



Sofistikované spracovanie dát

Metodiky

Vedúci tímu: Ing. Michal Holub

Členovia tímu: Bc. Igor Daniš, Bc. Jakub Kmeňko, Bc. Martin Košut, Bc. Martin Lošák, Bc. Stanislav Paľove, Bc. Alex Ostrovský, Bc. Peter uherek

Akademický rok: 2014/2015

1	MANAŽMENT MONITOROVANIA.....	5
1.1	Metodika technickej podpory pre jednotlivé činnosti.....	5
1.1.1	Komunikácia	5
1.1.2	Sledovanie úloh	5
1.1.3	Informácie o tíme.....	5
1.1.4	Verziovanie systému a codereview.....	5
1.1.5	Vývoj.....	5
1.2	Metodika monitorovania prehliadky vytváraného výsledku.....	5
1.3	Metodika stanovenia sledovaných charakteristík produktu.....	6
1.3.1	Flog	6
1.3.2	Flay.....	6
1.4	Metodika konvencie písania zdrojového kódu v Ruby	6
1.4.1	Všeobecné zásady pre písanie zdrojového kódu	6
1.4.2	Rozloženie zdrojového kódu	7
1.4.3	Syntax	9
1.4.4	Pomenúvanie	10
1.4.5	Komentáre.....	11
2	MANAŽMENT KVALITY	12
2.1	Metodika reportovania bugov	12
2.1.1	Reportovanie bugu	12
2.1.2	Bug report – čo má obsahovať.....	12
2.1.3	Pridelenie bugu	14
2.2	Opravenie bugu.....	14
2.2.1	Duplicate	14
2.2.2	Won't fix	14
2.2.3	Incomplete	14
2.2.4	Cannot Reproduce	14
2.2.5	Fixed	15
2.3	Uzatvorenie bugu	15
2.4	Metodika testov	15
2.4.1	Písania testov	15
2.4.2	Vyhodnotenie testov	15
2.4.3	Code review	15
2.4.4	Kto komu robí code review	15
2.4.5	Na čo pozerieť pri code review	16
2.5	Metodika údržby softvéru (odovzdaných častí)	16
2.6	Metodika riadenia požiadaviek na zmenu	16
3	MANAŽMENT RIZÍK	18
3.1	Metorika identifikácie a riešenia rizík	18
3.1.1	Dedikácia	18

3.1.2	Zoznam súvisiacich metodík.....	18
3.1.3	Vymedzenie pojmov	18
3.1.4	Vymedzenie rolí a zodpovedností účastníkov (Rola - Zodpovednosť)	18
3.1.5	Názov – Popis	18
3.1.6	Identifikované riziká podľa dopadu na projekt:	19
3.1.7	5. Neúčast člena tímu na tímovom stretnutí.....	20
3.1.8	Stupeň rizika: 0.3.....	20
4	MANAŽMENT ROZVRHU A PLÁNOVANIA	21
4.1	Metodika plánovania pre tím a jednotlivých členov tímu	21
4.1.1	Vymedzenie pojmov a skratiek	21
4.1.2	Zoznam nadväzujúcich metodík.....	21
4.1.3	Product Backlog	21
4.1.4	Šprint	21
4.1.5	Sprint Backlog.....	21
4.1.6	Tímové stretnutia.....	22
4.1.7	Stretnutie na rozhraní šprintu	22
4.1.8	Stretnutie v strede šprintu	23
4.1.9	Podporné nástroje	23
4.2	Metodika vyhodnocovania plnenia plánu a návrh úprav	25
4.2.1	Vyhodnocovanie plnenia plánu	25
4.2.2	Návrh úprav.....	25
4.3	Metodika udržiavania informácií o stave projektu	26
5	MANAŽMENT PODPORY VÝVOJA	27
5.1	Metodika manažmentu verzií	27
5.1.1	Vymedzenie pojmov a skratiek	27
5.1.2	Zoznam nadväzujúcich metodík a dokumentov.....	27
5.1.3	Vetvenie zdrojového kódu	28
5.1.4	Funkcionálne vetvy	29
5.1.5	Pomenovávanie odovzdaní	30
5.1.6	Príloha: Vizuálna schéma vetvenia.....	31
5.2	Metodika manažmentu softvérovej konfigurácie	31
5.2.1	Profil Test	32
5.2.2	Profil Vývoj	32
5.2.3	Profil Produkcia	32
5.2.4	Proces rozšírenia softvérovej konfigurácie projektu.....	32
5.3	Metodika integrácie softvéru	32
5.3.1	Integrácia na úrovni rozšírenia zdrojového kódu.....	32
5.3.2	Integrácia na úrovni verzie vytvorenej webovej aplikácie	33
5.4	Metodika vyhodnocovania plnenia plánu a návrh úprav	33
5.4.1	Vyhodnocovanie plnenia plánu	33
5.4.2	Návrh úprav.....	33
6	MANAŽMENT KOMUNIKÁCIE A ĽUDSKÝCH ZDROJOV	34

6.1	Metodika organizácie komunikácie v tíme.....	34
6.1.1	Zoznam nadväzujúcich metodík.....	34
6.1.2	Vymedzenie pojmov	34
6.1.3	Vymedzenie roly	34
6.1.4	Formálna komunikácia	34
6.1.5	Neformálna komunikácia	36
6.1.6	Hlavné komunikačné kanály.....	36
6.1.7	Vedľajšie komunikačné kanály	37
6.1.8	Kontaktovanie člena tímu.....	38
6.1.9	Informovanie vedúceho tímového projektu	38
6.2	Metodika informovania učiteľa o stave projektu	38
6.3	Metodika evidencie úloh	39
6.3.1	Zadávanie úloh v rámci šprintu	39
6.3.2	Evidencia úloh počas šprintu.....	40
6.3.3	Evidencia úloh na konci šprintu	40
6.4	Metodika organizácie zdrojov.....	40
6.4.1	Plánovanie organizácie	41
7	MANAŽMENT DOKUMENTOVANIA.....	43
7.1	Metodika riadenia procesu dokumentovania (verzia 1).....	43
7.1.1	Skratky a pojmy	43
7.1.2	Vysvetlenie pojmov.....	43
7.1.3	Postup dokumentovania	43
7.1.4	Spáva verzií dokumentov.....	44
7.1.5	Správa a formátovanie dokumentov	45
7.1.6	Formátovanie dokumentov	45
7.1.7	Tvorba dokumentov.....	46
7.1.8	Šablóna pre tvorbu zápisníc z tímového stretnutia	47
7.2	Metodika riadenia procesu dokumentovania (verzia 2).....	49
7.2.1	Skratky a pojmy	49
7.2.2	Vysvetlenie pojmov.....	49
7.2.3	Postup dokumentovania	49
7.2.4	Spáva verzií dokumentov.....	50
7.2.5	Správa a formátovanie dokumentov	51
7.2.6	Formátovanie dokumentov	51
7.2.7	Dokumentácia modulu	52
7.2.8	Názov úlohy (z User Story)	52
7.2.9	Šablóna pre tvorbu zápisníc z tímového stretnutia	53

1 Manažment monitorovania

1.1 Metodika technickej podpory pre jednotlivé činnosti

1.1.1 Komunikácia

Pre potreby komunikácie sa využíva produkt HipChat od firmy Atlassian. HipChat je voľne dostupný na stránkach : <https://www.atlassian.com/software/hipchat>. Pre použitie HipChatu je nutná registrácia. HipChat je dostupný v podobe webovej, desktopovej a mobilnej aplikácie. HipChat umožňuje vytváranie miestností pre jednotlivé témy čím sprehľadňuje komunikáciu.

1.1.2 Sledovanie úloh

Pre sledovanie úloh využívame produkt JIRA od firmy Atlassian. JIRA je webová aplikácia umožňujúca zadávanie úloh a ich priebehu. Pre využitie JIRA je nutné mať vytvorený účet na stránkach informačného systému STU. Inštancia JIRA je dostupná na stránke : <http://jira.fiit.stuba.sk/>

1.1.3 Informácie o tíme

Pre ukladanie metodík a iných informácií dôležitých pre fungovanie tímu využívame wiki aplikáciu Wikia. Naša wiki stránka je dostupná na <http://sk.datapoints.wikia.com/>

1.1.4 Verziovanie systému a codereview

Pre vyvíjanie jednotlivých verzií používame online úložisko na portály Github dostupné na: <https://github.com/DataPoints/DataPoints> . Portál Github využívame aj na codereview pre jeho schopnosť zobrazenia rozdielov v jednotlivých verziách, čo umožňuje písanie review len k relevantným častiam kódu.

1.1.5 Vývoj

Pre vývoj využívame IDE RubyMine, ktoré poskytuje licenciu zadarmo pre študentov. Je dostupné na stránke: <https://www.jetbrains.com/ruby/>

1.2 Metodika monitorovania prehliadky vytváraného výsledku

Pri vytváraní produktu sa sústreďujeme na jeho výslednú kvalitu. Sledujeme rozsah kódu ako aj jeho kvalitu či neobsahuje nadbytočné časti alebo príliš zložité časti.

Pre zložitosť využívame nástroj Flog poskytujúci pohľad na komplexnosť jednotlivých metód. V prípade, že metóda dosiahne skóre 20 bodov a vyššie je nutné preskúmať dôvod na jej komplexnosť ak je dôvod irelevantný je metódu nutné refaktorovať. Metódy s číslom väčším ako 50 sú považované za nebezpečné a je nutné ich refaktorovať.

Pre odstránenie duplicitného kódu používame nástroj Flay, ktorý reportuje duplicity a podobnosti v kóde. Pri nahlásení duplicity sa preverí, či daný kód má svoje jednoznačné jedinečné využitie a tým pádom sa nejedná o duplicitu.

Tieto metriky sledujeme pri code review kde nás výsledky môžu nasmerovať k častiam kódu, ktorý potrebuje refaktorovanie.

1.3 Metodika stanovenia sledovaných charakteristík produktu

1.3.1 Flog

Pomocou gemu flog sledujeme komplexitu jednotlivých metód v našom kóde. Flog predstavuje statickú analýzu kódu. Komplexita je vyjadrená v bezrozmernej jednotke. Čím vyšší výsledok tým vyššia je komplexita skúmaného kódu. Vo všeobecnosti metóda, ktorá dosiahne výsledok 20 sa považuje za vhodnú na refaktorovanie. Metódy ktoré dosahujú skóre 60 a vyššie sa považujú za nebezpečné.

1.3.2 Flay

Pomocou gemu flay sledujeme duplicitu kódu v našej aplikácii. Flay vyhľadáva kód, ktorý je výsledkom kopírovania alebo kódu, ktorý je podobný a líši sa len v špecifických identifikátoroch a použitých konštantách

1.4 Metodika konvencie písania zdrojového kódu v Ruby

Tento dokument je slúži ako záväzná štábná kultúra pre písanie v jazyku Ruby a k nemu prislúchajúcemu frameworku Ruby on Rails. Tento dokument vznikol na podporu zavedenia jednotného štýlu písania zdrojového kódu v tímovom projekte.

Tento dokument vychádza z už dostupných príručiek pre Ruby

Tieto príručky sú dostupné na:

1. <https://github.com/bbatsov/ruby-style-guide>
2. <http://naildrivin5.com/ruby-style/>

V nasledujúcej sekcii sú zásady pre písanie zdrojového kódu rozdelené do jednotlivých sekcií. V sekciách sú v bodoch uvedené jednotlivé zásady. Pre niektoré zásady je priamo pod nimi uvedená ukážka výzoru zdrojového kódu.

1.4.1 Všeobecné zásady pre písanie zdrojového kódu

- Zdrojový kód píšeme v anglickom jazyku
- Zdrojový kód píšeme tak aby bol samo dokumentujúci a nebolo nutné siahnuť po dokumentácii samotnej.
- Kód píšeme tak aby jednotlivé riadky kódu nepresiahli dĺžku 80 znakov. Vizuálna pomôcka nasledujúci riadok má presne 80 znakov. Pre ľahšie dodržiavanie odporúčam nakopírovať do zdrojového kódu počas písania.

```
#=====
=====
```

- V prípade, že potrebujeme zadať dlhý string použijeme znak na pokračovanie riadku \. Tento znak použijeme len v prípade stringu.

```
dlhy_string = 'Prva cast stringu' \
              'druha cast stringu' \
              'tretia cast stringu'
```

- V prípade volania dlhej metódy odsadíme každý argument na nový riadok v rovnakej úrovni. Ukončujúcu zátvorku odsadíme na nový riadok na úroveň metódy.

```
#zle (dlha externa metoda)
def send_mail(source)
  Mailer.deliver(to: 'bob@example.com', from: 'us@example.com', subject: 'Import
end

# dobre
def send_mail_ok(source)
  Mailer.deliver(
    to: 'bob@example.com',
    from: 'us@example.com',
    subject: 'Important message',
    body: source.text
  )
end
```

- V prípade poľa, ktoré obsahuje veľa prvkov a nezmestia sa do 80 znakov pole rozdelíme. Prvky ktoré sa nezmestili do jedného riadku dáme do druhého riadku a zarovnáme s prvým prvkom v poli.

```
# zle
example_array = ['example', 'example', 'example', 'example', 'example', 'example']

# dobre
example_array_two =
  ['example', 'example', 'example', 'example', 'example', 'example',
   'example', 'example', 'example']
```

1.4.2 Rozloženie zdrojového kódu

- Pri písaní zdrojového kódu v ruby používame dve medzery pre každú úroveň odsadenia.

```
# zle 4 medzery (JAVA style)
def some_method_one
  do_something
end

# dobre
def some_method_two
  do_something
end
```

- Nepoužívame bodkočiarku na konci riadku ani na oddelenie výrazov. Výraz vždy píšeme na nový riadok.

```
# zle
puts 'example'; # nadbytočná čiarka

puts 'foo'; puts 'bar' # dva výrazi v jednom riadku

# dobre
puts 'foobar'
```

- Nepoužívame jednoriadkové metódy.

```
# zle
def too_much; something; something_else; end

# dobre
def some_method
  body
end
```

- Používame medzery pre oddelenie operátorov, po čiarkach a bodkočiarkach

```
sum = 1 + 2
a, b = 1, 2
[1, 2, 3].each {|e| puts e}
```

- Žiadne medzery pred a po zátvorkách okrem kučeravých zátvoriek, ktoré reprezentujú blok.

```
def some_method(arg1, arg2)
  body
end

[1, 2, 3].each {|e| puts e}
```

- Žiadne medzery pred !

```
#zle
! some_method

# dobre
!some_method
```

- Žiadne medzery medzi rozsahy

```
# zle
0 .. 9
'a' ... 'z'

# dobre
0..9
'a'..'z'
```

- Jednotlivé metódy sa od seba oddeľujú prázdny riadkom. Definíciu metódy píšeme do jedného riadku a za posledným argumentom nepíšeme čiarku. Pri priraďovaní predvolenej hodnoty píšeme medzeru pred aj za znak =

```
def some_method_one
  body
end

def some_method_two(arg1, arg2)
  body
end

def some_method_three(arg1 = 25, arg2 = "example")
  body
end
```


- Pre výraz `case` odsadíme `when` o jednu úroveň.

```
# dobre
case
  when song.name == 'Misty'
    puts 'Not again!'
  when song.duration > 120
    puts 'Too long!'
  when Time.now.hour > 21
    puts "It's too late"
  else
    song.play
end
```

- Pri priradovaní z podmienky odsadíme podmienený výraz o jednu úroveň

```
# dobre
kind =
  case year
    when 1850..1889 then 'Blues'
    when 1890..1909 then 'Ragtime'
    else 'Jazz'
  end

result =
  if some_cond
    calc_something
  else
    calc_something_else
  end
```

1.4.3 Syntax

- Pri definovaní metódy používame zátvorky len ak má argumenty. Ak má metóda argumenty zátvorky používame vždy

```
# dobre
def some_method_one()
  # body omitted
end

# dobre
def some_method
  # body omitted
end

# dobre
def some_method_with_parameters(arg1, arg2)
  # body omitted
end
```

- Pri písaní `if/else` výrazu podmienku vždy píšeme na rovnaký riadok

```
# zle
if
  some_condition
end

# dobre
if some_condition
end
```

- Nepoužívame `and` a `or` kľúčové slová. Namiesto nich používame `&&` a `||`.
- Vynecháme zátvorky pre volania metód bez argumentov
- Využívame skrátene operátory priradenia

```
# zle
x = x + y
x = x * y
x = x**y
x = x / y
x = x || y
x = x && y

# dobre
x += y
x *= y
x **= y
x /= y
x ||= y
x &&= y
```

- Využívame metódy pre overenie, či je číslo párne alebo nepárne.

```
# zle
if x % 2 == 0
end

if x % 2 == 1
end

# dobre
if x.even?
end

if x.odd?
end
```

1.4.4 Pomenúvanie

Pri pomenúvaní nepoužívame jednopísmenkové názvy. Namiesto toho používame opisné názvy presne vyjadrujúce účel danej veci.

Príklad:

`a = 96` # Nič nevraviaca premenná, ktorá má hodnotu 96

`pocet_uchadzacov` # Premenná, ktorej hodnota 96 predstavuje počet uchádzačov

Pre pomenovanie symbolov, metód a premenných používame `snake_case`. Pre pomenovanie konštánt používame `SCREAMING_SNAKE_CASE`. Pre pomenovanie tried

a modulov používame CamelCase.

```
SOME_CONSTANT = "example"

class SomeClass

  def some_method

    some_variable
    body
  end
end
```

1.4.5 Komentáre

- Komentár píšeme vždy ihneď nad príslušnú časť kódu, ktorej sa týka.
- Použijeme jednu medzeru pre oddelenie komentára od začiatočného znaku #.
- Nepíšeme zrejmé komentáre

```
# sum je výsledkom scitania d a b
sum = d + b
```

- Existujúca komentáre udržujeme aktuálne, aby sme predišli dezinformácii.
- Komentáre nepoužívame na vysvetlenie zlého kódu. Zlý kód sa refaktoruje.
- Nepoužívame viacriadkové komentáre. Viacriadkový komentár napíšeme pomocou jednoriadkových komentárov.

```
# zle
=begin
riadok komentu
dalsi riadok komentu
=end

# dobre
# riadok komentu
# dalsi riadok komentu
```

Na začiatku každého zdrojového súboru bude vytvorená hlavička obsahujúca základné informácie o súbore. Informácie obsiahnuté v hlavičke: autor, dátum vytvorenia, dátum poslednej úpravy, meno človeka, ktorý vykonal úpravu, krátky popis o čom daný súbor je

Forma hlavičky :

```
# Name of Author: (Meno autora)
# Created at: (Datum vytvorenia)
#
# Updated by: (Meno osoby, ktorá vykonala posledné zmeny)
# Updated at: (Dátum vykonania posledných zmien)
#
# Description: (Krátky opis funkcie zdrojového kódu)
# Other: (Iné veci, ktoré môžu byť zaujímavé napr. externé závislosti, použité gemy)
```

2 Manažment kvality

2.1 Metodika reportovania bugov

2.1.1 Reportovanie bugu

Ak je odhalený bug v programe treba vytvoriť bug report ktoré sa reportujú v nástroji Jira. V ňom treba v hornej lište vybrať možnosť create a následne vyplniť nasledujúce údaje:

1. Project - DataPoints
2. Issue Type – Bug
3. Summary – vyplnenie názvu bugu
4. Priority – zvolenie priority bugu
5. Assignee – pridelenie bugu zodpovednej osobe
6. Enviroment – vyplnenie prostredia na ktorom sa daný bug vyskytol
7. Description – v tomto poli treba popísať časti bugu verzia programu, možnosť zreprodukovania, popis, kroky na zreprodukovanie bugu a očakávaný výsledok. Treba ich písať v tomto poradí a tak aby ich bolo v popise jasne a čitateľne vidno.
8. Attechment – sem treba vkladať prílohy
9. Sprint – zvoliť aktuálny Sprint v ktorom sa bug vyskytol

Po správnom vyplnení všetkých políček sa zvolí tlačidlo create pre vytvorenie bug reportu do systému.

2.1.2 Bug report – čo má obsahovať

2.1.2.1 Názov

Vytvoriť krátky ale jasný názov pre bug aby z neho bolo jasné o čo v ňom ide a aby sa dalo jednoducho podľa názvov bugov vyhľadávať aby nedochádzalo k duplicitám. Názvy bugov treba písať v slovenčine a bez diakritiky.

- **Zle:** “Spadlo to”, “Videl som error”, “Bug”
- **Dobre:** “Registracia neposiela potvrdzujuci mail”

2.1.2.2 Verzia

Identifikovanie verzie na ktorej sa bug nachádza.

- **Zle:** “Naša aplikacia”
- **Dobre:** “DataPointa v1.05”

2.1.2.3 Klasifikácia

Potrebuje vedieť ako závažný je tento bug, či ide len o maličkosť ako napríklad, vyskakovacie okno je o 10px väčšie ako má byť alebo o závažnú chybu ako, že nefunguje registrácia pretože nepošle autentifikačný email.

- **Zle:** nevyplniť
- **Dobre:** “Critical”, “Minor”, “Trivial”

2.1.2.4 Platforma

Potrebuje vedieť na akej platforme sa daný bug vyskytol. Keďže u nás ide o webovú aplikáciu tak potrebujeme vedieť operačný systém a webový prehliadač.

- **Zle:** nenapísať
- **Dobre:** “Windows 8, Internet Explorer 9”

2.1.2.5 Možnosť zreprodukovania

Potrebuje vedieť či sa daný bug vyskytuje vždy alebo len niekedy.

- **Zle:** nenapísať
- **Dobre:** “Za každým”, “Niekedy”, “Nemožno zreprodukovat”

2.1.2.6 Popis

Rýchly popis v prirodzenom jazyku čo si sa pokúšal spraviť keď nastal problém. Začatie s kontextom, kde si sa nachádzal v aplikácii a zamerať sa na to čo si urobil a čo urobila aplikácia.

- **Zle:** “Aplikácia nefunguje”
- **Dobre:** “Vyplnil som správne registračný formulár a klikol na tlačidlo registrovať kde mi malo zaslať aktivačný Email na mnou zadanú adresu ale nestalo sa tak.”

2.1.2.7 Kroky na zreprodukovanie bugu

Ak je možné tento bug zreprodukovat treba popísať presné kroky, ktoré treba vykonať aby nastal. Pri rýchlo krokoch treba byť úplne presný pretože je rozdiel “Klikol som na X”, “Dvoj klikol som na X”, “Klikol som na X a potom stlačil enter”.

- **Zle:** “Skúsil som sa registrovať”
- **Dobre:**
 1. Vyplnil som do registračného formulára tieto údaje: meno: xxx heslo: xxx@xxx.sk
 2. Stlačil som tlačidlo ‘Register’
 3. Skontroloval som mail a neprišiel

2.1.2.8 Očakávaný výsledok

Popísanie čo bol očakávaný výsledok, ktorý mal nastať pri správnom fungovaní programu. Ak sú kroky rovnaké ako v predchádzajúcom bode stačí napísať číslo v ktorom je rozdiel medzi očakávaným výsledkom a reálnym.

- **Zle:** “Neposlalo autentifikačný email”
- **Dobre:**
 - 3. Skontroloval som mail a nachádzal sa tam autentifikačný email
 - 4. Klikol som na link v maily pre aktiváciu účtu
 - 5. Presmerovalo ma na webovú stránku a objavilo sa hlásenie, že môj účet bol aktivovaný

2.1.2.9 Prílohy

Ak je k danému bugu vhodné spraviť screenshot pre jednoduchšie porozumenie tak ho treba spraviť a poslať v prílohe.

2.1.3 Pridelenie bugu

Bug treba pridelovať človeku kto mal na starosti danú funkcionality. Ak nevieš kto ju mal na starosti treba to prideliť na Martina Lošáka a ten následne rozdelí bugy kompetentným osobám.

2.2 Opravenie bugu

Po vytvorení bug reportu je daný bug pridelený osobe ktorá pracovala na danej funkcionalite na opravu. Keď sa na oprave začne pracovať treba v Jire pri danom bugu zvoliť možnosť Start Progress. Pri oprave si treba logovať hodiny rovnakým spôsobom ako je zadefinované pre pracovanie s taskami pre náš projekt.

Keď je práca s bugom hotová v Jire treba zvoliť Resolve Issue a zvoliť jednu z možností Resolution.

2.2.1 Duplicate

Táto možnosť znamená, že daný bug už bol raz napísaný a toto je duplicita. V tomto prípade treba do komentáru napísať číslo bugu ktorý je rovnaký ako tento a priradiť ho spať na osobu ktorá ho zadávala nech overí či ide naozaj o duplicitu a ak áno tak nechať len jeden správny bug report v systéme. Ak ide o duplicitu už uzatvoreného bugu tak ho treba znova otvoriť a poslať kompetentnej osobe na opravu.

2.2.2 Won't fix

Ak bol bug pridelený zlej osobe tak treba zvoliť túto možnosť a poslať manažérovi plánovania Martinovi Lošákovy na správne pridelenie. Do komentáru treba napísať, že bol bug nesprávne pridelený.

2.2.3 Incomplete

Táto možnosť označuje, že sa v bug reporte nenachádzajú všetky informácie ktoré má obsahovať alebo je s nimi nejaký problém. Do komentáru treba napísať aké informácie v ňom chýbajú a prideliť ho spať na osobu ktorá daný bug report vytvorila.

2.2.4 Cannot Reproduce

Keď nie je možné zadaný bug zreprodukovať tak treba zvoliť túto možnosť a do komentáru treba napísať prečo ho nie je možné zreprodukovať ako napríklad, že sa daný bug vyskytuje iba na Windowse a daný človek nemá k tomuto operačnému systému prístup alebo ak z nejakého dôvodu nie je možné bug podľa napísaných inštrukcií na danom stroji zreprodukovať. Prideliť ho treba na Martina Lošáka, ktorý bude následne riadiť jeho pridelenie.

2.2.5 Fixed

Ak bol bug opravený programátorom a u neho všetko už funguje ako má tak treba zvoliť možnosť fixed. Do komentáru treba napísať, že sa podarilo bug opraviť a ak sa niečo zmenilo v postupe ako sa daná funkcionálna vykonáva tak to treba napísať tiež. Pridelenie ide osobe ktorá bug report napísala.

2.3 Uzatvorenie bugu

Ak bol bug opravený a nastavený do stavu fixed a poslaný späť osobe ktorá daný bug napísala treba ho ešte otestovať či bol naozaj opravený. Ak sa bug stále vyskytuje tak je znova poslaný kompetentnej osobe na opravu s komentárom, že bug stále pretrváva ak je v rovnakej podobe a ak sa v niečom líši tak to treba napísať tiež. Ak sa zmenil postup pri zreprodukovávaní bugu treba ho znova popísať v komentári. Ak je bug otestovaný a je opravený tak so posunie do stavu done a následne sa spraví testovací scenár ktorý v sebe pokrýva daný bug.

2.4 Metodika testov

2.4.1 Písania testov

Do konca tretieho sprintu nemusia žiadny z členov tímu písať automatizované testy a to z dôvodu, že zatiaľ neovládame Ruby on Rails na toľko aby sme stihli napísať kód a k nemu príslušné automatické testy. Postačí ak každý spraví svoj task a vyskúša si ho a ak nenájde žiadne chyby tak pošle na code review. Od začiatku štvrtého sprintu bude dostupná metodika na písanie automatizovaných testov, ktoré si bude každý člen tímu písať samostatne.

2.4.2 Vyhodnotenie testov

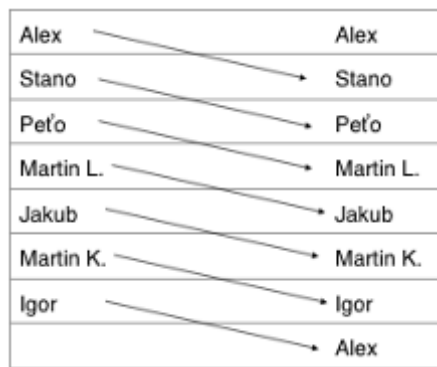
Táto časť bude pridaná až po zavedení automatizovaných testov vo štvrtom šprinte. Do vtedy si testujeme funkčnosť aplikácie všetci navzájom a keď niekto odhalí bug tak ho reportne podľa metodiky písania bugov.

2.4.3 Code review

Každá časť kódu ktorá sa má pushnúť do master breche musí prejsť code review. Každý člen bude robiť code review inému členovi tímu a tot poradie sa bude meniť každý šprint.

2.4.4 Kto komu robí code review

To že kto komu robí code review sa mení každým šprintom a to nasledovne:



Obrázok 4 – Code reviewers

Tento obrázok ukazuje kto komu robí code review v šprinte číslo 3. Tento stav berieme ako východiskový. V každom ďalšom šprinte si každý posunie šipku o jedno nižšie a tej osobe robí code review v danom šprinte. A takto to bude pokračovať každý šprint takže vždy robí jeden človek niekomu inému code review ako robil v predchádzajúcom šprinte.

2.4.5 Na čo pozerať pri code review

Keď sa robí code review treba dávať pozor na nasledujúce veci:

1. Či je dodržiavaná metodika na písanie kódu, ako napríklad mená premenných, písanie komentárov. Všetko z tejto metodiky musí byť splnené.
2. Či je daná časť kódu plne funkčná. Aj keď sa posielanie na code review vykonáva až keď si myslíme, že všetko funguje je potrebné aby to odskúšal aj člen tímu ktorý píše code review.
3. Či je potrebný refactoring alebo nie. Ak by sa niečo dalo napísať v ruby on rails jednoduchšie a refactoring kódu má zmysel tak aj toto treba napísať do core review.
4. Či je daná časť podľa očakávaní alebo tam niečo ešte chýba a treba to dorobiť. Ak sú nejasnosti v tomto bode treba kontaktovať manažéra kvality (Alex), ktorý to tiež pozrie a vyjadrí sa.

2.5 Metodika údržby softvéru (odovzdaných častí)

Každého časť ktorú naprogramoval sa považuje za hotovú len vtedy ak bolo k nej napísané správne code review, nenachádza sa žiaden bug k tejto časti v Jire a je k nej spravená kompletná dokumentácia. Až za týchto podmienok sa považuje programovací task v Jire za hotový a môže sa kód pushnúť z dev brenche do master na githube. Branch hotfix je v tomto prípade výnimkou kedy bol odhalený závažný problém ktorý treba okamžite opraviť na produkčnom serveri tak úlohy z tejto brenche nemusia spĺňať všetky podmienky ale musí byť dostatočne overené (minimálne dvoma členmi tímu), že to funguje správne a až potom je možné spraviť push na master brench.

2.6 Metodika riadenia požiadaviek na zmenu

Ak ma niektorý z členov tímu požiadavku na zmenu niektorej z funkcionality aplikácie, musí o tom informovať celý tím na HipChate v miestnosti Diginalninas aby mal sa mal možnosť k tomu vyjadriť každý. Následne nastane debata ohľadom tejto požiadavky. Či s tým všetci súhlasia alebo je niekto proti tomu. Ak je niekto proti tomu tak treba napísať aj dôvod prečo

nie. Ak nieje možné rozumne rozhodnúť či bude daná požiadavka na zmenu schválená alebo nie tak o ďalších krokoch rozhodne vždy vedúci tímu Peter Uherek. On môže definitívne schváliť požiadavku, zrušiť požiadavku alebo kvôli tomuto kontaktovať vedúceho Michala Holuba pre podrobnejšie informácie.

3 Manažment rizík

3.1 Metorika identifikácie a riešenia rizík

Obsahom tejto metodiky je identifikovanie rizík v tíme a poskytnutie ich riešení. Metodika je zameraná na riešenie rizík, ktoré by v značnej miere ohrozili prácu na projekte, a tým aj kvalitu samotného produktu. Metodika poskytuje riešenia na problémy s technikou, časom a členmi tímu.

3.1.1 Dedikácia

Touto metodikou sa riadia členovia tímu DigitalNinjas pri vypracovaní tímového projektu Sofistikované spracovanie dát v akademickom roku 2014/2015. Je vytvorená za účelom odhaliť kritické situácie ohrozujúce kvalitu výsledného produktu a predísť takýmto situáciám.

3.1.2 Zoznam súvisiacich metodík

- Metodika evidencie úloh
- Metodika plánovania pre tím a jednotlivých členov
- Metodika vyhodnocovania plnenia plánu a návrh úprav
- Metodika riadenia požiadaviek na zmenu

3.1.3 Vymedzenie pojmov

Šprint - Doba určená na plnenie zadaných úloh

Task - Úloha v šprinte

Upload - Nahranie dát na server

3.1.4 Vymedzenie rolí a zodpovedností účastníkov (Rola - Zodpovednosť)

Manažér rizík - Snaží sa predvídať kritické situácie pre tím

Manažér plánovania - Vytvára časový plán riešenia úloh

Člen tímu - Zúčastňuje sa riešení kritických situácií

3.1.4.1 Identifikácia rizík

Táto časť popisuje jednotlivé riziká, ktoré môžu vážne ohroziť kvalitu produktu. Riziká sú zoradené podľa dopadu na kvalitu projektu od najzávažnejšieho po menej vážny. Ku každému riziku je vytvorený popis rizika, ktorý sa skladá z častí uvedených nižšie.

3.1.5 Názov – Popis

Pravdepodobnosť výskytu - S akou pravdepodobnosťou riziko nastane (0 - 100%)

Dopad - Aký dopad bude mať riziko na kvalitu produktu (1 - 10b)

Príčina - Za akých okolností môže riziko nastať

Dôsledok - Aký dôsledok riziko so sebou prináša

Opatrenia - Ako sa riziku vyhnúť

Riešenie - Ako vyriešiť riziko, ak už nastalo

Stupeň rizika - Dopad (1 - 10) x Pravdepodobnosť výskytu (0.0 - 1.0)

3.1.6 Identifikované riziká podľa dopadu na projekt:

1. Odchod člena z tímu
2. Neplnenie si povinností člena tímu
3. Zlé odhadnutie náročnosti úlohy
4. Stratenie neuploadnutých dát z úlohy zo šprintu
5. Neúčast člena tímu na tímovom stretnutí

3.1.6.1 Popisy jednotlivých rizík

1. Odchod člena z tímu

Pravdepodobnosť výskytu: 15%

Dopad: 10b

Príčina: Nezvládnutie náročnosti štúdia

Dôsledok: Zníženie počtu členov tímu, časové omeškanie realizácie projektu

Opatrenia: Motivácia člena tímu, pomoc pri vypracovaní úloh, prideľovanie menej náročných úloh

Riešenie:

Prenesenie manažérskej roly odchádzajúceho člena tímu na iného člena. Vylúčenie menej dôležitých požiadavok na projekt.

3.1.6.2 Stupeň rizika: 1.5

2. Neplnenie si povinností člena tímu

Pravdepodobnosť výskytu: 15%

Dopad: 9b

Príčina: Nezáujem o tému projektu, problémy v súkromí, preťaženosť.

Dôsledok: Časové omeškanie v realizácii projektu

Opatrenia:

Keď sa členom tímu zdá nejaký člen tímu podozrivý (nestretáva sa s tímom, nemá záujem o to, čo sa v tíme deje, nevypracováva úlohy na čas alebo ich vypracuje na veľmi nízkej úrovni) je potrebné sa s týmto členom porozprávať, zistiť príčinu neplnenia si povinností a na základe zistených informácií riešiť situáciu.

Riešenie:

-> dávať mu úlohy takého typu, o ktoré má najväčší záujem

-> pokúsiť sa mu pomôcť s vypracovaním úlohy

- poskytnúť materiály, v ktorých sa rieši podobná úloha

- navrhnúť možné riešenie

-> vysvetliť mu časť úlohy, ktorej nerozumie

-> v neriešiteľných situáciách konzultovať tento problém s vedúcim tímu, prípadne s garantom predmetu

3.1.6.3 Stupeň rizika: 1.4

3. Zlé odhadnutie náročnosti úlohy

Pravdepodobnosť výskytu: 40%

Dopad: 3b

Príčina: Neznalosť používanej technológie programovania, nedostatočné skúsenosti v plánovaní

Dôsledok: Nesplnenie zadanej úlohy, splnenie úlohy v nedostatočnej kvalite, splnenie iba časti zadanej úlohy

Opatrenia: Ak je veľká odchýlka medzi jednotlivými bodovaniami členov tímu, je potrebné opakovanne prediskutovať problematiku tejto úlohy

Riešenie: Rozdelenie úlohy na viacero jednoduchších úloh a ich naplánovanie do nasledujúcich šprintov

3.1.6.4 Stupeň rizika: 1.2

4. Stratenie neuploadnutých dát z úlohy zo šprintu

Pravdepodobnosť výskytu: 10%

Dopad: 5b

Príčina: Porucha hardvéru alebo softvéru, nerozvážne konanie pri programovaní alebo zaobchádzaní s počítačom

Dôsledok: Nesplnená úloha zo šprintu a možné časové omeškanie realizácie projektu, ak na túto úlohu nadväzujú úlohy naplánované do nasledujúceho šprintu.

Opatrenia: Dáta po každom skončení práce na projekte zálohovať. Takisto každú zmenu na tímovej stránke najprv vykonať v dropbox foldri "\\Dropbox\DataPoints\Timovy Web - dokumenty\website-actual" až potom zosynchronizovať na server!

Riešenie: Člen tímu, u ktorého toto riziko nastane, si túto úlohu prenesie do ďalšieho šprintu ako úlohu navyše. Splnenie takejto úlohy je nutné do najbližšieho stretnutia.

Stupeň rizika: 0.5

3.1.7 5. Neúčast člena tímu na tímovom stretnutí

Pravdepodobnosť výskytu: 30%

Dopad: 1b

Príčina: Zdravotné problémy, súkromné problémy.

Dôsledok: Nedostatočná informovanosť chýbajúceho člena tímu.

Opatrenia: Zistenie účasti členov tímu v dostatočnom predstihu.

Riešenie: Ak je to možné, presunúť tímové stretnutie na iný termín.

3.1.8 Stupeň rizika: 0.3

4 Manažment rozvrhu a plánovania

4.1 Metodika plánovania pre tím a jednotlivých členov tímu

Cieľom tejto metodiky je definovanie pravidiel pre plánovanie celkového vývoja produktu v rámci predmetu Tímový projekt. Zaoberá sa úlohami jednotlivých členov tímu, plánovaním úloh do šprintu a pomocnými nástrojmi pre zlepšenie plánovania. Táto metodika je záväzná pre všetkých členov tímu.

4.1.1 Vymedzenie pojmov a skratiek

Project Owner – projektový vlastník, pedagogický vedúci (PO)

JIRA – systém na správu projektu (PMS)

Story Points – hodnotenie každej úlohy vyjadrené ako ľubovoľné meradlo (SP)

GitHub – správca verzií softvéru (SVN)

code review – revízia kódu ďalšími členmi tímu

commit – pridanie novej časti svojho kódu do SVN

task – úloha

4.1.2 Zoznam nadväzujúcich metodík

Metodika vyhodnocovania plnenia plánu a návrh úprav

Metodika evidencie úloh

4.1.3 Product Backlog

Všetci členovia tímu pri riešení pridelených úloh takisto vymýšľajú nové vlastnosti systému vo forme používateľských príbehov a zapisujú ich v JIRA do Product Backlogu. Odtiaľ môžu byť na tímovom stretnutí vytiahnuté, skonzultované s Product Ownerom, ktorý určí, že sa budú robiť. Pokiaľ sa jedná o zložitejšiu úlohu, tím môže navrhnúť rozbitie úlohy do menších úloh. Vybrané úlohy sa následne ohodnotia všetkými členmi tímu. Pridelia sa členovi tímu a presunú z Product Backlogu do Sprint Backlogu.

4.1.4 Šprint

Časové obdobie v trvaní dvoch týždňov (okrem prvého a posledného šprintu), v rámci ktorého tím rieši zvolené používateľské príbehy. Členovia tímu na tímovom stretnutí v strede šprintu predstavia prototyp úlohy, ktorú riešia. Ostatní členovia môžu pripomienkovať, či sa úloha zhoduje s víziou celého tímu, prípadne navrhnú zmeny. Dva dni pred odovzdaním úloh je potrebné každým členom commitnuť svoju implementáciu do GitHubu a prebehne Code Review.

4.1.5 Sprint Backlog

Používateľské príbehy, ktoré boli na tímovom stretnutí vybrané na riešenie v danom šprinte, sú presunuté z Product Backlogu do Sprint Backlogu. Každá úloha je pridelená jednému členovi tímu, ktorý zodpovedá splnenie danej úlohy. Má možnosť vytvárať pod-úlohy, ktoré môže pridelať ďalším členom tímu, v prípade, že sa jedná o zložitejšiu úlohu.

4.1.6 Tímové stretnutia

Tím sa stretáva raz do týždňa. Vzhľadom na dvojtyždňové šprinty sa niektoré stretnutia prekrývajú aj s termínmi ukončenia šprintu. Na základe toho identifikujeme dva typy stretnutí:

na rozhraní šprintov

v strede šprintu

Od typu stretnutia sa odvíja aj jeho priebeh.

4.1.7 Stretnutie na rozhraní šprintu

Stretnutie na rozhraní šprintu sa delí na dve časti a to ukončenie aktuálneho šprintu a začatie nového.

Proces ukončenia šprintu:

Vstup: výsledky práce medzi stretnutiami

Výstup: zaznamenaný aktuálny stav úloh na Scrum Board a zapísanie stavu úloh „Hotová“ do zápisnice

Zodpovedný: člen tímu, zapisovateľ

Popis:

1. Zapisovateľ si pripraví zápisnicu obsahujúcu tabuľku s úlohami
2. Product Owner prejde všetky úlohy aktuálneho šprintu a poprosí zodpovedného člena tímu o krátke vyjadrenie k svojej úlohe, či bola splnená. Skontroluje takisto či je v JIRA zapísaný stav úlohy RESOLVED a pridaný čas strávený vypracovaním.
3. Následne sa mu predstaví demo vypracovaných príbehov.
4. Product Owner následne zhodnotí úlohy a vyjadrí sa či boli splnené alebo nie.
 - a. Splnená – ak bola úloha splnená, zodpovedný člen za úlohu priradí sebe resp. ostatným členom Story Points podľa práce na úlohe a zmení stav úlohy v JIRA ako DONE
 - b. Nesplnená – v tomto prípade sa vedie diskusia s Product Ownerom o dôvode nesplnenia. PO môže nariadiť členovi tímu dopracovanie nesplnenej úlohy. V prípade nastania špecifických udalostí týkajúcich sa náročnosti úlohy môže rozhodnúť o rozbití úlohy na menšie úlohy a zaradenie do Product Backlogu. V tomto prípade sa Story Points neudeľujú nikomu.

Po tomto procese by mal mať zapisovateľ naplnenú tabuľku nasledovne:

ID	Popis úlohy	Zodpovednosť	Termín začatia	Termín ukončenia	Stav
301	Stiahnutie datasetu a pridanie do databázy	Martin Lošák	6.11.2014	20.11.2014	Hotová
302	Chcem vidieť základne textové info o datasete (atribúty, dátum, veľkosť)	Peter Uherek	6.11.2014	20.11.2014	Prebieha

Tabuľka 1. Stav úloh v zápisnici stretnutia

Proces zaznamenania novej úlohy:

Vstup: diskusia o príbehu aktuálneho šprintu

Výstup: zaznamenanie úlohy do Sprint Backlogu a do zápisnice

Zodpovedný: člen tímu, zapisovateľ

Popis:

1. *Zapisovateľ si pripraví tabuľku so stĺpcami obsahujúcimi ID, Popis úlohy s Zodpovednosť z úlohu*
2. *Po diskusii tímu s Product Ownerom aj s ohľadom na príbehy v Project Backlogu sa stanovia a napíšu na tabuľu vybrané úlohy do šprintu.*
3. *Následne sa vybrané úlohy ohodnotia pomocou kartičiek s číslami Story Points každým členom tímu. Úloha získa toľko bodov, koľko zahlasuje väčšina. V prípade nezhody rozhoduje priemer prípadne zmena hodnotenia niektorého člena tímu.*
4. *K úlohám sa následne prihlásia členovia, ktorí ich chcú vypracovať. V prípade, že sa súčet Story Points u zadelených úloh približne zhoduje s predchádzajúcim šprintom a každý člen má zadelenú minimálne jednu úlohu a postačuje mu, ostatné sa hodia do Product Backlogu.*
5. *Zapisovateľ vyplní tabuľku, kde ID úlohy určí v tvare napr. 304, kde 3 označuje tretí šprint a 04 označuje číslo úlohy v šprinte. Vyplní takisto popis pre jasnosť úlohy a zodpovedného člena tímu za úlohu.*
6. *Zapisovateľ je následne povinný tieto úlohy pridať do PMS systému JIRA, kde zaznačí ID + názov, popis, zodpovedného člena a Story Points úlohy.*
7. *Zapisovateľ do 24 hodín pridá úlohy do Sprint backlogu a nasledujúci deň po tímovom stretnutí o 10 hodine spustí šprint. Šprint trvá do začiatku ďalšieho stretnutia na rozhraní šprintu.*

Na konci procesu by mal mať zapisovateľ vytvorenú tabuľku podobnú Tabuľke 1 ale bez stĺpca Stav.

4.1.8 Stretnutie v strede šprintu

Stretnutie v strede šprintu slúži na priebežné zhodnotenie úloh. Product Owner sa každého člena spýta ako s úlohou pokročil, či má s niečím problém. Každý člen je povinný na toto stretnutie priniesť základný prototyp úlohy, aby vedel tím zhodnotiť kvalitu a pochopenie úlohy. Zároveň aby bol vyhradený druhý týždeň šprintu na prípadne prerobenie úlohy po kvalitatívnej stránke.

4.1.9 Podporné nástroje

4.1.9.1 Google Kalendár

Manažér plánovania má v rámci svojich povinností za úlohu udržiavať aktuálny kalendár celého tímu. Tímový kalendár máme vytvorený prostredníctvom nástroja Google Kalendár. Prehľadne zobrazuje všetky rozvrhy jednotlivých členov tímu s možnosťou editácie každého člena. Každý člen si teda môže pridávať ďalšie termíny kedy je nedostupný. V rámci týchto okien si vieme napláňovať neoficiálne tímové stretnutia.

4.1.9.2 Wikia - Harmonogram, dôležité termíny (míľniky)

Manažér plánovania má takisto povinnosť udržiavať aktuálny harmonogram dôležitých termínov a známe míľniky v projekte. Tieto informácie sú prehľadne uložené v jednoduchej tabuľke na spoločnej Wiki stránke, kde tím združuje všetky informácie ohľadne dokumentov a plánov.

Plán termínov na zimný semester 2014/2015	
Dátum	Úloha
23.10.2014	Stretnutie - Koniec Sprint-1
28.10.2014	Odovzdanie prihlášky na TP CUP
30.10.2014	Stretnutie - Priebežný stav Sprint-2
6.11.2014	Stretnutie - Koniec Sprint-2

Obrázok 5 - Harmonogram plánu v systéme Wikia

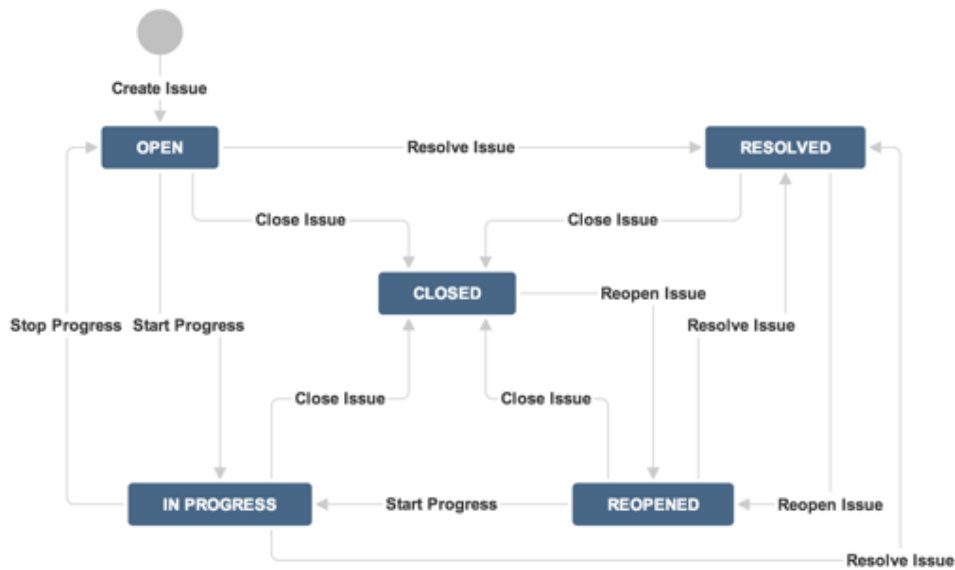
4.1.9.3 JIRA – Systém na správu projektu (PMS)

V rámci plánovania využívame JIRA PMS, kde tím plánuje všetky úlohy v Project Backlogu, prípadne v aktuálne úlohy v Sprint Backlogu. Každý člen tímu je povinný na základe svojej aktuálnej úlohy vytvárať v Product Backlogu nové používateľské príbehy, aby následne na tímovom stretnutí tieto mohli byť skonzultované a zaradené do nasledujúceho šprintu. Každý člen tímu je povinný zaznamenávať čas strávený riešením úlohy.

Evidencia úloh počas šprintu:

- Počas šprintu je, každý člen tímu povinný dbať na aktuálny stav svojej úlohy v JIRA. Pokiaľ sa na úlohe ešte nezačalo pracovať, tak úloha musí byť v stave *OPEN*. V prípade, že sa na úlohe začalo pracovať, tak úloha musí byť v stave *IN PROGRESS*. Po dokončení úlohy musí byť úloha prevedená do stavu *RESOLVED*.
- Za zmenu stavov počas šprintu je zodpovedný ten, kto je zodpovedný za riešenie úlohy. Inak povedané zodpovedný za zmenu stavov v úlohe je ten, kto je zapísaný v *Assignee*.
- Počas šprintu si musí každý člen tímu zaznamenávať prácu na svojej úlohe a to pomocou logovania času v JIRA.
- Pokiaľ sa zistí, že úloha je veľmi rozsiahla, je možné identifikovať viacero podúloh v rámci jednej úlohy. Práva na takúto zmenu prislúchajú zodpovednému za riešenie úlohy. Všetky nové úlohy musia byť typu SUB-TASK a musia obsahovať:
 1. Názov úlohy. Zmysluplný názov, ktorý vystihuje podstatu úlohy.
 2. Konečný termín, do kedy má byť úloha zhotovená. Tento termín môže byť rovnaký ako termín pôvodnej úlohy, ale nikdy nemôže presahovať tento termín.
 3. Zodpovedného člena tímu, ktorý je zodpovedný za riešenie, stav a evidenciu úlohy.

4. Odhad času na riešenie úlohy.



Obrázok 6 - Diagram jednotlivých stavov v systéme JIRA

Na konci každého šprintu sa vyhodnotia všetky úlohy. Pokiaľ riešenie úlohy bude splnené v akceptovateľnej forme zapisovateľ dá úlohu do stavu CLOSED. V inom prípade sa úloha presunie späť do Product Backlogu.

4.2 Metodika vyhodnocovania plnenia plánu a návrh úprav

4.2.1 Vyhodnocovanie plnenia plánu

Vyhodnocovanie plnenia plánu prebieha každý druhý týždeň. Pri ukončení šprintu sa tím stretne a s vedúci vyhodnotí, či bola úloha splnená v dostatočnej kvalite.

4.2.2 Návrh úprav

Prípadne návrhy úprav sú možné dva dni pre ukončením šprintu, kedy musia byť všetky kódy commitnute a prebieha code review. V prípade nedostatočnej kvality, sa nasledujúce stretnutie dohodne, či a kedy sa budú úpravy vykonávať. Task sa pridá do backlogu a vytiahne sa do ďalšieho šprintu

4.3 Metodika udržiavania informácií o stave projektu

V priebehu každého šprintu vzniká povinná a dôležitá množina dokumentov a kódu. V rámci nášho projektu sme identifikovali nasledovné náležitosti:

- Analýza
- Zdrojový kód
- Dokumentácia

Finalizácia, zmena, alebo aktualizácia spomínaných náležitostí musí byť jasne oznámená každému členovi tímu, s tým, že záznam o jej oznámení musí byť archivovaný. Niekoľko nami používaných nástrojov ponúkajú automatickú notifikáciu, pri zmenách v zdrojovom kóde a sledovaní progresu vývoja produktu budeme informovaný nasledovne:

- Zdrojový kód - emailová notifikácia - je potrebné byť prihlásený do tímu v GitHub
- Progres vývoja produktu - detailný prehľad v JIRA - bližšie info v:
 - Metodika plánovania pre tím a jednotlivých členov tímu
 - Metodika vyhodnocovania plnenia plánu a návrh úprav

Akúkoľvek inú zmenu ktorá môže brzdiť iného člena tímu, komplikáciu, zmenu v plánovaní, identifikáciu nedostatku a chýb, bez zbytočného odkladu hlásime na HipChat , tak, aby náplň oznamu súvisel s kategóriou:

- GitHub
- HMTL UI
- Jira
- Ruby on Rails

Ak nie je možné zaradiť náplň oznamu do žiadnej z existujúcich kategórií, oznamovateľ vytvorí novú kategóriu, ktorej názov bude zodpovedať téme problematiky, a to v širšom kontexte. Príklad názvu kategórie pre chybu v štruktúre tabuľky v databáze:

Nesprávna - chýbajúci stĺpec v tabuľke "Datasets"

Správny - Postgre

Nakoľko je každý člen tímu povinný odovzdávať výsledky svojej práce na kontrolu príslušnému manažérovi, stanovili sme nasledovné termíny odovzdávania:

Zdrojový kód - prototyp - najneskôr na prvom tímovom stretnutí v rámci šprintu

Zdrojový kód - code review - najneskôr 2 dni pred ukončením šprintu

Dokumentácia - najneskôr 2 dni pred koncom šprintu

5 Manažment podpory vývoja

5.1 Metodika manažmentu verzií

Účelom metodiky je oboznámenie všetkých prispievateľov so základnými postupmi, ktoré musia dodržiavať pri procesoch spojených so správou zmien v zdrojovom kóde počas vývoja projektu DataPoints. Metodika taktiež definuje štábnu kultúru pre správy sprevádzajúce jednotlivé odovzdania zdrojového kódu a taktiež spôsob značkovania verzií

Komu je metodika určená

Uvedená metodika je záväzná pre všetkých členov tímu pracujúcich na projekte DataPoints v rámci predmetu Tímový projekt, ktorí do uvedeného projektu prispievajú zdrojovými kódmi.

5.1.1 Vymedzenie pojmov a skratiek

Distribuuovaný systém zmien verzií - je nástroj slúžiaci na údržbu zmien v dokumentoch, súboroch so zdrojovými kódmi programov a iných artefaktoch softvérového vývoja

Repozitár - je centrálné úložisko, v ktorom sú uložené všetky vetvy, odovzdania a iné historické zmeny sledovaných súborov

Vetva (angl. branch) - je línia zdrojového kódu, ktorá umožňuje udržiavanie historických zmien v zdrojovom kóde oddelene od zmien publikovaných v iných vetvách

Odovzdanie (angl. commit) - špecifická snímka repozitára zhotovená v konkrétnom čase

Listové odovzdanie (angl. head) - najnovšie odovzdanie vytvorené v konkrétnej vetve

Správa odovzdania (angl. commit message) - správa, ktorá stručne charakterizuje dôvody zmien a samotné zmeny

Odovzdanie produktu, odovzdanie vlastníkovi produktu - odovzdanie vytvorenej aplikácie vlastníkovi produktu

5.1.2 Zoznam nadväzujúcich metodík a dokumentov

Metodika riešenia konfliktov - v metodike je popísaný spôsob riešenia konfliktov, ktoré vznikli počas vývoja a prejavili sa počas spájania verzií zdrojového kódu

5.1.3 Vetvenie zdrojového kódu

5.1.3.1 Zdrojové vetvy

V repozitári sú udržiavané 3 základné vetvy zdrojového kódu:

- origin/master
- origin/dev
- origin/hotfix

origin/master

Master je hlavná vetva odzrkadľujúca stabilný stav, v ktorom je možné webovú aplikáciu nainštalovať na produkčný server. Vlastníkovi produktu sa vždy odovzdáva verzia aplikácie, ktorá vznikla z otestovaného zdrojového kódu, odovzdaného maximálne tri hodiny pred začiatkom tímového stretnutia.

Na konci šprintu sa podľa poradového čísla šprintu posledné odovzdanie označuje. Konvencia pre názvoslovie značiek je popísaná v časti "pomenovávanie odovzdaní". Výslednú aplikáciu z poslednej odovzdanej verzie v rámci vetvy origin/master vytvára integračný server. História tejto vetvy je prísne zakázané retrospektívne modifikovať, toto neplatí pre manažéra verzií zdrojového kódu.

Vytvorená z vetvy	-
Spája sa s vetvou	-
Konvencia tvorby názvu	origin/master

origin/dev

Dev je hlavná integračná vetva v ktorej počas prebiehajúceho šprintu zdrojové kódy reflektujú najaktuálnejší stabilný stav projektu. Z každého odovzdania v rámci tejto vetvy musí byť možné zostrojiť výslednú aplikáciu podľa zostavovacieho skriptu. V tejto vetve je implementovaná funkcionálna, ktorá bude súčasťou ďalšej iterácie odovzdania vlastníkovi produktu po tom, čo sa na konci šprintu spojí do vetvy origin/master. Do tejto vetvy sa spájajú otestované zmeny z funkcionálnych vetiev a zmeny ktoré majú veľmi malý rozsah a predpokladá sa, že pracujú správne aj bez automatizovaného testovania.

Vytvorená z vetvy	origin/master
Spája sa s vetvou	origin/master - priebežne
Konvencia tvorby názvu	origin/dev

origin/hotfix

Nakoľko nie je možné nasadiť do produkcie listové odovzdanie z vetvy origin/dev (vetva origin/dev obsahuje funkcionality odovzdávanú v ďalšom šprinte), je nutná existencia tretieho typu zdrojovej vetvy. Hotfix je vetva používaná pri oprave kritických chýb. Nakoľko bude každé odovzdanie nachádzajúce sa v tejto vetve nasadené na produkčný server, vetva musí obsahovať iba kód pochádzajúci z vetvy master, rozšírený o kritickú opravu a prislúchajúce testy.

Postup pri vývoji kritickej opravy:

1. Zodpovedný vývojár zosynchronizuje vetvu origin/hotfix s vetvou origin/master
2. Oprava kritickej chyby a prislúchajúce testy sa odovzdajú do vetvy origin/hotfix
3. Odovzdanie vo vetve origin/hotfix sa prešíri do vetvy origin/master a origin/dev

Vytvorená z vetvy	origin/master
Spája sa s vetvou	origin/master - priebežne , origin/dev - priebežne
Konvencia tvorby názvu	origin/hotfix

5.1.4 Funkcionálne vetvy

Účelom funkcionálnej vetvy je implementácia atomickej funkcie aplikácie, prislúchajúcich testov a dokumentácie. Funkcionálne vetvy vznikajú vždy v deň začatia nového šprintu a každá funkcionálna vetva reflektuje schválenú atomickú funkcionality dodávanú v nasledujúcom odovzdaní vlastníčkovi produktu. Názov vetiev sa priebežne mení a jeho názvoslovie sa skladá z kľúčového reťazca "Feature#" a identifikátora úlohy zo systému na správu úloh. Implementácia a testy môžu byť rozdelené do viacerých odovzdaní, no v momente keď považuje vývojár implementáciu novej funkcionality za dokončenú, je vývojár povinný:

1. Spojiť prichádzajúce zmeny z vetvy origin/dev a vytvoriť nové listové odovzdanie
2. Overiť, že sú výsledky testov (funkcionálnych, integračných a regresných) v spojenom listovom odovzdaní splnené
3. Overiť, že spojený kód v listovom odovzdaní spĺňa kritériá kvality
4. Vytvoriť žiadosť o revíziu kódu
5. Po schválení revízie kódu spojiť kód v listovom odovzdaní funkcionálnej vetvy do vetvy origin/dev

6. Uzatvoriť funkcionálnu vetvu

Vývojárom sa navyše odporúča vykonávať spájanie z vetvy origin/dev do funkcionálnej priebežne, aby sa vyhlí problémom pri integrácii existujúcich zmien

Vytvorená z	origin/dev
Spája sa do	origin/dev
Konvencia tvorby názvu	Feature#JiraKey (napríklad Feature#25)

5.1.5 Pomenovávanie odovzdaní

5.1.5.1 Formát správy odovzdania

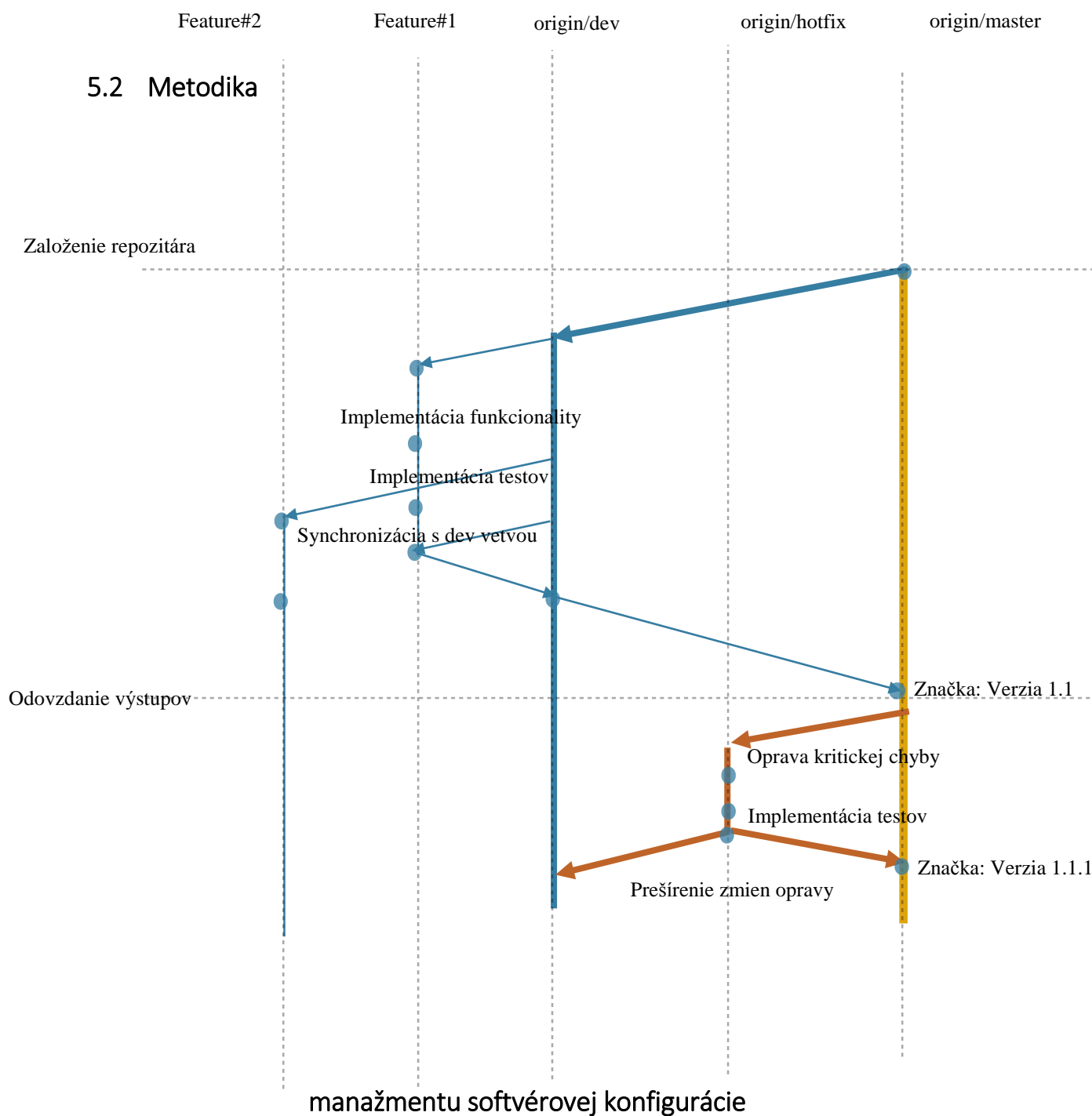
1. Nadpis toho, čo zmena obsahuje a prečo ju bolo nutné vykonať - maximálne 50 znakov
2. Prázdny riadok
3. Ak je potrebné, blok textu s detailnejším popisom vykonávanej zmeny - maximálne 72 znakov
4. Prázdny riadok
5. Identifikačné číslo úlohy zo systému na správu úloh, ku ktorému odovzdanie prináleží a pomlčkou oddelený nový stav úlohy

5.1.5.2 Formát značky odovzdania

Značkujú sa iba odovzdania vo vetve origin/master, z ktorých bola vytvorená aplikácia nasadená na produkčný server. Verzie sa značkujú viacúrovňovým číselníkom, začínajúcim číslom 1. Na konci každého šprintu sa začne číslovanie od novej úrovne číselníka.:

- 1.X - verzia nasadená na konci šprintu, kde X je poradové číslo šprintu
- 1.X.Y - opravovaná verzia pochádzajúca z vetvy origin/hotfix, kde X je poradové číslo šprintu a Y je poradové číslo opravy

5.1.6 Príloha: Vizuálna schéma vetvenia



Hlavným spôsobom ako je manažovaná softvérová konfigurácia je využitie profilov. Profil zahŕňa konfiguráciu kontextu v akom je webová aplikácia spúšťaná. Konfigurácia nastavuje systémové premenné, IP adresy a porty databáz a iných serverov na ktorých webová aplikácia závisí, zoznam používaných knižníc ako aj nastavenia používaných knižníc - napr. či budú dostupné iba počas buildu, testovania alebo až počas behu webovej aplikácie. Webová aplikácia môže byť spúšťaná resp. vyvíjaná v troch rôznych profiloch:

- Test

- Vývoj
- Produkcia

5.2.1 Profil Test

Predstavuje nastavenia kontextu, v ktorom sú spúšťané automatizované testy webovej aplikácie. Do tohto profilu sa pridávajú skripty na zaplnenie testovacej databázy vzorovými dátami, nastavenia testov a iná konfigurácia súvisiaca s automatizovaným testovaním.

5.2.2 Profil Vývoj

Predstavuje nastavenia kontextu, v ktorom je webová aplikácia vyvíjaná. Vývojár bežne testuje novú funkcionálnosť aj manuálne, preto musí tento profil zahŕňať všetku konfiguráciu nevyhnutnú na vývoj.

5.2.3 Profil Produkcia

Predstavuje nastavenia produkčného prostredia, v ktorom aplikácia beží v štandardnej prevádzke keď je dostupná pre koncového používateľa (tzv. v produkcii). Tento profil zahŕňa nastavenia produkčných serverov aby bol ich beh výkonnostne optimalizovaný, zoznam knižníc ktoré sú potrebné počas produkčnej prevádzky ktorý je podmnožinou vývojových knižníc a všetku ostatnú parametrizáciu.

5.2.4 Proces rozšírenia softvérovej konfigurácie projektu

Vývojár si počas návrhu bežne uvedomí, že je nutné nasadiť nový nástroj, resp. iný softvérový produkt spolupracujúci s vyvíjanou aplikáciou. Príkladom môže byť rozšírenie funkcionality o odosielanie emailov na čo je potrebný emailový server, ktorý treba nainštalovať. Pre zamedzenie chaosu je zavedený nasledujúci postup:

1. Vývojár oznámi manažérovi technickej podpory svoje potreby
2. Vývojár a manažér technickej podpory sa dohodnú kto nainštaluje softvérový produkt
3. Vytvorí sa úloha v nástroji na projektové plánovanie (Jira)
4. Prebehne inštalácia nového softvérového produktu
5. Vývojár prípadne oznámi úspešnosť rozšírenia softvérovej konfigurácie manažérovi technickej podpory (komentár v jire)
6. Manažér technickej podpory prípadne oznámi úspešnosť rozšírenia vývojárovi (komentár v jire)
7. Všetka potrebná konfigurácia a prípadný postup konfigurácie sa zaznamená do na to určeného dokumentu
8. Uzatvorí sa úloha v nástroji na projektové plánovanie

5.3 Metodika integrácie softvéru

Účelom vývoju aplikácie je poskytnutie požadovanej funkcionality. Evolučné metódy vývoja však dodávajú túto funkcionálnosť postupne, po častiach. Novú funkcionálnosť je nutné na konci každého obdobia vývoja integrovať k existujúcej v dvoch formách:

5.3.1 Integrácia na úrovni rozšírenia zdrojového kódu

Tento spôsob integrácie je detailne popísaný v metodológii týkajúcej sa spravovania zmien v zdrojovom kóde.

5.3.2 Integrácia na úrovni verzie vytvorenej webovej aplikácie

Pre účely integrácie na úrovni verzií je vytvorený Integračný server, ktorý spustí na konci každého šprintu proces, v ktorom je vytvorená nová koncová verzia webovej aplikácie. V prípade neúspechu je upozornená zodpovedná osoba, v prípade úspechu sa spustí nasadzovanie vytvorenej verzie na produkčný server. Nasadzovanie prebieha v nasledujúcich krokoch:

1. Na produkčnom serveri sa vo vopred určenom adresári vytvorí nový adresár, do ktorého sa nahrá nová verzia webovej aplikácie
2. Ak je nahratie úspešné, upraví sa symbolický odkaz na najnovšiu verziu webovej aplikácie tak, aby odkaz ukazoval na novú verziu
3. Stará verzia je naďalej archivovaná, ak by sa zistili kritické chyby v novej verzii.
4. Do systému je zaznamenaný výsledok procesu nasadenia

5.4 Metodika vyhodnocovania plnenia plánu a návrh úprav

5.4.1 Vyhodnocovanie plnenia plánu

Vyhodnocovanie plnenia plánu prebieha každý druhý týždeň. Pri ukončení šprintu sa tím stretne a s vedúci vyhodnotí, či bola úloha splnená v dostatočnej kvalite.

5.4.2 Návrh úprav

Prípadne návrhy úprav sú možné dva dni pre ukončením šprintu, kedy musia byť všetky kódy commitnute a prebieha code review. V prípade nedostatočnej kvality, sa nasledujúce stretnutie dohodne, či a kedy sa budú úpravy vykonávať. Task sa pridá do backlogu a vytiahne sa do ďalšieho šprintu.

6 Manažment komunikácie a ľudských zdrojov

6.1 Metodika organizácie komunikácie v tíme

Metodika upresňuje komunikáciu v rámci tímu DigitalNinjas. Komunikáciu v tíme môžeme rozdeliť na dve skupiny a to:

- Formálna komunikácia prebieha v rámci tímových stretnutí.
- Neformálna komunikácia prebieha mimo tímových stretnutí.

Metodika opisuje tieto dve skupiny a taktiež opisuje komunikáciu s vedúcim tímového projektu. Metodika sa využíva pri tímových stretnutiach, ale i pri komunikácií s ostatnými členmi tímu. Jej použitie je záväzné pre všetkých členov tímu.

6.1.1 Zoznam nadväzujúcich metodík

Metodika plánovania

Metodika evidencie úloh

Metodika vyhodnocovania plnenia plánu a návrh úprav

Metodika monitorovania prehliadky vytváraného výsledku

6.1.2 Vymedzenie pojmov

User Story – používateľský príbeh. Je to akcia, ktorú by chcel používateľ vedieť vykonať v systéme.

User Points – body vyjadrujúce obtiažnosť implementácie používateľského príbehu.

Jira – webová aplikácia na zaznamenávanie úloh v rámci tímového projektu.

GitHub – systém na správu a manažment verzí softvéru.

HipChat – webová aplikácia na instantné posielanie správ

Wiki – externá stránka na zdieľanie opisov riešenia, metodík a návodov v rámci tímového projektu.

6.1.3 Vymedzenie roly

Vedúci tímového projektu – učiteľ, v niektorých prípadoch taktiež product owner.

Manažér komunikácie – člen tímu, ktorý je zodpovedný za zabezpečenie a dohliadanie komunikácie v rámci tímu.

Vedúci stretnutia – člen tímu, ktorý vedie tímové stretnutie.

Zapisovateľ – člen tímu, ktorý ma na starosti zapisovanie zápisnice.

6.1.4 Formálna komunikácia

Formálna komunikácia tímu prebieha v rámci tímových stretnutí. Tímové stretnutia sa konajú raz za týždeň podľa harmonogramu semestra. Mimoriadne sa môže konať inokedy než ustanovuje harmonogram semestra, avšak musia byť vopred oboznámení a musia súhlasiť všetci členovia tímu a aj mentor tímu.

Zápisnica

Výstupom formálneho tímového stretnutia je zápisnica. Na každom stretnutí je určený jeden člen tímu, ktorý zapisuje zápisnicu, takzvaný zapisovateľ. Rola zapisovateľa sa strieda abecedne podľa priezviska člena tímu. Zapisovateľ je povinný odovzdať zápisnicu v deň, kedy sa stretnutie konalo a to tak, že ju nahrá na stránku tímu vo formáte pdf a docx.

Zápisnica musí byť vyplnená podľa šablony, ktorá je uvedená na stránke nášho tímového projektu. V zápisnici sa musí uviesť dátum a čas, miesto konania, prítomní a neprítomní členovia, zapisovateľ stretnutia, vedúci stretnutia a téma stretnutia. Ďalej je nutné zaznamenať všetky rozhodnutia, alternatívne riešenia a preberané témy. Zápisnica taktiež musí obsahovať všetky nové, prebiehajúce alebo práve ukončené úlohy v tabuľke.

Tabuľka obsahuje ID číslo úlohy, názov úlohy, zodpovedného člena za úlohu, termín začatia úlohy, termín ukončenia úlohy a stav úlohy. Úlohy môžu byť v stave:

- Nová - úloha bola zadaná zadefinovaná a zadaná na príslušnom tímovom stretnutí.
- Nezačatá - na úlohe sa nezačalo ešte reálne pracovať.
- Prebiehajúca - na úlohe sa začalo pracovať.
- Hotová - úloha bola ukončená do času tímového stretnutia.

V zápisnici býva jedna alebo dve tabuľky. V prípade, že sa koná stretnutie počas behu šprintu, tak zápisnica obsahuje jednu tabuľku s názvom: Plnenie úloh z predchádzajúcich stretnutí, ktorá obsahuje len prebiehajúce, nezačaté a práve hotové úlohy z predchádzajúcich stretnutí. V prípade, že sa stretnutie koná na začiatku alebo konci šprintu, tak zápisnica obsahuje aj tabuľku Plnenie úloh z predchádzajúcich stretnutí, ale i tabuľku Nové úlohy, kde sú zadefinované len nové úlohy z daného tímového stretnutia.

Vedenie stretnutia

Na každom stretnutí je určený vedúci stretnutia, ktorý moderuje stretnutie. Vedúci stretnutia býva ten, kto predchádzajúce stretnutie bol určený ako zapisovateľ. V prípade, že sa koná stretnutie na konci a na začiatku stretnutia, tak forma stretnutia je nasledovná:

- Vyhodnotenie úloh a ich splnenie - každý člen tímu povie či je splnená úloha, za ktorú je zodpovedný. V prípade, že úloha je splnená, tak člen tímu popíše svoje riešenie a zhodnotí ho. V prípade, keď úloha nie je splnená, tak uvedie dôvody prečo sa úlohu nepodarilo splniť.
- Predstavenie produktu zákazníkovi - jeden člen tímu predstaví výsledný produkt posledného šprintu zákazníkovi. Nepredstavuje sa vnútorná architektúra alebo technické riešenie, ale iba prípady použitia, ktoré boli zadefinované v danom šprinte.
- Retrospektíva za minulým šprintom (Lessons Learned) - každý člen tímu má právo povedať svoj názor na riadenie tímu v poslednom šprinte a môže navrhnúť zmenu v tejto oblasti, ktorá sa musí odhlasovať a väčšinový počet členov musí s ňou súhlasiť.
- Definovanie úloh do nového šprintu - členovia tímu spolu s vedúcim tímu najskôr zadefinujú viaceré úlohy, z ktorých sa následne vyberú tie najzaujímavejšie do budúceho šprintu. Ich presný počet nie je definovaný. Úlohy sú definované, ako používateľský príbeh (User Story).
- Ohodnotenie úloh - tím ohodnotí vybrané úlohy pomocou bodov story points), ktoré odhadujú úsilie vynaložené na implementovanie úlohy. Hlasovanie úloh prebieha pomocou kartičiek, ktoré obsahujú body. Škála bodov je od 1 po 80, kde menšia hodnota bodov predstavuje ľahšie implementovateľnú úlohu. Väčší počet bodov predstavuje veľmi ťažko implementovateľnú úlohu. Kartičky taktiež

obsahujú otáznik. Pokiaľ člen tímu takúto kartičku použije, naznačuje, že nemá dostatok informácií a nevie priradiť počet bodov k danej úlohe, v takomto prípade sa mu úloha musí znova vysvetliť.

Hodnotenie prebieha ku každej úlohe zvlášť. Vedúci stretnutia povie o akej úlohe sa ide hlasovať. Každý člen tímu si bezovplyvnenia ostatných členov tímu vyberie kartičku s bodmi. Vedúci stretnutia dohliada na to aby každý člen tímu ukázal svoje zvolené body naraz. V prípade, že prevažuje rovnaká hodnota, tak body sa pripíšu k volenej úlohe. V prípade, keď úloha je na rozhraní dvoch hodnôt, tak môžu jednotlivci nakloniť k jednej alebo druhej hodnote. V prípade, že hodnoty bodov jednotlivých členov sú príliš rozmanité, napr. jeden člen tímu dá 8 bodov druhý člen tímu dá 80 bodov, tak jednotliví členovia musia vysvetliť prečo dali takéto hodnotenie. Následne sa hodnotenie úlohy musí zopakovať.

- Pridelenie úloh – jednotlivé úlohy si následne vyberú členovia tímu. Každý člen tímu musí byť zodpovedný aspoň za jednu úlohu.

Stretnutie počas behu šprintu má nasledovnú formu:

- Zhodnotenie prebiehajúcich úloh – vedúci tímového stretnutia sa opýta, každého zodpovedného za riešenie úlohy tri otázky:
 - Čo si robil?
 - Čo plánuješ robiť?
 - Máš nejaký problém?
- Predstavenie riešení – každý člen tímu predstaví riešenie svojej úlohy a povie jeho výhody a nevýhody.
- Voľná diskusia – každý člen tímu môže otvoriť rôzne témy ohľadom tímového projektu.

6.1.5 Neformálna komunikácia

Neformálna komunikácia tímu prebieha viacerými kanálmi. Rozdeľujeme ich na hlavné a vedľajšie.

6.1.6 Hlavné komunikačné kanály

Hlavné komunikačné kanály predstavujú najviac používané spôsoby komunikácie v tíme. Poznáme tri najdôležitejšie kanály:

- Chat
- Tímový email
- Osobné stretnutie

Chat

Chat sa využíva na rýchle informovanie všetkých členov tímu. Na chat sa využíva aplikácia HipChat, v ktorej sa dajú založiť viaceré miestnosti. Pre naše potreby sú zatiaľ zadané tieto miestnosti:

- DigitalNinjas – všeobecná miestnosť na rýchle informovanie ostatných členov tímu.
- Help – miestnosť na riešenie problémov spojených s vývojom produktu.

- UI – miestnosť učená na predstavovanie a spoločné hodnotenie užívateľského rozhrania.
- Jira – miestnosť na informovanie o zmenách v systéme na manažment vývoja softvéru.
- GitHub – miestnosť, kde sa automaticky pridávajú informácie o pridaniach nových verzií produktu do systému na správu softvéru.
- Error – miestnosť na oznamovanie chýb v produkte.

Pridanie novej miestnosti do HipChatu je dovolené každému členovi tímu. Pri vytváraní takejto miestnosti je povinnosť vytvárajúceho pridať prvú správu o zámere tejto miestnosti do vytváranej miestnosti a oboznámiť všetkých členov tímu v miestnosti DigitalNinjas o vytvorení novej miestnosti a jej zámere.

Likvidácia miestností je povolená iba manažérovi komunikácie a to v takom prípade, že po dlhšom čase sa ukáže miestnosť ako nevyužitá. Nevyužitie miestnosti nastáva v tom prípade, ak do miestnosti nebola pridaná viac ako jeden mesiac nová správa.

Tímový email

Spoločný email tímu je založený na webovej službe Gmail. Emailová adresa je tp3fiit@gmail.com.

Osobné stretnutie

Nemá predpísanú žiadnu formu a je len na účastníkoch stretnutia ako si ju zdefinujú.

6.1.7 Vedľajšie komunikačné kanály

Vedľajšie komunikačné kanály predstavujú spôsoby komunikácie, ktoré nie sú majoritne využívané v rámci nášho tímového projektu. Ich zoznam je nasledovný:

- FB stránka – využíva sa Facebook skupina.
<https://www.facebook.com/groups/534482923320884/>
- Hlasový hovor – využíva sa na hovor či už cez telefón alebo Skype.
- Jira – využíva sa pri komunikácii a zaznamenávaní stavu úloh.
<http://jira.fiit.stuba.sk>
- Osobný email – email, každého člena tímu.
- Wiki – využíva sa externá stránka
http://sk.datapoints.wikia.com/wiki/Datapoints_Wiki

V nasledujúcej tabuľke sú uvedené prípady použitia jednotlivých spôsobov komunikácie. Červenou farbou sú označené prípady, o ktorých je potrebné informovať ihneď.

Informovanie o pridání dokumentu na webovú stránku tímového projektu	HipChat miestnosť DigitalNinjas
Predstavenie riešenia problému alebo úlohy ostatným členom tímu	HipChat miestnosť DigitalNinjas alebo osobné stretnutie
Hlásenie chyby	Jira – zaevidovanie chyby a oznámenie o chybe na HipChat miestnosť Error
Informovanie o stave úlohy	Jira – pridanie komentáru ku konkrétnej úlohe, zmena stavu úlohy
Informovanie o úlohe, ktorá nie je definovaná v Jire	HipChat miestnosť DigitalNinjas

Pripomenutie dôležitého termínu	HipChat miestnosť DigitalNinjas
Komunikácia s tretími stranami	Tímový email
Komunikácia s vedúcim tímového projektu	Tímový email a v osobitých prípadoch osobný email
Upozornenie na zaujímavý článok alebo tému	FB stránka
Pýtanie sa pomoci pri implementovaní alebo riešení úloh	HipChat miestnosť Help alebo osobné stretnutie
Pridanie alebo zmenenie opisu riešenia alebo návodu	Wiki a stručné informovanie v HipChat miestnosti DigitalNinjas
Nahlásenie choroby alebo inej situácie ohrozujúcej plnohodnotné nasadenie člena v tíme	V takomto prípade je potrebné kontaktovať manažéra komunikácie
Informovanie o heslách a prístupoch	FB stránka
Informovanie o nevyriešení pridenej úlohy na čas	HipChat miestnosť DigitalNinjas

Tabuľka 2 – Tabuľka prípadov použitia komunikačných kanálov

6.1.8 Kontaktovanie člena tímu

Postup kontaktovania konkrétneho člena tímu je rozdelený do 3 krokov:

1. Najskôr sa použije HipChat na kontaktovanie člena tímu. Pokiaľ neodpovedá môže sa použiť Facebook chat.
2. Pokiaľ nie je člena tímu možné zastihnúť ani na Facebook chate je možné mu zavolať na osobné telefónne číslo.
3. V prípade neodpovedania na telefonát je možné mu poslať osobný email.

Jednotlivé kroky je možné preskakovať a upravovať, ale iba vo výnimočných situáciách, ktoré každý člen tímu musí vedieť zhodnotiť a vyhodnotiť podľa seba.

6.1.9 Informovanie vedúceho tímového projektu

Informovanie vedúceho o stave projektu prebieha každý týždeň na tímovom stretnutí.

- Mimo stretnutí je možné kontaktovať vedúceho o stave projektu pomocou emailu. Email by mal byť poslaný z tímového emailu a odosielateľ by mal preposlať email všetkým členom tímu. Členovia tímu sú zodpovední za informovanie a odosielanie správ v rámci ich kompetencií v tíme alebo ich úloh v danom šprinte.
- Každý člen tímu môže informovať vedúceho o stave projektu z osobného emailu a nie z tímového emailu a to v takom prípade ak by obsah odosielanej správy ohrozil morálku tímu alebo by vážne zasiahol do súkromia člena tímu.

Mimo stretnutí tímového projektu je takisto možné informovať učiteľa o stave projektu osobne a to tak, že si člen tímu alebo členovia tímu predom dohodnú stretnutie s vedúcim pomocou emailu.

6.2 Metodika informovania učiteľa o stave projektu

Informovanie učiteľa o stave projektu prebieha každý týždeň na tímovom stretnutí.

Mimo stretnutí je možné kontaktovať učiteľa o stave projektu pomocou emailu. Email by mal byť poslaný z tímoveho emailu a odosielateľ by mal preposlať email všetkým členom tímu. Členovia tímu sú zodpovední za informovanie a odosielanie správ v rámci ich kompetencií v tíme alebo ich úloh v danom šprinte.

Každý člen tímu môže informovať učiteľa o stave projektu z osobného emailu a nie z tímoveho emailu a to v takom prípade ak by obsah odosielanej správy ohrozil morálku tímu alebo by vážne zasiahol do súkromia člena tímu.

Mimo stretnutí tímového projektu je taktisto možné informovať učiteľa o stave projektu osobne a to tak, že si člen tímu alebo členovia tímu predom dohodnú stretnutie s učiteľom pomocou emailu.

6.3 Metodika evidencie úloh

6.3.1 Zadávanie úloh v rámci šprintu

Pre každú novú úlohu je potrebné vytvoriť novú Issue v Jire.

V prípade zadávania nových úloh v rámci nového šprintu je potrebné vytvoriť nové Issue. Za zadávanie týchto Issue do Jiry sú zodpovední vždy zapisovatelia z tímového stretnutia, kde sa ustanovili dané úlohy v rámci nového šprintu. Issue musia byť zadané do Jiry najneskôr do 24 hodín od začiatku tímového stretnutia.

Pri zadávaní Issue do Jiry je potrebné vyplniť tieto polia:

V Issue je potrebné vybrať Issue Type: STORY. Následne je potrebné zadať názov úlohy v tvare: Číslo NÁZOV.

Číslo: predstavuje číslo šprintu a číslo úlohy v šprinte. Napr. číslo 205 predstavuje piatu úlohu z druhého šprintu.

Názov: názov úlohy musí byť výstižný a nie metúci. Pokiaľ vytvárame nejakú user story názov musíme písať: Ako používateľ, chcem _____. Napr. Ako používateľ, chcem vidieť všetky svoje nahrané datasety.

V nasledujúcich krokoch je potrebné vyplniť opis úlohy. Opis úlohy by mal byť čo najvýstižnejší, bez zbytočných možností zavádzania riešiteľa úlohy. Pokiaľ je možné opis úlohy by mal opisovať stručný popis ako úlohu vyriešiť a taktiež by mal obsahovať odkaz na existujúce riešenia.

Predposledným krokom, ktorý musí zadávateľ úlohy spraviť je vyplniť pole Assignee a pripísať úlohu, tomu kto sa za ňu stal zodpovedný po dohode na tímovom stretnutí.

Posledným krokom je vyplnenie hodnoty Story Points. Sem sa zadávateľ úlohy zadá body, na ktorých sa tím spoločne zhodol pre danú úlohu na tímovom stretnutí.

Týmto práca zadávateľovi nekončí. Je povinný napísať na HipChat do skupiny DigitalNinjas aby všetci zadali svoje časové odhady na splnenie úlohy.

Ostatný členovia tímu sú povinní vyplniť túto informáciu. V Jire sa tento časový odhad volá Original Estimate a je možné ho upraviť po kliknutí na tlačidlo Edit. Každý člen tímu musí reálne odhadnúť časovú dobu, koľko mu bude vyriešenie úlohy trvať.

Po zadaní všetkých časových odhadov je zadávateľ úloh povinný oboznámiť administrátora Jiry aby spustil nový šprint.

6.3.2 Evidencia úloh počas šprintu

- Počas šprintu je, každý člen tímu povinný dbať na aktuálny stav svojej Issue v Jire. Pokiaľ sa na úlohe ešte nezačalo pracovať, tak úloha musí byť v stave OPEN. V prípade, že sa na úlohe začalo pracovať, tak úloha musí byť v stave IN PROGRESS. Po dokončení úlohy musí byť úloha prevedená do stavu RESOLVED.
- Za zmenu stavov počas šprintu je zodpovedný ten, kto je zodpovedný za riešenie úlohy. Inak povedané zodpovedný za zmenu stavov v Issue je ten, kto je zapísaný v Assignee.
- Počas šprintu si musí každý člen tímu zaznamenávať prácu na svojej úlohe a to pomocou logovania času v Jire.
- Pokiaľ sa zistí, že úloha je veľmi rozsiahla, je možné identifikovať viacero podúloh v rámci jednej úlohy. Práva na takúto zmenu prislúchajú zodpovednému za riešenie úlohy. Všetky nové úlohy musia byť typu SUB-TASK a musia obsahovať:
 - 1. Názov úlohy. Zmysluplný názov, ktorý vystihuje podstatu úlohy.
 2. Konečný termín, do kedy má byť úloha zhotovená. Tento termín môže byť rovnaký ako termín pôvodnej úlohy, ale nikdy nemôže presahovať tento termín.
 3. Zodpovedného člena tímu, ktorý je zodpovedný za riešenie, stav a evidenciu úlohy.
 4. Odhad času na riešenie úlohy.

6.3.3 Evidencia úloh na konci šprintu

Na konci každého šprintu sa vyhodnotia všetky úlohy. Pokiaľ riešenie úlohy bude splnené v akceptovateľnej forme supervisor dá úlohu do stavu CLOSED. V inom prípade sa úloha presunie späť do Backlogu.

6.4 Metodika organizácie zdrojov

Pojem "zdroje" rozumej ako "personálne zdroje v projekte", prípadne "personálne zdroje v tíme" - nakoľko charakter projektu viac ako jeden tím nedovoľuje.

V rámci organizácie zdrojov definujeme následovné kategórie:

1. plánovanie organizácie:
 - projektové rozhrania

- požiadavky na členov tímu
 - organizačná štruktúra tímu
2. plánovanie zdrojov:
- identifikovanie a pridelenie projektových rolí
 - identifikovanie vzťahov

6.4.1 Plánovanie organizácie

6.4.1.1 Požiadavky na členov tímu

V rámci projektu sú jasne definované záväzky a požiadavky ktoré je nutné splniť. Z tohto zadania vyplývajú tieto

6.4.1.2 Projektové rozhrania

6.4.1.3 Požiadavky na členov tímu

- spoznanie a zdokonalovanie sa v jazyku Ruby
- spoznanie a zdokonalovanie sa vo frameworku Rails
- znalosti z databázových systémov

6.4.1.4 Organizačná štruktúra tímu:

V rámci 1. semestra sú identifikované a pridelené nasledovné role:

Projektová rola	Zodpovedná osoba	Zodpovednosť
Tím líder (zástupca vedúceho)	Peter Uhrek	vie, čo sa deje, kto čo robí, čo sa nestíha
Manažér rizík	Stanislav Paľove	proaktívne vysporiadanie s problémami, odhady šance na neúspech. Predvídanie krízových situácií.
Manažér rozvrhu a plánovania	Martin Lošák	– sledovanie dôležitých dátumov - skúšky, zápočtové testy, sviatky. Naplánovanie nasadzovania produktu, jednotlivých procesov v rámci tímu atď.
Manažér kvality	Alex ostrovský	sledovanie kvality výsledných produktov. Nasadzovanie a následné hodnotenie akceptačných testov alebo regresných testov. Sledovanie jednotlivých procesov v tíme, ich hodnotenie a prípadne navrhovanie zmien.
Manažér	Martin Košut	monitorovanie procesných vecí či

monitorovania		bežia, tak ako majú podľa stanovených kritérií.
Manažér podpory vývoja	Igor Daniš	nasadzovanie, obstarávanie a zabezpečovanie prístupu vývojového prostredia pre každého člena tímu. Staranie sa o chod serverov.
Manažér dokumentovania	Jakub kmeťko	sledovanie kvality všetkých dokumentov. Integrácia jednotlivých dokumentov. Nahrávanie dokumentov na stránku.

Tabuľka 2 – Organizačná štruktúra tímu

Toto prerozdelenie môže byť priebežne aktualizované, s tým, že máme stanovený minimálny jeden fixný termín zmeny projektových rolí, a to Január 2015.

6.4.1.5 Plánovanie Zdrojov:

Zahrňa identifikáciu zdrojov, pridelenie úloh, a vzťahov medzi členmi tímu. Tieto náležitosti môžu byť rozdelené tak ako medzi jednotlivcov, aj do skupín.

Úlohy identifikujeme buď:

- v analytickom šprinte, počas ktorého na základe preskúmania konkrétnej oblasti vytvárame do backlogu user stories
- jasným vyplnutím zo zadania
- postupným pridávaním v rámci nových znalostí o problematike
- refaktoringom uzavretých úloh

7 Manažment dokumentovania

7.1 Metodika riadenia procesu dokumentovania (verzia 1)

Metodika definuje pracovné postupy, ktorými sa členovia tímu budú riadiť pri písaní dokumentácie projektu pomocou Microsoft Word a Google Documents. Okrem pokynov pre vytváranie a písanie konkrétnych dokumentov, informáciách o ich štruktúre a formátovaní, obsahuje aj model vydávacieho/schvaľovacieho procesu.

7.1.1 Skratky a pojmy

MS Word – Microsoft Word

gDocs – Google Documents

gDrive – [Google Drive](https://drive.google.com/)

7.1.2 Vysvetlenie pojmov

MS Word – textový procesor, ktorý používame na vytváranie finálnych verzií dokumentov, a ich spájanie do oficiálnej verzie. Dbá sa na formátovanie podľa bodu x.y.

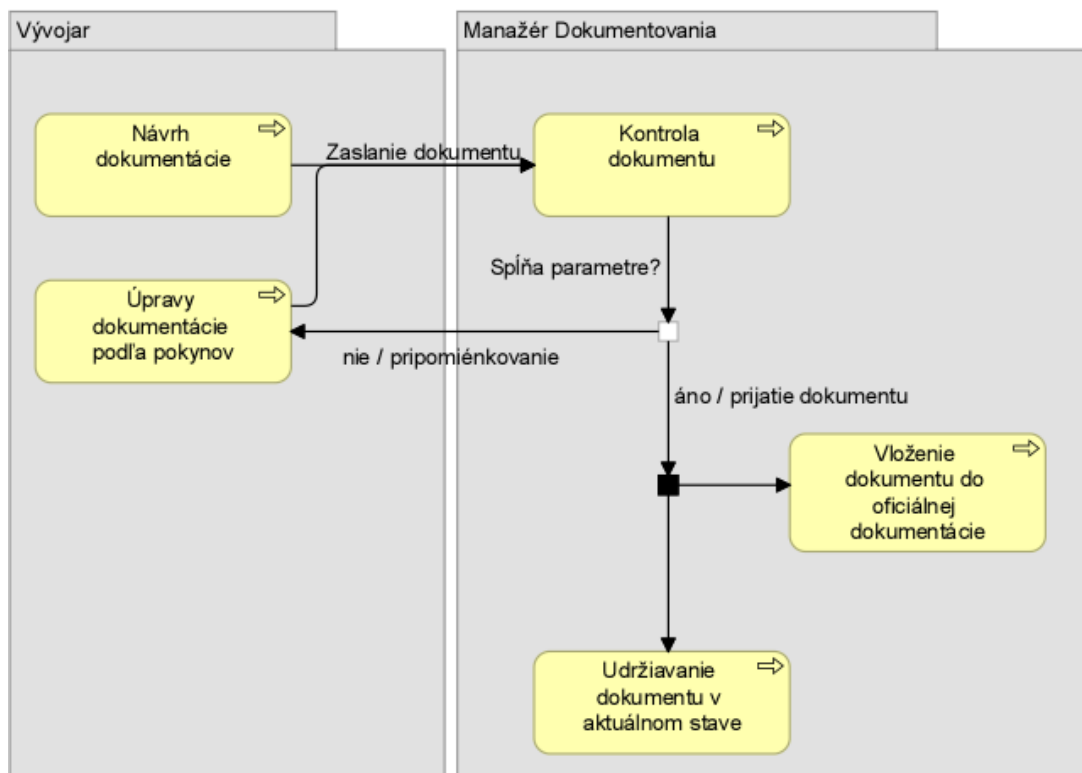
gDocs – nástroj na online editáciu v textovom, tabuľkovom, a prezentačnom procesore. Vytvorené dokumenty sú zdieľané v rámci celého tímu, s tým, že každý člen môže do nich pridávať komentáre, a podľa stanovených pravidiel (viž bod. x.y) aj upravovať.

gDrive – online zdieľaný priečinok, v ktorom sú dostupné všetky gDocs.

7.1.3 Postup dokumentovania

Postup:

1. Vývojár vytvorí návrh systémovej dokumentácie vyplnením relevantných častí šablóny dokumentácie, a zašle ho manažérovi dokumentácie.
2. Manažér dokumentovania skontroluje prijatý dokument a zhodnotí ho. Ak dokument nespĺňa požiadavky, manažér dokumentácie ho vráti vývojárovi, spolu s pripomienkami.
3. Vývojár úpravy/doplní segmenty dohodnuté v bode 3., a výsledok zašle Manažérovi dokumentácie
4. Manažér dokumentácie reviduje zmeny, výsledný dokument schváli a vloží do oficiálnej dokumentácie
5. Manažér dokumentácie vloží dokument do oficiálnej dokumentácie, a od tohto bodu preberá zodpovednosť za udržiavanie dokumentu v aktuálnom stave



Obr. 1.1 – work flow diagram k postupu vytváranie dokumentácie

7.1.4 Spáva verzií dokumentov

Vývojár zaznamená každú zmenu vykonanú na svojich dokumentoch, s tým, že túto informáciu zaznačí do tabuľky „Verzie dokumentu“ (viz. Tabuľka 1.3.1).

Manažér dokumentácie zaznamená každú zmenu vykonanú na oficiálnom dokumente, s tým, že túto informáciu zaznačí do tabuľky „Verzie dokumentácie“, ktorá ma rovnakú štruktúru ako Tabuľka 1.3.1.

Dátum	Verzia	Autor	Sekcia	Stručný popis úpravy

Tabuľka 3 – Verzie dokumentu

7.1.5 Správa a formátovanie dokumentov

7.1.5.1 Vytváranie dokumentov na GoogleDrive

Dokumenty vytvárame pod názvom #ID_ulohy-Nazov_ulohy, kde:

- Id_ulohy je identifikátor konkrétnej úlohy, pridelený na tímovom stretnutí
- Názov úlohy stručná konkretizácia problematiky, pridelená na tímovom stretnutí

Pri nahrávaní lokálneho dokumentu, ktorý nevznikol za použitia služby gDocs, je potrebné dokument vytvoriť a jednoduchým copy-paste jeho obsah prekopírovať z lokálneho dokumentu. V opačnom prípade, a to keď dokument nahráme na server, nie je možná jeho nasledovná editácia, a musí sa pred úpravou sťahovať do stroja, čo je nežiadúce.

7.1.6 Formátovanie dokumentov

7.1.6.1 Štandardný text:

- Font – Calibri light
- Veľkosť písma – 11
- Riadkovanie – 1.1

7.1.6.2 Nadpisy:

Pre lepšiu orientáciu, kaskádujeme na základe číslicového označenia, bez ohľadu na hĺbku pod-levelu.

Nadpis – Heading 1:

Označujú najvyššiu váhu kategorizácie, ktorá vyjadruje vysokú dôležitosť témy, alebo pod ňu spadá väčší logický celok

- Veľkosť písma - 11
- Pozadie - modré
- Farba - biela
- Šírka - 100%

Nadpis – Heading 2:

Označuje druhú najvyššiu váhu kategorizácie, pod ktorú spadá významný celok, patriaci pod kategóriu s nadpisom Heading 1

- Veľkosť písma: 11
- Pozadie: svetlo-modrá
- Farba: čierna
- Šírka – 100%

Nadpis – Heading 3:

Označuje druhú najvyššiu váhu kategorizácie, pod ktorú spadá významný celok, patriaci pod kategóriu s nadpisom Heading 2

- Veľkosť písma: 11
- Pozadie: transparentné
- Farba: modrá

- Šírka – 100%
- Vrchné orámovanie - modré

7.1.6.3 Obrázky:

- Zarovnať na stred
- Pridať popis v tvare „Obrázok # - názov“
- # označuje poradové číslo obrázka v dokumente
- názov označuje názov obrázku

7.1.6.4 Tabuľky:

- Zarovnať na stred
- Pridať popis v tvare „Tabuľka # - názov“
- # označuje poradové číslo tabuľky v dokumente
- názov označuje názov tabuľky

7.1.7 Tvorba dokumentov

7.1.7.1 Názov

Názov témy sa zadáva vo formáte Heading 1. Zadáva sa názov user story / úlohy, ktorý je pridelený na tímovom stretnutí.

7.1.7.2 Opis

High-level opis user story / úlohy z biznis pohľadu.

Zahŕňa biznis procesy a komponenty vstupujúce do biznis procesu.

7.1.7.3 Analýza

Analýza problematiky.

7.1.7.4 Model

Ak je možné, respektíve vhodné, vložiť model z UML. Pokiaľ sa jedná o vizuálne rozhranie, priložiť obrázok.

7.1.7.5 Implementácia

Popis postupu implementácie, a detaily implementácie.

7.1.7.6 Technológia

Stručný opis použitých technológií a ich verzie.

7.1.7.7 Detaily podpory

Ak bola použitá knižnica / API tretej strany uviesť informácie zahrňujúce:

1. Názov spoločnosti
2. Odkaz na stiahnutie podkladov, ktoré sú implementované
3. Odkaz na dokumentáciu

Identifikácia časti knižnice / API ktorá bola použitá (napr. verifikácia platby, zobrazenie markerov na mape..) vo forme: „Názov časti z dokumentácie – strany v dokumentácii“

7.1.8 Šablóna pre tvorbu zápisníc z tímového stretnutia

Zápisnica č: # zo stretnutia tímu 03

Miesto: FIIT STU, miestnosť

Dátum: DD.MM.YYYY

Čas:

Vedúci stretnutia: Ing. Michal Holub

Zapisovateľ:

Prítomní: Ing. Michal Holub

Bc. Igor Daniš

Bc. Martin Košut

Bc. Martin Lošák

Bc. Alex Ostrovský

Bc. Stanislav Paľove

Bc. Peter Uherek

Bc. Jakub Kmeťko

Neprítomní: Chýbajúci člen (chorý)

Téma stretnutia: Doplniť preberané témy

Priebeh stretnutia

- *Návrhy, prednesy, dohody – v tvare:*
 - On navrhol toto, a takto argumentoval
 - On predložil protinávrh, a takto argumentoval
 - Tím sa nakoniec dohodol na
- Zosumarizovanie výsledkov úloh z predchádzajúceho stretnutia:
 - Igor Daniš:
 - Peter Uherek:
 - Martin Lošák:
 - Stano Paľove:
 - Alex Ostrovský:
 - Jakub Kmeťko:
 - Martin Košut:

Bodovanie aktuálneho šprintu

ID	Popis úlohy	Zodpovednosť	Body

Tabuľka 4 – Bodovanie aktuálneho šprintu

Plnenie úloh z predchádzajúcich stretnutí

ID	Popis úlohy	Zodpovednosť	Termín zač.	Termín uk.	Stav

Tabuľka 5 – Plnenie úloh z predchádzajúcich stretnutí

7.2 Metodika riadenia procesu dokumentovania (verzia 2)

Dátum	Verzia	Autor	Stručný popis úpravy
10.12.2014	2	Jakub Kmeťko	Aktualizácia postupov pri vytváraní dokumentácie, a aktualizácie šablóny k dokumentácii modulu

Metodika definuje pracovné postupy, ktorými sa členovia tímu budú riadiť pri písaní dokumentácie. Okrem pokynov pre vytváranie a písanie konkrétnych dokumentov, informáciách o ich štruktúre a formátovaní, obsahuje aj model revízneho/schvaľovacieho procesu.

7.2.1 Skratky a pojmy

MS Word – Microsoft Word

gDocs – Google Documents

gDrive – [Google Drive](#)

Drop Box

7.2.2 Vysvetlenie pojmov

MS Word – textový procesor, ktorý používame na vytváranie finálnych, aj lokálnych verzií dokumentov, a ich spájanie do oficiálnej verzie. Dbá sa na formátovanie podľa šablón v dropbox (viz. 7.2.2 - DropBox)

gDocs – nástroj na online editáciu v textovom, tabuľkovom, a prezentačnom procesore. Vytvorené dokumenty nie sú ďalej zdieľané v rámci celého tímu prostredníctvom tohto nástroja

gDrive – online zdieľaný priečinok, v ktorom sú dostupné všetky gDocs, no jeho používanie sme zastavili

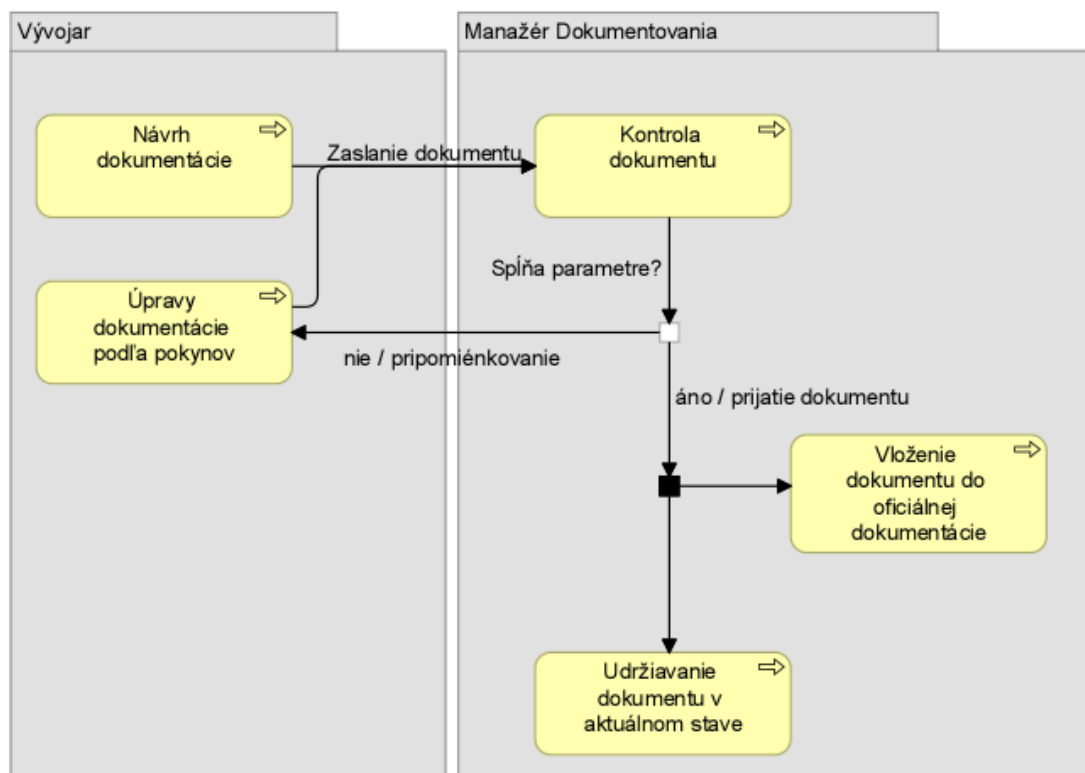
DropBox – tímové úložisko pre vzniknuté dokumenty. Pri pristupovaní k nim je nutné pred otvorením dokumentu zapísať na koniec jeho názvu „-LOCK-meno-priezvisko“, pričom v takto uzavretom dokumente môže v jednom čase pracovať iba človek, ktorý si dokument uzavrel. Po dokončení svojej práce, alebo na vyzvanie manažérom dokumentácie je tento človek ukončiť prácu na dokumente a odstrániť z jeho názvu koncovku „-LOCK-meno-priezvisko“.

7.2.3 Postup dokumentovania

Postup:

1. Vývojár vytvorí návrh systémovej dokumentácie vyplnením relevantných častí šablóny pre dokumentáciu (nachádzajúcej sa v DropBox/dokumentácia/sablony), a zašle ho manažérovi dokumentácie najneskôr dva dni pred ukončením šprintu.
2. Manažér dokumentovania skontroluje prijatí dokument a zhodnotí ho. Ak dokument nespĺňa požiadavky, manažér dokumentácie ho vráti vývojárovi, spolu s pripomienkami.
3. Vývojár úpravy/doplní segmenty dohodnuté v bode 3., a výsledok zašle Manažérovi dokumentácie

4. Manažér dokumentácie reviduje zmeny, výsledný dokument schváli a vloží do oficiálnej dokumentácie
5. Manažér dokumentácie vloží dokument do oficiálnej dokumentácie, a od tohto bodu preberá zodpovednosť za udržiavanie dokumentu v aktuálnom stave



Obr. 1.1 – work flow diagram k postupu vytváranie dokumentácie

7.2.4 Spáva verzií dokumentov

Vývojár zaznamená každú zmenu vykonanú na svojich dokumentoch, s tým, že túto informáciu zaznačí do tabuľky „Verzie dokumentu“ (viz. Tabuľka 1.3.1).

Manažér dokumentácie zaznamená každú zmenu vykonanú na oficiálnom dokumente, s tým, že túto informáciu zaznačí do tabuľky „Verzie dokumentácie“, ktorá ma rovnakú štruktúru ako Tabuľka 1.3.1.

Dátum	Verzia	Autor	Sekcia	Stručný popis úpravy

Tabuľka 3 – Verzie dokumentu

7.2.5 Správa a formátovanie dokumentov

7.2.5.1 Vytváranie dokumentov na DropBox

Dokumenty vytvárame pod názvom #ID_ulohy-Nazov_ulohy, kde:

- Id_ulohy je identifikátor konkrétnej úlohy, pridelený systémom JIRA
- Názov úlohy je stručná konkretizácia problematiky, pridelená na tímovom stretnutí

7.2.6 Formátovanie dokumentov

7.2.6.1 Štandardný text:

- Font – Calibri light
- Veľkosť písma – 11
- Riadkovanie – 1.1

7.2.6.2 Nadpisy:

Pre lepšiu orientáciu, kaskádujeme na základe číslicového označenia, bez ohľadu na hĺbku pod-levelu.

Nadpis – Heading 1:

Označujú najvyššiu váhu kategorizácie, ktorá vyjadruje vysokú dôležitosť témy, alebo pod ňu spadá väčší logický celok

- Veľkosť písma - 11
- Pozadie - modré
- Farba - biela
- Šírka - 100%

Nadpis – Heading 2:

Označuje druhú najvyššiu váhu kategorizácie, pod ktorú spadá významný celok, patriaci pod kategóriu s nadpisom Heading 1

- Veľkosť písma: 11
- Pozadie: svetlo-modrá
- Farba: čierna
- Šírka – 100%

Nadpis – Heading 3:

Označuje druhú najvyššiu váhu kategorizácie, pod ktorú spadá významný celok, patriaci pod kategóriu s nadpisom Heading 2

- Veľkosť písma: 11
- Pozadie: transparentné
- Farba: modrá
- Šírka – 100%
- Vrchné orámovanie - modré

7.2.6.3 Obrázky:

- Zarovnať na stred
- Pridať popis v tvare „Obrázok # - názov“
- # označuje poradové číslo obrázka v dokumente
- názov označuje názov obrázku

7.2.6.4 Tabuľky:

- Zarovnať na stred
- Pridať popis v tvare „Tabuľka # - názov“
- # označuje poradové číslo tabuľky v dokumente
- názov označuje názov tabuľky

7.2.7 Dokumentácia modulu

7.2.8 Názov úlohy (z User Story)

7.2.8.1 Špecifikácia

High-level opis špecifikácie user story / úlohy z biznis pohľadu.

Zahrňa biznis procesy a komponenty vstupujúce do biznis procesu.

7.2.8.2 Analýza

Analýza problematiky a navrhované riešenia. Ak je riešení viacero, doplniť metriky, na základe ktorých sme sa rozhodli (porovnanie).

7.2.8.3 Model

Ak je možné, respektíve vhodné, vložiť model z UML. Pokiaľ sa jedná o vizuálne rozhranie, priložiť obrázok.

7.2.8.4 Implementácia

Popis postupu implementácie, a detaily implementácie.

Na konci pridať zoznam použitých technológií a ich verzie, plus odkaz na dokumentáciu a hlavnú stránku tretej strany.

7.2.8.5 Testovanie

Stručný opis použitých technológií a ich verzie.

Podľa metodiky testovania. Šablona na: DropBox/testy/sablona.docx

7.2.9 Šablóna pre tvorbu zápisníc z tímového stretnutia

Zápisnica č: # zo stretnutia tímu 03

Miesto: FIIT STU, miestnosť

Dátum: DD.MM.YYYY

Čas:

Vedúci stretnutia: Ing. Michal Holub

Zapisovateľ:

Prítomní: Ing. Michal Holub
Bc. Igor Daniš
Bc. Martin Košut
Bc. Martin Lošák
Bc. Alex Ostrovský
Bc. Stanislav Paľove
Bc. Peter Uherek
Bc. Jakub Kmeťko

Neprítomní: Meno chýbajúceho člena (*dôvod neprítomnosti*)

Téma stretnutia: *Doplniť preberané témy*

Priebeh stretnutia

- *Návrhy, prednesy, dohody – v tvare:*
 - On navrhol toto, a takto argumentoval
 - On predložil protinávrh, a takto argumentoval
 - Tím sa nakoniec dohodol na
- Zosumarizovanie výsledkov úloh z predchádzajúceho stretnutia:
 - Igor Daniš:
 - Peter Uherek:
 - Martin Lošák:
 - Stano Paľove:
 - Alex Ostrovský:
 - Jakub Kmeťko:
 - Martin Košut:

Bodovanie aktuálneho šprintu

ID	Popis úlohy	Zodpovednosť	Body

Tabuľka 4 – Bodovanie aktuálneho šprintu

Plnenie úloh z predchádzajúcich stretnutí

ID	Popis úlohy	Zodpovednosť	Termín zač.	Termín uk.	Stav

Tabuľka 5 – Plnenie úloh z predchádzajúcich stretnutí

- Lessons Learned
 - zoznam dôležitých zistení počas šprintu, z ktorých sa do budúcnosti poučíme