu a práci na projekte	spoločné programovanie

Na stretnutí sme sa zhodli, že prvý šprint vôbec nedopadol podľa našich predstáv. Máme problémy s koncentráciou na stretnutiach a konštruktívnou diskusiou. Konfigurácia prostredí prebehla v poriadku, avšak čo sa týka programovania v prvom šprinte, zaostávalo za očakávaniami.

9 Šprint 2: Tullamore Dew

9.1 Údržba existujúceho systému BOINC@FIIT

9.1.1 Úloha

Vzhľadom na fakt, že v doterajšej verzii projektu BOINC@FIIT nebola registrácia nového používateľa patrične zabezpečená proti automatickým hromadným registráciám, denne do systému pribúdalo niekoľko tisíc nových, falošných užívateľov, čo neprimerane vyťažovalo prostriedky infraštruktúry. Navrhnite a implementujte mechanizmus na automatizovanú kontrolu a mazanie hromadne vytvorených používateľov. Dôležité je pamätať na ochranu existujúceho spôsobu registrácie vhodnou formou kontroly osoby registrujúcej sa.

9.1.2 Návrh

Pre zabránenie ďalšieho hromadného registrovania, sme navrhli rozšírenie registračného formulára o ochranný prvok nazývaný CAPTCHA. Pre odstránenie už registrovaných používateľov sme stanovili pravidlo podľa ktorého sa vymažú všetci používatelia, ktorý neposkytli svoj procesorový čas a sú registrovaný viac ako mesiac.

9.1.3 Implementácia

V našom projekte používame implementáciu CAPTCHA testu voľne, zdarma dostupnú službu reCAPTCHA[12] od spoločnosti Google. reCAPTCHA, okrem samotného CAPTCHA testu, používateľovi umožňuje vypočuť si zvukovú verziu vygenerovaného textu a v prípade jeho nečitateľnosti možnosť nechať si vygenerovať text nanovo.

Na našej stránke bol do registračného formulára zapracovaný komponent odkazujúci na backend služby reCAPTCHA, ktorý poskytovaný spoločnosťou Google. Tento komponent je implementovaný v súboroch *create_account_form.php* a *create_account_action.php* v adresári */html/user*.

Nakoniec bol vytvorený skript v jazyku PHP, ktorý vyberie z tabuľky *user* používateľov, ktorý boli zaregistrovaní pred viac ako mesiacom ale ich *total_credit* je stále nulový. Keďže v databáze nie sú väzby, nedá sa použiť kaskádové mazanie pomocou SQL príkazu. Z tohto dôvodu bolo nutné ručne napísať jednotlivé príkazy, ktoré boli referencované tabuľkou *user*. Po mesiaci spúšťania tohto skriptu každú noc by sa mali v databáze nachádzať len používatelia, ktorí sú reálne aktívni.

[12] https://www.google.com/recaptcha/intro/index.html

9.2 Poskytnutie informácií o projekte BOINC@FIIT

9.2.1 Úloha

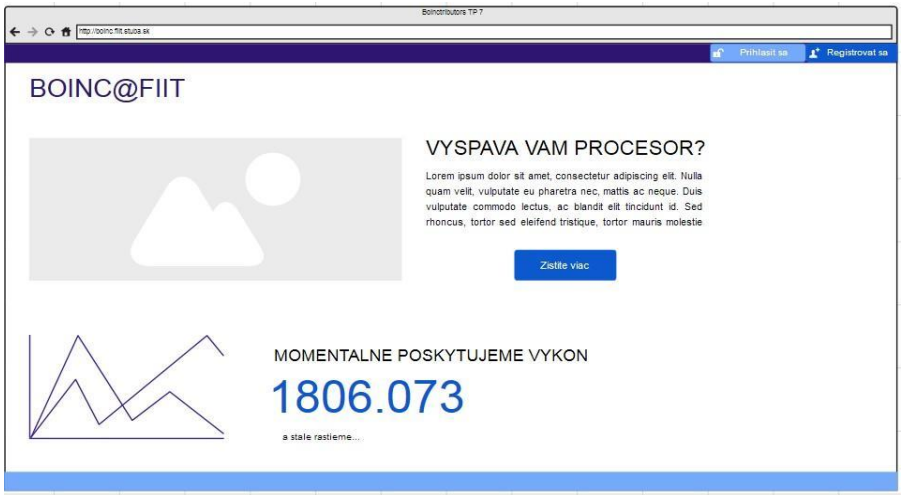
V zmysle budovania povedomia a šírenia informácií o projekte BOINC, spravovanom a realizovanom na fakulte, považujeme za kľúčové navrhnúť jednotné, intuitívne a hlavne jednoducho použiteľné webové rozhranie podávajúce potrebné informácie konzistentne a priamočiaro. Tieto vlastnosti považujeme za fundamentálne vzhľadom na oslovenie širokého spektra fakultných aj mimo fakultných používateľov.

9.2.2 Návrh

Nízkoúrovňový návrh bol vytvorený pomocou nástroja *Moqups*[13]. Umiestnením tlačidiel slúžiacich na prihlásenie, resp. registráciu nového používateľa do projektu, spĺňa a dodržiava základné konvencie súčasného webu. V záujme zjednotenia poskytovaného obsahu je prezentovaný text jednoduchý, dobre čitateľný a doplnený o ilustračné obrázky. Dvojfarebná vizuálna schéma je jasne rozpoznateľná a je konzistentná v rámci celého webového sídla. Je odvodená od farebnej schémy BOINC platformy, no je zasadená do modernejších, navzájom kontrastnejších a príjemnejších odtieňov.

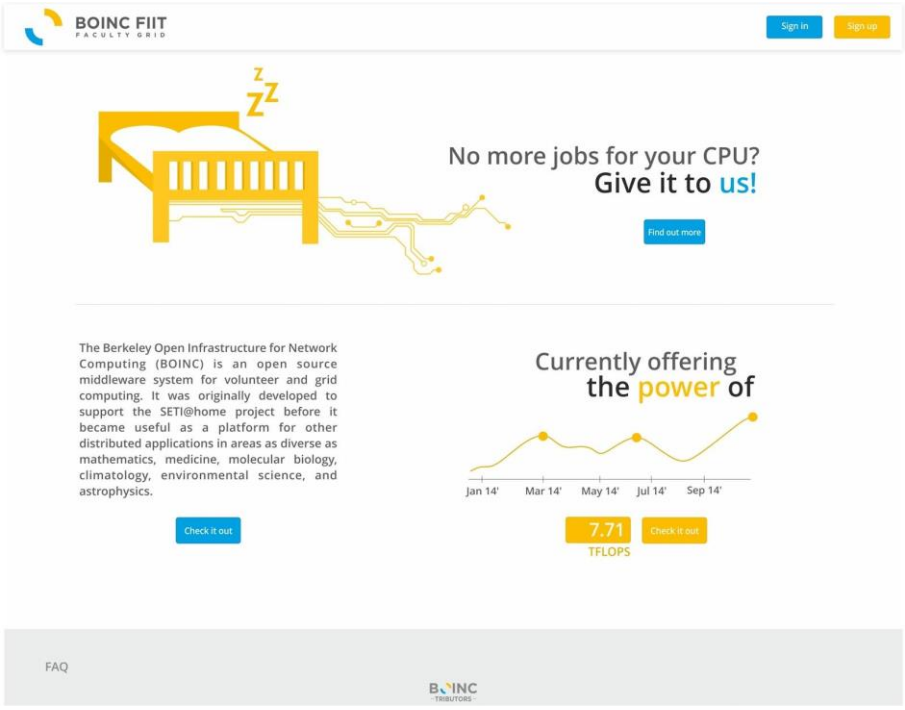
9.2.3 Implementácia

Vzhľadom na základné princípy vytvárania rozhraní sme postupovali od jednoduchých konceptov a náčrtov, cez návrhy s nižšou grafickou náročnosťou až po výsledné plne farebné a verné návrhy výsledného rozhrania. Pre zrozumiteľnejšie zobrazenie budú návrhy prezentované spolu s ich nízkoúrovňovými návrhmi.

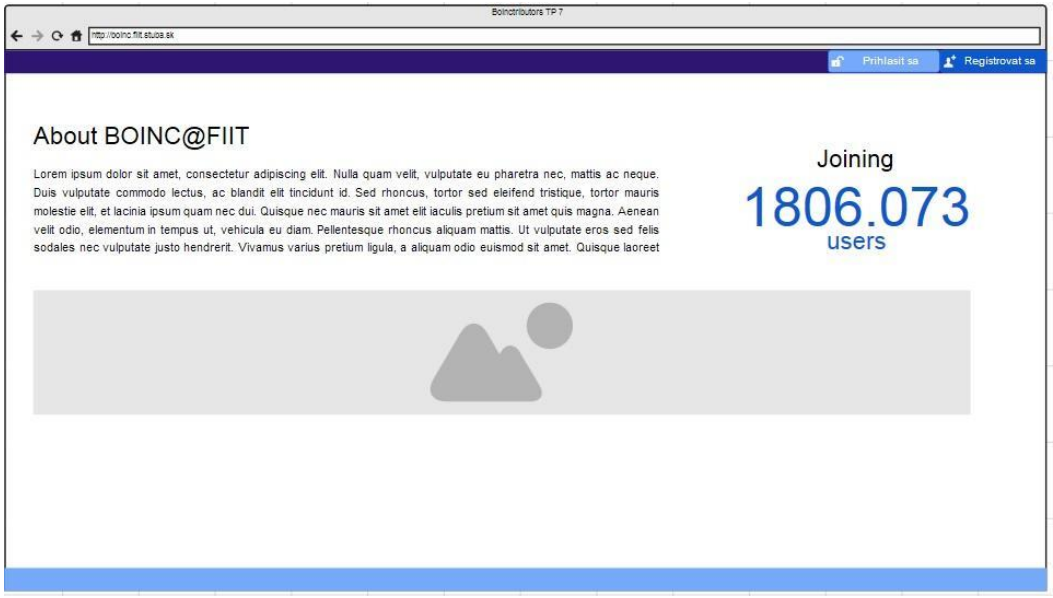


Obr. 3.: Nízkoúrovňový návrh úvodnej stránky projektu

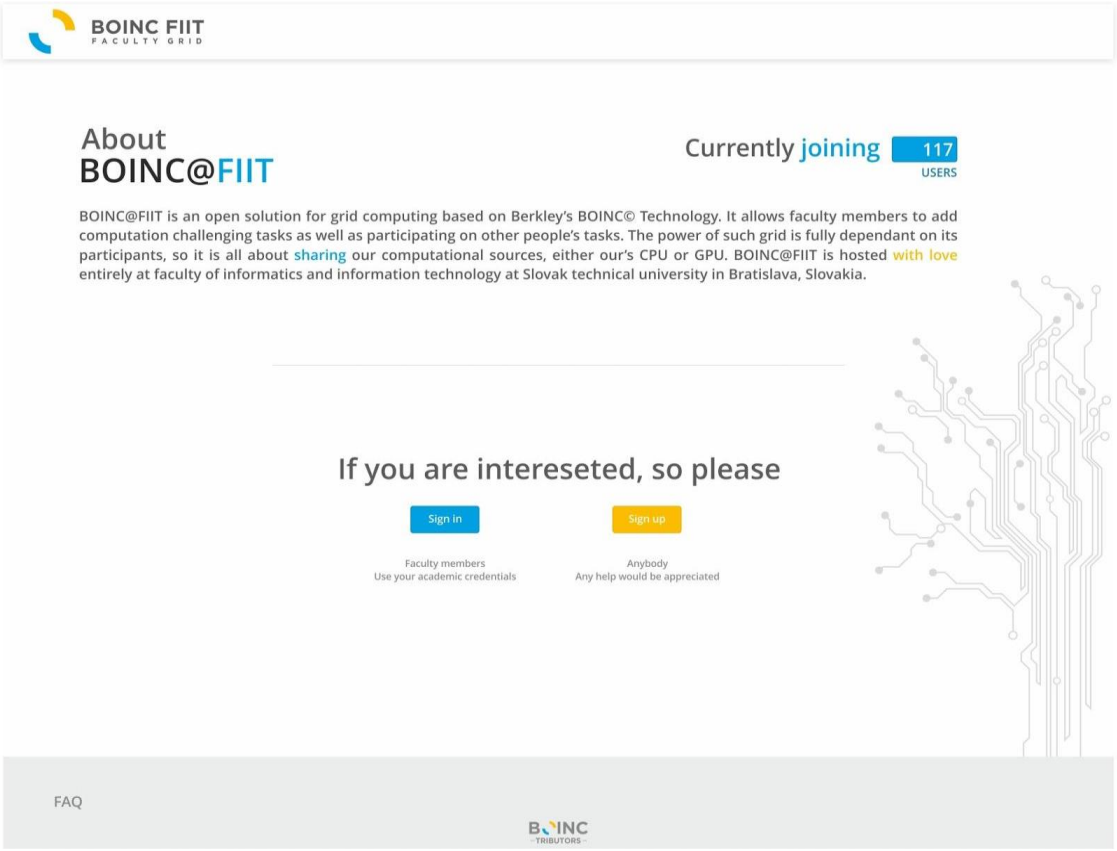
[13] https://moqups.com



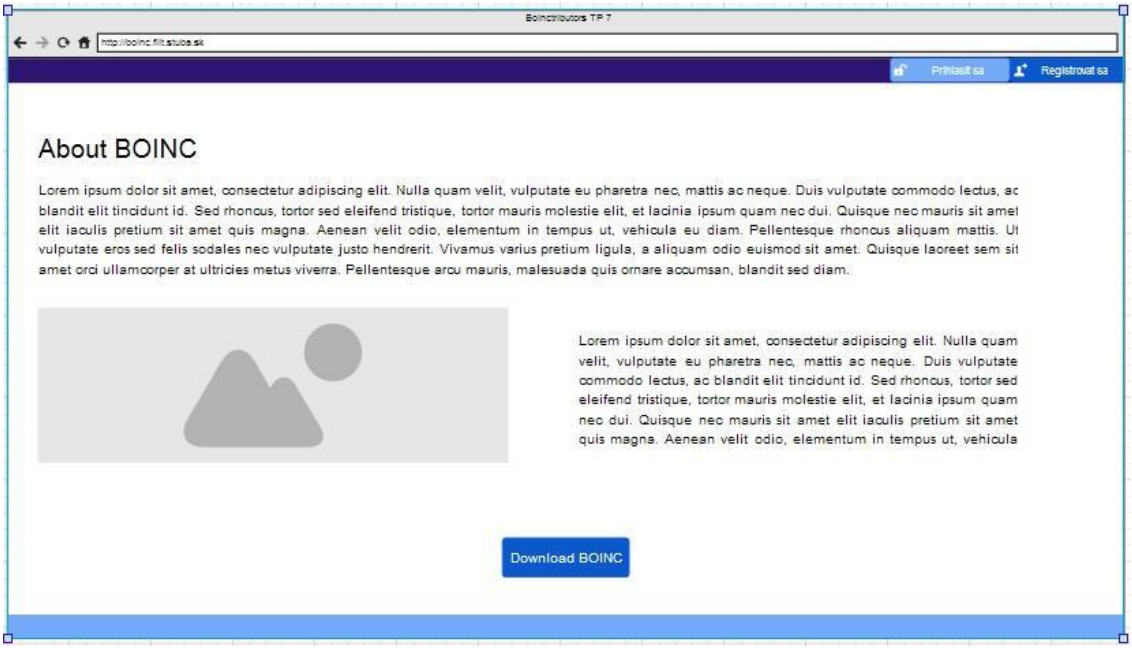
Obr. 4.: Výsledný návrh domovskej stránky projektu



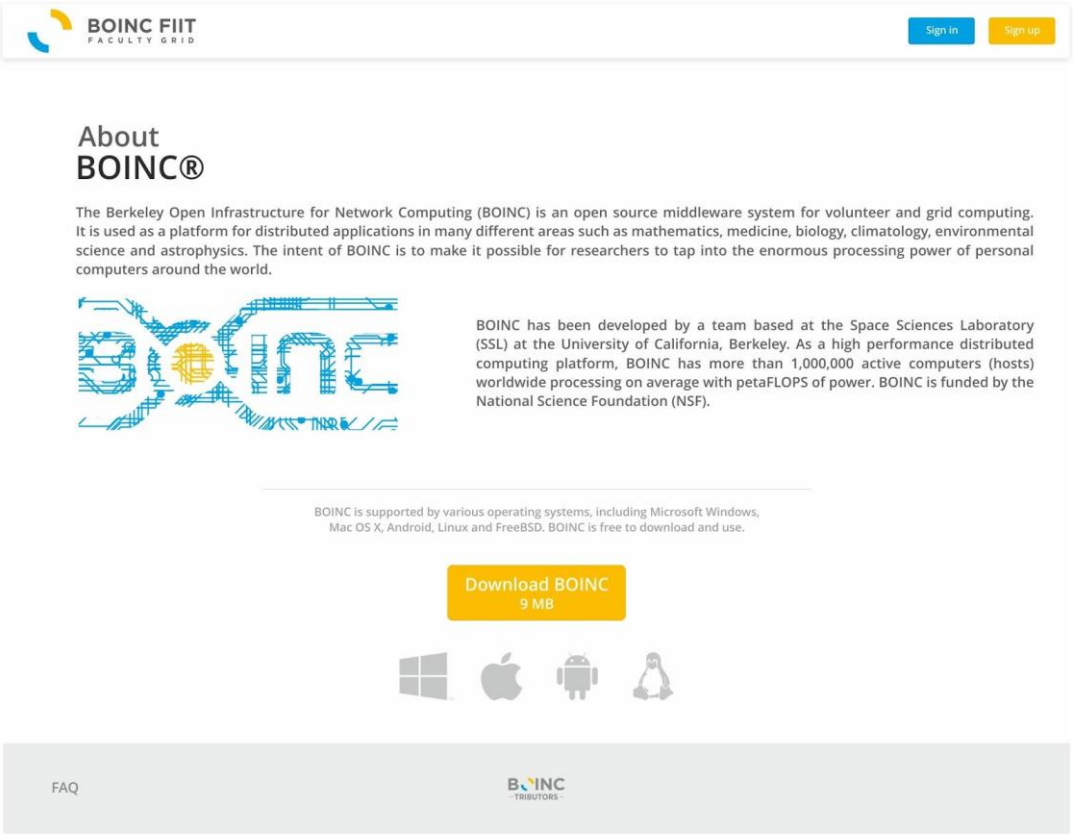
Obr. 5.: Nízkoúrovňový návrh informačnej stránky projektu BOINC@FIIT



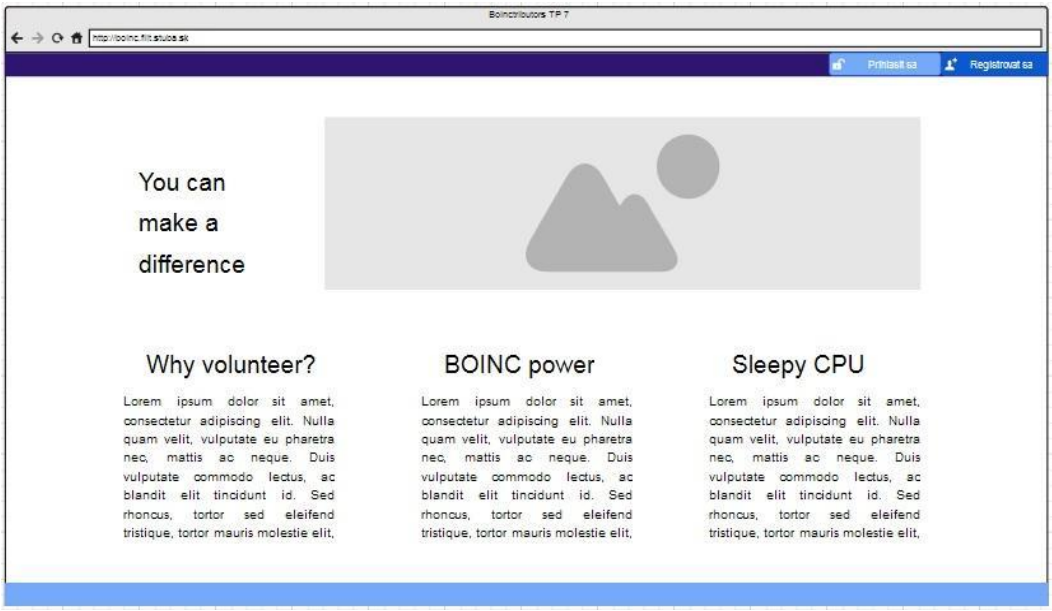
Obr. 6.: Návrh informačnej stránky projektu BOINC@FIIT



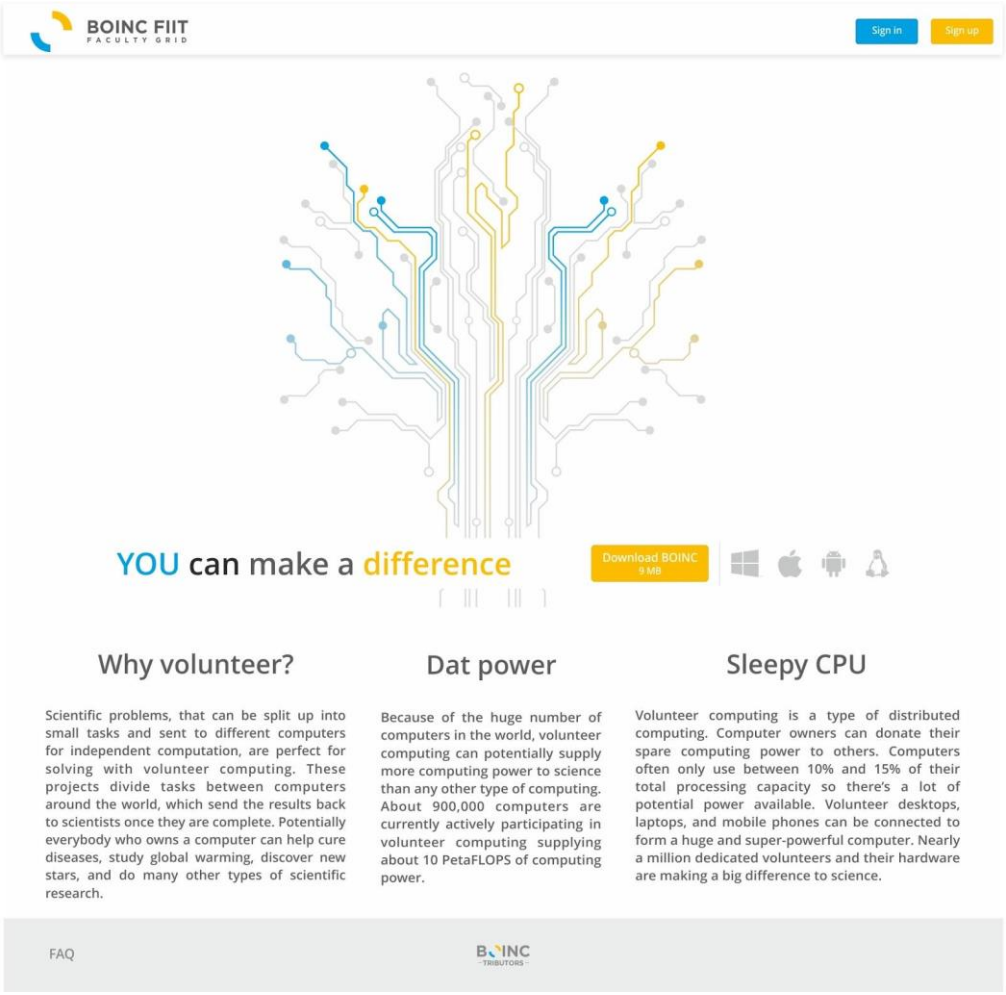
Obr. 7.: Nízkoúrovňový návrh domovskej stránky platformy BOINC



Obr. 8.: Návrh domovskej stránky platformy BOINC



Obr. 9.: Nízkoúrovňový návrh stránky motivujúcej k participácií na projektoch BOINC



Obr. 10.: Návrh stránky motivujúcej k participácii na projektoch BOINC

9.3 Registrácia nového používateľa

9.3.1 Úloha

Zabezpečte možnosť registrácie nového používateľa, ktorý sa chce zapojiť do projektu BOINC@FIIT.

9.3.2 Návrh

Táto úloha sa bude riešiť vo viacerých paralelných krokoch. V prvom rade do RoR aplikácie musíme pridať gem *Devise*, ktorý rieši autentifikáciu používateľa a umožňuje jednoduchú konfiguráciu používateľského modelu v databáze, ako aj logiky, ktorá sa vykonáva pri registrácii, prihlasovaní, zabudnutí hesla a pod. Tento gem zároveň umožňuje ľahko do

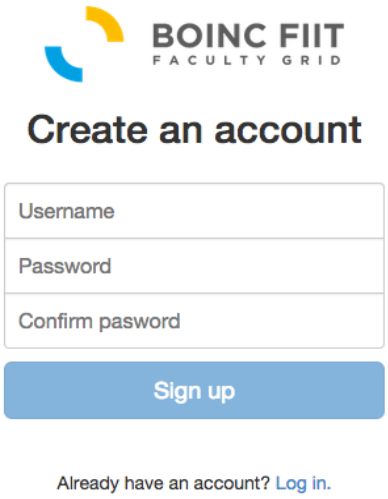
používateľského modelu pridať sledovanie niektorých základných štatistík o správaní používateľa (napr. kedy sa naposledy prihlásil, z akých IP adries sa prihlásil a pod.)

Je nutné podotknúť, že tieto štatistiky budeme ďalej vedieť sledovať zrejme len pri prihlásení na webové rozhranie projektu. Tento predpoklad však kvôli nižšej priorite overíme a otestujeme v niektorom z nasledujúcich šprintov.

V rámci tejto úlohy sa musí vytvoriť aj frontendová časť tohto používateľského príbehu, teda návrh vzhľadu všetkých možných krokov registrácie a tiež napojenie SPA modulu na REST backend.

Klientska časť aplikácie bude komunikovať s REST backendom pomocou správ vo formáte JSON dokumentov. Samotné prihlásenie/registrovanie bude realizované vyplnením formulára a následným zavolaním požiadavky na vopred špecifikovanú URL. SPA aplikácia si bude držať stav prihlásenia/odhlásenia, a na základe toho zobrazí/schová prihlasovací formulár.

Obr. 11.: Návrh formulára pre registráciu



Obr. 12.: Snímka obrazovky skutočného formulára pre registráciu

9.3.3 Implementácia

Ako už bolo spomenuté v návrhu, na implementáciu registračnej logiky sme použili gem *Devise*. Preťažíme controllery, ktoré štandardne ponúka gem *Devise* tak, aby odpovedali vo formáte JSON a tiež ich doplníme o potrebnú logiku. Tiež bude nutné upraviť funkciu na zabezpečenie hesla pri registrácii (a neskôr aj prihlasovaní a overovaní hesla) aby sa zhodovala s hash funkciou, ktorú pre overovanie používa program BOINC.

Kritickou časťou tejto úlohy však bolo spojenie modelu používateľa, ako ho používa pôvodná databáza programu BOINC a nami použitý gem *Devise*. Tieto modely bolo nutné vhodne namapovať a spojiť tak, aby nenastali žiadne konflikty, ktoré by obmedzovali funkčnosť programu BOINC alebo našej aplikácie.

9.3.4 Testovanie

Testovanie REST backendu používame gem *RSpec*, ktorý je štandardom pre testovanie v rámci RoR. Pre čo najmenšie zaťaženie databázy využívame gem *factory_girl* zabezpečujúci vytváranie modelov pre testovanie pomocou factory. Ako bolo spomenuté v odseku vyššie, nami napísané testy pokrývajú základnú funkcionalitu navrhnutého REST API, najmä teda, či odpovedá na JSON požiadavky tak, ako očakávame.

Testovanie v klientskej aplikácií je zabezpečené jednoduchým unit testom. Je nutné podotknúť, že pri unit testoch pre zlepšenie rýchlosti vykonania nie je žiaduce, aby testy prebiehali s reálnymi požiadavkami na server. Preto bol vytvorený „mock“ prihlásenia/registrácie, nad ktorým sa funkcionalita testuje.

Na unit testovanie v SPA bol použitý rámec *Jasmine*[14], pričom testy spúšťame pomocou nástroja *Karma*[15]. *Karma* nám umožňuje spúšťať testy vo virtuálnom prehliadači. Rámec *Jasmine* sme vybrali kvôli tomu, že je primárnou voľbou pri testovaní AngularJS aplikácií, avšak stále existuje možnosť pri ďalších testoch zvoliť iný rámec.

9.4 Autentifikácia

9.4.1 Úloha

Poskytnúť používateľovi možnosť prihlásiť sa do webového rozhrania, pomocou ktorého bude mať možnosť manažovať a sledovať svoje výpočtové úlohy.

9.4.2 Návrh

Návrh spočíva z využitia knižnice *Devise*[16], ktorá pokrýva väčšinu funkcionality pre prihlasovanie. Proces prihlasovania je v *Devise* pokrytý pomocou štandardného HTML rozhrania postaveného na posielaní formulárov a vykresľovaní pohľadov. Tento mechanizmus nezapadá do nášho architektonického návrhu. Preto bude potrebné, rovnako ako v predchádzajúcej úlohe, rozšíriť triedy obsluhujúce prihlasovanie o komunikáciu založenú na REST dopytoch komunikujúcich pomocou JSON správ. Je dôležité mať na pamäti bezpečnostné riziká, ktoré nám z návrhu REST služieb vyplývajú. Základným bezpečnostným rizikom, ktoré musíme ošetriť, je CSRF. SQL Injection je v tomto prípade ošetrené na úrovni *ActiveRecord*[17] v RoR.

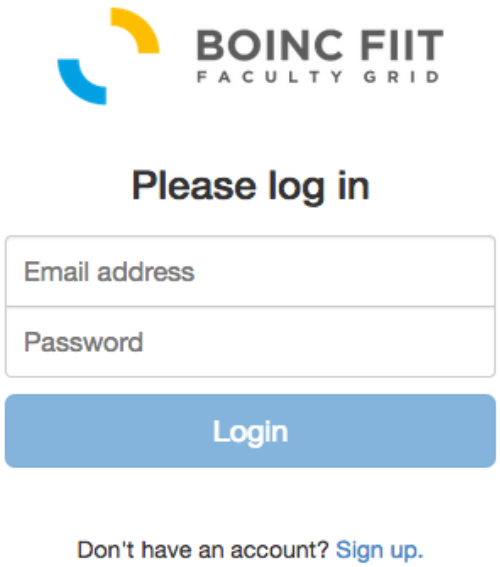
[14] Jasmine: http://jasmine.github.io/
[15] Karma: http://karma-runner.github.io/
[16] Devise: https://github.com/plataformatec/devise
[17] ActiveRecord: http://guides.rubyonrails.org/active_record_basics.html

Obr. 13.: Návrh prihlasovacieho formulára.

9.4.3 **Implementácia a testovanie**

Klientska časť aplikácie zobrazí používateľovi formulár na prihlásenie. Stav o prihlásení používateľa je na backend strane riešený pomocou http sedení, preto nie je potrebné v aplikácií držať token, ktorý by následne autentifikoval každý dopyt. Namiesto toho, požiadavka, ktorá je vyhodnotená ako neautorizovaná (HTTP kód odpovede 403), spôsobí v SPA aplikácií odhlásenie a zobrazenie prihlasovacieho formulára. Automatické odhlásenie používateľa zo sedenia nastane po dlhšej nečinnosti alebo samotnom odhlásení používateľom.

Na strane backendu je tato funkcionalita implementovaná taktiež pomocou gemu *Devise*. Pre HTTP sedenia sme preťažili všeobecný controller gemu *Devise* a vytvorili samostatný controller, ktorý spracúva požiadavky HTTP požiadavky a odpovedá na ne vo formáte JSON.



Obr. 14.: Reálny formulár pre prihlásenie

9.5 Zhodnotenie šprintu

Druhý šprint slúžil prevažne na návrh výslednej aplikácie, ako SPA, tak aj REST modulu projektu. Definovali sme jednotlivé postupy prenosu údajov medzi týmito modulmi, ako aj medzi pôvodnou BOINC databázou a naším SPA modulom. Schéma aj charakter pôvodnej BOINC databázy bol upravený pre potreby našich modulov. Venovali sme sa záväznému stanoveniu si implementačných detailov, no po stránke komunikácie v tíme to stále nedosahovalo požadovanú kvalitu. V druhom šprinte sme sa venovali aj zefektívneniu prác so systémom na správu projektov JIRA.

9.5.1 Retrospektíva šprintu

Dátum:	12.11.2014		
Miesto stretnutia:	U80		
Čas:	16:30		
Účastníci:	Členovia tímu:	Bc. Juraj Kochjar	Bc. Martin Kaššay
		Bc. Matej Kloska	Bc. Patrik Gallik
		Bc. Pavol Čurilla	Bc. Roman Roštár
Vypracoval:	Bc. Martin Kaššay		
Téma stretnutia:	Zhodnotenie práce počas druhého šprintu.		

Výstup zo stretnutia:

Okruh	Pozitívum	Negatívum
Analýza a návrh	proces navrhovania návrhy rozhraní aplikácie wireframes	nedostatočné načasovanie postupov nejasné, ako automatizovať procesy nedostatočná analýza
Plánovanie	ukladanie úloh v rámci systému JIRA	plánovanie šprintov ohodnocovanie úloh zlé vytváranie backlogu
Tím a stretnutia	morálka tímu zlepšenie efektivity a výstupov tímu v porovnaní z predošlým šprintom lepšia komunikácia ako v minulom šprinte	celková komunikácia v tíme neskoré uverejňovanie dokumentov chýbajúca motivácia nedodržiavanie termínov zle využitý čas na stretnutiach produktívne vákuum v prvej polovici šprintu
Programovanie	pokrok v implementácii	nedostatočné vytváranie testov rozdiely medzi SPA a REST implementáciami, nekompatibilita

Zhodnotenie stretnutia

Hoci druhý šprint dopadol lepšie ako ten prvý, stále pociťujeme zlyhávanie komunikácie v tíme. Aj keď nastal pokrok v oblasti implementácie, samotné programovanie funkcionality v rámci SPA a REST modulov je časovo posunuté. Vzniká ako časový, tak aj kvantitatívny rozdiel medzi týmito modulmi. Čo sa týka grafických návrhov výsledného rozhrania SPA aplikácie, tie markantne pokročili. Zistili sme, že sme nesprávne používali systém JIRA, čo v konečnom dôsledku spôsobilo nekorektné exporty našej práce.

10 Šprint 3 - Jameson

10.1 Sprístupnenie informácií novému návštevníkovi

10.1.1 **Úloha**

Poskytnúť používateľovi relevantný a motivujúci informačný obsah o základných princípoch zdieľania výpočtového výkonu v rámci platformy BOINC. Priblížiť používateľom fakultný projekt BOINC@FIIT spolu s návodom ako sa stať jeho participantom.

10.1.2 **Implementácia**

Pre bližšie informovanie o projekte a zároveň pre motivovanie nových návštevníkov k zapojeniu sa do projektu zdieľaním výkonu svojho počítača v čase jeho nečinnosti, sme nasledujúci obsah poskytli používateľovi na webovom sídle nášho projektu:

- Čo je to BOINC – používateľa informujeme o samotnom systéme BOINC, jeho princípoch, účele, používateľoch, vzniku, tvorcoch, pracoviskách, ktoré ho využívajú a o platformách, pre ktoré je dostupný.
- Čo je to dobrovoľné poskytnutie výpočtového výkonu – poskytujeme informácie o množstve nevyužitého výkonu v bežnej prevádzke počítača, zároveň informujeme používateľa o možnostiach tento výkon poskytnúť v prospech projektu BOINC@FIIT.
- O projekte BOINC@FIIT – sprístupňujeme informácie o aktuálne spustených aplikáciách v rámci BOINC@FIIT a o počte ich aktívnych používateľov.
- Prečo byť dobrovoľníkom – motivujeme používateľa k poskytnutiu svojho, často, nevyužitého výpočtového výkonu počítača na prospešné a potrebné účely, a vysvetľujeme ako môže aj jeden používateľ znamenať pokrok a ovplyvniť smerovanie vedeckých výskumov.
- Ako začať – poskytujeme základný návod k inštalácii a spusteniu klientskej aplikácie BOINC, a opisujeme nevyhnutné kroky pre sprístupnenie svojho nadbytočného výpočtového výkonu v prospech systému BOINC, resp. jej fakultnej inštancie.

K spomínaným textom sa používateľ bude môcť dostať pri navštívení domovskej stránky projektu BOINC@FIIT a k nej prislúchajúcich odkazov. Po dohode s vedúcim tímového projektu sú domovská stránka projektu BOINC@FIIT a všetky prislúchajúce stránky webového sídla projektu v anglickom jazyku.

10.2 Práca s existujúcou DB schémou

10.2.1 Úprava schémy vzhľadom na BOINC schému

10.2.1.1 Úloha

Zlúčiť existujúcu databázovú BOINC schému so schémou, ktorá je požadovaná modulom Devise pre registráciu a prihlasovanie.

10.2.1.2 Implementácia

Pred samotným zlúčením schém existovala samostatne BOINC schéma vo svojej databáze a samostatne schéma pre *Devise* vo svojej. Pri zlúčení sme museli spojiť tabuľku *user* (BOINC) a tabuľku *users* (*Devise*). Pre zaistenie kompatibility v názvoch tabuliek z pohľadu BOINC sme sa rozhodli ponechať názov *user* a toto pomenovanie premietnuť do nastavení *Devise* modulu. Zásadný problém nastal pri spájaní stĺpcov tabuliek, nakoľko bolo potrebné zjednotiť názvy stĺpcov pre ukladanie emailovej adresy a hesla. Prioritne sme sa snažili prispôsobiť názvom systému BOINC, nakoľko do jeho zdrojových kódov nechceme zasahovať. Z tohto dôvodu sa použil názov stĺpca *email_addr* namiesto email. Tento stav bolo potrebné premietnuť do nastavení *Devise*.

Zásadnejším problémom bolo ukladanie hesla, nakoľko BOINC PHP generovanie hesla používa iný spôsob ako *Devise* RoR. V tomto prípade sme sa rozhodli ponechať oba stĺpce a na aplikačnej úrovni dopočítať heslo pre BOINC PHP. Tento špecifický spôsob generovania hesla pre BOINC môže spôsobiť pri prechode na novú verziu potenciálne problémy a je nutné na to myslieť pri testovaní produktu pri prechode.

10.2.2 RoR ORM mapovanie

10.2.2.1 Úloha

Vygenerovať *ActiveRecord* modely pre databázové tabuľky existujúcej schémy BOINC.

10.2.2.2 Implementácia

RoR v aktuálne verzii, ktorú používame pri vývoju produktu nepodporuje reverzné vytváranie *ActiveRecord* modelov na základe existujúcich databázových tabuliek. Na internete vznikla komunita okolo aktívne vyvíjaného Ruby gemu RMRE, ktorý ponúka danú funkcionalitu. Na oficiálnej stránke projektu[18] je možné nájsť postup akým sa dajú pomocou jedného príkazu vygenerovať základné modely. Po vygenerovaní modelov bolo potrebné prejsť k doplneniu väzieb medzi modelmi, nakoľko DB schéma je neúplná a nie všetky väzby boli identifikované. V tomto prípade sme sa rozhodli pre doplnenie iba jednoznačných väzieb, nakoľko neexistuje

[18] RMRE GitHub: https://github.com/bosko/rmre

oficiálna dokumentácia k schéme a v čase generovania modelov nebolo jasné, ktoré modely bude reálne potrebné k ďalšiemu vývoju.

10.3 Nastavenia používateľského účtu

10.3.1 Úloha

Umožniť používateľovi zmeniť nastavenia svojho účtu. Tieto nastavenia by mali kopírovať možnosti nastavení, ktoré sa nachádzajú v pôvodnej verzii webového rozhrania BOINC. To znamená možnosť zmeniť osobné údaje ako email a meno ako aj zmeniť heslo.

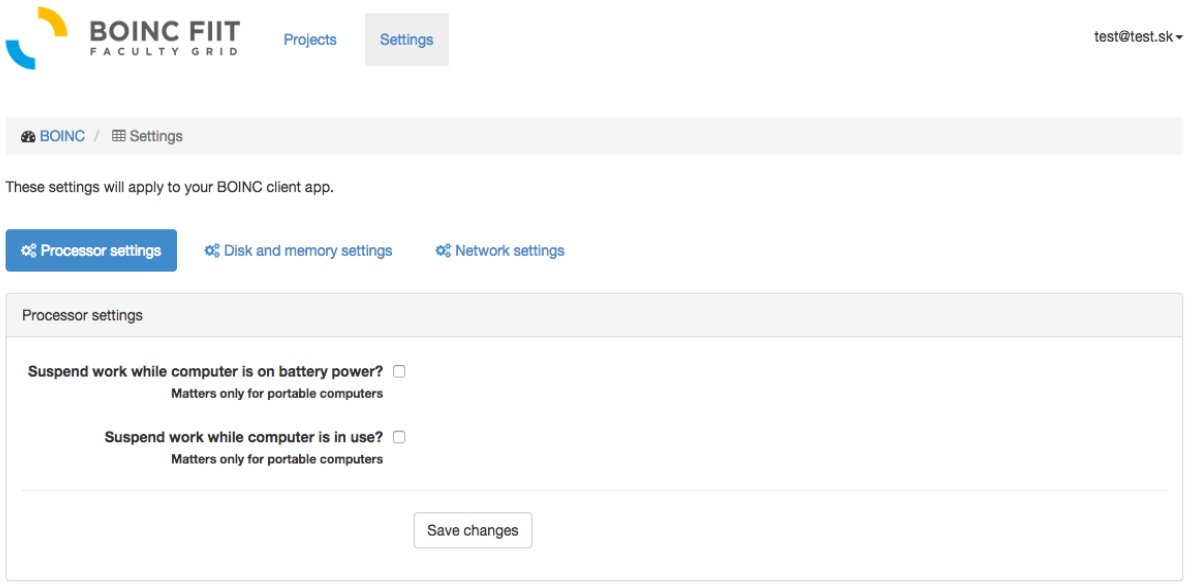
10.3.2 Návrh

Návrh serverovej strany aplikácie úzko súvisí s úlohou „Práca s existujúcou DB schémou“. Je potrebné namapovať existujúce údaje k účtu používateľa v databáze na RoR ORM, a poskytnúť ich cez REST API. Samotný návrh obrazovky nebol potrebný, dohodli sme sa na podobnom formulári ako bol v starom systéme BOINC s tým rozdielom, že nastavenia sa rozdelia do záložiek podľa typu.

10.3.3 Implementácia

Pomocou použitého rámca Bootstrap sme v klientskej časti pomerne rýchlo vytvorili formuláre, ktoré sme zapracovali do AngularJS šablón. Pre každý typ nastavenia (zmena hesla, zmena osobných údajov) sme vytvorili samostatnú obrazovku. Prístup do nastavení má používateľ po kliknutí na názov svojho účtu v pravej hornej časti obrazovky. Samotné nastavenia sa načítajú a upravia volaním špecifickej REST požiadavky na server.

V klientskej časti poskytuje potrebné metódy na získanie a úpravu používateľských nastavení služba *SettingsService*. Je to návrhový vzor *singleton*, takže jeho inštancia je dostupná počas celého behu aplikácie.



Obr. 15.: Formulár pre nastavenie BOINC klientov

10.4 Zhodnotenie šprintu

Tretí šprint slúžil na implementáciu používateľského rozhrania. Na strane REST modulu bolo potrebné spraviť migráciu databázy a doplnenie niektorých stĺpcov, keďže *RoR* device modul pre manipuláciu s dátovou entitou používateľ dodržiaval iné konvencie názvov pre stĺpce ako poskytovala platforma BOINC. Čo sa týka SPA modulu, začala sa implementácia nastavení používateľského účtu, ako aj nastavení pre samotnú aplikáciu BOINC. Nakoniec bola dokončená statická stránka pre prezentovanie platformy BOINC pre neprihláseného používateľa, kde sa čaká už len na zobrazovanie reálnych dát.

10.4.1 **Retrospektíva šprintu**

Dátum:	20.11.2014		
Miesto stretnutia:	Jobsovo softvérové štúdio		
Čas:	12:30		
Účastníci:	Členovia tímu:	Bc. Juraj Kochjar	Bc. Martin Kaššay
		Bc. Matej Kloska	Bc. Patrik Gallik
		Bc. Pavol Čurilla	Bc. Roman Roštár
Vypracoval:	Bc. Martin Kaššay		
Téma stretnutia:	Zhodnotenie práce počas tretieho šprintu.		

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	proces navrhovania návrhy rozhraní aplikácie wireframes	wireframes na poslednú chvíľu problémy s automatizovaním procesov pri nasadzovaní
Plánovanie	pochopili sme, ako sa dostať k burndown chart	plánovanie šprintov ohodnocovanie úloh zlé vytváranie backlogu
Tím a stretnutia	morálka tímu zlepšenie efektivity a výstupov tímu v porovnaní z predošlým šprintom	celková komunikácia v tíme neskoré uverejňovanie dokumentov chýbajúca motivácia nedodržiavanie termínov zlý rozbeh šprintu
Programovanie	pokrok v implementácii SPA modulu	zanedbávanie vytvárania testov

Zhodnotenie stretnutia:

Môžeme skonštatovať, že tretí šprint z hľadiska komunikácie dopadol zatiaľ najlepšie. V polovici šprintu sme zlepšili procesy súvisiace s manažmentom pridávania a zadávania úloh v nástroji JIRA tak, aby sme mali aj korektné výstupy v podobe Burndown grafov. Na strane REST modulu sa vyskytli nezrovnalosti s databázou a bolo potrebné zasiahnuť a upraviť jej štruktúru. V module SPA pokročila implementácia ako v oblasti vytvárania výsledných používateľských rozhraní, tak aj pri implementácií klientskych nastavení, či už sa jedná o nastavenie používateľovho profilu, alebo samotnej platformy BOINC.

11 Šprint 4 – Four Roses

11.1 Automatizované nasadzovanie

11.1.1 Úloha

Uľahčenie práce vývojárom pri nasadzovaní nových verzií aplikácie na server, ako aj pri prepájaní už existujúcich vetiev repozitárov zdrojového kódu.

11.1.2 Návrh

Pokiaľ dôjde k potvrdeniu akcie *pull-request* a nastane prepojenie nejakej vetvy do hlavnej vývojovej vetvy projektu, alebo do produkčnej vetvy projektu, dôjde k automatickému spusteniu skriptu, ktorý spustí všetky nutné procesy, ako napr. testy, overenie závislostí, a pod. Pokiaľ sú všetky procesy úspešné, tak sa daná produkčná vetva stáva aktuálnou verziou produktu a je nasadená na server.

11.1.3 Implementácia

Pri implementácií sme využili službu pre kontinuálnu integráciu *codeship.io*[19]. Pri každom odovzdaní do vývojovej alebo hlavnej vetvy projektu sa spustí integrácia pomocou služby *codeship*. V prvom rade sa nastaví prostredie pre projekt, vrátane vytvorenia celej databázovej schémy. Druhým krokom je spustenie všetkých testov projektu. Pokiaľ sú všetky testy akceptované, tretím krokom je spustenie skriptu pre automatické nasadenie. Pre nasadenie používame gem *Capistrano*[20], ktorý prekopíruje novú verziu na server, spustí automatický *build* aplikácie a nakoniec reštartuje server.

11.2 Nasadenie aktuálnej verzie na server

11.2.1 Úloha

Nasadenie aktuálnych verzií modulu REST a SPA aplikácie na server, ako aj doplnenie údajov na webovú stránku tímového projektu pre sfinalizovanie a ukončenie 1. kontrolného bodu.

[19] https://codeship.com/projects
[20] https://github.com/capistrano/capistrano

11.2.2 Nasadenie

Pred finálnym nasadením v zimnom semestri sme dokončili vývojové vetvy, ktorých funkcionalitu sme sa rozhodli skompletizovať k prvému kontrolnému bodu. Následne sme vytvorili *pull-request.* Ten prešiel metódou prehliadky kódu, tak ako sme si záväzne stanovili v príslušnej metodike. Až po odstránení prípadných nezrovnalostí boli následne vývojové vetvy prepojené do vetiev *develop* a *master*. Pre webovú stránku tímového projektu boli vygenerované burndown grafy a globálne reporty pre jednotlivé šprinty. Tie boli vložené spolu so všetkými zápisnicami zo stretnutí, retrospektívami šprintov, ako aj dokumentmi riadenia a inžinierskeho diela k prvému kontrolnému bodu.

11.3 Pridanie lightbox

11.3.1 Úloha

Aby sme verejnosti priblížili spôsob našej práce počas stretnutí, rozhodli sme sa zverejniť niektoré fotografie našich náčrtov vytvorených počas tímových stretnutí. Všetci členovia tímu sa zhodli na tom, že bude vhodné vytvoriť galériu pre prehliadanie týchto fotografií a nepísaných podkladov.

11.3.2 Návrh

Rozhodli sme sa zobrazenie fotografií implementovať ako jednoduchú galériu. Po kliknutí na miniatúru fotografie sa zobrazí jej plnohodnotný ekvivalent.

11.3.3 Implementácia

Na webové sídlo tímu bol pomocou *Bower* manažéra pridaný nový zásuvný modul *Magnific Popup*[21]. Tento modul bez väčšej námahy pokryl všetky naše požiadavky. Do výsledného *HTML* kódu webovej stránky boli pridané obrázky a vďaka nainštalovanému modulu je zobrazovanie plnohodnotných fotografií automatické. Grafická stránka modulu bola prijateľná, a preto sme sa ju rozhodli ponechať.

11.4 Technická prezentácia projektu

11.4.1 Úloha

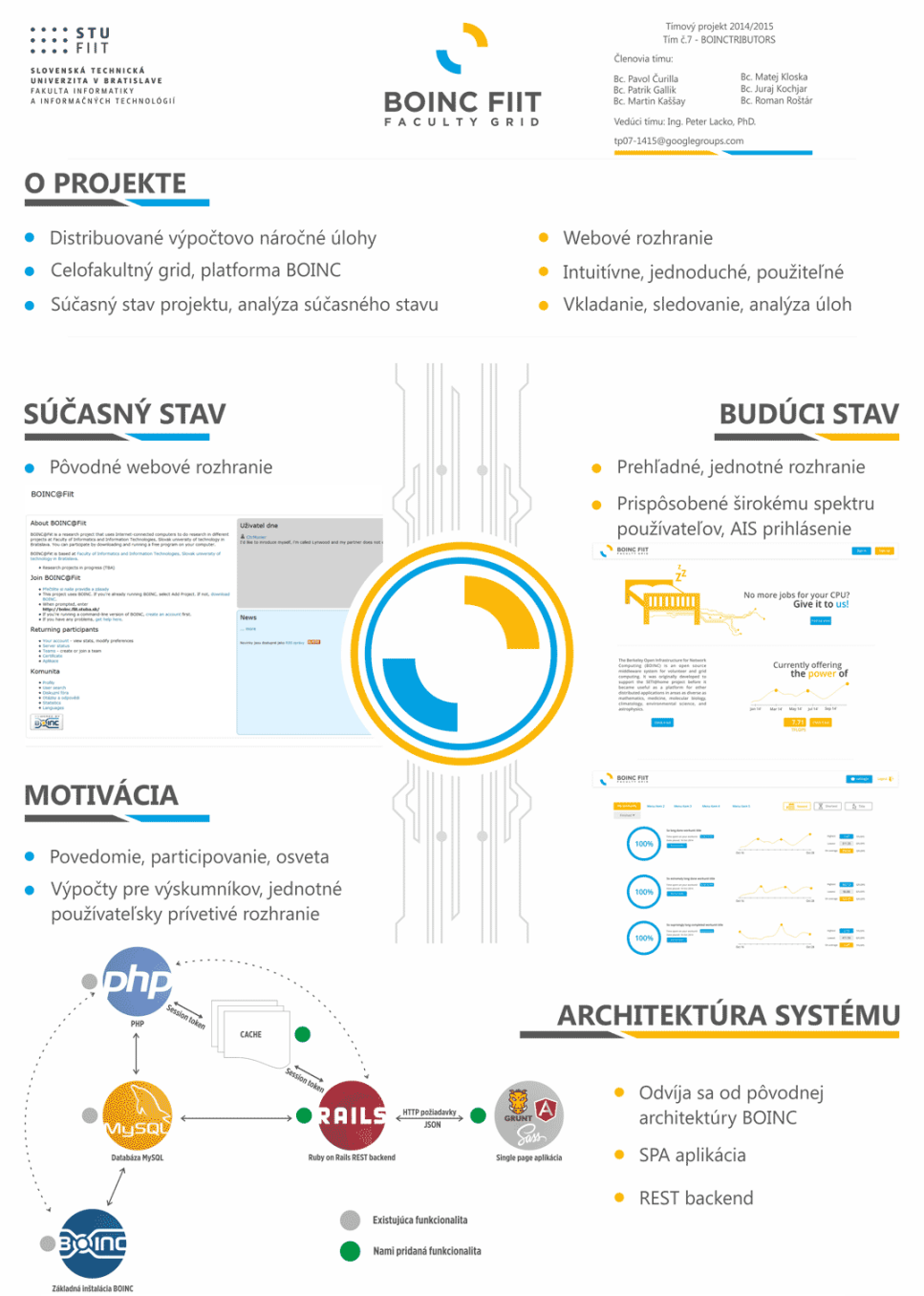
Pripravenie a nacvičenie panelovej prezentácie nášho tímového projektu.

[21] http://dimsemenov.com/plugins/magnific-popup

11.4.2 **Príprava**

Proces prípravy tvorili dôsledné vytvorenie tímovej prezentácie, spolu s vytvorením vizuálu plagátu. Prezentáciu sme si niekoľkokrát nacvičili s prihliadnutím na komunikačné pravidlá prezentované v rámci prednášok predmetu tímový projekt.

11.4.3 **Výstup**



Obr. 16.: Plagát prezentovaný na panelovej prezentácii tímu

11.5 E2E testy v SPA module

11.5.1 Úloha

Na serverovej aj klientskej časti aplikácie sú prítomné unit testy, ktoré testujú izolované časti zdrojového kódu. Potrebovali sme vytvoriť testy, ktorý by pokrývali naraz výslednú funkcionalitu klientskej, ale aj serverovej časti.

11.5.2 Návrh

Ako najvhodnejšie riešenie sa javilo použitie End to End testov (ďalej E2E). Tieto testy predstavujú otestovanie čo najväčšieho počtu používateľských úkonov, pričom celý tento proces sa deje automatizovane. Ako najvhodnejšie riešenie na E2E testovanie v rámci Angular aplikácií (Angular je rámec napísaný v jazyku JavaScript, ktorý používame na zobrazovanie klientskej časti), sa javil testovací rámec *Protractor*[22].

11.5.3 Implementácia

Na implementáciu E2E testov sme použili rámec *Protractor*. Tento rámec používa syntax rámca *Jasmine,* ktorý už používame na unit testy v rámci klientskej časti. Samotné E2E testovanie je realizované pomocou nástroja *Selenium*[23]. *Protractor* nám umožňuje preklad našich testov z jazyku JavaScript do jazyku Selenia, pričom berie do úvahy charakter rámca Angular. Písanie testov je vďaka tomu veľmi jednoduché a odbremeňuje nás od riešenia problémov s rôznymi asynchrónnymi akciami, ktoré rámec Angular používa. Pomocou týchto testov sme pokryli prvý zložitejší používateľský úkon, a to prihlásenie. Bola pokrytá väčšina testovacích scenárov, ktoré môžu pri tomto úkone nastať, ako napr. nesprávne heslo, nezadanie žiadnych údajov, ale aj samotné presmerovanie po úspešnom prihlásení.

11.6 SPA – Nastavenie používateľského profilu, ako aj polí na nastavenie preferencií platformy BOINC

11.6.1 Úloha

Ďalšou časťou pri implementovaní pôvodnej funkcionality rozhrania BOINC bolo umožnenie používateľovi meniť systémové nastavenia a preferencie počítania pre platformu BOINC, ale takisto nastavenie týkajúce sa jeho osobného profilu.

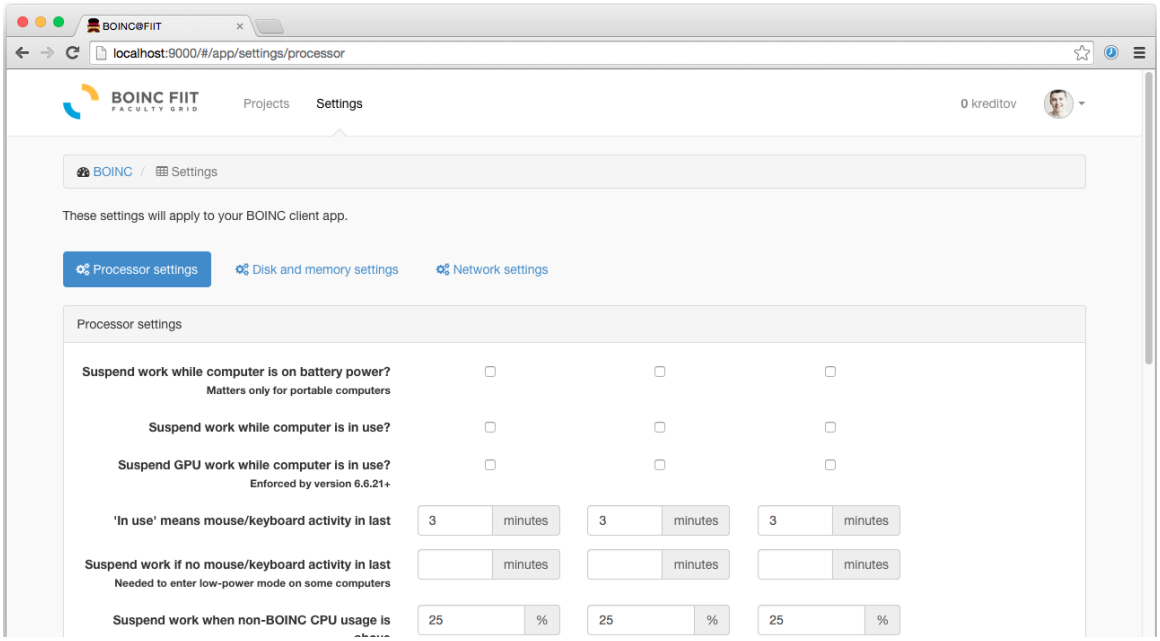
[22] https://github.com/angular/protractor
[23] http://www.seleniumhq.org/

11.6.2 **Návrh**

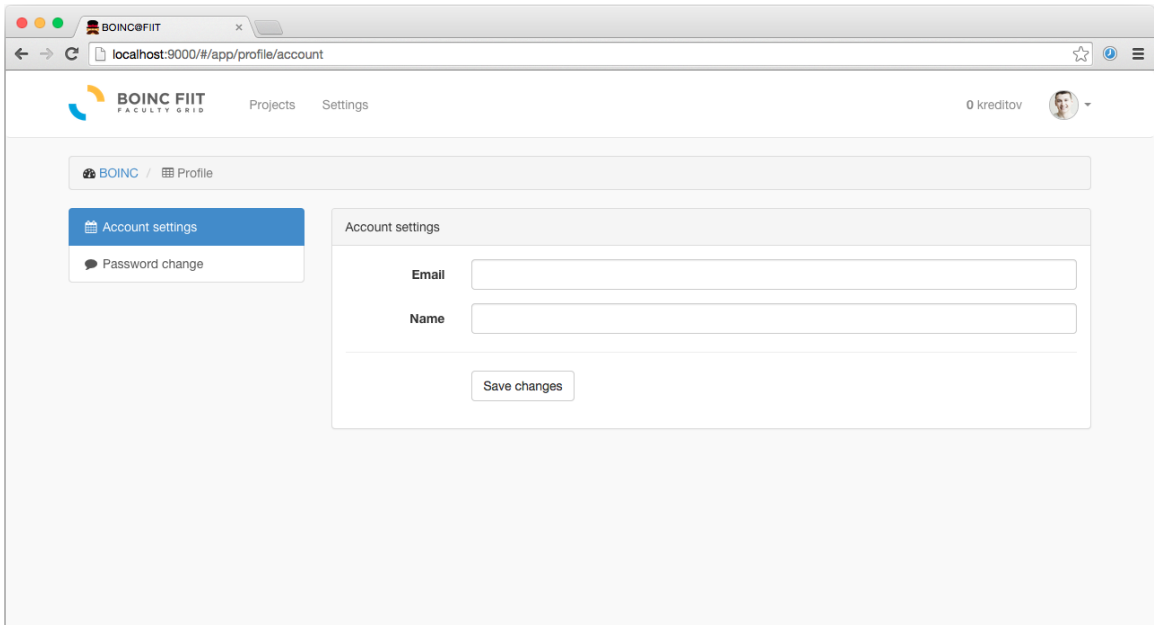
Aby nevznikali problémy s rozlišovaním týchto dvoch druhov nastavení, rozhodli sme sa nastavenia používateľovho profil volať jednoducho „Profil“ a nastavenia platformy BOINC „Nastavenia“. Rozhodli sme sa ponechať rozloženie polí tak, ako boli zobrazené so základnou serverovou inštaláciou platformy BOINC. Týmto rozhodnutím sme eliminovali prípadné zmätenie používateľa spôsobené neprehľadnosťou rozhrania vyvolanou prechodom na našu verziu rozhrania projektu BOINC@FIIT.

11.6.3 **Implementácia**

Pomocou frontend rámca Bootstrap sme vytvorili a implementovali obrazovky znázorňujúce vyššie spomenuté možnosti nastavenia v našom SPA module. Tieto obrazovky majú vlastnú *URL* adresu, pričom obrazovka prezentujúca nastavenie platformy BOINC je prístupná z hlavnej navigácie, zatiaľ čo obrazovka prezentujúca nastavenie používateľovho profilu je dostupná až po rozkliknutí príslušného prvku umiestneného v pravej hornej časti obrazovky. Nastavenie platformy BOINC sme oproti pôvodným nastaveniam zjednodušili tým, že nastavenia pre rôzne prostredia sme umiestnili vedľa seba do stĺpcov.



Obr. 17.: Zobrazenie nastavení platformy BOINC



Obr. 18.: Zobrazenie používateľských nastavení v rámci SPA modulu

11.7 Zhodnotenie šprintu

V štvrtom šprinte sme pozorovali mierne zlepšenie a zefektívnenie vývoja narozdiel od toho tretieho, no ku koncu šprintu slabá komunikácia viedla k nezhodám medzi časťami REST a SPA, kde sa objavila mierna nekompatibilita implementovaných komponentov. Podcenenie manažmentu rizík viedlo k oneskoreniu niektorých termínov, k čomu prispelo aj neadektvátne narábanie s nástrojom JIRA, kde sme badali rozdiely medzi našimi a výslednými časovými estimáciami. Pozitívom bolo zautomatizovanie procesu nasadzovania a zavádzanie a praktická implementácia testov v rámci obidvoch vyvíjaných moduloch.

11.7.1 **Retrospektíva šprintu**

Zápis z retrospektívneho stretnutia po 4. šprinte

Dátum:	11.12.2014		
Miesto stretnutia:	Jobsovo softvérové štúdio		
Čas:	16:30		
Účastníci:	Členovia tímu:	Bc. Juraj Kochjar	Bc. Martin Kaššay
		Bc. Matej Kloska	Bc. Patrik Gallik
		Bc. Pavol Čurilla	Bc. Roman Roštár
Vypracoval:	Bc. Martin Kaššay		

Téma stretnutia: Zhodnotenie práce počas štvrtého šprintu.

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	plagát na prezentáciu automatické nasadzovanie	wireframes na poslednú chvíľu
Plánovanie	vytváranie podúloh	plánovanie šprintov ohodnocovanie úloh zlé vytváranie backlogu
Tím a stretnutia	morálka tímu zlepšenie efektivity a výstupov tímu v porovnaní z predošlým šprintom	neskoré uverejňovanie dokumentov chýbajúca motivácia nedodržiavanie termínov
Programovanie	implementácia light box na tímovú stránku vytvorenie E2E testov codeship.io a capistrano implementácia nastavení	rozdiely medzi SPA a REST implementáciami, nekompatibilita

Zhodnotenie stretnutia:

V štvrtom šprinte sme sa snažili dokončiť automatizované nasadzovanie scriptov na server. Využili sme službu pre kontinuálnu integráciu *codeship*.io. Pre nasadzovanie sme sa rozhodli použiť gem Capistrano. Plánovanie nám nevyšlo podľa predstáv aj preto, lebo sme ho upravovali aj po spustení šprintu, čo zapríčinilo skreslenie výsledného Burndown grafu. Taktiež sme sa pripravovali na prezentáciu tímu vytvorením plagátu a prezentácie. Na tímovú stránku sme dorobili light box pre úhľadnejšie zobrazovanie príloh a fotiek. Začali sme s vytváraním testov pre SPA. Zvolili sme E2E testy.

12 Šprint 5 – Johnnie Walker

12.1 Zjednotenie repozitárov

12.1.1 Úloha

Pri práci v predchádzajúcom semestri boli zdrojové kódy frontendu a backendu oddelené v samostatných repozitároch. To spôsobovalo problémy pri zjednocovaní verzií frontend/backendu pri vývoji. Takisto vznikol problém pri nasadzovaní produkčnej verzie, keď bolo vytvorenie buildu aplikácie príliš komplikované. Preto sme sa rozhodli (aj na odporúčanie pri obhajobe z predmetu Manažment v informačných systémoch) zdrojové kódu spojiť do jedného repozitára.

12.1.2 Návrh

Cieľom je ponechať adresár backendu a do samostatného adresára vložiť zdrojový kód fronentu. SPA aplikáciu potom napojiť na backend tak, aby bola spustiteľná zároveň s backendom.

12.1.3 Implementácia

Kompletne celý zdrojový kód klienta (front-end) sme presunuli do repozitára back-endu do adresára /client. Všetky doterajšie spôsoby vývoja klienta (build, kompilácia sass) zostali nezmenené a spúšťajú sa pomocou nástroja Grunt. Pribudol však rake task s názvom *client:start,* ktorý zostaví front-end, nalinkuje ho do public adresára rails aplikácie, ktorá sa následne spustí a je k dispozícii na localhoste. Samotný klient aplikácie je potom prístupný na adrese *localhost:3000/app/*.

12.1.4 Testovanie

Nový spôsob najprv používali len dvaja členovia tímu, neskôr sa pridali ostatní. Odozva bola pozitívna a všetci sme sa zhodli, že je to značné vylepšenie procesov pri vývoji, a teda správne rozhodnutie. Nasadzovanie na produkciu bolo značne zjednodušené.

12.2 Zabudnuté heslo

12.2.1 Úloha

Potrebujeme používateľovi umožniť zresetovať zabudnuté heslo. Riešenie by nemalo nijak vybočovať zo zaužívaných štandardov.

12.2.2 **Návrh**

Pod prihlasovací formulár je potrebné pridať odkaz na obnovenie hesla. Náš systém budeme teda musieť rozšíriť o tieto dve nové funkcionality:

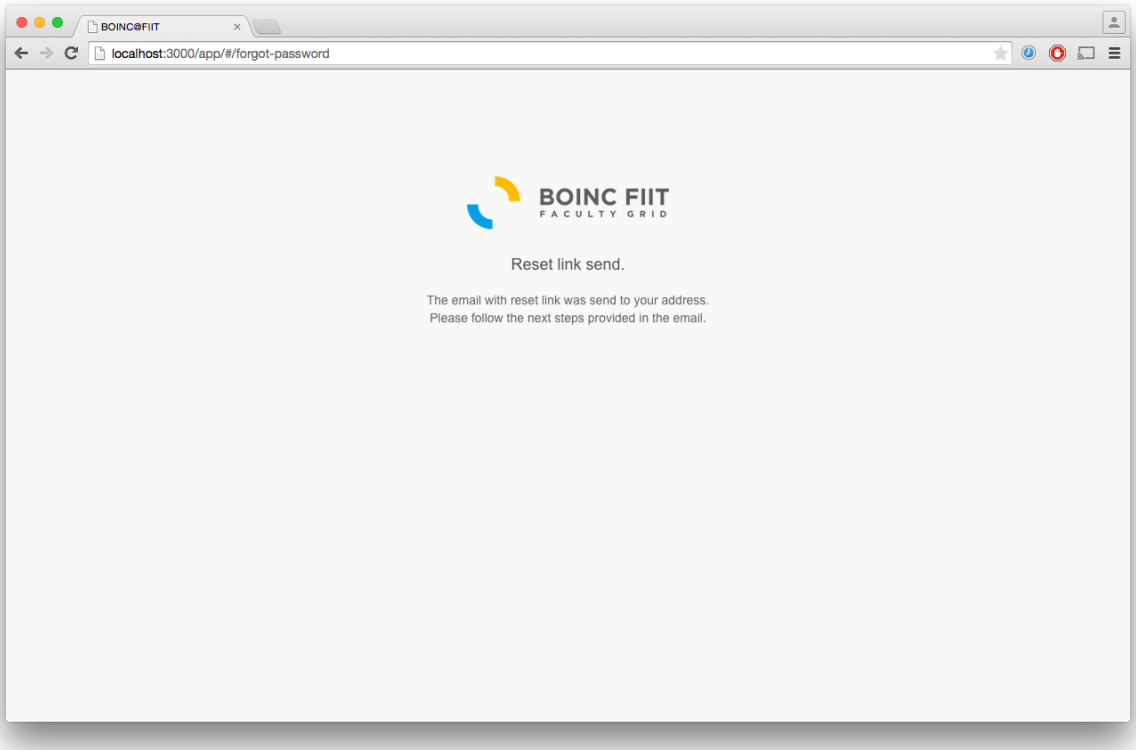
- Používateľ požiada o obnovu hesla:
  - používateľské rozhranie pre požiadanie o obnovenie hesla,
  - endpoint API, ktorý bude implementovať funkcionalitu generovania obnovovacích tokenov a zasielania prislúchajúcich mailov,
- Používateľ si obnoví heslo:
  - používateľské rozhranie pre obnovenie hesla,
  - endpoint API implementujúci logiku obnovovania hesla pomocou overeného obnovovacieho tokenu.

12.2.3 **Implementácia**

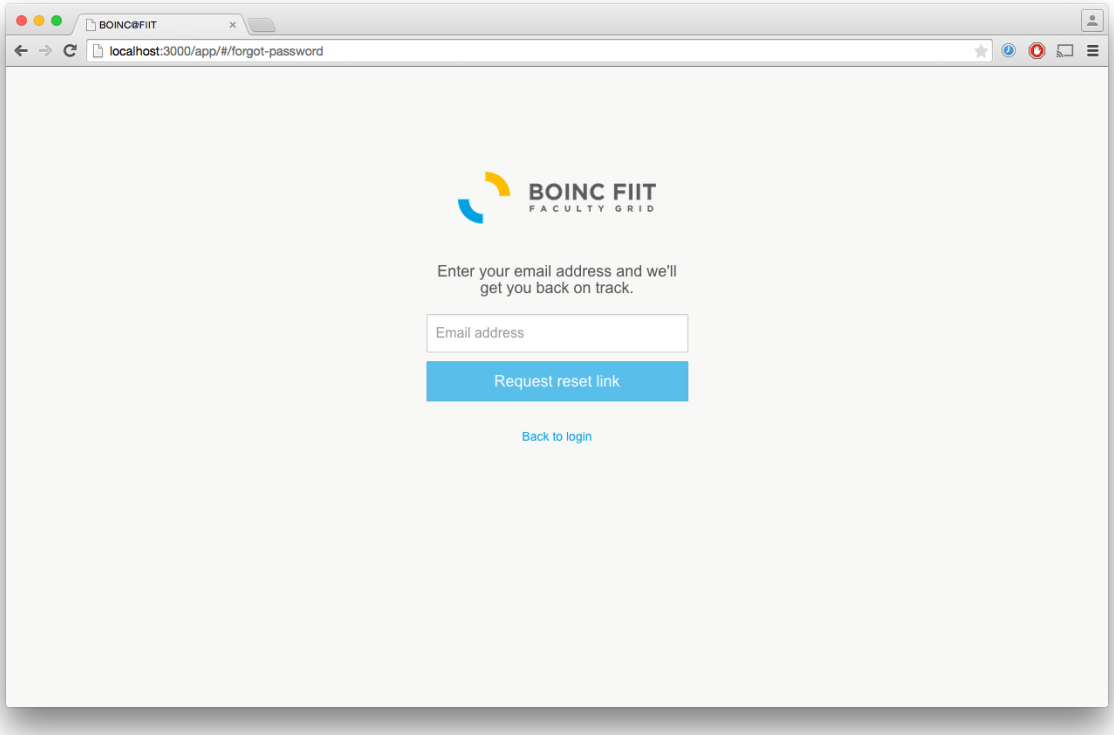
Pre implementáciu logiky ohľadom správy používateľských účtov sme požili gem *Devise*, tak ako vo zvyšných častiach systému. Rozšírili sme logiku poskytovanú defaultnym *Devise* controllerom starajúcim sa o resetovanie hesla a prispôsobili sme ho našim potrebám. V prvom rade odpovedáme len na JSON správy a ďalšou výzvou, ktorú sme museli vyriešiť bolo implementácia metódy, ktorá sa použila v mail templatoch na generovanie cesty pre obnovenie hesla, ktorej rozumie frontend, zároveň aj s validnými parametrami (token, ktorý sa do takejto cesty vygeneroval musel byť správny).

Na klienskej časti aplikácie boli implementované obrazovky, ako vidieť na obrázkoch nižšie. Na backend boli napojené volania, ktoré zažiadajú o zresetovanie hesla. Na obrazovke vytvorenia nového hesla musel byť použitý overovací token, ktorý bol vložený do cesty pre obnovenie hesla, ktorú používateľ obdŕžal v obnovovacom emaile.

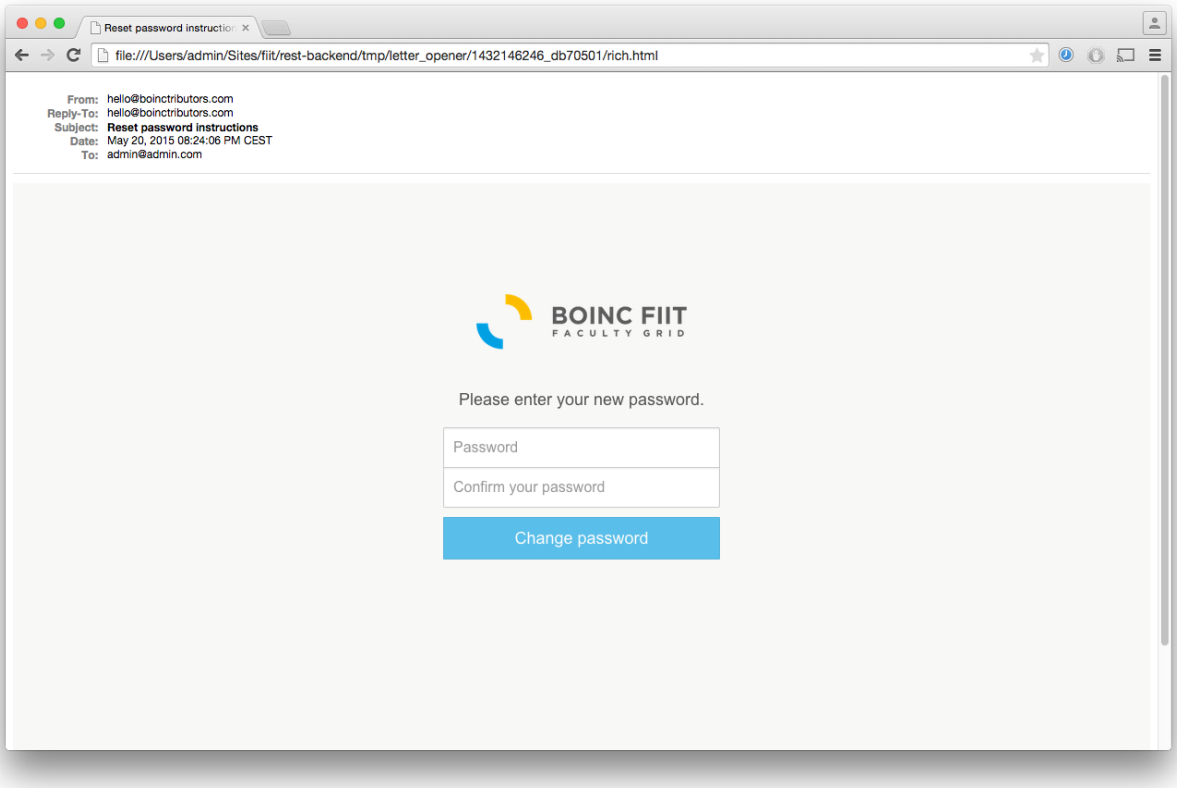
Pre posielanie emailov využívame v produkcii službu Mandrill a vo vývojovom prostredí imitujeme mailovú službu pomocou gemu *mailcatcher*, ktorá maily odoslané pomocou rails ActiveMailer modulov odchytí a zobrazí lokálne v prehliadači.



Obr. 19.: Náhľad obrazovky po vyžiadaní o obnovu hesla



Obr. 20.: Náhľad obrazovky pre obnovu hesla po prekliknutí z emailu.



Obr. 21.: Ukážka emailu počas vývoja obnovy hesla.

12.2.4 **Testovanie**

Pre otestovanie scenára obnovenia hesla boli vytvorené automatizované testy na strane backendu, ktoré overovali základnú funkcionalitu tejto časti systému. Na mockovanie posielania emailov sme použili štandardné nastavenie ActionMailer testovej konfigurácie kedy sa odoslané emaily ukladajú do fronty.

12.3 Úprava deployment scenára a CI

12.3.1 **Úloha**

Táto úloha súvisí s predošlou úlohou *Zjednotenie repozitárov*. V podstate neexistuje aktuálny proces automatizovaného nasadenia produkčnej verzie na server. V predchádzajúcich šprintoch sme už využívali automatický deployment scenár pri nasadzovaní hlavnej vývojovej vetvy.

12.3.2 **Implementácia**

Vypracovanie úlohy spočívalo v úprave existujúceho Capistrano skriptu pre deployment a pridaní rake task-u pre zostavenie klientskej SPA aplikácie v prostredí Ruby on Rails. Je dôležité

poznamenať, že pre optimalizáciu celého procesu sme museli upraviť ukladanie pomocných súborov npm a bower inštalácie tak, aby sa prenášali naprieč každým deployment procesom.

12.4 Konfigurácia SPA

12.4.1 **Úloha**

Táto úloha taktiež súvisí zo zjednotením repozitárov. Pri vytváraní buildu aplikácie, je potrebné previazať konfiguráciu klientskej časti so serverom (správne nastaviť adresu servera, ktorý ma dopytovať klient)

12.4.2 **Návrh**

Vytvoria sa dva konfiguračné súbory. Jeden bude slúžiť pri vývoji aplikácie a druhý pri vytváraní buildu aplikácie.

12.4.3 **Implementácia**

Bolo vytvorené východiskové nastavenie adresy servera. Do rake tasku, ktorý stavia aplikáciu, bola pridaná časť, ktorá vytvorí súbor *client/app/scripts/config.js* (tento súbor nie je súčasťou repozitára). Front-end potom číta nastavenie z tohto súboru (ak existuje).

12.4.4 **Testovanie**

Otestovanie riešenia bolo splnené testovacím nasadením na testovací server, kde prebehlo úspešne automatické nasadenie a spustenie pomocou CI služby.

12.5 Základné údaje o používateľovi

12.5.1 **Úloha**

Frontendová časť aplikácie nemá dostatok informácií o prihlásenom používateľovi, ako je napríklad jeho ID. Bez toho frontendový klient nevie vytvárať špecifické dopyty do API s daným id používateľa, resp. nemá informácie o práve prihlásenom používateľovi.

12.5.2 **Návrh**

Po úspešnom prihlásení do aplikácie (už implementované), vrátiť okrem kódu 200 aj JSON dáta s údajmi ako id, email, token. Takisto, nechceme používať autentifikáciu na základe session, preto je potrebné vrátiť identifikačný token. Autentifikáciu pomocou tokenu na strane servera by sme mali vyriešiť už nejakým existujúcim gemom.

12.5.3 Implementácia

Odosielanie základných informácií o používateľovi po prihlásení sme doplnili do príslušného controllera, jednalo sa len o vybranie správnych atribútov User objektu, ktoré sa budú prezentovať na frontend vo formáte JSON. Pri riešení autentifikácie pomocou tokenu sme najprv ošetrili zanedbanú prevenciu CSRF útokov, ako na strane frontendu, kde sme nastavovali príslušný CSRF token re každú požiadavku na API a aj na strane backendu, kde sme vytvorili filtre okolo každej akcie vykonanej naším API. Menovite before filter, ktorý kontroloval autenticitu tokenu každej požiadavky a after filter, ktorý nastavoval nový CSRF token pre klienta. Čo sa týka tokenovej autentifikácie, rozhodli sme sa požiť gem *simple_token_authentication*, ktorý pridáva do gemu *Devise* práve túto funkcionalitu.

12.6 Zhodnotenie šprintu

Bohužiaľ, musíme označiť tento šprint za nevydarený z hľadiska dodržiavania termínov, výstupov a celkovo kvality a efektivity práce. Morálka tímu, naopak, počas sviatkov stúpla, no návrat a pokračovanie na prácach mali pomalý spád. Spoločnou diskusiou sme sa zhodli, že pre potreby vývoja bude efektívnejšie prepojiť doteraz dva separátne repozitáre do jedného.

12.6.1 Retrospektíva šprintu

Zápis z retrospektívneho stretnutia po 5. šprinte

Dátum: 2.3.2015
Miesto stretnutia: Jobsovo softvérové štúdio
Čas: 12:00
Účastníci: Členovia tímu: Bc. Juraj Kochjar, Bc. Martin Kaššay, Bc. Matej Kloska, Bc. Patrik Gallik, Bc. Pavol Čurilla, Bc. Roman Roštár
Vypracoval: Bc. Martin Kaššay

Téma stretnutia: Zhodnotenie práce počas piateho šprintu.

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	úprava capistrano script	nepripravené chybové hlášky

Plánovanie	vytváranie podúloh	plánovanie šprintov ohodnocovanie úloh neodhadujú sa story pointy
Tím a stretnutia	morálka tímu zlepšenie efektivity a výstupov tímu v porovnaní z predošlým šprintom	pozastavenie práce na projekte počas sviatkov nedodržiavanie termínov
Programovanie	spojenie repozitárov codeship.io a capistrano implementovanie zabudnutého hesla	implementáciami, nekompatibilita veľmi málo sa programovalo

Zhodnotenie stretnutia:

V piatom šprinte sme poľavili. Prišli sviatky a práca na projekte sa zastavila. Po sviatkoch prišlo skúškové, kedy sa nám tiež moc do práce nechcelo. Z toho dôvodu sa tento šprint veľa vecí nestihlo dokončiť a boli presunuté do ďalšieho šprintu. Podarilo sa nám doladiť capistrano script na automatizované testovanie a nasadzovanie na náš virtuálny server v škole. Rozhodli sme sa po prvom semestri práce, že bude lepšie, ak repozitáre pre backend a frontend spojíme do jedného. Z implementačných úloh sa dokončovala agenda okolo profilu používateľa.

13 Šprint 6 – The Wild Geese

13.1 Návrhy a redesign klientskeho rozhrania pre vkladanie úloh

13.1.1 **Úloha**

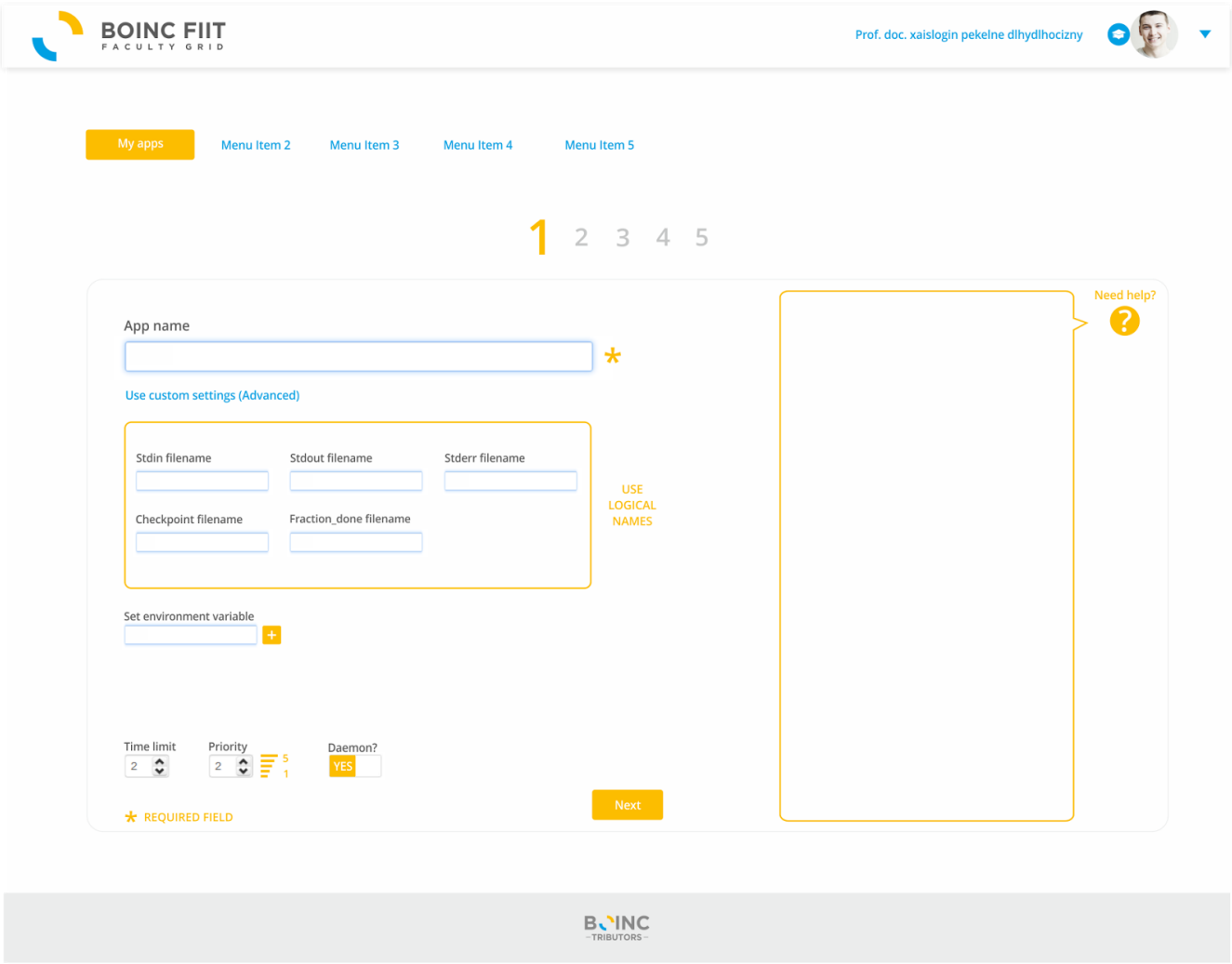
Jednou z hlavných našich predsavzatých vylepšení BOINC aplikácie bolo umožnenie vkladania nových úloh do aplikácie prostredníctvom zjednodušeného rozhrania.

13.1.2 **Návrh**

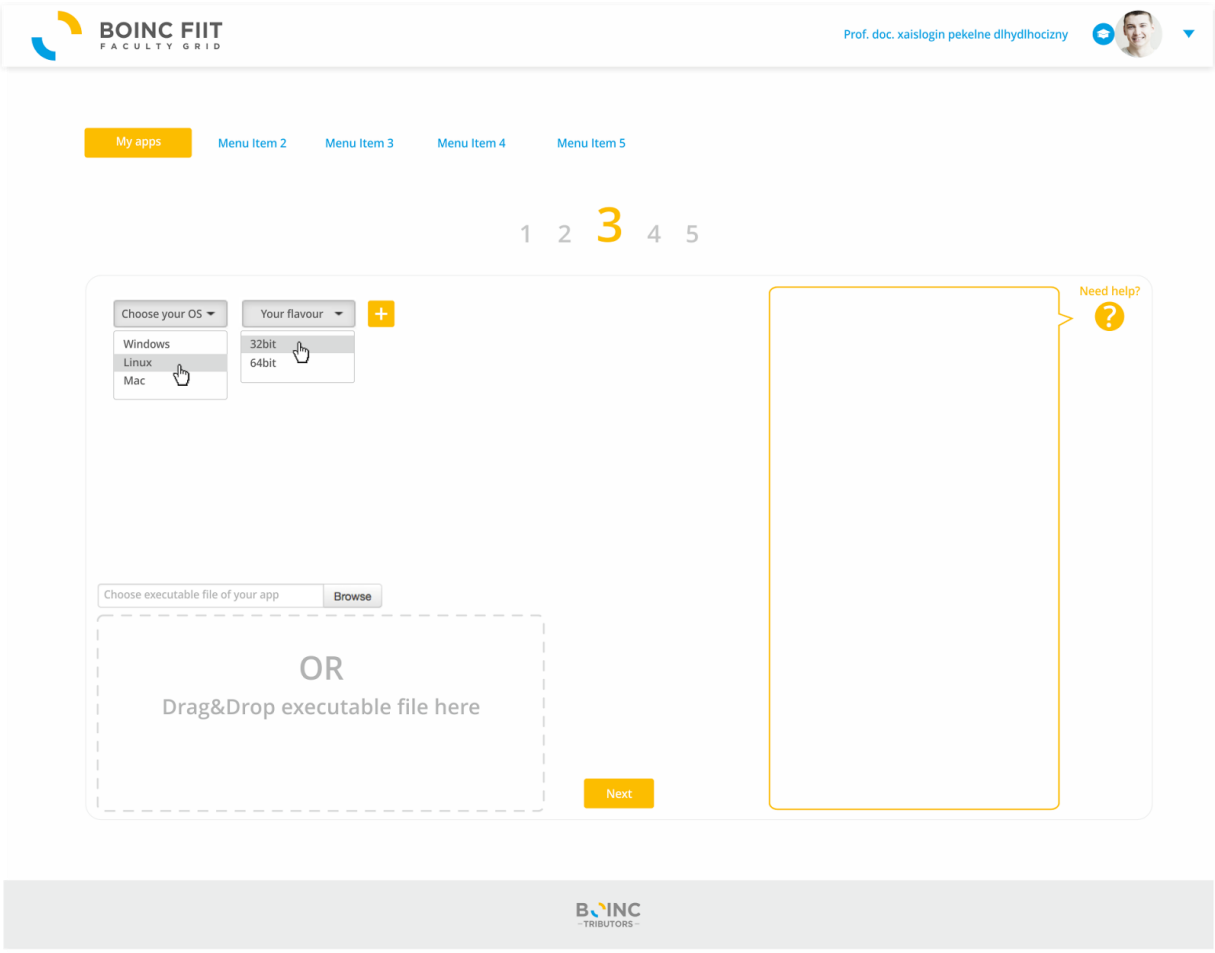
Po diskusii sme dospeli k názoru, že pridávanie aplikácie je príliš komplexné (obsahuje množstvo potrebných informácií a súborov). Preto sme sa rozhodli pridávanie rozdeliť do viacerých krokov. Po analýze databázy a použitým techník reverzného inžinierstva pri analýze BOINCU sme boli schopný vytvoriť špecifikáciu dát a súborov, ktoré môžeme vo formulári použiť.

13.1.3 **Implementácia**

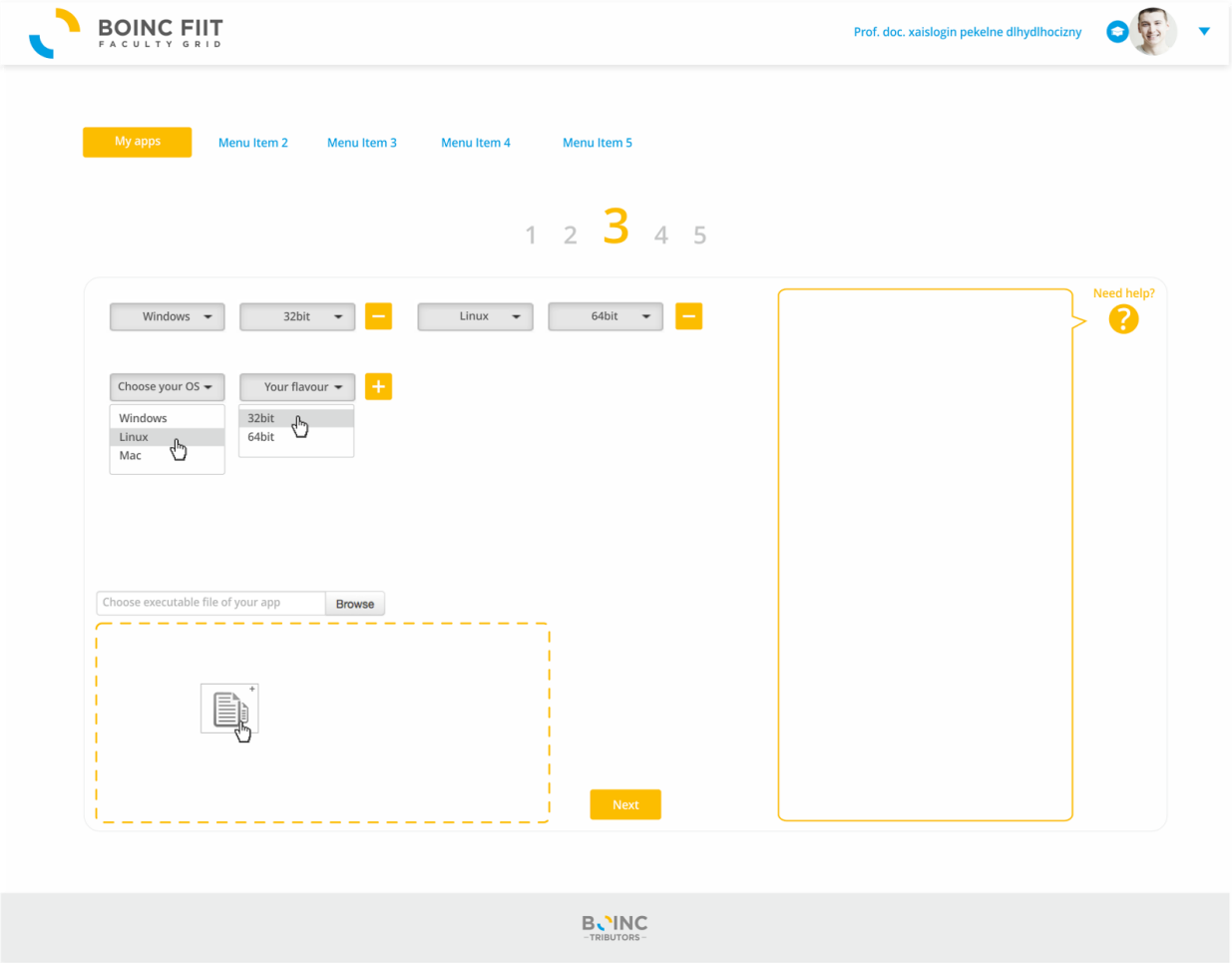
Keďže táto úloha sa týka iba návrhu, za implementáciu možno považovať samotné vytvorenie grafických návrhov:



Obr. 22.: Návrh vkladania aplikácie – krok 1.



Obr. 23.: Návrh vkladania aplikácie – krok 3a.



Obr. 24. Návrh vkladania aplikácie – 3b.

13.2 Informácie o user aktivite

13.2.1 Úloha

Zobraziť používateľovi po prihlásení uvítaciu stránku - dashboard zobrazujúci základný prehľad a štatistiky o práve počítaných aplikáciách v projekte.

13.2.2 Návrh

Rozhodli sme sa na túto stránku umiestniť grafy pre každú aplikáciu zvlášť, ktoré budú zobrazovať:

- mnou vypočítaný percentuálny podiel aplikácie,
- percentuálny podiel príspevkov ostatných používateľov,
- doposiaľ nevypočítaný percentuálny podiel (ak existuje).

13.2.3 Implementácia

Po rozkliknutí na tlačidlo pri danej aplikácii sa zobrazí bližší popis súvisiaci už len s vybranou aplikáciou. Práve tu môžeme nájsť informácie ako:

- dátum pridania,
- počet vygenerovaných workunitov,
- denné výpočtové štatistiky uvádzané v GFLOPS,
- denné štatistiky výsledkov aplikácie,
- grafy priebehu aplikácie.

Keďže výpočet štatistík o výsledkoch by pri veľkom počte inštancií výsledkov mohol trvať veľmi dlhú dobu a tým pádom sa to pre náš scenár použitia (prezentácia štatistických dát frontendu) stáva nevhodným, aby sme urýchlili načítanie štatistík, vytvorili sme rake task, ktorý predpočíta globálne štatistiky pre aplikácie za určené obdobie a tiež rake task predpočítavajúci denné štatistiky za určené obdobie. Takto získané dáta sa perzistujú do databázovej tabuľky. Tieto rake tasky sa budú spúšťať pomocou asynchrónnych jobov raz za určitú dobu, napr. raz za deň, pričom aktualizujú tabuľky predvypočítaných dát o dáta za nové obdobie.

Praktické ukážky sú zobrazené v šprinte č.9 v časti štatistiky a grafy.

13.3 Customer Development dokument

13.3.1 Úloha

Cieľom bolo vytvoriť dokument, ktorý bolo potrebné odovzdať vrámci súťaže TP CUP.

Originálne znenie zadania:

Každý tím v paneli vypracuje odpovede na otázky v dokumente Customer development, ktorý je tiež dostupný na dokumentovom serveri. Je to zjednodušená verzia tej skutočnej. Dôležité je, aby ste si urobili v tíme mozgovú búrku a odpovedali na tie otázky, ktoré sú pre vás relevantné. Teda ide o vyjadrenie vašich vlastných hypotéz o produkte, zákazníkoch... Ak na niečo nebudete vedieť odpovedať alebo sa ukáže, že ste to nepochopili správne - presne na to je panel (aj keď budeme časovo obmedzení, takže všetko určite nevyriešime).

13.3.2 Implementácia

V elektronickej prílohe tejto dokumentácie sa nachádza celý dokument, ktorý sme vypracovali. Vzhľadom na povahu projektu sme neboli schopní odpovedať na všetky otázky.

13.4 Úprava hlášok a chýb po prihlásení

13.4.1 **Úloha**

Proces registrácie, prihlasovania a obnovy hesla môže v mnohých prípadoch skončiť chybou. Úlohou je pokryť chybové stavy a správne oznámiť chybovou hláškou používateľovi daný stav.

13.4.2 **Návrh**

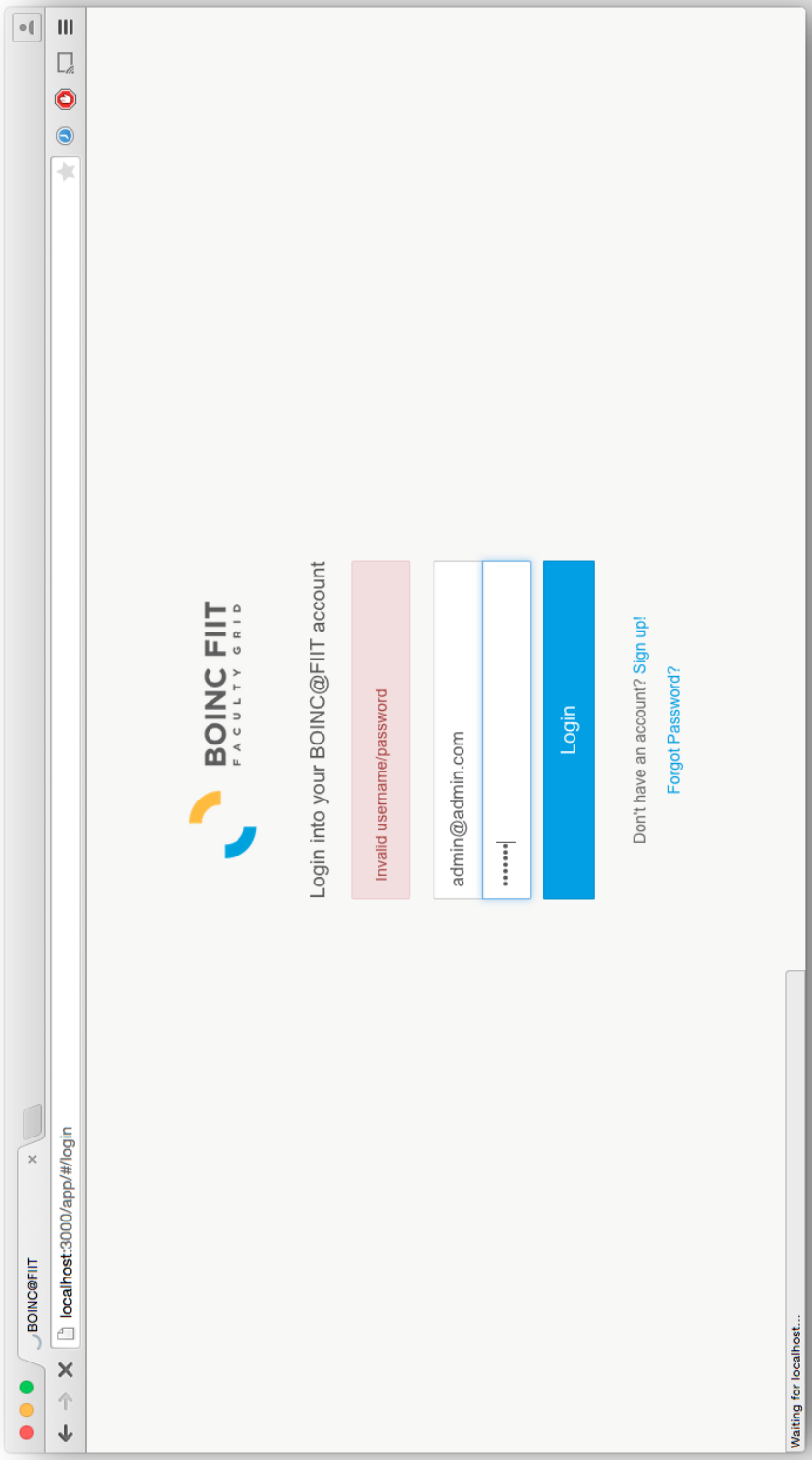
Chybové hlášky sa budú generovať na dvoch miestach:

- priamo v prehliadači - HTML5 validácia,
- na strane servera - Ruby On Rails Active Model validácia.

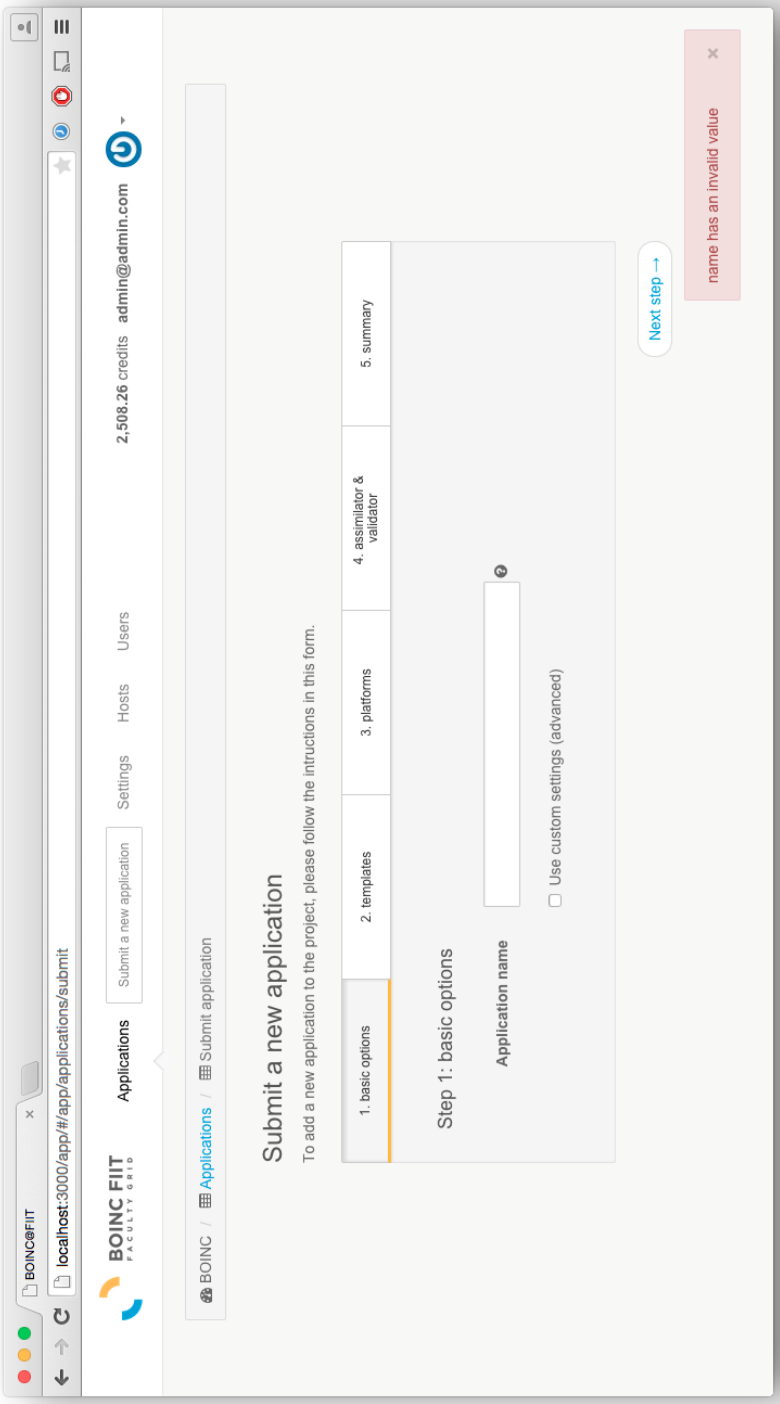
V prípade chyby na strane servera bude správa o chybe preposlaná pomocou správneho HTTP status kódu 4XX.

13.4.3 **Implementácia**

Na zobrazovanie chybových a úspešných hlášok na strane klienta sme použili open-source module s názvom Angular Growl. Inštalácia bola jednoduchá pomocou nástroja Bower, ktorý v projekte používame. Zobrazovanie hlášky potom znamenalo jednoduché zavolanie metódy tohto modulu s parametrami text hlášky a typ (chyba, úspech, info).



Obr. 25.: Ukážka chybovej hlášky pri prihlásení



Obr. 26.: Ukážka chybovej hlášky pomocou modulu Angular Growl (vpravo dole)

13.5 Vloženie aplikácie do nášho BOINC servera

13.5.1 **Úloha**

Úlohou je úspešne vložiť vzorovú aplikáciu do BOINC servera pomocou konzolového rozhrania za účelom zdokumentovania celého procesu vkladania.

13.5.2 **Implementácia**

Ako testovaciu aplikáciu sme použili aplikáciu, ktorá je pribalená v BOINCu medzi ukážkovými aplikáciami. Použili sme aplikáciu *worker*. Po dlhšom študovaní používateľskej príručky sa nám konečne podarilo nájsť vhodnú postupnosť krokov tak, že aplikácia sa nakoniec poslala na klientskú aplikáciu boincu, kde sa vypočítala a odoslala späť na validáciu.

13.6 Pridanie validácií BOINC nastavení

13.6.1 **Úloha**

Cieľom úlohy je pridať všeobecne platné validácie na vstupné dáta nastavení rôznych lokalít používania BOINC klientov.

13.6.2 **Návrh**

Nastavenia spolu s pravidlami je nutné prevziať priamo zo zdrojových kódov starého BOINC rozhrania, nakoľko neexistuje oficiálna dokumentácia, ktorá by pokrývala všetky nastavenia spolu s hraničnými hodnotami.

13.6.3 **Implementácia a testovanie**

Validácia prebieha na úrovni modulu *prefs*, ktorý v sebe zapuzdruje okrem iného logiku pre prácu s konfiguračným formátom BOINC XML. Validátor akceptuje generické typy ako napríklad:

- INT,
- BOOL,
- STR,
- ENUM.

Samotné pravidlá sú definované pomocou dvojíc VLASTNOST => TYP.

Validácia nastavení je pokrytá štandardnými RSpec testami.

13.7 Zhodnotenie šprintu

Po predošlom šprinte nastalo citeľné zlepšenie, no stále sme boli v sklze, i keď sa nám podarilo začať dokonca nové úlohy. Dokončovanie úloh z predošlého šprintu a konsolidácia nových úloh boli hlavnými aktivitami tohto šprintu. Podarilo sa nám zefektívniť a lepšie uchopiť prácu s nástrojom JIRA, dôsledkom čoho bolo aj zlepšenie výstupov v podobe grafov, alebo neustále narastajúci a zmenšujúci sa backlog úloh. Jednou z veľmi pozitívnych správ bolo začatie implementácie pridávania aplikácie podľa nami vypracovaných návrhov, keďže sme úspešne pridali vzorovú aplikáciu do platformy BOINC. Popri prácach na vývoji sme sa aktívne venovali aj tímovým povinnostiam v rámci súťaže TP CUP.

13.7.1 **Retrospektíva šprintu**

Zápis z retrospektívneho stretnutia po 6. šprinte

Dátum: 16.3.2015
Miesto stretnutia: Jobsovo softvérové štúdio
Čas: 12:00
Účastníci: Členovia tímu: Bc. Juraj Kochjar, Bc. Martin Kaššay, Bc. Matej Kloska, Bc. Patrik Gallik, Bc. Pavol Čurilla, Bc. Roman Roštár
Vypracoval: Bc. Martin Kaššay

Téma stretnutia: Zhodnotenie práce počas šiesteho šprintu.

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	vloženie vzorovej aplikácie do BOINC servera vypracovanie otázok na TP Cup	oneskorené návrhy
Plánovanie	vytváranie backlogu	naplánovanie veľkého množstva práce, ktorá sa nestíha neodhadujú sa story pointy
Tím a stretnutia	morálka tímu	pozastavenie práce na projekte počas sviatkov nedodržiavanie termínov

Programovanie	úprava modelov DB vylepšenia SPA validácie	väčšie množstvo chýb v kóde

Zhodnotenie stretnutia:

V šiestom šprinte sme sa museli naštartovať. Čakalo na nás pár dokumentov, ktoré bolo treba odovzdať na TP Cup. Vytvorili sme ďalšie návrhy, aby sme mohli ďalej programovať. Upravili sa chybové hlášky, presmerovania po prihlásení a rôzne ďalšie veci, ktoré bolo treba upraviť. Medzičasom vyšla nová aktualizácia BOINC serveru, a preto bolo potrebné upraviť DB modely. Pokračovali sme vo validáciách nastavení a nakoniec sme začali robiť prehľad vložených aplikácií.

14 Šprint 7 – Dalmore

14.1 Vkladanie úlohy

14.1.1 Úloha

Keďže dizajn na vkladanie a analýza sú hotové, potrebujeme implementovať samotné pridávanie úlohy do aplikácie.

14.1.2 Návrh

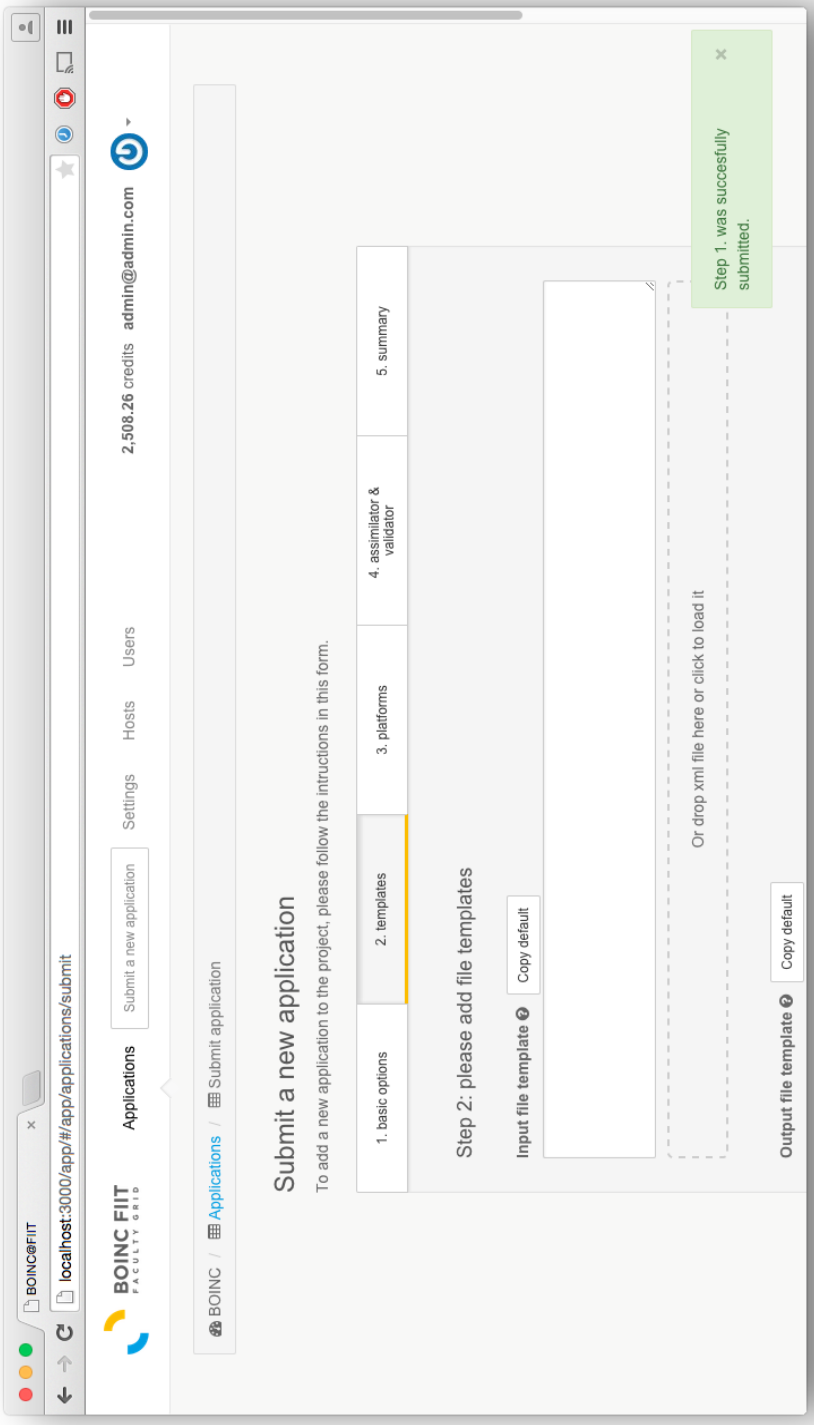
Dizajnový návrh je už pripravený. Technologicky, je nutné zabezpečiť nasledujúce veci:

- front-end rozhranie (formulár),
- back-end, ktorý bude obsluhovať odosielanie jednotlivých krokov formulára, ukladať dočasne informácie súbory vyplnené v týchto krokoch a udržiavať stav o pridávanej aplikácii v session,
- skript, ktorý pridá aplikáciu do BOINCU. Tento skript bude spustený serverovou časťou aplikácie (Rails), so špecifickými parametrami vyplývajúcimi z vyplneného formulára.

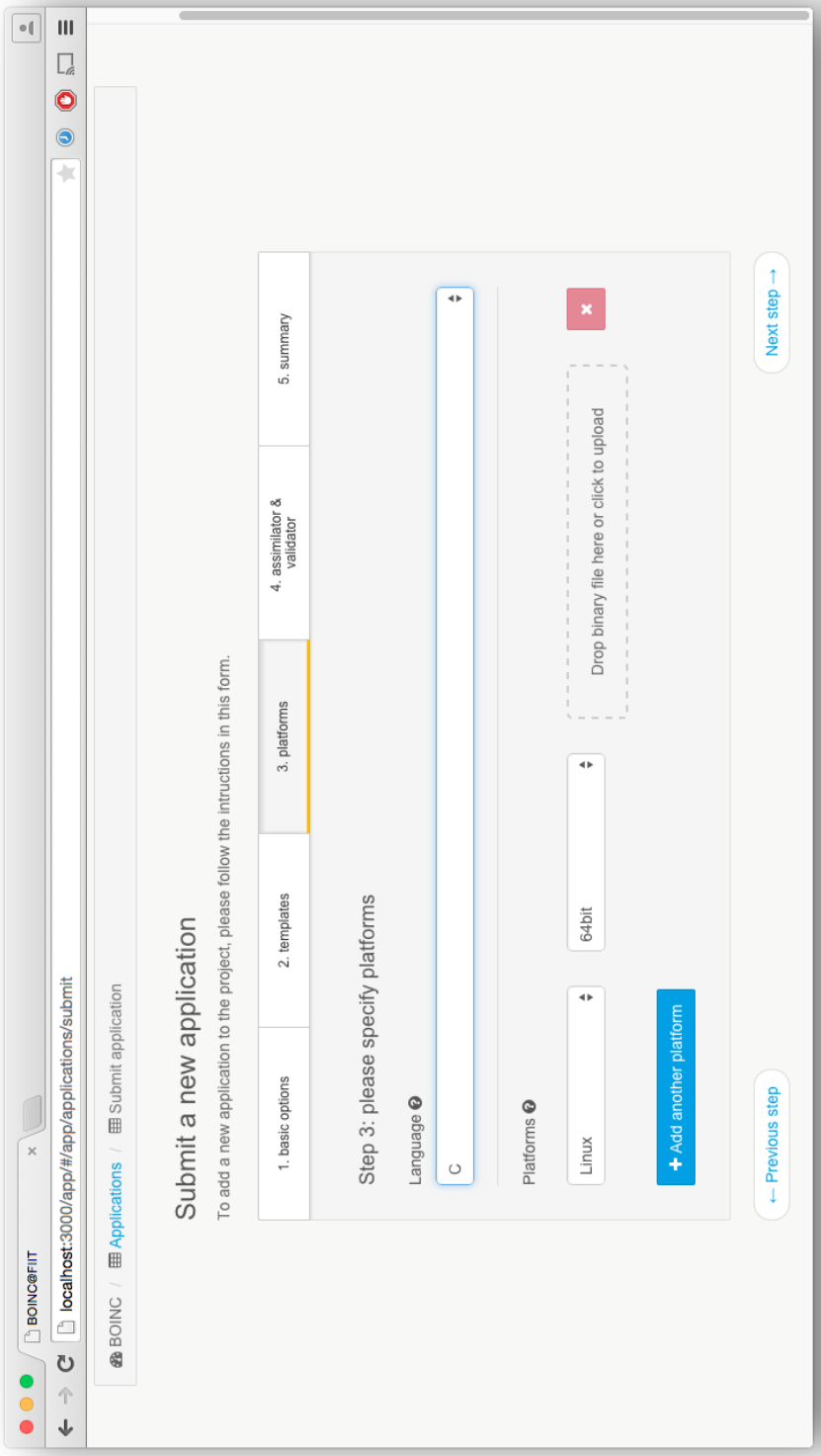
14.1.3 Implementácia

Aj keď naše API je vo väčšine REST, funkcionalitu takéhoto typu sme sa rozhodli vyriešiť pomocou ukladania dát do session, keďže sme chceli minimalizovať nepotrebnú záťaž na databázu, ktorá by vznikla tým, že by bolo nutné kontrolovať uložené záznamy o vkladaných aplikáciách a odstraňovať tie nedokončené. Aby sme vhodne od seba oddelili logiku vkladania aplikácie, kľúčové časti sme rozdelili do dvoch modulov, a to AppWizardStore, ktorý zabezpečuje ukladanie dočasných informácií o vkladanej aplikácii a modul AppWizard obsahujúci všetku logiku týkajúcu sa samotného vloženia aplikácie do BOINC projektu. Ukladanie aktuálneho stavu do session riešime pomocou vstavaného Ruby Marshall objektu, ktorý vie serializovať Ruby objekty, pričom pred každým dotazom na API zodpovedné za kroky vkladania aplikácie a najprv znovu vytvorí objekt AppWizardStore z údajov uložených v session. Ak takéto údaje nie su, vraciame sa na prvý krok a session sa obnovuje. Pri každom odoslaní kroku sa zároveň kontroluje správnosť zadaných údajov a vracajú sa chybové hlášky, ak niektoré z údajov boli chybné. Ak sa používateľ úspešne dostane do posledného kroku vkladania a potvrdí, resp. schváli vloženie takejto aplikácie, modul AppWizard vykoná všetky potrebné akcie, od naplnenia, či vytvorenia príslušných konfiguračných súborov pre BOINC aplikáciu a presunutia všetkých potrebných uploadnutých súborov do projektového BOINC adresára, až po samotné spustenie BOINC shell skriptov starajúcich sa o pridanie novej aplikácie do projektu.

Najväčšou výzvou na strane frontendu predstavoval upload viacerých súborov a zároveň json dát naraz.



Obr.27.: Krok pridávania aplikácie, vidieť pole na písanie XML.



Obr. 28.: Krok pridávania aplikácie, výber technológie.

14.1.4 Testovanie

Odoslanie všetkých 5 krokov formulára sme na back-ende pokryli testami. Testy kontrolujú, či funguje validácia na textové polia, xml reťazce, a či back-end správne vytvára dočasne súbory.

14.2 Overiť použitie JAVA na BOINC

14.2.1 Úloha

Analyzovať možnosti pre tvorbu a následné pridávanie aplikácií programovaných v programovacom jazyku JAVA. V prípade úspešného vloženia aplikácie pomocou príkazového riadku spísať všetky kroky potrebné pre úspešné vloženie.

14.2.2 Implementácia

BOINC poskytuje možnosť pridať JAVA aplikáciu pomocou jej kompilácie do štandardného .exe súboru pomocou nástroja Jsmooth. Dôležité je pribaliť k výslednej aplikácii špeciálny typ JAVA runtime environment, ktorý bol modifikovaný BOINC tímom pre účely optimalizácie výpočtov. Samotné vloženie následne musí v konfiguračnom XML obsahovať okrem referencii na kompilovanú aplikáciu taktiež referenciu na spomínaná JRE.

14.3 Zhodnotenie šprintu

Celý siedmy šprint sme strávili pri implementovaní, testovaní a ladení najdôležitejšej funkcie nášho rozhrania - pri procese vkladania výpočtových úloh. Overovali sme a skúmali sme zároveň spôsoby, akými bude možné pridávať do platformy BOINC aplikácie napísané v jazyku Java. Túto úlohu sa nám podarilo vyriešiť, no stále sme zápasili s dlhmi minulých šprintov, i keď pokrok na nových úlohách bol badateľný a odrazilo sa to pozitívne aj na morálke tímu. Komunikácia, odhad a včasné dodanie ostávajú stále našimi slabinami.

14.3.1 Retrospektíva šprintu

Zápis z retrospektívneho stretnutia po 7. šprinte

Dátum:	30.3.2015		
Miesto stretnutia:	Jobsovo softvérové štúdio		
Čas:	12:00		
Účastníci:	Členovia tímu:	Bc. Juraj Kochjar	Bc. Martin Kaššay
		Bc. Matej Kloska	Bc. Patrik Gallik
		Bc. Pavol Čurilla	Bc. Roman Roštár

Vypracoval: Bc. Martin Kaššay

Téma stretnutia: Zhodnotenie práce počas siedmeho šprintu.

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	návrh vloženia aplikácie pomocou nášho rozhrania	neuvedomili sme si všetky nástrahy a chyby, ktoré môžu vzniknúť
Plánovanie	vytváranie podúloh	neodhadujú sa story pointy
Tím a stretnutia	morálka tímu	nedodržiavanie termínov problém s komunikáciou v tíme
Programovanie	overenie Javy práca na vkladaní úlohy	nedokončenie úloh, ktoré sme mali spraviť

Zhodnotenie stretnutia:

V siedmom šprinte sme začali s implementáciou procesu vkladania úloh na server BOINC. Prvotný kód, ktorý sme napísali pre SPA, bol vytvorený rýchlo, avšak REST trval oveľa dlhšie ako sme očakávali. Pomimo toho sme ešte skúsili vložiť nejakú aplikáciu, ktorú sme vytvorili v programovacom jazyku Java, aby sme overili správnosť vkladania aplikácií napísaných v tomto programovacom jazyku. Následne sme si uvedomili, že vkladanie úloh, hoci je značne zjednodušené, stále je príliš zložité pre človeka, ktorý uvidí BOINC prvýkrát. Preto sme sa rozhodli spraviť nápovede, ktoré sme však podcenili, a preto sme ich nestihli dokončiť tento šprint.

15 Šprint 8 – Glenturret

15.1 Upraviť dizajn krokov pri vkladaní aplikacie

15.1.1 **Úloha**

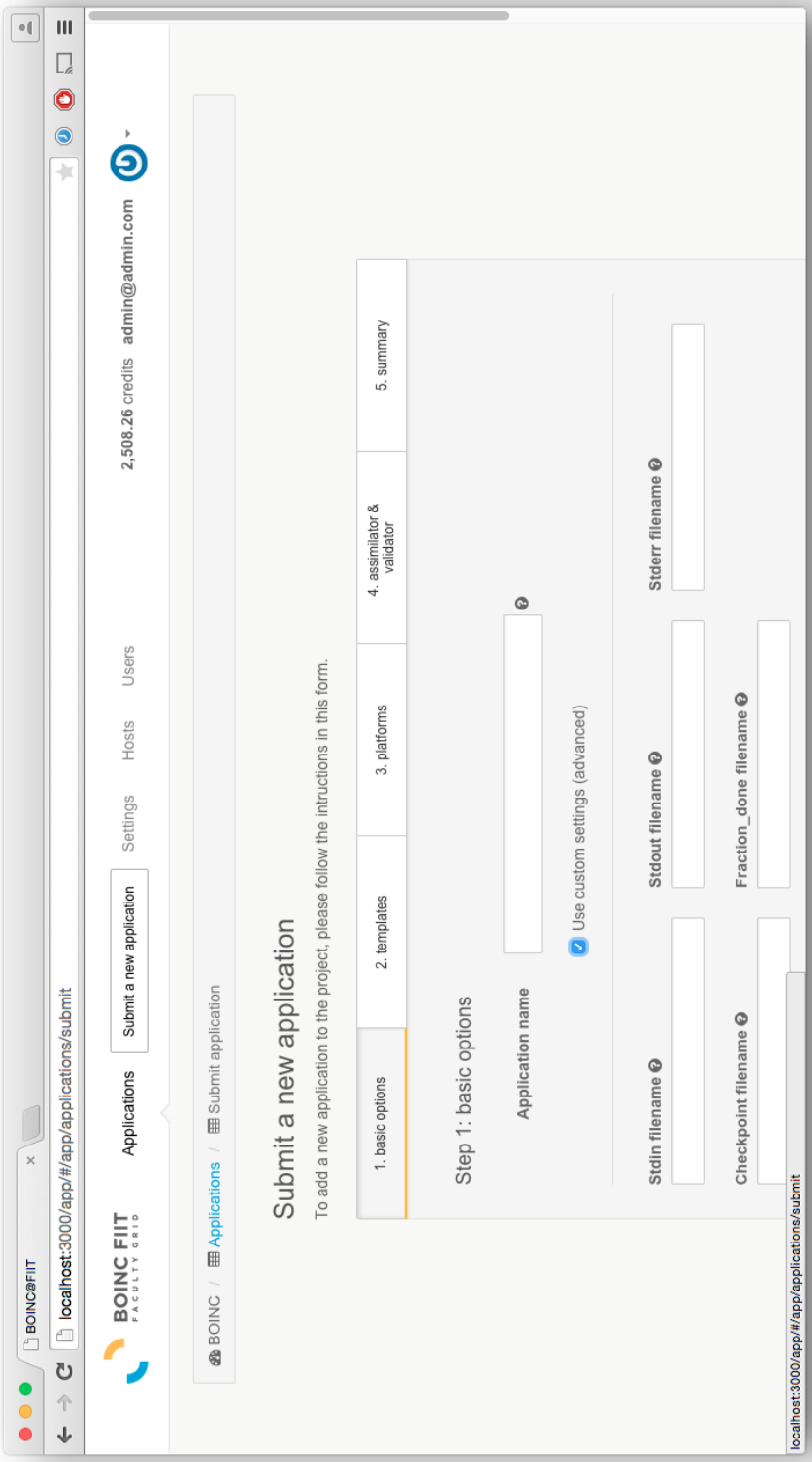
Aktuálny implementovaný dizajn k pridávaniu krokov do aplikácie sa nám nepozdával, preto sme sa rozhodli o zmenu.

15.1.2 **Návrh**

Vedľa čísel krokov sme sa rozhodli zobraziť aj názov kroku. To uľahčí používateľovi orientáciu v aktuálnom progrese a lepšie odhadnúť, koľko a akých zložitých krokov mu ostáva. Takisto sú tieto názvy použité pri konečnom kontrolnom sumáre všetkých údajov poskytnutých používateľom.

15.1.3 **Implementácia**

Pri implementácií nebolo nutné vykonať žiadne úpravy na backende. Implementácia zahŕňala len zmenu html šablón klientskej časti aplikácie a css úpravy.



Obr. 29.: Formulár na pridanie aplikácie s novým designom krokov.

15.1.4 Testovanie

Keďže ide len o UX zmenu, testovanie neprebehlo. Môžeme len subjektívne posúdiť, že formulár sa používa lepšie.

15.2 Nasadenie aplikácie na server

15.2.1 Úloha

Pokúsiť sa nasadiť vzorovú aplikáciu pomocou nového webového rozhrania. Zaznamenať všetky potenciálne problémy a vytvoriť report s úlohami, ktoré nastali počas používania.

15.2.2 Testovanie

Na testovanie sme použili vzorovú x64 Linux aplikáciu s dátovým súborom obsahujúcim náhodný reťazec o veľkosti 1MB. Za úspešné testovanie považujeme taký stav, kde klientske aplikácie vypočítajú nadefinované pracovné úlohy a server ich po overení všetky označí za správne.

15.3 Definovanie JSON pre aplikácie

15.3.1 Úloha

Cieľom je vytvoriť špecifikáciu JSON správ, ktoré sa posielajú medzi REST a SPA aplikáciou.

15.3.2 Implementácia

Chýbajúce JSON formáty správ boli dodefinované a zapísané vo wiki nášho GIT repozitára:

- zoznam aplikácií,
- zoznam pracovných úloh,
- štatistiky pre detail aplikácie.

15.4 Oprava serializácie dát pri vkladaní aplikácie

15.4.1 Úloha

Pri serializácii dát do session štandardným Ruby marshallerom dochádza k zvláštnym chybám v encodingu dát. Je nutné vymyslieť spôsob , ktorým budeme serializovať dáta o vkladanej aplikácii v module AppWizardStore.

15.4.2 **Návrh**

Použijeme na serializáciu formát JSON, ktorý by mohol byť pre naše potreby postačujúci.

15.4.3 **Implementácia**

Do triedu AppWizardStore sme pridali metódy pre serializáciu triedy AppWizardStore do formátu JSON a vytvorenie nového objektu tejto triedy z takto serializovaných JSON dát. Následne pri každej požiadavke na niektorý z krokov vkladania aplikácie znovu vytvoríme inštanciu AppWizardStore z JSON dát uložených v session a dáta pri každom úspešnom potvrdení kroku vkladania serializujeme do session.

15.4.4 **Testovanie**

Keďže väčšina funkcionality krokov vkladania aplikácie je pokrytá integračnými controller testami, nebolo nutné inak túto zmenu otestovať.

15.5 Zhodnotenie šprintu

Hlavnou náplňou ôsmeho šprintu boli opravy chýb vo vkladaní úloh. Podarilo sa nám pomocou nášho rozhrania vložiť aplikáciu do BOINC servera, ešte je však v ďalšom šprinte nutné otestovať, či takto zadaná úloha naozaj rozosiela pracovné jednotky prihláseným BOINC klientom. Negatívnym bodom tohto šprintu bol nečakaný odchod člena tímu, čo zvýšilo množstvo práce, ktoré sme museli za šprint zvládnuť na ostatných členov.

15.5.1 **Retrospektíva šprintu**

Zápis z retrospektívneho stretnutia po 8. šprinte

Dátum:	13.4.2015		
Miesto stretnutia:	Jobsovo softvérové štúdio		
Čas:	12:00		
Účastníci:	Členovia tímu:	Bc. Juraj Kochjar	Bc. Martin Kaššay
		Bc. Matej Kloska	Bc. Patrik Gallik
		Bc. Roman Roštár	
Vypracoval:	Bc. Martin Kaššay		

Téma stretnutia: Zhodnotenie práce počas ôsmeho šprintu.

Výstup zo stretnutia:

Okruh	+	-
Analýza a návrh	nasadenie aplikácie na server pomocou nášho rozhrania definovanie JSON	neuvedomili sme si všetky nástrahy a chyby, ktoré môžu vzniknúť
Plánovanie	vytváranie podúloh	naplánované príliš veľké množstvo úloh neodhadujú sa story pointy
Tím a stretnutia	skype call	odstúpenie člena tímu nedodržiavanie termínov nepracovanie počas voľna
Programovanie	opraviť kroky vkladania vytvorenie seedu	nedokončenie úloh, ktoré sme mali spraviť zoznam aplikácií ostal prázdny

Zhodnotenie stretnutia:

V ôsmom šprinte sme mali veľa práce, lebo čas TP Cupu sa blíži. Nasadili sme vkladanie aplikácie na server, avšak došlo k viacerým komplikáciám, ktoré sme nečakali. Preto bolo potrebné, aby sme začali s úpravami tak, aby sme to stihli. Taktiež sme si boli vedomí, že nemáme žiadne dáta, ktorými by sme mohli prezentovať náš projekt, preto bolo potrebné vytvoriť seed, na ktorom by sa dalo pekne ukázať fungovanie našej aplikácie. Nakoniec sme všetky ciele, ktoré sme si dali na tento šprint nesplnili, a teda spísanie nápovedí sa znova presúva do ďalšieho šprintu. K tomu všetkému sme sa museli vysporiadať s faktom, že je nás odteraz o jedného menej.

16 Šprint 9 – Midleton

16.1 Vylepšiť prázdny zoznam aplikácií

16.1.1 **Úloha**

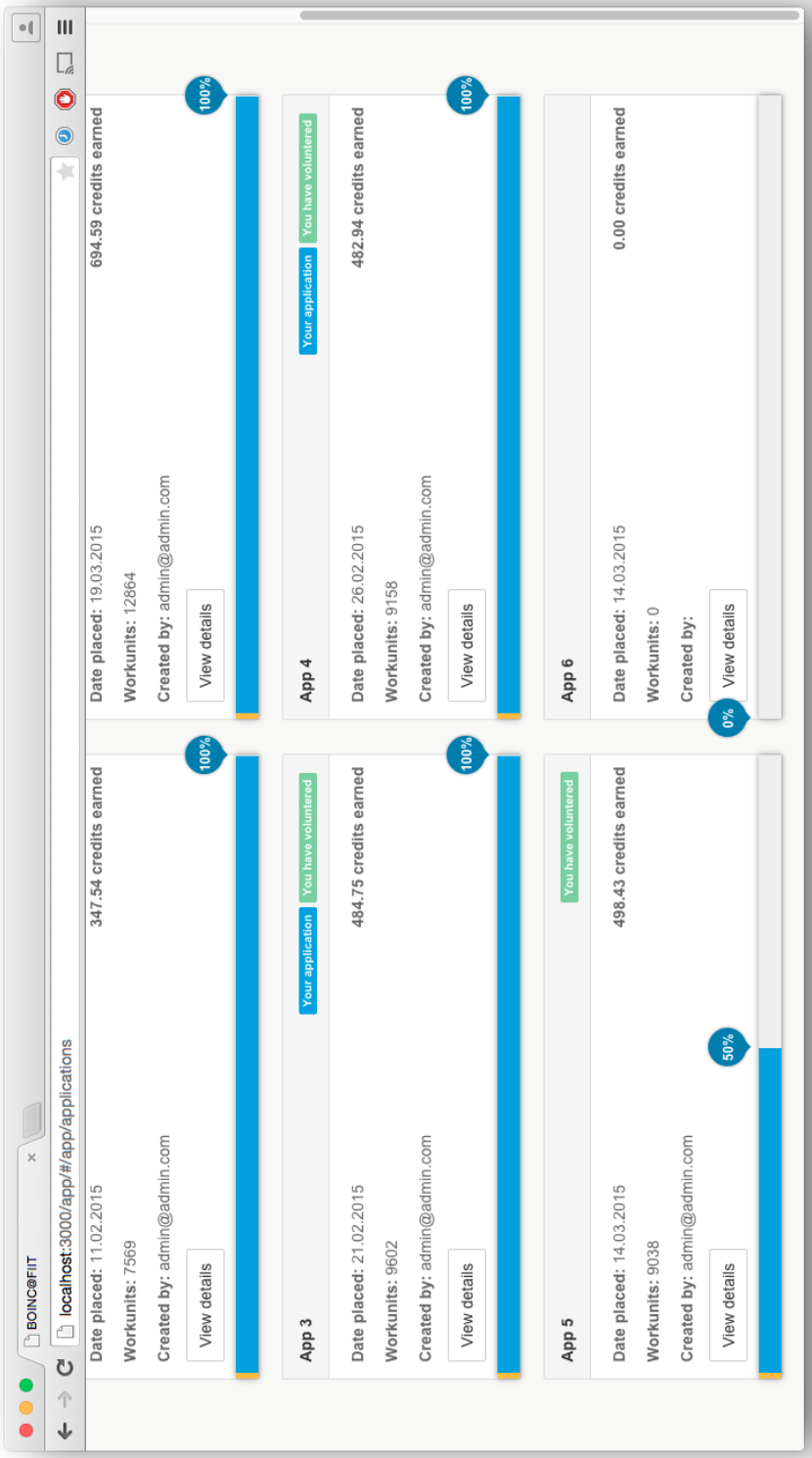
Na úvodnej obrazovke aplikácie (obrazovka Aplikácie) je zobrazený zoznam aplikácií, ktoré pridal používateľ. Uvedomili sme si ale, že tento zoznam bude väčšinou prázdny, keďže väčšina používateľov úlohy len počíta.

16.1.2 **Návrh**

Namiesto toho, aby na úvodnej obrazovke bol zobrazený zoznam aplikácií pridaných daným používateľom, rozhodli sme sa zobraziť všetky, ktoré sa nachádzajú vo @FIIT projekte. S tým, že aplikácie ktoré pridal používateľ, budú vizuálne odlíšené od tých, ktoré pridal niekto iný, a zároveň budú zvýraznené tie, na ktorých používateľ pracoval. Vytvorili sme grafický návrh bez wireframov, ten tu ale neuvádzame, lebo je prakticky totožný z implementáciou.

16.1.3 **Implementácia**

Bolo upravené API na serveri, ktoré vracia zoznam aplikácií. Pre každú aplikáciu v zozname bolo potrebné pridať atribút *owner,* ktorý nadobúda hodnotu true/false podľa toho, či je prihlásený používateľ vlastník danej aplikácie, a atribút *volunteer,* ktorý nadobúda hodnotu true/false podľa toho, či daný používateľ rátal danú aplikáciu.



Obr. 30.: Zoznam aplikácií.

16.1.4 Testovanie

Do GUI pribudol mierne zložitejší ukazovať aktuálneho priebehu, ten sme otestovali vo viacerých prehliadačoch (Chrome, Safari, Firefox).

16.2 Reverse proxy

16.2.1 Úloha

Vytvoriť generický konfiguračný súbor pre server nginx tak, aby bolo možné prevádzkovať v produkcii naraz Apache2 server pre projektovú komunikáciu a webové rozhranie za jednotnou IP adresou.

16.2.2 Implementácia

Konfiguračný súbor predpokladá beh Apache servera na porte 8088 a webového rozhrania na porte 8000. Samotný prepis sa deje na základe regulárnych výrazov, ktoré vyhodnocujú smerovanie na správny logický lokálny server fyzického servera. V prípade projektovej komunikácie je nevyhnutné nasmerovať adresy začínajúce na cgi-bin na apache a adresy začínajúce na client resp. data na nové webové rozhranie.

16.3 Štatistiky a grafy

16.3.1 Úloha

Aby sme zväčšili prínos aplikácie, už od začiatku projektu sme plánovali do aplikácie pridať grafy a štatistiky, ktoré vizuálne znázornia informácie:

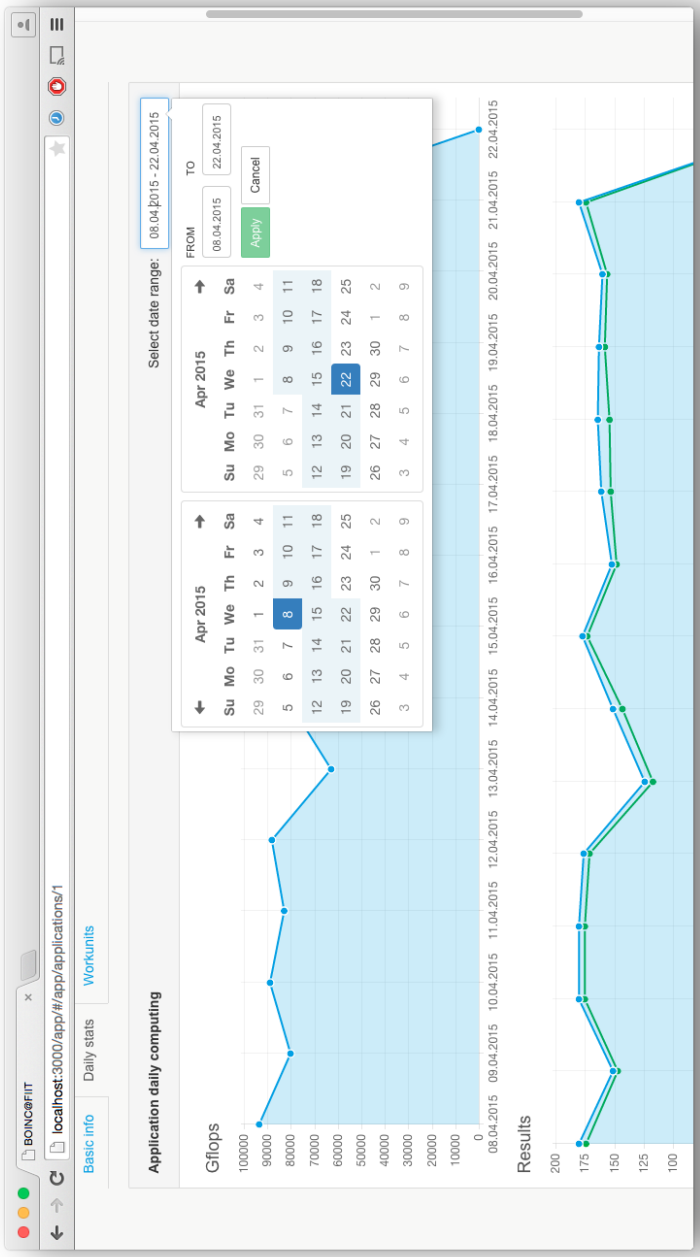
- úspešné/neúspešné výsledky v čase,
- počet vyrátaných Gflopov v čase,
- zoznamy hostov, z ktorých daný používateľ rátal,
- zoznam workunitov, ktoré požívateľ počítal.

16.3.2 Návrh

Na implementáciu pomerne zložitej funkcie ako grafy na frontende sme sa rozhodli použiť nejakú open-source knižnicu, najlepšie takú ktorá pokryje naše potreby.

16.3.3 **Implementácia**

Na implementáciu grafov na klientskej časti sme použili open-source knižnicu Chart.js. Doplnili sme element, ktorý umožňuje používateľovi vybrať rozpätie dátumov, pomocou knižnice Angular UI



Obr. 31: Grafy v detaile aplikácie.

Na implementáciu tabuľkových štatistík sme použili modul ui-grid, ktorý je súčasťou knižnice Angular UI. Tento modul generuje tabuľky, ktoré možné zoraďovať podľa stĺpcov.

Obr. 32.: Zoznam hostov, na ktorých počítal klient.

16.3.4 **Testovanie**

Otestovali sme grafy a tabuľky v rôznych prehliadačoch, aby sme sa uistili, že funkcionalita funguje. Zistili sme drobné chyby pri responzívnosti grafov (deklarované knižnicou že je podporované), kedy pri mobilných zariadeniach grafy trčia von z obrazovky.

16.4 Integrácia vkladania úlohy

16.4.1 **Úloha**

Je nutné manuálne overiť, či proces vkladania aplikácie, tak ako sme ho implementovali, naozaj umožní používateľovi vložiť novú aplikáciu do BOINC projektu.

16.4.2 **Návrh**

Overenie prebehne vložením skúšobnej aplikácie a netriviálnej aplikácie do BOINC servera pomocou nami implementovaného rozhrania.

16.5 LDAP login

16.5.1 **Úloha**

Pridať možnosť prihlásenia pomocou AIS prihlasovacích údajov.

16.5.2 **Návrh**

Knižnica Devise, ktorú používame na registráciu a štandardné prihlásenie poskytuje možnosť prihlásenia pomocou LDAP - túto možnosť chceme využiť vzhľadom na jednotnosť so štandardným procesom prihlasovania.

16.5.3 **Implementácia**

Implementácia spočíva v konfigurácii pomocou súboru ldap.yml v adresári config a rozšírenia aplikácie o vlastný spôsob vyhodnocovania priority pomocou čoho sa bude vyhľadávať používateľ. V našom prípade sú priority:

- lokálna databáza,
- AIS LDAP.

Testovanie prebieha pomocou RSpec testov, ktoré sú dodávané spolu s knižnicou Devise.

16.6 Chyby vo vkladaní úloh

16.6.1 **Úloha**

Vzhľadom na komplexitu úlohy vkladania novej aplikácie sa pri jej implementácii objavilo niekoľko ďalších netriviálnych chýb, ktoré bolo nutné vyriešiť. Konkrétne išlo tieto chyby:

- názvy súborov prezentované na frontend po ich uploade nesmú ukazovať celú cestu do dočasného priečinka na serveri,
- nesprávne nastavovanie kroku na ktorom používateľ opustil vkladanie novej aplikácie, resp. znovuobnovenie kroku na ktorom používateľ skončil zo session.

16.6.2 **Implementácia**

Vyššie spomenuté chyby sme úspešne odstránili. V triede AppWizardStore sme vytvorili metódu, ktorá dáta uložené v session vráti pre frontend v rozumnej podobe, teda všetky cesty k súborom sú osekané na meno súboru apod. Nastavovanie kroku pri refreshnutí sme vyriešili na strane frontendu, je nutné však poznamenať, že pri znovu načítaní stránky sa klientovi lokálne stratia už uploadnuté súbory, hoci na serveri už ich uložil (tato situácia nastane len ak sa vráti na niektorý predchádzajúci krok a refreshne prehliadač).

16.7 Zhodnotenie šprintu

Môžeme zhodnotiť, že deviaty šprint bol najťažším šprintom doteraz. Z implementačnej stránky sme síce zvládli spraviť pomerne štandardný počet úloh, kritickým však bola nepredpokladaná chyba, ktorá nastávala pri používaní rozhrania pre vkladanie úlohy. Jednalo sa o chyby a nekorektné programovacie rutiny v zdrojových kódoch samotnej platformy BOINC a vzhľadom na takmer nulové chybové výpisy, z ktorých by sme mohli zistiť príčinu nás stálo obrovskú námahu sa nakoniec dopátrať ku zdroju chyby. Tieto chyby sa nám však nakoniec podarilo odhaliť a opraviť. Chyby objavné v platforme BOINC reportneme ako issue v issue trackeri boincu, prípadne prispejeme pull requestom, ktorý tieto chyby opravuje.

16.7.1 **Retrospektíva šprintu**

Zápis z retrospektívneho stretnutia po 9. šprinte

Dátum:	27.4.2015
Miesto stretnutia:	Jobsovo softvérové štúdio
Čas:	12:00
Účastníci:	Členovia tímu: Bc. Juraj Kochjar, Bc. Martin Kaššay, Bc. Matej Kloska, Bc. Patrik Gallik

Bc. Roman Roštár

Vypracoval: Bc. Martin Kaššay

Téma stretnutia: Zhodnotenie práce počas deviateho šprintu.

Výstup zo stretnutia:

Okruh	+	-
Plánovanie	rozumné plánovanie článok na Robime.it	neodhadujú sa story pointy
Tím a stretnutia	stretnutia aj mimo povinných stretnutí programovanie v tíme dobrá pracovná atmosféra	
Programovanie	LDAP prihlásenie, reverse proxy vytvorenie príručky	

Zhodnotenie stretnutia:

V deviatom šprinte sme mali veľa práce. Bolo sa treba pripraviť na prezentovanie nášho projektu na TP Cup. lebo čas TP Cupu sa blíži.

17 Príloha A: Inštalačná príručka

Celá inštalačná príručka predpokladá serverové prostredie operačného systému Debian resp. akýkoľvek operačný systém z rodiny Linux. Inštalácia nového webového rozhrania pozostáva z nasledujúcich základných krokov:

- inštalácia podporných softvérových prostriedkov,
- inštalácia serverovej časti BOINC platformy,
- vytvorenie nového projektu,
- inštalácia samotného webového rozhrania.

Podporné softvérové prostriedky

Okrem samotného operačné systému budeme potrebovať nainštalovať ďalšiu softvérovú výbavu potrebnú pre správny chod celého servera. Zoznam balíčkov, ktoré je potrebné nainštalovať (názvy balíčkov sa líšia pre každú verziu OS, uvádzame preto všeobecné názvy):

- make,
- git,
- pkg-config, dh-autoreconf,
- m4,
- apache2,
- php5, apache2-mod-php5
- mysql-server, mysql-server-dev
- openssl, libssl-dev
- curl,
- python, python-mysqldb
- rvm, ruby
- nodejs, npm, grunt, bower.

Inštalácia BOINC

Inštalácia serverovej časti BOINC pozostáva z naklonovania repozitára najnovšej verzie:

```
$ cd ~
$ git clone git://boinc.berkeley.edu/boinc-v2.git boinc-src
```

Aktualizácie zdrojových kódov

```
$ cd ~/boinc-src
$ git pull
```

Konfigurácia kompilácie

```
$ ./_autosetup
```

```
$ ./configure --disable-client --disable-manager
```

Kompilácie softvéru

```
$ make
```

Inštalácia

```
$ make install
```

Vytvorenie nového projektu

Zmena pracovného adresára (vzhľadom na adresár, kde sa nachádzajú zdrojové súbory inštalácie)

```
$ cd ~/boinc-src-tools
```

Spustenie procesu vygenerovania nového projektu.

```
$ ./make_project --db_name=NAZOV --db_user=POUZIVATEL --db_passwd=HESLO
PROJEKT
```

Pre pokračovanie v generovaní je potrebné potvrdiť kontrolnú otázku.

Príklad výstupu po generovaní:

```
Creating project 'prirucka' (short name 'prirucka'):
  PROJECT_ROOT = /home/team7/projects/prirucka/
      URL_BASE = http://team07-14.ucebne.fiit.stuba.sk/
 HTML_USER_URL = http://team07-14.ucebne.fiit.stuba.sk/prirucka/
  HTML_OPS_URL = http://team07-14.ucebne.fiit.stuba.sk/prirucka_ops/
       KEY_DIR = /home/team7/projects/prirucka/keys/
       DB_NAME = prirucka
       DB_HOST = localhost
Continue? [Y/n]  Y
Creating directories
Generating encryption keys
Copying files
Setting up database
```

```
Writing config files
Linking CGI programs
update_translations finished
Done installing default daemons.

Done creating project.  Please view
/home/team7/projects/pririucka/pririucka.readme
for important additional instructions.
```

Pridanie URL adresy projektu do Apache:

```
$ a2enmod mod_cgi cgi
$ echo ~/projects/pririucka/pririucka.httpd.conf >> /etc/apache2/httpd.conf
$ service apache2 restart
```

Inštalácia webového rozhrania

Inštalácia webového rozhrania pozostáva z naklonovania repozitára najnovšej verzie:

```
$ cd ~
$ git clone git@bitbucket.org:tp71415/rest-backend.git web-interface
```

Nastavenie konfiguračných súborov:

```
~/web-interface/config/database.yml (podľa database.yml.tpl)
~/web-interface/config/settings/production.yml (nastaviť cestu to vygenerovaného projektu)
```

Inštalácia Ruby on Rails závislostí:

```
$ cd ~/web-interface
$ bundle install
```

Migrácia DB schémy

```
$ rake db:migrate
```

Spustenie aplikácie

```
$ rake client:start (tu sa automaticky vygeneruje a nainštaluje SPA časť)
```

Alternatívny spôsob zostavenia SPA aplikácie

```
$ cd ~/web-interface/client
$ npm install
$ bower install
$ grunt build
$ cd ..
$ rake client:inject
```

18 Príloha B: Akceptačné testy

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií, Ilkovičova 3, 826 03 Bratislava

Webové rozhranie pre distribuované výpočt

Dokumentácia k inžinierskemu dielu – Akceptačné testy

Tímový projekt

Členovia tímu:
Bc. Pavol Čurilla
Bc. Patrik Gallik
Bc. Martin Kaššay
Bc. Matej Kloska
Bc. Juraj Kochjar
Bc. Roman Roštár

Vedúci tímu:
Ing. Peter Lacko, PhD.

V Bratislave Apríl 2015

1. Webové sídlo

Nasledujúca tabuľka znázorňuje možné scenáre použitia pri orientácii na webovom sídle projektu BOINC@FIIT – landing page (domovskej stránke).

Umiestnenie	Komponent	Očakávaná akcia	Uskutočnená akcia
Webové sídlo	Tlačidlo „log in“	Presmerovanie mimo webové sídlo na prihlasovaciu obrazovku	
Webové sídlo	Tlačidlo „sign up“	Presmerovanie mimo webové sídlo na registračnú obrazovku	
Webové sídlo	Tlačidlo „find out more“ - modré	Presmerovanie v rámci sídla na doplnkové informácie	
Webové sídlo	Tlačidlo „find out more“ – žlté	Presmerovanie v rámci sídla na doplnkové informácie	
Webové sídlo	Tlačidlo „check it out“	Presmerovanie v rámci sídla na doplnkové informácie	
Webové sídlo	Textový odkaz „FAQ“	Presmerovanie v rámci sekciu časté otázky	

2. Webové rozhranie

Nasledujúce tabuľky slúžia ako prehľad možných akcií vykonávaných v prostredí nami vytvoreného rozhrania.

2.1. Prihlásenie

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie - login	Vstupné pole s placeholderom „email address“	Vloženie valídneho prihlasovacieho údaju	Vzor v tvare: [:alnum:]+’@’[\w]+’.’[\w]+	
Webové rozhranie - login	Vstupné pole s placeholderom „password“	Vloženie valídneho prihlasovacieho údaju	Vzor v tvare: [:alnum:]{8,}	

Webové rozhranie - login	Tlačidlo „login“	Prihlásenie do systému[24]	-	
Webové rozhranie - login	Textový odkaz „sign up!“	Presmerovanie v rámci rozhrania na registračnú obrazovku	-	
Webové rozhranie - login	Textový odkaz „forgot password?“	Presmerovanie v rámci rozhrania na obrazovku zmeny hesla	-	

2.2. Manipulácia s rozhraním a nastavenia

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie	Tlačidlo „applications“	Presmerovanie v rámci rozhrania na domovskú stránku zobrazujúcu aktívne aplikácie	-	
Webové rozhranie	Tlačidlo „submit new application“[25]	Presmerovanie v rámci rozhrania na obrazovku pridávania novej aplikácie[26]	-	
Webové rozhranie	Tlačidlo „settings“	Presmerovanie v rámci rozhrania na nastavenia BOINC preferencí počítania	-	
Webové rozhranie	Tlačidlo dropdown „↓“	Presmerovanie v rámci rozhrania na nastavenia používateľa	-	
Webové rozhranie	Select box „show only“	Výber požadovanej hodnoty	{all, volunteered by me, submitted by me}	

[24] Vloženie nevalídnych prihlasovacích údajov spôsobí chybu a prihlásenie bude neúspešné
[25] Tlačidlo je skryté v prípade kliknutia na tlačidlo „settings“
[26] Umožnené iba používateľom s AIS prihlásením

Webové rozhranie	Tlačidlo „view details“	Zobrazenie detailov a štatistík o aplikácii	-	

Po úspešnom prihlásení bude používateľ presmerovaný na domovskú stránku, kde budú zobrazené aktívne aplikácie projektu BOINC@FIIT. V našom rozhraní rozlišujeme dva typy nastavení: nastavenia používateľa, nastavenia BOINC preferencií počítania.

Nastavenia používateľa:

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – nastavenia používateľa : karta „account settings“	Vstupné pole s placeholderom „email“	Vloženie validného prihlasovacieho údaju	Vzor v tvare: [:alnum:]+’@’[\w]+’.’[\w]+	
Webové rozhranie – nastavenia používateľa : karta „account settings“	Vstupné pole s placeholderom „name“	Vloženie validného prihlasovacieho údaju	Vzor v tvare: [:alnum:]*	
Webové rozhranie – nastavenia používateľa : karta „account settings“	Tlačidlo „save changes“	Uloženie a zmena vložených údajov	-	
Webové rozhranie – nastavenia používateľa : karta „password change“	Vstupné pole s placeholderom „current password“	Vloženie validného prihlasovacieho údaju	Vzor v tvare: [:alnum:]{8,}	
Webové rozhranie – nastavenia používateľa : karta „password change“	Vstupné pole s placeholderom „new password“	Vloženie validného prihlasovacieho údaju	Vzor v tvare: [:alnum:]{8,}	
Webové rozhranie – nastavenia používateľa : karta „password change“	Vstupné pole s placeholderom „retype password“	Vloženie validného prihlasovacieho údaju	Vzor v tvare: [:alnum:]{8,}	
Webové rozhranie – nastavenia používateľa : karta „password change“	Tlačidlo „save changes“	Uloženie a zmena vložených údajov	-	

Nastavenia BOINC preferencií počítania:

Preferencie procesoru:

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „use at most % of cpus“	Uloženie požadovaného nastavenia	[:digit:]{1,3}	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „use at most % of cpu time“	Uloženie požadovaného nastavenia	[:digit:]{1,3}	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Checkbox „Suspend work while computer is on battery power?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Checkbox „Suspend work while computer is in use?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Checkbox „Suspend GPU work while computer is in use?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „'In use' means mouse/keyboard input in last“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „Suspend work if no mouse / keyboard activity in last“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „Suspend when non-BOINC CPU usage is above“	Uloženie požadovaného nastavenia	0 < [:digit:] < 100	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor	Select box „do work only between the hours	Uloženie požadovaného nastavenia	(0 < [:digit:] < 24){2}	

settings“	of“			
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Checkbox „leave tasks in memory while suspended?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „switch between tasks every“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Vstupné pole „Request tasks to checkpoint at most every“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „processor settings“	Tlačidlo „save settings“	Uloženie a zmena vložených údajov	-	

Preferencie diskov a pamätí:

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Disk: use at most“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Disk: leave free at least“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Disk: use at most“	Uloženie požadovaného nastavenia	0 < [:digit:] < 100	

Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Tasks checkpoint to disk at most every“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Swap space: use at most“	Uloženie požadovaného nastavenia	0 < [:digit:] < 100	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Memory: when computer is in use, use at most“	Uloženie požadovaného nastavenia	0 < [:digit:] < 100	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Vstupné pole „Memory: when computer is not in use, use at most“	Uloženie požadovaného nastavenia	0 < [:digit:] < 100	
Webové rozhranie – nastavenia BOINC preferencií : karta „disk and memory settings“	Tlačidlo „save changes“	Uloženie a zmena vložených údajov	-	

Preferencie siete:

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „maintain enough tasks to keep busy for at least“	Uloženie požadovaného nastavenia	(0 < [:digit:] < 10){1,2}	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „... and up to additional“	Uloženie požadovaného nastavenia	(0 < [:digit:] < 10){1,}	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Checkbox „confirm before connecting to internet?“	Uloženie požadovaného nastavenia	boolean	

Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Checkbox „disconnect when done?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „maximum download rate“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „maximum upload rate“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Select box „use network only between the hours of“	Uloženie požadovaného nastavenia	(0 < [:digit:] < 24){2}	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „transfer at most“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Vstupné pole „... every“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Checkbox „skip image file verification?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – nastavenia BOINC preferencií : karta „network settings“	Tlačidlo „save settings“	Uloženie a zmena vložených údajov	-	

2.3. Vkladanie úlohy

Nasledujúce tabuľky zobrazujú možné spôsoby ako sa orientovať v sekcii vloženie novej aplikácie – *submit new application*. Sú rozdelené podľa logického členenia postupu pri ivkladaní úlohy.

1. Basic options

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „application name“[27]	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Checkbox „use custom settings (advanced)“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „stdin filename“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „stdout filename“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „stderr filename“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „checkpoint filename“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „fraction_done filename“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Tlačidlo „add another variable“	Zobrazenie ďalších komponentov	-	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „env_name“[28]	Uloženie požadovaného nastavenia	[\w]-{special characters}{,8}	

[27] Je zobrazované stále, ostatné komponenty nižšie iba po kliknutí na checkbox „*use custom settings (advanced)*“
[28] Zobrazené iba v prípade ak bolo stlačené tlačidlo „*add another variable*“. Počet zobrazení komponentu závisí od počtu stlačenia tlačidla „*add another variable*“

Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „env_value“[28]	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Tlačidlo „X“[28]	Vymazanie riadku s hodnotami	-	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „provide command line arguments“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Vstupné pole „time limit(s)“	Uloženie požadovaného nastavenia	[:digit:]+	
Webové rozhranie – vkladanie úlohy : krok 1	Selectbox „priority“	Uloženie požadovaného nastavenia	{1,2,3,4,5}	
Webové rozhranie – vkladanie úlohy : krok 1	Switch „daemon?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – vkladanie úlohy : krok 1	Tlačidlo „next step →“	Pokračovanie do kroku 2	-	

2. Templates

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – vkladanie úlohy : krok 2	Vstupná oblasť „input file template“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 2	Tlačidlo „copy default“	Nakopírovanie základných možností	-	
Webové rozhranie – vkladanie úlohy : krok 2	Drag & Drop + upload oblasť „or drop xml file here or click to load it“	Uloženie požadovaného súboru	-	

Webové rozhranie – vkladanie úlohy : krok 2	Vstupná oblasť „output file template“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 2	Tlačidlo „copy default“	Nakopírovanie základných možností	-	
Webové rozhranie – vkladanie úlohy : krok 2	Drag & Drop + upload oblasť „or drop xml file here or click to load it“	Uloženie požadovaného súboru	-	
Webové rozhranie – vkladanie úlohy : krok 2	Drag & Drop + upload oblasť „drop file here or click to upload“	Uloženie požadovaného súboru	-	
Webové rozhranie – vkladanie úlohy : krok 2	Tlačidlo „←previous step“	Návrat do kroku 1	-	
Webové rozhranie – vkladanie úlohy : krok 2	Tlačidlo „next step →“	Pokračovanie do kroku 3	-	

3. Platforms

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – vkladanie úlohy : krok 3	Select box „language“	Uloženie požadovaného nastavenia	{C, Java}	
Webové rozhranie – vkladanie úlohy : krok 3	Selectbox „platforms (1)“[29]	Uloženie požadovaného nastavenia	{Windows, Linux, MAC OS X}	
Webové rozhranie – vkladanie úlohy : krok 3	Selectbox „platforms (2)“[29]	Uloženie požadovaného nastavenia	{32bit, 64bit}	

[29] Počet zobrazení závisí od počtu kliknutia tlačidla „*add another platform*“, maximálne však 6x

Webové rozhranie – vkladanie úlohy : krok 3	Drag & Drop + upload oblasť „drop binary file here or click to upload“	Uloženie požadovaného súboru	-	
Webové rozhranie – vkladanie úlohy : krok 3	Tlačidlo „X“	Vymazanie riadku s hodnotami	-	
Webové rozhranie – vkladanie úlohy : krok 3	Tlačidlo „add another platform“	Zobrazenie ďalších komponentov	-	
Webové rozhranie – vkladanie úlohy : krok 3	Tlačidlo „←previous step“	Návrat do kroku 2	-	
Webové rozhranie – vkladanie úlohy : krok 3	Tlačidlo „next step →“	Pokračovanie do kroku 4	-	

4. Assimilator & validator

Umiestnenie	Komponent	Očakávaná akcia	Očakávaná hodnota	Uskutočnená akcia
Webové rozhranie – vkladanie úlohy : krok 4	Switch „use custom assimilator?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – vkladanie úlohy : krok 4	Drag & Drop + upload oblasť „drop binary file here or click to upload“[30]	Uloženie požadovaného súboru	-	
Webové rozhranie – vkladanie úlohy : krok 4	Vstupné pole „provide a command to run your assimilator“[30]	Uloženie požadovaného nastavenia	[:alnum:]+	

[30] Zobrazené iba v prípade, keď je switch „*use custom assimilator*“ zvolený na „*yes*“

Webové rozhranie – vkladanie úlohy : krok 4	Switch „use custom validator?“	Uloženie požadovaného nastavenia	boolean	
Webové rozhranie – vkladanie úlohy : krok 4	Drag & Drop + upload oblasť „drop binary file here or click to upload“	Uloženie požadovaného súboru	-	
Webové rozhranie – vkladanie úlohy : krok 4	Vstupné pole „provide a command to run your validator“	Uloženie požadovaného nastavenia	[:alnum:]+	
Webové rozhranie – vkladanie úlohy : krok 4	Tlačidlo „←previous step“	Návrat do kroku 3	-	
Webové rozhranie – vkladanie úlohy : krok 4	Tlačidlo „next step →“	Pokračovanie do kroku 5	-	

5. Summary

Umiestnenie	Komponent	Očakávaná akcia	Uskutočnená akcia
Webové rozhranie – vkladanie úlohy : krok 5	Karta „basic options“	Zobrazenie uložených hodnôt	
Webové rozhranie – vkladanie úlohy : krok 5	Karta „templates“	Zobrazenie uložených hodnôt	
Webové rozhranie – vkladanie úlohy : krok 5	Karta „platforms“	Zobrazenie uložených hodnôt	
Webové rozhranie – vkladanie úlohy : krok 5	Karta „platforms“	Zobrazenie uložených hodnôt	

Webové rozhranie – vkladanie úlohy : krok 5	Tlačidlo „← previous step“	Návrat do kroku 4	
Webové rozhranie – vkladanie úlohy : krok 5	Tlačidlo „next step →“	Vloženie aplikácie	

19 Príloha C: Elektronické médium

Elektronické médium, ktoré je súčasťou tejto dokumentácie má nasledujúci obsah:

- zdrojové súbory webového rozhrania, site
- generovaná dokumentácia Ruby on Rails, doc/rails
- generovaná dokumentácia AngularJS, doc/angular
- webové sídlo projektu team_website
- publikácie publications
- reporty reports