Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Projektová dokumentácia - riadenie

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

Obsah

1 Role členov tímu a podiel práce **3**
1.1 Role členov tímu 3
1.2 Pridelené zodpovednosti členov tímu 3
1.3 Podiel práce 5

2 Aplikácie manažmentov **5**
2.1 Manažment dokumentácie 5
2.2 Manažment komunikácie 5
2.3 Manažment plánovania 6
2.4 Manažment verziovania 6
2.5 Manažment testovania 6
2.6 Manažment kvality písania zdrojového kódu 6

3 Sumarizácie šprintov **7**
3.1 Predšprintová príprava 7
3.2 Šprint 1 - Layer 7
3.3 Šprint 2 - Lifeline 9
3.4 Šprint 3 - Message 10
3.5 Šprint 4 - Create 11
3.6 Šprint 5 - State 12
3.7 Šprint 6 - State-Machine 13
3.8 Šprint 7 - Continous-Integration 14
3.9 Šprint 8 - Test-Cases 15
3.10 Šprint 9 - Pop-Up 16
3.11 Šprint 10 - Animation 16

4 Používane metodiky **17**
4.1 Metodika plánovania 17
4.2 Metodika konvencií písania zdrojového kódu 17
4.3 Metodika testovania 17
4.4 Metodika prehliadky zdrojového kódu - code review 18
4.5 Metodika pre mapovanie služieb 18
4.6 Metodika verziovania zdrojového kódu 18
4.7 Metodika písania dokumentácie 18

5 Globálna retrospektíva **18**

Úvod

Tento dokument predstavuje dokumentáciu riadenia projektu, ktorý je vytváraný v rámci predmetu Tímový projekt v akademickom roku 2017/2018. Dokument zahŕňa roly členov tímu, podiel ich práce na dokumentácii projektu, sumarizácie šprintov, použitých metodík a globálnu retrospektívu pre zimný semester.

Náš tím sa volá *Bystander Effect* a téma s ktorou pracujeme je 3D-UML. Cieľom projektu je vytvorenie webového nástroja na vytváranie 3D sekvenčných diagramov, ktorý dovolí používateľovi vytvárať sekvenčný diagram s viacerými vrstvami. Interakcia s diagramom bude podobná interakcii v známych komerčných nástrojoch ako napr.: Enterprise Architect.

Prvá kapitola obsahuje prehľad rolí jednotlivých členov tímu a ich zodpovednosti na projekte. Takisto tu je tabuľka zobrazujúca podiel práce na dokumentácií riadenia projektu.

V druhej kapitole je popísaná aplikácia jednotlivých manažmentov v projekte. V tejto časti je opis činností potrebných pre riadenie projektu.

Tretia kapitola dokumentu slúži ako sumarizáciu jednotlivých šprintov. V sekcií šprintu sú vidieť úlohy ktoré boli, resp. neboli ukončené a burndown grafy.

Štvrtá kapitola dokumentu obsahuje zoznam a stručný opis metodík, ktorými sme sa riadili pri práci na projekte.

V piatej kapitole dokumentu je uvedená globálna retrospektíva pre zimný semester. Táto časť zahŕňa zoznam činností, ktoré boli vyhodnotené za pozitívne, zoznam činností, ktorým by sme sa mali v budúcnosti vyhnúť.

1 Role členov tímu a podiel práce

1.1 Role členov tímu

Meno člena tímu	Zodpovednosť v tíme
Štefan Motko	Projektový manažér
Ronald Demeter	Hlavný vývojár
Marcel Furucz	Scrum a komunikácia
Patrik Sčensný	Testovanie a kvalita
Peter Psota	Grafický odborník
Martin Gembec	Administrácia databáz a dokumentácia
Martin Dieška	Vývoj serverovej strany

Tabuľka 1: Role členov tímu

1.2 Pridelené zodpovednosti členov tímu

Štefan Motko

- Prezentácia projektu
- Doménový expert
- Návrh systémových komponentov

Ronald Demeter

- Dohliadať na vývojový tím
- Rozhodovať o zdrojoch
- Dodávať kód, ktorý je dobre otestovaný a bezchybný
- Napísať čistý, štruktúrovaný a dobre zdokumentovaný kód

Marcel Furucz

- Udržiavať Scrum a Šprinty
- Spravovať tímovú komunikáciu

Patrik Sčensný

- Sledovať písanie testov a testovania
- Dohliadať nad kvalitou systému
- Vytvárať, štylizovať a spravovať tímovú webstránku
- Spravovať tímový server

Peter Psota

- Spravovať vizualizácie diagramov
- Spravovanie používateľského zážitku

Martin Gembec

- Vytváranie zápisníc zo stretnutí
- Vytváranie a spravovanie dokumentácie
- Správa databázy

Martin Dieška

- Zabezpečiť funkčnú serverovú stranu aplikácie.
- Poskytnúť front-endu všetky potrebné služby pre získanie a zapisovanie dát.
- Zabezpečiť komunikáciu medzi databázou a serverom a medzi serverom a klientom.

1.3 Podiel práce

Časť dokumentácie	Vypracovali
Úvod	Patrik Sčensný
Role členov tímu	relevantní členovia tímu
Aplikácie manažmentov	Ronald Demeter, Patrik Sčensný
Sumarizácie šprintov	Martin Gembec
Používané metodiky	relevantní členovia tímu
Globálne retrospektíva	Martin Dieška

Tabuľka 2: Podiel práce na dokumentácií riadenia projektu

2 Aplikácie manažmentov

2.1 Manažment dokumentácie

Každé tímové stretnutie vzniká zápisnica, ktorú píše každý týždeň zodpovedný člen tímu. Táto zápisnica obsahuje opísaný priebeh stretnutia a taktiež zoznam úloh, ktoré je potrebné spraviť. Na túto zápisnicu bola v úvode práce na tímovom projekte vytvorená šablóna aby každá mala jednotný tvar. Zápisnica sa publikuje každý týždeň aj na tímovú stránku, aby bola dostupná každému členovi tímu.

2.2 Manažment komunikácie

Komunikácia mimo stretnutí prebieha cez nástroje Slack a Trello. Slack je rozdelený na viaceré kanály(anglicky: channels), ktoré slúžia na konkretizovanie diskusie a dva automatizované kanály ktoré slúžia na logovanie informácií a udalostí z Trella a githubu. Trello umožňuje komentovať jednotlivé používateľské príbehy, reprezentované kartami(anglicky: cards), čo slúži na konkretizovanie diskusie.

Počas stretnutí je komunikácia voľná. Na konci šprintu sa vytvorí retrospektíva kde každý člen vyjadrí svoje pozitíva a negatíva, ktoré zaznamenal počas šprintu.

2.3 Manažment plánovania

Na manažovanie plánovania sa používa Trello. V Trelle máme zadefinovaných šesť stĺpcov:

- Epics - obsahuje príbehy, ktoré sa sťahujú na viaceré šprinty
- Backlog - obsahuje používateľské príbehy sťahujúce na šprint
- Development - obsahuje používateľské úlohy, na ktorých sa momentálne pracuje. Vo vnútri týchto príbehov sú definované úlohy, ktoré sa musia splniť aby príbeh bol splnený
- Testing - Príbehy, na ktorých prebiehajú prehliadky kódu a testy
- Pending - Príbehy, ktoré majú vytvorený pull request a čakajú na zlúčenie do dev konára
- Done - Dokončené používateľské príbehy

Tím pracuje v šprintoch dĺžky dvoch týždňov. Na začiatku šprintu zadefinujeme úlohy, ktoré by na konci šprintu mali byť hotové.

2.4 Manažment verziovania

Každý člen tímu je osobne zodpovedný za dodržiavanie pravidiel verziovania kódu, ktoré sú zdokumentované v Metodike verziovania.

2.5 Manažment testovania

Každý člen tíme je zodpovedný za požiadanie napísania testu. Člen tímu zodpovedný za písanie testu bude návrh testu konzultovať so žiadateľom návrhu testu. Testy sú rozdelené na backend a frontend. Testy sú písané v nástroji Mocha. Pre detaily si preštudujte metodiku testovania.

2.6 Manažment kvality písania zdrojového kódu

Každý člen tímu, ktorý píše zdrojový kód je zodpovedný za to, aby bol čitateľný, funkčný a dodržiaval metodiku písania zdrojového kódu.

Všetok napísaný kód podlieha prehliadke zdrojového kódu, ktorý sa vytvorí počas stretnutia. Na prehliadku zdrojového kódu dohliadajú dvaja alebo

viacerí členovia tímu. Postup je opísaný v metodike prehliadke zdrojového kódu.

3 Sumarizácie šprintov

3.1 Predšprintová príprava

Úlohou predšprintovej prípravy bolo definovanie technológií, ktoré by mohli byť využité v projekte, ako aj oboznámenie sa s technológiami použitými v minuloročnom riešení tohto projektu.

V prvom týždni boli rozdelené úlohy jednotlivým členom tímu a vytvorený prvotný návrh plagátu. Každý člen tímu dostal za úlohu preštudovať si SCRUM a prezrieť si dokumentácie starších projektov, webových stránok a zápisníc.

Témou stretnutia v druhom týždni bolo bližšie oboznámenie sa s projektom prezentáciou minuloročných zadaní. Prezentáciu viedol Štefan Motko. Následne boli uvedené možnosti, ako by sme zmenou niektorých technológií mohli zefektívniť prezentované riešenia. Nakoniec sme spojazdnili nástroj na vytváranie a manažovanie user stories a taskov (Trello), vytvorili sme repozitár v GitHube, formálne sme ukončili prvý šprint a priradili sme si prvé úlohy. Do ďalšieho stretnutia sme sa rozhodli preskúmať konektivitu medzi C# a MongoDB a vygenerovať JSONy pre účely testovania. Každý člen tímu dostal za úlohu naštudovať si odprezentovaný metamodel.

- ukončené:

- príprava prezentácie - Štefan Motko
- prezretie dokumentácii, stránok a zápisníc starších projektov - všetci
- príprava plagátu - Marcel Furucz
- preštudovanie scrumu - všetci
- rozdelenie rol medzi členov tímu - všetci

3.2 Šprint 1 - Layer

V prvom šprinte sme začali s počiatočnou implementáciou služieb a prvkov grafického rozhrania.

Prvý týždeň prvého šprintu bol venovaný prezentácií metamodelu. Prezentáciu viedol Štefan Motko spolu s vedúcim tímu. Po odprezentovaní prebehla diskusia o možnom prvotnom modeli JSONu. Následne sme zostavili počiatočnú schému JSONu. Dôležitou úlohou do ďalšieho týždňa sa stalo vytvorenie webovej stránky a jej nasadenie na server. Ešte počas stretnutia bola celým tímom vybraná vizuálna schéma stránky.

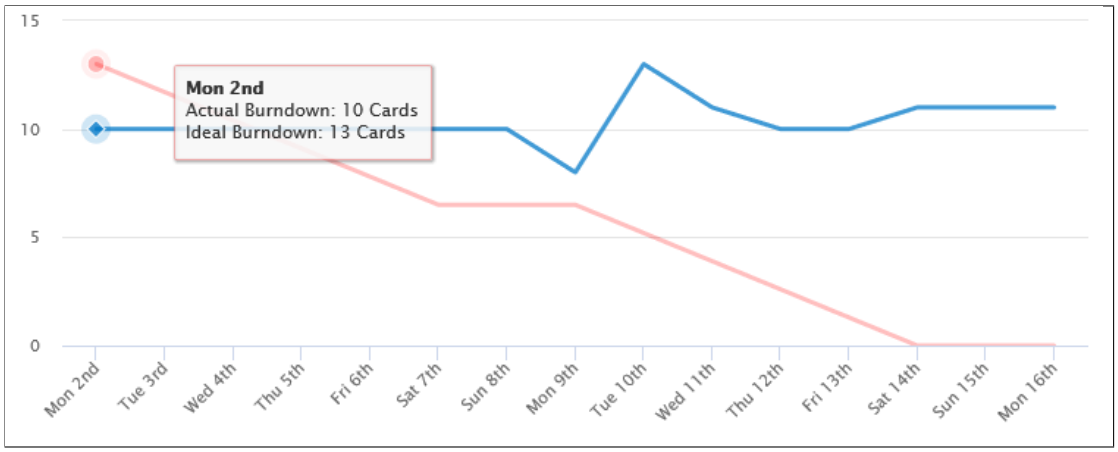
Na začiatku druhého stretnutia prvého šprintu boli odprezentované dosiahnuté výsledky vedúcemu tímu. Následne boli zosumarizované splnené a nesplnené úlohy a boli pomenované chyby, ako napr zlé pushovanie do GitHubu a pod., spolu s riešeniami, ako sa takýmto chybám vyvarovať. Po hraní Scrum Poker hry boli vybrané a pridelené úlohy do ďalšieho šprintu.

-ukončené:

- preskúmanie konektivity C# a MongoDB - Martin Dieška, Martin Gembec
- vygenerovanie testovacích jsonov - Martin Gembec
- vytvorenie statického prototypu WebGL cez ThreeJS - Marcel Furucz, Peter Psota
- vytvorenie webstránky - Martin Dieška
- nasadenie webstránky na server - Patrik Sčensný
- vytvorenie prvotných služieb na získavanie dat z MongoDB - Martin Gembec, Martin Dieška
- vizualizácia objektov pomocou THreeJS - Peter Psota

-neukončené:

- definícia modelových schém - Štefan Motko, Ronald Demeter, Patrik Sčensný
- preštudovanie metamodelu - všetci
- implementácia ovládania kamery pomocou ThreeJS - Marcel Furucz



Obr. 1: Burndown chart pre šprint Layer

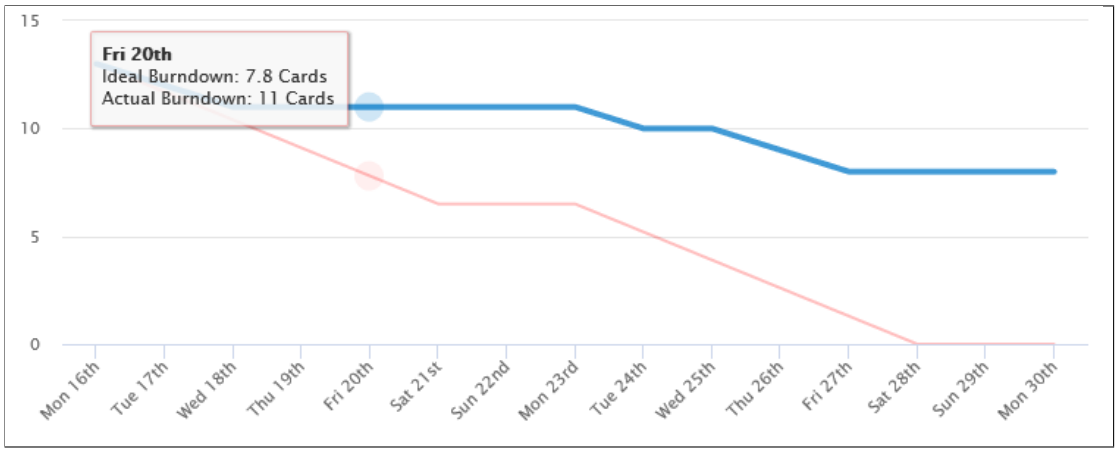
3.3 Šprint 2 - Lifeline

Druhý šprint bol zameraný na implementáciu pokročilejších grafických prvkov, ako napríklad lifeline, a spojením backendu s frontendom do jedného celistvého prototypu.
-ukončené:

- vytvorenie služby pre ukladanie dat v MongoDB - Martin Dieška, Martin Gembec
- dokončenie implementácie ovládania kamery pomocou ThreeJS - Marcel Furucz
- vloženie lifeline na prvú intersect layeru - Peter Psota
- implementácia metamodelu do dátového modelu - Štefan Motko, Ronald Demeter, Peter Psota, Martin Gembec, Martin Dieška
- preštudovanie metamodelu - všetci

-neukončené:

- definícia modelových schém - Štefan Motko, Ronald Demeter, Patrik Sčensný



Obr. 2: Burndown chart pre šprint Lifeline

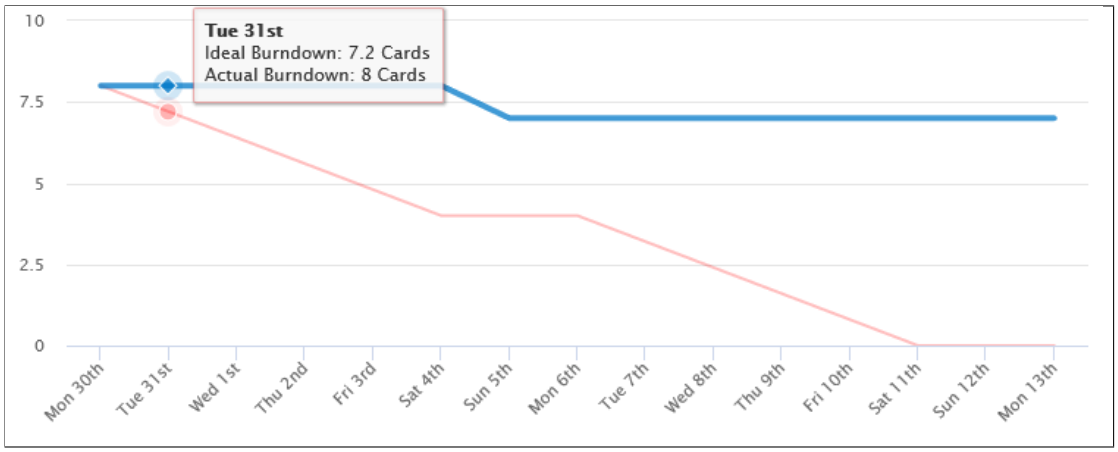
3.4 Šprint 3 - Message

Ako cieľ tretieho šprintu sme si určili pridávanie správ do diagramu a komunikáciu frontendu s backendom. Počas implementácie sme narazili na komplikácie pri orderingu správ, ktoré sme vyriešili diskusiou v druhom týždni šprintu a dohodnutím sa na ďalšom postupe pri riešení.
-ukončené:

- jednoduchý ordering - Štefan Motko, Ronald Demeter, Peter Psota
- automatické layoutovanie prvkov - Peter Psota
- riešenie komplikácii s kamerou z dôvodu migrácie na typescript - Marcel Furucz

-neukončené:

- serializér objektov - Martin Dieška, Martin Gembec
- deserializér objektov - Martin Dieška, Martin Gembec
- komunikácia pomocou GET/PUT/POST - Martin Dieška, Martin Gembec
- definícia modelových schém - Štefan Motko, Ronald Demeter, Patrik Sčensný



Obr. 3: Burndown chart pre šprint Message

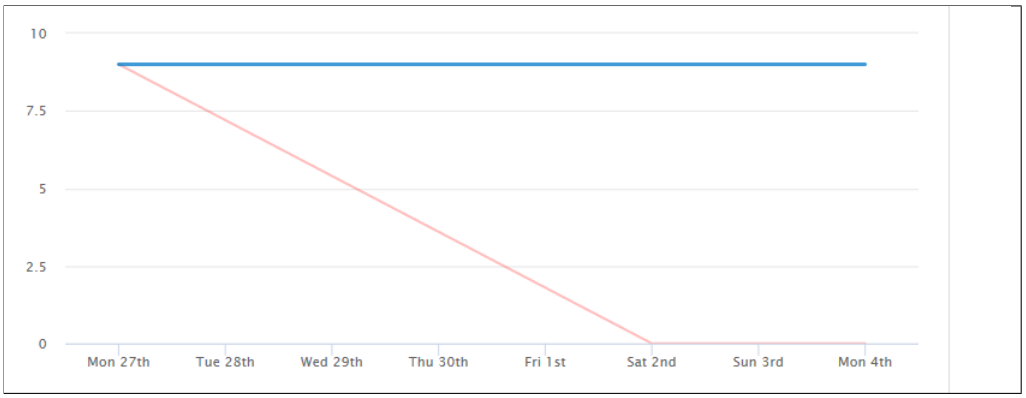
3.5 Šprint 4 - Create

Cieľom štvrtého šprintu bola prvá implementácia vytvárania správ, vrstiev a čiar života. Taktiež pokračovala implementácia serializeru a deserializeru na backende.
-ukončené:

- východiskové umiestnenie kamery - Marcel Furucz, Ronald Demeter
- vylepšenie raycastingu - Peter Psota

-neukončené:

- serializér objektov - Martin Dieška, Martin Gembec
- deserializér objektov - Martin Dieška, Martin Gembec
- komunikácia pomocou GET/PUT/POST - Martin Dieška, Martin Gembec
- spracovanie udalostí na plátne - Štefan Motko, Ronald Demeter



Obr. 4: Burndown chart pre šprint Create

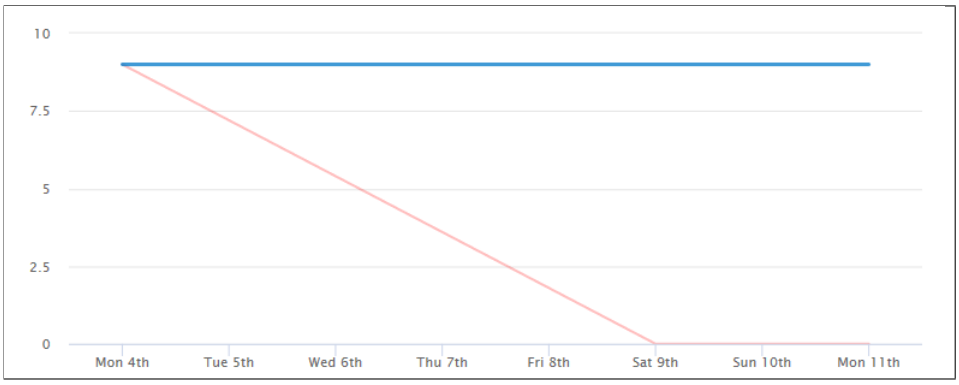
3.6 Šprint 5 - State

Piaty šprint bol zameraný na implementáciu stavového stroja na frontende. Taktiež sa začala implementácia odstraňovania správ, čiar života a vrstiev.
-ukončené:

- odstránenie správy - Ronald Demeter
- odstánenie čiary života - Ronald Demeter
- spracovanie udalostí na plátne - Štefan Motko, Ronald Demeter

-čakajúce:

- serializér objektov - Martin Dieška, Martin Gembec
- deserializér objektov - Martin Dieška, Martin Gembec
- komunikácia pomocou GET/PUT/POST - Martin Dieška, Martin Gembec
- odstránenie vrstvy - Ronald Demeter

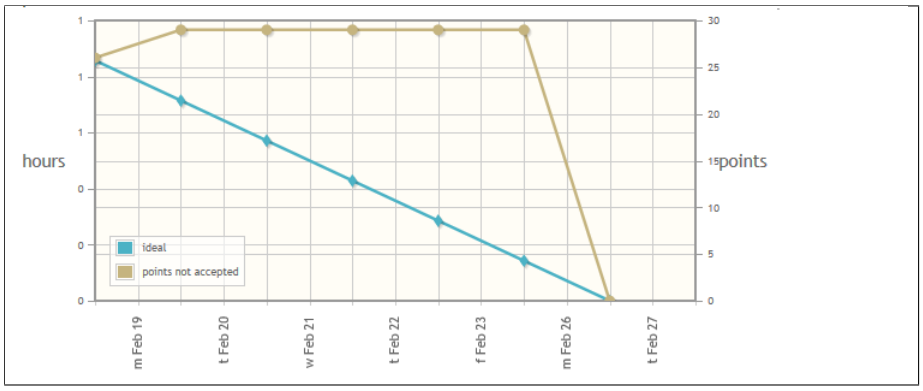


Obr. 5: Burndown chart pre šprint State

3.7 Šprint 6 - State-Machine

V šiestom šprinte sme sa venovali zdokonaleniu stavového stroja. Okrem toho tím pokračoval v implementácií funkcionality pre správy, konkrétne pridávanie správ na index. Taktiež sa vytvorili príklady testovania pre overenie už implementovaných funkcií.
-ukončené:

- Prípady testovania - Patrik Sčensný
- Integrácia modelu front-end back-end - Marcel Furucz, Martin Dieška
- Stavový automat - Štefan Motko
- Pridanie Message na index - Peter Psota



Obr. 6: Burndown chart pre šprint State-Machine

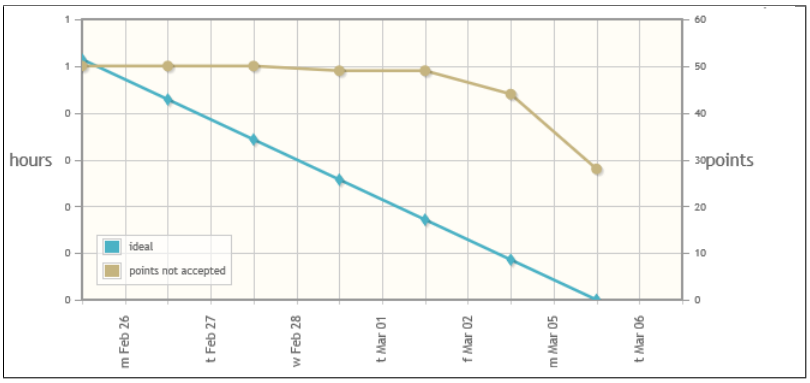
3.8 Šprint 7 - Continous-Integration

Siedmy šprint bol zameraný na kontinuálnu integráciu. Pri testovaní sa tiež objavila chyba, ktorá nadmerne vyťažovala grafickú kartu a musela byť opravená. Tiež započala práca na animáciách pri interakcií s prvkami diagramu a na grafickom rozhraní.
-ukončené:

- Grafické rozhranie - Ronald Demeter

-čakajúce:

- Animácie - Peter Psota
- Testovanie - Patrik Sčensný, Martin Gembec, Martin Dieška, Marcel Furucz



Obr. 7: Burndown chart pre šprint Continous-Integration

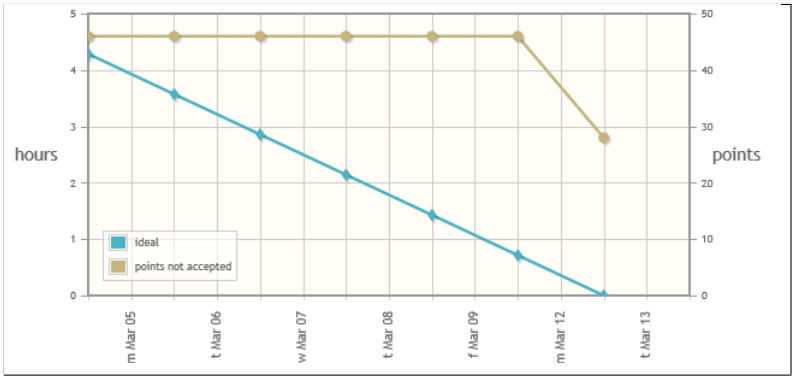
3.9 Šprint 8 - Test-Cases

Počas ôsmeho šprintu sme začali so širším a podrobnejším testovaním. Pokračovalo sa na práci na animáciách a začala príprava fragmentu pre implementáciu do nášho projektu.
-ukončené:

- Pridávanie správ medzi správy - Peter Psota
- Príprava fragmentu - Štefan Motko

-čakajúce:

- Animácie - Peter Psota
- Testy - Patrik Sčensný, Martin Dieška, Martin Gembec, Marcel Furucz



Obr. 8: Burndown chart pre šprint Test-Cases

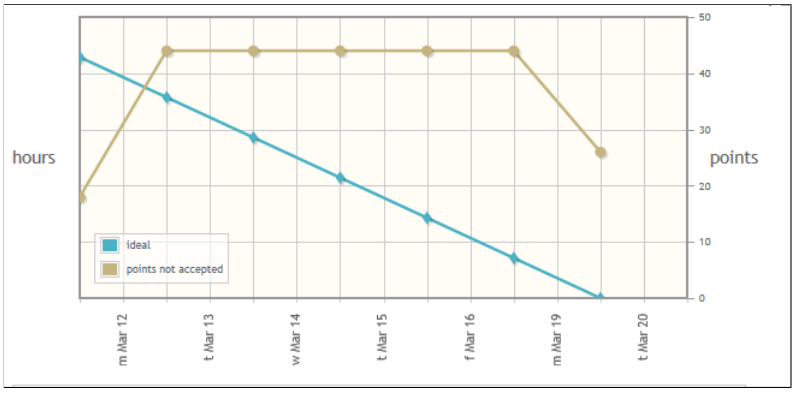
3.10 Šprint 9 - Pop-Up

Prioritou deviateho šprintu bolo dokončenie práce na animáciách a implementácia vyskakovacieho okna, pre modifikáciu a editáciu parametrov v prvkoch diagramu. Tiež sa rozšírila funkcionalita kamery o centrovanie na vrstvu.
-ukončené:

- Animácie - Peter Psota
- Testy - Patrik Sčensný, Martin Dieška, Martin Gembec, Marcel Furucz
- Vyskakovacie okno - Ronald Demeter
- Centrovanie kamery na vrstvu - Marcel Furucz

-čakajúce:

- Zmena ukotvenia správy - Štefan Motko



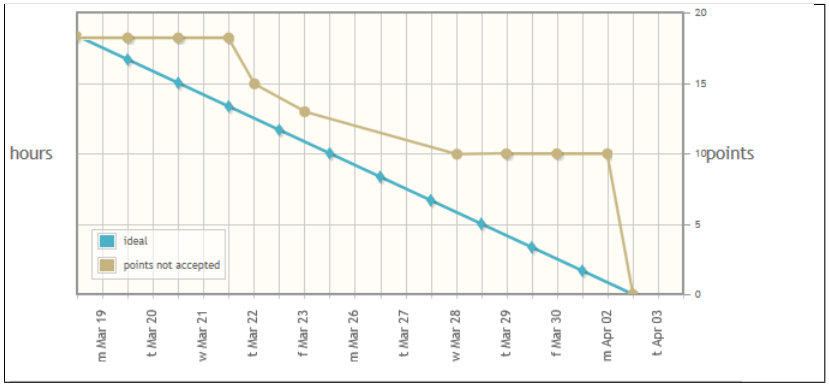
Obr. 9: Burndown chart pre šprint Pop-Up

3.11 Šprint 10 - Animation

V tomto šprinte bolo úlohou implementovať pokročilé animácie pri centrovaní kamery na vrstvu a rozšíriť funkcionalitu vyskakovacieho okna.
-ukončené:

- Zmena ukotvenia správy - Štefan Motko
- Animácia pri centrovaní kamery - Marcel Furucz, Peter Psota

- Funkcionalita vyskakovacieho okna - Martin Dieška, Ronald Demeter



Obr. 10: Burndown chart pre šprint Animation

4 Používane metodiky

4.1 Metodika plánovania

Táto metodika popisuje metódy používané pri plánovaní, opisuje nástroje a metodiky na to používané. Obsahuje pravidlá pre vytváranie a narábanie s používateľskými príbehmi a úlohami.

4.2 Metodika konvencií písania zdrojového kódu

Pri backend sa používa štandardná c# anotácia odporúčaná Microsoftom. Pri frontende sa používa podobná anotácia ako pri backende.

4.3 Metodika testovania

Pri testovaní používame JavaScript testovací framework Mocha bežiaci na Node.js a v prehliadači. Testy sú písané popri vytváraní písaní zdrojového kódu a konzultácií s autorom funkcionality a projektovým manažérom.

4.4 Metodika prehliadky zdrojového kódu - code review

Pri prehliadky zdrojového kódu sa využíva funkcionalita poskytovaná githubom (Pull request). Žiadateľ vytvori pull request. Ten request sa zhodnotí a podľa zhodnotenia sa vykonajú zmení.

4.5 Metodika pre mapovanie služieb

Metodika určujú architektúru, spôsoby pomenovania a miesto uloženia služby. Takisto je v nich napísaný spôsob mapovania volania týchto služieb.

4.6 Metodika verziovania zdrojového kódu

Táto metodika opisuje pravidla pre vetvenie, verziovanie a zálohovanie zdrojového kódu v rámci Github. Píše sa v nej na akých vetvách sa vypracujú používateľské príbehy, ako tie vetvy vytvárať. Zároveň obsahuje sadu pravidiel, ktoré sa majú dodržiavať počas práci s githubom.

4.7 Metodika písania dokumentácie

Táto metodika popisuje základné pravidlá a postupy pri písaní dokumentácie a zápisníc po každom stretnutí. V tejto metodike nájdeme pravidlá, ako napr.: použité textové editory, veľkosť písma, formát tabuliek a grafov.

5 Globálna retrospektíva

V tejto kapitole je opísaná globálna retrospektíva tímového projektu počas zimného aj letného semestra. Našou snahou bolo celé dva semestre zlepšovať tímovú komunikáciu a spoluprácu. Snažili sme sa zabezpečiť čo najlepšiu efektivitu riešenia problémov a rozdelenia práce medzi členov tímu. Vždy na konci šprintu, čo bolo v zimnom semestri každé dva týždne sme sa venovali retrospektíve v ktorej sme sa rozprávali o tom čo sme dosiahli a čo by sme chceli zlepšiť. V letnom semestri sme sa dohodli na jednotýždňových šprintoch čo viedlo k lepšej efektivite práce. Každý člen tímu mal možnosť sa vyjadriť k problémom s ktorými sa stretol a vysvetliť čo sa mu podarilo

spraviť. Dôležitou súčasťou retrospektívy bolo poukázať na ďalšie smerovanie projektu a upovedomiť ostatných členov tímu a častiach projektu, ktoré nefungujú správne.

Prvé týždne semestra sme sa zameriavali na zvolenie správnych technológií. Bolo pre nás veľmi dôležité zanalyzovať všetky dostupné technológie a zvoliť tie najvhodnejšie, pomocou ktorých sme schopný naplniť všetky požiadavky zákazníka. Práve zle zvolená technológia bola často dôvodom zdržania a neúspechu predchádzajúcich tímov, ktoré sa venovali Web UML tématike. Výber technológii prebiehal na základe dohody členov tímu a o každej technógii sme spolu diskutovali a ukázali si jej výhody a nevýhody.

Niektorí členovia tímu nemali väčšie skúsenosti s verziovaním kódu a preto vedúci tímu urobil prednášku zameranú na GIT technológiu v ktorej všetkým členom tímu vysvetlil nevyhnutné základy. Oboznámili sme sa s technológiou na komunikáciu, konkrétne služba Slack a nástrojom na riadenie užívateľských príbehov s názvom Trello.

Vďaka predchádzajúcim skúsenostiam niektorých členov tímu v danej tématike sme sa rýchlo prepracovali k prvým návrhom projektu. Na základe retrospektív sme vždy dokázali odhaliť aktuálne problémy s ktorými sme sa následne potýkali v ďalších šprintoch. Jednalo sa o rôzne problémy:

- nedostatočná komunikácia tímu počas šprintu, kde členovia spolu komunikovali len málo a radšej problém riešili sami
- technické problémy spôsobené novými technológiami, s ktorými sa členovia tímu nikdy predtým nestretli
- odkladanie práce na stretnutie
- nedostatočné vysvetlenie problému spôsobujúce zlý postup pri jeho riešení

Koncom zimného semestra sa nám podarilo sa nám vytvoriť systém, ktorý poskytuje prvotné grafické rozhranie, ktoré zobrazuje niekoľko vrstvový diagram, ktorý je priamo pomocou servera načítaný z databázy a zobrazovaný na klientovi, konkrétne vo webovom prehliadači. Používateľ môže otáčať kamerou a zobrazovať si diagram z rôznych uhlov a vzdialeností. Taktiež je možné pridávať do diagramu layers, lifelines a messages. Každá zmena sa ukladá do databázy a dáta sú perzistentné.

Začiatok letného semestra bol zameraný na zdokonalenie stavového stroja, ktorý je kľúčovým prvkom našej webovej aplikácie. Taktiež sme vytvorili prvé

príklady testovania pre overenie už implementovaných funkcií a namiesto Trella sme začali používať nástroj Redmine, ktorý je omnoho profesionálnejší a poskytoval nám požadovanú funkcionalitu.

Pre lepší používateľský zážitok sme zapracovali do vizuálnej stránky animácie a zjednodušili interakciu s rozhraním. Testovanie aplikácie sa postupne prehlbovalo a prešli sme k príprave našej aplikácie na implementáciu fragmentu. Pre jednoduchšiu úpravu položiek v diagrame z dôvodu variabilného počtu polí vo formulári sme implementovali editačné vyskakovacie okno. Následne sme ďalej pracovali na krajších animáciách v aplikácii.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25

Motivačný dokument

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Akademický rok: 2017/2018

1 Predchádzajúce skúsenosti tímu

- vizualizácia a interakcia s vizualizovanými dátami, prezentácia dát
  - vizualizácia vo webovom prostredí
    - Štefan Motko
    - Ronald Demeter
    - Marcel Furucz
    - Martin Dieška
- manipulácia s dátami(XML, JSON)
- skúsenosti vo vývoji webových aplikácií
  - HTML, CSS, JavaScript, ASP.NET
    - Štefan Motko
    - Martin Dieška
    - Ronald Demeter
    - Marcel Furucz
    - Patrík Sčensný
- skúsenosti v práci s grafickými prostrediami
  - Three.js, D3.js, DevExtreme
    - Ronald Demeter
    - Martin Dieška

Téma prioritného záujmu:

2 14. 3D-UML

Viacerí členovia tímu sa v rámci svojich bakalárskych projektov venovali vizualizáciám v prehliadači, pričom niektorí aj konkrétne vizualizáciám správania softvéru. Viacerí sa zároveň chcú v inžinierskom štúdiu ďalej venovať oblastiam modelovania a architektúry softvéru, ďalší majú záujem o činnosť v oblasti vizualizácie. Tiež sme absolvovali predmety zamerané na modelovanie v UML z ktorých získané poznatky vieme využiť v danom projekte.

3 Príloha A

Zoradenie tém podľa priority od najvyššej po najnižšiu

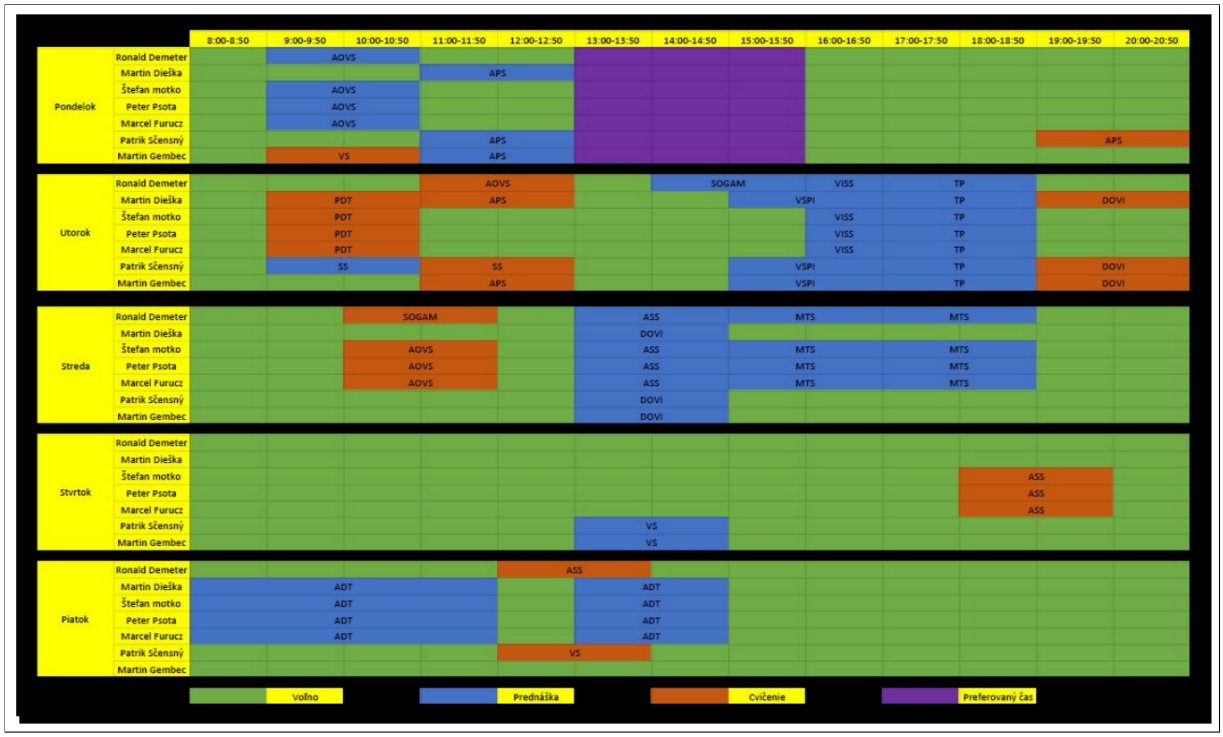
- 3D-UML(14)
- IBazar(4)
- reCommers(9)

Následne hociktorá z nasledujúcich tém s rovnakou prioritou.

- Group(2)
- Zmluvy(3)
- Invest(24)
- EduVirtual(8)
- Look-Inside-Me(10)
- Collab-UI(12)
- DeepSearch(15)
- iWeb(19)
- BehaMetrics(20)
- SmartParking(26)
- OntoSEC(18)
- Mob-UX(13)
- FIIT-DU(11)
- PubDatasets(1)

A v poslednom rade hociktorá zo zostávajúcich tém.

4 Priloha B



Obr. 1: Rozvrh kazdeho clena tymu

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika pre prehliadku kódu

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Prehliadka kódu

Človek zodpovedný za používateľský príbeh vytvorí pull request s názvom Code Review_MenoŽiadateľa a vyberie jedného alebo viacerých členov tímu, ktorý bude mať rolu posudzovateľa.

Posudzovateľ po žiadosti o review vykoná prehliadku kódu a komentármi označí miesta nesprávne napísaného kódu. Popritom berie do úvahy metodiku písania kódu.

Po prehliadke posudzovateľ pošle žiadateľovi informáciu ohľadom nedostatkov. Žiadateľ potom ide vyriešiť nedostatky po ich odstránení to oznámy posudzovateľovi. Ak posudzovateľ nenájde ďalšie nedostatky tak schváli pull request. Predchádzajúce kroky sa opakujú pokiaľ sa neodstránia všetky nedostatky.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika písania zdrojového kódu

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Serverový kód

Pri písaní serverového kódu (od teraz backend) sme sa rozhodli použiť štandardnú metodiku písania kódu jazyka C# odporúčanú Microsoftom.

1.1 Konvencia nazývania

Triedy a metódy sa píšu vo formáte PascalCase každé slovo začína veľkým písmenom. Premenné a atribúty sa píšu camelCase prvé slovo začína malým písmenom a každé ďalšie slovo veľkým. Privátne premenné začínajú "_".

1.2 Rozmiestnenie

Jedna deklarácia a vyraz na riadok. Používanie tabulátora veľkosťou 4 medzery. Jeden prázdny riadok po definícii funkcie, metódy a podmienky.

1.3 Komentáre

Komentáre sa píšu na samostatný riadok a nie na koniec. Komentáre začínajú veľkým písmenom a sú oddelene od // jednou medzerou a zakončené bodkou.

1.4 Jazyk

Jazyk písania kódu je angličtina.

2 Klientský kód

2.1 Konvencia nazývania

Triedy a metódy sa píšu vo formáte PascalCase každé slovo začína veľkým písmenom. Premenné a atribúty sa píšu camelCase prvé slovo začína malým písmenom a každé ďalšie slovo Veľkým. Privátne premenné začínajú "_"

2.2 Rozmiestnenie

Jedna deklarácia a vyraz na riadok. Používanie tabulátora veľkosťou 4 medzery. Jeden prázdny riadok po definícii funkcie, metódy a podmienky.

2.3 Komentáre

Komentáre sa píšu na samostatný riadok a nie na koniec. Komentáre začínajú veľkým písmenom a sú oddelene od // jednou medzerou a zakončené bodkou.

2.4 Jazyk

Jazyk písania kódu je angličtina.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika dokumentácie

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Dokumentácia

Táto sekcia obsahuje pravidlá a postupy, ktoré musia byť dodržiavané pri písaní dokumentácie.

1.1 Použité nástroje

Dokumentácia ku tímovému projektu je vytváraná v textovom editore ShareLatex.

1.2 Veľkosť a štýl písma

Font a veľkosť písma boli ponechané na východiskový stav article v Latex-u. Pre vytvorenie nadpisov boli použité príkazy ShareLatex-u: hlavné nadpisy príkazom section, nadpisy druhej úrovne príkazom subsection a nadpisy tretej úrovne príkazom subsubsection.

1.3 Tabuľky

Pre tabuľky v dokumente je špecifické modré zafarbenie hlavičky tabuľky. Číslo tabuľky spolu s jej názvom je zadávaný pod tabuľku, centrovaný do stredu dokumentu.

Meno člena tímu	Zodpovednosť v týme
Štefan Motko	Projektový manažér
Ronald Demeter	Hlavný vývojár
Marcel Furucz	Scrum a komunikácia
Patrik Sčensný	Testovanie a kvalita
Peter Psota	Grafický odborník
Martin Gembec	Administrácia databáz a dokumentácia
Martin Dieška	Vývoj serverovej strany

Tabuľka 1: Role členov tímu

Obr. 1: Príklad tabuľky zo zápisnice

2 Zápisnica

V tejto časti sú uvedené pravidlá pri písaní zápisnice.

2.1 Použité nástroje

Pre vytváranie zápinsníc zo stretnutí bol použitý textový editor Microsoft Word.

2.2 Veľkosť a štýl písma

V zápisniciach zo stretnutí je použitý font Calibri, veľkosti 11. V prípade nadpisov je použitý font Calibri Light, veľkosti 13 a modrej farby. Jedná sa o prednastavené písmo pre nadpisy, ktoré ponúka textový editor Microsoft Word.

2.3 Tabuľky

Tabuľky majú predpísanú formu, do ktorej sa po stretnutí dopĺňajú informácie. Na obrázku nižšie je uvedený príklad vyplnenej tabuľky zo zápisnice.

#	Úloha	Zodpovedné osoby	Dátum ukončenia	Status
1	Vytvorenie webstránky	Martin Dieška	16.10.2017	Dokončená
2	Vytvorenie prvotných služieb na získavanie dát z MongoDB	Martin Dieška, Martin Gembec	16.10.2017	Dokončená
3	Implementácia ovládania kamery pomocou ThreeJS	Marcel Furucz	16.10.2017	Nedokončená
4	Vizualizácia objektov pomocou ThreeJS	Peter Psota	16.10.2017	Dokončená
5	Nasadenie webstránky na server	Štefan Motko, Ronald Demeter, Patrik Sčensný	16.10.2017	Dokončená
6	Definícia modelových schém	Štefan Motko, Ronald Demeter, Patrik Sčensný	16.10.2017	Nedokončená
7	Preštudovanie metamodelu	Všetci	16.10.2017	Dokončená

Obr. 2: Príklad tabuľky zo zápisnice

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika plánovania

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

Metodika plánovania hovorí o vytváraní, prideľovaní a manažmente úloh a používateľských príbehov.

1 Šprint

Cieľom šprintu je okrem iného aj vytvoriť príbehy s dostatočnou granularitou. Každý používateľský príbeh je vytvorený, ohodnotený príbehovými bodmi a poprípade následne rozdelený na menšie. Pri vytvarani príbehov je pár pravidiel, ktoré sa dorživajú:

- Príbeh je dobre a jasne popísaný
- Príbeh ma určené aspoň približné úlohy

2 Vytváranie uloh

Po vytvorení príbehov sú prideľované členom tímu na vývoj podľa sekcie. Je možné, aby na jednom príbehu pracovalo viacej členov tímu, nie však na jednej úlohe.

3 Stavy používateľského príbehu

V priebehu zivota používateľský príbeh prechádza rôznymi stavmi:

- Backlog - východiskový stav. Príbeh sa takto označí po vytvorení, pred zaradením do šprintu.
- In development - Príbeh je zaradneý do šprintu a je aktívne vyvýjaný
- Testing - Príbeh je funkčný, prebieha kontrola kvality kódu a testovanie
- Pending - Príbeh bol skontrolovaný a má vytvorený pull request.
- Done - Príbeh bol zlúčený do dev vetvy.

Prechody medzi stavmi majú niekoľko pravidiel:

- Z backlogu do in development sa úlohy presúvajú len priradenými členmi tímu, potom ako na ňom začnú pracovať
- Pull request musí byť schválený všetkými členmi tímu, ktorý boli na príbeh priradený.

4 Spravovanie používateľských príbehov

Každý používateľský príbeh je zaradený do sekcie podľa jeho zamerania. Vedúci sekcie dohliada na používateľský príbeh pod jeho sekciou a organizuje stretnutia členov k ním priradeným.

5 Nástroj Trello

Na plánovanie a spravovanie používateľských príbehov používame voľne dostupný nástroj Trello. Pouzivatelske príbehy sú karticky, ktoré sa presúvaju medzi stĺpcami, reprezentujúce ich stav. Členovia tímu sú registrovaný na Trelle a priraďujú sa k príbehom, ktoré su im určené. Príbehove karticky majú zoznamy položiek, reprezentujuce úlohy. Sekcie používateľských príbehov sú reprezentovane farbami kartičiek.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika pre mapovanie služieb

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Architektúra

Metodika webových služieb vychádza z REST architektúry. Každá služba musí obsahovať niektorú z RESTových metód, aby sme mohli hovoriť o RESTovej architektúre webových služieb. Jedná sa o metódy:

- GET - čítanie a len čítanie zdrojov
- PUT - aktualizácia existujúceho zdroja
- DELETE - odstránenie zdroja
- POST - pridanie nového zdroja

2 Pomenovanie služieb

Služby pomenúvame podľa objektu s ktorým pracujú. Pokiaľ sa jedná o prácu s viacerými objektami, pomenúva službu podľa funkcionality, ktorú poskytuje nad danými objektami.

3 Smerovanie

Pre poskytovanie služieb sa využívaju objekty Controller, keďže serverová architektúra je typu MVC. Každý controller obsahuje niekoľko metód a môžeme hovoriť o službe. Metódy sú mapované nasledovne: Na začiatku controllera sa nastaví URL cesta na ktorej sa controller nachádza. Následne každá jednotlivá metóda ďalej špecifikuje, ako sa bude volať:

api/[controller]/[action]/parameters

4 Framework

ASP .NET Core 2 nám poskytuje ASP.NET MVC 6 Controller, ktorý spája funkcionalitu predchádzajúceho MVC modelu a Web API. Obsahuje niekoľko typov smerovaní, vie vracať View aj objekt, ktorý dokáže serializovať.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika testovania

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Unit testing

V projekte rozdelíme testovanie na dve časti:

- **testovanie backendu** - na tetovanie serverového kódu sa používajú štandardné anotácie pre testovanie dostupné v jazyku C#
- **testovanie frontendu** - na testovanie klientskeho kódu prebieha pomocou nástroja Mocha

Na základe požiadaviek a špecifikácií od člena týmu, osoba zodpovedná za písanie testov navrhne a napíše test, ktorý sa bude využívať na zistenie správnej funkcionality.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Metodika verziovania

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

1 Vetvenie zdrojového kódu

Na verziovanie zdrojového kódu používame verziovací systém Git. Spoločné úložisko zdrojových kódov udržiavame na stránke GitHub. Vytvorili sme dve primárne vetvy(angl. branch).

- master - produkčná vetva - obsahuje poslednú prezentovanú a akceptovanú verziu vyvíjaného softvéu
- dev - vývojová vetva - obsahuje najnovšie dokončené črty

Pre každý používateľský príbeh sa vytvorí nová vetva s názvom feature/"popis"(kde popis je názov vyvíjanej črty), v ktorej sa pracuje na danom používateľskom príbehu. V prípade zistenia chýb vo vetvách master alebo dev sa vytvára nová vetva s názvom fix/"popis".

1.1 Postup vytvárania vetvy pre používateľský príbeh

Na používateľskom príbehu sa pracuje na samostatnej vetve. Vetvu vytvorí jeden z členov tímu, ktorému bolo priradené riešenie daného používateľského príbehu. Pre vetvy platia nasledujúce pravidla:

- Názov sa píše po anglicky.
- Názov má tvar feature/"príbeh"napr. "feature/state-machine"
- Viacslovné pomenovania sa oddeľujú pomlčkou.

Postup pri vytvorení novej vetvy:

1. Otvoriť konzolu schopnú spustiť program git
2. Prepnúť sa na vetvu dev pomocou príkazu "git checkout dev"
3. Vytvoriť novú vetvu pomocou príkazu "git checkout -b <meno novej vetvy>"
4. Publikovať novú vetvu na vzdialené úložisko pomocou príkazu "git push –set-upstream origin <meno vytvorenej vetvy>"

1.2 Pravidla pracovania vo vetve

1. Do vetvy master a dev sa neprispieva priamo
2. Pracuje sa exkluzívne vo vetvách feature/* a fix/*
3. Webová prezentácia projektu má vlastné úložisko nezávislé od vyvíjaného softvéru
4. Pred vytvorením novej feature vetvy je potrebné aktualizovať lokálnu verziu vetvy dev
5. Revízie zdrojového kódu sa ukladajú po čo najmenších logických celkoch pre udržiavanie čitateľnosti histórie
6. Akýkoľvek text zadávaný do verziovacieho systému je v anglickom jazyku
7. Pri ukončení práce na používateľskom príbehu vytvorí spracovateľ pull request do vetvy dev, pričom všetkých kolaborátorov zaradí do vetvy ako schvaľovateľov
8. Po integrácii vývojovej vetvy do vetvy dev sú členovia tímu pracujúci na ostatných vývojových vetvách povinní spätne integrovať zmeny vo vetve dev do svojich vývojových vetiev

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Projektová dokumentácia - inžinierske dielo

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

Obsah

1 Úvod **2**
1.1 Globálne ciele projektu 2
1.1.1 Zimný semester 2
1.1.2 Letný semester 2

2 Celkový pohľad na aplikáciu **3**

3 Moduly systému **4**
3.1 Backend 4
3.1.1 ASP.NET Core 2 - Webserver 4
3.1.2 MongoDB Databáza 4
3.1.3 Testovanie backendu 4
3.2 Frontend 4
3.2.1 Grafické plátno 4
3.2.2 Stavový stroj 5
3.2.3 Konfigurácia stavového stroja 5
3.2.4 ThreeJS 5
3.2.5 Zobrazovač diagramu 5
3.2.6 Model diagramu 5
3.3 Dátový model 5

1 Úvod

Tento dokument slúži ako dokumentácia inžinierskeho diela, pre tímový projekt v akademickom roku 2017/2018. Dokument zahŕňa globálne ciele projektu, celkový pohľad na štruktúru a funkcionalitu projektu.

Téma projektu je 3D uml. Cieľom projektu je vytvoriť webovú aplikáciu, ktorá dovolí používateľovi vytvárať a modifikovať sekvenčné diagramy v 3D priestore. Vykresľovanie diagramu bude realizované pomocou WebGL. Dokument má niekoľko častí. V prvej sa dozvieme o cieľoch projektu, plánoch pre konkrétne semestre a pohľadu na navrhnutú formu finálnej aplikácie.

1.1 Globálne ciele projektu

V tejto kapitole opisujeme ciele projektu pre zimný a letný semester vzhľadom na konzultácie a predstavu našeho vedúceho projektu. Globálnymi cieľmi sú:

1. Reprezentácia sekvenčného diagramu pomocou trojorozmených elementov
2. Vytváranie dvojrozmerných vrstiev do ktorých sa budú kresliť 3D elementy
3. Vytváranie viacerých vrstiev podľa vstupov používateľa
4. Prepojenie vrstiev medzivrstvovými správami
5. Integrácia kódových fragmentov do našej reprezentácie sekvenčného diagramu

1.1.1 Zimný semester

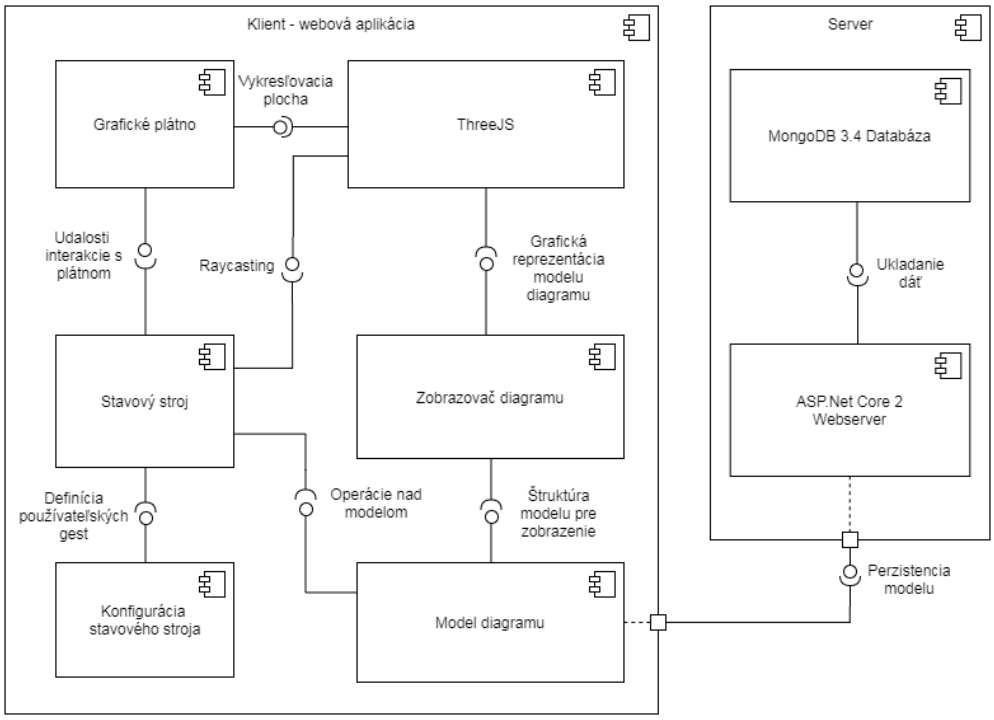
Cieľom zimného semestra je definovanie dátového modelu a následná implementácia základných funkcionalít vytvárania, odstraňovania elementov z diagramu a ich následné uloženie do databázy.

1.1.2 Letný semester

Cieľom letného semestra je implementácia zvyšných cieľov v projekte. Predbežne je naplánované implementovanie fragmentov a medzivrstvovú interakciu.

2 Celkový pohľad na aplikáciu

Aplikácia je spravená vo forme klient-server. Máme tučného klienta, čo znamená, že väčšina našich výpočtov sa odohráva na strane klienta.
V našom prípade server slúži primárne ako databáza obsahujúca používateľské sekvenčné diagramy.
Klientská časť je sprístupnená v prehliadači. Používateľ s ňou interaguje, čiže kreslí elementy, vytvára sekvenčné diagramy, prehliada a ukladá si ich.
Klient so serverom komunikujú za pomoci AJAX-u.



Obr. 1: Projektové moduly a ich komponenty.

3 Moduly systému

3.1 Backend

Táto sekcia popisuje analýzu a technológie použité na backende našej aplikácie, teda serveri. Server má na starosti perzistenciu dát a ich poskytovanie cez preddefinové rozhranie.

3.1.1 ASP.NET Core 2 - Webserver

Hlavnou úlohou webového severa je poskytovať webové služby, ktoré umožňujú frontendu ukladať a získavať dáta, respektíve ich nejak modifikovať. Dátami sa v tomto prípade myslí reprezentácia diagramu, zobrazujúceho sa na frontende.

3.1.2 MongoDB Databáza

Server používa MongoDB, čo je nerelačná databáza, ktorá ukladá dáta vo forme JSON dokumentov. Tieto vlastnosti zaručujú nízku odozvu pri prístupe k dátam a nie je nutné rozkladať diagram na menšie časti, ako u relačnej databázy, stačí ho vo forme JSON objektu uložiť do databázy.

3.1.3 Testovanie backendu

Pre testovanie backendu sme sa snažili vyhnúť problémom ktoré sa ukázali v minuloročnom tímovom projekte. Keďže sme v tejto fázy vývoja nemali funkčné automatické testovanie, tak sme dopredu definovali správnu funkcionalitu jednotlivých modulov. Túto definíciu sme použili na odhalenie nedostatkov, ale namiesto automatických testov sme použili manuálne testovanie.

3.2 Frontend

V tejto sekcií sa budeme venovať klientskej aplikácií.

3.2.1 Grafické plátno

Poskytuje vykresľovaciu plochu, kde sa môže 3d diagram vykresľovať. Zachytáva udalosti, ktoré sú neskôr používane v stavovom stroji.

3.2.2 Stavový stroj

Stavový stroj zabezpečuje vykonávanie funkcií podľa udalosti, ktoré boli zachytené na grafickom plátne. Informácie o platnosti udalosti čerpá z konfigurácie stavového stroja

3.2.3 Konfigurácia stavového stroja

Tento modul obsahuje zadefinované postupnosti akcií. Ktoré sú platné a používateľ je schopný vykonávať.

3.2.4 ThreeJS

Implementácia externej knižnice, ktorá umožňuje vytváranie 3D útvaroch a prvkov vo WebGl v rámci prehliadača. je zodpovedná za udržiavanie grafického rozhrania na vyžiadanie zobrazovača diagramu.

3.2.5 Zobrazovač diagramu

Poskytuje grafickú reprezentáciu modelov diagramu. Tieto grafické reprezentácie sú následne rozšírené potrebnými prvkami na ich graficku reprezentáciu a vykresľované na platne.

3.2.6 Model diagramu

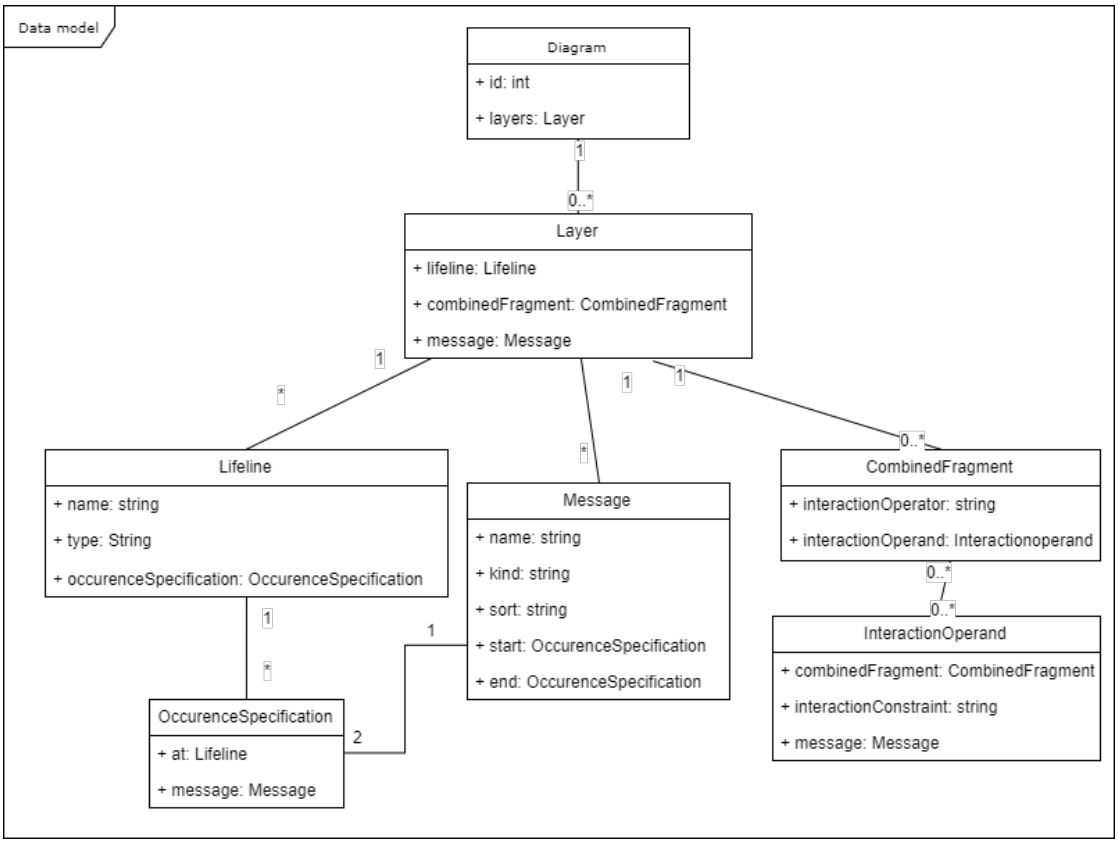
Je podstatná pre správne mapovanie údajov z klientského prostredia na serverovú databázu. Definuje štruktúru dát ktoré sa ukladajú do databázy a načítavajú ak klient potrebuje nahrať už predošle vytvorený diagram.

3.3 Dátový model

Dátový model aplikácii je vidno na 2.

- **Diagram** - Predstavuje súbor všetkých prvkov, ktoré sa majú vykresľovať.
- **Layer** - Predstavuje vrstvu do ktorej sa kreslia prvky sekvenčného diagramu.
- **Lifeline** - Predstavuje životný cyklus objektu a modeluje jeho interakcie v čase v diagrame.

- **Message** - Predstavuje správu medzi dvoma lifeline .
- **CombinedFragment** - Predstavuje interakčný fragment v diagrame sekvencií.
- **InteractionOperand** - Predstavuje podmienku, ktorá musí byť splnená aby sa vykonával časť diagramu, ktorá je v CombinedFragment
- **OccurenceSpecification** - Je bod naviazania meesage na lifeline



Obr. 2: Dátovy model databázy

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Projektová dokumentácia - moduly systému

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

Obsah

1 Úvod **3**

2 Architektúra **4**
2.1 Klient-Server 4
2.1.1 Analýza 4
2.1.2 Analýza predošlého projektu 4
2.2 Súčasná implementácia 4

3 Komponent: Grafické plátno **5**
3.1 Analýza 5
3.2 Návrh 5
3.3 Implementácia 5

4 Komponent: ThreeJS **7**
4.1 Analýza 7
4.2 Návrh 7
4.3 Implementácia 7

5 Komponent: Stavový stroj **7**
5.1 Analýza 7
5.2 Návrh 8
5.3 Implementácia 9

6 Konfigurácia stavového stroja **9**
6.1 Návrh 10
6.2 Implementácia 10

7 Komponent: Model diagramu **11**
7.1 Analýza 11
7.2 Návrh 11
7.3 Implementácia 11

8 Komponent: Zobrazovač diagramu **12**
8.1 Analýza 12
8.2 Návrh 12
8.2.1 Grafická reprezentácia objektov diagramu 12
8.2.2 Rozloženie objektov 12

8.3 Implementácia 12
8.3.1 GraphicElement 12
8.3.2 LayoutControl 14
8.3.3 Text3D 14
8.3.4 CustomMash 14

9 Backend **14**
9.1 Analýza 14
9.1.1 ASP.NET Core Webserver 14
9.1.2 MongoDB Databáza 14
9.2 Návrh 15
9.3 Implementácia 15

1 Úvod

Tento dokument slúži ako detailnejšia dokumentácia inžinierskeho diela, popisujúca jednotlivé moduly a ich komponenty do hĺbky. Každý popis obsahuje analýzu a odôvodnenie, návrh vyplývajúci z analýzy a súčasnú implementáciu. Taktiež je popísaná a odôvodnená architektúra nášeho systému.

2 Architektúra

2.1 Klient-Server

Náš projekt je založený na klient server architektúre.

2.1.1 Analýza

Pre klient server architektúru sme sa rozhodli z dôvodu poskytovania služieb viacerým používateľom. Potencionálne je neskôr možné aplikáciu monetizovať ako SaaS.

2.1.2 Analýza predošlého projektu

Náš projekt je priamym následovníkom projektu z predošlého roku. Dôvodom je identifikovanie nedostatkov predošlého projektu, ich napravenie a reimplementácia.
Najväčším nedostatkom bola rýchlosť odozvy. Za pomoci expertízy člena tímu z predošlého projektu sme dospeli k tomuto záveru.
Zdroj odozvy bola synchronizácia so serverom vykonávaná pri každej zmene. Preto sme sa rozhodli synchronizáciu implementovať asynchrónne. Klient teda priebežne posiela úpravy na server, ale nečaká na potvrdenie. Udržuje si vlastný model modifikovaného diagramu v pamäti.
Toto môže neskôr vytvoriť konfliky s modelom uloženým na serveri. V tomto prípade sa konzultuje s používateľom aktuálnosť uložených modelov.
Aplikácia by mala poskytnúť možnosť vybrať jednu verziu ako správnu, alebo forknutie diagramu na dve rôzne verzie.
Väčšina výpočtov a algoritmov sa odohráva na klientovi, pre minimalizáciu rizika konfliktov, čo má za následok tučného klienta.

2.2 Súčasná implementácia

Implementovali sme architektúru priebežne, smerujúc ku definovanému konečnému stavu. Momentálne v prípade konfliktu je prioritizovaný stav klienta.
So serverom sa komunikuje asynchrónne a modely klienta sú úspešne ukladané na server. Následne sú aj úspešne načítané naspäť do klienta.

3 Komponent: Grafické plátno

Úlohou grafického plátna je zabezpečiť prostredie schopné na vykresľovanie 3D elementov a prvkov.
Zároveň musí poskytovať možnosť interakcie používateľom.
Taktiež musí zabezpečovať efektívnu a rýchlu komunikáciu, s minimalizáciou odozvy.

3.1 Analýza

Na realizovanie požiadaviek treba implementovať nasledovne funkcie:

1. Vytváranie plátna
2. Zachytávanie udalosti

Vytváranie plátna bude prebiehať pomocou html elementu canvas a modulu Threejs.
Po vytvorení plátna používateľ bude môcť vykonávať operácie vytvorenia, odstránenia a modifikácie prvkov na diagrame. Údaje sa budú zachytávať pomocou eventListenerov priamo na plátne.

3.2 Návrh

Scénu dokážeme navigovať za pomoci kamery. Nad touto kamerou sme teda vytvorili **CameraControls**, ktoré interpretujú vstupy používateľa a aplikujú ich na kameru.

3.3 Implementácia

Na používateľnú implementáciu plátna používame **HTML canvas** 1. Po jeho vytvorení sa v typescripte inicializuje webgl canvas 2. Udalosti na plátne sa zachytávajú pomocou **EventListener** 3, ktorú ma typescript v sebe implementovanú .

```
<div id="main" class="container" >
    <button id="openMenu">Open</button>
    <canvas id="webglCanvas" oncontextmenu="return false;"></canvas>
</div>
```

Obr. 1: Html canvas

```
public initializeScene(): void {

    this._canvas = document.getElementById(Config.canvasId) as HTMLCanvasElement

    // initialize renderer
    this._renderer = new WebGLRenderer({
        canvas: this._canvas,
        antialias: true
    });

    this.renderer.setClearColor(0xffffff);
    this.renderer.setPixelRatio(window.devicePixelRatio);
    this.renderer.setSize(window.innerWidth, window.innerHeight);

    // initialize camera
    this._camera = new PerspectiveCamera(35, window.innerWidth / window.innerHeight, 0.1, 3000);

    // initialize scene
    this._scene = new Scene();

    // initialize camera controls
    this._cameraControls = new CameraControls(this.camera);

    // initialize state machine
    this._stateMachine = new StateMachine(this._canvas,stateNeutral);
    initializeStateTransitions();

    // start rendering
    requestAnimationFrame(this.renderLoop.bind(this));

}
```

Obr. 2: Inicializácia webgl canvasu pomocou typescript

```
public constructor(
    private _canvas: HTMLCanvasElement,
    private _callback: (event: Event) => void
) {
    this._canvas.addEventListener('mousemove', this.handleMouseMove.bind(this));
    this._canvas.addEventListener('mousedown', this.handleMouseDown.bind(this));
    this._canvas.addEventListener('mouseup', this.handleMouseUp.bind(this));
    document.addEventListener('click', this.handleButtonPressed.bind(this));
}
```

Obr. 3: Inicializácia eventlistenerov na zachytávanie udalosti

4 Komponent: ThreeJS

4.1 Analýza

Úlohou modulu je zabezpečiť efektívne spracovanie a vykresľovanie grafických elementov. Pri analýze elementu sme brali do úvahy minuloročný tím, ktorý mal problém s grafickou časťou vykresľovania, menovite odozvu.
Analyzovali sme minuloročný tím a identifikovali slabinu a možný dôvod spomalenia. Predošlý tím používal grafickú knižnicu CSS 3D. Analyzovali sme alternatívy a objavili WebGL. WebGL patrí medzi najnovšie prístupy k webovému vykresľovaniu a používa priamo výkon používateľovej grafickej karty.

4.2 Návrh

ThreeJS poskytuje scénu v ktorej sa odohrávajú grafické udalosti. Do tejto scény pridávame objekty za pomoci integrovanej funkcionality ThreeJS. ThreeJS navyše sprostredkováva raycasting pre stavový stroj.

4.3 Implementácia

ThreeJS má preddefinované triedy reprezentujúce grafické objekty a poskytuje funkcie pre narábanie s nimi.
CameraControls zabaľujú kameru do dvoch Grafických objektov, yawObject a pitchObject(yawObject je zabalený do pitchObjectu).
Pri pokuse o zmenu polohy kamery sa aplikuje daná zmena na polohu pitchObject(yawObject a kamera odvíjajú svoju polohu od polohy pitchObjectu).
Pri pokuse o rotáciu kamery sa aplikuje rotácia okolo osi X na pitchObject, rotácia okolo osi Z na yawObject.

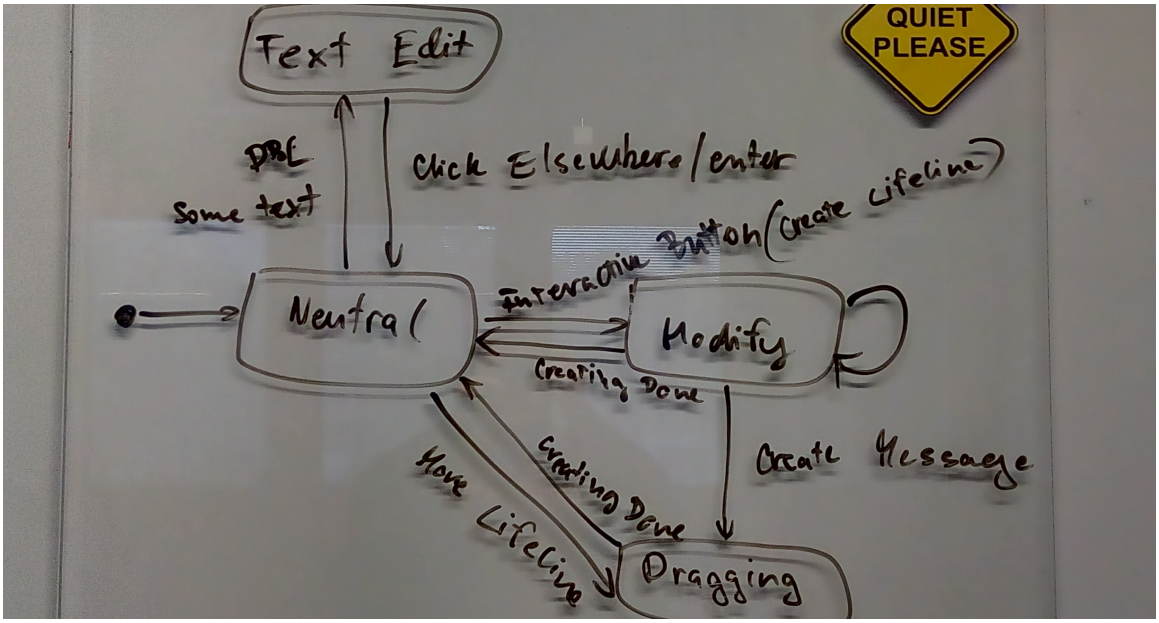
5 Komponent: Stavový stroj

5.1 Analýza

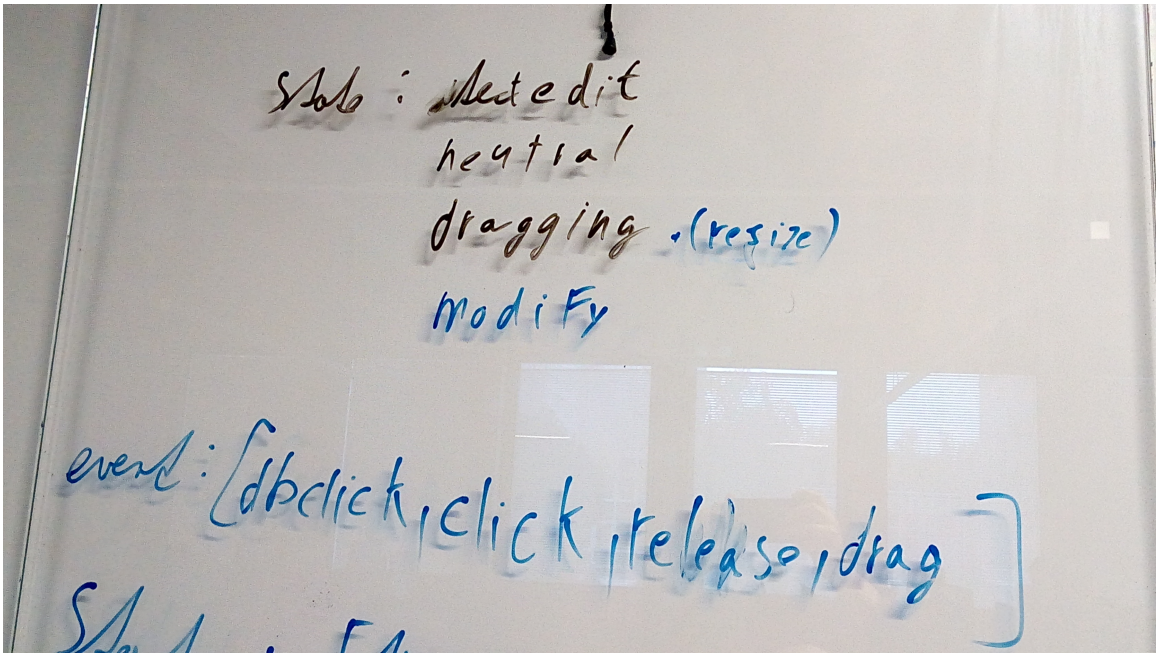
Úlohou modulu je zabezpečiť vykonávanie funkcií podľa aktuálneho stavu plátna.

5.2 Návrh

Pri analýze tejto problematiky sme brali do úvahy viaceré spôsoby zabezpečenia vykonávania funkcionality. Rozhodovali sme sa medzi klasickými event based akcií a stavovým strojom.
Stavový stroj predstavoval ideálne riešenie lebo dovoľoval jednoduché prechody medzi udalosťami a jednoduché zadefinovanie možnosti interakcie.
V prvej fáze sme navrhli možné stavy, vidno na 4.



Obr. 4: Náčrt možných stavov prvotnej implementácie stavového stroja



Obr. 5: Vymenované možne stavy a udalosti, ktoré sú akceptovateľné

5.3 Implementácia

Na implementáciu sme používali navrhnutý stavový stroj, ktorú sme rozdelili na nasledujúce triedy.

1. **EventBus** - zachytáva udalosti na plátne a mimo plátna a informuje stavový stroj o ich nastatí.
2. **State** - predstavuje stav v ktorom sa stavový stroj môže nachádzať.
3. **StateMachine** - predstavuje celú štruktúru stavového automatu.
4. **StateMachineBuilder** - poskytuje programovateľné rozhranie na jednoduchú konštrukciu stavov a stavových prechodov.
5. **StateTransition** - predstavuje stavový prechod medzi dvomi stavmi.

6 Konfigurácia stavového stroja

Prehľadný zápis scenárov editačných operácii, ktoré prototyp podporuje.

6.1 Návrh

StateMachineInitializer obsahuje postupne definované operácie, ktoré umožnuje prototyp vykonávať používateľovi.

6.2 Implementácia

Konfigurácia je implementovaná v **StateMachineInitializer** kde sa zadefinujú všetky validné prechody medzi stavmi a implementácia ich prechodov. Každá sekvencia prechodov a akcií začína start(Názov sekvencie) a končí finish.
Medzi start a finish sa zadefinujú akcie, ktoré sa vykonávajú 6.

```
StateSequence
.start('CREATE_LIFELINE')
.button('sideLife')
.click((e: Event, h: CustomMesh[]) => {
    for (let obj of h) {
        if (obj.metadata.parent instanceof LayerView) {
            return true;
        }
    }
    return false;
},(e: Event, h: CustomMesh[]) => {

    // TODO refactor this

    for (let obj of h) {
        if (obj.metadata.parent instanceof LayerView) {
            let lifelineNew = new Lifeline('Standard name','',[], obj.metadata.parent.businessElement);
            obj.metadata.parent.businessElement.AddLifeline(lifelineNew);
            LayoutControl.magic(Globals.CURRENTLY_OPENED_DIAGRAM);
            // for (let child of GLContext.instance.scene.children) {
            //     GLContext.instance.scene.remove(child);
            // }
            // GLContext.instance.scene.add(Globals.CURRENTLY_OPENED_DIAGRAM.diagramView);
        }
    }
})
.finish(() => {});
```

Obr. 6: Ukážka definície platnej sekvencie prechodov.

7 Komponent: Model diagramu

7.1 Analýza

Úlohou tohoto modulu je definovať objekty používané pri vykresľovaní elementov diagramov.

7.2 Návrh

Vychádzajúc z definície schémy modelu sme identifikovali tieto objekty:

- Diagram
- Message
- OccurenceSPecification
- Layer
- Lifeline

7.3 Implementácia

Na implementáciu sme sa snažili napodobniť funkcionalitu backendu, pre najpresnejšie simulovanie správania.

1. **Diagram** - Obsahuje všetky objekty a referencie ktoré ktoré vytvárajú zobrazovaný diagram.
2. **Layer** - Objekt, v 3D diagrame, v ktorom sa nachádzajú lifeline-y a message-y.
3. **Lifeline** - Objekt reprezentujúci pomenovaný element, ktorý reprezentuje účastníka v interakcii.
4. **Message** - ktorý definuje špecifickú komunikáciu medzi lefline-ami.
5. **OcccurenceSpecification** - Objekt ktorý nám definuje začiatok alebo koniec medzi message a lifeline-om.

8 Komponent: Zobrazovač diagramu

8.1 Analýza

Úlohou tohto modulu je sprostredkovať vizuálnu reprezentáciu modelu diagramu, ktorá bude komponentom ThreeJS vykresľovaná na grafické plátno.

8.2 Návrh

8.2.1 Grafická reprezentácia objektov diagramu

Jednotlivé zobrazované objekty modelu sme sa rozhodli reprezentovať korešpondujúcimi "view" objektmi dediacimi z ThreeJS Object3D a spájať ich do objektovej hierarchie začínajúcej v koreňovom elemente DiagramView. Toto nám umožní jednoducho manipulovať ich pozície a rotácie v priestore. Zároveň do týchto objektov budeme agregovať jednotlivé mesh-e tvoriace vizuálnu reprezentáciu ktorá sa bude zobrazovať používateľom.

8.2.2 Rozloženie objektov

Z dôvodu komplexnosti riešenia, ktoré by upravovalo iba rozmiestnenie elementov, ktorých pozícia sa má zmeniť, sme sa rozhodli pri každej zmene ovplyvňujúcej rozmiestnenie prekalkulovať a aktualizovať pozície existujúcich objektov vo vizuálnej hierarchii a pridať do nej nové v prípade že daná zmena nové vizuálne objekty potrebuje.
Efektivitu tohto riešenia vylepšuje vlastnosť zoradenia v zoznamoch objektov v modeli diagramu a navrhnutá hierarchia objektov.

8.3 Implementácia

8.3.1 GraphicElement

Abstraktná trieda z ktorej dedia všetky view objekty tvoriace hierarchiu vizuálnych objektov.
Rozširujú ju triedy pre každý vizualizovaný objekt modelu: DiagramView, LayerView, LifelineView a MessageView. Všetky tieto grafické elementy sú ThreeJS funkciou "add"spájané do hierarchie.
Každý konkrétny graficky element implementuje vlastnú metódu updateLay-

out, ktorá okamžite aktualizuje vizuálnu reprezentáciu alebo začne animáciu ak sa daný objekt má animovať.

Konkrétne grafické elementy dedia z GraphicElement členy sprostredkujúce animáciu:

- shouldAnimate: vypovedá to tom či sa daný element začne animovať, alebo okamžite prekresli na správnom mieste pri volaní jeho funkcie layout.
- Update: funkcia volaná v GLContext na DiagramView ktorá sa propaguje po hierarchii a vyvoláva Animate na animovaných elementoch
- Animate: Abstraktná funkcia sprostredkujúca zmeny v elemente reprezentujúce animáciu.
- Animation: objekt pre uchovávanie začiatočného a koncového stavu animácie
- Animator: interpolačná funkcia pre výpočet súčasného stavu animácie

MessageView okrem všetkých metód grafického elementu tiež poskytuje metódy umožňujúce priame prekresľovanie obchádzajúce funkciu layout, používané zo stavového stroja na spätnú väzbu používateľovi pri niektorých akciách (napr. pri zmene správy potiahnutím jej konca):

- redraw: funkcia zabezpečujúca aktualizáciu elementu zavyslu iba na source a destination.
- redrawSimple: rýchlejšia varianta redraw vhodná iba na horizontálne šípky v 2D, ako pri rôznych verziách ťahania šípky za začiatok/koniec používateľom prostredníctvom redrawBySource a redrawByDestination.
- redrawBySource: obal metódy redrawSimple ktorý tiež zarovná súradnice y a z konca správy so súradnicami aktualizovaného začiatočného bodu.
- redrawByDestination: obal metódy redrawSimple ktorý tiež zarovná súradnice y a z začiatku správy so súradnicami aktualizovaného koncového bodu.

8.3.2 LayoutControl

Poskytuje funkciu layout, ktorá podaný objekt Diagram prechádza po objektoch Layer a tie objektoch Lifeline a Message. Pre každý prejdený objekt je na jemu korešpondujúcom "view" objekte volana metóda updateLayout. Objektom ktoré nemajú korešpondujúci "view" objekt je jeden najskôr vytvorený. Novo vytvorene "view" objekty sú tiež pridane do hierarchie ThreeJS funkciou add.

8.3.3 Text3D

Sprostredkuje 3D obojstranne zobrazenie textu. Viazaný na grafické elementy s textom(MessageView a LifelineView). Jeho update funkcia musí byt volaná v rámci updatelayout jeho korešpondujúceho view objektu.

8.3.4 CustomMash

Rozšírenie triedy Mesh z ThreeJS o get metódu ViewObject ktorá vracia grafický element ktorý ma daný CustomMash pod sebou v hierarchii, čo uľahčuje prácu s raycasting-om.

9 Backend

9.1 Analýza

9.1.1 ASP.NET Core Webserver

Úlohou modulu webservera je zabezpečovať permanentnosť dát (diagramov), čiže spravuje pridávanie a mazanie diagramov v databáze, úpravu vytvorených diagramov a opätovné načítanie diagramov pri zapnutí klienta.

9.1.2 MongoDB Databáza

Úlohou tohto modulu je rýchly prenos diagramu do databázy. Je nutné porovnať vlastnosti relačných a nerelačných databáz a následne vybrať vhodnú databázu. Diagramy, ktoré budú vkladané do databázy, budú prichádzať v JSON formáte. Vhodným riešením je teda nerelačná MongoDB databáza. Databáza poskytne bezpečné uloženie diagramu a následný rýchly prístup ku nemu.

9.2 Návrh

Komunikácia klient-server bude asynchrónna, čiže bude zabezpečená lepšia odozva na klientovi. Server bude poskytovať služby na získavanie diagramov, ich pridávanie, modifikáciu a mazanie z databázy.

9.3 Implementácia

Pri získavaní dát z pohľadu frontendu server komunikuje s databázou, z ktorej vytiahne požadované dáta, ako objekty reprezentujúce diagram.
Tieto objekty sú pred odoslaním na frontend serializované do JSON formátu, v ktorom ich frontend dokáže prijať.
V prípade ukladania dát, čiže uloženia diagramu, server prijme požiadavku na uloženie dát (diagramu) do databázy.
V tele požiadavky sa nachádza JSON obsahujúci reprezentáciu diagramu a je potrebné tento JSON deserializovať do objektov zodpovedajúcich dátovému modelu diagramu a následne ho uložiť do databázy pomocou knižničnej funkcie na vkladanie JSON dát do MongoDB.

Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Tím 25 - Bystander Effect

3D-UML Improved

Projektová dokumentácia - používateľská príručka

Členovia tímu
Bc. Štefan Motko
Bc. Ronald Demeter
Bc. Marcel Furucz
Bc. Patrik Sčensný
Bc. Martin Dieška
Bc. Peter Psota
Bc. Martin Gembec

Vedúci
doc. Ing. Ivan Polášek, PhD.

Akademický rok: 2017/2018

Obsah

1 Inštalačný manuál **2**

2 Používateľská príručka **2**

2.1 Pridávanie čiary života 2
2.2 Odstránenie čiary života 4
2.3 Posunutie čiary života 4
2.4 Pridávanie správ medzi čiarami života 4
2.5 Odstránenie správy medzi čiarami života 6
2.6 Pridanie vrstvy 6
2.7 Odstránenie vrstvy 6
2.8 Pohyb v 3D priestore 7
2.9 Ukladanie perspektívu pohľadu 7
2.10 Návrat na základnú perspektívu 7

1 Inštalačný manuál

Na inštaláciu projektu je potrebné nainštalovať nasledujúce technológie.

- ASP.Net core 2.0
- MongoDB 3.4
- Node.js

Toto sa vykoná pomozou NodeJS príkazu "npm install", ktorý dané technológie nainštaluje do priečinka Node_Modules.

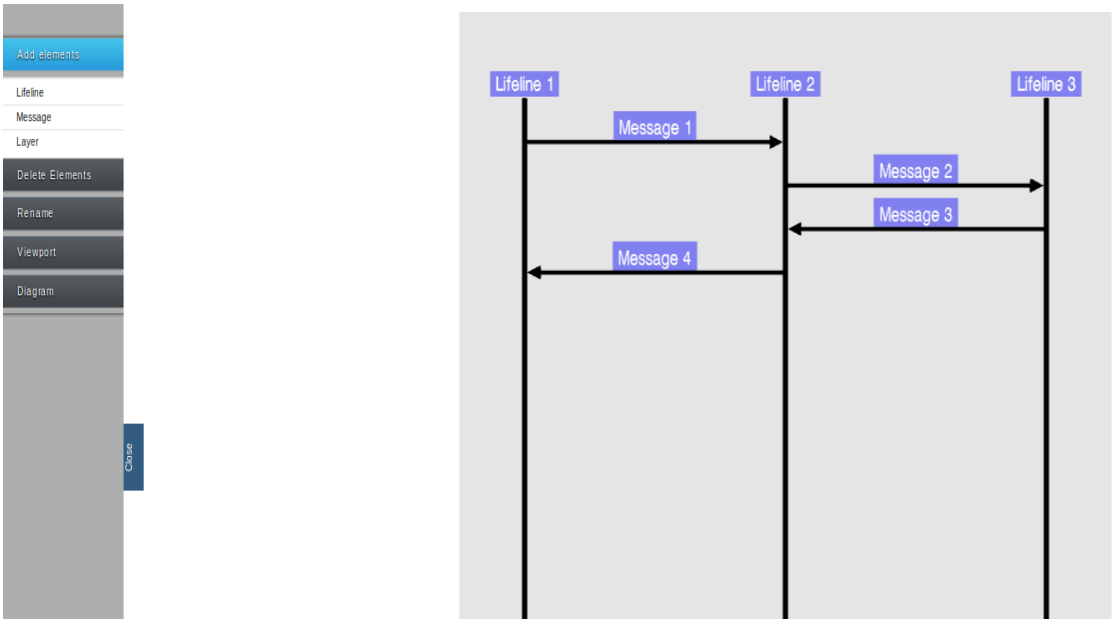
Prototyp sa spúšťa použitím príkazu "./start.sh" v priečinku "TP_webApp/TP_webApp". Skript sa pokúsi spolu s protypom spustiť MongoDB server. Toto sa vykoná ale len v prípade, že je nastavená premenná prostredia MONGOD_DB_PATH(cesta k adresáru úložiska údajov pre databázu). Inak sa použije predvolené nastavenie.

2 Používateľská príručka

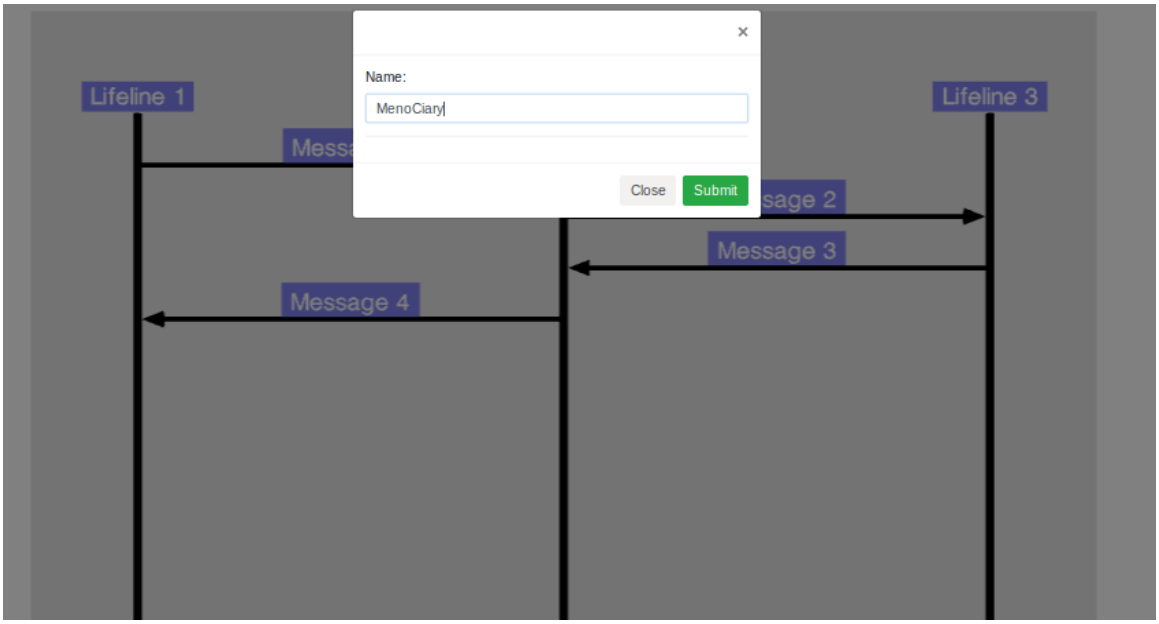
2.1 Pridávanie čiary života

Každý používateľ je schopný pridávať čiary života[1]. Pre ich vytvorenie musí používateľ ale najprv otvoriť bočný navigačný panel. Vo vnútri panelu rozklikne možnosť "Add elements" a klikne na možnosť "Lifeline" 2 a následne klikne na vrstvu, kam sa pridá čiara života. Následne sa zobrazí vyskakovacie okno[2] v ktorom používateľ napíše meno čiary života. Čiaru života je možné vytvoriť len na vrstve.

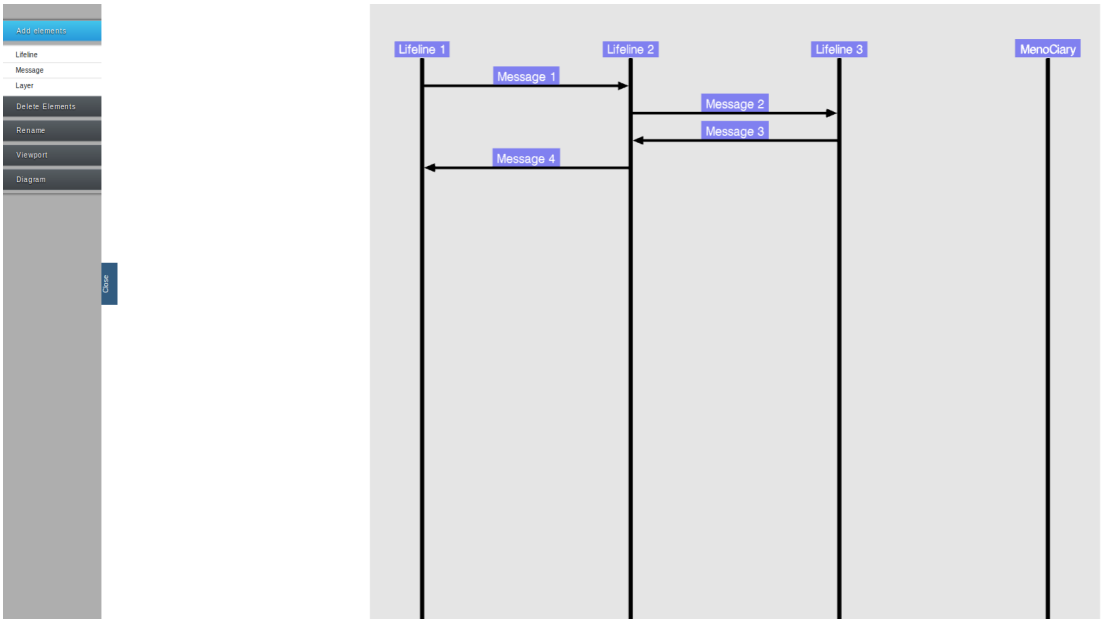
[1] z angl.(lifeline)
[2] z angl.(pop-up)



Obr. 1: Otvorene bočne menu a rozkliknutá možnosť s pridaním elementov.



Obr. 2: Napísanie mena čiary života.



Obr. 3: Diagram po pridaní čiary života.

2.2 Odstránenie čiary života

Odstránenie čiary života prebieha podobne ako pridávanie. Používateľ otvorí bočné menu, rozklikne možnosť "Delete Elements" klikne na tlačidlo "Lifeline". V prípade, že na čiaru života sú napojené správy, či vychádzajúce alebo vstupujúce, všetky sa vymažú spolu s čiarou života.

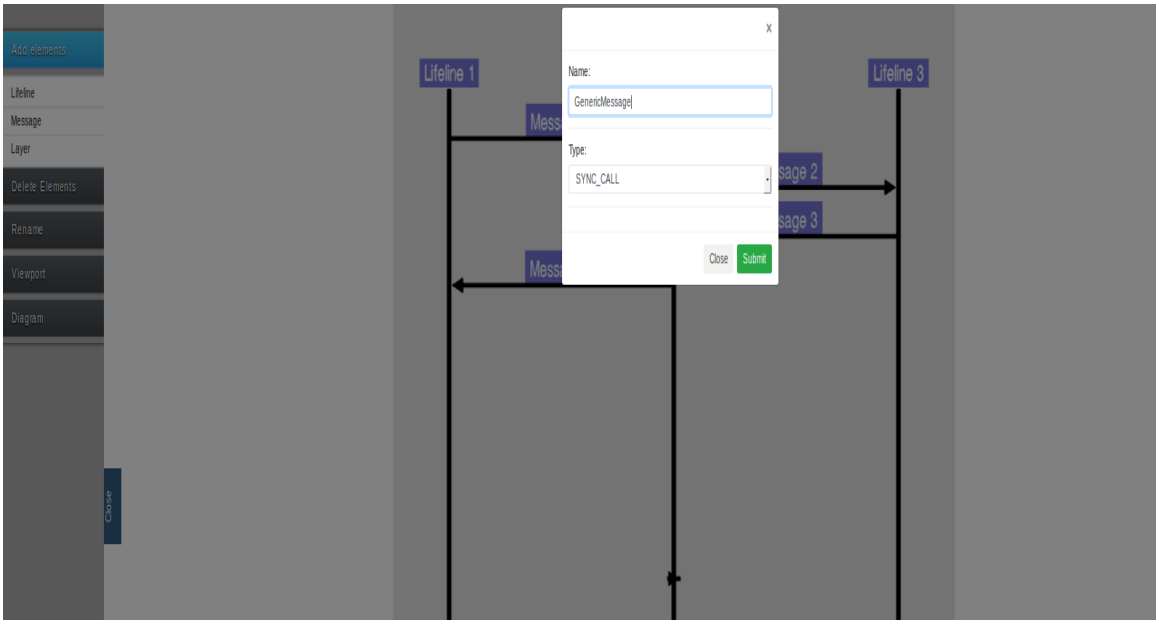
2.3 Posunutie čiary života

Ak sa nám nepáči poradie v ktorom sú usporiadané čiary života môžeme ich jednoducho posunúť. Stačí kliknúť a podržať ľavé tlačidlo myši nad zvolenou čiarou a posúvať v horiznotálnom smere.

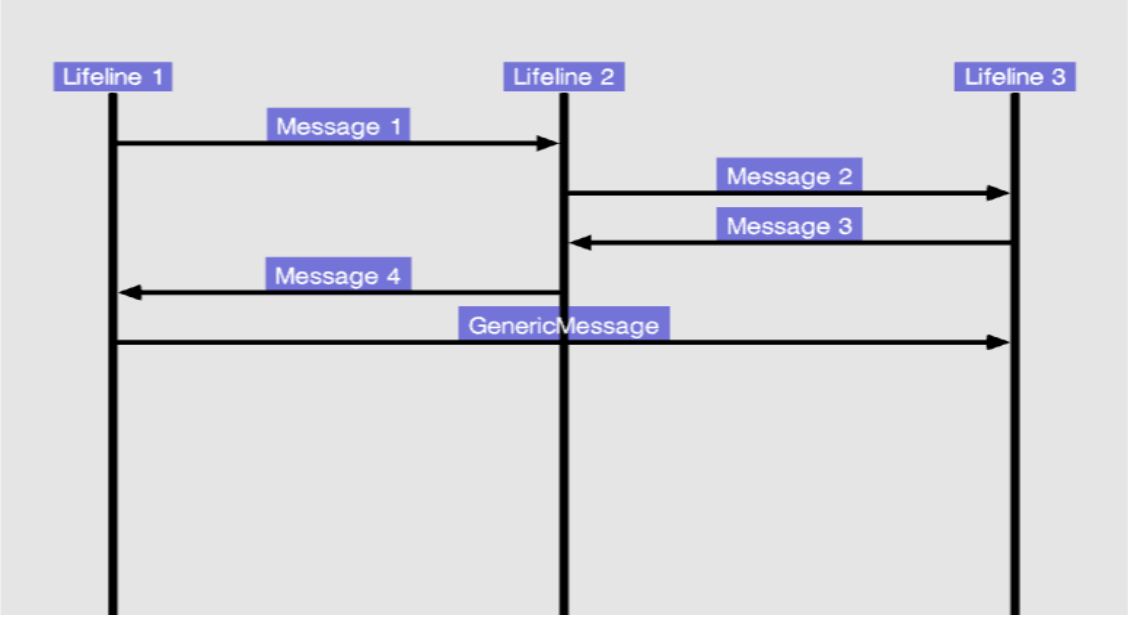
2.4 Pridávanie správ medzi čiarami života

Na pridávanie správ medzi čiarami života sa najprv musí v bočnom paneli vybrať možnosť "Add element" a následne možnosť "Message". Po vybratí možnosti sa kliknutím vyberie začiatočná čiara života a potiahnutím

a uvoľnením koncová čiara života. Po tejto akcii sa nám zobrazí okno kde napíšeme názov správy a vyberieme typ správy.



Obr. 4: Okno na zadanie názvu správy a typ správy



Obr. 5: Diagram po pridaní čiary života

2.5 Odstránenie správy medzi čiarami života

Pre odstránenie správy sa vyberie z bočného menu rozklikneme "Delete Element" možnosť "Message". Po vybratí možnosti používateľ klikne na správu, ktorú chce vymazať. Táto správa sa následne odstráni.

2.6 Pridanie vrstvy

Pre pridanie vrstvy sa v bočnom menu rozklikne možnosť "Add element" a následne klikneme na tlačítko "Layer". Po tejto akcii sa na plátne zobrazí nová vrstva na ktorú môžeme pridávať ďalšie čiary života a správy medzi nimi.

2.7 Odstránenie vrstvy

V bo4nom menu rozklikneme "Delete Element" a následne klikneme na tlačítko "Layer". Po tejto akcii klikneme na vrstvu ktorú chceme vymazať a táto vrstva bude následne vymazaná aj zo všetkými prvkami nachádzajúcimi sa v nej.

2.8 Pohyb v 3D priestore

Aplikácia dovoľuje používateľovi voľný pohyb v 3D priestore. Použitím kláves WASD sa používateľ môže pohybovať v štyroch smeroch po rovine.

- W - dopredu
- S - dozadu
- A - doľava
- D - doprava

Pomocou kláves Q,E sa používateľ môže pohybovať hore a dole.

- Q - hore
- E - dole

Podržaním pravého tlačidla myšky, používateľ môže pohybovať kamerou. Všetky pohyby tlačidlami sú orientované podľa myšky.

2.9 Ukladanie perspektívu pohľadu

Používateľ má možnosť ukladať viaceré perspektívy pohľadu. Používateľ má možnosť sa na ne hocikedy vrátiť. Na uloženie perspektívy treba otvoriť bočný panel a kliknúť na tlačidlo "saveViewpoint". Po uložení sa používateľovi v bočnom menu vytvoria dve tlačidla. Tlačidlo "viewpoint N"(kde N je číslo perspektívy) vráti kameru späť na konkrétny perspektívny pohľad.

2.10 Návrat na základnú perspektívu

Pomocou tlačidla "resetViewpont"v bočnom menu sa používateľ vráti na základnú perspektívu diagramu.