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

Autor	Zhrnutie zmien	Verzia	Dátum
Matej Kloska	Vytvorenie dokumentu, definovanie štýlov a formátu, vytvorenie štruktúry (kapitol)	1.0	06.11
Roman Roštár a Patrik Gallik	Doplnenie úvodu a cieľov pre zimný semester	1.1	08.11
Celý tím	Prvý šprint: Ballantine’s	2.0	10.11
Celý tím	Druhý šprint: Tullamore Dew	3.0	13.11
Celý tím	Tretí šprint: Jameson	4.0	17.11
Celý tím	Kontrola a doplnenie chýbajúcich častí	4.1	18.11
Juraj Kochjar	Finalizácia dokumentu	4.2	19.11

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), použíateľské 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

2 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
  - štatististiky
    - 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 Šprint 1 - Ballantine's

3.1 BOINC Inštalácia

3.1.1 Úloha

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

3.1.2 Implementácia

Inštalácia BOINC servera sa v našom prípade odvíjala od operačného systému, ktorý je v našom prípade Ubuntu Server 14.04. Oficiálna dokumentácia inštálacie[1] 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,
- 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.

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

3.2 BOINC Analýza

3.2.1 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.

3.2.2 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 prerekvizitou pre participovanie na akomkoľvek distribuovanom výpočtovom projekte vytvorenom v rámci fakultnej inštalácie BOINC@FIIT je mať nainštalovanú a spustenú klientskú aplikáciu platformy BOINC, voľne dostupnú na oficiálnych stránkach BOINC[2]. Logická architektúra distribuovanej výpočtovej platformy BOINC je nasledovná:

- Projekt (1)
    - Aplikácia (1..i)
        - Úloha - *job* (1..j)
            - Pracovná jednotka - *Workunit* (1..k)
            - Výsledok - *Result* (1..k)

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.

3.3 Základná kostra SPA aplikácie

3.3.1 Úloha

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

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

3.3.2 Návrh

Rozhodli sme sa, že aplikácia bude postavená na rámci AngularJS[3]. Tento rámec je momentálne pre tento účel najpoužívanejší a má veľku komunitu, preto nebude problém s podporou a ani s prípadným ďalším vývojom iným tímom. Pre rýchle prototypovanie nových obrazoviek použijeme rámec Bootstrap[4]. Na automatizáciu úloh použijeme jeden z nástrojov Grunt/Gulp.

3.3.3 Implementácia

Na inicializáciu projektu sme použili generátor nástroja Yeoman[5], konkrétne generátor generator-angular[6]. 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[7], 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[8].

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

3.4 Základná kostra REST aplikácie

3.4.1 Úloha

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

3.4.2 Návrh

Aplikácia bude postavená na rámci RoR[9], 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.

---

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

3.4.3 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, 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.

3.5 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.

3.5.1 Retrospektíva šprintu

Negatívum	**Pozitívum**	**Okruh záujmu**
• nedostatočné načasovanie postupov • stále veľmi nejasné a abstraktné, čo vlastne budeme robiť • nejasné, ako automatizovať procesy • nedostatočná analýza	• grafické návrhy rozhraní aplikácie	**Analýza a návrh**
	• inštalácia a konfigurácia nutných prostredí a samotnej BOINC platformy • relatívne rýchle vyriešenie podporných systémov, virtuálnych operačných systémov a	**Konfigurácia**

	nutných prerekvizít	
• časový sklz		**Plánovanie**
• 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	**Tím a stretnutia**
• očakávaná vyššia produktivita a výstupy tímu • veľmi veľa administratívy brzdí pokrok na projekte • málo času venovaného programovaniu a práci na projekte	• spoločné programovanie	**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.

4 Šprint 2: Tullamore Dew

4.1 Údržba existujúceho systému BOINC@FIIT

4.1.1 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.

4.1.2 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.

4.1.3 Implementácia

V našom projekte používame implementáciu CAPTCHA testu voľne, zdarma dostupnú službu reCAPTCHA[10] od spoločnosti Google. reCAPTCHA, okrem samotného CAPTCHA testu, používateľovi umožňuje vypočuť si zvukovú verziu vygenerovaného textu a v prípade jeho nečítateľnosti 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 kompomnent je implementovaný v súboroch create_account_form.php a create_account_action.php v adresári /html/user.

Nakoniec bol vytvorený script 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 pre boli referencované tabuľkou user. Po mesiaci spúšťania tohoto skriptu každú noc by sa mali v databáze nachádzať len používatelia, ktorí sú reálne aktívni.

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

4.2 Poskytnutie informácií o projekte BOINC@FIIT

4.2.1 Úloha

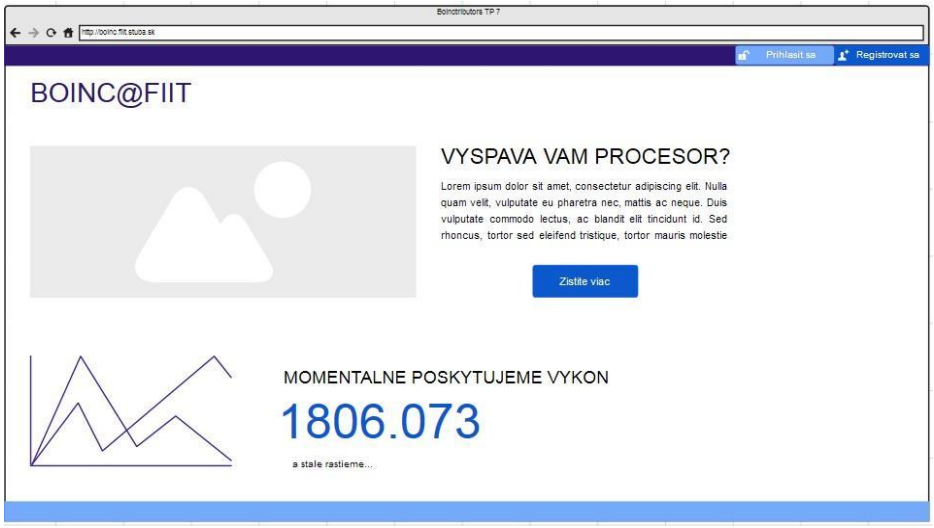
V zmysle budovania povedomia a šírenia informácií o projekte BOINC, spravovanom a realizovanom na fakulte, považujeme za kľúčové navrhnúť jednotné, intuitívne a hlavne jednoducho použiteľné webové rozhranie podávajúce potrebné informácie konzistentne a priamočiaro. Tieto vlastnosti považujeme za fundamentálne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

4.2.2 Návrh

Nízkoúrovňový návrh bol vytvorený pomocou nástroja Moqups[11]. Umiestnením tlačidiel slúžiacich na prihlásenie, resp. registráciu nového používateľa do projektu, spĺňa a dodržiava základné konvencie súčasného webu. V záujme ucelenia 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.

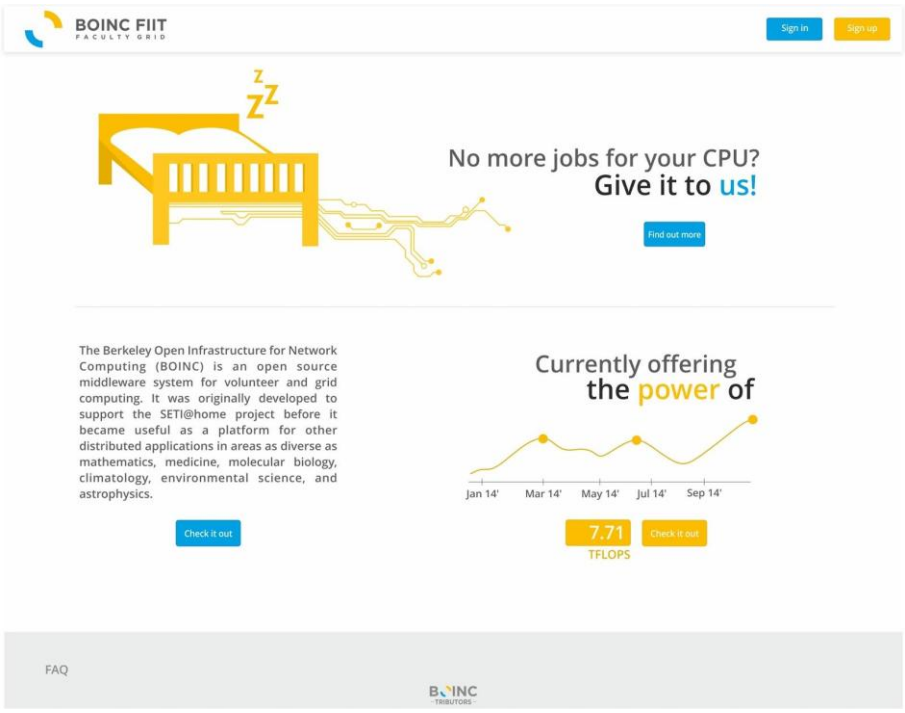
4.2.3 Implementácia

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.

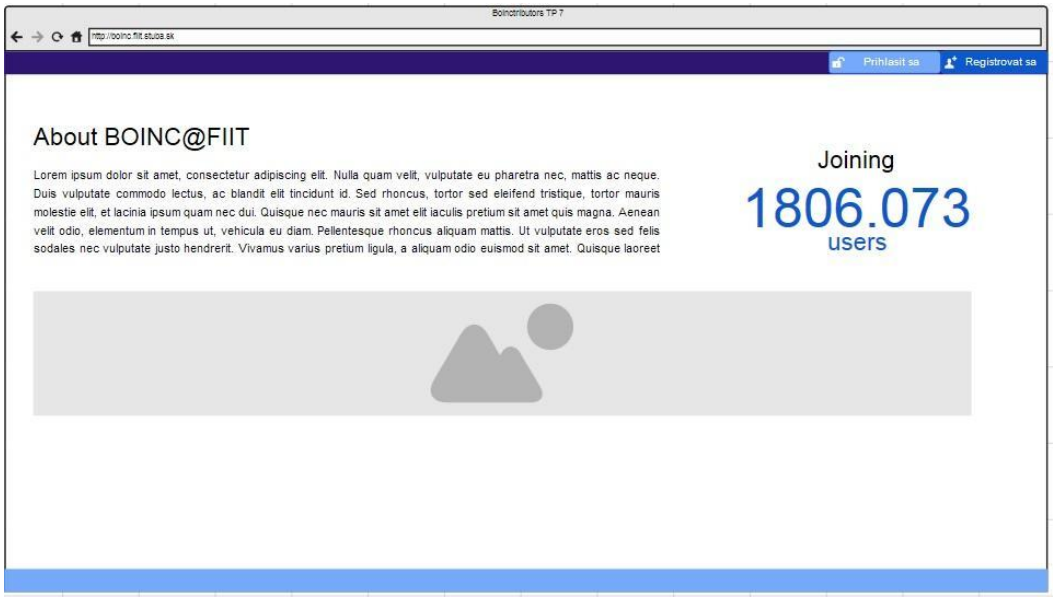


Obr. 1.: Nízkoúrovňový návrh úvodnej stránky projektu

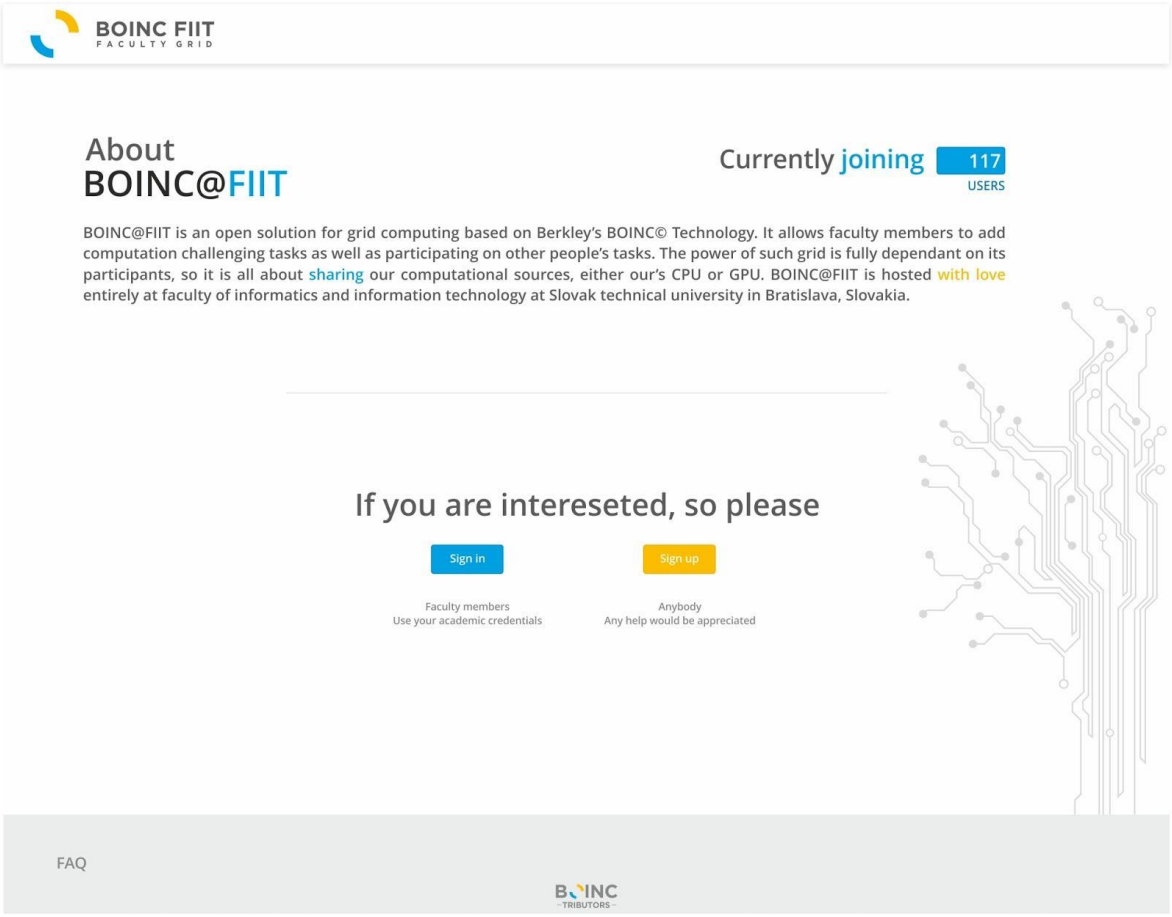
[11] https://moqups.com



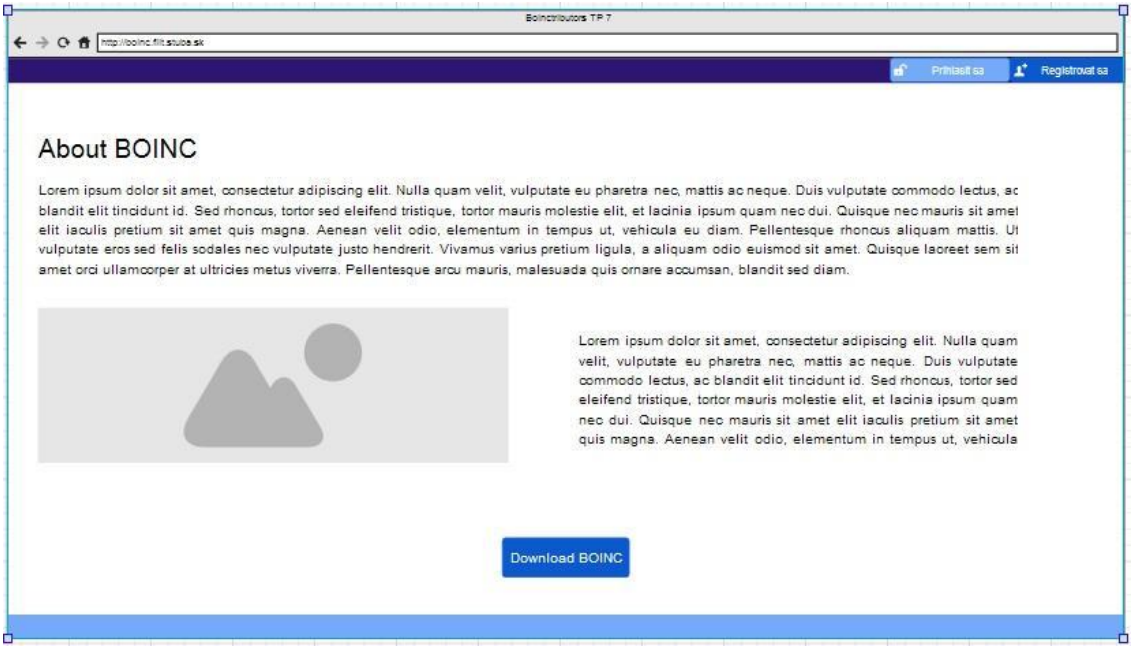
Obr. 2.: Výsledný návrh domovskej stránky projektu



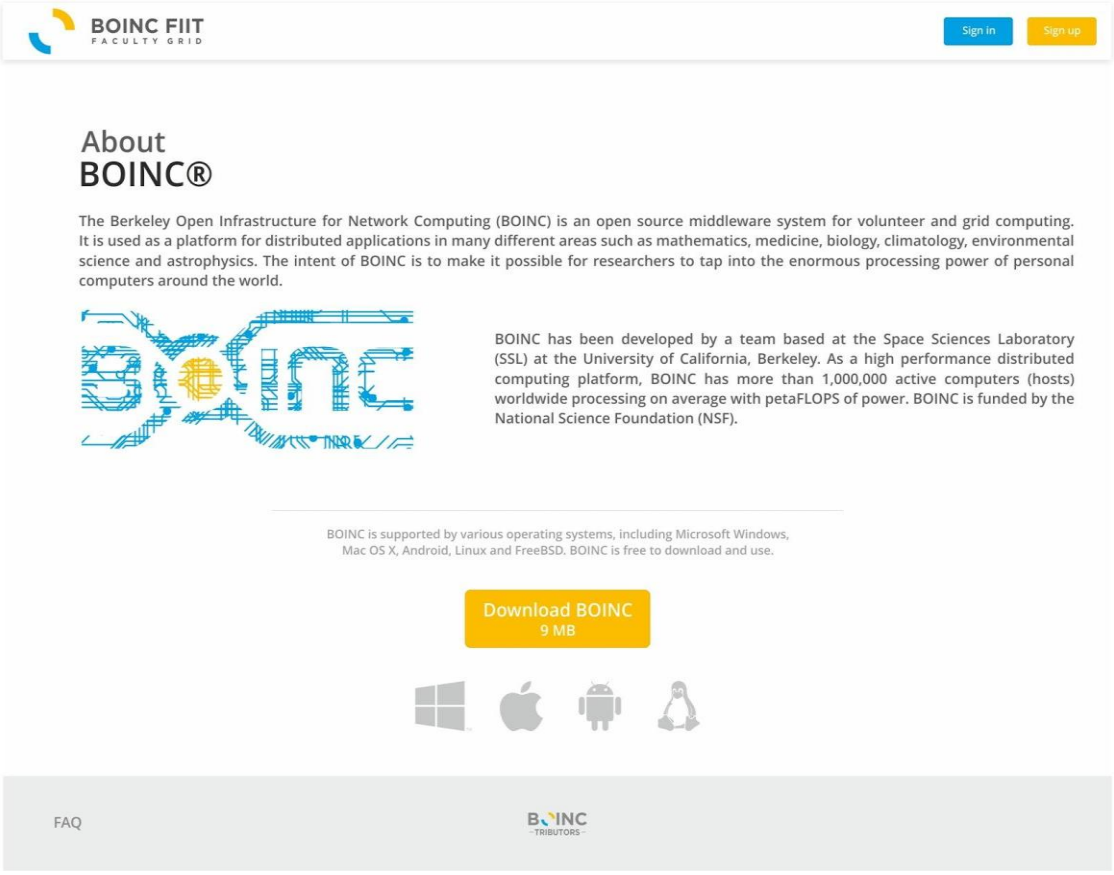
Obr. 3.: Nízkoúrovňový návrh informačnej stránky projektu BOINC@FIIT



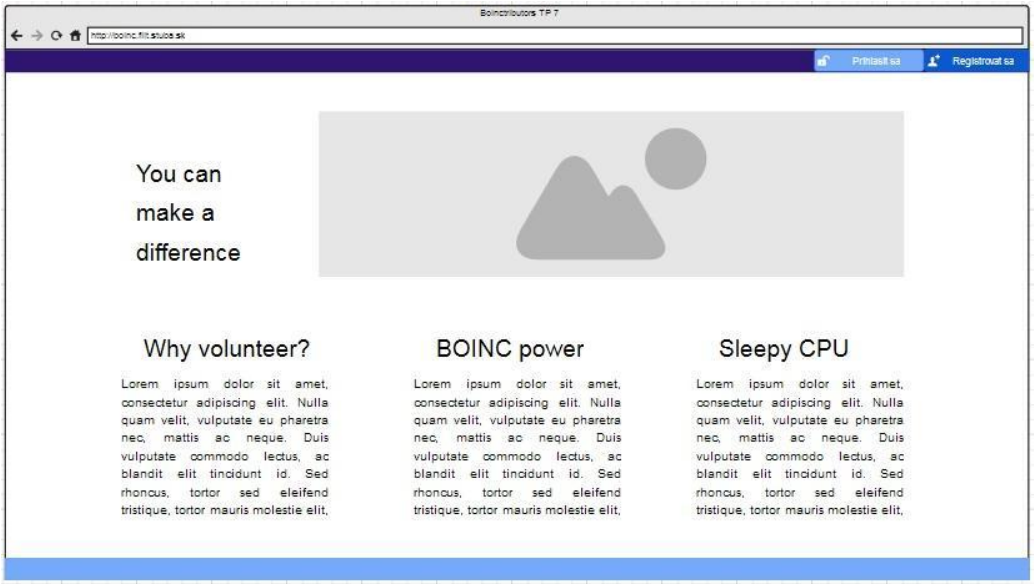
Obr. 4.: Návrh informačnej stránky projektu BOINC@FIIT



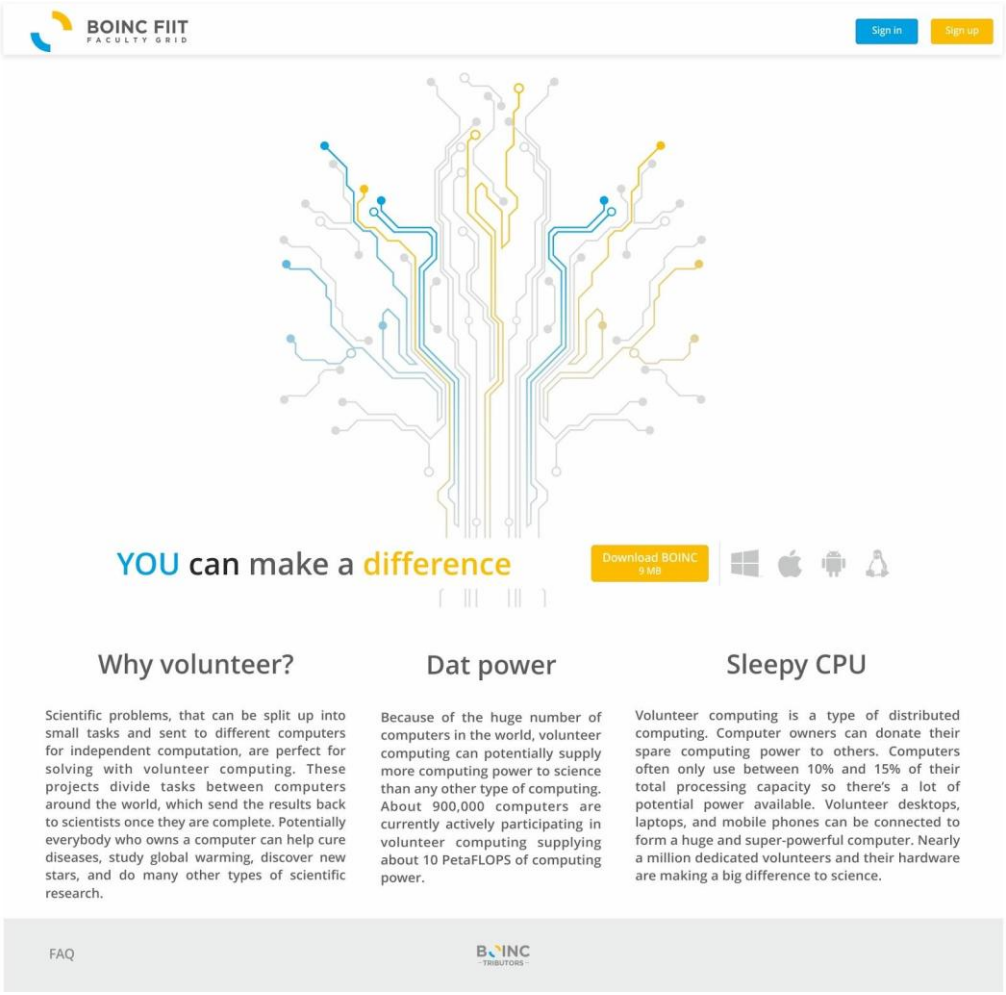
Obr. 5.: Nízkoúrovňový návrh domovskej stránky platformy BOINC



Obr. 6.: Návrh domovskej stránky platformy BOINC



Obr. 7.: Nízkoúrovňový návrh stránky motivujúcej k participácií na projektoch BOINC



Obr. 8.: Návrh stránky motivujúcej k participácií na projektoch BOINC

4.3 Registrácia nového používateľa

4.3.1 Úloha

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

4.3.2 Návrh

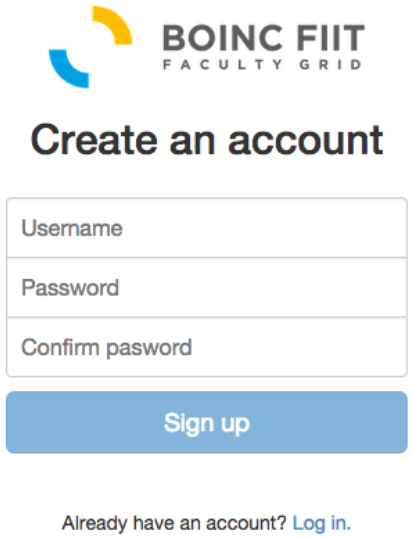
Táto úloha sa bude riešiť vo viacerých paralelných krokoch. V prvom rade do RoR aplikácie musíme pridať gem *Devise*, ktorý rieši autentifikáciu používateľa a umožňuje jednoduchú konfiguráciu používateľského modelu v databáze, ako aj logiky, ktorá sa vykonáva pri registrácii, prihlasovaní, zabudnutí 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 vizuálu všetkých možných krokov registrácie a tiež napojenie SPA modulu na REST backend.

Klientská 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. 9.: Návrh formulára pre registráciu



Obr. 10.: Snímka obrazovky skutočného formulára pre registráciu

4.3.3 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 hashovacou 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.

4.3.4 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 odstavci vyššie, nami napísané testy pokrývajú základnú funkcionalitu navrhnutého REST API, najmä teda, či odpovedá na JSON požiadavky tak, ako očakávame.
Testovanie v klienskej aplikácií je zapezpečené jednoduchým unit testom. Je nutné podotknúť, že kedže pri unit testoch kvôli rýchlosti vykonania nie je žiadúce, aby testy prebiehali s reálnymi požiadavkami na server, bol vytvorený „mock“ prihlásenia/registrácie, nad ktorým sa funkcionalita testuje.

Na unit testovanie v SPA bol použitý rámec Jasmine[12], pričom testy spúšťame pomocou nástroja Karma[13]. Karma nám umožňuje spúšťat testy vo virtuálnom prehliadači. Rámec Jasmine sme vybrali kvôli tomu, že je vychodzou voľbou pri testovaní AngularJS aplikácií, avšak stále existuje možnosť pri ďalších testoch zvoliť iný rámec.

4.4 Autentifikácia

4.4.1 Úloha

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

4.4.2 Návrh

Návrh spočíva z využitia knižnice Devise[14], ktorá pokrýva väčšinu funkcionality pre prihlasovanie. Proces prihlasovania je v Devise pokrytý pomocou štandardného HTML rozhrania postaveného na preposielaní 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[15] v RoR.

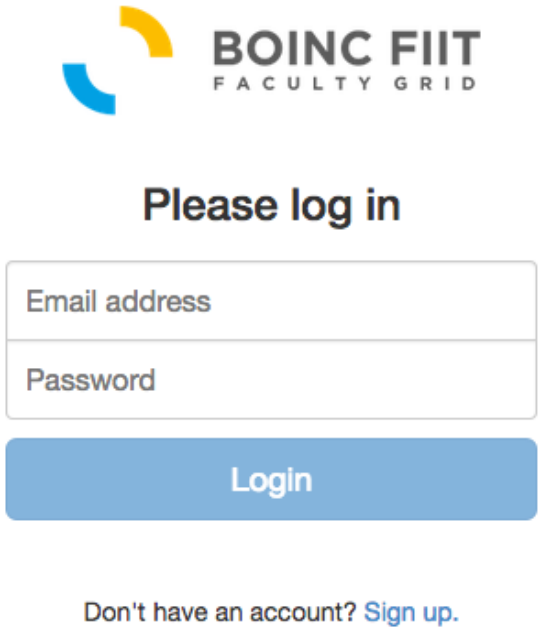
[12] Jasmine: http://jasmine.github.io/
[13] Karma: http://karma-runner.github.io/
[14] Devise: https://github.com/plataformatec/devise
[15] ActiveRecord: http://guides.rubyonrails.org/active_record_basics.html

Obr. 11.: Návrh prihlasovacieho formulára.

4.4.3 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 dlhej neaktivite alebo samotnom odhlásení používateľom.

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



Obr. 12.: Reálny formulár pre prihlásenie

4.5 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 špinte sme sa venovali aj zefektívneniu prác so systémom na správu projektov JIRA.

4.5.1 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:

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

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.

5 Šprint 3 - Jameson

5.1 Sprístupnenie informácií novému návštevníkovi

5.1.1 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.

5.1.2 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, častokrát, 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.

5.2 Práca s existujúcou DB schémou

5.2.1 Úprava schémy vzhľadom na BOINC schému

5.2.1.1 Úloha

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

5.2.1.2 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.

5.2.2 RoR ORM mapovanie

5.2.2.1 Úloha

Vygenerovať ActiveRecord modely pre databázové tabuľky existujúcej schémy BOINC.

5.2.2.2 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[16] 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

[16] 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.

5.3 Nastavenia používateľského účtu

5.3.1 Úloha

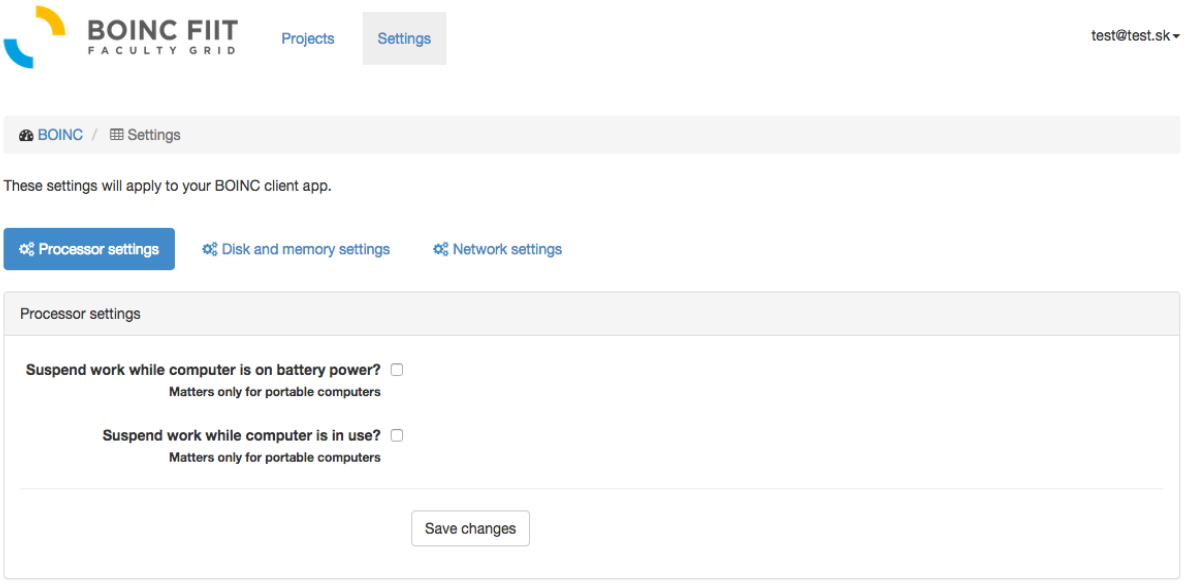
Umožniť používateľovi zmeniť nastavenia svojho účtu. Tieto nastavenia by mali kopírovať možnosti nastavení, ktoré sa nachádzajú v pôvodnej verzii webového rozhrania BOINC. To znamená možnosť zmeniť osobné údaje ako email a meno ako aj zmeniť heslo.

5.3.2 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.

5.3.3 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 singleton, takže jej inštancia je dostupná počas celého behu aplikácie.



Obr. 13.: Formulár pre nastavenie BOINC klientov

6 Celkový pohľad na prvý kontrolný bod

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

7 Použité technológie

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

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

Ruby
Oficiálny web: https://www.ruby-lang.org/en/
Verzia: 2.1.5
Popis: vysokoúrovňový objektovo orientovaný interpretovaný programovací jazyk

Ruby On Rails
Oficiálny web: http://rubyonrails.org/
Verzia: 4.1.5
Popis: opensource webový rámec pre programovací jazyk Ruby

JavaScript
Oficiálny web: http://www.w3schools.com/js/
Verzia: ECMA-262, revizia 5 (V8 engine)
Popis: interpretovaný webový programovací jazyk, ktorý sa primárne používa vo webových prehliadačoch, no svoje miesto má i na strane servera.

AngularJS
Oficiálny web: https://angularjs.org/
Verzia: 1.3.0
Popis: webový rámec naprogramovaný v JavaScripte pre SPA aplikácie

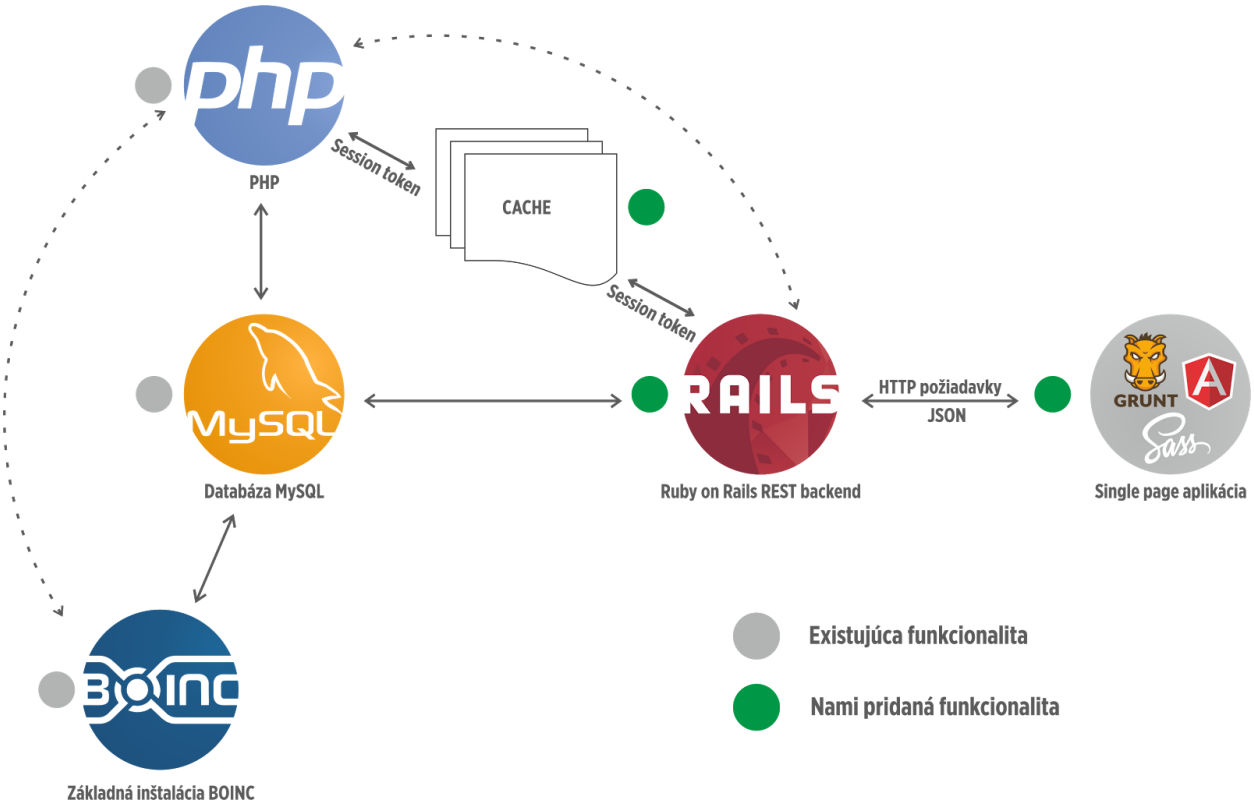
CSS3
Oficiálny web: http://www.w3schools.com/css/css3_intro.asp
Verzia: 3.0
Popis: štandard pre zápis kaskádových štýlov pre webové stránky.

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.

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.

8 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. 14.: Návrh architektúry riešenia

8.1 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.