Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

# Crowdex

*Dokumentácia k riadeniu projektu*

| | |
|---|---|
| Vedúci tímu: | Ing. Michal Kompan, PhD. |
| Členovia tímu: | Bc. Dušan Cymorek, Bc. Peter Gašpar, Bc. Vladimír Ľalík, Bc. Michal Polko, Bc. Miroslav Šafárik, Bc. Slavomír Šárik, Bc. Štefan Šmihla |
| Akademický rok: | 2014/15 |

# Obsah

# 1 Úvod

Tento dokument obsahuje dokumentáciu riadenia projektu Manažment experimentov (Crowdex) platnú k prvému kontrolnému bodu. Dokument vznikol v rámci predmetu Tímový projekt v akademickom roku 2014/15 na Fakulte informatiky a informačných technológií Slovenskej technickej univerzity v Bratislave.

Dokument je členený do logických kapitol. Kapitola 2 obsahuje rozdelenie úloh jednotlivých členov tímu, spolu s prehľadom podielu práce na jednotlivých častiach dokumentácie.

Kapitoly 3 až 10 obsahujú opis a zodpovednosti manažérskych úloh členov tímu a pridružené metodiky.

Prílohy A a B obsahujú zápisy zo stretnutí a retrospektívy k šprintom.

# 2 Úlohy členov tímu

## 2.1 Manažérske úlohy

Manažérske úlohy v rámci projektu sme si rozdelili nasledovne:

| Manažérska úloha | Zodpovedná osoba |
|---|---|
| Manažment dokumentácie | Dušan Cymorek |
| Manažment rozvrhu | Peter Gašpar |
| Manažment rizík | Vladimír Ľalík |
| Manažment rozsahu | Michal Polko |
| Manažment podpory vývoja a integrácie | Slavomír Šárik |
| Manažment komunikácie | Miroslav Šafárik |
| Manažment testovania a prehliadok | Štefan Šmihla |

*Tabuľka 2.1: Rozdelenie manažérskych úloh*

## 2.2 Krátkodobé úlohy

V tabuľke nižšie uvádzame krátkodobé úlohy jednotlivých členov tímu, ktoré sa vyskytli počas riešenia projektu.

| Úloha | Zodpovedná osoba |
|---|---|
| Tvorba zápisov a retrospektív | Cyklické striedanie všetkých členov |
| Webová stránka tímu | Peter Gašpar, Michal Polko |
| Konfigurácia produkčného servera, vývojového a produkčného prostredia, nástrojov pre manažment úloh - YouTrack a pre verziovanie zdrojového kódu – Github | Slavomír Šárik |
| Prihláška na TP Cup | celý tím |

*Tabuľka 2.2: Rozdelenie krátkodobých úloh*

## 2.3 Podiel práce na jednotlivých častiach dokumentácie

### 2.3.1 Dokumentácia k riadeniu projektu

| Kapitola | Autori (podľa metodík) |
|---|---|
| Úvod | Dušan Cymorek |
| Úlohy členov tímu | Dušan Cymorek, Peter Gašpar, Slavomír Šárik |

| | |
|---|---|
| Manažment dokumentácie | Dušan Cymorek |
| Manažment plánu | Peter Gašpar |
| Manažment rizík | Vladimír Ľalík, Miroslav Šafárik |
| Manažment kvality | Vladimír Ľalík, Peter Gašpar |
| Manažment rozsahu | Michal Polko |
| Manažment podpory vývoja a integrácie | Slavomír Šárik |
| Manažment komunikácie | Miroslav Šafárik |
| Manažment testovania a prehliadok | Štefan Šmihla, Michal Polko |
| Zápisy zo stretnutí | Celý tím |
| Retrospektívy k šprintom | Celý tím |

*Tabuľka 2.3: Autori jednotlivých kapitol dokumentácie k riadeniu projektu*

### 2.3.2 Dokumentácia k inžinierskemu dielu

| **Kapitola** | **Autori** |
|---|---|
| Úvod | Michal Polko |
| Ciele na zimný semester | Michal Polko |
| Architektúra systému | Miroslav Šafárik |
| Registrácia používateľa | Štefan Šmihla |
| Autentifikácia používateľa | Slavomír Šárik |
| Profil používateľa | Miroslav Šafárik |
| Detail experimentu | Miroslav Šafárik |
| Pridanie a správa experimentu | Dušan Cymorek, Peter Gašpar, Vladimír Ľalík |
| Notifikácie | Slavomír Šárik |
| Nastavenia profilu používateľa | Štefan Šmihla |

| | |
|---|---|
| Dizajn a interakcia s používateľom | Peter Gašpar, Michal Polko |

*Tabuľka 2.4: Autori jednotlivých kapitol dokumentácie k inžinierskemu dielu*

# 3 Manažment dokumentácie

Manažér dokumentácie dohliada na tvorbu dokumentácií vytváraných priebežne v jednotlivých šprintoch, ale i na dokumentovanie zdrojového kódu. Vytvára šablóny a upravuje dokumenty tak, aby bola dodržaná rovnaká identita naprieč dokumentáciou v tíme. Okrem toho v projekte *Crowdex* zodpovedá za údržbu prekladov v aplikácii.

Dokumenty sú v súčasnej dobe vytvárané v prostredí *Google Drive*, odkiaľ sa pred publikovaním upravujú v programe *Microsoft Word/Excel*. V budúcich fázach riešenia projektu bude uskutočnený prechod na tvorbu dokumentácie v nástroji *LaTeX* v spojení s distribuovaným nástrojom riadenia revízií *Git*.

## 3.1 Metodika – Dokumentovanie kódu

### 3.1.1 Úvod

Táto metodika je zameraná na oblasť dokumentovania zdrojového kódu v prostredí webového rámca *Ruby on Rails*. Cieľom tejto metodiky je stanoviť jednotné pravidlá a postupy pri dokumentovaní tried, metód a atribútov, uviesť užitočné tipy a prispieť tak k sprehľadneniu zdrojového kódu, ktorý je zdieľaný medzi viacerými autormi. Metodika opisuje prácu s dokumentačným nástrojom *YARD* a vývojovým prostredím *RubyMine*.

Metodika vychádza z odporúčaných prístupov ku dokumentovaniu zdrojového kódu v jazyku *Ruby* a webovom rámcom *Ruby on Rails*. Je určená pre všetkých členov tímu, ktorí nielen implementujú nové triedy a metódy, ale aj používajú už implementované časti aplikácie.

### 3.1.2 Zoznam nadväzujúcich metodík a dokumentov

1. http://www.rubydoc.info/gems/yard/file/docs/GettingStarted.md

### 3.1.3 Vymedzenie pojmov a skratiek

- ***Dokumentácia*** – opis triedy, metódy, atribútu, prípadne časti kódu, ktorý slúži na lepšie pochopenie kódu písaného programátorom.
- ***Trieda*** – okrem triedy sa tento pojem súhrnne používa aj pre *controller*, *model*, *helper*.
- ***Značka*** (*angl. tag*) – slúži na definovanie rôznych častí komentárov pre dokumentačný nástroj YARD.

### 3.1.4 Postupy

V tejto kapitole sú uvedené postupy, ktoré je potrebné dodržiavať pri dokumentovaní zdrojového kódu. Nachádzajú sa tu aj užitočné tipy a klávesové skratky, ktoré uľahčujú vytváranie a používanie dokumentácie.

#### 3.1.4.1 Vytvorenie dokumentácie a jej používanie

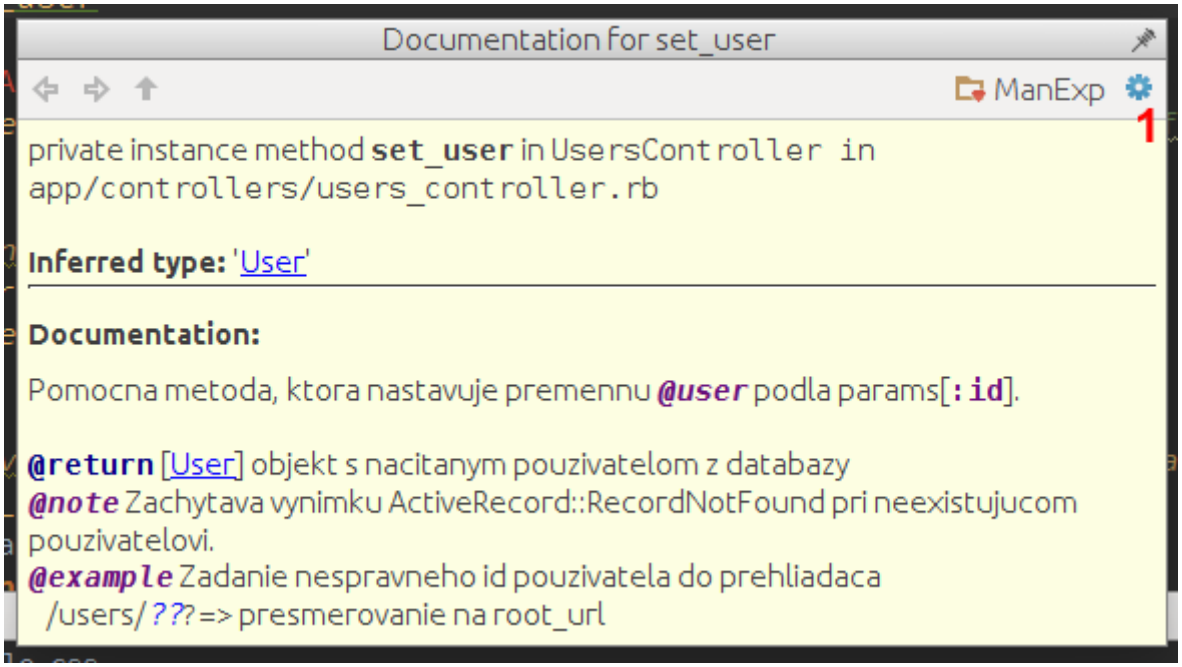
Spôsob vytvorenia dokumentácie závisí od spôsobu jej používania:

1. Zobrazenie dokumentácie priamo v prostredí RubyMine,

2. Zobrazenie formátovanej dokumentácie na lokálnom serveri.

**Zobrazenie dokumentácie priamo v prostredí RubyMine**

Dokumentácia sa zobrazuje po nastavení kurzora na názov triedy, metódy, prípadne atribútu a stlačení klávesovej skratky *Ctrl+Q*. Dokumentácia sa automaticky obnovuje po úpravách, nie je teda potrebné vykonávať žiadne ďalšie kroky.




*Obrázok 3.1: Okno s dokumentáciou v prostredí RubyMine*

Opis obrazovky:

1. Po kliknutí na ikonu ozubeného kolieska môžeme nastaviť veľkosť textu v okne

**Zobrazenie formátovanej dokumentácie na lokálnom serveri**

V tomto prípade je dokumentácia zobrazená ako formátovaná webová stránka s možnosťou vyhľadávania. Server sa štandardne nachádza na adrese http://localhost:8808. Pre použitie tohto spôsobu zobrazenia je potrebné spustiť server príkazom *yard server --reload*, ktorý zabezpečí automatickú aktualizáciu dokumentácie pri obnovení stránky.


*Obrázok 3.2: Webová verzia dokumentácie*

Opis obrazovky:

1. Názov metódy
2. Poznámka
3. Opis metódy
4. Príklady použitia, v našom prípade ukážka prípadu riešenia požiadavky na zobrazenie neexistujúceho používateľa
5. Návratová hodnota metódy
6. Odkaz na zobrazenie, prípadne skrytie zdrojového kódu metódy

### 3.1.4.2 Všeobecné pravidlá pre dokumentáciu kódu

Označenie a forma komentárov:

- Komentáre v zdrojových súboroch sa označujú znakom #, za ktorým nasleduje práve jedna medzera.
- Značky pre HTML, CSS a JS komentáre sú od textu komentára oddelené jednou medzerou, forma je nasledovná:
  - HTML komentár:

```
<!-- Komentar -->
<!-- Komentar
  -- na viacero
  -- riadkov
  -->
```

  - CSS a JS komentár:

```
/* Komentar */
```

```
/* Komentar
 * na viacero
 * riadkov
 */
```

- Komentáre umiestňujte vždy nad časťou kódu, na ktorú sa komentár vzťahuje. Toto pravidlo platí celoplošne pre všetky komentáre.
- Komentáre píšte v slovenskom jazyku, s veľkým začiatočným písmenom. V prípade použitia značiek dodržiavajte štýl používaný v uvedených príkladoch.
- Používajte iba povolené značky, pričom dodržujte ich poradie.

Dokumentácia má vo všeobecnosti nasledovnú štruktúru:

```
# Opis dokumentovaného prvku (100 znakov na riadok)
# Opis dokumentovaného prvku - pokračovanie (100 znakov na riadok)
# Prázdny riadok
# @značka
# @značka
def metoda_x(parametre)
  ...
  # Tento komentar opisuje, zdvodnuje implementaciu kodu nizsie
  parameter.to_i unless parameter.blank?
  ...
end
```

#### 3.1.4.3 Dokumentovanie tried

Pre dokumentovanie tried použite nasledovné značky (v dokumentácii nasledujú v uvedenom poradí):

**@author**

Význam:

- Uvedenie mena a priezviska autora, *@git_username*.
- Povinné.

Príklad:

```
# @author Dušan Cymorek, @dusancymorek
```

**@since**

Význam:

- Uvedenie verzie programu, v ktorej bola daná trieda pridaná. Verziu programu určuje manažér integrácie, aktuálna verzia aplikácie (na *Git master vetve*) je uvedená v súbore *README.md*.
- Povinné.

Príklad:

```
# @since 0.2.4
```

**@note**

Význam:

- Poznámka, ktorá môže pomôcť ďalšiemu členovi tímu.

Príklad:

```
# @note Pri vytváraní inicializuje hodnotu atribútu x na 1 (nie nula).
```

**@deprecated**

Význam:

- Vysvetlenie a upozornenie na nahradenie triedy inou triedou, na ktorú je potrebné uviesť referenciu.

Príklad:

```
# @deprecated Zlúčené s controllerom pre triedu X
# {#controller_pre_trieduX}, z dôvodu deduplikovania kódu.
```

**@see odkaz popis**

Význam:

- Uvedenie odkazu na inú triedu, metódu, atribút, prípadne užitočnú webovú stránku.

Príklad:

```
# @see http://yardoc.org/ YARD
```

**@todo nadpis**

**opis**

Význam:

- Uvedenie chýbajúcich častí v triede, ktoré je treba implementovať, prípadne opraviť.
- Opis musí byť odsadený o tri medzery od znaku #.

Príklad:

```
# @todo Pridať prepojenie na tabuľku user_skills
#   Potrebné pred implementáciou vyplňovania skúseností používateľa
#   v nastavení profilu.
```

#### 3.1.4.4 Dokumentovanie metód

Dokumentovanie metód realizujte prostredníctvom nasledovných značiek (opäť záleží na poradí):

**@param názov [typ] popis parametra**

Význam:

- Opisuje vstupný parameter metódy.
- Povinné, ak má metóda vstupné parametre.

Príklad:

```
# @param user_fullname [String] meno a priezvisko používateľa oddelené
# medzerou
```

**@option názov Hash parametra [typ] kľúč pre parameter (štandardná hodnota) opis parametra**

Význam:

- Táto značka sa používa v spojení so značkou *@param*, ktorý opisuje parameter typu Hash. Značka *@option* opisuje jednotlivé prvky v množine.
- Povinné, ak má metóda vstupné parametre vo forme *Hash.*

Príklad:

```
# @param user_params [Hash] parametre triedy User získané z prehliadača
# @option user_params [String] :id friendly-id používateľa
# @option user_params [String] :firstname meno používateľa
# @option user_params [String] :surname priezvisko používateľa
# @option user_params [String] :school (‘FIIT’) škola používateľa
```

**@return [typ] popis návratovej hodnoty**

Význam:

- Uvedenie návratovej hodnoty metódy.
- Podobne ako pri značke *@param*, v prípade viacerých návratových hodnôt je potrebné každú opísať v samostatnej značke *@return* s vysvetlením, kedy je ktorá hodnota vrátená.
- Povinné.

Príklad:

```
# @return [User] nájdený požívateľ podľa id
```

**@raise [typ] popis výnimky**

Význam:

- Uvedenie prípadu, v ktorom metóda vyvoláva výnimku, ktorú neošetruje.

Príklad:

```
# @raise [BudgetBalanceError] ak nemá zadávateľ dostatok kreditu na účte
# na zvýšenie rozpočtu experimentu
```

**@note** – viď predchádzajúca kapitola

**@deprecated** – viď predchádzajúca kapitola

**@see odkaz popis** – viď predchádzajúca kapitola

**@todo nadpis**

**opis** – viď predchádzajúca kapitola

**@example nadpis**

**príklad**

Význam:

- Uvedenie príkladu použitia metódy, ošetrenia výnimky, prípadne vyvolania výnimky.
- Príklad musí byť odsadený o tri medzery od znaku #.

Príklad:

```
# @example Presmerovanie pri zadaní nesprávneho používateľa
#   /users/??? => presmerovanie na root_url, zobrazenie chybovej hlášky
```

#### 3.1.4.5 Dokumentovanie atribútov

Pre dokumentovanie atribútov dodržujte nasledovné pravidlá:

- Všetky atribúty musia byť zdokumentované – uveďte na čo atribút slúži.
- Atribúty združujte do logických celkov v prípade, že zastrešujú spoločnú funkcionalitu. Podobne združujte atribúty spolu so súvisiacimi pomocnými funkciami. Napr.:

```
# Adresy - fakturacna, dodacia
has_many :addresses, dependent: :destroy
accepts_nested_attributes_for :addresses
```

#### 3.1.4.6 Dokumentovanie iných častí kódu

Rozsah ďalších komentárov v kóde sa snažte obmedziť. V prípade, že je nutné vysvetliť úsek kódu, myšlienku, prípadne upozorniť na niečo špeciálne, použite štandardný komentár. Ak je potrebné niečo upraviť, doimplementovať, použite na začiatku komentára slovo *TODO*.

Príklad:

```
def metoda_x
  ...
  # Overenie hash v pripade, ze je experiment skryty vo vyhladavani
  # TODO Refactor - odstranenie zlozenej podmienky
  ...
end
```

#### 3.1.4.7 Dokumentovanie HTML, CSS a JS kódu

**HTML** súbory komentujte nasledovne:

- Na začiatku súboru (pred samotným kódom) uveďte krátky opis obsahu HTML súboru – na čo slúži, čo zobrazuje.
- Nekomentujte jednotlivé prvky kódu, ale logické celky – napríklad formuláre, karty, rozbaľovacie menu.
- Iné komentáre používajte len v krajných prípadoch – ošetrenie chýb, prípadne potreby dodatočnej úpravy (*TODO*).

Pre **CSS** platia analogické pravidlá. Pre logické celky (jednotlivé bloky so štýlmi) uveďte, na čo sa daný blok používa.

Pre **JS** metódy použite rovnaký štýl dokumentovania ako pre metódy v jazyku *Ruby* (kapitola 3.1.4.4 Dokumentovanie metód).

# 4 Manažment plánu

Manažér plánu dohliada na prideľovanie, plnenie a zaznamenávanie priebehu úloh v rámci celého projektu. V našom tíme sme zaviedli viaceré pravidlá, ktoré zabezpečujú korektné a včasné plnenie úloh a ich náležitú archiváciu. Využívame pritom viaceré podporné nástroje:

- *YouTrack* – nástroj na manažment projektu a správu úloh,
- *Toggl* – nástroj na meranie času stráveného nad úlohou,
- *Google Drive* – nástroj na vzájomnú výmenu súborov s možnosťou kolaboratívneho hodnotenie a poznámkovania.

Pri stanovovaní úloh a plánu sme postupovali v súlade s metodikou vývoja *Scrum*. V spolupráci s vlastníkom produktu (angl. *Product Owner*) sme zostavili hlavný *Product Backlog* – zoznam všetkých základných úloh, ktoré budeme plniť počas realizácie projektu.

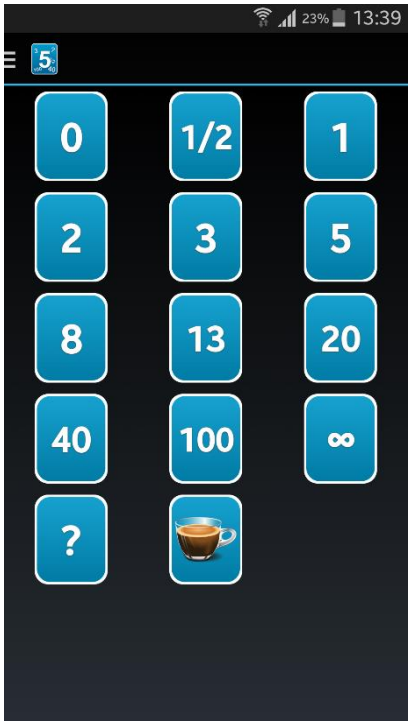
Pred začiatkom každého šprintu vybral vlastník produktu skupinu úloh, ktoré sa riešili nasledujúce obdobie. Postupovali sme pritom v súlade s ohodnotením podľa ich časovej náročnosti. Pri stanovení náročnosti sme používali nasledujúce mobilné aplikácie:

- Scrum Poker[1] (operačný systém Android),
- Scrum Card Desk[2] (operačný systém iOS).

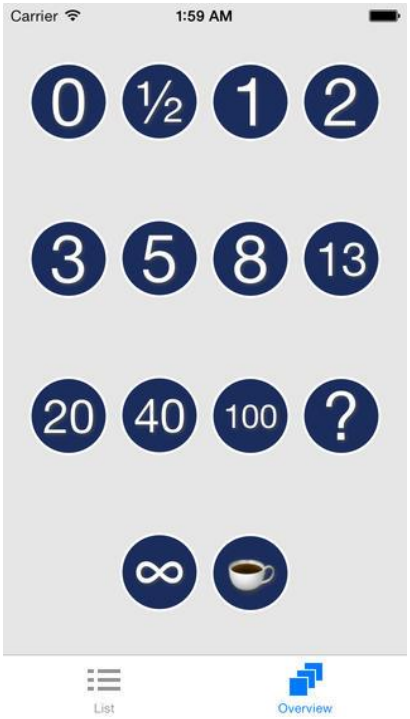
Každý člen tímu ohodnotil úlohu číslom kartičky, ktoré zodpovedalo odhadovanému počtu potrebných človekohodín (8 hodín) na realizáciu danej úlohy. Následne sa všetky odhady ukázali a členovia, ktorí ohodnotili úlohu vyšším, prípadne nižším číslom ako väčšina, musel zdôvodniť, prečo považuje úlohu za náročnejšiu, respektíve jednoduchšiu. Na základe diskusie sa určila finálna náročnosť úlohy, ktorá bola východiskom pri rozdeľovaní úloh opísanom v ďalšom odseku.

[1] https://play.google.com/store/apps/details?id=artarmin.android.scrum.poker

[2] https://itunes.apple.com/us/app/scrum-card-deck/id779873086?mt=8




*Obrázok 4.1: Ukážka aplikácie Scrum Poker*




*Obrázok 4.2: Ukážka aplikácie Scrum Card Desk*

Pri rozdeľovaní úloh sme sa primárne zamerali aj na svoje vlastné schopnosti a vedomosti v rámci tímu. Platili pritom nasledujúce pravidlá:

- Každý člen tímu sa zapísal na aspoň 1 úlohu.
- Každá úloha nižšieho rozsahu (ohodnotená rozsahom maximálne 1 deň) mala prideleného aspoň 1 člena (riešiteľa).

- Každá úloha vyššieho rozsahu (ohodnotená rozsahom viac ako 1 deň) mala pridelených aspoň 2 členov (riešiteľov).
- Každej úlohe bola pridelená zodpovedná osoba, ktorá mala za úlohu vytvoriť špecifikáciu pre danú úlohu. Na základe tejto špecifikácie sa následne postupovalo pri implementácii a testovaní.

Tvorba špecifikácia bola časovo ohraničená na základe dĺžky šprintu. Tabuľka vyjadruje stanovené rozloženie času na šprint. Keďže sme postupovali inkrementálne a iteratívne, vytvorené špecifikácie sa aj v období implementácie a testovania podľa potreby upravovali po dohode so zodpovednou osobou.

| **Deň / Dĺžka šprintu** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 T | Tvorba špecifikácie | | | Implementácia / Testovanie | | | | - | | | | | | |
| 2 T | Tvorba špecifikácie | | | | Implementácia / Testovanie | | | | | | | | | |

*Tabuľka 4.1: Rozdelenie činností počas šprintov*

**Poznámka:** T = týždeň

Po dokončení špecifikácie zodpovedné osoby vytvorili v nástroji *YouTrack* príslušné podúlohy, ktoré vznikli zvýšením granularity základných úloh z *Product Backlogu*. Tieto podúlohy boli následne pridelené jednotlivým riešiteľom (danej základnej úlohy). Po dokončení každej podúlohy zodpovedná osoba verifikovala stav jej riešenia a na základe toho rozhodla, či bola úspešne splnená alebo či sa vyžadovali ďalšie korekcie. V prípade, že boli splnené všetky podúlohy základnej úlohy, zodpovedná osoba označila základnú úlohu ako verifikovanú a pre daný šprint aj uzavretú.

Na zaznamenávanie času stráveného nad riešením projektu sme používali bezplatný nástroj *Toggl*[3]. Časy sme pritom priebežne ukladali do *YouTracku* k príslušným úlohám. Na záver šprintu sme pre účely tvorby retrospektívy vygenerovali export z nástroja *YouTrack*. Ten zahŕňal najmä stav splnenia úloh, riešiteľov, zodpovedné osoby a celkový čas trvania.

Tabuľka uvádza súhrnne dĺžky šprintov, ktoré sme doposiaľ úspešne zrealizovali.

| **Č. šprintu** | **Dátum začiatku** | **Dátum ukončenia** | **Dĺžka trvania** |
|---|---|---|---|
| 1 | 14. 10. 2014 | 27. 10. 2014 | 2 týždne |
| 2 | 28. 10. 2014 | 10. 11. 2014 | 2 týždne |
| 3 | 11. 11. 2014 | 17. 11. 2014 | 1 týždeň |

[3] www.toggl.com

*Tabuľka 4.2: Dĺžky trvania šprintov*

Súčasťou nášho projektu boli aj nasledujúce úlohy, ktoré sme nerealizovali v rámci jednotlivých šprintov:

- vytvorenie webového sídla,
- vytvorenie prihlášky do TP Cupu,
- prepojenie aplikácie s Git službou.

Tieto úlohy sa nachádzajú v kapitole č. 2 –

Úlohy členov tímu aj s príslušnými riešiteľmi.

# 5 Manažment rizík

Manažér rizík zodpovedá za riešenie možných rizík, ktoré môžu nastať počas vývoja projektu. Jeho úlohou v našom tíme je identifikovať možné riziká, následne analyzovať ich následky a definovať kroky potrebné pre minimalizovanie dôsledkov rizika. Konkrétne v našom tíme máme aplikovaných niekoľko postupov ako predchádzať rizikám. Napríklad počas šprintu máme určené dátumy, kedy musia byť vytvorené špecifikácie, ktoré sú záväzné pre celý tím. Na konci šprintu vždy zodpovedná osoba za danú úlohu vykoná prehliadku kódu a až po schválení tejto funkcionality sa vykoná integrácia tohto kódu. Tieto postupy nám pomáhajú zachovávať vysokú kvalitu kódu, čo nám uľahčí vývoj v nasledujúcich etapách projektu.

## 5.1 Metodika – prehliadka zdrojového kódu

### 5.1.1 Úvod

Cieľom tejto metodiky je zabezpečiť kroky, vykonaním ktorých dôjde k zlepšeniu čistoty a konzistentnosti zdrojového kódu a k nájdeniu skrytých chýb. Čistý kód je jeden zo základných predpokladov jeho udržateľnosti, znižuje sa pri ňom počet chýb, znižujú sa riziká v projekte a výrazne zvyšuje šance pre jeho úspech. Hoci riešitelia úloh majú k dispozícii metodiku, ktorá popisuje konvencie pre písanie zdrojového kódu, mnohokrát dochádza k ich porušovaniu z rozličných dôvodov (časový stres, komplexnosť úlohy, zložitosť riešenia a pod.). Z tohto dôvodu je podstatné, aby človek, ktorý danú úlohu neriešil, skontroloval zdrojový kód a rozhodol, či sa v kóde nenachádzajú tzv. pachy, či nie sú v kóde skryté chyby a či je dané riešenie v súlade so špecifikáciou. Metodika nepokrýva spôsoby a kroky, akými vykonať nápravu problematických častí zdrojového kódu (ako vykonať refaktorizáciu).

Táto metodika je určená pre zodpovedné osoby. Okrem iných činností, ktoré tieto osoby vykonávajú (písanie špecifikácie, dohliadnutie na stav plnenia úlohy, vytvorenie úloh v systéme *YouTrack*), je teda ich zodpovednosťou aj vykonať prehliadku kódu v súlade s touto metodikou. Riešitelia úlohy nevykonávajú prehliadku zdrojového kódu, t.j. priamo sa ich táto metodika nedotýka, no dodržiavaním konvencií v tejto metodike môžu výrazne urýchliť proces ukončenia prác na úlohe.

### 5.1.2 Zoznam nadväzujúcich metodík a dokumentov

Táto metodika sa na určitých miestach odvoláva na nasledujúce metodiky, ktoré vytvorili autori tímu Code Crushers a sú uvedené v tomto dokumente:

- Dokumentovanie zdrojového kódu
- Testovanie softvéru
- Nahlasovanie chýb
- Písanie kódu

#### 5.1.2.1 Vymedzenie pojmov a skratiek

- ***Šprint*** – časové obdobie (zvyčajne jeden alebo dva týždne) v metodike *Scrum*, počas ktorého sa pracuje na dohodnutých úlohách.
- ***Zodpovedná osoba*** (ZO) – osoba, ktorá v danom šprinte zodpovedá za splnenie príslušnej úlohy, resp. príslušných úloh.
- ***Riešiteľ*** – osoba, ktorá rieši príslušnú úlohu, resp. príslušné úlohy v danom šprinte.

### 5.1.3 Postupy

V tejto časti opíšeme jednotlivé postupy, ktoré treba vykonať pri prehliadke kódu.

#### 5.1.3.1 Začiatok prehliadky

**Stav úlohy**

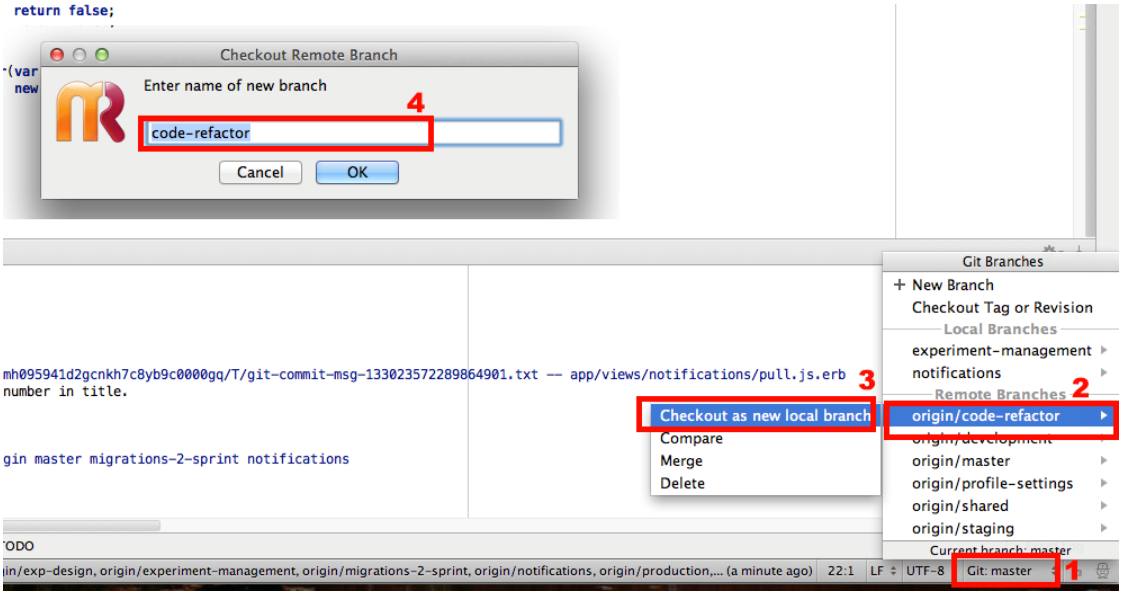
Zodpovedná osoba začne vykonávať prehliadku zdrojového kódu v momente, keď riešiteľ označí v systéme *YouTrack* danú úlohu ako *Fixed*. Informáciu o stave úlohy je možné zistiť dvomi spôsobmi:

- priebežnou kontrolou emailovej schránky, na ktorú zodpovednej osobe prichádzajú všetky informácie o úlohe (čas riešenia, zmeny stavu úlohy),
- priebežnou kontrolou systému *YouTrack*:
    1. prihláste sa do systému *YouTrack* s Vašimi prihlasovacími údajmi,
    2. kliknite v hornom menu na odkaz *Agile Boards,*
    3. skontrolujte, či je príslušná úloha umiestnená v stĺpci *Fixed.*

**Stiahnutie príslušnej vetvy**

V ďalšom kroku je potrebné stiahnuť lokálnu kópiu príslušnej *Git* vetvy, v ktorej bola daná úloha vyvíjaná. Stiahnutie vetvy prebieha v nástroji *RubyMine* (Obrázok 5.1). Postup stiahnutia vetvy je nasledujúci:

1. kliknite na aktuálnu vetvu v nástroji *RubyMine* v pravom dolnom rohu,
2. kliknite na vzdialenú vetvu, nad ktorou chcete vykonať prehliadku,
3. kliknite na *checkout as new local branch,*
4. zadajte názov pre lokálnu kópiu vetvy, prípadne ponechajte odporúčaný a potvrďte.

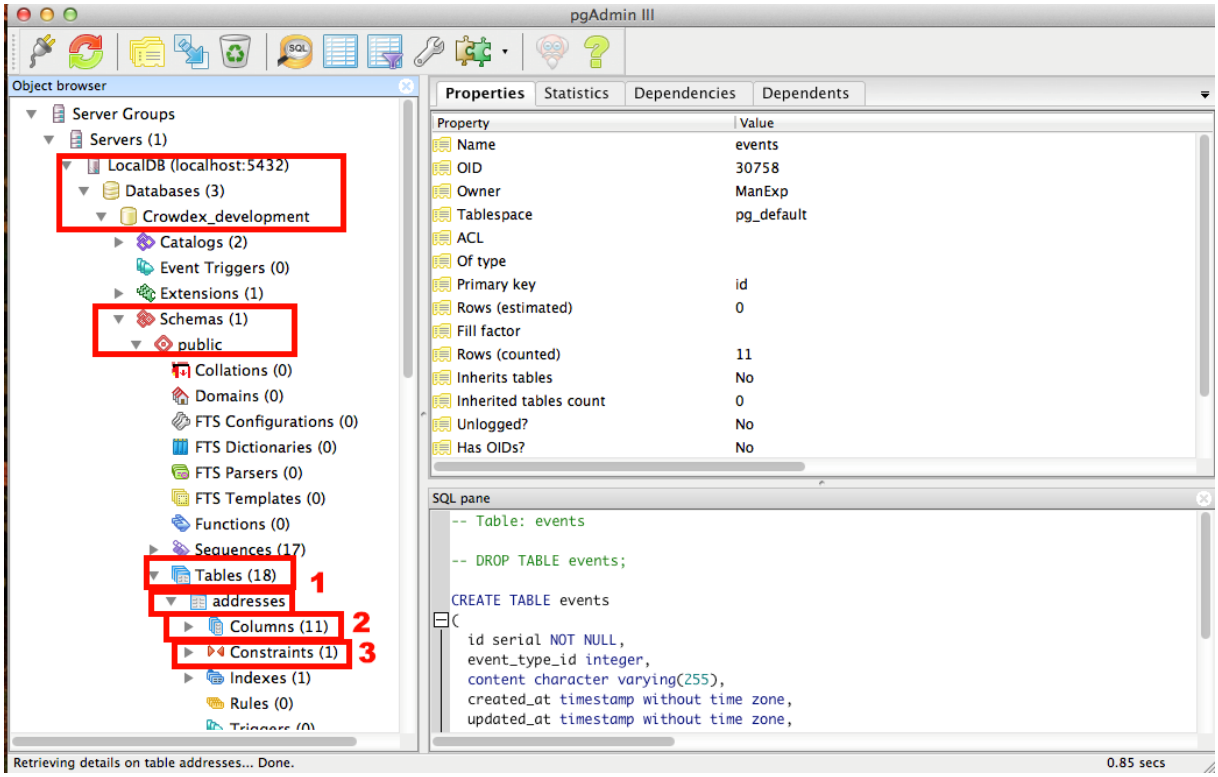



*Obrázok 5.1: Stiahnutie príslušnej vetvy*

### 5.1.3.2 Súlad so špecifikáciou

V rámci samotnej prehliadky je ako prvé potrebné skontrolovať súlad implementácie so špecifikáciou. Sústrediť sa pritom treba na dátový model – konkrétne názvy tabuliek, názvy stĺpcov a existenciu primárnych kľúčov. Túto kontrolu vykonajte prostredníctvom nástrojov *RubyMine* (ak je to potrebné) a *pgAdmin* (Obrázok 5.2) v týchto krokoch:

- vykonajte prípadné nevykonané zmeny nad Vaším dátovým modelom prostredníctvom migrácií:
  1. v nástroji *RubyMine* kliknite v kontextovom menu na *Tools*,
  2. vyberte položku *Run Rake Task...*,
  3. vyberte alebo zadajte príkaz *db:migrate*.
- v prípade, že mala databáza obsahovať aj predpripravené statické údaje (uvedené v špecifikácii), postupujte ako v predchádzajúcom bode, pričom namiesto príkazu *db:migrate* spusťte príkaz *db:seed*,
- otvorte si nástroj pgAdmin a prihláste sa s Vašimi prihlasovacími údajmi,
- zvoľte si databázový server (zvyčajne *LocalDB*), databázu *Crowdex_development*, rozbaľte príslušné schémy kliknutím na *Schemas* a vyberte schému *public*,
- kliknite na položku T*ables*, čím si si zobrazíte všetky tabuľky, a postupne:
  1. skontrolujte, či sa v databáze nachádzajú tie tabuľky, ktoré boli uvedené v špecifikácií a najmä, či ich názvy sú v súlade so špecifikáciou,
  2. kliknutím na príslušnú tabuľku v zozname zobrazíte ďalšie možnosti – vyberte možnosť *columns* a skontrolujte, či názvy stĺpcov sú v súlade so špecifikáciou,
  3. kliknutím na možnosť *constraints* môžete overiť existenciu primárneho kľúča – musí existovať pre každú tabuľku.




*Obrázok 5.2: Kontrola dátového modelu v nástroji pgAdmin*

### 5.1.3.3 Čistý kód

Po kontrole súladu so špecifikáciou treba pristúpiť ku kontrole čistoty kódu. Táto sa vykonáva v programovacom prostredí *RubyMine*. Pri tejto kontrole sa treba zamerať najmä na body uvedené v tabuľke č. 1, no všímať si treba aj prípadné nedodržanie konvencií uvedených v metodike pre písanie zdrojového kódu. V tabuľke je uvedené poradové číslo, ďalej tzv. pach – problematické miesto v zdrojovom kóde, ktoré vyžaduje pozornosť – podmienky použitia a náprava. Náprava je realizovaná dvomi spôsobmi:

- odstránenie zodpovednou osobou, ak je táto náprava rýchla a neexistuje priestor pre vznik chyby,
- označenie pachu v zdrojovom kóde zodpovednou osobou vo forme komentáru so skratkou *TODO* – zoznam *TODO* komentárov eviduje prostredie *RubyMine*. Nápravu následne vykoná riešiteľ.

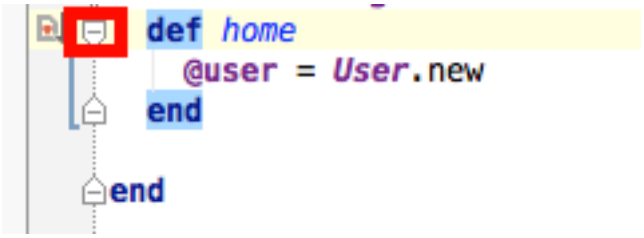
| Por. číslo | Názov | Podmienky použitia | Náprava |
|---|---|---|---|
| **1.** | Príliš dlhý riadok v kóde | Riadok presahujúci deliacu čiaru (120 znakov) v nástroji Rubymine. | ZO: zalomte príslušný riadok |
| **2.** | Dlhý názov metódy | Názov metódy má dĺžku viac ako 25 znakov. | TODO metoda s dlhym nazvom |
| **3.** | Metóda s veľkým počtom parametrov | Metóda obsahuje viac ako 3 parametre. | TODO metoda s velkym poctom parametrov |
| **4.** | Príliš dlhá metóda | Metóda pozostáva z viac | TODO dlha |

| Por. číslo | Názov | Podmienky použitia | Náprava |
|---|---|---|---|
| | | ako 10 riadkov kódu. | metoda |
| **5.** | Zložitá podmienka | Podmienka obsahuje aspoň 3 logické spojky. | TODO zlozita podmienka |
| **6.** | Metóda s logickým parametrom | Metóda obsahuje aspoň 1 logický (*boolean*) parameter. | TODO metoda s logickym parametrom |
| **7.** | Duplikovaný kód | Rovnaké alebo podobné časti kódu na viacerých miestach v zdrojovom kóde. | Všade, kde sa vyskytuje duplikát: TODO duplikovany kod |
| **8.** | Zakomentovaný zdrojový kód | Zdrojový kód obsahuje zakomentované časti. | ZO: odstráňte zakomentovaný kód |
| **9.** | Zdokumentovanie metód | Metóda neobsahuje dokumentačný komentár alebo je tento v nesprávnom formáte. Formát určuje metodika pre dokumentovanie zdrojového kódu. | TODO dokumentacia |
| **10.** | Komentáre problematických častí kódu | Zložité a problematické časti kódu neobsahujú komentáre. | TODO chybajuci komentar |
| **11.** | Neznáme konštanty | V kóde sa vyskytujú nepomenované konštanty. | TODO – neznama konstanta |

*Tabuľka 5.1: Problematické časti zdrojového kódu*

V prípade kontroly nedostatkov č. 2, 3 a 6 odporúčame skryť telá metód kliknutím na trojuholník v červenom rámiku (Obrázok 5.3).

V prípade nesúladu s konvenciami pre písanie zdrojového kódu treba dané miesto označiť ako *TODO konvencia*.




*Obrázok 5.3: Skrytie tela metódy vo vývojovom prostredí RubyMine*

#### 5.1.3.4 Funkčný kód

Po kontrole čistoty kódu je potrebné skontrolovať aj jeho funkčnosť za účelom odhalenia prípadných chýb. Kontrolu funkčnosti vykonajte všetkými spôsobmi, ktoré povaha úlohy umožňuje – za úplné minimum sa považuje vykonanie kontroly funkčnosti cez zdrojový kód.

**Kontrola funkčnosti cez vizuálne testovanie**

V rámci špecifikácie sú uvedené akceptačné testovacie prípady. Pokiaľ to charakter úlohy dovoľuje (boli zrealizované príslušné *views*), je potrebné tieto scenáre vizuálne overiť na zrealizovanej implementácii v nasledujúcich krokoch:

- spusťte si na Vašom počítači Rails server:
  - v nástroji *RubyMine* kliknite v kontextovom menu na položku *Run*,
  - kliknite na položku *Run...* a vyberte položku *Development: Crowdex*.
- po naštartovaní servera choďte na adresu *http://localhost:3000/,*
- vykonajte scenáre uvedené v špecifikácii a všímajte si reálny stav s očakávaným stavom.

**Kontrola funkčnosti cez zdrojový kód**

Ďalší spôsob odhalenia chýb je skontrolovať vybrané časti zdrojového kódu, ktoré sú náchylné na chybovosť. Postupne skontrolujte nasledujúce časti kódu:

- cykly:
  - sústreďte sa pritom najmä na cykly, ktorých telo obsahuje viac ako 4 riadky.
- riadiace podmienky cyklu:
  - dôraz dajte najmä na okrajové hodnoty podmienky a podmienky spĺňajúce bod č. 5 v tabuľke č. 1.
- vetvenie:
  - skontrolujte logiku vetviacich podmienok (špeciálne v prípade, že podmienka spĺňa bod č. 5 v Tabuľka ),
  - skontrolujte logiku vetvenia – najmä či môže dôjsť k vykonaniu každej z vetiev.
- zamerajte svoju pozornosť na premenné reprezentujúce inštancie triedy, nad ktorými sa vykonáva prístup k atribútom:
  - skontrolujte hlavne fakt, či je ošetrená situácia, keď je inštancia triedy *nil*.

**Kontrola funkčnosti cez automatizované testy**

V prípade, že k danej úlohe boli vytvorené automatizované testy, je možné funkčnosť kódu overiť ich spustením. Spustenie automatizovaných testov vykonajte spôsobom, ktorý nájdete v metodike pre písanie testov na obr. 6: Overenie implementovanej funkcionality.

#### 5.1.3.5 Ukončenie prehliadky

**Kontakt s riešiteľmi**

Po skončení prehliadky je potrebné kontaktovať riešiteľa, resp. riešiteľov úlohy a oboznámiť ich s výsledkom prehliadky. V prípade, že pri prehliadke neboli odhalené žiadne

nedostatky, nie je nutné tento bod vykonať. Riešiteľov kontaktujte prostredníctvom sociálnej siete Facebook vytvorením skupinového chatu.

**Náprava nedostatkov**

V prípade nájdených nedostatkov je potrebné s riešiteľmi vykonať ich nápravu.

Náprava nedostatkov bude vykonaná nasledujúcim spôsobom:

- v prípade odhalenia nesúladu špecifikácie a implementácie, alebo odhalenia chýb počas kontroly funkčnosti opísanej v predchádzajúcej kapitole sa vyžadujte od riešiteľa vykonanie bezodkladnej nápravy, ak to povoľujú časové možnosti šprintu. V opačnom prípade bude v systéme *YouTrack* do ďalšieho šprintu zaevidovaná úloha typu *bug* v súlade s metodikou pre nahlasovanie chýb,
- pri nedostatkoch č. 1 a č. 8 v Tabuľka  len informujte riešiteľa o tejto skutočnosti a vykonanej náprave,
- pri zvyšných nedostatkoch zostáva záznamom o nich informácia v zdrojovom kóde o *TODO* úlohe. V prípade, že budú od riešiteľa uvedené relevantné dôvody, ktoré ho viedli k danému nedostatku, odstráňte príslušné *TODO* úlohy. V opačnom prípade sa dohodnite s riešiteľom, kedy dané nedostatky odstráni. V prípade, že bude všeobecný súhlas celého tímu spolu s vlastníkom produktu na zrealizovaní špeciálneho šprintu určeného na refaktorizáciu zdrojového kódu, budú tieto nedostatky odstránené v danom šprinte.

**Zmena stavu úlohy**

Po náprave nedostatkov, resp. po dohode na ich odstránení v ďalšom šprinte, zmeňte stav úlohy na *Verified*. Prehliadka zdrojového kódu sa týmto končí.

# 6 Manažment kvality

Manažment kvality zodpovedá za kvalitu vytvoreného kódu. Je mimoriadne dôležité mať nielen fungujúci kód, ale aj kvalitný kód. Keďže na vývoji sa podieľajú všetci členovia tímu, nutne musíme kód udržiavať dobre štruktúrovaný a prehľadný. V súčasnosti máme vytvorené metodiky na dodržiavanie základných princípov, ktoré nám zjednodušia v budúcnosti rozširovanie funkcionalít a pomôžu udržať kód funkčný.

## 6.1 Metodika – Písanie kódu

### 6.1.1 Úvod

Metodika popisuje základné princípy ako správne písať kód, konkrétne v programovacom rámci *Ruby on Rails*. Hlavným cieľom je zjednotenie štýlu písania kódu, používanie osvedčených postupov. Dodržiavanie týchto praktík nám pomôže udržať kód prehľadný, zrozumiteľný. Keďže problematika je rozsiahla a samotný rámec *Ruby on Rails* má množstvo zaužívaných konvencií, preto táto metodika popisuje vybrané postupy a situácie, ktoré sa môžu najčastejšie vyskytnúť a nie sú úplne štandardné.

Metodika poslúži celému tímu pretože všetci členovia sa podieľajú na vývoji. Momentálne nenadväzuje na žiadnu inú metodiku. Vychádza z odporúčaných postupov, ktoré postupne vytvorili samotný programátori používajúci rámec *Ruby on Rails*. Metodika opisuje vývoj aplikácie v prostredí nástroja *RubyMine*.

### 6.1.2 Zoznam nadväzujúcich metodík a dokumentov

1. https://github.com/bbatsov/ruby-style-guide
2. http://guides.rubyonrails.org/active_record_basics.html#naming-conventions
3. http://www.toptal.com/ruby-on-rails/top-10-mistakes-that-rails-programmers-make
4. http://rubyglasses.blogspot.sk/2007/08/actsasgoodstyle.html
5. https://github.com/styleguide/ruby?utm_source=rubyweekly&utm_medium=email

### 6.1.3 Vymedzenie pojmov

- ***Session*** – permanentné sieťové spojenie medzi klientom a serverom.
- ***Cookie*** – označuje malé množstvo dát, ktoré server pošle webovému prehliadaču a ten ich uloží na počítači používateľa.

### 6.1.4 Postupy

#### 6.1.4.1 Rozdelenie funkcií do tried

Keď vytvárate alebo dopĺňate funkcionalitu je dôležité zachovávať nasledujúce princípy.

**Controller**

Dbajte na to aby obsahoval len tieto typy aplikačnej logiky:

- **Spracovanie *Session* a *Cookie*** – môže zahŕňať autentifikáciu a autorizáciu, alebo spracovanie *cookie*.
- **Práca s modelom** – logika, ktorá zahŕňa vyhľadanie správneho modelu pomocou parametrov z požiadavky, následne zobrazenie výsledku.
- **Správa parametrov požiadaviek** – spracovanie parametrov požiadavky a následne vyvolanie odpovedajúcej funkcii v modeli, ktorá ich ďalej spracuje (aktualizácia, uloženie).
- **Zobrazenie/presmerovanie –** zobrazenie výsledkov (html, xml, json) alebo presmerovanie.

**View**

- Snažte sa vyhnúť použitiu vetvených podmienok, aby ste zachovali *view* prehľadný.
- Vždy keď sa nejaká časť kódu opakuje používajte *Partials* a *Layouts.*

**Model**

V modeli môžete spracovávať len nasledujúcu aplikačnú logiku:

- **Nastavenie *ActiveRecord*** – relácie, validácia atribútov modelu.
- **Jednoduché metódy** slúžiace na aktualizáciu údajov v databáze alebo uloženie, ktoré by sa mohli zlúčiť s využitím parametrov zlúčte do jednej metódy.
- **Zložité dopyty**, napríklad tie, ktoré zahrňujú viac ako jednoduchý *find* vytvorte ako metódy v modeli. Nepoužívajte zložitejšie dopyty mimo modelu.
- **Obmedzenie prístupu** – skryť interné informácie modelu.

**Helpers**

Keď pri vytvárate nové funkcionality a neviete ich zaradiť do predchádzajúcich kategórií môžete vytvoriť triedu odvodenú od *Helper*, kde zoskupíte podobné funkcie a podľa toho túto triedu pomenujete.

#### 6.1.4.2 Tvorba názvov

**Premenné**

V Ruby môžu premenné obsahovať dáta rôzneho typu. Rozsah ich použitia určuje ich názov. Názov definujte v anglickom jazyku tak aby vysvetľoval význam, pričom slová obsahujú iba malé písmená a na spájanie použite podtržník. Preto pri vytváraní nových premenných postupujte nasledovne:

- **Premenná inštancie –** názov začína znakom `@` a pokračuje názvom rovnako ako lokálna premenná. Je dôležité ich používať len v prípade posielania dát z *Controller* do *View*, inak ich nepoužívajte.

- **Premenné triedy** – názov začína dvomi znakmi @@ a za nimi nasleduje meno. Snažte sa vyhnúť použitiu tohto typu premenných.
- **Konštanty** - Názov konštanty obsahuje len veľké písmená a slová sa oddeľujú pomocou podtržníku.

```
# nesprávny názov lokálnej premennej – nevysvetľuje význam
d = 5

# správne vytvorené názvy
# lokálna premenná
current_user = User.find(1)

# premenná inštancie
@all_users = User.all

# premenná triedy
@@user = user

# konštanta
EXPERIMENT_CREATE = 'new experiment'
```

**Metódy**

V názve metódy môžeme použiť špeciálne znaky na konci ako *?, !* . Otáznik na konci použite vtedy, keď metóda vracia booleovskú hodnotu napríklad User#is_logged_in?.

Názvy potencionálne nebezpečných metód zakončite znakom *!*. Napríklad vtedy ak metóda mení objekt, pre ktorý je definovaná. Nesmiete vytvoriť metódu s *!* na konci ak nie je definovaná metóda bez *!*.

### 6.1.4.3 Formátovanie zdrojového kódu

Na formátovanie kódu v prostredí RubyMine používajte klávesové skratky. ***Ctrl+Alt+L*** pre automatické formátovanie kódu a ***Ctrl+Alt+I*** pre automatické odsadenie riadkov. Ak pred použitím týchto skratiek neoznačíte presnú časť kódu, ktorú treba formátovať RubyMine automaticky vykoná úpravy pre celý súbor.

Nie všetko dokážu vyriešiť predchádzajúce skratky. Pre nasledujúce situácie použite tieto pravidlá:

- Keď vytvárate metódu s preddefinovanými parametrami vložte medzery okolo =.

```
def some_method(arg1 = :default, arg2 = nil, arg3 = [])
  # do something...
end
```

- V metódach používajte na oddelenie logických celkov jeden prázdny riadok.
- Ak vytvoríte riadok, ktorý má viac ako 100 znakov, musíte ho rozdeliť.

- Pri zreťazenom volaní metód, ktoré presiahne rozsah jedného riadku postupujte tým spôsobom, že za poslednou metódou v prvom riadku necháte .

```
User.find(1).followers.first.
  is_active?
```

- Ak voláte metódu, ktorá má veľa parametrov a ich dĺžka prekročí maximálnu dĺžku jedného riadku tak uveďte každý parameter na nový riadok. Parametre odsaďte od začiatku volania metódy dvomi medzerami.

```
params.require(:experiment).permit(
  :user_id,
  :string_hash,
  :slug,
  :title,
  :desc,
  :budget,
  :allow_comments,
  :delete_flag,
)
```

**Triedy**

Keď vytvárate triedu alebo ju dopĺňate o nové metódy, atribúty je nutné aby ste dodržiavali jednotnú štruktúru tried. Ako prvé po hlavičke triedy doplňte triedy, ktoré používate v danej triede. V modeloch je nutné ako prvé deklarovať vzťahy modelu k ostatným. Potom doplňte definície konštánt a hneď za nimi nasledujú atribúty triedy. Potom definujte makrá pre danú triedu a za nimi nasledujú verejné metódy triedy, potom chránené metódy a na koniec súkromné metódy triedy. Na oddelenie jednotlivých definícií používajte jeden prázdny riadok.

```
class UsersController < ApplicationController
  # include ide ako prvý
  include NotificationHelper

  # konštanty
  EXPERIMENT_CREATE = 'new experiment'

  # atribúty
  attr_reader :education_lvl

  # nasledujú makrá
  before_action :correct_user, only: [:edit, :update]
  before_action :admin_user, only: :destroy

  # metódy triedy
  def index
    @users = User.paginate(page: params[:page])
  end

  # chránené a súkromné metódy su uvedené na konci
```

```
  protected

  def admin_user
    redirect_to(root_url) unless current_user.admin?
  end

  private

  def user_params
    params.require(:user).permit(:name, :email, :password,
                                 :password_confirmation)
  end
end
```

```
class Micropost < ActiveRecord::Base
  belongs_to :user
  has_many :comments
  # ... nasleduje zvyšok modelu rovnako ako v iných triedach
```

#### 6.1.4.4 Syntax

**Podmienky**

- Keď vytvárate podmienku, ktorá kontroluje rozsah používajte metódu *Comparable#between* namiesto vytvárania viacnásobných podmienok.

```
# nesprávny spôsob
do_something if User.age > 18 && User.age < 35

# správny spôsob
do_something if User.age.between?(18, 35)
```

- Nepoužívajte *and* a or keď vytvárate booleovské výrazy a tak isto ani na riadenie toku, ale použite tieto znaky && a ||.
- V podmienkach nepoužívať zbytočne *then* aj keď má časť za podmienkov viac riadkov.

```
if is_logged_in?
  # do_something
  # do_something_else
end
```

- Používajte ternárny operator ?: keď je to možné, namiesto konštrukcie *if/then/else/end*.

```
# nesprávny spôsob
current_user = if is_active then User.find(id) else nil end

# správny spôsob
current_user = is_active ? User.find(id) : nil
```

- Používajte podmienky v jednom riadku všade, kde je to možné.

```
# nesprávny spôsob
if is_logged_in?
  @experiment = Experiment.new
end

# správny spôsob
@experiment = Experiment.new if is_logged_in?
```

- V podmienkach používajte *unless* namiesto záporných podmienok. Nepoužívajte však *unless* spolu s else vetvou. V tomto prípade obrátte podmienku a použite ako prvý pozitívny prípad.

```
# správny spôsob
redirect_to root_url unless is_logged_in?

# nesprávny spôsob
unless is_logged_in?
  redirect_to root_url
else
  @experiment = Experiment.new
end

# správny spôsob
if is_logged_in?
  @experiment = Experiment.new
else
  @experiment = Experiment.new
end
```

- Nepoužívajte zátvorky okolo podmienok.
- Nepoužívajte operáciu priradenia v podmienkach.

**Ostatné**

- Na inicializáciu premenných, pri ktorých je možnosť, že už boli inicializované predtým používajte ||=. Inicializácia sa v tomto prípade vykoná iba ak je hodnota premennej *false* alebo *nil*. Nepoužívať v žiadnom prípade na inicializáciu booleovských premenných.

```
@current_user ||= User.find_by(id: session[:user_id])
```

- V triedach nepoužívajte *self* keď to nie je nutné.
- Pri definovaní validačných podmienok sa vyhnite opakovaniu tých istých podmienok.

```
# nesprávny spôsob
validates_presence_of :field_one
validates_presence_of :field_two
validates_presence_of :field_three
validates_presence_of :field_four

# správny spôsob
validates_presence_of :field_one, :field_two, :field_three, :field_four
```

- Ak budete musieť použiť SQL v podmienkach pre vyhľadávanie alebo čokoľvek iné, argumenty pre túto podmienku vždy vkladajte ako premenné.

```
User.where(["name = ? and email = ?", name, email])
```

Blok *{}* používajte ked ide o blok kódu v jednom riadku inak v prípade, že tento blok obsahuje viac riadkov kódu použite blok *do-end*.

**Reťazce**

- Keď potrebujete skladať reťazec z niekoľkých premenných použite spôsob interpolácie reťazcov namiesto spájania.

```
# nesprávny spôsob
user_name = user.first_name + ' ' + user.last_name

# správny spôsob
user_name = "#{user.first_name} #{user.last_name}"
```

- Keď nepotrebujete použiť interpoláciu reťazcov alebo niektorý zo špeciálnych znakov použite jednoduché úvodzovky v opačnom prípade dvojité úvodzovky.

```
name = 'Jan Slovak'
```

- Pri vytváraní veľkých reťazcov nepoužívajte na skladanie reťazca znakov metódu *String#+*, ale použite metódu *String#<<*.

```
table << '<table class="users">'

users.each { |user| table << "<tr><td>#{user.name}</td></tr>" }
```

## 6.2 Metodika – Návrh a zobrazovanie formulárov

Predmetom tejto metodiky je definícia implementácie dizajnových prvkov tímovej webovej aplikácie. Táto metodika sa zaoberá definovaním pravidiel a princípov, ktoré budú členia tímu uplatňovať pri tvorbe formulárov v celom systéme. Kladie sa pritom dôraz na dizajnovú stránku, a to najmä štrukturálne usporiadanie prvkov v jazyku *HTML* s využitím rámca *Ruby on Rails*. Cieľom tejto metodiky nie je objasniť, ako spracovávať výstupy z formulárov a reagovať na chyby spôsobené nekorektným vyplnením takýchto formulárov. Metodika tiež striktne nestanovuje, aké konkrétne formulárové prvky sú adekvátne na zobrazenie určitej požadovanej informácie. Predpokladá sa základná znalosť programovacieho jazyka Ruby a základná znalosť značkovacieho jazyka *HTML*.

Metodika vychádza zo štandardov webového rámca *Boostrap*[4] a základných pravidiel pre tvorbu formulárov[5] využívaných vo webovom rámci *Ruby on Rails*. Základným nástrojom pre jej realizáciu je ľubovoľný **editor kódu Ruby a HTML**. Určená je pritom pre každého člena tímu.

### 6.2.1 Zoznam nadväzujúcich metodík a dokumentov

1. Problematika tvorby formulárov v Rails aplikácii - http://guides.rubyonrails.org/form_helpers.html

### 6.2.2 Vymedzenie pojmov a skratiek

- ***Asynchrónne volanie*** – volanie, ktoré sa vykoná bez nutnosti opätovného úplného načítania a vykreslenia webovej stránky.
- ***Entita*** – v kontexte tejto metodiky sa pod entitou myslí entita databázového modelu.
- ***Náhradný popis*** (angl. *placeholder*) – popisuje textový formulárový prvok formou krátkeho pomocného textu; popis sa nachádza na mieste, kde sa očakáva textový vstup zadaný používateľom.
- ***Značka*** – otváracie (a koncové) pomenovanie prvku zo značkovacieho jazyka HTML.

### 6.2.3 Postupy

Základným prvkom, prostredníctvom ktorého vykreslíte v kóde formulár je pomocná metóda `form_for`. Pri tejto metóde budeme okrem základných parametrov využívať aj pomocný atribút `role` s hodnotou `form`.

V prípade, že vytvárame formulár, ktorého odosielanie sa spracúva cez **asynchrónne volanie**, ďalším potrebným parametrom je `remote` s hodnotou `true`. Pokiaľ to explicitne nevyžaduje špecifikácia, v aplikácii používajte primárne formuláre spracúvajúce vzdialené volania. Výnimkou môžu byť nasledujúce situácie:

---

[4] http://www.getbootstrap.com

[5] http://guides.rubyonrails.org/form_helpers.html

- formulár určený na autentifikáciu používateľa,
- formulár spracúvajúci rozsiahle množstvo údajov (napr. textový editor),
- spracovanie finančných alebo akýchkoľvek iných transakcií využívajúcich služby tretích strán.

V týchto situáciách sa rozhodnutie ponecháva na autorovi formulára.

#### 6.2.3.1 Základná štruktúra formulára

Vychádzajúc z predchádzajúcich pravidiel, základná štruktúra každého formulára v aplikácii bude nasledovná:

```
<%= form_for(<ENTITA>, html: { role: "form" }) do |f| %>
  ...prvky formulára...
<% end %>
```

Formulár spracúvajúci sa cez vzdialené volanie bude rozšírený o parameter `remote`:

```
<%= form_for(<ENTITA>, html: { role: "form", remote: true }) do |f| %>
  ...prvky formulára...
<% end %>
```

kde `<ENTITA>` je zodpovedajúca entita alebo pomenovanie formulára.

#### 6.2.3.2 Interakcia s používateľom

Pri spracovávaní formulára dochádza k interakcii s používateľom, a preto dodržiavajte nasledujúce pravidlá:

- v prípade, že si spracovanie formulára v aplikácii vyžaduje podporu zapnutého Javascriptu, informujte používateľa o tejto povinnosti pred odoslaním samotného formulára,
- v prípade nesprávneho odoslania formulára informujte používateľa o chybe, a to jedným z nasledujúcich spôsobov:
    - výpisom chyby pri príslušnom problematickom prvku,
    - výpisom zoznamu chýb súhrnne pre celý formulár,
    - zvýraznením príslušného problematického prvku.

Zobrazovanie súhrnných chýb pre celý formulár realizujte prostredníctvom nasledujúceho kódu:

```
<% if <ENTITA>.errors.any? %>
  <div id="error_explanation">
     <h2><%= pluralize(@<ENTITA>.errors.count, "error") %> prohibited
this <ENTITA> from being saved:</h2>
     <ul>
      <% @<ENTITA>.errors.full_messages.each do |message| %>
      <li><%= message %></li>
      <% end %>
    </ul>
  </div>
<% end %>
```

kde <ENTITA> je entita, ktorá sa prostredníctvom formulára mení. Výpis chýb pri príslušnom problematickom prvku zahŕňajú ukážky kódov v kapitole ***Prvky formulára*.**

Pri **zostavovaní chybových textov pre formuláre** sa riaďte nasledujúcimi odporučeniami:

- každá chybová hláška musí mať pre používateľa nielen informatívnu hodnotu, ale cieľom je poskytnúť aj možnosť zotavenia z chyby vzniknutej nesprávnym vyplnením formulárového prvku,
- poskytnite informácie o tom, ktorý formulárový prvok bol nesprávne vyplnený,
- v popise chyby vždy uvádzajte názov korešpondujúci s popisným názvom formulárového prvku (angl. *label*).

Príklad:

| | |
|---|---|
| Popisný názov formulárového prvku: | Priezvisko |
| Korektná chybová hláška: | Ľutujeme, ale priezvisko môže obsahovať maximálne 30 znakov. |
| Nekorektná chybová hláška | Ľutujeme, ale vo formulári sa nachádza nesprávne vyplnený prvok. |

#### 6.2.3.3 Prvky formulára

Každý prvok formulára predstavuje samostatný ohraničený podkomponent. **Poradie**, v ktorom sú prvky vo formulári umiestnené nemá pevne stanovené pravidlá, postupujte preto v súlade s nasledujúcimi odporučeniami:

- všetky prvky zoraďte najmä podľa dôležitosti v kontexte konkrétneho formulára:
  - príklad: pri registrácii sa najskôr vypĺňa meno, priezvisko a až potom dátum narodenia,
- prvé v poradí uveďte najmä textové políčka, následne zaškrtávacie a výberové políčka,
- textové plochy umiestnite ako posledné,
- formulár uzatvorte uvedením potvrdzovacích tlačidiel, ktoré slúžia na odoslanie formulára alebo jeho zneplatnenie (obnovu),
- v prípade formulárov vyžadujúcich si informovanie / odsúhlasenie určitých podmienok (napríklad Podmienky používania, Ochrana osobných údajov) neuvádzajte túto informáciu za potvrdzovacími tlačidlami.

Pre každý formulárový prvok uveďte **popis** v rozsahu 1-3 slová:

- umiestnite ho vždy pred (teda naľavo alebo nad) samotný formulárový prvok,
- popis začnite veľkým začiatočným písmenom a zakončite dvojbodkou „:“.

**Príklady:** *Meno:*, *Priezvisko:*, *Dátum narodenia:*

#### 6.2.3.4 Definícia formulárových prvkov

Nasledujúce ukážky definujú štruktúru pre jednotlivé obsahy formulárových prvkov aplikácie.

Vo všeobecnosti platí:

- **<ID>** = identifikátor prvku formulára; v prípade, že o formulár určený pre vytvorenie/úpravu entity, použite názov príslušného atribútu,
- **<POPIS>** = popisný názov formulárového prvku.

Každý prvok musí byť obalený značkou `div` s triedou `form-control`:

```
<div class="form-group">
  ... formulárový prvok ...
</div>
```

**Textové políčko**

Textové políčko používajte pre jednoduché a stručné texty s dĺžkou maximálne 50 znakov.

```
<%= f.label <ID>, <POPIS> %>
<%= f.text_field <ID>, class: "form-control" %>
```

**Textové políčko využívajúce ikonku**

Špeciálnym prípadom je textové políčko, ktoré namiesto štandardného popisu využíva iba náhradný popis, ktorý sa nachádza priamo v textovom políčku. Súčasťou tohto prvku je aj ikonka. Textové políčka s ikonkou používajte iba vo formulároch určených na autentifikáciu.

```
<span class="form-icon form-icon-<ID_IKONKY>"></span>
<%= f.text_field <ID>, { class: "form-control", placeholder: <POPIS> } %>
```

Pre voľbu `<ID_IKONKY>` si vyberáte z nasledujúcich možností pre význam textového políčka:

- `email` – pre e-mailovú adresu,
- `lock` – pre zadávanie citlivých údajov (heslo, kód overenia),
- `user` – pre meno a priezvisko používateľa.

**Textová plocha**

Textovú plochu používajte pre dlhšie texty, pri ktorých sa očakáva dĺžka viac ako 50 znakov.

```
<%= f.label <ID>, <POPIS> %>
<%= f.text_area <ID>, class: "form-control" %>
```

**Výberové políčko (zoznam)**

Výberové políčko slúži na výber jednej z viacerých možností. Tieto možnosti sú pritom spočiatku skryté a používateľovi sa zobrazia až po kliknutí na políčko.

```
<%= f.select(<ID>, <MOŽNOSTI>, { }, { :class => “form-control” }) %>
```

`<MOŽNOSTI>` nahraďte poľom možností v tvare: `[ <POPIS>, <ID_MOŽNOSTI> ]` ako parameter metódy `options_for_select`.

Príklad:

```
<%= f.select(:pohlavie, options_for_select([ [„muž“, 1], [„žena“, 2] ]))
%>
```

**Výberové políčko (vymenovaním prvkov)**

Výberové políčko slúži na výber jednej z viacerých možností. Tieto možnosti sú pritom vždy viditeľné.

```
<%= f.label <ID>, <POPIS> %>
<div class="radio radio-default">
  <%= f.radio_button <ID>, <HODNOTA_1>, :checked => true %>
  <%= f.label <ID>, <POPIS_HODNOTY_1>, :value => <HODNOTA_1> %>
  ...
  <%= f.radio_button <ID>, <HODNOTA_N> %>
  <%= f.label <ID>, <POPIS_HODNOTY_N>, :value => <HODNOTA_N> %>
</div>
```

Pri rozhodovaní medzi výberovým políčkom formou vymenovania prvkov alebo formou zoznamu sa riaďte pravidlom:

- počet prvkov pri políčku formou *vymenovania prvkov* môže byť najviac 5 prvkov ($N \leq 5$), **inak** sa použite políčko formou *zoznamu*.

Toto pravidlo sa **neuplatňuje** v nasledujúcich situáciách:

- v kontexte formulárového prvku sa vyžaduje, aby boli všetky možnosti vždy viditeľné,
- formulárový prvok generuje systém na základe používateľských nastavení, ktoré si explicitne vyžadujú použitie zoznamu (príklad: používateľsky generovaný dotazník).

**Zaškrtávacie políčko**

Zaškrtávacie políčko používajte na potvrdenie výroku alebo výber aspoň jednej z viacerých možností.

```
<div class="checkbox checkbox-default">
  <%= f.check_box <ID> %>
  <%= f.label (<ID>, <POPIS>) %>
</div>
```

**Tlačidlo**

```
<%= f.submit <POPIS>, class: "btn btn-<TYP>" %>
```

Pri tlačidle rozlišujte `<TYP>` podľa dôležitosti akcie, ktorú tlačidlo vykonáva:

- `primary` – fatálna zmena v systéme - uloženie údajov, odstránenie položky,
- `success` – vyjadruje úspešný prechod do nového stavu,
    - tento typ používajte v prípade, že tlačidlo umožňuje vykonanie dvojstavovej činnosti (napr. sledovať používateľa, označiť položku ako „páči sa mi to“) a došlo k úspešnému prechodu do druhého stavu (napr. používateľ je sledovaný, položka sa používateľovi páči),
- `default` - presmerovanie na inú stránku, vymazanie formulára a všetky ostatné nešpecifikované aktivity.

#### 6.2.3.5 Zobrazenie chýb a pomocný text

**Pomocný text** používajte v prípade, že potrebujete používateľa informovať:

- o **ohraničeniach** vyplývajúcich z formulárového prvku (maximálna dĺžka, obmedzenia z hľadiska použitia číslic, písmen),
- o podrobnostiach súvisiacich s vyplnením formulárové prvku (príklad: „zaškrtnutím tejto možnosti sa vám zablokuje prístup k experimentu“).

```
<span class="help-block">
  <POMOCNÝ_TEXT>
</span>
```

Pre **výpis chýb** pre jednotlivé formulárové prvky použite nasledujúci kód:

```
<div class="form-control-errors" id="errors-<ID>" style="display:
none"><div class="arrow"></div><div class="errors"></div></div>
```

Uvedené kódy umiestnite pred uzatváraciu značku `</div>` s triedou `form-control`.

Príklad:

```
<div class=form-control>
  ... formulárový prvok s ID password...
  <span class="help-block">Heslo musí obsahovať 4 – 12 znakov.</span>
  <div class="form-control-errors" id="errors-password" style="display:
none"><div class="arrow"></div><div class="errors"></div></div>
</div>
```

# 7 Manažment rozsahu

Úlohou manažéra rozsahu v našom tíme je zabezpečiť dve základné požiadavky:

- Produkt musí obsahovať požadované vlastnosti, ktoré boli dohodnuté na začiatku projektu.
- Produkt by nemal obsahovať vlastnosti navyše, ktoré nie sú potrebné pre úspešné dokončenie produktu.

Aby boli splnené tieto požiadavky, v rannej fáze projektu bola dohodnutá funkcionalita a vlastnosti, ktoré by produkt mal obsahovať. Tieto požiadavky boli spísané do dokumentov *Funkcionality* a *Product Backlog*, podľa ktorého sa riadi aj plánovanie šprintov.

Na každom stretnutí, na ktorom sa diskutuje o ďalšej funkcionalite (typicky začiatky šprintov), ktorá bude do produktu implementovaná manažér rozsahu kontroluje, či nepresahuje dohodnutý rozsah projektu.

Po ukončení šprintu sa kontroluje, či boli vlastnosti produktu implementované v požadovanom rozsahu. Prípadné zistené nedostatky sú zaznamenané a riešené v niektorom z ďalších šprintov.

# 8 Manažment podpory vývoja a integrácie

Úloha manažéra podpory vývoja a integrácie pozostáva z manažmentu verzií, konfigurácií softvérového systému a integrácie použitých súčastí do celku. V projekte *Crowdex* je manažér priamo zodpovedný za nasledujúce úlohy:

- správa a údržba aplikačného servera
- správa a údržba produkčného prostredia
- nasadzovanie aplikácie na produkčný server
- monitorovanie aktuálneho stavu servera a aplikácie
- riešenie vzniknutých problémov počas behu aplikácie na serveri
- správa a údržba vývojového prostredia
- správa a údržba nástroja *YouTrack*[6] pre manažment úloh projektu
- správa a údržba repozitárov v nástroji *Github*[7] pre manažment verziovania zdrojového kódu
- spravovanie verzií produktu, použitých zásuvných modulov, knižníc a ich vzájomnej kompatibility s aplikáciou

Jeho úlohou je taktiež dohľad nad dodržiavaním metodiky pre verziovanie zdrojového kódu zvyšnými členmi tímu a udržiavať zdrojový kód v repozitári v konzistentnom stave.

## 8.1 Metodika – Verziovanie zdrojového kódu

### 8.1.1 Úvod

Táto metodika má za cieľ poukázať na problematiku manažmentu zdrojového kódu spolu s jeho verziovaním v projekte *Crowdex*. Dokument sa bližšie zameriava na proces vytvárania vetiev postupne od opisu vytvárania vetvy až po ukážku vo vybranom nástroji Git[8] a postup uvádzaný v editore *RubyMine*[9].

Metodika je určená pre každého člena v tíme, ktorý sa podieľa na implementácii kódu. Cieľom je ukázať a zaviesť jednotný postup pre zahájenie práce na rôznych vetvách počas implementácie, či už nových, existujúcich funkcií alebo ich opravy.

[6] https://www.jetbrains.com/youtrack/

[7] https://github.com/

[8] http://git-scm.com/

[9] https://www.jetbrains.com/ruby/

### 8.1.2 Zoznam nadväzujúcich metodík a dokumentov

Existuje taktiež oficiálna dokumentácia k práci s vetvami počas verziovania softvéru prostredníctvom nástroja Git, bližšie uvedená v podkapitolách 3.3 a 3.4[10].

### 8.1.3 Vymedzenie pojmov a skratiek

V metodike sú používané aj anglické pojmy spájajúce sa s verziovaním softvéru.

- ***Git*** – predstavuje voľne dostupný nástroj pre správu verzií
- ***Github*** – je to služba webová služba, ktorá umožňuje spravovanie gitovských repozitárov, pričom umožňuje ďalšie pokročilé funkcionality spravovania zdrojového kódu
- ***Branch*** (vetva) – predstavuje istú fázu vývoja zdrojového kódu v Gite
- ***Commit*** – predstavuje uloženie aktuálneho stavu zdrojového kódu vo vetve
- ***Pull request*** – predstavuje overený stav vetvy pripravený pre zlúčenie s inou vetvou
- ***Produkcia*** – stav aplikácie dostupnej pre používateľov

### 8.1.4 Postupy

V tejto kapitole sa nachádza uvedený postup, ako vytvárať vetvy a následne napojenie sa do vetvy. Jednotlivé kroky v metodike sú usporiadané postupne v časovej následnosti a sú záväzné pre každého člena tímu.

#### 8.1.4.1 Pravidlá pre názov novej vety

- formát pomenovania vetvy - [typ]-[názov funkcionality]
- typ - predstavuje typ implementovanej funkcionality, napr. *feature*, *fix*, *refactor*
- celý názov vetvy musí byť uvedený v angličtine
- názov musí byť čo najstručnejší, aby vhodne reprezentoval prácu, ktorá sa vykonáva na vetve, najlepšie 1-2 kľúčové slová, žiadne vety alebo súvetia
- názov vetvy obsahuje iba malé písmená
- oddeľovanie slov v názve pomocou spojovníkov

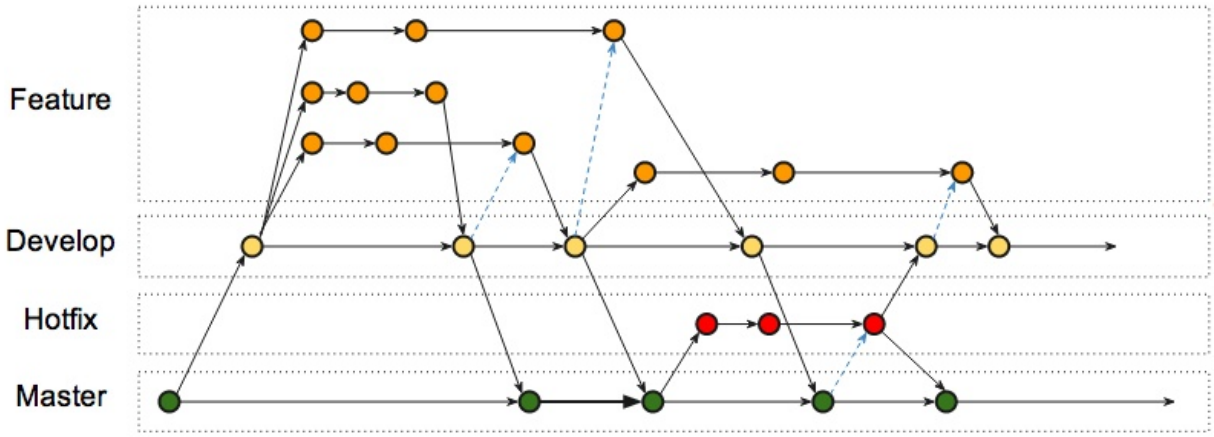
[10] http://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

#### 8.1.4.2 Typy vetiev

- development
    - vetva slúži ako hlavné prepojenie implementovaných funkcionalít a vetiev, ktoré sa s touto vetvou zlučujú
    - vetva vznikla oddelením od vetvy *master* a priebežne sa zlučuje s vetvou pre implementáciu funkcionality a zlučuje do vetvy *master,* resp. *staging*
- feature
    - ide o vetvu, kde prebieha implementácia novej funkcionality softvéru
    - názov vetvy pozostáva z kľúčových slov obsahujúcich danú funkcionalitu
    - vetva vzniká oddelením od vetvy *development* a zlučuje sa do vetvy *development*
- fix (bug, refactor)
    - táto vetva slúži na opravenie zdrojového kódu malého rozsahu, väčšinou ide o opravenie preklepov, drobných chýb alebo refaktorovanie zdrojového kódu
    - vetva vzniká oddelením od vetvy *development* alebo *master* a zaniká zlúčením do vetvy *development* alebo tiež do vetvy *master* pre priamu opravu chyby v produkcii
- shared
    - táto vetva vznikla odčlenením od *development* vetvy, pričom slúži na vytváranie commitov pre zdielané časti zdrojového kódu, ako sú migrácie databázy alebo knižnice
    - priebežne sa zlučuje do feature alebo fix vetiev
- staging
    - na tejto vetve prebieha testovanie aplikácie a novo pridaných alebo opravených funkcionalít
    - vetva sa zlučuje s vetvou *development* a po otestovaní funkcionality je zlúčená do vetvy *master*
- master
    - ide o vetvu, ktorá obsahuje zlúčené a prekontrolované a otestované commity z *development* a *fix* vetiev
    - táto vetva slúži pre nasadenie aplikácie do produkcie
    - vetva sa zlučuje s vetvou *staging*

### 8.1.4.3 Princíp vetvenia z existujúcej vetvy

Na základe predchádzajúceho rozdelenia typov vetiev je potrebné si pred samotným vytvorením novej vetvy rozmyslieť, ako sa bude vytvárať nová vetva. Treba zvoliť jednu z existujúcich vetiev ako východiskový základ novej vetvy. Po výbere a prepnutí na existujúcu vetvu sa môže oddeliť od tejto vetvy a vytvoriť úplne nová vetva, ktorá už bude mať základy zvolenej existujúcej vetvy. Napríklad pre implementovanie novej funkcionality sa treba prepnúť na existujúcu vetvu *develop* a následne vytvoriť novú vetvu, ktorá bude vychádzať z vetvy *develop*, pričom treba dodržať konvencie pre pomenovanie vetvy. Po vytvorení *commitov* v novej vetve a ukončení implementácie sa vytvorí *Pull Request* a dôjde k zlúčeniu novej vetvy do vetvy *develop*. Názorná ukážka vetvenia v *Gite* je zobrazená na Obrázok 8.1.

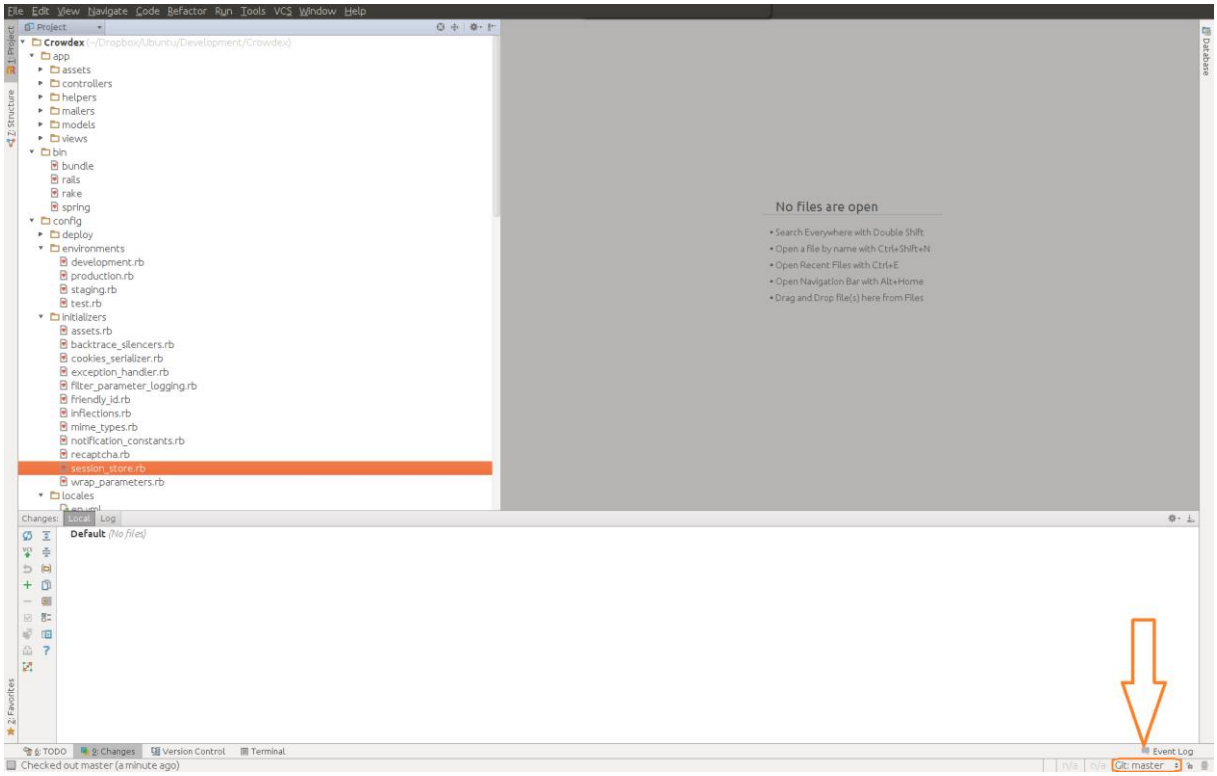



*Obrázok 8.1: Ukážka schémy vetvenia v Gite*[11]

### 8.1.4.4 Postup vytvorenia vetvy

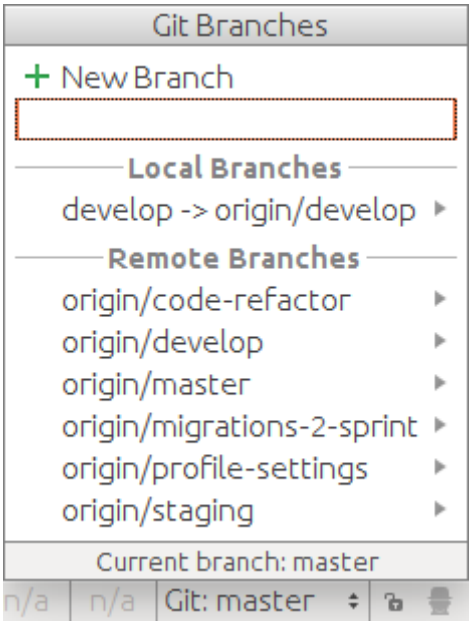
1. Kliknite do dolného pravého rohu pre zobrazenie zoznamu vetiev (Obrázok 8.2)

[11] https://blog.networld.to/963/the-beauty-of-git-and-how-it-affects-our-day-to-day-work



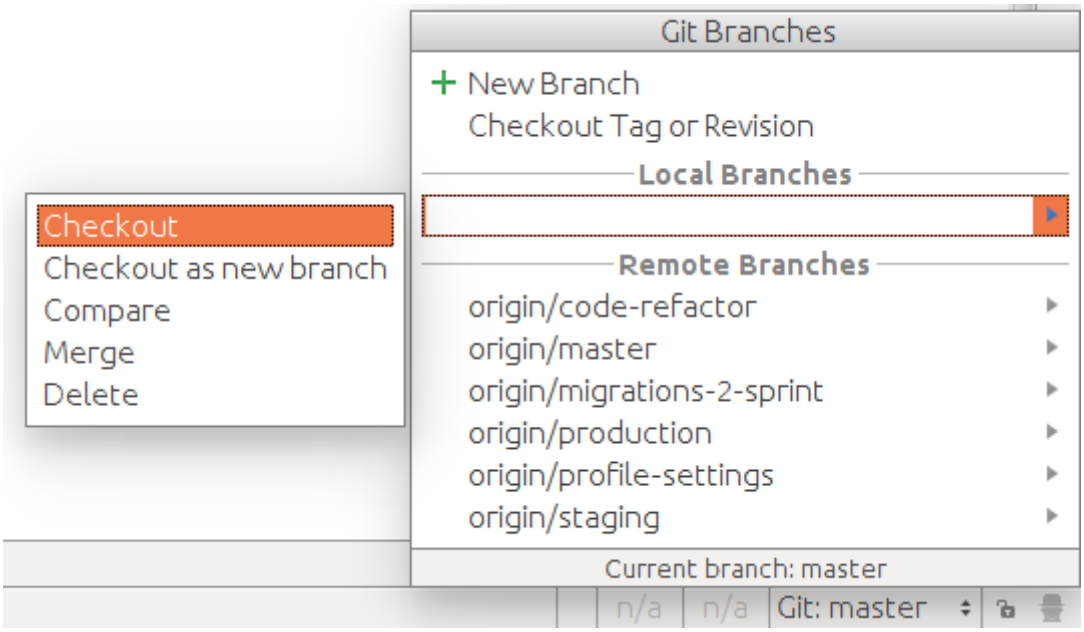

*Obrázok 8.2: Ukážka prostredia RubyMine*

2. Kliknite na jednu z existujúcich vetiev v zozname *Local* alebo *Remote Branches* pre voľbu východiskovej vetvy (Obrázok 8.3)



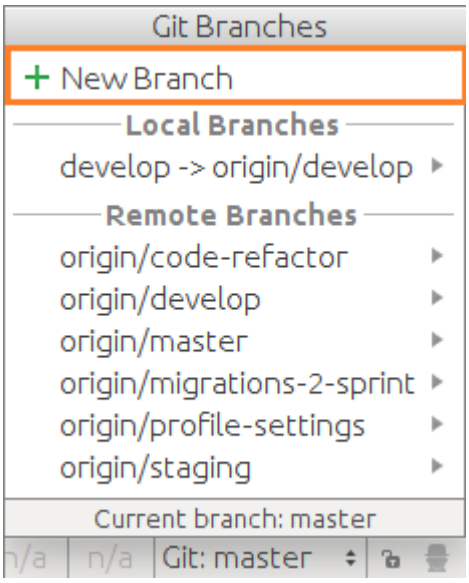

*Obrázok 8.3 - Zoznam existujúcich vetiev*

3. Zvoľte možnosť *Checkout* (Obrázok 8.4)



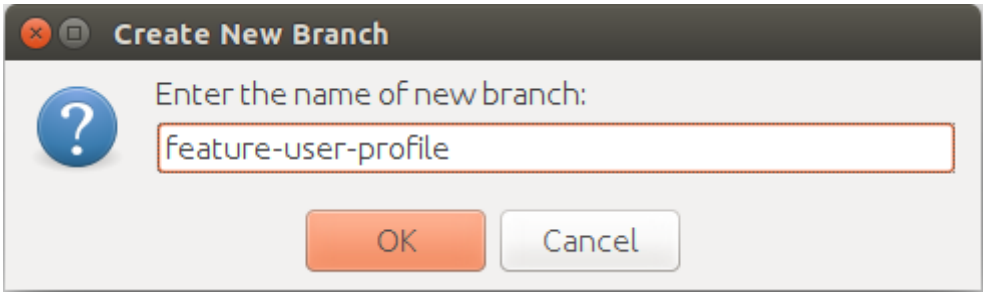

*Obrázok 8.4 - Výber možnosti pre napojenie na existujúcu vetvu*

4. Kliknite opäť do dolného pravého rohu (Obrázok 8.2)
5. Zobrazí sa zoznam vetiev (Obrázok 8.3)
6. Kliknite na prvý riadok na značku + *New Branch* pre vytvorenie novej vetvy (Obrázok 8.5)




*Obrázok 8.5 - Výber možnosti pre vytvorenie vetvy*

7. Zadajte názov vetvy podľa metodiky (Obrázok 8.6)




*Obrázok 8.6 – Formulár pre názov novej vetvy*

8. Potvrďte vytvorenie novej vetvy

# 9 Manažment komunikácie

Manažér komunikácie zodpovedá za spôsoby komunikácie medzi jednotlivými členmi tímu. Zabezpečuje, aby bola komunikácia možná, čo najefektívnejšia a aby sa všetky informácie dostali ku všetkým členom tímu.

V rámci tímu bola hneď po jeho sformovaní vytvorená tajná skupina na sociálnej sieti *Facebook*. Do tejto skupiny boli pridaní všetci členovia tímu, pričom po prvých pár stretnutiach bol do nej pozvaný aj vedúci projektu, Ing. Michal Kompan, PhD. Skupina sa využíva primárne pre rýchlu komunikáciu medzi jednotlivými členmi tímu. Komunikujú sa hlavne zložitejšie otázky, ktoré chceme mať v rámci komunikácie aj nejakým spôsobom archivované. Keďže sa v skupine nachádza aj vedúci projektu, môže sa vyjadrovať ku všetkým veciam, ktoré v nej rozoberáme, prípadne sme aj my schopní ho priamo osloviť a v relatívne rýchlom čase dostať odpoveď.

Okrem skupiny využívame na sociálnej sieti *Facebook* aj skupinový chat. Tento je používaný na riešenie menších a nie tak významných problémov, prípadne sa cez neho zdieľajú krátke informácie, ktorých archivovanie je pre nás nepodstatné. Počet správ odoslaných cez tento chat jednotlivými členmi tímu činí v priemere 20 správ za hodinu (počítame len hodiny počas dňa). Tento druh komunikácie pre nás reprezentuje komunikáciu v reálnom čase, keďže mnohí členovia tímu sú na sociálnej siete prístupní takmer počas celého dňa. Inými slovami, spravidla nenastáva situácia, že ak niektorý z členov tímu napíše do chatu otázku, čakal by na odpoveď, resp. názor od iného člena tímu dlhšie ako 10 minút.

Ďalší kanál využívaný na komunikáciu je *Google Drive*, kde sú zdieľané všetky dokumenty členov tímu. Komunikácia tu prebieha dvomi spôsobmi:

1. komunikácia pri tvorbe určitého dokumentu prostredníctvom chatu, ktorý *Google Drive* v dokumentoch poskytuje
2. komunikácia prostredníctvom poznámok (komentárov) v jednotlivých dokumentoch

Tento druh komunikácie využívame primárne pri skupinovej tvorbe dokumentov. Konkrétne chat využívame najčastejšie pri tvorbe dokumentov, za vytvorenie ktorých sú zodpovední viacerí členovia tímu (napr. retrospektíva). V rámci chatu je komunikované napr. rozdelenie práce na dokumente, okamžité pripomienky a pod. Naopak komentáre využívame na vyjadrenie návrhov na zmeny či úpravy na dokumenty, ktoré vytvorila jedna osoba (napr. špecifikácie).

Okrem internetu využívame na komunikáciu aj formálne a neformálne stretnutia.

Pri formálnych stretnutiach s vedúcim prebieha komunikácia primárne o funkcionalite, aktuálnom stave daného šprintu a prípadných problémoch, ktorí riešitelia v danom šprinte majú. Každý člen tímu sa každý týždeň vyjadruje k stavu riešenej úlohy (čo urobil, čo neurobil, s prípadnými dôvodmi). Zároveň sa na začiatku každého šprintu dohodneme na zložitosti jednotlivých úloh. Komunikácia pri tejto dohode prebieha v prvom kroku prostredníctvom aplikácie *Scrum Poker*, kde každý člen tímu vyjadrí svoj individuálny názor na zložitosť danej úlohy. Následne sa členovia tímu, ktorí mali najdlhší, resp. najkratší odhad

vyjadria k dôvodom, ktoré ich viedli k stanoveniu daného odhadu. Na záver prebehne diskusia a urobí sa definitívny odhad náročnosti.

Neformálne stretnutia sú zvyčajne stretnutia na pôde fakulty, na ktorých sa nezúčastňuje zadávateľ projektu a ktoré sa realizujú najčastejšie vtedy, ak je potrebné vypracovať nejaký dokument, kde je potrebná kooperácia a diskusia všetkých (alebo väčšiny) členov tímu. Príkladom takéhoto dokumentu bola prihláška do súťaže *TP Cup*. Takéto stretnutia sa zvyčajne konajú v pondelok okolo 15:00. Každý člen tímu pri nich komunikuje svoje názory a zvyčajne prebehne aj tvorba príslušného dokumentu cez *Google Drive*.

Okrem neformálnych stretnutí na fakulte sa nám podarilo zrealizovať aj stretnutie v podniku v meste – teambuilding – na ktorom sa zúčastnil aj vedúci projektu, a ktorého výsledkom bolo ďalšie zlepšenie vzťahov medzi členmi tímu.

# 10 Manažment testovania a prehliadok

Úlohou manažéra testovania a prehliadok je merať kvalitu a funkcionalitu kódu. Na meranie funkcionality, pokrytia a stability kódu sa používa *unit testing*. Tento pojem zahŕňa nástroje, metodiky a činnosti spojené s overením funkcionality kódu. Nasledovná metodika sa venuje práve tejto problematike a udáva smernice, ako správne testovať kód. Prehliadkou kódu *(angl. code review)* zahŕňa kontrolu komplexností a duplicít kódu, nedodržiavanie konvencii a pod. Na toto používame externú službu *Code Climate*. Metodika sa primárne zameriava na testovanie kódu.

## 10.1 Metodika – Testovanie softvéru

### 10.1.1 Úvod

Táto metodika sa zaoberá testovaním vyvíjaného softvérového projektu *Crowdex* zameraného na manažment experimentov. Cieľom tejto metodiky je stanoviť vhodné postupy a normy pre vytváranie a údržbu automatizovaných testov pri vývoji a údržbe softvéru, tak ako aj objasniť technológie, ktoré tvorbu testovacích prípadov zjednodušujú. Metodika nepokrýva výukový návod ako písať jednotlivé testy, iba stručné ukážky.

Metodika je určená všetkým osobám, ktoré sú priamo zainteresované vo vývoji projektu *Crowdex*, predovšetkým serverovej časti.

### 10.1.2 Zoznam nadväzujúcich metodík a dokumentov

Tento dokument nenadväzuje na žiadne metodiky. Pre efektívne písanie testov je však dôrazne odporúčané naštudovať si nasledujúce dokumenty:

- Výukový kurz na prácu s gemom *RSpec* – http://rspec.codeschool.com/
- Konvencie ako správne komentovať a opisovať testy – http://betterspecs.org/

### 10.1.3 Vymedzenie pojmov a skratiek

V dokumente sú obsiahnuté slovenské pojmy, avšak väčšina zahraničných materiálov používa anglické pojmy. Pre lepšiu orientáciu je v tejto časti obsiahnutý zoznam pojmov a skratiek, aj s ich anglickým ekvivalentom.

- ***Testovací prípad*** *(angl. Test Case)* – testovací prípad, ktorý zahŕňa drobnú funkcionalitu určenú na testovanie.
- ***Jednotkový test*** *(angl. Unit Test)* – drobná porovnávacia podmienka, ktorá overuje správanie daného testovacieho prípadu.
- ***Testom riadený vývoj*** *(angl. Test Driven Development – TDD)* – spôsob vývoja softvéru, pri ktorom sa najprv napíše testovací prípad a následne sa vyskúša či zlyhá. Po tejto etape nasleduje implementácia funkcionality, po ktorej by testovací prípad mal prejsť a overiť túto funkcionalitu.

- ***Pokrytie kódu*** *(angl. Code Coverage)* – údaj, ktorý udáva percentuálne množstvo pokrytia kódu testami. Ukazuje ktoré riadky kódu sú pokryté testami a ktoré nie. Užitočné pri kontrole kvality a vyhodnocovaní rizík vzniknutia neodhalenej chyby.
- **Code Climate** - služba poskytujúca prehliadku a analýzu kódu.

### 10.1.4 Postupy

Metodika opisuje použité technológie na testovanie, pokyny na udržiavanie priečinku s testami a v závere udáva pokyny na testom riadený vývoj, ktoré sú všeobecne záväzné.

#### 10.1.4.1 Používané technológie

Na testovanie sú použité nasledujúce technológie (gemy):

- **RSpec** – knižnice určené na testovanie aplikácií v *Ruby on Rails*. Krátka ukážku s vysvetlením prestavuje Obrázok 10.1.

```ruby
require 'spec_helper'

# Opis triedy, ktorá je určená na testovanie
describe Address do

  # Blok before sa vykonáva pred každým unit testom
  before do
    @address = Address.new(
      street: 'King Alley 7',
      postal_code: 77777,
      city: 'Gondor',
      country: 'Middle-earth'
    )
  end

  # Inštancia triedy, ktorá sa testuje
  subject { @address }

  # Jednotlivé unit testy. Testujú správanie inštancie, ktorá je v bloku
  # subject. Treba mať napamäti, že jednotlivé unit testové bloky sa
  # vykonávajú v náhodnom poradí.
  it { should respond_to(:street) }
  it { should respond_to(:postal_code) }
  it { should_not respond_to(:state) }
  it { should be_valid }

  # Opis testovacieho prípadu
  describe 'empty fields' do
    it { @address.street = '    ' }
    it { @address.postal_code = '    ' }

    # Vykonáva sa po každom unit teste nachádzajucom sa v príslušnom bloku.
    after { should_not be_valid }
  end
end
```

*Obrázok 10.1: Ukážka s vysvetleniami základných prvkov RSpec*

- **FactoryGirl** - pomôcka na zjednodušené vytváranie modelov, ktoré budú následne predmetom testovania (Obrázok 10.2).

```ruby
require 'spec_helper'

# Ukážka definovania 'továrne' na vytváranie modelu pre addresu.
# Nachádzajú sa module 'spec/factories.rb'.
factory :address do
  # Sequence pri každom volaní metódy inkrementuje premennú 'n'.
  sequence(:street) { |n| "King Alley #{n}" }
  postal_code 77777
  city 'Gondor'
  country 'Middle-earth'
end

describe Address do

  before do
    # Ukážka vytvárania modelu bez pomoci FactoryGirl.
    @address = Address.new(
      street: 'King Alley 7',
      postal_code: 77777,
      city: 'Gondor',
      country: 'Middle-earth'
    )

    # Ukážka vytvárania ekvivalentného model s pomocou FactoryGirl.
    @address = FactoryGirl.build(:address)
  end
```

*Obrázok 10.2: Ukážka vytvárania modelov pomocou FactoryGirl*

- **codeclimate-test-reporter** - gem slúžiaci na prepojenie testov so službou Code Climate a hodnotenie pokrytia kódu testami.

#### 10.1.4.2 Štruktúra rspec priečinku

Na udržanie prehľadnosti priečinku *rspec* bolo nevyhnutné stanoviť určitú štruktúru. Je podstatné aby ste ju dodržiavali:

- **config/***

  Testy v tomto priečinku sú určené na testovanie konfigurácie. Užitočnou utilitou v tomto priečinku je testovací skript *localization_spec.rb* testujúci lokalizačné súbory, najmä či ich štruktúra a jednotlivé kľúče sú identické.

- **helpers/***

  Priečinok obsahuje testy určené na overenie funkcionality komplexnejších pomocných metód.

- **mailers/***

Priečinok obsahujúci testy pre automatické odosielanie emailov.

- **models/***

  Jeden z najdôležitejší priečinkov. Obsahuje testy, ktoré sú určené na validáciu modelov. Testuje ich správanie, limity, obmedzenia, funkcionalitu, a podobne.

- **support/***

  Priečinok obsahujúci pomocné metódy, ktoré majú zjednodušovať samotné testovanie. Jedným príkladom takejto metódy je metódy *wait_for_ajax*, ktorá ošetruje čakanie na asynchrónne *ajax* volania.

- **views/***

  Tento priečinok obsahuje testy, ktoré priamo testujú funkcionalitu stránky (vypĺňanie formulárov, prihlasovanie, funkčnosť odkazov, funkčnosť tlačidiel, a podobne). Taktiež sem patria testy validujúce samotné html súbory (správnosť štýlov, elementov, a podobne).

- **factories.rb**

  Súbor obsahujúci "továrne" na vytváranie rôznych modelov. Zjednodušuje vytváranie modelov pri ostatných testoch.

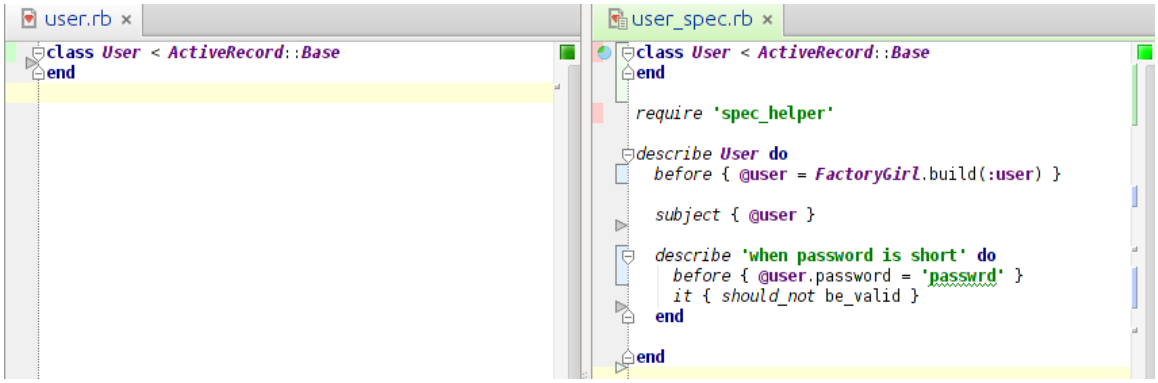
- **spec_helper.rb**

  Konfiguračný súbor pre knižnicu RSpec . Obsahuje rôzne nastavenia pre testovanie.

#### 10.1.4.3 Postup písania testov

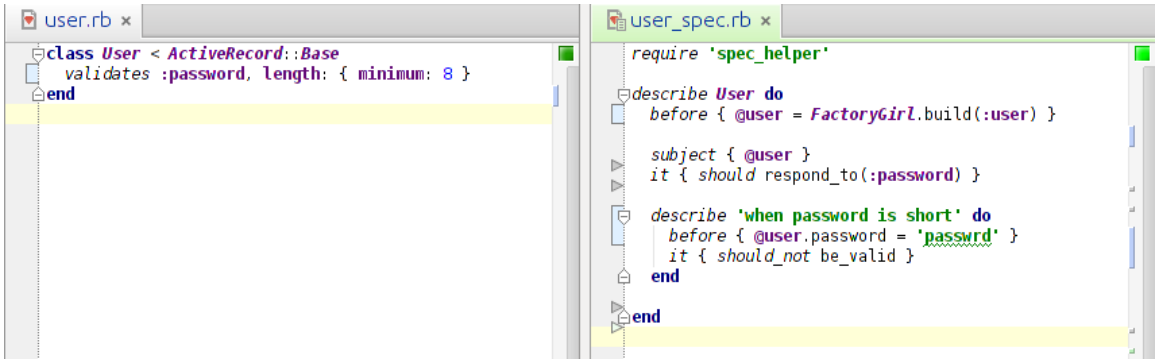
Všeobecne zaužívaný princíp testom riadeného vývoja je písanie jednotlivých prípadov testovania ešte pred samotnou implementáciou. Tento postup je záväzný pre celý tím. Stručný príklad je ilustrovaný na nasledovnom príklade implementácie validácie dĺžky hesla:

1. Napíšete test, pri ktorom sa otestuje pripravovaná funkcionalita. Obrázok 10.3.zobrazuje napísaný test, telo triedy *User* je prázdne.
2. Overíte, či daný test spadne. Obrázok 10.4 zobrazuje, že test na validáciu prítomnosti hesla prešiel, ale test, ktorý kontroluje minimálnu dĺžku spadol.
3. Naimplementujete funkcionalitu. V triede *User* pribudol riadok, na validovanie minimálnej dĺžky hesla (Obrázok 10.5).
4. Overíte funkcionalitu, pričom testy musia prejsť. V našom prípade táto funkcionalita funguje a prešla testami (Obrázok 10.6).

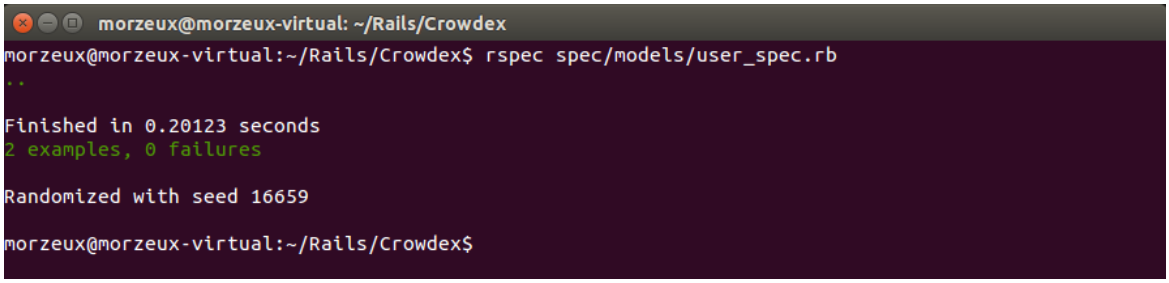



*Obrázok 10.3: Napísanie testu pre pripravovanú funkcionalitu*


*Obrázok 10.4: Overenie spadnutého testu*




*Obrázok 10.5: Implementácia funkcionality*




*Obrázok 10.6: Overenie implementovanej funkcionality*

## 10.2 Metodika – Nahlasovanie chýb

### 10.2.1 Úvod

Táto metodika sa zaoberá nahlasovaním chýb v nástroji *YouTrack*. Jej cieľom je presne zadefinovať, čo má byť obsahom hlásenia o chybe a uľahčiť tak prácu vývojárom.

Metodika pokrýva postup pre nahlásenie jednoznačne reprodukovateľných chýb v aplikácii. Naopak, metodika nepokrýva nahlasovanie vylepšení a požiadaviek na vlastnosti aplikácie, taktiež nepokrýva ďalší životný cyklus chyby od jej nahlásenia – spracovanie a jej uzatváranie.

Metodika je určená pre všetkých vývojárov aplikácie. Nie je určená pre verejnosť, ktorá aplikáciu používa, nakoľko ide o prácu s interným nástrojom.

### 10.2.2 Zoznam nadväzujúcich metodík a dokumentov

Táto metodika nenadväzuje na žiadnu inú metodiku.

### 10.2.3 Vymedzenie pojmov a skratiek

- ***aplikácia*** – Webová aplikácia Crowdex.
- ***regresia*** – Chyba, ktorá bola do aplikácie zavedená po jej aktualizácií (v pôvodnej verzii sa nevyskytujúca).
- ***vizuálna chyba*** – Chyba týkajúca sa vzhľadu používateľského rozhrania aplikácie.

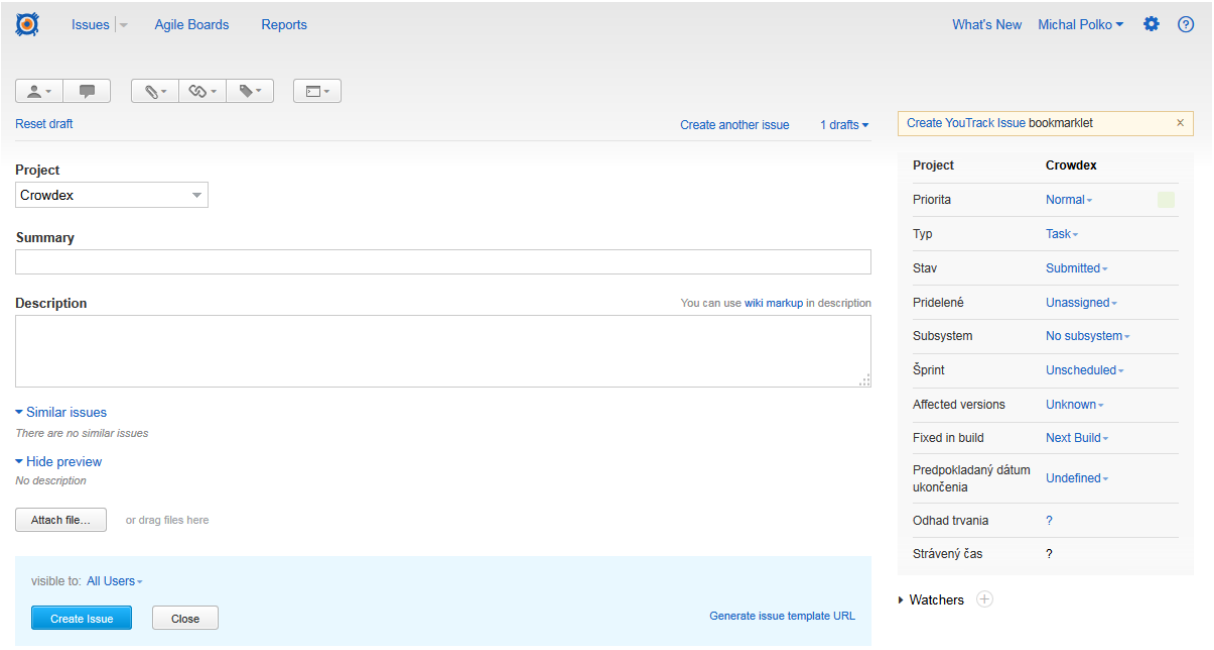
### 10.2.4 Postupy

V tejto kapitole sú popísané kroky, ktoré musíte vykonať pre nahlásenie chyby v nástroji *YouTrack*.

#### 10.2.4.1 Nahlásenie chyby

Pre nahlásenie chyby:

1. Prihláste sa do nástroja *YouTrack*.
2. Prepnite sa na zoznam chýb pomocou kliknutia na tlačidlo *Issues*.
3. Pred nahlásením chyby:
    a. Skontrolujte, či vami nájdená chyba už nebola nahlásená.
    b. Ak je to možné, skontrolujte, či vo vývojovej verzii už nebola chyba opravená.
4. Ak chyba ešte nebola nahlásená, kliknite na tlačidlo *Create Issue* vo vrchnom paneli.



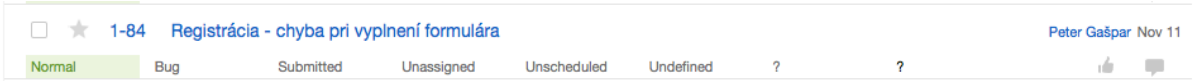

*Obrázok 10.7: Rozhranie pre nahlásenie chyby.*

V nástroji sa zobrazí obrazovka ako na obr. 1. Polia formuláru vyplňte v slovenskom jazyku nasledovným spôsobom:

- ***Summary***
  - Zadajte názov chyby podľa nasledujúceho pravidla:
    - Názov komponentu, v ktorom sa chyba vyskytuje (napr. „Prihlásenie“ alebo „Pridanie experimentu“)
    - Pomlčka
    - Výstižný názov chyby
    - Príklad dobrého názvu: „Registrácia – chybne zobrazené tlačidlo pre odoslanie formuláru“.
  - Po vložení názvu sa v sekcii *Similar issues* objaví zoznam chýb, ktoré majú podobný názov. Každú z nich otvorte a opäť skontrolujte, či sa nejedná o tú istú chybu (duplikát). Ak je už chyba nahlásená, kliknutím na tlačidlo *Close* nahlasovanie predčasne ukončíte.
- ***Description***
  - Vložte detailný popis chyby.
  - Do tohto poľa zaznamenajte:
    - Aké akcie treba v aplikácií vykonať, aby sa prejavila nahlasovaná chyba. Kroky vložte ako očíslovaný zoznam.
    - Čo sa malo udiať po vykonaní uvedených krokov.
    - Čo sa namiesto toho udialo.
    - Ak ide o vizuálnu chybu, uveďte názov a verziu webového prehliadača a názov a verziu operačného systému.
  - Ak sú vám informácie známe, zaznamenajte aj:
    - Či ide o regresiu, alebo sa chyba prejavovala vždy.

- ***Prílohy***
  - Prílohu vložte pomocou kliknutia na tlačidlo *Attach file...*
  - Ak ide o vizuálnu chybu, priložte k hláseniu aj snímku obrazovky, na ktorom je viditeľná.
  - Ak nastáva interná chyba aplikácie, je vhodnejšie vyhotoviť snímku obrazovky namiesto jej kopírovania ako text do popisu.
  - Pre ostatné chyby priložte relevantné súbory podľa vlastného uváženia.
- ***Priorita***
  - Prioritu zvoľte podľa závažnosti chyby a nasledovného kľúča s príkladmi:
    - *Show-stopper*
      - Nefunkčný server, resp. aplikácia.
    - *Critical*
      - Bezpečnostná chyba.
      - Chyba ovplyvňujúca väčšinu používateľov.
      - Chyba spôsobujúca stratu údajov používateľov.
    - *Major*
      - Chyba, ktorá ovplyvňuje malé množstvo používateľov.
      - Problémy s rýchlosťou aplikácie.
    - *Normal*
      - Všetky ostatné chyby, pre ktoré nie je vhodná iná priorita.
    - *Minor*
      - Vizuálne chyby, ktoré neovplyvňujú používanie aplikácie.
      - Kozmetické chyby kódu (komentáre a pod.)
- ***Typ***
  - Zo zoznamu vyberte *Bug*.
- ***Pridelené***
  - Ak ide o vizuálnu chybu, zaškrtnite políčka pri používateľoch *Peter Gašpar* a *Michal Polko*.
  - Ak ide o chybu servera, zaškrtnite políčko pri používateľovi *Slavo Šárik*.
  - V ostatných prípadoch ponechajte nevyplnené (hodnota *Unassigned*).
- Ostatné polia ponechajte na ich pôvodných hodnotách, resp. nevyplnené.

Nahlásenie chyby potvrďte kliknutím na tlačidlo *Create Issue*. Po úspešnom vložení sa chyba zobrazí v zozname chýb na hlavnej stránke nástroja, ako na obr. 3.




*Obrázok 10.8: Úspešne nahlásená chyba*

# Príloha A Zápisy zo stretnutí

## A.1 Zápis č. 0 z neformálneho stretnutia tímu

### A.1.1 Priebeh stretnutia

- Diskusia o prvotnom návrhu riešenia
- Práca na dokumente Zoznam kompetencií tímu

## A.2 Zápis č. 1 zo stretnutia tímu

### A.2.1 Priebeh stretnutia

- Poznámka: na nultom stretnutí, ktoré sa konalo dňa 22.9.2014 po úvodnej prednáške z predmetu Tímový projekt, vedúci tímu Ing. Michal Kompan, PhD. priblížil svoju predstavu o riešení projektu.
- Úvod do povinností na predmete Tímový projekt a ukážka dokumentácie projektu z minulých ročníkov.
- Diskusia o prvotnom návrhu riešenia.
    - Tím sa zhodol na potrebe motivácie používateľov, ktorá môže byť okrem finančnej odmeny aj vo forme bodového ohodnotenia, na základe ktorého sa budú vytvárať zoznamy najlepších používateľov.
    - Načrtnutie potreby odolnosti systému voči útokom (napr. zvyšovaniu získaných bodov).
    - Návrh zakomponovať do systému dotazníky, ktoré by umožnili zadávateľom získať dodatočné informácie o participantoch. Systém by mal ponúkať možnosť použiť preddefinované i vytvoriť vlastné dotazníky.
- Určenie frekvencie a termínu stretnutí. Stretnutia sa štandardne uskutočnia v týždňových intervaloch v utorok o 13:30.

### A.2.2 Nové úlohy do nasledujúceho stretnutia

- Stiahnuť a sfunkčniť nejaký systém na podporu vývoja a management projektov (odporúčané Redmine alebo Jira).
- Porovnať existujúce riešenia (napr. Amazon Mechanical Turk) a vypracovať špecifikáciu požiadaviek (najmä v čom sa chceme odlíšiť od konkurencie).

## A.3 Zápis č. 2 zo stretnutia tímu

### A.3.1 Priebeh stretnutia

- Kontrola úloh z predošlého týždňa – zistenie informácii o konkurencii
- Diskusia o možných funkcionalitách systému
    - priame oslovenie participantov prostredníctvom emailových notifikácií
    - vyplácanie participantov vďaka kreditu
    - zvýšiť motiváciu – súťaž pre najlepšieho
    - spôsob provízie pri nefinančných odmenách
    - systémom vypočítaná automatická výška odmeny participantov
    - odporúčanie výšky odmeny na základe podobných experimentov
    - import výsledkov z iných systémom
    - štatistika výkonnosti participantov
    - zahrnutie dotazníkov v systéme
    - nahrávanie súborov – výsledkov experimentu
- Diskusia o dodatočných funkcionalitách systému
    - sledovanie pohľadu participantov pri vykonávaní experimentu
    - sledovanie biometriky participantov

### A.3.2 Nové úlohy do nasledujúceho stretnutia

- Vymyslieť názov tímu
- Vypracovať plagát tímu – zahrnúť názov, logo a číslo tímu
- Dokončiť stránku tímu
- Načrtnúť obrázkový prototyp systému
- Vytvoriť dokument so špecifikáciou požiadaviek
- Vytvoriť predbežný plán úloh
- Diskusia k TP CUP
- Vytvoriť projektový denník

## A.4 Zápis č. 3 zo stretnutia tímu

### A.4.1 Priebeh stretnutia

- Kontrola úloh z predošlého týždňa
    - vytvorený názov tímu
    - vytvorená stránka tímu a doplnená o reálne texty a obrázky
    - každý sa priebežne oboznamuje s framework-om Ruby on Rails
- Diskusia o návrhu používateľského prostredia, spoločne sme vytvorili prototypy dizajnu
    - úvodná stránka
    - používateľský profil
    - detail experimentu
    - detail používateľa
    - detail zaregistrovanej firmy
    - výsledky vyhľadávania experimentov
- Diskusia o dodatočných funkcionalitách systému
    - notifikácie používateľov pri niektorých udalostiach v systéme
    - správy v systéme slúžiace na kontakt používateľov
    - vytvoriť možnosť používateľovi sledovať činnosť iných používateľov
    - vytvoriť rozhranie pre administrátora systému
- Spoločne sme vytvorili backlog s úlohami, ktorým sme postupne určili prioritu
- Diskusia o softvére pomáhajuceho s manažovaním projektu
    - odhlasovali sme si používanie softvéru YouTrack

### A.4.2 Nové úlohy do nasledujúceho stretnutia

- Navrhnúť dátový model
- Vytvoriť časový plán
- Pokračovať v učení sa framework-u Ruby on Rails

## A.5 Zápis č. 4 zo stretnutia tímu

### A.5.1 Priebeh stretnutia

- Kontrola úloh z predošlého týždňa
  - vytvorený „papierový“ prototyp (príloha A)
  - vytvorený predbežný dátový model (diagram v prílohe B a na Google Drive, textová verzia na Google Drive)
  - vytvorený produktový backlog spolu s prioritou jednotlivých úloh a približným rozdelením do jednotlivých šprintov
- Riešené úlohy
  - diskusia o rozdelení manažérskych rolí v tíme
    - pravdepodobne bude súvisieť s MSI/MIS, o čom sa budeme informovať
  - príprava na 1. šprint
    - dekompozícia kľúčových úloh z produktového backlogu zaradených do 1. šprintu na jednotlivé pod-úlohy
    - hlasovanie o náročnosti jednotlivých úloh
    - pridelenie náročnosti jednotlivým úlohám
    - zavedenie zodpovedných osôb za jednotlivé úlohy
    - rozdelenie úloh medzi jednotlivých členov tímov (riešitelia)
    - rozdelenie úloh je nasledujúce:

| Úloha | Zodpovedná osoba | Riešitelia |
|---|---|---|
| Jazykové nastavenia | D. Cymorek | S. Šárik |
| Registrácia nového používateľa | Š. Šmihla | S. Šárik, Š. Šmihla |
| Autentifikácia používateľa (štandardne)<br>(+ obnovenie zabudnutého hesla, overenie prekročenia maximálneho počtu pokusov) | S. Šárik | Š. Šmihla, S. Šárik |
| Pridávanie experimentov | V. Ľalík | M. Šafárik, P. Gašpar |
| Detail profilu používateľa (osoba) | M. Šafárik | D. Cymorek, V. Ľalík |
| Šablóna - farby, základné prvky | P. Gašpar | M. Polko, P. Gašpar |
| Šablóna - horný panel, bočný panel | M. Polko | M. Polko, P. Gašpar |
| Šablóny - footer | P. Gašpar | M. Polko, P. Gašpar |

    - tieto informácie boli zapísané aj do produktového backlogu

- M. Kompan bol pozvaný na Google Drive a tiež do skupiny na Facebooku
- prebehla krátka diskusia o používaní Perconiku

### A.5.2 Nové úlohy do nasledujúceho stretnutia

- Rozdeliť jednotlivé pod-úlohy z úloh medzi daných riešiteľov
- Zapísať jednotlivé úlohy a pod-úlohy do YouTracku
- Začať pracovať na jednotlivých úlohách
- Prihláška do TP Cupu

## A.6 Zápis č. 5 zo stretnutia tímu

### A.6.1 Priebeh stretnutia

- Kontrola úloh z predošlého týždňa
  - Prihláška na TP Cup
  - Zhodnotenie stavu projektu (príloha 2)
- Riešené úlohy
  - Diskusia k prihláške na TP Cup
    - Treba doladiť posledné detaily, vytlačiť
  - Konkurencia, ktorá robí to isté www.gesis.org/iirpanel
  - Manažérske role (Príloha 1)
  - Informácie z AIS cez ldap.stuba.sk
  - Time Tracking
    - Značiť si čas, koľko venujeme projektu
    - Viesť si projektový denník
  - Názov aplikácie, doména
    - CrowdTasker – problém s doménou
    - Crowdex – zatiaľ pracovný názov
    - Crowdexp – joke (Crowdex Premium)
  - Stav projektu
    - Dokončiť Captchu pre registráciu a zabudnuté heslo
    - Dokončiť unit testy
    - Pridávanie experimentu, follow používateľov je takmer dokončené
    - Zobrazenie profilu používateľa je takmer dokončené

### A.6.2 Nové úlohy do nasledujúceho stretnutia

- Dokončiť prihlášku na TP Cup
- Dokončiť úlohy vyšpecifikované do šprintu 1

### A.6.3 Prílohy

#### A.6.3.1 Príloha 1 – Rozdelenie rolí

| **Meno** | **Rola** |
|---|---|
| Dušan Cymorek | Manažment dokumentácie |
| Peter Gašpar | Manažment rozvrhu |
| Vladimír Ľalík | Manažment rizík |
| Michal Polko | Manažment rozsahu |
| Miroslav Šafárik | Manažment komunikácie |
| Slavomír Šárik | Manažment podpory vývoja a integrácie |

| Meno | Rola |
|---|---|
| Štefan Šmihla | Manažment testovania, prehliadok |

*Tabuľka 10.1: Rozdelenie rolí*

### A.6.3.2 Príloha 2 – Stav úloh

| Úloha | Zodpovedná osoba | Stav úlohy |
|---|---|---|
| Jazykové nastavenia | Dušan Cymorek | Hotovo |
| Registrácia nového používateľa | Štefan Šmihla | Hotovo |
| Autentifikácia používateľa (štandardne) (+ obnovenie zabudnutého hesla, overenie prekročenia maximálneho počtu pokusov) | Slavomír Šárik | Takmer hotovo |
| Pridávanie experimentov | Vladimír Ľalík | Takmer hotovo |
| Detail profilu používateľa (osoba) | Miroslav Šafárik | Takmer hotovo |
| Šablóna - farby, základné prvky | Peter Gašpar | Hotovo |
| Šablóna - horný panel, bočný panel | Michal Polko | Rozpracované |
| Šablóny - footer | Peter Gašpar | Rozpracované |

*Tabuľka 10.2: Stav úloh pre šprint 1*

## A.7 Zápis č. 6 zo stretnutia tímu

### A.7.1 Priebeh stretnutia

- Zhodnotenie predchádzajúcich úloh: všetky úlohy boli úspešne splnené.
- Odovzdali sme prihlášku na TP Cup.
- Úspešne sme dokončili 1. šprint:
    - všetky požadované úlohy boli splnené,
    - chceme vytvoriť automatické premazávanie neaktívnych e-mailov,
    - diskutovali sme problém enormného vytvárania záznamov v databáze pri každom prihlásení (riešenie bolo odložené na neskoršie fázy projektu),
    - záznamy v súbore prekladov budeme zoraďovať podľa abecedy kvôli lepšej prehľadnosti,
    - prechádzali sme si jednotlivé úlohy a vyhodnocovali sme, ktoré sme stihli načas a s ktorými sme, naopak, mali problémy.
- Musíme si zadefinovať proces určený na prehliadky kódu:
    - koľko % kódu sa náhodne pozrie,
    - stanovíme si členov tímu, ktorí budú mať túto úlohu na starosti a budú sa striedať,
    - nesmieme zabúdať aj dokumentáciu,
    - dohodli sme sa, že prehliadky kódu bude vykonávať osoba zodpovedná za danú úlohu.
- Mali by sme si rozdeliť úlohy pre manažment projektu a produktu.
- Budeme tlačiť iba niektoré časti dokumentácie, a to po semestri.
- Riešili sme vytváranie a mergovanie branchov na GitHube:
    - Slavo napísal stručný návod ako postupovať v tomto procese.
- Stanovili sme si metodiky potrebné pre predmet MIS/MSI, ktoré použijeme aj v dokumentácií k Tímovému projektu.
- Prebrali sme dokumentáciu, ktorá sa bude odovzdávať po 3. šprinte.
- Brainstormovali sme myšlienku oprávnení na zobrazenie a pridávanie sa do experimentov:
    - usúdili sme, že nie je potrebné skrývať experimenty a zakazovať ich zobrazenie.
- Diskutovali sme potrebu naštudovania si protokolu na prihlasovanie do AIS v našom projekte (ldap.stuba.sk).
- Musíme vyriešiť problém prístupu na web cez pevnú sieť.
- Definovali sme si príbehy druhého šprintu, ktorý bude dvojtýždňový (28.10. - 11.11.).
- Dohodli sme sa, že úlohy v YouTracku bude vždy vytvárať osoba zodpovedná za danú úlohu.

### A.7.2 Nové úlohy do nasledujúceho stretnutia

- Riešenie príbehov druhého šprintu.
- Rozdelenie a priebežná práca na metodikách tímu.

### A.7.3 Prílohy

#### A.7.3.3 Príloha 1 – Rozdelenie úloh pre druhý šprint

| Úloha | Zodpovedná osoba | Riešitelia |
|---|---|---|
| Správa experimentu | Vlado, Dušan | Slavo, Peťo |
| Detail experimentu | Miro | Štefan |
| Nastavenia – môj profil + vyplnenie | Štefan | Dušan, Mišo |
| Notifikácie | Slavo | Vlado, Miro |
| Šablóna – profil používateľa/spoločnosti | Peťo | Mišo, Peťo |
| Šablóna – stránka experimentu | Mišo | Mišo, Peťo |
| Šablóna – registrácia/prihlásenie | Peťo | Mišo, Peťo |

*Tabuľka 10.3: Rozdelenie úloh pre druhý šprint*

## A.8 Zápis č. 7 zo stretnutia tímu

### A.8.1 Stav úloh 2. šprintu

| ID | Názov | Stav | Riešitelia |
|---|---|---|---|
| 1-27 | Profilový obrázok | Reopened | Dušan |
| 1-40 | Vytvoriť inštrukcie pre návrh formulárov | In Progress | Peter |
| 1-44 | Správa experimentu | In Progress | Peter, Slavo |
| 1-45 | Šablóna - stránka experimentu | Submitted | Michal, Peter |
| 1-46 | Notifikácie | In Progress | Miro, Vlado |
| 1-47 | Šablóna - profil používateľa/spoločnosti | In Progress | Michal, Peter |
| 1-48 | Šablóna - registrácia/prihlásenie | In Progress | Michal, Peter |
| 1-49 | Detail experimentu | In Progress | Štefan |
| 1-50 | Pridanie neaktivovaného používateľa do experimentu | Verified | Peter |
| 1-51 | Nastavenia profilu | In Progress | Štefan |
| 1-52 | Úprava prekladov | In Progress | Dušan |
| 1-54 | Upgrade Youtracku | Verified | Slavo |
| 1-55 | Pridať roly tímu na hlavnú stránku | Verified | Peter |
| 1-56 | Vytvoriť retrospektívu za 1. šprint | Verified | Dušan, Mišo, Miro, Štefan, Peter, Slavo, Vlado |
| 1-59 | Prepojenie nástrojov pre správu kódu s projektom | Verified | Slavo |
| 1-62 | Nastavenia profilu - príprava view + napojenie na controller | Submitted | Dušan |
| 1-63 | Vytvorenie udalosti v systéme | In Progress | Vlado |
| 1-64 | Nastavenie monitorovania servera a aplikácie | Verified | Slavo |
| 1-65 | Zobrazenie používateľského rozhrania pre administrátora experimentu (admin UI) | In Progress | Slavo |
| 1-66 | Nastavenia detailov experimentu | In Progress | Slavo |
| 1-67 | Nastavenie podmienok pre vstup používateľa do experimentu | Submitted | Slavo |
| 1-68 | Detail experimentu - zobrazenie informácií | Submitted | Štefan |
| 1-69 | Detail experimentu - prihlásenie do experimentu | Submitted | Štefan |
| 1-71 | Po vyžiadaní obnovy hesla sa podarí prihlásiť aj neaktivovanému používateľovi | Verified | Peter, Slavo |

| ID | Názov | Stav | Riešitelia |
|---|---|---|---|
| 1-72 | Vytváranie zdieľaných migrácií | Verified | Slavo |
| 1-73 | Zobrazenie notifikácií | In Progress | Miro |
| 1-74 | Rekonfigurácia SMTP | Verified | Slavo |

*Tabuľka 10.4: Stav úloh 2. šprintu*

### A.8.2 Poznámky zo stretnutia

- Dohodli sme sa na deadline pre tvorbu špecifikácií:
  - pre dvojtýždňový šprint - do troch dní od začiatku (piatok),
  - pre jednotýždňový šprint - do dvoch dní od začiatku (štvrtok).
- Musíme priebežne dokumentovať kód:
  - zatiaľ jednoduché komentáre - čo metóda robí, aké má parametre, ošetruje výnimky?, čo je výstupom,
  - neskôr aj prostredníctvom nástroja *YARD*.
- Dohodli sme sa, že budeme komentovať účel gemov v *Gemfile*.
- Slavo nám predstavil ideu, ako budeme medzi sebou zdieľať migrácie databázy a zredukujeme výskyt konfliktov:
  - vytvorí sa *migration* vetva, do ktorej budú primárne vkladané migrácie,
  - v jednotlivých vetvách treba pravidelne sťahovať zmeny z vetiev *master* a *migration*.
- Dohodli sme si spôsob ukončovania jednotlivých úloh.
- Prešli sme si stav aktuálneho šprintu.

## A.9 Zápis č. 8 zo stretnutia tímu

### A.9.1 Stav úloh 2. šprintu

| ID | Názov | Stav | Riešitelia |
|---|---|---|---|
| 1-27 | Profilový obrázok | Verified | Michal |
| 1-44 | Správa experimentu | Verified | Peter, Slavo |
| 1-45 | Šablóna - stránka experimentu | Verified | Michal, Peter |
| 1-46 | Notifikácie | Verified | Miro, Vlado |
| 1-47 | Šablóna - profil používateľa/spoločnosti | Verified | Michal, Peter |
| 1-48 | Šablóna - registrácia/prihlásenie | Verified | Michal, Peter |
| 1-49 | Detail experimentu | Verified | Štefan |
| 1-50 | Pridanie neaktivovaného používateľa do experimentu | Verified | Peter |
| 1-52 | Úprava prekladov | Verified | Dušan |
| 1-54 | Upgrade Youtracku | Verified | Slavo |
| 1-55 | Pridať roly tímu na hlavnú stránku | Verified | Peter |
| 1-56 | Vytvoriť retrospektívu za 1. šprint | Verified | Všetci |
| 1-59 | Prepojenie nástrojov pre správu kódu s projektom | Verified | Slavo |
| 1-60 | Nastavenie profilu – upload obrázka | Verified | Michal |
| 1-63 | Vytvorenie udalosti v systéme | Verified | Vlado |
| 1-64 | Nastavenie monitorovania servera a aplikácie | Verified | Slavo |
| 1-65 | Zobrazenie používateľského rozhrania pre administrátora experimentu (admin UI) | Verified | Slavo |
| 1-66 | Nastavenia detailov experimentu | Verified | Slavo |
| 1-67 | Nastavenie podmienok pre vstup používateľa do experimentu | Verified | Slavo |
| 1-68 | Detail experimentu - zobrazenie informácií | Verified | Štefan |
| 1-69 | Detail experimentu - prihlásenie do experimentu | Verified | Štefan |
| 1-71 | Po vyžiadaní obnovy hesla sa podarí prihlásiť aj neaktivovanému používateľovi | Verified | Peter, Slavo |
| 1-72 | Vytváranie zdieľaných migrácií | Verified | Slavo |
| 1-73 | Zobrazenie notifikácií | Verified | Miro |

| ID | Názov | Stav | Riešitelia |
|---|---|---|---|
| 1-74 | Rekonfigurácia SMTP | Verified | Slavo |
| 1-75 | Správa experimentu – Značky | Verified | Peter |
| 1-76 | Správa experimentu – Budget | Verified | Peter |
| 1-77 | Správa experimentu – Správa participantov | Verified | Peter |
| 1-79 | Príprava unit testov pre model v šprinte 2 | Fixed | Štefan |
| 1-80 | Nastavenie servera na produkciu a staging | Verified | Slavo |
| 1-81 | Integrácia systému – 2. šprint | Verified | Slavo |

*Tabuľka 10.5: Stav úloh 2. šprintu*

### A.9.2 Poznámky zo stretnutia

- Konzultovali sme stav úloh. Dušan nestihol spraviť svoju úlohu – zostáva doladiť adresy (fakturačná a dodacia), doplniť pridávanie skúseností používateľa.
- Slavo vysvetlil zavedenie zoznamu zmien na službe *Github,* aby bol product owner lepšie informovaný o priebehu vývoja aplikácie. Používa sa sémantické číslovanie verzií.
- Slavo upozornil na aktualizovanie metodiky používania *Git practices*, ktorá popisuje prácu so systémom Git (vytváranie vetiev, práca na úlohách, postup pri pull requestoch, odstraňovaní vetiev).
- Opätovne sme sa dohodli na záväznom dodržiavaní pravidiel a postupov opísaných v metodikách.
- Slavo navrhol zmenu oznamovacieho textu pri obnovovaní hesla. Pri zadaní správneho, aj neprávneho emailu sa zobrazí informácia o zaslaní obnovovacieho odkazu, ak email existuje. Týmto zamedzíme špekulantom zisťovanie emailov, ktoré sú v systéme zaregistrované.
- Slavo predniesol požiadavku o použitie aplikácie na monitorovanie serveru od tímu č. 10. Dohodli sme sa na tom, že Slavo sa spýta, akú funkcionalitu poskytujú a či je to stabilné (aby to neohrozilo vývoj našej aplikácie).
- Dohodli sme sa na priebehu tretieho šprintu – tento šprint bude "upratovací" a dokumentačný. Nebudeme pridávať novú funkcionalitu, ale refactorovať a vylepšovať existujúcu, popri tom sa bude finalizovať dokumentácia.
- Dohodli sme sa na požiadavke o zvýšenie dostupných prostriedkov na serveri – pamäť +1GB a disk +15GB.
- Na návrh prof. Bielikovej hlbšie preskúmame možnosti konkurenčnej služby *CrowdFlower* a ich *API.*
- Je potrebné upraviť pridávanie podmienok v experimente tak, aby systém umožňoval ich voľné zadávanie, pričom napríklad pri podmienke na vek rozpozná, že ide o rozsah.

- Diskutovali sme o značkách a skúsenostiach používateľa. Skúsenosti budú riešené podobne ako značky experimentu – voľný text + automatické dopĺňanie.
- Odporúčanie experimentov by malo byť realizované okrem priraďovania značiek aj prostredníctvom informácií o predchádzajúcich experimentoch používateľa.
- Dohodli sme sa na premenovaní stĺpca *„Stav zverejnenia”* na *„Stav”* pri správe experimentu.
- Rozdelili sme si úlohy na ďalší šprint, ktorý bude trvať 1 týždeň.

### A.9.3 Úlohy pre 3. šprint

| ID | Názov | Stav | Riešitelia |
|---|---|---|---|
| 1-51 | Nastavenia profilu | In Progress | Dušan, Michal |
| 1-57 | Vytvorenie a odovzdanie dokumentácie - inžinierske dielo | Submitted | Všetci |
| 1-58 | Vytvorenie a odovzdanie dokumentácie - riadenie | Submitted | Všetci |
| 1-62 | Nastavenia profilu - príprava view + napojenie na controller | In Progress | Dušan |
| 1-82 | Integrácia systému – 3. šprint | In Progress | Slavo |
| 1-83 | Refactoring - pridávanie experimentu | Verified | Peter |
| 1-84 | Chyba pri vyplnení registračného formulára | Submitted | |
| 1-85 | Preformulovanie textov | Submitted | Peter |
| 1-86 | Refactoring zdrojového kódu | In Progress | Všetci |
| 1-87 | Refactoring – experiment | In Progress | Slavo, Štefan |
| 1-88 | Refactoring – participanti | In Progress | Peter |
| 1-89 | Refactoring – dizajn | Submitted | Michal, Peter |
| 1-90 | Refactoring – preklady | Submitted | Dušan, Peter |
| 1-91 | Predbežná príprava unit testov | Submitted | Štefan |
| 1-92 | Refactoring – follow | Submitted | Vlado |
| 1-93 | Refactoring – notifikácie | Submitted | Miro, Vlado |

*Tabuľka 10.6: Úlohy pre 3. šprint*

# Príloha B Retrospektívy k šprintom

## B.1 Retrospektíva k 1. šprintu

**Autori:** Dušan Cymorek, Peter Gašpar, Slavomír Šárik

**Trvanie šprintu:** 14.10. – 27.10

### B.1.1 Zoznam úloh a stav ich riešenia

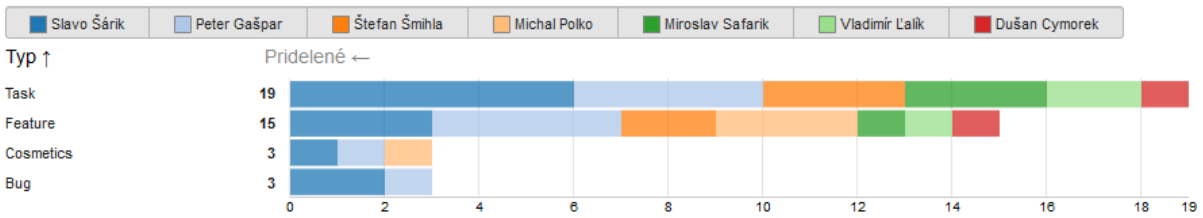
V tabuľke č. 1 sa nachádza prehľad úloh, ktoré sme riešili počas prvého šprintu, stav ich splnenia a poznámky z nich vyplývajúce. Náplňou tohto šprintu bolo implementovanie základných funkcionalít systému.

| ID | Popis | Typ | Riešiteľ | Záver |
|---|---|---|---|---|
| 1-6 | Jazykové nastavenia | Feature | Slavomír | akceptovaná |
| 1-7 | Registrácia nového používateľa | Feature | Štefan, Slavomír | akceptovaná |
| 1-8 | Registrácia nového používateľa - Vytvorenie modelu pre používateľa | Task | Štefan, Slavomír | akceptovaná |
| 1-9 | Registrácia nového používateľa - Zasielanie mailov | Task | Štefan, Slavomír | akceptovaná |
| 1-10 | Registrácia nového používateľa - Captcha | Task | Štefan | akceptovaná |
| 1-11 | Registrácia nového používateľa - Aktivácia konta | Task | Slavomír | akceptovaná |
| 1-12 | Pridávanie experimentov | Feature | Miroslav, Peter | akceptovaná |
| 1-13 | Pridávanie experimentov - Vytvorenie modelu pre experiment | Task | Miroslav | akceptovaná |
| 1-14 | Pridávanie experimentov - Názov, popis | Task | Miroslav | akceptovaná |
| 1-15 | Pridávanie experimentov - Oprávnenia na prístup | Task | Miroslav | akceptovaná |
| 1-16 | Pridávanie experimentov - Budget | Task | Peter | akceptovaná |
| 1-17 | Pridávanie experimentov - Účastníci | Task | Peter | akceptovaná |
| 1-18 | Pridávanie experimentov - Zverejnenie | Task | Peter | akceptovaná |
| 1-19 | Autentifikácia používateľa (štandardne) | Feature | Štefan, Slavomír | akceptovaná |
| 1-20 | Autentifikácia používateľa (štandardne) - Obnovenie zabudnutého hesla | Task | Slavomír | akceptovaná |
| 1-21 | Autentifikácia používateľa (štandardne) - Overenie prekročenia maximálneho počtu pokusov + dočasné zablokovanie konta | Task | Slavomír | akceptovaná |
| 1-22 | Autentifikácia používateľa (štandardne) - Exspirácia tokenu (pri zabudnutí hesla) | Task | Slavomír | akceptovaná |

| ID | Popis | Typ | Riešiteľ | Záver |
|---|---|---|---|---|
| 1-25 | Detail profilu používateľa | Feature | Dušan, Vladimír | akceptovaná |
| 1-26 | Detail profilu používateľa - Zobrazenie info o používateľovi | Task | Dušan | akceptovaná |
| 1-27 | Detail profilu používateľa - Profilový obrázok | Task | Dušan | presunutá |
| 1-28 | Detail profilu používateľa - Priatelia (follow) | Task | Vladimír | akceptovaná |
| 1-29 | Detail profilu používateľa - Poznámka o spoločnosti | Task | Vladimír | akceptovaná |
| 1-30 | Šablóna - farby, základné prvky | Feature | Michal, Peter | akceptovaná |
| 1-31 | Šablóna - horný panel, bočný panel | Feature | Michal, Peter | akceptovaná |
| 1-32 | Šablóny - footer | Feature | Michal, Peter | akceptovaná |
| 1-33 | Vyriešiť automatický deploy do produkcie | Bug | Slavomír | akceptovaná |
| 1-34 | Aktualizácia stránky tímu | Cosmetics | Michal, Peter | akceptovaná |
| 1-36 | [BUG] Preklady nefunguju v mailoch | Bug | Slavomír | akceptovaná |
| 1-39 | Aktualizácia plánu projektu na webe | Bug | Peter | akceptovaná |
| 1-42 | Aktualizácia dizajnu webu | Task | Peter | akceptovaná |
| 1-43 | Úprava chybových stránok | Cosmetics | Slavomír | akceptovaná |

*Tabuľka 10.7: Prehľad úloh*

Na nasledujúcom grafe môžeme vidieť distribúciu jednotlivých typov úloh medzi autorov.




*Obrázok 10.9: Distribúcia jednotlivých typov úloh medzi autorov*

### B.1.2 Časový prehľad

Tabuľka 10.8 obsahuje prehľad úloh a potrebného času na ich realizáciu pre jednotlivých členov tímu.

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| *Dušan* | Development | 5 | 5,42 |
| | **Celkovo** | **5** | **5,42** |

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| *Michal* | Development | 6 | 13 |
| | **Celkovo** | **6** | **13** |
| *Miroslav* | Development | 5 | 5,58 |
| | Testing | 1 | 2 |
| | Documentation | 1 | 2 |
| | **Celkovo** | **7** | **9,58** |
| *Peter* | No type | 5 | 9,8 |
| | Development | 8 | 9,37 |
| | Testing | 2 | 1 |
| | **Celkovo** | **15** | **20,17** |
| *Slavomír* | No type | 3 | 7 |
| | Development | 15 | 27 |
| | Testing | 1 | 0,5 |
| | Documentation | 3 | 5,67 |
| | **Celkovo** | **22** | **40,17** |
| *Vladimír* | Development | 5 | 8,67 |
| | Testing | 2 | 1,67 |
| | Documentation | 1 | 2 |
| | **Celkovo** | **8** | **12,33** |
| *Štefan* | No type | 1 | 4 |
| | Development | 5 | 13 |
| | Testing | 2 | 3 |
| | Documentation | 1 | 2 |
| | **Celkovo** | **9** | **22** |

*Tabuľka 10.8: Časový prehľad úloh*

### B.1.3 Zhodnotenie

Pri retrospektíve k šprintu sme sa na stretnutí vyjadrili k jednotlivým úlohám a problémom, ktoré pri ich riešení nastali. Poznámky k jednotlivým úlohám sa nachádzajú v Tabuľka 10.9.

| ID úlohy | Popis | Poznámky |
|---|---|---|
| 1-6 | Jazykové nastavenia | Dohodli sme sa, že položky v súbore s jazykmi budú z dôvodu prehľadnosti zoradené podľa abecedy.<br><br>Úlohy, ktoré vyplynuli pre ďalšie šprinty:<br>Vytvorenie metodiky na pridávanie a používanie prekladov, zjednotenie používania prekladov v kóde. |
| 1-11 | Registrácia nového používateľa - Aktivácia konta | Prehodnotili sme, že v nasledujúcich šprintoch bude potrebné vyriešiť automatické premazávanie nedokončených aktivácií konta.<br><br>Úlohy, ktoré vyplynuli pre ďalšie šprinty:<br>Vytvorenie automatickej úlohy na serveri na odstránenie starších už neplatných aktivácií. |
| 1-16 | Pridávanie experimentov - Budget | Pri nastavovaní budgetu sme zhodnotili, že je nezmyselné ho nastavovať pomocou políčka "Zvýšiť/Znížiť budget". Výška peňažného budgetu sa bude nastavovať priamo ako číselná hodnota.<br><br>Úlohy, ktoré vyplynuli pre ďalšie šprinty:<br>V experimente zavedieme možnosť pridať aj iný ako finančný budget. Zmenu budgetu nebude možné vykonať po spustení experimentu. |
| 1-18 | Pridávanie experimentov - Zverejnenie | Prehodnotili sme, že súčasná klasifikácia ochrany experimentu z hľadiska prístupu (verejný, súkromný, na odkaz) nie je potrebná.<br><br>Úlohy, ktoré vyplynuli pre ďalšie šprinty:<br>Z pôvodného návrhu ponecháme iba možnosť skryť experiment z výsledkov vyhľadávania. |
| 1-19 | Autentifikácia používateľa (štandardne) | Diskutovali sme problém enormného vytvárania záznamov v databáze pri každom prihlásení. Riešenie bolo odložené na neskoršie fázy projektu. |
| 1-27 | Detail profilu používateľa - | Bol implementovaný cez Gravatar, rozhodli sme |

| ID úlohy | Popis | Poznámky |
|---|---|---|
| | Profilový obrázok | sa však, že umožníme používateľom nahrávať vlastný obrázok priamo na náš server. Úloha tak bola presunutá do druhého šprintu, kde súvisí s úlohou 1-51 Nastavenia profilu. |

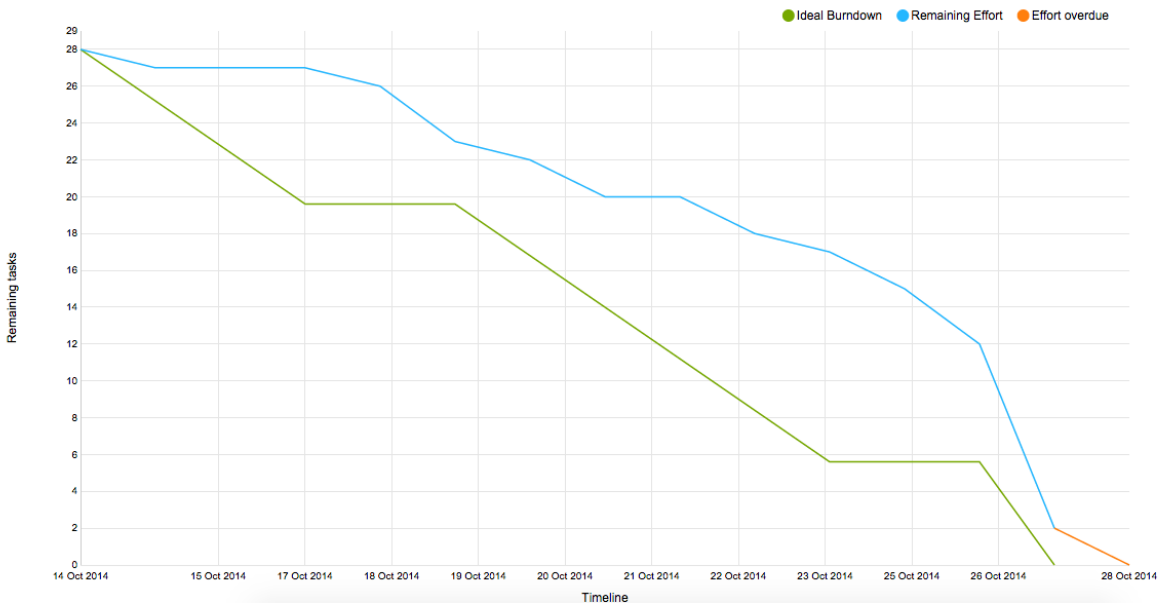
*Tabuľka 10.9: Poznámky k úlohám*

Ďalšie úlohy, ktoré nám vyplynuli pre nasledujúce šprinty:

- Musíme si zadefinovať proces určený na prehliadky kódu.
- V nástroji YouTrack musíme pravidelne meniť stavy jednotlivých úloh. Niektoré z úloh sme v rámci zoznamovania sa s nástrojom označili ako vyriešené neskôr, ako sa nám ich skutočne poradilo vyriešiť. Taktiež nesmieme zabúdať predbežne si zaznamenávať čas trávený na jednotlivých úlohách.
- Počas prvého šprintu vytvárala úlohy v YouTracku osoba, ktorá bola stanovená ako vedúci stretnutia. Dohodli sme sa, že vo všetkých nasledujúcich šprintoch bude úlohy v YouTracku vždy vytvárať osoba zodpovedná za danú úlohu. Odbremeníme tým záťaž na vedúceho stretnutia a zároveň bude zodpovedný za úlohu vždy informovaný (prostredníctvom upozornenia zo systému YouTrack) o stave jej plnenia.
- Zistili sme, že v mnohých prípadoch bola odhadovaná náročnosť úloh podhodnotená. V ďalších šprintoch už odhadujeme náročnosť aj s prihliadnutím na detaily. Zaviedli sme tiež používanie tzv. SCRUM pokeru, ktorý obsahuje štandardizované odhady úloh.
- Na základe prehľadu distribúcie úloh a ich časového prehľadu sme zistili, že je potrebné rovnomernejšie rozloženie záťaže medzi jednotlivých členov tímu.

### B.1.4 Príloha 1: Burndown Chart

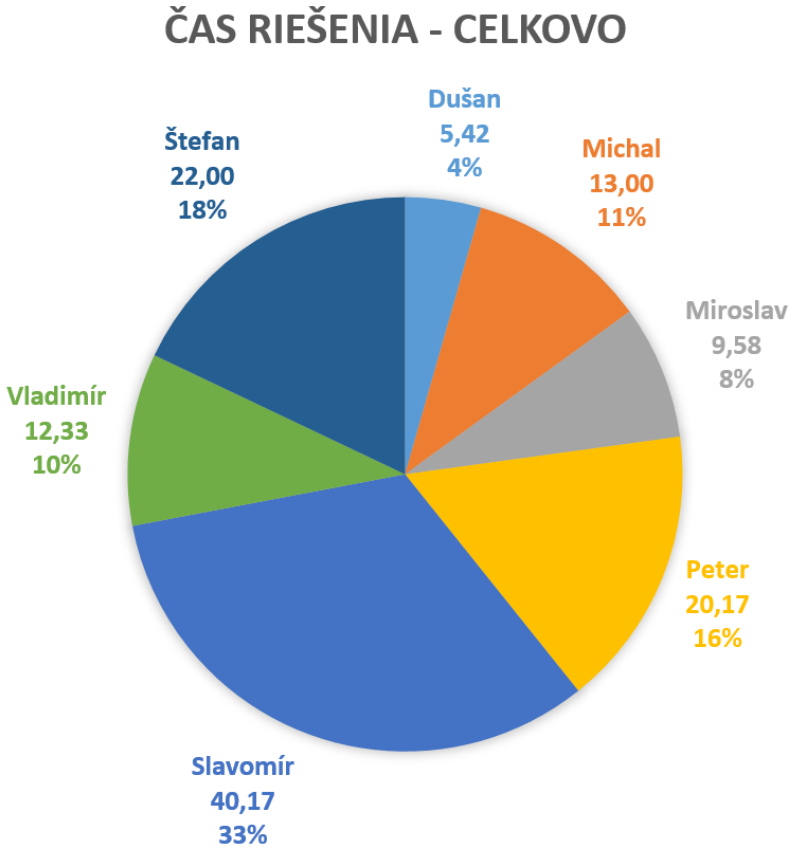
Na Obrázok 10.10 sa nachádza Burndown Chart za prvý šprint.




*Obrázok 10.10: Burndown Chart za prvý šprint*

### B.1.5 Príloha 2: Distribúcia celkového času riešenia úloh medzi autorov

V grafe (Obrázok 10.11) sa nachádza časové a následne percentuálne vyjadrenie podielu práce pre každého z autorov.




*Obrázok 10.11: Distribúcia celkového času riešenia úloh medzi autorov*

## B.2 Retrospektíva k 2. šprintu

**Autori:** Vladimír Ľalík, Miroslav Šafárik

**Trvanie šprintu:** 28.10. – 11.11.

### B.2.1 Zoznam úloh a stav ich riešenia

V Tabuľka 10.10 sa nachádza prehľad úloh, ktoré sme riešili počas druhého šprintu, stav ich splnenia a poznámky z nich vyplývajúce. Náplňou šprintu bolo implementovať rozširujúce funkcionality pre experiment (správa experimentu, detail experimentu) a nastavenia profilu používateľa. Okrem rozširujúcej funkcionality sa ako úplne nové implementovali notifikácie (ich vytvorenie v systéme a zobrazovanie).

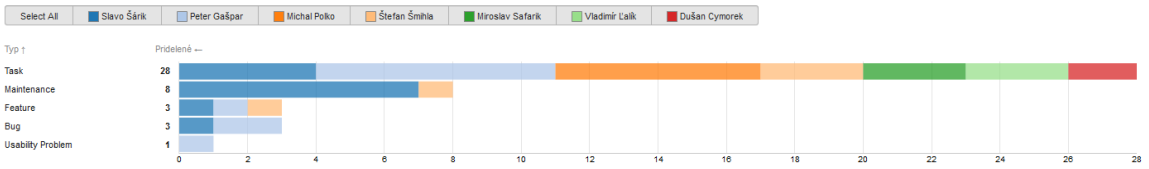
| ID | Názov | Typ | Riešitelia | Záver |
|---|---|---|---|---|
| 1-27 | Profilový obrázok | Task | Michal | akceptovaná |
| 1-44 | Správa experimentu | Feature | Peter, Slavo | akceptovaná |
| 1-45 | Šablóna - stránka experimentu | Feature | Michal, Peter | akceptovaná |
| 1-46 | Notifikácie | Feature | Miro, Vlado | akceptovaná |
| 1-47 | Šablóna - profil používateľa/spoločnosti | Feature | Michal, Peter | akceptovaná |
| 1-48 | Šablóna - registrácia/prihlásenie | Feature | Michal, Peter | akceptovaná |
| 1-49 | Detail experimentu | Feature | Štefan | akceptovaná |
| 1-50 | Pridanie neaktivovaného používateľa do experimentu | Bug | Peter | akceptovaná |
| 1-51 | Nastavenia profilu | Feature | Dušan, Mišo | riešená |
| 1-52 | Úprava prekladov | Task | Dušan | akceptovaná |
| 1-54 | Upgrade Youtracku | Maintenance | Slavo | akceptovaná |
| 1-55 | Pridať roly tímu na hlavnú stránku | Usability problem | Peter | akceptovaná |
| 1-56 | Vytvoriť retrospektívu za 1. šprint | Task | Všetci | akceptovaná |
| 1-59 | Prepojenie nástrojov pre správu kódu s projektom | Maintenance | Slavo | akceptovaná |

| ID | Názov | Typ | Riešitelia | Záver |
|---|---|---|---|---|
| 1-60 | Nastavenie profilu – upload obrázka | Task | Michal | akceptovaná |
| 1-61 | Nastavenie profilu - príprava migrácie | Task | Štefan | akceptovaná |
| 1-62 | Nastavenie profilu - príprava view + napojenie na controller | Task | Dušan | riešená |
| 1-63 | Vytvorenie udalosti v systéme | Task | Vlado | akceptovaná |
| 1-64 | Nastavenie monitorovania servera a aplikácie | Maintenance | Slavo | akceptovaná |
| 1-65 | Zobrazenie používateľského rozhrania pre administrátora experimentu (admin UI) | Task | Slavo | akceptovaná |
| 1-66 | Nastavenia detailov experimentu | Task | Slavo | akceptovaná |
| 1-67 | Nastavenie podmienok pre vstup používateľa do experimentu | Task | Slavo | akceptovaná |
| 1-68 | Detail experimentu - zobrazenie informácií | Task | Štefan | akceptovaná |
| 1-69 | Detail experimentu - prihlásenie do experimentu | Task | Štefan | akceptovaná |
| 1-71 | Po vyžiadaní obnovy hesla sa podarí prihlásiť aj neaktivovanému používateľovi | Bug | Peter, Slavo | akceptovaná |
| 1-72 | Vytváranie zdieľaných migrácií | Maintenance | Slavo | akceptovaná |
| 1-73 | Zobrazenie notifikácií | Task | Miro | akceptovaná |
| 1-74 | Rekonfigurácia SMTP | Maintenance | Slavo | akceptovaná |
| 1-75 | Správa experimentu – Značky | Task | Peter | akceptovaná |
| 1-76 | Správa experimentu – Budget | Task | Peter | akceptovaná |
| 1-77 | Správa experimentu – Správa participantov | Task | Peter | akceptovaná |
| 1-79 | Príprava unit testov pre model v šprinte 2 | Maintenance | Štefan | dokončená |

| ID | Názov | Typ | Riešitelia | Záver |
|---|---|---|---|---|
| 1-80 | Nastavenie servera na produkciu a staging | Maintenance | Slavo | akceptovaná |
| 1-81 | Integrácia systému – 2. šprint | Maintenance | Slavo | akceptovaná |

*Tabuľka 10.10: Prehľad úloh*

Na nasledujúcom grafe (Obrázok 10.12) môžeme vidieť distribúciu jednotlivých typov úloh medzi autorov.




*Obrázok 10.12: Distribúcia jednotlivých typov úloh medzi autorov*

### B.2.2 Časový prehľad

Tabuľka 10.11 obsahuje prehľad úloh a potrebného času na ich realizáciu pre jednotlivých členov tímu.

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| Dušan | Development | 1 | 11,00 |
| | Documentation | 3 | 16,97 |
| | Celkovo | 4 | 27,97 |
| Michal | Development | 3 | 8,81 |
| | Design | 2 | 5,82 |
| | Celkovo | 5 | 14,63 |
| Miroslav | Development | 1 | 16,50 |
| | Testing | 1 | 1,00 |
| | Documentation | 1 | 3,00 |
| | Celkovo | 3 | 20,50 |
| Peter | Development | 7 | 24,05 |
| | Testing | 3 | 2,90 |
| | Documentation | 2 | 3,29 |

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| | Design | 1 | 0,50 |
| | Celkovo | 13 | 30,24 |
| Slavomír | Development | 7 | 27,08 |
| | Testing | 1 | 0,50 |
| | Documentation | 2 | 13,50 |
| | Maintenance | 6 | 9,10 |
| | Celkovo | 16 | 50,18 |
| Vladimír | Development | 1 | 9,67 |
| | Testing | 1 | 4,29 |
| | Documentation | 2 | 4,39 |
| | Celkovo | 4 | 18,35 |
| Štefan | Development | 3 | 13,00 |
| | Testing | 1 | 3,00 |
| | Documentation | 1 | 3,00 |
| | Celkovo | 5 | 19,00 |

*Tabuľka 10.11: Časový prehľad úloh*

### B.2.3 Zhodnotenie

Počas hodnotenia šprintu č.2 sme sa na stretnutí vyjadrili k jednotlivým úlohám a problémom s nimi spojenými. Poznámky k úlohám sa nachádzajú v Tabuľka 10.12.

| ID úlohy | Popis | Poznámky |
|---|---|---|
| 1-49 | Detail experimentu | Dohodli sme sa na zmene názvu stĺpca “Stav zverejnenia” na “Stav”. |
| 1-51 | Nastavenie profilu | Zistili sme, že úlohu sa nepodarilo splniť v tomto šprinte. Dohodli sme sa, že skúsenosti používateľa budú ako voľný text + automatické |

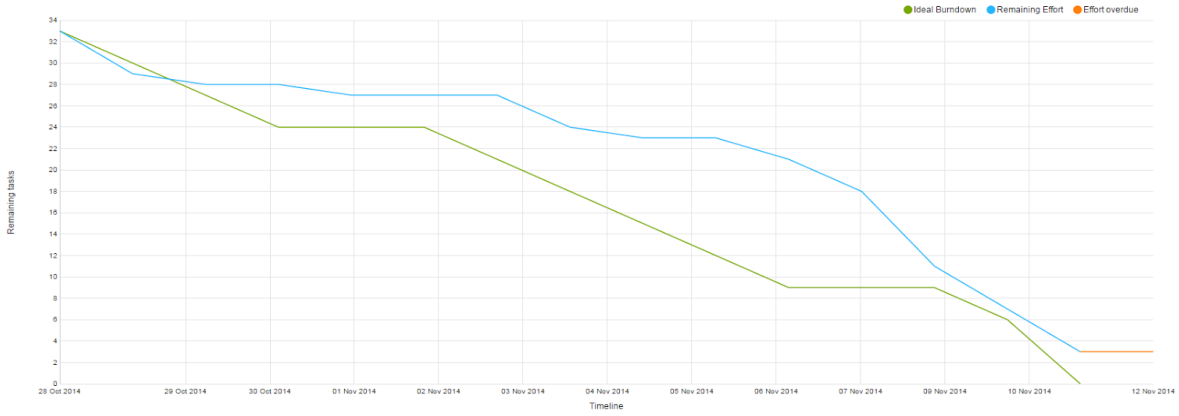
| ID úlohy | Popis | Poznámky |
|---|---|---|
| | | dopĺňanie textu. Zostáva dokončiť a doladiť fakturačnú a dodaciu adresu, doplniť pridávanie skúseností používateľa. |
| 1-52 | Úprava prekladov | Prebehla úprava prekladov, pričom je vytvorená metodika na implementovanie prekladov, ktorou sa musia riadiť všetci členovia tímu. |
| 1-44 | Správa experimentu | Dohodli sme, že je potrebné zmeniť systém pridávania podmienok pre experiment. |
| 1-46 | Notifikácie | Po zmenách vykonaných vo vytváraní experimentu sme sa dohodli, že notifikácia o vytvorení experimentu sa odošle až po jeho zverejnení a nie okamžite po vytvorení. |

*Tabuľka 10.12: Poznámky k úlohám*

Ďalšie úlohy, ktoré vyplynuli pre nasledujúce šprinty:

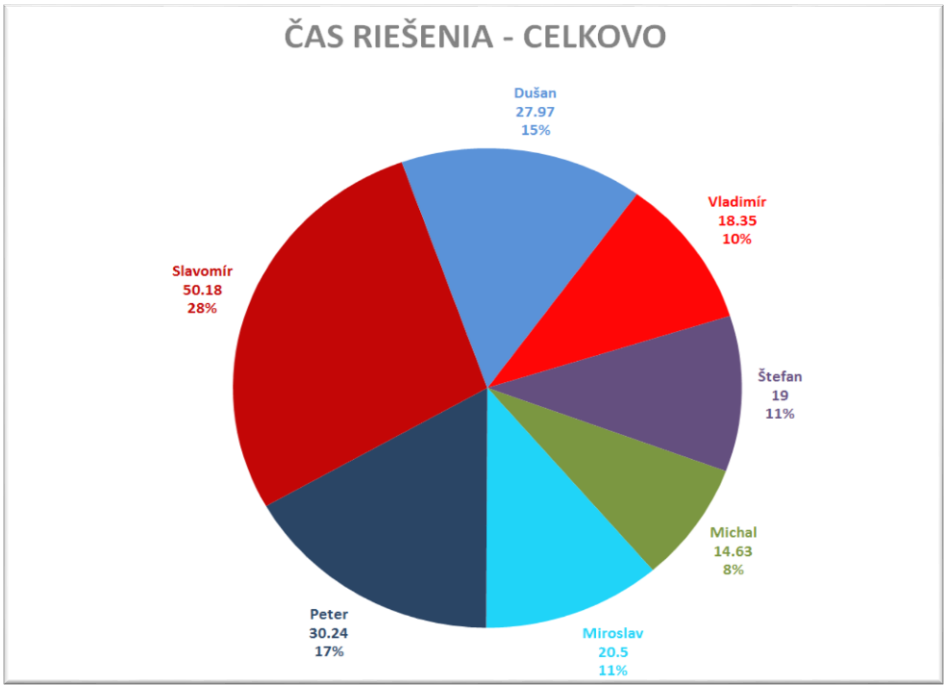
- Každý člen tímu sa musí riadiť metodikami, ktoré sme si zadefinovali.
- Zmeniť oznamovací text pri obnovení hesla, aby sme zamedzili zisťovaniu emailových adries, ktoré sú v systéme zaregistrované
- Zvážiť možnosť použitia aplikácie na monitorovanie serveru, ktorú vytvára tím č. 10.
- V nasledujúcom šprinte refaktorovať doteraz implementované funkcionality, pričom je potrebné sa riadiť vytvorenými metodikami.
- Upraviť dokumentáciu tak aby bola konzistentná s implementovanou funkcionalitou.
- Na návrh prof. Bielikovej je potrebné preskúmať konkurenčnú službu CrowdFlower spolu s ich API.
- Realizovať odporúčanie experimentov nielen pomocou priraďovania značiek, ale aj na základe informácií o predchádzajúcich experimentov používateľa.

### B.2.4 Príloha 1: Burndown Chart




*Obrázok 10.13: Burdown Chart za druhý šprint*

### B.2.5 Príloha 2: Distribúcia celkového času riešenia úloh medzi autorov




*Obrázok 10.14: Distribúcia celkového času riešenia medzi autorov*

## B.3 Retrospektíva k 3. šprintu

**Autori:** Michal Polko, Štefan Šmihla

**Trvanie šprintu:** 11.11. – 18.11.

### B.3.1 Zoznam úloh a stav ich riešenia

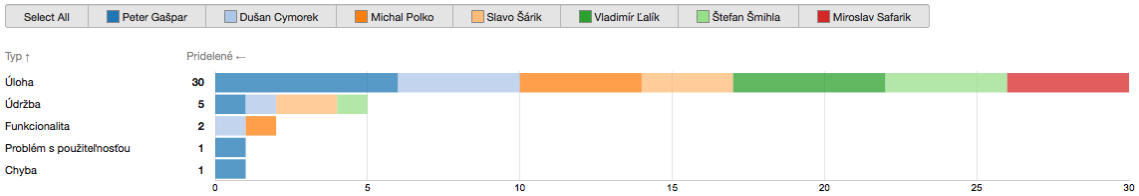
V Tabuľka 10.13 sa nachádza prehľad úloh, ktoré sme riešili počas tretieho šprintu, stav ich splnenia a poznámky z nich vyplývajúce. Náplňou šprintu bol refactoring existujúcej funkcionality a opravovanie chýb.

| ID | Názov | Typ | Riešitelia | Záver |
|---|---|---|---|---|
| 1-51 | Nastavenia profilu | Feature | Dušan, Michal | akceptovaná |
| 1-57 | Vytvorenie a odovzdanie dokumentácie – inžinierske dielo | Task | všetci | dokončená |
| 1-58 | Vytvorenie a odovzdanie dokumentácie – riadenie | Task | všetci | dokončená |
| 1-62 | Nastavenia profilu – príprava view + napojenie na controller | Task | Dušan | akceptovaná |
| 1-82 | Integrácia systému – 3. šprint | Maintenance | Slavo | akceptovaná |
| 1-83 | Refactoring – pridávanie experimentu | Task | Peter | akceptovaná |
| 1-84 | Chyba pri vyplnení registračného formulára | Bug | Peter | akceptovaná |
| 1-85 | Preformulovanie textov | Usability problem | Peter | akceptovaná |
| 1-87 | Refactoring – experiment | Maintenance | Štefan, Slavo | akceptovaná |
| 1-88 | Refactoring – participanti | Task | Peter | akceptovaná |
| 1-89 | Refactoring – dizajn | Task | Peter, Michal | akceptovaná |
| 1-90 | Refactoring – preklady | Maintenance | Peter, Dušan | akceptovaná |
| 1-91 | Priebežná príprava unit testov | Task | Štefan | akceptovaná |
| 1-92 | Refactoring – follow | Task | Vlado | akceptovaná |

| ID | Názov | Typ | Riešitelia | Záver |
|---|---|---|---|---|
| 1-93 | Refactoring – notifikácie | Task | Miro, Vlado | akceptovaná |
| 1-95 | Vytvoriť retrospektívu za 2. šprint | Task | všetci | akceptovaná |

*Tabuľka 10.13: Prehľad úloh*

Na nasledujúcom grafe (Obrázok 10.15) môžeme vidieť distribúciu jednotlivých typov úloh medzi autorov.




*Obrázok 10.15: Distribúcia jednotlivých typov úloh medzi autorov*

### B.3.2 Časový prehľad

Tabuľka 10.14 obsahuje prehľad úloh a potrebného času na ich realizáciu pre jednotlivých členov tímu.

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| Dušan | Development | 1 | 9,77 |
| | Documentation | 2 | 7,13 |
| | Maintentance | 1 | 0,85 |
| | Celkovo | 4 | 19,75 |
| Michal | Documentation | 2 | 4,03 |
| | Design | 1 | 2,13 |
| | Celkovo | 3 | 6,17 |
| Miroslav | Development | 1 | 1,25 |
| | Documentation | 2 | 6,75 |
| | Celkovo | 3 | 8,00 |

| Riešiteľ | Typ | Počet úloh | Počet hodín |
|---|---|---|---|
| Peter | Development | 4 | 8,82 |
| | Documentation | 2 | 8,88 |
| | Design | 1 | 1,97 |
| | Celkovo | 7 | 19,67 |
| Slavomír | Development | 1 | 9,58 |
| | Documentation | 3 | 7,62 |
| | Maintenance | 1 | 1,5 |
| | Celkovo | 5 | 18,7 |
| Vladimír | Development | 2 | 4,58 |
| | Documentation | 5 | 5,33 |
| | Celkovo | 7 | 9,92 |
| Štefan | Development | 4 | 5,00 |
| | Documentation | 2 | 1,50 |
| | Celkovo | 6 | 6,5 |

*Tabuľka 10.14: Časový prehľad úloh*

### B.3.3 Zhodnotenie

Počas hodnotenia šprintu č. 3 sme sa na stretnutí vyjadrili k jednotlivým úlohám a problémom s nimi spojenými. Poznámky k úlohám sa nachádzajú v Tabuľka 10.15.

| ID úlohy | Popis | Poznámky |
|---|---|---|
| 1-83 | Refactoring – pridávanie experimentu | Dohodli sme sa, že pridávanie experimentu bude riešené cez modálne okno, nie Ajax. |
| 1-90 | Refactoring – preklady | Texty na stránke boli preformulované a smajlíky boli odstránené. |
| 1-89 | Refactoring – dizajn | Na stretnutí prebehli krátke diskusie k návrhom na zlepšenie GUI. |

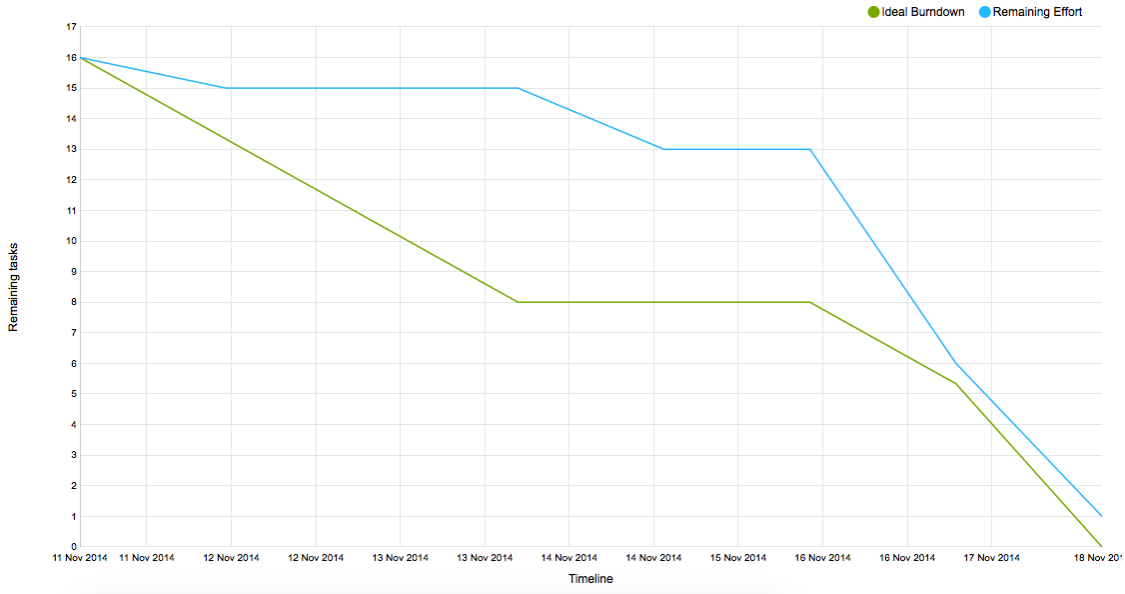
| ID úlohy | Popis | Poznámky |
|---|---|---|
| 1-87 | Refactoring – experiment | Pridávanie používateľa do experimentu, pri požadovanom vzdelaní sa zatiaľ rieši podmienkou zhody. |
| 1-51 | Dokončenie nastavení profilu | Úloha bola úspešne dokončená z predchádzajúceho šprintu. Bude však potrebné upraviť priečinok pre používateľské fotografie na produkcii. |
| 1-91 | Priebežná príprava unit testov | Testy sú v súčasnosti funkčné a takmer kompletne pokrývajú registráciu, prihlásenie a reset hesla. Ostatné funkcionality sú pokryté len čiastočne. |
| 1-57<br>1-58 | Dokumentácia | Dohodli sme sa na používaní LaTeX pre písanie dokumentácie. |

*Tabuľka 10.15: Poznámky k úlohám*

Ďalšie úlohy, ktoré vyplynuli pre nasledujúce šprinty:

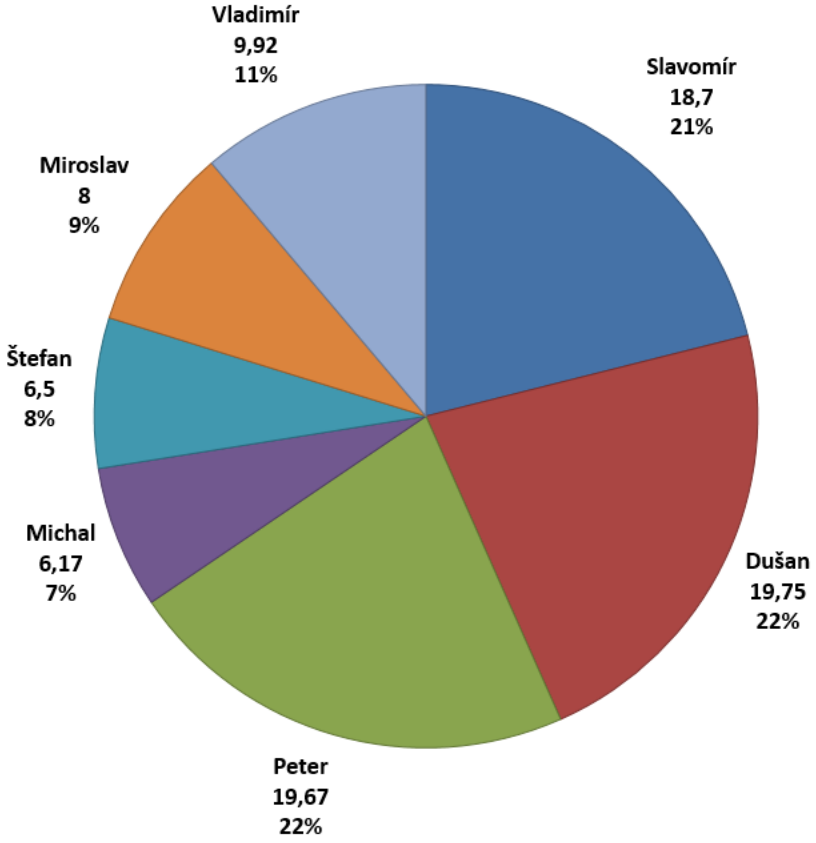
- Pozrieť nastavenia YouTracku týkajúce sa Burndown Chart – v súčasnom stave sa graf mení len ak sa úloha označí za Fixed. Nereflektuje počet strávených hodín pri práci na úlohách, čo vyúsťuje vo veľkej odchýlke voči ideálnemu stavu.
- Nasadiť *continuous integration* na server pre zvýšenie kvality kódu
- Prejsť ponuku *team collaboration* portálov, nasadiť u nás a presunúť tam metodiky a návody z tímového Google Drive.

### B.3.4 Príloha 1: Burndown Chart




*Obrázok 10.16: Burdown Chart za tretí šprint*

### B.3.5 Príloha 2: Distribúcia celkového času riešenia úloh medzi autorov




*Obrázok 10.17: Distribúcia celkového času riešenia medzi autorov*