Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

3D UML

Dokumentácia k riadeniu projektu

Tím č. 15

Vedúci tímu: Ing. Ivan Polášek, Phd.

Členovia tímu: Bc. Matej Víťaz, Bc. Ľubomír Jesze, Bc. Lukáš Skala, Bc. Peter Zajac, Bc. Dominik Žilka, Bc. Matúš Gáspár, Bc. Jakub Minárik

Akademický rok: 2016/2017

Obsah

1 Úvod 5

2 Role členov tímu 5

3 Podiel práce na vytváraní dokumentácie 5

4 Aplikácie manažmentov 6

4.1 Manažment plánovania 6

4.2 Manažment rizík 6

4.3 Manažment vývoja a integrácie 7

4.4 Manažment dokumentácie 7

4.5 Manažment komunikácie 7

4.6 Manažment testovania 7

5 Sumarizácie šprintov 8

5.1.1 Retrospektíva – 1. šprint Ciele šprintu 8

5.1.2 Čo išlo podľa našich predstáv 8

5.1.3 Čo nešlo podľa našich predstáv 8

5.1.4 Čo plánujeme zlepšiť v nadchádzajúcom šprinte 8

5.2 Retrospektíva - 2. šprint 8

5.2.1 Trvanie: 25.10. – 8.11.2016 8

5.2.2 Ciele šprintu 8

5.2.3 Čo išlo podľa našich predstáv 8

5.2.4 Čo nešlo podľa našich predstáv 9

5.2.5 Čo plánujeme zlepšiť v nadchádzajúcom šprinte 9

6 Používané metodiky 9

6.1 Metodika rizík 9

6.2 Metodika plánovania 10

6.3 Opodstatnenosť, dedikácia a účel metodiky 10

6.3.1 Identifikácia a opis rol 10

6.3.2 Identifikácia a opis procesov 10

6.4 Metodika kontroly kvality kódu 10

6.4.1 Opodstatnenosť, dedikácia a účel metodiky 10

6.4.2 Identifikácia a opis rol 11

6.4.3 Vstupy 11

6.4.4 Identifikácia a opis procesov 11

6.4.5 Výstupy metodiky 11

6.4.6 Nadväzujúce metodiky 11

6.5 Metodika vývoja a integrácie 11

6.5.1 Opodstatnenosť, dedikácia a účel metodiky 11

6.5.2 Detaily vývoja a integrácie 11

6.5.3 Vstupy 11

6.5.4 Fázy vývoja a roly 12
6.5.5 Opis procesov vývoja 12
6.5.6 Výstupy metodiky 13
6.5.7 Nadväzujúce metodiky 13
6.6 Metodika testovania a nasadenia 14
6.6.1 Identifikácia a opis rol 14
6.6.2 Identifikácia a opis (manažérskych) procesov 14
6.7 Metodika dokumentácie 14
6.7.1 Identifikácia a opis rol 15
6.7.2 Identifikácia a opis (manažérskych) procesov 15
6.8 Metodika komunikácie 17
6.8.1 Identifikácia a opis rol 18
6.8.2 Identifikácia a opis (manžérskych) procesov 18
7 Globálna retrospektíva ZS/LS 19
7.1.1 Čo išlo podľa našich predstáv 19
7.1.2 Čo nešlo podľa našich predstáv 19
7.1.3 Čo plánujeme zlepšiť 19
8 Motivačný dokument 19
9 Export evidencie úloh 22
9.1 Šprint 1 22
9.1.1 Ako pouzivatel chcem ulozit sekvencny diagram, aby som sa k nemu mohol neskor vratit. 22
9.1.2 Ako tím potrebujeme softvérové nástroje pre manažment v tíme a podporné vývojové nástroje pre lepšiu organizáciu a komunikáciu v rámci tímu. 23
9.2 Šprint 2 24
9.2.1 Ako tím chceme integrovať knižnicu Go.JS do projektu 24
9.2.2 Ako tím chceme integrovať do projektu JSONAPI 25
9.2.3 Ako tím chceme integrovať do projektu Angular.js na zjednodušenie vývoja 26
9.2.4 Ako tím chceme MVC architektúru pre projekt, aby sme mohli dekomponovať aplikáciu. 26
9.3 Šprint 3 27
9.3.1 Ako tím chceme zverejniť výstupnú aplikáciu na produkčnom serveri 27
9.3.2 Ako používateľ chcem načítať predtým uložený sekvenčný diagram 28
9.3.3 Ako pouzivatel chcem upravit interakciu. 29
9.3.4 Ako pouzivatel chcem vymazat interakciu. 29
9.3.5 Ako pouzivatel chcem pridat interakciu. 30
9.3.6 Ako pouzivatel chcem zmazat lifeline 31
9.3.7 Ako pouzivatel chcem pridat lifeline. 32
9.3.8 [Ako používateľ chcem prostredníctvom 3D UML aplikácie vytvárať plátna sekvenčného diagramu, pretože plátna tvoria základ každého 3D UML digramu 32

9.4 Príloha A - Zápisnice 33

1 Úvod

Na nasledujúcich stranách sa nachádza dokumentácia k riadeniu tímového projektu, v rámci zimného semestra 2016/2017 na Fakulte informatiky a informačných technológií na STU v Bratislave. Nachádza sa tu podiel práce pri vytváraní dokumentácie, role členov tímu a podiel práce na projekte. V ďalších odsekoch sú rozobrané aplikácie jednotlivých manažmentov v tíme. Postupne sú rozobrané jednotlivé šprinty a ich priebeh. Metodológie, ktorými sa v našom tíme riadime sú detailne rozpísané a vysvetlené. V závere dokumentu sa nachádza globálna retrospektíva a prílohy.

2 Role členov tímu

#	*Rola člena tímu*	*Člen tímu*
1	*Dokumentácia a code review*	Bc. Gáspár Matúš
2	*Head developer*	Bc. Jesze Ľubomír
3	*Databázový špecialista*	Bc. Minárik Jakub
4	*Plánovanie*	Bc. Skala Lukáš
5	*Vedúci tímu a developer*	Bc. Víťaz Matej
6	*Testovanie a kvalita*	Bc. Zajac Peter
7	*Scrum Master a komunikácia*	Bc. Žilka Dominik

3 Podiel práce na vytváraní dokumentácie

#	**Časť dokumentácie**	**Člen tímu**
1	Úvod	Bc. Gáspár Matúš, Bc. Žilka Dominik
2	Manažment plánovania	Bc. Gáspár Matúš
3	Manažment rizík	Bc. Gáspár Matúš
4	Manažment vývoja a integrácie	Bc. Víťaz Matej
5	Manažment dokumentácie	Bc. Gáspár Matúš
6	Manažment komunikácie	Bc. Žilka Dominik
7	Manažment testovania	Bc. Zajac Peter
8	Sumarizácie šprintov	Bc. Zajac Peter, Bc. Minárik Jakub
9	Metodika rizík	Bc. Minárik Jakub
10	Metodika plánovania	Bc. Skala Lukáš
11	Metodika kontroly kvality kódu	Bc. Jesze Ľubomír
12	Metodika vývoja a integrácie	Bc. Víťaz Matej
13	Metodika testovania a nasadenia	Bc. Zajac Peter
14	Metodika dokumentácie	Bc. Gáspár Matúš
15	Metodika komunikácie	Bc. Žilka Dominik
16	Globálna retrospektíva	Bc. Skala Lukáš
17	Export evidencie úloh	Bc. Skala Lukáš
18	Globálne ciele projektu	Bc. Jesze Ľubomír
19	Celkový pohľad na 3D UML aplikáciu	Bc. Minárik Jakub
20	Vykresľovanie diagramov Analýza Návrh Implementácia Testovanie	Bc. Víťaz Matej Bc. Zajac Peter, Bc. Skala Lukáš Bc. Minárik Jakub Bc. Jesze Ľubomír, Bc. Víťaz Matej Bc. Zajac Peter
21	Serverová aplikácia Analýza	Bc. Jesze Ľubomír Bc. Gáspár Matúš, Bc. Minárik Jakub, Bc. Žilka Dominik

	Návrh Implementácia Testovanie	Bc. Gáspár Matúš, Bc. Žilka Dominik Bc. Jesze Ľubomír, Bc. Víťaz Matej Bc. Zajac Peter
22	Jadro klientskej aplikácie Analýza Návrh Implementácia Testovanie	Bc. Minárik Jakub Bc. Minárik Jakub Bc. Minárik Jakub Bc. Jesze Ľubomír, Bc. Víťaz Matej Bc. Zajac Peter

4 Aplikácie manažmentov

4.1 Manažment plánovania

Plánovanie je jedným z veľmi dôležitých etáp životného cyklu tvorby softvéru. Je potrebné mať naplánované, kedy sa ktoré časti softvéru navrhujú, implementujú a testujú, a taktiež je potrebné naplánovať ciele, ktoré chceme v rámci nášho projektu dosiahnuť. Plánuje sa tak isto aj čas a úsilie, ktoré je potrebné vynaložiť pri spĺňaní jednotlivých úloh. Plánovanie prebieha v nami určenom nástroji pre lepšiu organizáciu a jednotný a zdieľaný prístup k úlohám.

Dôvodom, prečo v našom projekte existuje tento typ manažmentu je ten, že sa ukázali nedostatky v rámci toho, kedy, ako a kto bude vykonávať požadované úlohy. Tomuto sme chceli predísť a vytvoriť jednotný tím, ktorý bude prosperovať spoločne a každý bude mať naplánované svoje úlohy. Úlohou tohto manažmentu je aj to, aby sme si prácu rozdelili a nevykonávali rovnaké úlohy a tým nevznikali duplicity v rámci tvorby softvéru. Takisto je potrebné sledovať plnenie naplánovaných úloh v nami zvolenom nástroji - JIRA. Sledovanie je možné burndown chartom alebo označením “resolved” pri danej naplánovanej úlohe.

Týmto sme vytvorili stabilný proces, ktorý je funkčný a takto dokážeme tvoriť a vykonávať úlohy oddelene a zároveň efektívne.

4.2 Manažment rizík

Pri tvorbe každého projektu dochádza k rizikám, ktoré môžu nastať. Jedným z rizík je nesprávne naplánovanie úloh a nesprávne pridelenie odhadovaného času úsilia. Keď sa naplánuje príliš veľa úloh v rámci jedného šprintu, nastáva riziko nesplnenia stanovených požiadaviek, a teda nesplnenia požiadaviek zákazníka. Na druhej strane, ak sa stanovia iba dve úlohy s veľkým odhadovaným časom úsilia, to taktiež znamená riziko nesplnenia, keďže tieto úlohy sa pravdepodobne dajú rozdeliť ešte do viacerých podúloh.

Využívame nástroj JIRA, v ktorom hodnotíme “user stories”. Na každom stretnutí ohodnotíme dané úlohy a podľa produktového backlogu ich potom vyberáme do šprintu podľa odhadovaného času úsilia a náročnosti. Toto úsilie je zvolené a odhadované nami na stretnutiach, kde si každý ukáže náročnosť podľa seba a následne, po diskusii, dospejeme k spoločnému záveru a odhadneme finálnu náročnosť.

Potrebné je sledovať aj to, aby sme mali neustále funkčný prototyp, ktorý si môže zákazník kedykoľvek pozrieť. Teda tu prichádza riziko spojené s nefunkčným prototyp a tým by sme nesplnili požiadavku zákazníka. Riešenie pozostáva z toho, že udržujeme prototyp vo funkčnom stave stále a pridáva sa funkcionalita len na základe dôkladného code review a testovania, a tým sa predíde nefunkčnosti prototypu.

4.3 Manažment vývoja a integrácie

Pri vytváraní produktu je nutné dbať na to, aby bol vyvíjaný priebežne a bol dostupný stále funkčný prototyp aplikácie, keďže využívame agilný vývoj a metodiku SCRUM. Jednotlivé časti funkcionality aplikácie sú dodávané v ucelených jednotkách – moduloch. Možnosti a funkcie GIT-u nám poskytujú požadované prostredie pre vývoj takéhoto druhu. Jednotlivé moduly vyvíjanej aplikácie je možné vytvárať samostatne vo vetvách repozitára a zároveň udržiavať stále funkčný prototyp aplikácie v master vetve. Tento spôsob práce s GIT-om sme si zvolili, kvôli možnosti paralelnej práce na viacerých úlohách súčasne.

Podrobnosti sú opísane v metodike vývoja a integrácie.

4.4 Manažment dokumentácie

Tvorba dokumentácie slúži na zdokumentovanie výstupu nášho výsledného produktu. Potrebná je z toho dôvodu, aby sme mali ucelený výstup z projektu v písanej forme, a aby sme sa mohli k nemu kedykoľvek vrátiť, odvolávať sa na neho, dopĺňať prípadné zmeny a v prípade pokračovania, poskytli výstup iným vývojárom.

Dokumenty, ktoré v projekte používame sú:

- **Dokumentácia k riadeniu**
- **Dokumentácia k inžinierskemu dielu**
- **Zápisnice zo stretnutí**
- **Dokumenty o retrospektíve**
- **Dokumenty o používaných metodikách**

4.5 Manažment komunikácie

Počas behu projektu je nevyhnutná správna komunikácia medzi členmi tímu, a taktiež medzi tímom a ostatnými osobami. Na tieto účely využívame už vytvorené služby, ktoré nám komunikáciu sprehľadňujú a uľahčujú. Na komunikáciu so zákazníkom využívame GoogleGroups. Na komunikáciu medzi členmi tímu používame službu Hipchat. Na Hipchate si vytvárame pre každý task alebo problém nový room a tam o ňom diskutujeme. Takto potom prehľadne vieme kde sa čo nachádza. Aj na verbálnej úrovni má náš tím jasne stanovené pravidlá. Aby sme nestrácali čas dohadovaním sa a skákaním si do reči, máme určené pravidlá rétoriky. Tieto metódy sú uvedené v metodike komunikácie nižšie.

4.6 Manažment testovania

Po implementácií jednotlivých modulov je potrebné implementáciu otestovať. Testovanie realizujeme v našom projekte pomocou porovnania výstupov s požadovanými výsledkami definovanými v user stories v nástroji JIRA. Taktiež používame metódu "unit testing". Táto metóda je automatické testovanie a overovanie funkčnosti a korektnosti jednotlivých častí (unit) zdrojového kódu. Túto metódu využíva nástroj Bamboo, ktorý je súčasťou nástroja JIRA.

Po implementácii jednotlivých user stories vykonávame nad pridanou funkcionalitou sériu manuálnych testov, ktoré má povinnosť vypracovať člen tímu, ktorý má za úlohu implementovať danú funkcionalitu. Pri tomto testovaní sa využívajú jednoduché testy porovnania výsledkov s požadovanými výstupmi. Po pridaní nového zdrojového kódu sa v nástroji Bamboo pomocou preddefinovaných príkazov na vykonanie unit testov nad novým zdrojovým kódom. Snažíme sa týmto manažmentom testovania dosiahnuť aby bol zdrojový kód aplikácie korektný a správne fungoval.

5 Sumarizácie šprintov

5.1.1 Retrospektíva – 1. šprint

Ciele šprintu

- Nastavenie softvérových nástrojov pre manažment v tíme a odporných vývojových nástrojov pre lepšiu komunikáciu v tíme (JIRA, Hipchat).
- Nastavenie repozitára na GitHube s možnosťou Code Review a kontrolou Code style
- Nasadenie webového sídla na školský server.
- Vytvorenie jednoduchého prototypu sekvenčného diagramu vo WebGL
- Analýza testovacích scenárov minuloročného tímového projektu
- Stanovenie vývojového prostredia pre backend a frontend

5.1.2 Čo išlo podľa našich predstáv

- Nastavili sa potrebné nástroje pre komunikáciu a manažment úloh v tíme
- Analýza testovacích scenárov minuloročného tímového projektu
- Nastavili sa potrebné veci na GitHube
- Nasadenie webového sídla na GitHub pages a školský server
- Bol vytvorený jednoduchý prototyp sekvenčného diagramu vo WebGL za účelom vyhodnotenie použiteľnosti danej technológie
- Bolo stanovené vývojové prostredie pre backend a databázu – PHP framework Laravel, PosgreSQL databáza

5.1.3 Čo nešlo podľa našich predstáv

- Problémy pri určení vývojového prostredia pre frontend (WebGL alebo CSS3D)
- Zdržanie vo vývoji v dôsledku zmeny prostredia z WebGL do CSS3D transformácií
- Problémy s nahodením webového sídla na školský server

5.1.4 Čo plánujeme zlepšiť v nadchádzajúcom šprinte

- Komunikáciu v tíme – vytváranie miestností na Hipchate k daným úlohám
- Finálne definovanie vývojového prostredia frontendu – knižnica pre elementy sekvenčného diagramu /vytvorenie vlastných elementov pre diagram
- Vytvorenie dátového modelu zo metamodelu, ktorý je diagram tried
- Lepšie plnenie úloh – skôr ako tesne pred deadlinom a koncom šprintu

5.2 Retrospektíva - 2. šprint

5.2.1 Trvanie: 25.10. – 8.11.2016

5.2.2 Ciele šprintu

- Vytvorenie MVC architektúry pre projekt, aby sme mohli dekomponovať aplikáciu
- Integrovanie Angular.js do projektu na zjednodušenie vývoja
- Integrovanie JSONAPI do projektu
- Integrovanie knižnice Go.js do projektu

5.2.3 Čo išlo podľa našich predstáv

- Podarilo sa nám splniť všetky stanovené úlohy
- Spojazdnenie frontendu projektu
- Úspešné integrovanie Go.js do projektu
- Úspešné integrovanie JSONAPI do projektu
- Úspešné integrovanie Angular.js do projektu
- Využitie tímového ducha na definitívne stanovenie použitej technológie

- Komunikácia s vedúcim a komunikácia v tíme
- Vzájomná výpomoc s problémami od ostatných členov tímu
- Konštruktívna diskusia s kolegami o problémoch a úlohách

5.2.4 Čo nešlo podľa našich predstáv

- Kvôli zmene technológie v strede šprintu sa nám nepáči burndown chart
- Problémy s vytvorením fyzického modelu z metamodelu
- Metodiky nie sú vo finálnej verzii
- Nahradenie jedného stretnutia, ktoré odpadlo kvôli štátnemu sviatku
- Reorganizácia miestnosti na tímové stretnutia, spôsobujúca oddelenie členov tímu a následnú zhoršenú kolaboráciu

5.2.5 Čo plánujeme zlepšiť v nadchádzajúcom šprinte

- Komunikáciu a spoluprácu v rámci tímu
- Systematickejšia práca na jednotlivých úlohách
- Odhad pri stanovení obtiažnosti úloh

6 Používané metodiky

6.1 Metodika rizík

Definuje správne postupy riešenia rizík, ktoré sa môžu vyskytnúť pri projekte. Tieto riziká súvisia so všetkými oblasťami vývoja. Napríklad pri návrhu a plánovaní môže nastať, že zle odhadneme náročnosť úlohy a čas potrebný na jej vypracovanie. Preto pri každej fáze projektu je potrebné identifikovať a analyzovať riziká. Pre všetky úlohy podľa šablóny takto explicitne :

1. Určiť pravdepodobnosť, že nastane negatívna udalosť, ktorá ovplyvní chod projektu
   1.1. Identifikovať udalosť, ktorá ovplyvní chod projektu
   1.2. Do detailov skúmať možnosti, kedy táto situácia môže nastať
   1.3. Určiť slabé stránky daných možností zlyhania
      1.3.1.Technologické
      1.3.2.Časové
2. Určiť príznaky, že táto udalosť nastane
3. Dopady tejto udalosti
   3.1. Časové dopady
   3.2. Technologické dopady (je treba zmeniť technológiu)
   3.3. Komunikačné dopady (problém nastal, lebo situácia bola zle odkomunikovaná v rámci tímu)
4. Predídenie tejto udalosti
   4.1. Navrhnúť riešenia problému
   4.2. Testovať či riešenia naozaj vyriešia daný problém
   4.3. Nasadenie riešenia
5. Ako sa preventívne opatrenia odrazili na pláne
   5.1. Planning poker
   5.2. Analýza, či sa daným spôsobom riziko oplatí riešiť
   5.3. Preplánovanie práce
6. Zmiernenie následkov udalosti, ak sa jej nepodarilo zabrániť
   6.1. Analýza vhodných metód zmiernenia problému
   6.2. Oprava problému aspoň na čiastočnej úrovni

6.2 Metodika plánovania

6.3 Opodstatnenosť, dedikácia a účel metodiky

Plánovanie je jednou z najdôležitejších činností pri tvorbe softvéru. Je veľmi dôležité aby boli jednotlivé etapy v ktorých sa softvér nachádza dobre špecifikované z pohľadu plánovania aby bol vývoj softvéru kontinuálny a správny, na základe čoho sa dopracujeme k vytýčenému cieľu.

Z pohľadu oblastí riadenia je v projekte možné naplánovať následnosť činností, rozsah projektu, časovú náročnosť a náklady, či pridelenie ľudských zdrojov k jednotlivým činnostiam.

Metodikou plánovania sa riadia všetci členovia tímu na stretnutiach k projektu, pri nasledujúcich procesoch plánovania.

Nadväzujúce metodiky:

1. Metodika dokumentácie
2. Metodika komunikácie

6.3.1 Identifikácia a opis rol

Roly: tím, vedúci plánovania, vedúci dokumentácie, vlastník

Vlastník - zadávateľ daného projektu

Vedúci plánovania - člen tímu, zodpovedný za plánovanie v rámci projektu

Vedúci dokumentácie - člen tímu, zodpovedný za dokumentáciu v rámci projektu

Vedúci tímu - člen tímu, ktorý je zodpovedný za projekt

Tím - skupina ľudí, ktorí spoločne pracujú na jednom projekte

6.3.2 Identifikácia a opis procesov

6.3.2.1 Tvorba produktového backlogu

Vstupy – diskusia v tíme, požiadavky zadávateľa

Výstupy - produktový backlog

Na tvorbe produktového backlogu sa podieľa celý tím a vlastník, respektíve zadávateľ projektu ktorý určí požiadavky, ktoré musí vytváraný softvér spĺňať. Pri určovaní požiadaviek na softvér vedie diskusiu vedúci tímu, pričom prebieha diskusia medzi tímom a vlastníkom. Prostredníctvom tejto diskusie tím zisťuje od vlastníka všetky potrebné špecifikácie, ktoré je potrebné vedieť pre návrh softvéru. Vedúci dokumentácie tieto požiadavky spíše do dokumentu.

Používateľské príbehy (user stories) tvoríme prostredníctvom diskusie celého tímu, ktorú vedie vedúci plánovania. Vedúci plánovania vyberie požiadavku a otvorí diskusiu v ktorej členovia tímu navrhujú zadanie používateľského príbehu. Následne tím za jednotlivé znenia používateľských príbehov hlasuje a na základe hlasovania vyberie používateľský príbeh pre danú požiadavku, ktorý vložíme do produktového backlogu. Tento proces iterujeme nad všetkými požiadavkami.

6.4 Metodika kontroly kvality kódu

6.4.1 Opodstatnenosť, dedikácia a účel metodiky

Základ projektu tvorí zdrojový kód produktu a preto je veľmi dôležitá jeho organizácia a štábna kultúra. Hlavným dôvodom je prehľadnosť kódu a to nielen medzi členmi tímu ale aj pre potenciálnych budúcich vývojárov produktu, ktorí budú nadväzovať na nami vytvorenú aplikáciu. Taktiež je medzi členmi tímu

veľmi dôležitá vzájomná kontrola, ktorá je prospešná pre celkovú kvalitu kódu, nižšiu chybovosť a taktiež odovzdanie vedomostí o naprogramovanom kóde aplikácie.

6.4.2 Identifikácia a opis rol

Programátor - autor zdrojového kódu

Kontrolér kvality – autor code review

6.4.3 Vstupy

Vstupom pre kontrolu kvality kódu je zdrojový kód, ktorý vznikol ako výstup podľa metodiky vývoja a integrácie.

6.4.4 Identifikácia a opis procesov

6.4.4.1 Kontrola programátorského štýlu (coding style)

V projekte je integrovaná kontrola štýlu na strane klienta (Typescript a Angular) pomocou odporúčaných pravidiel pre framework Angular 2. Taktiež pre serverovú aplikáciu v PHP s použitím frameworku Laravel sú použité odporúčané pravidlá PSR-2 a PSR-4 pre autoloading tried.

Kontrolu kódu možno spustiť nasledovným príkazom v koreňovom adresáre projektu pre PHP

```
$ php-cs-fixer fix
```

A pre klientskú časť v Angulari

```
$ npm run lint
```

6.4.4.2 Kontrola kódu (code review)

Kód musí po vytvorení prejsť kontrolou kódu na GitHube. Aby bolo možné kód spojiť s hlavnou vetvou musí byť chválená aspoň jedným ďalším členom tímu. Ak boli vyžadované zmeny, je potrebné kód upraviť podľa požiadaviek až pokým nie je všetko podľa požiadaviek.

6.4.5 Výstupy metodiky

1. Kód, ktorý spĺňa stanovené pravidlá
2. Code review

6.4.6 Nadväzujúce metodiky

1. Metodika vývoja a integrácie
2. Metodika testovania a nasadenia

6.5 Metodika vývoja a integrácie

6.5.1 Opodstatnenosť, dedikácia a účel metodiky

Metodika vývoja a integrácie je určená pre developerov, ktorých úlohou je zapojiť sa do vývoja aplikácie. Cieľom metodiky je oboznámiť so zaužívanými spôsobmi a metódami vývoja v našom tíme a ozrejmiť potrebné detaily v jednotlivých fázach životného cyklu modulu aplikácie. Vďaka metodike bude programátor schopný vedieť rýchlo nadviazať na kolektívnu prácu tímu a bude schopný zapojiť sa do vývoja aplikácie.

6.5.2 Detaily vývoja a integrácie

6.5.3 Vstupy

Keďže vyvíjaná aplikácia je iteratívne rozširovaná o ucelené jednotky funkcionality, nazývané moduly, je potrebné najskôr poznať špecifikáciu tejto funkcionality. Hlavným vstupom metodiky vývoja a integrácie je teda konkrétne zadanie úlohy, ktorú ideme plniť. Zadanie úlohy je výstupom metodiky plánovania.

6.5.4 Fázy vývoja a roly

Vývoj modulu aplikácie je možné dekomponovať na štyri hlavné fázy, pričom na jednotlivé fázy vývoja je možné nahliadať ako na procesy v rámci vývoja modulu aplikácie. Procesy vývoja modulu aplikácie:

1. návrh a vývoj modulu,
2. overenie zdrojového kódu,
3. otestovanie modulu,
4. integrovanie do novej verzie aplikácie.

Integrovanie nového modulu, ktorý obsahuje požadovanú funkcionalitu je podmienené úspešným dokončením všetkých štyroch procesov vývoja. V prípade potreby je možné sa viac krát vracať na predchádzajúce procesy. Typicky z procesu testovania je možné viackrát iterovať na proces návrhu a vývoja modulu, až pokým nebude funkcionalita vyvíjaného modulu spĺňať špecifikované požiadavky.

K jednotlivým procesom vývoja modulu aplikácie prislúchajú roly členov tímu. Člen tímu zastupujúci určitú rolu je zodpovedný za príslušný proces vývoja modulu aplikácie. V praxi je však bežné, že v rôznych procesoch vývoja sú jednotlivé roly zastúpené jedným členom nášho tímu. Nasledujúca tabuľka zobrazuje priradenie procesu vývoja modulu aplikácie k role člena tímu.

Proces vývoja modulu aplikácie	**Rola člena tímu**
Návrh a vývoj modulu	Programátor
Overenie zdrojového kódu	Kontrolér kódu
Otestovanie modulu	Tester funkcionality
Integrovanie do novej verzie aplikácie	Správca repozitára

6.5.5 Opis procesov vývoja

6.5.5.1 Návrh a vývoj modulu

1. **Inštalácia nástroja git** – nástroj git je nevyhnutným predpokladom na úspešné začlenenie nového programátora do procesu vývoja modulov aplikácie.
2. **Vytvorenie konta na GitHube** – zdrojové kódy nami vyvíjanej sú prístupné pomocou služby GitHub. Táto služba taktiež poskytuje grafické rozhranie pre jednoduchšiu správu repozitára aplikácie a taktiež možnosť pracovať s pull requestami.
3. **Prístup k repozitáru** – pre zapisovanie do repozitára je nutné požiadať o udelenie povolenia správcu repozitára.
4. **Klonovanie repozitára na lokálny počítač** – git je distribuovaný systém na správu verzií, preto je nutné pred začiatkom vývoja nového modulu zdrojový kód aplikácie naklonovať na lokálny počítač programátora. Detailný postup sa vždy nachádza v súbore `readme.md` repozitára aplikácie.
5. **Inštalácia závislostí** – vyvíjaná aplikácie závisí na moduloch, komponentoch a knižniciach od tretích strán, preto je potrebné tieto závislosti nainštalovať. Detailný postup je uvedený v súbore `readme.md` repozitára aplikácie.
6. **Vytvorenie novej vetvy** – každý modul aplikácie je vyvíjaný v samostatnej vetve repozitára. Vytvorenie novej vetvy je možné vykonať príkazom:

```
$ git checkout -b názov_vetvy
```

7. **Naprogramovanie zmien –** nami vyvíjaná aplikácia je vytvorená pre prostredia programovacích jazykov TypeScript a PHP. Vykonané zmeny v kóde je potrebné spracovať s ohľadom na túto skutočnosť.

8. **Naprogramovanie unit testov –** pre nové jednotky kódu je vhodné vytvoriť unit testy.

9. **Zverejnenie zmien** – jednotlivé zmeny zdrojového kódu je potrebné zverejniť príkazom:
   `$ git commit -a -m "popis_zmien"`

10. **Uloženie zmien do repozitára** – aby mali prístup k zmenám zdrojového kódu aj ostatní členovia tímu, je potrebné tieto zmeny uložiť do repozitára príkazom:
    `$ git push origin názov_vetvy`

11. **Pull request** – po uložení všetkých zmien do vetvy repozitára je potrebné vytvoriť pull request, aby bolo možné danú vetvu spojiť s hlavnou vetvou master. Aby bolo možné vetvu spojiť, musia byt splnené nasledujúce požiadavky:

    a. nie je možné vykonávať zmeny priamo vo vetve master,

    b. unit testy a build aplikácie musia korektne fungovať.

6.5.5.2 Overenie zdrojového kódu

12. **Overenie štýlu kódu** – služba Bamboo sa postrará o vykonanie code-style tesov. V prípade zistených problémov, je ich potrebné opraviť.

13. **Code review** – ostatní členovia tímu musia prehliadnuť vykonané zmeny a potvrdiť ich správnosť, až potom bdue možné zmeny spojiť s vetvou master.

6.5.5.3 Otestovanie kvality

14. **Otestovanie unit testov** – v prípade, že unit testy zlyhajú, nebude možné vykonané zmeny spojiť s vetvou master.

15. **Otestovanie funkčnosti** – tester spustí aplikáciu a odskúša správnosť všetkej funkcionality.

Integrovanie do novej verzie aplikácie

16. **Spojenie vetiev** – správca repozitára spojí vetvy, čím integruje nový modul do aktuálnej verzie aplikácie.

6.5.6 Výstupy metodiky

1. Zdrojový kód nového modulu
2. Vetva modulu v repozitári aplikácie
3. Pull request v repozitári aplikácie
4. Code review vykonaných zmien
5. Build status aktuálnej verzie zdrojového kódu

6.5.7 Nadväzujúce metodiky

1. Metodika kontroly kvality kódu
2. Metodika testovania a nasadenia

6.6 Metodika testovania a nasadenia

Metodika testovania je rozvrhnutá na 2 časti: manuálne testovanie podľa porovnania výstupov s očakávanými výsledkami a automatické unit testy. Unit testy je potrebné vykonávať pre backendovú časť naprogramovanú v jazyku PHP a frontendovú časť v jazyku Typescript.

6.6.1 Identifikácia a opis rol

Role: tím, vedúci tímu, manažér testovania, vlastník

Vlastník - zadávateľ daného projektu

Tím - viacero zamestnancov, ktorí spoločne pracujú na jednom projekte v danom čase

6.6.2 Identifikácia a opis (manažérskych) procesov

Vstupy – implementované moduly programu

Výstupy - otestované moduly programu, opravené nekorektné časti zdrojového kódu , nasadený produkt

Manuálne testy na základe požiadaviek

Manuálne testy vykonáva člen tímu pre svoju úlohu po implementácií sériu testov, ktoré si definuje v závislosti od špecifikácie úlohy, a na základe výsledkov zhodnotí výsledky testovania.

Unit testy

Pre PHP jazyk používame framework PHPUnit, ktorý nám umožní používať jeho funkcionalitu pre realizáciu unit testov. Unit testy pre PHP triedy je potrebné napísať v priečinku ***tests,*** kde budú vytvorené unit test PHP triedy pre jednotlivé triedy, v ktorých chceme testovať dané metódy. Hlavná trieda je v súbore ***TestCase.php,*** ktorá dedí metódy frameworku PHPUnit pre porovnávanie výstupov pri porovnávaní dunkcionality metód. Teda unit testy, ktoré je potrebné napísať, dedia od triedy v súbore TestCase.php. Taktiež v súbore ***phpunit.xml*** sú špecifikované vlastnosti PHPUnit testovania (napr. To, že testovacie súbory sú v priečinku /tests).

Pre AngularJS a Typescript používame testovacie nástroje Karma a Protractor. Tieto nástroje nám umožňujú vykonávať unit testy nad novo implementovanými modulmy v AngularJS.

Testy uložené v priečinku *tests* sú pri pull requeste vykonané nástrojom Bamboo pomocou nakonfigurovaných príkazov pre PHP unit testy. Následne je o výsledku testov oboznámený člen tímu, ktorý robí prehliadku kódu s informáciou o tom, či bol unit test úspešný alebo nie, spolu s informáciami, pre ktoré testovacie metódy prebehol/neprebehol.

Nasadenie produktu do prevádzky

Nástroj Bamboo zabezpečuje okrem unit testingu aj nasadenie projektu do prevádzky na školský server. Nástroj skontroluje či prebehla kontrola kvality kódu a unit testy a následne prebehne build projektu pomocou príkazov zadefinovaných v nástroji Bamboo. Projekt je po builde nástrojom Bamboo nasadený do prevádzky na školskom serveri.

6.7 Metodika dokumentácie

V našom projekte sme si vybrali metodiku dokumentácie, ktorá je potrebná pre spisovanie spoločných stretnutí, tvorby retrospektívy či tvorby výstupných dokumentov o našej práci v projekte.

Metodika dokumentácie je previazaná so všetkými ostatnými metodikami, pretože sa dokumentuje všetko od komunikácie až po tvorbu zdrojového kódu.

6.7.1 Identifikácia a opis rol

Role: tím, vedúci tímu, vedúci dokumentácie, vlastník

Vlastník - zadávateľ daného projektu

Tím - viacero zamestnancov, ktorí spoločne pracujú na jednom projekte v danom čase

Vedúci dokumentácie - člen tímu, ktorý má na starosti dokumentáciu v rámci projektu, ktorý dozerá na to, či všetky časti sú splnené alebo nie

6.7.2 Identifikácia a opis (manažérskych) procesov

Vstupy – diskusia v tíme, rozdelenie úloh, tvorba pripomienok, tvorba softvéru

Výstupy - vytvorené zápisnice, retrospektívy, metodiky, dokument o riadení a inžinierske dielo

Tvorba zápisníc

Zápisnice sa vytvárali každý týždeň na spoločnom stretnutí. Vždy sme si zvolili jedného člena tímu, ktorý mal túto úlohu na starosti. Písala sa podľa presne stanovenej štruktúry, ktorú je možné vidieť na obrázku č.1 . V zápisnici sú stanovené úlohy, ktoré sa rieši daný šprint, a taktiež sú k ním priradení členovia tímu, ktorí ju majú na starosti a podieľajú sa na jej vykonaní.

next dimension team

Tímový projekt 2016/2017

Zápis zo stretnutia č. X

Vedúci tímu: Ing. Polášek Ivan, PhD.	**Dátum: 27.9.2016 11:00** **Čas: 11:00 – 14:00** **Miesto: 3D Lab (FIIT STU)** **Vypracoval: Bc. Lukáš Skala**
Prítomní: Bc. Gáspár Matúš **Bc. Jesze Ľubomír** **Bc. Minárik Jakub** **Bc. Skala Lukáš** **Bc. Víťaz Matej** **Bc. Zajac Peter** **Bc. Žilka Dominik**	

Téma stretnutia:

Nejaka tema.

Predchádzajúce úlohy:

#	Úloha	Zodpovedné osoby	Status
1			✓✗

Obsah Stretnutia:

1. *Niečo*
2. *Niečo*
3. *Ešte niečo*

Obrázok č. 1 - Šablóna zápisnice

Tvorba retrospektív

Po každom dokončenom dvojtýždňovom šprinte sme si vytvorili retrospektívu šprintu. Zistili sme čo sa v danom šprinte podarilo, resp. nepodarilo, čo treba zlepšiť , či už v rámci komunikácie v tíme alebo vykonávaní zadaných úloh. Pri tvorbe retrospektívy je vstupným procesom diskusia v tíme medzi všetkými členmi na dané úlohy v rámci šprintu a dodržiavania stanovených metodík. Každý z tímu sa vyjadrí ako sa mu šprint podaril a čo by bolo možné ešte zlepšiť, resp. Upraviť. Výstupom je dokument retrospektívy daného šprintu a zistené nedostatky. Retrospektíva je dôležitá z toho hľadiska, aby sme nevykonávali znova tie isté chyby, a taktiež, aby sa zlepšila komunikácia v tíme.

Tvorba exportov

Keďže sme používal nástroj JIRA pre prideľovanie úloh a identifikáciu šprintov, tak sme exporty jednoducho vytvárali v tomto nástroji. Dokument exportu úloh obsahoval názov "user story", ktorá sa rieši, názov "tasku", stav úlohy (open, resolved, in-progress...), projekt do ktorého patrí danú úloha, typ úlohy, prioritu, meno toho, kto úlohu zadal a meno toho kto je zodpovedný za danú úlohu, určený "estimate" úlohy a nakoniec stav splnenia úlohy.

Dokumenty v kontrolných bodoch

Dokumenty, ktoré sú dôležité v kontrolných bodoch sú:

- Dokumentácia k riadeniu projektu
- Dokumentácia k inžinierskemu dielu

V rámci týchto dokumentov je dôležité mať rozpracované aj metodiky, ktoré v tíme používame.

Oba dokumenty majú preddefinovanú štruktúru:

- Názov vysokej školy a fakulta
- Adresa
- Názov dokumentu
- Mená autorov
- Meno vedúceho projektu

- Akademický rok

Dokumentácia k riadeniu projektu má tieto hlavné časti:

- Úvod, role členov tímu a podiel práce
- Aplikácie manažmentov
- Sumarizácie šprintov
- Používané metodiky
- Globálna retrospektíva
- Motivačný dokument
- Export evidencie úloh
- Prílohy - zápisnice, retrospektívy

Dokumentácia k inžinierskemu dielu má tieto hlavné časti:

- Úvod
- Globálne ciele
- Moduly systému - Analýza, návrh, implementácia, testovanie
- Inštalačná príručka
- Technická dokumentácia

Dokumenty sa musia robiť priebežne a stále sa iba nabaľujú o novo vzniknuté problematiky. Big picture dokumentu sa už veľmi meniť nebude, kdežte ostatné časti, ako napr. Inštalačná príručka, technická dokumentácia alebo sumarizácia šprintov áno.

Zdieľanie dokumentov

Všetky dokumenty sú prístupné na našom webovom sídle pre budúce potreby. Zápisnice sa vytvárali v Microsoft Worde. Nie je potrebné viacerých účastníkov zainteresovať do tohto procesu, pretože to vykonával vždy jeden člen tímu po vzájomnej dohode. Zapisoval si všetky pripomienky na stretnutiach, dohodnuté úlohy a nakoniec to spísal do jedného dokumentu, ktorý uverejnil na našom webovom sídle a odoslal vedúcemu.

Retrospektívy boli vytvárané spoločne na stretnutiach v rámci diskusie a spisoval to znova jeden člen tímu po dohode.

Dokumenty v kontrolných bodoch sa vytvárali všetkými účastníkmi tímu v Google Docs alebo OneDrive, kde bolo možné vidieť online zmeny v dokumente.

6.8 Metodika komunikácie

Metodika komunikácie (ďalej MK) sa zaoberá verbálnou ale aj neverbálnou komunikáciou medzi členmi tímu, ale aj medzi tímom ako celkom a inými účastníkmi procesu. Obsahuje kroky, ktoré je nutné dodržiavať, aby častá komunikácia medzi členmi tímu prebiehala nielen hladko, ale hlavne korektne, zrozumiteľne a neskôr dohľadateľne a prehľadne. To isté musí platiť aj pre komunikáciu mimo daného tímu.

MK sa riadia zamestnanci , ktorý ako jeden celok (tím) pracujú na jednom, danom projekte.

Na MK nadväzuje Metodika plánovania, Metodika kontroly kvality kódu a Metodika vývoja a integrácie.

Pojmy:

MK – Metodika komunikácie

Požiadavka - čokoľvek čo môže spustiť proces (otázka, podnet, príkaz)

Projekt - úloha, ktorú stanoví vlastník danému tímu a tím sa ju postupnými krokmi za pomoci metodík snaží vyriešiť

6.8.1 Identifikácia a opis rol

Zamestnanec – osoba vo firme, ktorá je platená za výkon svojej práce

Tím - viacero zamestnancov, ktorí spoločne pracujú na jednom projekte v danom čase

Vlastník - zadávateľ daného projektu, osoba s ktorou tím konzultuje, či správne chápe/rieši dané úlohy pre splnenie cieľa projektu

Vedúci tímu - osoba v tíme, ktorá spája všetkých ostatných a dbá na to, aby všetci plnili svoje úlohy svedomite. Je to osoba s najväčšími znalosťami o daných technológiách a procesoch a svojím know-how napomáha ostatným členom tímu.

Role - tím, vedúci tímu, vedúci komunikácie, vlastník,

6.8.2 Identifikácia a opis (manžérskych) procesov

6.8.2.1 Proces komunikácie tímu s inou osobou

Vstupy - požiadavka na tím, požiadavka od tímu

Výstupy - vyriešená požiadavka, nevyriešená požiadavka

V priebehu riešenia projektu môžu nastať skutočnosti, kedy sa tím nemôže s danou osobou stretnúť osobne, ale musia svoje problémy odkomunikovať elektronicky. Podľa MK využívame na tieto činnosti službu od spoločnosti Google – GoogleGroups. Každý zamestnanec má konto na Gmail-y. Po vytvorení tímu na riešenie projektu sa tomuto tímu vytvorí aj GoogleGroup účet. To zebezpečí, že akákoľvek správa príde celému tímu, celý tím na ňu môže reagovať a taktiež každý člen tímu vidí, ak niekto z tímu na správu reaguje.

Do procesu teda vstupuje požiadavka, ktorá sa pomocou e-mailového klienta zobrazí všetkým členom tímu. Prvotne ju však spracováva osoba zodpovedná za komunikáciu v tíme. Ak je to triviálny problém, ktorý nevyžaduje vyššie technické zdatnosti ako má táto osoba, tak jednoducho len v mene celého tímu odpovie na požiadavku a tá sa považuje za vyriešenú. Ak tento problém vyžaduje väčšie technické znalosti, požiada osoba zodpovedná za komunikáciu v tíme vedúceho tímu, aby priradil túto požiadavku na spracovanie jednému z členov tímu. Vedúci tímu túto osobu informuje o pridelení úlohy. Následne osoba na požiadavku odpovie prostredníctvom GoogleGroups, takže všetci vidia ako je problém riešený. Osoba zodpovedná za komunikáciu ďalej odsleduje, či sa problém podarilo vyriešiť a či si každý splnil v tomto procese svoje náležitosti. Ak áno požiadavka sa považuje za vyriešenú, ak nie kontaktuje zodpovednú osobu pridelenú na riešenie problému a požaduje urýchlené riešenie situácie.

6.8.2.2 Proces komunikácie v tíme pri riešení úloh

Niekedy sa môže stať, že na riešenie nejakej požiadavky je nutné prepojiť prácu viacerých členov tímu. Nie vždy je však možné, aby sa osobne stretli. Je však nutné aby boli tieto konverzácie ľahko dohľadateľné a prehľadne pomenované. V tímoch sa teda okrem e-mailových klientov využíva služba Hipchat. Tá obsahuje priamy chat (od jedného člena tímu druhému), ale aj možnosť vytvárať chatovacie skupiny. Pri riešení taskov z backlogu MK presne určuje postup akým majú medzi sebou členovia tímu komunikovať. Zodpovedná osoba, ktorá dostane task na starosti vytvorí na Hipchate nový room. Jeho názov presne kopíruje názov daného tasku. Následne do tohto roomu pridá ostatných členov, ktorým bol task pridelený a ak už na tomto tasku nepracuje, tak aj osobu zodpovednú za komunikáciu v tíme. Po skončení práce na tasku môžu všetci tí, ktorí na ňom pracovali (vrátane osoby zodpovednej za komunikáciu) jednoducho túto konverzáciu vyhľadať a vyčítať z nej čo potrebujú.

6.8.2.3 Proces verbálnej komunikácie na stretnutiach

Aby mali osobné stretnutia tímu čo najväčší prínos, musia podliehať pravidlám. Preto má každé stretnutie jasne stanovenú metodiku a postup. Najprv prednesie vedúci tímu o čom sa bude diskutovať. Potom postupne vyzve každého aby vyjadril svoj názor. Je nutné aby všetko riadil vedúci tímu a prideľoval slovo. V danom momente rozpráva len jedna osoba a neskáču si do rečí. Tento prístup vnáša do diskusie viac poriadku. Stretnutia tak majú vecnejší a rýchlejší spád. Jediná osoba, ktorá môže niekoho prerušiť je vedúci tímu. Po vypočutí si návrhov a nápadov vedúci diskusiu uzatvára hlasovaním, či sa väčšina zhodla na danom výsledku. Ak áno, pokračuje sa na ďalší bod diskusie. Tento proces je úzko spätý s priraďovaním taskov z backlogu, alebo planning pokerom.

7 Globálna retrospektíva ZS/LS

7.1.1 Čo išlo podľa našich predstáv

- Nastavili sa potrebné nástroje pre komunikáciu a manažment úloh v tíme
- Analýza testovacích scenárov minuloročného tímového projektu
- Nastavili sa potrebné veci na GitHube
- Nasadenie webového sídla na GitHub pages a školský server
- Bol vytvorený jednoduchý prototyp sekvenčného diagramu vo WebGL za účelom vyhodnotenie použiteľnosti danej technológie
- Bolo stanovené vývojové prostredie pre backend a databázu – PHP framework Laravel, PosgreSQL databáza

7.1.2 Čo nešlo podľa našich predstáv

- Problémy pri určení vývojového prostredia pre frontend (WebGL alebo CSS3D)
- Zdržanie vo vývoji v dôsledku zmeny prostredia z WebGL do CSS3D transformácií
- Problémy s nahodením webového sídla na školský server

7.1.3 Čo plánujeme zlepšiť

- Komunikáciu v tíme – vytváranie miestností na Hipchate k daným úlohám
- Finálne definovanie vývojového prostredia frontendu – knižnica pre elementy sekvenčného diagramu /vytvorenie vlastných elementov pre diagram
- Vytvorenie dátového modelu zo metamodelu, ktorý je diagram tried
- Lepšie plnenie úloh – skôr ako tesne pred deadlinom a koncom šprintu

8 Motivačný dokument

Spoločný e-mail: team.winners.15@gmail.com

Číslo tímu: 15

Počet členov: 7

Členovia:

- Dominik Žilka
- Matej Víťaz
- Peter Zajac
- Ľubomír Jesze
- Lukáš Skala
- Matúš Gáspár
- Jakub Minárik

Minulá spolupráca:

- Na bakalárskom štúdiu sme v tímoch (dvojice, trojice aj štvorice) pracovali a pomáhali si na viacerých projektoch:
    - Princípy informačných systémov
    - Princípy softvérového inžinierstva
    - Interakcia človeka s počítačom
    - Princípy počítačovej grafiky a spracovania obrazu

Naše silné stránky:

- Webové technológie
    - **Frontend:** HTML5, CSS3, JavaScript, jQuery, AngularJS, EmberJS
    - **Backend:** PHP (Laravel, CodeIgniter, Zend), J2E
    - **Databázy:** Oracle, MySQL, PostgresSQL, MSSQL, Redis, MongoDB, Elastic Search
- Návrh softvéru: Školské zadania
- Práca v tíme: skúsenosti s verziovacími systémami (Git).
- Poznáme sa, navzájom sa dopĺňame a máme chuť sa učiť nové technológie.

Osobné skúsenosti:

- Matej Víťaz
    - Autor bakalárskej práce "Podpora výučby jazyka SQL" - https://query.fiit.stuba.sk/ v jazyku PHP (Laravel)
    - Expert na HTML, CSS, JavaScript, PHP
    - Ovláda aj: Java, C, SQL
    - Vytvoril desiatky webových stránok na mieru (Backend aj Frontend), napr.:
        - http://www.ak-autopotahy.sk/produkty
        - http://www.esbm.cz/
        - http://www.businessinstitut.cz/
        - http://www.houpi.cz/
- Dominik Žilka
    - Pokročilý v jazykoch C, Java
    - Základy PHP, JavaScript, Bash2, SQL
    - Bakalárska práca v jazyku C, využívanie paralelných technológii na spočítavanie reťazcov DNA
    - Silné stránky - komunikatívnosť, práca v tíme, kreativita, výdrž, zodpovednosť
    - Praktické skúsenosti - práca v IBM (Java, JavaScript, PHP), práca v ISDD+ (PHP - Zend2 základy)
- Peter Zajac
    - Pokročilý v jazyku Java a SQL, učí sa webové technológie
    - Bakalárska práca vyvíjaná v jazyku PHP
    - Praktické skúsenosti s jazykom Java v IBM
- Ľubomír Jesze
    - Autor bakalárskej práce "Internetové rozhranie pre multimediálny systém do vlakov" v jazyku PHP (Laravel), ktorá sa zaoberá správou multimediálneho obsahu prehrávaného vo vlakoch.
    - Prispievanie do jadra frameworku Laravel a iných open source projektov
    - Pracuje v oblasti tvorby J2E portálov
- Lukáš Skala
    - V bakalárskej práci vytvoril automatizovaný nástroj pre obohacovanie zdrojového kódu webových stránok o poznámky
    - Pokročilý v jazykoch java, C#, HTML, CSS, SQL
    - Pracuje vo firme Plaut kde sa zdokonaľuje v J2E technológiách
- Matúš Gáspár
    - Bakalárska práca: Hľadanie vzorov v údajoch z počítačovej myši (jazyk R) - rád sa učí nové veci

    - Ovláda Javu, HTML, CSS
    - Z databáz: PostgreSQL
    - Pracuje vo firme Soitron - support - administrácia účtov, zodpovednosť, komunikácia s významnými zákazníkmi
- Jakub Minárik
    - Databázový expert: MS SQL, MySQL, Oracle, PostgresSQL
    - Bakalárska práca : Využitie obohatenej reality pre navigáciu na mape (objective-c)
    - Pokročilý v jazykoch Java, C#, HTML5, CSS3

Priorita č. 1 - Tvorba vzdelávacích simulácií [EduSim]

Táto téma je naša prvá preferencia, pretože v oblasti webových aplikácií máme bohaté skúsenosti a radi sa učíme nové technológie a postupy v oblasti webu. Zaujímame sa najmä o tvorbu interaktívnych používateľských rozhraní pomocou webových technológií (AngularJS, EmberJS, KnockoutJS, jQuery, ReactJS, MeteorJS ...). Konkrétnou skúsenosťou v tejto oblasti je napr. bakalárska práca člena tímu "Podpora výučby jazyka SQL" alebo konfigurátor autopoťahov (http://www.ak-autopotahy.sk/produkty). Taktiež bakalárska práca ďalšieho člena tímu zameraná na správu multimediálneho obsahu disponovala používateľsky prívetivým rozhraním. Väčšina členov nášho tímu má zapísané odporúčané predmety Pokročilé databázové technológie a Architektúry informačných systémov. Ako zamestnanci firiem máme taktiež skúsenosti z praxe s prácou v tíme na projektoch, komunikáciou so zákazníkmi a riešením problémov.

Priorita č. 2 - Extrakcia dát z webu [WebExtraction]

O túto tému sa zaujímame, pretože v oblasti webových aplikácií máme bohaté skúsenosti a radi sa učíme nové technológie a postupy v oblasti webu. Väčšina členov nášho tímu taktiež absolvovala predmet "Webové publikovanie" v bakalárskom štúdiu, kde sme rozšírili naše znalosti v oblasti XML, HTML, CSS, JavaScript, XPath, XML Schema. Taktiež v inžinierskom štúdiu majú niektorí členovia zapísané odporúčané predmety : Vyhľadávanie informácií a Objavovanie znalostí. Jeden člen nášho tímu je autorom bakalárskej práce "Poznámkovanie webu", v ktorej pracoval s extrakciou údajov z webu.

V súvislosti s riešením problémov sa radi učíme nové moderné webové technológie potrebné pre vypracovanie zadaných úloh. Ako zamestnanci firiem máme taktiež skúsenosti z praxe s prácou v tíme na projektoch, komunikáciou so zákazníkmi a riešením problémov.

Príloha A

17. Tvorba vzdelávacích simulácií [EduSim]
14. Extrakcia dát z webu [WebExtraction]
16. Inteligentný sklad [SmartStore]
3. Nový návrh systému pre MOD [Future MOD]
20. Navigácia v budove [VirtualFEI]
9. Zber a vyhodnocovanie požiadaviek [Story Teller]
5. Pomôcky pre aktívnych programátorov [CodeCrutches]
12. Manažment zdravotného stavu pacienta prostredníctvom monitoringu emócií [eMotion]
7. 3D UML, improved version [3D UML]
11. Prepájanie dát o vývoji softvéru [TRACKS]
4. Simulácia správania dronov v roji [DronSim V2]
8. Automatická podpora moderácie diskusií [ModerateIT]
10. Monitorovanie bezpečnostných udalostí [SecMon]
6. Servisný modul pre stratosférický balón [StratosFIIT]
15. Kam na obed 2.0 [FoodCourt]
2. Rekonštrukcia 3D scény [3D-Recon V2]
19. Aplikácia deterministického ethernetu pre distribuovaný vnorený systém [Slovak TTTech]
18. Monitorovanie a vyhodnocovanie fyziologických procesov človeka [StresMonitor]
21. 3D simulovaný robotický futbal [3D futbal]
1. Vizualizácia informácií v obohatenej realite [AugReality]
13. Vyhľadávanie so sémantikou [DeepSearch]

9 Export evidencie úloh

9.1 Šprint 1

9.1.1 Ako pouzivatel chcem ulozit sekvencny diagram, aby som sa k nemu mohol neskor vratit.	
Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Matej Víťaz
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	Key	Summary	Type	Status	Assignee
	UML-35	Vytvoriť modely v Laraveli	Sub-task	Resolved	Dominik Žilka
	UML-36	Vytvoriť migrácie v Laraveli	Sub-task	Resolved	Lukáš Skala
	UML-37	Vytvoriť seedy v Laraveli	Sub-task	Resolved	Matej Víťaz
	UML-42	Vypracovať fyzický dátový model databázy	Sub-task	Resolved	Matúš Gáspár
	UML-43	Vytvoriť databázu a pripojiť ju k bac...	Sub-task	Resolved	Jakub Minarik
	UML-44	Vytvoriť JSON API na backende	Sub-task	Resolved	Matej Víťaz
Sprint:	UML Sprint 1				

9.1.2 Ako tím potrebujeme softvérové nástroje pre manažment v tíme a podporné vývojové nástroje pre lepšiu organizáciu a komunikáciu v rámci tímu.	
Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**
Reporter:	Lukáš Skala	**Assignee:**	Ľubomír Jesze
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	Key	Summary	Type	Status	Assignee
	UML-16	Prezentácia o používaní vývojových ná...	Sub-task	Resolved	Dominik Žilka
	UML-17	Stanoviť coding rules a zvoliť potreb...	Sub-task	Resolved	Ľubomír Jesze
	UML-18	Nakonfigurovať Jiru pre potreby tímu ...	Sub-task	Resolved	Lukáš Skala
	UML-19	Inštalácia vývojových nástrojov na se...	Sub-task	Resolved	Ľubomír Jesze
	UML-46	Nakonfugurovať TSLint	Sub-task	Resolved	Matej Víťaz
Sprint:	UML Sprint 1				

9.2 Šprint 2

9.2.1 Ako tím chceme integrovať knižnicu Go.JS do projektu

Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**
Reporter:	Lukáš Skala	**Assignee:**	Lukáš Skala
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	Key	Summary	Type	Status	Assignee
	UML-61	Analyzovať možnosti integrovania Go.j...	Sub-task	Resolved	Matej Víťaz
	UML-63	Vytvoriť example v Go.js	Sub-task	Resolved	Lukáš Skala
	UML-64	Integrovať lifeline z	Sub-task	Resolved	Lukáš Skala

		Go.js do projektu			
	UML-65	Integrovať message z Go.js do projektu	Sub-task	Resolved	Lukáš Skala
	UML-66	Integrovať šablónu sekvenčného diagra...	Sub-task	Resolved	Lukáš Skala
Sprint:	UML Sprint 2				

9.2.2 Ako tím chceme integrovať do projektu JSONAPI	
Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**
Reporter:	Lukáš Skala	**Assignee:**	Ľubomír Jesze
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-57	Analyzovať existujúce knižnice JSON API	Sub-task	Resolved	Ľubomír Jesze
	UML-58	Integrovať vybranú knižnicu JSON API ...	Sub-task	Resolved	Ľubomír Jesze
	UML-59	Nadviazať komunikáciu prostredníctvom...	Sub-task	Resolved	Matej Víťaz
	UML-67	Vytvoriť JSON API modely	Sub-task	Resolved	Peter Zajac
Sprint:	UML Sprint 2				

9.2.3 Ako tím chceme integrovať do projektu Angular.js na zjednodušenie vývoja	
Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**
Reporter:	Lukáš Skala	**Assignee:**	Matej Víťaz
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-52	Nainštalovať Angular.js	Sub-task	Resolved	Matej Víťaz
	UML-53	Nakonfigurovať kompilačné skripty	Sub-task	Resolved	Ľubomír Jesze
	UML-54	Nakonfirurovať Bamboo pre zmenený bui...	Sub-task	Resolved	Peter Zajac
Sprint:	UML Sprint 2				

9.2.4 Ako tím chceme MVC architektúru pre projekt, aby sme mohli dekomponovať aplikáciu.	
Status:	Resolved
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**

Reporter:	Lukáš Skala	**Assignee:**	Dominik Žilka
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-55	Vytvoriť adresárovú štruktúru	Sub-task	Resolved	Lukáš Skala
	UML-56	Vytvoriť angular komponenty	Sub-task	Resolved	Matúš Gáspár
Sprint:	UML Sprint 2				

9.3 Šprint 3

9.3.1 Ako tím chceme zverejniť výstupnú aplikáciu na produkčnom serveri	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Technical Story**	**Priority:**	**Medium**
Reporter:	Matej Víťaz	**Assignee:**	Ľubomír Jesze
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-90	Zverejniť výstupnú aplikáciu na produ...	Sub-task	Open	Jakub Minarik
Sprint:	UML Sprint 3				

9.3.2 Ako používateľ chcem načítať predtým uložený sekvenčný diagram	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Matej Víťaz	**Assignee:**	Ľubomír Jesze
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:

Key	Summary	Type	Status	Assignee
UML-72	Vytvoriť JSON API transformery	Sub-task	Open	Ľubomír Jesze
UML-78	Vytvoriť príklad diagramu v databáze ...	Sub-task	Open	Matej Víťaz
UML-82	Upraviť migrácie podľa zmien od vedúceho	Sub-task	Open	Dominik Žilka
UML-84	Upraviť modely na backende podľa zmie...	Sub-task	Open	Matúš Gáspár
UML-87	Upraviť modely JSON API na fronende	Sub-task	Open	Peter Zajac
UML-88	Vytvoriť databázové factories	Sub-task	Open	Peter Zajac

Sprint:	UML Sprint 3

9.3.3 Ako pouzivatel chcem upravit interakciu.	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Peter Zajac
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-85	Upraviť správu v diagrame	Sub-task	Open	Peter Zajac
	UML-86	Zaznamenať zmenu správy na backende	Sub-task	Open	Peter Zajac
Sprint:	UML Sprint 3				

9.3.4 Ako pouzivatel chcem vymazat interakciu.	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Jakub Minarik
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	Key	Summary	Type	Status	Assignee
	UML-80	Pridať button na zmazanie správy	Sub-task	Open	Jakub Minarik
	UML-81	Zmazať správu z diagramu	Sub-task	Open	Jakub Minarik
	UML-83	Zaznamenať zmazanie správy na backende	Sub-task	Open	Jakub Minarik
Sprint:	UML Sprint 3				

9.3.5 Ako pouzivatel chcem pridat interakciu.	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Matúš Gáspár
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	Key	Summary	Type	Status	Assignee
	UML-76	Pridať button na pridanie správy	Sub-task	Open	Matúš Gáspár
	UML-77	Pridať správu do diagramu	Sub-task	Open	Matúš Gáspár
	UML-79	Zaznamenať pridanie	Sub-task	Open	Matúš Gáspár

		správy na backende			
Sprint:	UML Sprint 3				

9.3.6 Ako pouzivatel chcem zmazat lifeline	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Lukáš Skala
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-73	Pridať button na zmazanie lifeliny	Sub-task	Open	Matej Víťaz
	UML-74	Zmazať lifelinu z diagramu	Sub-task	Open	Lukáš Skala
	UML-75	Zaznamenať odobranie lifeliny na back...	Sub-task	Open	Lukáš Skala
Sprint:	UML Sprint 3				

9.3.7 Ako pouzivatel chcem pridat lifeline.	
Status:	Open
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Peter Zajac	**Assignee:**	Dominik Žilka
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified
Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-69	Pridať button na pridanie lifeliny	Sub-task	Open	Dominik Žilka
	UML-70	Pridať lifelinu do diagramu	Sub-task	Open	Dominik Žilka
	UML-71	Zaznamenať novú lifelinu na backende	Sub-task	Open	Dominik Žilka
Sprint:	UML Sprint 3				

9.3.8 [Ako používateľ chcem prostredníctvom 3D UML aplikácie vytvárať plátna sekvenčného diagramu, pretože plátna tvoria základ každého 3D UML digramu	
Status:	In Progress
Project:	3D UML
Component/s:	None
Affects Version/s:	None
Fix Version/s:	None

Type:	**Story**	**Priority:**	**Medium**
Reporter:	Lukáš Skala	**Assignee:**	Matej Víťaz
Resolution:	Unresolved	**Votes:**	0
Labels:	None		
Σ Remaining Estimate:	Not Specified	**Remaining Estimate:**	Not Specified

Σ Time Spent:	Not Specified	**Time Spent:**	Not Specified
Σ Original Estimate:	Not Specified	**Original Estimate:**	Not Specified

Sub-Tasks:	**Key**	**Summary**	**Type**	**Status**	**Assignee**
	UML-11	Urobiť zobrazenie viacerých plátien	Sub-task	Reopened	Matej Víťaz
	UML-12	Urobiť vytvorenie plátna	Sub-task	Reopened	Ľubomír Jesze
	UML-13	Urobiť menu s buttonom ktorý vytvorí ...	Sub-task	Reopened	Lukáš Skala
Sprint:	UML Sprint 3				

9.4 Príloha A - Zápisnice

Zápisnice sú dostupné na webovom sídle nášho tímu v sekcii „Dokumenty“: http://labss2.fiit.stuba.sk/TeamProject/2016/team15is-si/ .