Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Projektová dokumentácia

Tím 13 - Chaos

Vedúci tímu: Ing. Juraj Vincúr
Členovia tímu: Bc. Ivan Andrejkovič
Bc. Tadeáš Broniš
Bc. Gabriela Hózová
Bc. Nikolas Janec
Bc. Jaroslav Lišiak
Bc. Tomáš Ofčarovič
Bc. Michal Škuta

Názov témy: Pohlcujúci web [iWeb]

Akademický rok: 2017/2018 Dátum: 13.11.2017

Obsah

Dokumentácia riadenia

1 Úvod

Dokumentácia vznikla v rámci predmetu Tímový projekt a opisuje prácu na projekte Pohlcujúci web. Obsahom tejto dokumentácie je predstavenie členov tímu, podrobný opis manažmentu, zhrnutie jednotlivých šprintov a súhrn metodík. Špecifikácia vytváraného systému je opísaná v dokumentácii inžinierskeho diela.

Opis kapitol:

Kapitola 2 obsahuje predstavenie jednotlivých členov tímu, sú v nej definované zodpovednosti každého člena tímu a tiež podiel práce na tomto dokumente. Kapitola 3 bližšie špecifikuje procesy manažmentu projektu. Kapitola 4 sumarizuje priebeh a výsledky všetkých troch šprintov vykonaných v priebehu zimného semestra. Poskytuje tiež zhodnotenie príslušného šprintu. Kapitola 5 je venovaná súhrnu metodík definovaných a využívaných v tíme. Táto kapitola sa odkazuje na prílohu A v ktorej sa nachádza podrobný opis použitých metodík. Kapitola 6 obsahuje globálnu retrospektívu práce v zimnom semestri. V retrospektíve je stručne zhrnuté v čom chceme pokračovať na projekte a s čím chceme skončiť.

2 Členovia tímu a ich roly

Táto kapitola sa zaoberá členmi tímu a ich úlohami na tomto projekte.

2.1 Predstavenie členov tímu

Bc. Ivan Andrejkovič

Študent druhého stupňa vysokoškolského štúdia na Fakulte informatiky a informačných technológií STU v BA. Dlhodobo aktívne programuje v c a venuje sa etickému hackingu a penetračnému testovaniu.

Bc. Tadeáš Broniš

Absolvent bakalárskeho štúdia na Fakulte Informatiky a Informačných technológii STU. Vo svojej bakalárskej práci sa venoval prehľadávania územia pomocou jedného, či viacerých dronov. Počas vyhotovenia tejto práce sa oboznámil s niekoľkými vyhľadávacími algoritmami a naučil sa programovať v jazyku Python. Počas bakalárskeho štúdia sa naučil programovať v jazykoch JavaScript, Java, C#. Momentalne je študentom prvého ročníka inžinierskeho štúdia v obore Internetové Technológie na FIIT STU.

Bc. Gabriela Hózová

Študentka prvého ročníka inžinierskeho štúdia v odbore Inteligentné softvérové systémy na FIIT STU. Na tejto fakulte absolvovala aj bakalársky stupeň v odbore Informatika. Jej bakalárskou prácou bolo vytvorenie systému na prípravu vzoriek pre porovnávanie podobností. Počas štúdia na fakulte získala skúsenosti v programovaní v jazykoch C, Java, Python, SQL. Téma 3D virtuálnej prehliadky ju zaujala z dôvodu čoraz väčšej prístupnosti tejto technológie bežnému používateľovi.

Bc. Nikolas Janec

Bakalárske štúdium absolvoval na FIIT STU v odbore Internetové technológie. Jeho bakalárskou prácou bolo vytvorenie serverovej aplikácie pre autentifikáciu používateľov a následnú správu ich prístupových práv vo viacerých objektov. Počas štúdia na fakulte získal skúseností s programovaním v PHP, Java, C#, C++, C.

Bc. Jaroslav Lišiak

Bakalárske štúdium ukončil na FIIT STU. Záverečná práca bola zameraná na návrh a implementáciu vlastného prototypu IoT zariadenia, komunikujúceho prostredníctvom technológie LoRa. Bol spoluautorom dvoch komunikačných protokolov, ktoré boli použite v

bakalárskej práci. Navrhnuté protokoly boli prezentované na konferencii IIT.SRC 2017. Najviac skúseností má s programovacími jazykmi C#, Java, a C.

Bc. Tomáš Ofčarovič

Študuje v prvom ročníku inžinierskeho štúdia vzdelávacieho programu Internetové technológie na FIIT STU. Bakalársku prácu spracoval na tému „Pokročilé monitorovanie sietí v reálnej prevádzke“, kde sa systém na monitorovanie prevádzky siete stále používa do. Počas školy nadobudol mnoho zaujímavých poznatkov o programovaní v jazykoch C, JAVA, C#, JavaScript a SQL. Ďalšími poznatkami sú sieťové technológie a ich implementácia, v ktorých sa zdokonaľuje účasťou na certifikovaných kurzoch.

Bc. Michal Škuta

Študent inžinierskeho štúdia na Fakulte Informatiky a Informačných Technológií Slovenskej technickej univerzity v Bratislave, ktorý sa amatérsky zaujíma o prácu s openGL. Popri škole sa zoznámil s množstvom sieťových technológií potrebných na fungovanie internetu.

2.2 Rozdelenie rolí v tíme

Člen tímu	Dlhdobá úloha	Krátkodobá úloha
Bc. Ivan Andrejkovič *(manažér testovania)*	Vývojár Tester	Analýza problémovej oblasti Zapisovateľ Informovanie o aktuálnom stave projektu
Bc. Tadeáš Broniš *(manažér kvality)*	Správca architektúry projektu	Testovanie nástrojov potrebných na prácu Zapisovateľ Tvorca retrospektív
Bc. Gabriela Hózová *(manažér komunikácie)*	Dohliadanie na komunikáciu v tíme, dodržiavanie úloh a zodpovedností Správca projektovej dokumentácie	Tvorca retrospektív Zapisovateľ Analýza problémovej oblasti Scrum master
Bc. Nikolas Janec *(manažér tvorby dokumentácie)*	Správca fotografických podkladov pre tvorbu 3D modelov	Zapisovateľ Analýza problémovej oblasti Informovanie o aktuálnom stave projektu
Bc. Jaroslav Lišiak *(manažér úloh)*	Administrátor tímového webu (serveru)	Dohliadanie na dodržiavanie termínov Zapisovateľ Tvorca web stránky
Bc. Tomáš Ofčarovič *(manažér plánovania)*	Dohľad nad testovaním	Tvorca retrospektív Tvorca web stránky Zapisovateľ
Bc. Michal Škuta *(manažér vývoja)*	Správca repozitára Vývojár Tester	Analýza problémovej oblasti Zapisovateľ Informovanie o aktuálnom stave projektu

2.3 Podiel práce na jednotlivých častiach dokumentácie

Bc. Ivan Andrejkovič

- Dokumentácia inžinierskeho diela: analýza, návrh, implementácia

Bc. Tadeáš Broniš

- Dokumentácia inžinierskeho diela: návrh, testovanie
- Metodiky

Bc. Gabriela Hózová

- Dokumentácia riadenia: Aplikácia manažmentov, sumarizácia šprintov, globálna retrospektíva
- Dokumentácia inžinierskeho diela: analýza
- Metodiky

Bc. Nikolas Janec

- Dokumentácia riadenia: úvod
- Dokumentácia inžinierskeho diela: úvod, analýza

Bc. Jaroslav Lišiak

- Dokumentácia inžinierskeho diela: testovanie
- Metodiky

Bc. Tomáš Ofčarovič

- Dokumentácia riadenia: metodiky
- Dokumentácia inžinierskeho diela: analýza
- Metodiky

Bc. Michal Škuta

- Dokumentácia inžinierskeho diela: analýza, implementácia
- Metodiky

3 Aplikácie manažmentov

3.1 Manažment dokumentácie

V rámci manažmentu dokumentácie vytvárame jednotlivé dokumenty. Priebežne vytvárame zápisnicu z každého tímového stretnutia, retrospektívu po každom šprinte a dokumentáciu zdrojového kódu. Tieto dokumenty sa buď priebežne menia, alebo sa priebežne vytvárajú nové verzie podľa potreby. Ďalšie vytvárané dokumenty sú napríklad tímové metodiky, dokumentácia riadenia a dokumentácia inžinierskeho diela. Každý z vytváraných dokumentov v rámci tímového projektu má zadefinovanú šablónu, podľa ktorej sa tento dokument vytvára. Šablóny sme si zadefinovali v metodike dokumentácie.

3.2 Manažment komunikácie

Komunikáciu môžeme rozdeliť na komunikáciu na stretnutí a komunikáciu mimo stretnutí. V rámci komunikácie na stretnutí má každý člen tímu priestor povedať o svojich výsledkoch a problémoch, ktoré nastali, a ak je to nutné, otvoríme priestor pre hlbšiu diskusiu o vzniknutých problémoch. Komunikácia mimo stretnutia funguje hlavne pomocou nástroja Slack. Pre rôzne účely sme vytvorili viacero kanálov podľa druhu témy, ktorej sa daná komunikácia týka. Komunikácia s produktovým vlastníkom projektu je taktiež veľmi dôležitá. Produktový vlastník tohto projektu je zároveň vedúci nášho tímu, a tak s ním komunikujeme taktiež najmä pomocou nástroja Slack. Vedúceho taktiež informujeme o aktuálnom stave projektu a úloh na pravidelných týždenných tímových stretnutiach.

3.3 Manažment úloh

V rámci manažmentu úloh je vytvorená metodika manažmentu úloh, ktorá opisuje postupy na správu úloh v nástroji GitLab. Je potrebné mať určené ako vytvárať, priraďovať a dokončovať úlohy. Úlohy sa vytvárajú na základe konzultácii na začiatku šprintu ale aj počas šprintu. Metódou “planning poker” určujeme náročnosť úloh. Úlohy sú priraďované členom tímu počas plánovania šprintu a vykonávajú sa podľa priority. Priorita úloh sa určuje podľa ich uloženia v “To Do” sekcii. Manažér úloh je zodpovedný za správu všetkých úloh v projekte v spomínanom nástroji GitLab.

3.4 Manažment testovania

Testovanie v našom projekte prebieha podľa testovacích scenárov opísaných v metodike testovania. Testovanie je zabezpečené pomocou HMD a ovládačov. Projekt je konkrétne testovaný pomocou HTC Vive. Pri testovaní sa zameriavame na funkčnosť projektu podľa

požiadaviek produktového vlastníka ale aj na vizuálny výstup. Pri testovaní je prítomný autor testovanej časti a nezávislý tester. Testovanie vizuálneho výstupu v sebe zahŕňa napríklad testovanie plynulosti zobrazovania, prekrývania objektov a podobne.

3.5 Manažment vývoja

V našom projekte pracujeme paralelne na dvoch spôsoboch riešenia. Tým, že máme dva možné spôsoby ako problém vyriešiť, rozhodli sme sa pracovať na dvoch prototypoch. Počas postupného vyvíjania oboch prototypov zistíme, ktorý je viac vhodný pre výsledný produkt. Vývoj na projekte je kontrolovaný manažérom vývoja, ktorý sa stará o to, aby boli dodržiavané pravidlá z metodiky písania kódu a aby boli vyvíjané časti včas hotové.

3.6 Manažment kvality

Na zabezpečenie dostatočnej kvality jednotlivých úloh máme zadefinované rôzne metodiky, ako napr. metodiku revidovania, metodiku verziovania, metodiku písania kódu a taktiež tímový “definition of done”.

“Definition of done” pre používateľský príbeh (user story):

Súbor úloh (story) spĺňa tímovú “definition of done” vtedy, keď:

1. Prejde testami, ktoré boli pre dané úlohy vytvorené
2. Kód bol zrevidovaný aspoň jedným ďalším členom tímu, ktorý sa nepodieľal na tvorbe kódu
3. Spĺňat požiadavky používateľského príbehu (akceptačné kritériá od produktového vlastníka)

“Definition of done” pre úlohy:

1. Kód implementovaný pre danú úlohu musel byť zrevidovaný
2. Ak sú pre túto úlohu vytvorené testy, kód musí byť taktiež otestovaný

3.7 Manažment plánovania

Na začiatku semestra boli vytvorené základné používateľské príbehy produktu aj s popisom, teda minimálne požiadavky, ktoré by mali byť splnené. Na stretnutí na začiatku šprintu vždy podľa priority používateľských príbehov zoberieme ich časť, pre ktorú sa zaviažeme, že ju stihneme spraviť. Tieto používateľské príbehy následne rozdelíme na menšie úlohy, ktoré obsahujú všetko to, čo je potrebné splniť pre úspešné vykonanie používateľského príbehu. Jednotlivé úlohy sa pridelia členom tímu tak, aby každý člen tímu mal určenú prácu na šprint a taktiež tak, aby sme vedeli, že má predstavu o tom, čo bude robiť. Ku každej úlohe je

uvedený jej názov, popis (description) a odhad (v počte story pointov), koľko bude táto úloha trvať. Pred tým, ako sa začne robiť na úlohe, musí mať úloha jasne určené, kto na nej bude pracovať. Po uplynutí prvého týždňa šprintu sa stretávame na tímovom stretnutí, kde si navzájom prezentujeme pokroky v našich prácach na projekte alebo riešime vzniknuté problémy. Ak je to potrebné, vytvoríme ďalšie úlohy, ktoré sme pri prvotnom plánovaní nevideli.

Úlohy, ktoré boli členom tímu zadané postupne prechádzajú určitými stavmi, pričom dodržiavame pravidlá o zmene stavu úlohy určené v metodike manažmentu úloh. Na začiatku má úloha stav “To Do”, po ňom nasleduje “Doing”, “Review”, “Testing” a následne “Closed”. Sledovaný je taktiež aj stav šprintu. Tento stav je reprezentovaný burndown grafom. Každý šprint má svoj vlastný burndown graf, na ktorom sledujeme, koľko percent zo šprintu je hotových. Po každom šprinte na stretnutí tím prezentuje produktovému vlastníkovi nový prírastok k produktu a aj všetky splnené a nesplnené úlohy. Všetky informácie týkajúce sa plánovania a stavov úloh sú uložené v nástroji GitLab.

4 Sumarizácie šprintov

Kapitola obsahuje sumarizáciu jednotlivých šprintov. Každý šprint je po jeho ukončení zhodnotený celým tímom na stretnutí. Retrospektíva šprintu je následne zapísaná a uverejnená na webovej stránke tímu.

4.1 Šprint 1 - Scooby-Doo

Pri prvom tímovom stretnutí sme sa lepšie zoznámili medzi sebou v tíme, s vedúcim tímu a v neposlednom rade s našou témou. Pri diskutovaní o téme a o predstavách a požiadavkách zákazníka (vedúci tímu) sme si určili základné úlohy, ktoré sme sa zaviazali v prvom šprinte splniť. Všetky úlohy sa nám nepodarilo dokončiť včas, takže sme boli nútení niektoré z nich presunúť do nasledujúceho šprintu. Cieľom tohto šprintu bolo najmä zanalyzovať oblasť, do ktorej naša téma zapadá. Oboznámili sme sa tiež s nástrojmi využívanými pri práci na projekte, pri manažmente a tiež pri komunikácii v tíme. Vytvorili sme a nasadili webovú stránku tímového projektu. Každý z nás ako jednotlivec zhodnotil tento prvý šprint a následne sme spravili spoločnú retrospektívu, kde sme navrhli, čo by bolo vhodné v budúcich šprintoch vylepšiť alebo prestať robiť. Spolu s nedokončenými úlohami sme vytvorili úlohy na ďalší šprint a tie si medzi sebou rozdelili. Najväčší problém, ktorý sme identifikovali, bol problém v komunikácii medzi členmi tímu.

4.2 Šprint 2 - Lajka

Počas druhého šprintu sme pokračovali v hlbšej analýze oblasti a začali sme pracovať na prototype webovej aplikácie. Pri tvorbe prototypu sme narazili na niekoľko problémov týkajúcich sa zhotovovania 3D modelov objektov, ktoré sa nachádzajú v 3D labe. Nástroje, ktoré sme plánovali použiť na vytvorenie 3D modelov sa dajú zaradiť do dvoch kategórií. V tomto šprint sme zistili, že výsledky nástrojov na skenovanie objektov nevyzerajú podľa našich predstáv a rozhodli sme sa v nasledujúcom šprinte použiť namiesto toho techniku fotogrametrie. Začalo sa taktiež pracovať na vytváraní jednotlivých metodík, podľa ktorých sa budeme riadiť počas celého vývoja. Počas retrospektívy sme opäť prediskutovali plusy a mínusy uplynulého šprintu. Problém s komunikáciou, ktorý sa v minulom šprinte objavil, sa nám v tomto šprinte podarilo eliminovať a komunikácia sa citeľne zlepšila.

4.3 Šprint 3 - Bobi

V priebehu tretieho šprintu pokračovali práce na prvom prototype z minulého šprintu a taktiež bol vytvorený druhý prototyp. Začali sme pracovať na vizualizácii priestoru pomocou 360° fotografie. Doterajšie práce boli snahou o vymodelovanie celého priestoru pomocou 3D

objektov. Cieľom tretieho šprintu bolo taktiež aj dokončiť všetky metodiky. Všetky úlohy sa nám nepodarilo dokončiť včas, nakoľko sa nám skomplikovala situácia odchodom jedného člena tímu. Boli sme teda nútení niektoré z nich presunúť do nasledujúceho šprintu.

5 Používané metodiky

Na bezproblémovú kolaboratívnu prácu na projekte, bolo nutné zaviesť určité pravidlá. Preto sme sa riadili niekoľkými metodikami.

V projekte sme zadefinovali nasledovné metodiky:

- Metodika dokumentácie
- Metodika písania kódu
- Metodika komunikácie
- Metodika revízií
- Metodika verziovania
- Metodika testovania
- Metodika manažmentu úloh

Táto kapitola sa venuje stručným opisom použitých metodík. Plné znenie príslušnej metodiky sa nachádza v prílohe A tohto dokumentu.

5.1 Metodika dokumentácie

V metodike dokumentácie sú definované konvencie písania dokumentu, rozsah dokumentu, forma a štýly pre jednotlivé dokumenty. Okrem dokumentácie riadenia a dokumentácie inžinierskeho diela sa spracovávajú po každom stretnutí zápisnice a po ukončení špritnu sa spíše retrospektíva špritu. Po spísaní dokumentu je posunutý ostatným členom tímu pre zhodnotenie, odsúhlasenie, prípadne upravenie do finálnej podoby. Tieto dokumenty sú potom vložené na webové sídlo nášho tímu. Dokumentácia zdrojového kódu je riešená formou komentárov v zdrojovom kóde pre jednoduché pochopenie pribudnutého kódu.

5.2 Metodika písania kódu

Súčasťou tejto metodiky je konvencia písania kódu. Definovali sme, akým spôsobom budeme písať názvy premenných, formátovanie blokov zdrojového kódu ale taktiež aj štruktúru súborov a adresárov v repozitári na našom serveri na GitLabe. Najdôležitejšou súčasťou zdrojového kódu sú komentáre, ktoré napomáhajú ostatným ľudom pomôcť rýchlo pochopiť upravený kód. Metodika je záväzná pre všetky zdrojové kódy, ktoré vzniknú počas prác na

projekte a platí pre všetkých členov, ktorí sú súčasťou tímu. Určili sme programovací jazyk, ktorý sa bude v rámci tímového projektu využívať, a je ním Javascript.

5.3 Metodika komunikácie

Metodika komunikácie má za cieľ definovanie procesov komunikácie, ktoré v rámci tímového projektu uplatňujeme. Medzi hlavné nástroje na komunikáciu v tíme je nástroj na komunikáciu Slack s mnohými komunikačnými kanálmi, v ktorých sa riešia príslušné témy. Slack používame aj na formálnu aj na neformálnu komunikáciu. Ďalším nástrojom na komunikáciu v tíme je náš nástroj na manažment projektu Gitlab. V Gitlabe komunikujeme pomocou komentárov na daný problém (Issue), alebo vybratím úlohy zo zoznamu úloh “na spracovanie” a presunutím do zoznamu “práve spracovávané”. Podrobný popis práce s nástrojmi je popísaný v prílohe. Na instantnú komunikáciu je možné zvoliť aj telefonát, nakoľko každý zverejnil svoje telefónne číslo, na ktorom je dosiahnuteľný v prípade urgentnej potreby.

5.4 Metodika revízií

Metodiku revízií sme rozdelili do dvoch častí, a to:

- revízia dokumentácie
- revízia zdrojového kódu

Zdrojový kód musí pred spojením s hlavnou vetvou zrevidovaný. Ak boli vyžadované zmeny, je potrebné zdrojový kód upraviť podľa požiadaviek až do podoby, pokým zdrojový kód kontrolór neschváli. Dodržiavanie jednotlivých pravidiel písania kódu aj dokumentácie zvyšuje celkovú kvalitu a kód bude prehľadnejší a ľahší na pochopenie, preto je revízia dokumentu aj kódu dôležitá.

5.5 Metodika verziovania

Účelom tejto časti je prezentovanie pravidiel, ktoré musia byť dodržiavané pri verziovaní zdrojového kódu v projekte. Na manažment verzií využívame systém Git. V tejto metodike je popísané, ako postupovať od pridelenia novej úlohy, cez odovzávanie, požiadanie o predkladania príspevkov, až po spojenie novej vetvy pre danú úlohu s vyššou vetvou. Je tu dôkladne opísaný postup, ktorým musia autori kódu prejsť, aby vytvorili nový predklad príspevku.

5.6 Metodika testovania

Táto metodika určuje spôsob akým je nutné testovať funkcionalitu pred tým, ako sa zmena v projekte schváli a integruje sa do vyššej vetvy. Uviedli sme kto bude zodpovedný za test časti kódu, kedy bude testovať kód, a akým spôsobom bude vykonávať testy. Táto metodika sa dodržiava počas celého vývoja projektu, a platí pre každého testera.

5.7 Metodika manažmentu úloh

Táto časť sa venuje opisu metodiky manažmentu úloh v rámci práce na tímovom projekte. Ide o úlohy pri tvorbe backlogu, šprintov, úloh, príbehu, ale aj odstraňovaní a úprave úloh v nástroji Gitlab. Metodika je určená primárne pre manažéra úloh, ale slúži všetkým členom tímu.

6 Globálna retrospektíva

Aplikovaná metóda vývoja nášho projektu bola Scrum. Poctivo sme dodržiavali pravidlá a zásady počas všetkých troch uplynulých šprintoch. Každý z týchto troch šprintov trval 2 týždne. Na začiatku každého šprintu sme absolvovali stretnutie s vlastníkom produktu (vedúci tímu) a určili sme ciele na celý šprint. Pomocou týchto cieľov sme identifikovali úlohy, ktoré sme museli vypracovať aby boli ciele splnené. Úlohy sme si napokon rozdelili medzi všetkých členov spravodlivo. Úlohy sme rozdeľovali na základe preferencií a schopností členov tímu. nestalo sa, že by niekto protestoval s pridelenou úlohou. Po uplynutí šprintu prebehla retrospektíva, na ktorej sme zhodnotili celý šprint najprv jednotlivo každý člen a následne aj spoločne ako celý tím.

Prvé dva šprinty sme venovali najmä analýze pracovnej oblasti, do ktorej zapadá náš projekt, a to konkrétne virtuálna realita, technika fotogrametrie, nástroje na tvorbu 3D modelov a už spomínanú fotogrametriu a taktiež sme do analýzy zhrnuli možnosti implementácie. Pri plánovaní druhého šprintu sme definovali nové požiadavky na MVP pre zimný semester. V polovici druhého šprintu sme začali získané informácie aplikovať aj prakticky a začali sme vytvárať prototyp výstupného produktu. V treťom šprinte sme sa analýze venovali už iba minimálne a viac sme sa zamerali už na samotnú implementáciu v dvoch rôznych prototypoch. Počas práce na projekte sme zaviedli metodiky, ktoré sme sa snažili dodržiavať.

Čo chceme robiť do budúcna:

- dôkladné dokumentovanie zdrojového kódu
- včasné plánovanie šprintov
- dôkladné dodržiavanie metodík
- aplikovať párové programovanie
- stretávať sa aj mimo oficiálnych stretnutí
- konzultovanie pomocou spoločného hovoru ak nie je možné osobne sa stretávať

V čom chceme pokračovať:

- rozširovanie metodík
- komunikácia cez nástroj Slack
- rozdeľovanie úloh podľa preferencií členov tímu

S čím chceme skončiť:

- komunikačný šum a rozptýlenie pri oficiálnych tímových stretnutiach

Dokumentácia inžinierskeho diela

7 Úvod

Obsahom tejto dokumentácie je pohľad na technickú stránku projektu vypracovaného v rámci predmetu Tímový projekt na tému Pohlcujúci web. Cieľom projektu je navrhnúť a vytvoriť nový druh webovej stránky, ktorá umožní prehľadávať informačný priestor v pohlcujúcom prostredí virtuálnej reality. Stránka bude slúžiť na prezentáciu spomínaného 3D Labu, jeho vybavenia a projektov realizovaných vďaka nemu. Kapitola 8 opisuje ciele, ktoré sme si stanovili pre zimný semester. Obsahuje tiež požiadavky na systém a globálne ciele. Základným cieľom pre zimný semester je vytvorenie MVP prototypu aplikácie. Obsahom kapitoly 9 je analýza projektu. V jej časti sa venujeme analýze virtuálnej reality, fotogrametrie pre tvorbu čo najlepších 3D modelov, použitými technológiami a multiplatformovosti. Kapitola 10 sa zameriava na celkový návrh systému, ktorého súčasťou je architektúra, definované prípady použitia a diagram tried.

8 Ciele na zimný semester

Na začiatku prvého šprintu boli definované hlavné ciele na zimný semester:

- Analýza potrebných technológií
- Vytvorenie 3D modelu miestnosti 3D Lab
- Vytvorenie prototypu pokrývajúceho základnú funkcionalitu finálnej aplikácie

Po ukončení prvého šprintu sme vytvorili špecifikáciu výsledného produktu so zákazníkom (vedúci tímu). Definovali sme požiadavky na finálny projekt, vyhľadali dostupné a použiteľné technológie a nástroje na fotogrametriu, ktoré budú potrebné pri vývoji finálneho produktu.

Požiadavky na projekt

- zobrazenie vizualizovaného priestoru návšetvníkovi prostredníctvom HMD a browsera
- navigácia návšetvníka stránky vo vizualizovanom priestore
- možnosť interakcie návštevníka s objektami v scéne pomocou ovládačov
- pridávanie modelov
- pridávanie popisov a bližších informácií

MVP pre zimný semester

Za cieľ v zimnom semestri sme si vytýčili vytvorenie prototypu, ktorý bude musieť spĺňať nasledovné požiadavky:

- Interakcia používateľa s objektami v scéne
- Zobrazenie prostredia na základe reálneho pohybu osoby
- Vytvorenie prototypovej scény

9 Analýza

Táto kapitola je zameraná na analýzu oblasti, do ktorej zapadá náš projekt.

9.1 Fotogrametria

Proces fotogrametrie sa zameriava na sériu fotografií z ktorých zostavuje presné polohy povrchových bodov a rekonštruuje na základe polohy fotoaparátu, orientácie, typu objektívu, skreslenia šošoviek…

Ako nástroj na tvorbu 3D modelov z fotiek bol vybratý Autodesk ReMake na základe najlepšie dosiahnutých doterajších výsledkov. Pre čo najlepšie výsledné 3D moduly je potrebné dodržiavať nasledujúce pokyny:

Nastavenie fotoaparátu + základné pravidlá:

- Fotky musia byť ostré po celej scéne nie len snímaný objekt (Nevhodné fotky vymažte)
- Výsledné fotografie neupravujte
- Použite statív (+rovnaká výška +akýkoľvek pohyb môže mať vplyv na ostrosť fotografií)
- Ideálne fotiť s pevným objektívom, 50mm
- Nemeňte zoom, skôr sa pohybujte smerom k a mimo objektu
- Počas rovnakej akvizície nemeňte expozíciu
- Nastavte ISO na 100 (čím vyššie ISO tým väčší šum)
- Formát JPEG

Svetelné podmienky

- Je potrebné sa vyhnúť prostrediu, v ktorom svetlo vytvára veľké kontrastné tiene, pretože tiene sú jedným z najväčších nepriateľov fotogrametrii
- Nepoužívať blesk
- Vyhnúť sa časom počas najväčšieho pôsobenia slnka pri objektoch v exteriéry

Stratégie pri nastavovaní scény podľa objektov

Ak je objekt jednofarebný:

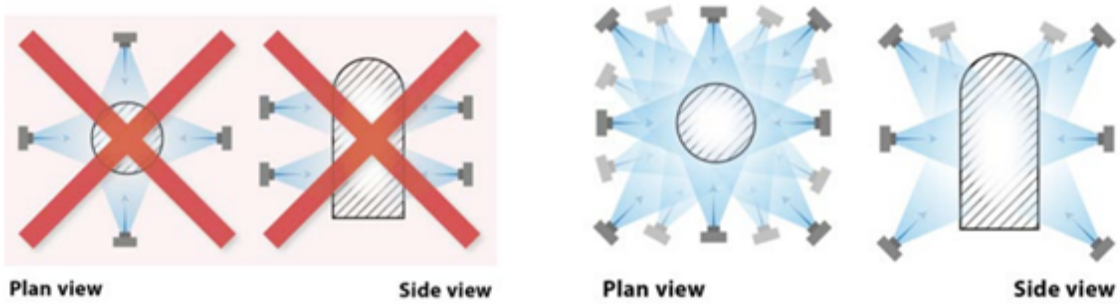
- Umiestniť ho na čo najviac farebné pozadie, alebo ak pozadie je pozadie jednofarebné tak pridať čo najväčšie množstvo rôznych farebných objektov, ktoré nebránia získavaniu referenčných bodov
- Objekt by mal byť statický a pohybovať počas fotenia by sa mal iba fotoaparát

Ak má objekt viac farieb:

- Umiestniť ho na nejaké jednofarebné pozadie najideálnejšie je ho umiestniť na biele pozadie
- Počas celého fotenia buď je objekt alebo fotoaparát je statický

Všeobecne:

- Objekt by mal byť umiestnený v strede danej fotografie a pokrývať najmenej 70% z celej fotografie
- Pri fotení nepohybujte s okolitými predmetmi, ktoré sú taktiež zachytené na daných fotografiách
- Snímajte celý objekt, aby ste zakryli každý jeho aspekt. Posúvať by ste sa mali po každých 5-15 stupňov v rovnakej výške z dôvodu dostatočného prekrytia záchytných bodov na fotkách. Pre čo najlepšiu úroveň detailov fotiť objekt v 2-3 rôznych výškach.
- Počet fotografií pre malé až stredné objekty 70-100, pre budovy 180-200, štýl fotenia je ukázaný na obrázku č. 1

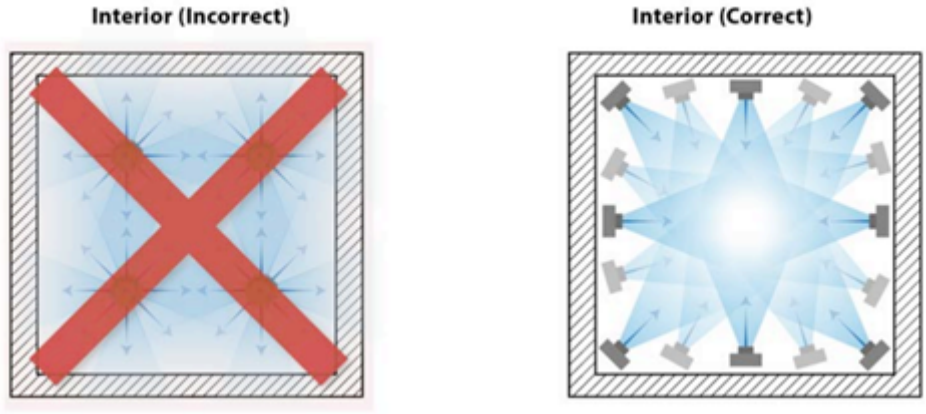


Obrázok č. 1: Ukážka štýlu fotografovania

- Čím viac fotografií tým lepšie ale stále platí podmienka aby sa fotografie dostatočne prekrývali

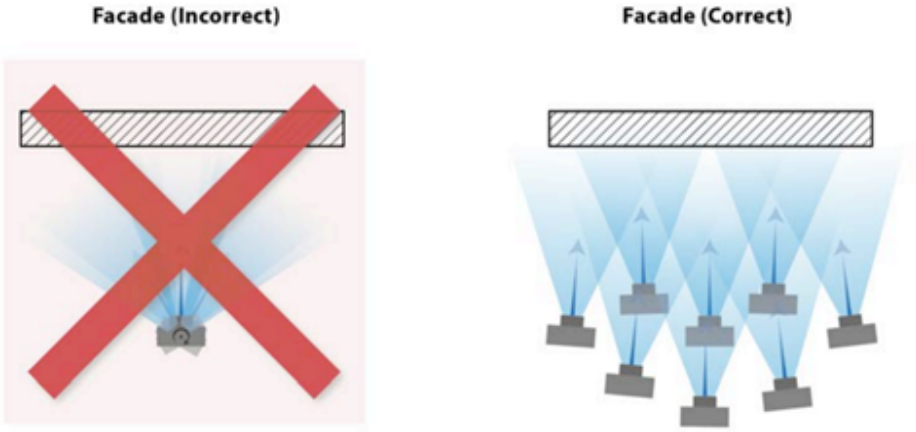
<u>Miestnosti/Budovy</u>

- Pri snímaní v interiéry treba postupovať podľa pokynov na obrázku č. 2. (Avšak pri interiéroch s prázdnymi jednofarebnými stenami, zlyhajú všetky metódy fotenia, keďže obsahujú strašne málo záchytných bodov)



Obrázok č. 2: Ukážka správnej polohy fotoaparátu v interiéri

- Pri snímaní v exteriéri treba postupovať podľa pokynov na obrázku č. 3. (Pozor na pohybujúce sa objekty napr. chodci, vtáctvo...)



Obrázok č. 3: Ukážka správnej polohy fotoaparátu v exteriéri

9.2 Virtuálna realita

Všetko, čo poznáme o našej realite, prichádza prostredníctvom našich zmyslov. Z toho vyplýva, že naša celá skúsenosť s realitou je jednoducho kombinácia zmyslových informácií a nášho rozumu vytvárajúci zmysluplné mechanizmy pre tieto informácie. Je teda možné, že ak dokážeme prezentovať svoje zmysly s novo vytvorenými informáciami, naše vnímanie reality sa v reakcii na to zmení. Budeme mať predstavenú verziu reality, ktorá v skutočnosti neexistuje, ale z nášho hľadiska by bola vnímaná ako skutočná.

Z technického hľadiska v skratke je virtuálna realita termín, ktorý sa používa na opis trojrozmerného prostredia vytvoreného prostredníctvom počítača, ktoré môže človek preskúmať a s ním spolupracovať. Táto osoba sa stáva súčasťou tohto virtuálneho sveta,

alebo je ponorená do tohto prostredia a je schopná manipulovať s objektmi alebo vykonávať sériu akcií, pomocou ktorých dané prostredie skúma.

Čiže virtuálna realita je vytvorenie virtuálneho prostredia prezentovaného našim zmyslov takým spôsobom, že ho zažijeme, akoby sme boli naozaj v ňom. Na dosiahnutie tohto cieľa využíva množstvo technológií, ktoré musia zohľadňovať naše vnímanie a poznanie.

9.3 Technológie (nástroje)

Na vytvorenie 3D modelov či už miestnosti alebo objektov, ktoré sa v nej nachádzajú existuje niekoľko spôsobov. Ako prvý spôsob sme si vybrali skenovanie. Bolo vytvorených mnoho nástrojov na skenovanie objektov a následné vytváranie 3D objektov. Jedným z nich je Tango Point Cloud.

9.3.1 Tango Point Cloud

Aplikácia Tango umožňuje mobilnému zariadeniu snímanie hĺbky, respektíve vzdialenosti od objektu v reálnom čase. Technológia tango umožňuje snímanie objektov vzdialených od 0,5 m do 4m. Nie je teda vhodná na snímanie blízkych objektov. Tango snímané objekty prevádza do takzvaného “pointcloudu“, teda do množiny bodov v priestore. Každý bod v takomto priestore má súradnice (x, y, z). Táto technológia je vhodná na snímanie jednoduchších objektov. Pri snímaní komplikovaných objektov vznikajú vo finálnom objekte defekty, ktoré je náročné odstrániť.

Po niekoľkých pokusoch sme túto aplikáciu zavrhli, pretože výstup nebol postačujúci pre náš projekt.

9.3.2 3DSOM

Spoločnosť Creative Dimension Software Ltd vydalo 3DSOM pro, profesionálny nástroj na vytváranie a prezentáciu 3D modelov z fotografií. Sada nástrojov, ktorú softvér obsahuje, umožňuje používateľovi vytvárať modely, ako sú vázy, misy, alebo aj organické zložité objekty, ktoré sú tradične ťažko modelovateľné.

Ďalšou schopnosťou softvéru je modelovať objekty bez kalibračnej rohože (vytlačený list A4 umiestnený pod cieľovým objektom), ale aj napriek tomu dokáže zautomatizovať modelovanie. Kalibračný prístup funguje veľmi dobre pri malých objektoch, ale môže byť veľmi zložitý pri modelovaní väčších objektov, ako sú skrine, stoly, auto a pod.

Upustili sme od tohto nástroja, z toho dôvodu, že sa nám nepodarilo vygenerovať použiteľné objekty vo VR prostredí kvôli veľkým objektom, ktoré sa nachádzajú v 3D labe.

9.3.3 Scann3D

Scann3D je mobilná aplikácia od spoločnosti SmartMobileVision. Je funkčná iba na operačnom systéme Android. Využíva technológiu fotogrametrie na vytváranie 3D modelov z mobilného telefónu či tabletu. Na rozdiel od ostatných nástrojov Scann3D nepoužíva "cloud processing", čiže všetky výpočty sú realizované priamo na zariadení. Modely vytvorené pomocou tejto aplikácie sú najčastejšie využívané v rozšírenej alebo virtuálnej realite.

Tak ako všetky predchádzajúce programy, ani ten nemal požadovaný výstup pre náš projekt. Fotografie, ktoré boli predpripravené v príkladoch síce mali výstup pekný ale po použití nami vytvorených fotografií bol výstup nepoužiteľný.

9.3.4 3DF Zaphyr

3DF Zaphyr umožňuje automatické vytváranie 3D modelov z fotografií. Nie je potrebné žiadne manuálne upravovanie fotografií. 3DF Zaphyr prichádza s jednoduchým používateľským rozhraním. Je vhodný na vytváranie 3D modelov z fotografií reálneho prostredia. 3DF Zaphyr disponuje možnosťou exportovania do mnohých 3D formátov tiež vygenerovaním videa bez potreby externých nástrojov. Podporuje aj exportovanie "mesh" z fotografií do formátov ako: .ply, .obj/mtl a aj do nástroja Sketchfab. Verzia zadarmo umožňuje vytvárať 3D modely z maximálne 50-tich fotografií. Študentská verzia má tento limit nastavený na 500 fotografií.

Pri testovaní programu s nami vytvorenými fotografiami program zlyhal a výstup bol opäť nepoužiteľný.

9.3.5 Samsung Gear 360

Samsung Gear 360 kamera ukázaná na obrázku č. 4 je technológia ktorá vytvára 360° fotografie s 8MP fotoaparátom. Sprievodná aplikácia je kompatibilná iba s mobilnými telefónmi Samsung Galaxy vyššej triedy ako 6, no s novou verziou v roku 2017 pribudla aj kompatibilita s mobilnými telefónmi značky Apple.

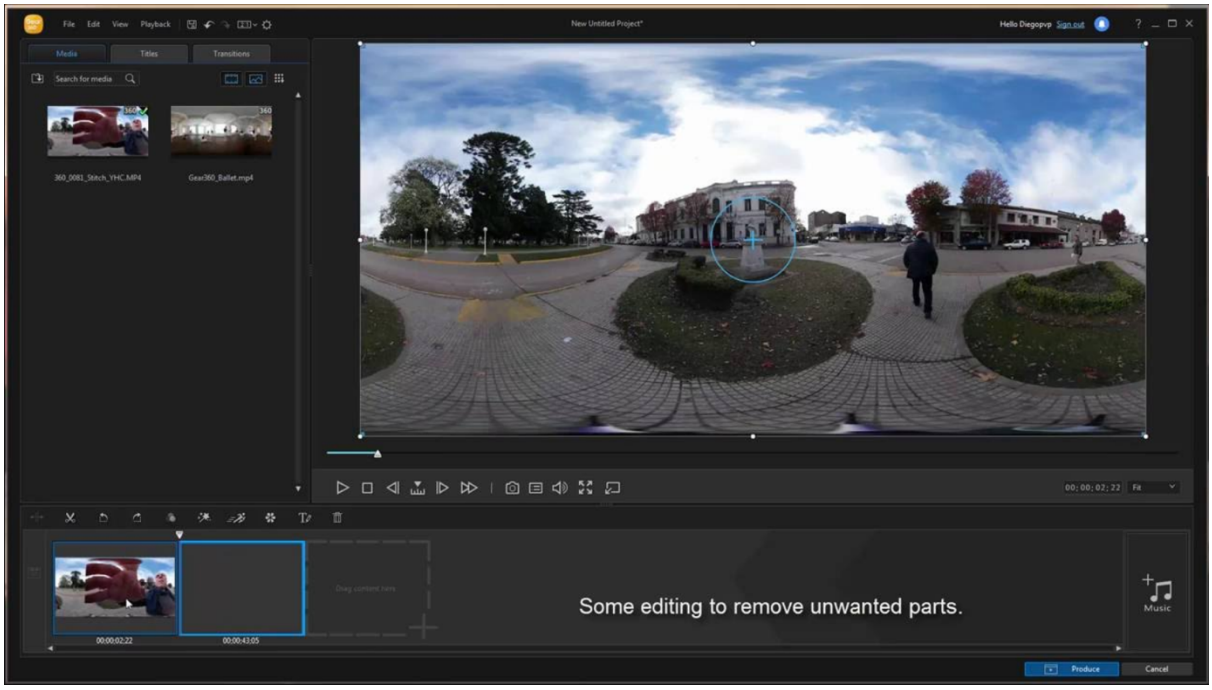
Obrázok č. 4: Samsung Gear 360

Fotoaparát sa spája s mobilným telefónom prostredníctvom pripojenia Wi-Fi. Aplikácia vám umožňuje jednoducho prepínať medzi režimami videozáznamov, fotografií, časových pásiem, videosekvencií a režimu HDR na šírku. Následne je možná kontrola nad podrobnejšími nastaveniami, ako je kompenzácia expozície a vyváženie bielej farby.

Veľkou výhodou je, že zariadenie máme k dispozícií v samotnom 3D Labe, čo nám umožňuje ho využiť na prototype Panorama, ktorý je tvorený z týchto fotografií (Príklad fotografie je ukázaný na obrázku č. 5).

Obrázok č. 5: Fotografia časti 3D Labu vyhotovená pomocou kamery Samsung Gear 360

Na spracovanie výstupov zo zariadenia je navrhnutý špeciálny softvér Gear 360 ActionDirector, ktorý je ukázaný na obrázku č. 6. Softvér poskytuje niekoľko základných funkcií úprav, ako je orezanie, pridávanie titulov a vytváranie prechodov vo videách a fotografiách. Po spracovaní fotografií je zobrazený kvalitný 3D priestor a vo formáte, ktorý sa dá ľahko premietnuť vo virtuálnej realite pomocou HTC Vive.



Obrázok č. 6: Ukážka softvéru Gear ActionDirector

9.4 Multiplatformovosť

Významnou výhodou nášho projektu je jeho multiplatformovosť. Technológie webGL a webVR nám umožňujú zobrazovať 3D realitu na všetkých zariadeniach, ktoré tieto technológie podporuju. Jedná sa o širokú škálu zariadení. V nasledujúcich kategóriach si opíšeme jednotlivé technológie.

9.4.1 WebGL

Technológia webGL nám zabezpečuje bezplatný štandard vykresľovania 3D grafiky na HTML5 plátne (canvas). Najnovšia verzia webGL 2.0 ponúka podobné API rozhranie ako OpenGL ES 3.0. Najväčšou výhodou tejto technológie je jej multiplatformovosť a podpora od bežných PC po mobilné telefóny (smartphony) rôznych výrobcov. Nižšie môžeme vidieť webové prehliačače, ktoré v súčasnosti (November 2017) podporujú webGL technológiu.

Webové prehliadače podporujúce webGL:

- Google Chrome – verzia 56+
- Mozilla Firefox – verzia 51+
- Midori
- Safari – podpora webGL 1.0
- Opera + verzia 43+
- Internet Explorer – webGL 1.0 čiastočné podporované od verzie 11
- Microsoft Edge – webGL 1.0 podporuje od verzie 10240. Podpora webGL 2.0 je plánovaná.
- Vivaldi

Mobilné prehliadače:

- BlackBerry 10 – WebGL 1.0 od verzie 10.0
- BlackBerry PlayBook – WebGL 1.0 od verzie 2.00
- Android Prehliadač – webGL 1.0 v základe nepodporované, ale niektoré zariadená ho podporujú po nainštalovaní novšieho formware. Na väčšine zariadení je sprístupnený pomocou prehliadača Google Chrome
- Internet Explorer – WebGL 1.0 je podporované na Windows Phone 8.x (11+)
- Firefox for mobile – WebGL 1.0 pre Android a MeeGo zariadenia od verzie 4
- Firefox OS
- Google Chrome – WebGL 1.0 od verzie 25
- MeeGo – WebGL 1.0 v základe nepodporované, ale dostupne cez Firefox.
- Microsoft Edge – WebGL 1.0 je dostupne on Windows 10 Mobile
- Opera Mobile – Opera Mobile 12 supports WebGL 1.0 (iba na Android zariadeniach)
- Sailfish OS – WebGL 1.0 je podporované
- Tizen – WebGL 1.0 podporované
- Ubuntu Touch
- WebOS
- iOS – WebGL 1.0 je dostupné pre mobile Safari

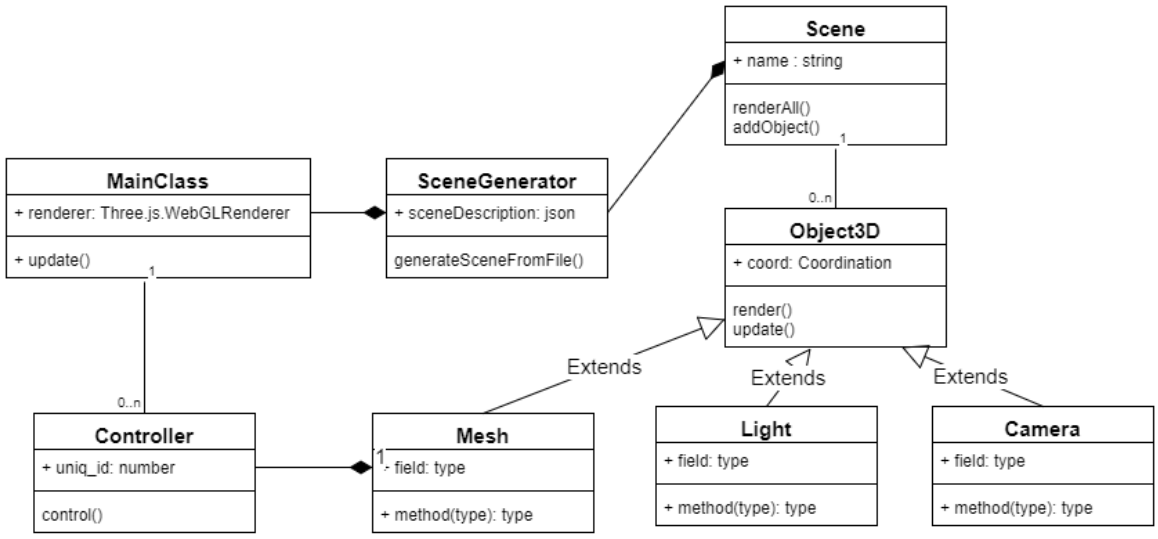
9.4.2 WebVR

Táto technológia nám umožňuje vytvárať aplikácie pre virtuálnu realitu pomocou webových technológií HTML5 a javascript. Vďaka webVR je možné vytvárať jednu aplikáciu pre virtuálnu realitu, ktorá bude fungovať na rôznych druhoch VR headsetov (náhlavných sústav pre Virtualnu realitu). Vďaka tejto aplikácii môžeme predpokladať širšie publikum pre našu aplikáciu.

10 Návrh

V tejto kapitole rozoberieme podrobný návrh architektúry vyvíjanej aplikácie a taktiež návrh vizualizácie.

10.1 Diagram tried



Obrázok č. 7: Diagram tried

10.1.1 Opis tried

V tejto podkapitole sú opísané triedy, ktoré sa nachádzajú v diagrame tried na obrázku č. 7.

MainClass

Hlavná trieda inicializuje všetky potrebné objekty na vytvorenie 3D scény pre virtuálnu realitu. Taktiež zodpovedná za spúšťanie hlavného cyklu vykresľovania a aktualizovania scény.

Controller

Trieda je určená na spracovanie informácii z pohybových ovládačov, akými sú aj informácie o stlačených tlačidlách alebo informácie o polohe ovládača v priestore. Na základe týchto informácii sa vykonávajú funkcie interakcie s ostatnými objektami v scéne.

SceneGenerator

Trieda načítava z predurčeného súboru štruktúru scény (objekty a ich parametre) a vytvorí komplexnú scénu potrebnú na vykreslenie 3D priestoru.

Mesh

Trieda dedí od triedy Object3D základne vlastnosti o objekte a pridáva ďalšie informácie na zobrazenie komplexných objektov skladajúcich sa z mnohých polygónov.

Object3D

Základná trieda pre všetky objekty obsahujúca potrebné vlastnosti pre zobrazenie v 3D priestore (napr.: x,y,z súradnice, rotáciu).

Scene

Trieda obsahujúca všetky potrebné objekty (3D objekty, kameru a svetlá).

Light

Trieda dedí od triedy Object3D základne vlastnosti a pridáva vlastnosti potrebné na opis svetla v priestore.

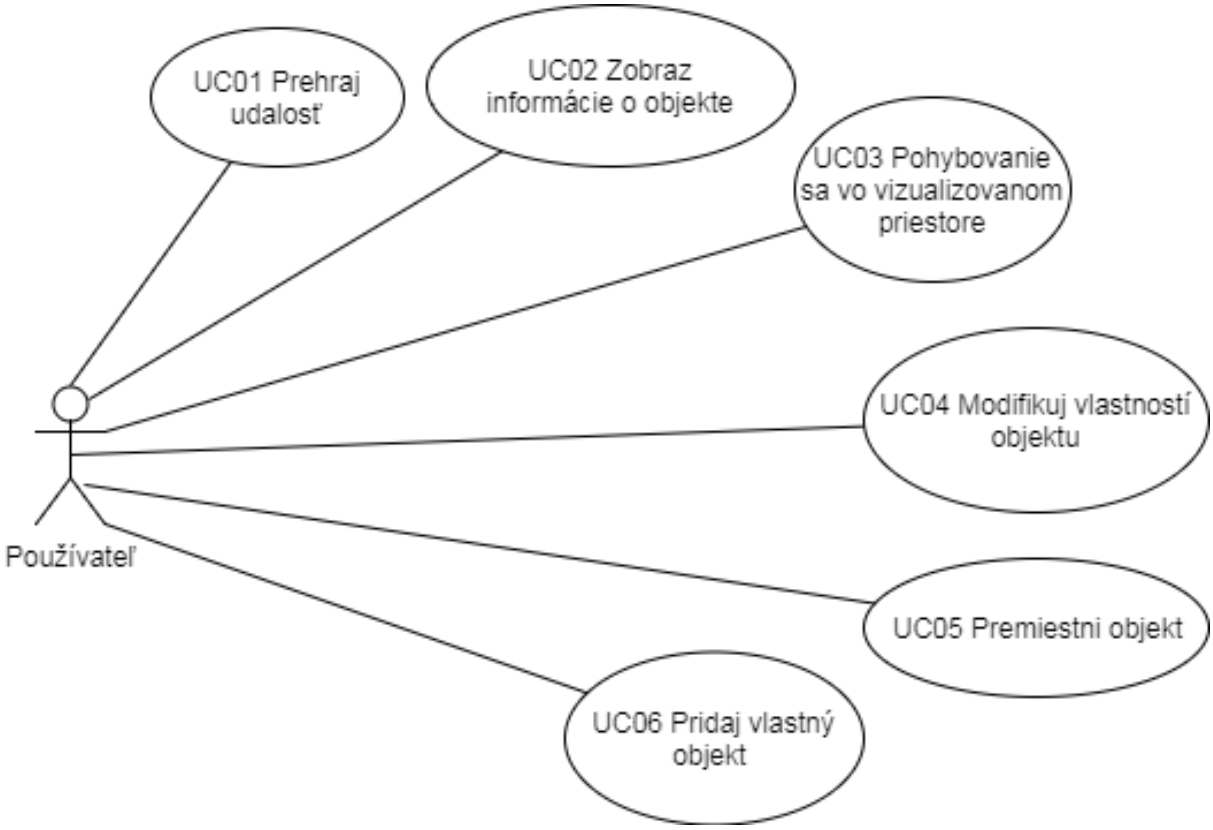
Camera

Trieda dedí od triedy Object3D základne vlastnosti a pridáva vlastnosti potrebné na opis kamery v scéne. Kamera je dôležitý prvok ovplyvňujúci výsledný vzhľad prostredia vo virtuálnej realite.

10.2 Prípady použitia

Prípady použitia nám definujú funkcionálne požiadavky aplikácie.

Diagram prípadov použitia zobrazuje závislosti medzi doteraz identifikovanými prípadmi použitia. Na obrázku č. 8 je diagram prípadov použitia. Používateľ môže jednoducho a priamočiaro používať systém.



Obrázok č. 8: Diagram prípadov použitia

Jednotlivé prípady použitia sú spísané formou tabuliek pre jednoduché čítanie a orientáciu sa. Doteraz sme z požiadaviek zákazníkov identifikovali nasledovné prípady použitia.

UC01 Prehraj udalosť

Prípad použitia	
ID:	01
Názov:	Prehraj udalosť
Popis:	Používateľ sa ocitne v priestore, v ktorom sa môže otáčať a pozerať tak na udalosti, ktoré sa deju vôkol neho.
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a má k dispozícii HMD.
Dôsledky:	Používateľ sa ocitne v prehrávanom deji.

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ klikne na objekt, ktorý umožní prehrať video	Systém zobrazí používateľa v strede prehrávanej udalosti.
2	Používateľ kliknutím na “ukončenie prehrávania” ukonči prehrávanie	Systém skončí prehrávanie videa a zobrazí používateľa na pôvodnej pozícií

UC02 Zobrazenie informácie o objekte

Prípad použitia	
ID:	02
Názov:	Zobrazenie informácie o objekte
Popis:	Používateľovi sa po namierení na objekt zobrazia podrobnosti.
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a má k dispozícii HMD.
Dôsledky:	Používateľovi sa zobrazí vo vizualizovanom priestore okno s informáciami o objekte

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na objekt	Systém zobrazí okno s informáciami o objekte
2	Používateľ namieri mimo daný objekt	Systém skryje okno s informáciami o objekte
Alternatívny tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na objekt, ktorý neumožňuje zobrazenie popisu.	Systém nezobrazí nič

UC03 Pohybovanie sa vo vizualizovanom priestore

Prípad použitia	
ID:	03
Názov:	Pohybovanie sa vo vizualizovanom priestore
Popis:	Používateľ sa vo vizualizovanom priestore môže presúvať pomocou takzvaného teleportovania. Kliknutím na plochu, ktorá umožňuje presunutie sa používateľ objaví na danej pozícií.
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a nie je v móde prehrávania udalosti a má k dispozícií HMD.
Dôsledky:	Používateľ sa presunie vo vizualizovanom priestore na iné miesto.

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na podlahu	Systém zvýrazní plochu
2	Používateľ sa kliknutím presunie	Systém zmení pozíciu používateľa tak aby sa zjavil na zvolenej pozícií

UC04 Modifikuj vlastností objektu

Prípad použitia	
ID:	04
Názov:	Modifikuj vlastností objektu
Popis:	Používateľ sa vo vizualizovanom priestore môže modifikovať vlastností objektu. Vybraním objektu môže meniť pomocou spôsobu na to určeného vlastností jednotlivého objektu ako sú: farba, rozmery, priehľadnosť a mnohé ďalšie...
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a nie je v móde prehrávania udalosti a má k dispozícii HMD.
Dôsledky:	Objekt ostane zmodifikovaný

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na objekt	Systém zvýrazní objekt
2	Používateľ kliknutím zvolí možnosť modifikovať objekt	Systém umožní používateľovi modifikovať vlastností objektu
3	Používateľ modifikuje dáta objektu a uloží zmeny	Systém nevratne uloží vykonané zmeny a zobrazí používateľa na pôvodnej pozícií vo vizualizovanom priestore

Alternatívny tok:		
Číslo kroku	Používateľ	Systém
3	Používateľ preruší modifkovanie objektu	Systém zobrazí používateľa na pôvodnej pozícií vo vizualizovanom priestore

UC05 Premiestni objekt

Prípad použitia	
ID:	05
Názov:	Premiestni objekt
Popis:	Používateľ môže vo vizualizovanom priestore presúvať jednotlivé objekty. Kliknutím na objekt, podržaním tlačidla a jednoduchým pohybom ruky premiestni objekt na inú pozíciu.
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a nie je v móde prehrávania udalosti a má k dispozícii HMD.
Dôsledky:	Objekt ostane na novej pozícií

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na objekt	Systém zvýrazní objekt.
2	Používateľ kliknutím vyvolá presúvanie objektu	Systém umožní používateľovi presúvať objekt.
3	Používateľ pohybom premiestni objekt	Systém priebežne zobrazuje aktuálnu pozíciu objektu.
4	Používateľ skončí s premiestňovaním objektu	Systém nevratne uloží novú pozíciu objektu.

UC06 Pridaj vlastný objekt

Prípad použitia	
ID:	06
Názov:	Pridaj vlastný objekt
Popis:	Používateľ môže vo vizualizovanom priestore pridávať vlastné objekty. Umožní to pomocou pridania nového objektu medzi existujúce objekty.
Používateľ:	Používateľ
Predpoklady:	Používateľ je vo vizualizovanom priestore a nie je v móde prehrávania udalosti a má k dispozícii HMD.
Dôsledky:	Nový objekt zostane súčasťou vizualizovaného priestoru

Základný tok:		
Číslo kroku	Používateľ	Systém
1	Používateľ namieri na objekt	Systém zvýrazní objekt.
2	Používateľ kliknutím vyvolá presúvanie objektu	Systém umožní používateľovi presúvať objekt.
3	Používateľ pohybom premiestni objekt	Systém priebežne zobrazuje aktuálnu pozíciu objektu.
4	Používateľ skončí s premiestňovaním objektu	Systém nevratne uloží novú pozíciu objektu.

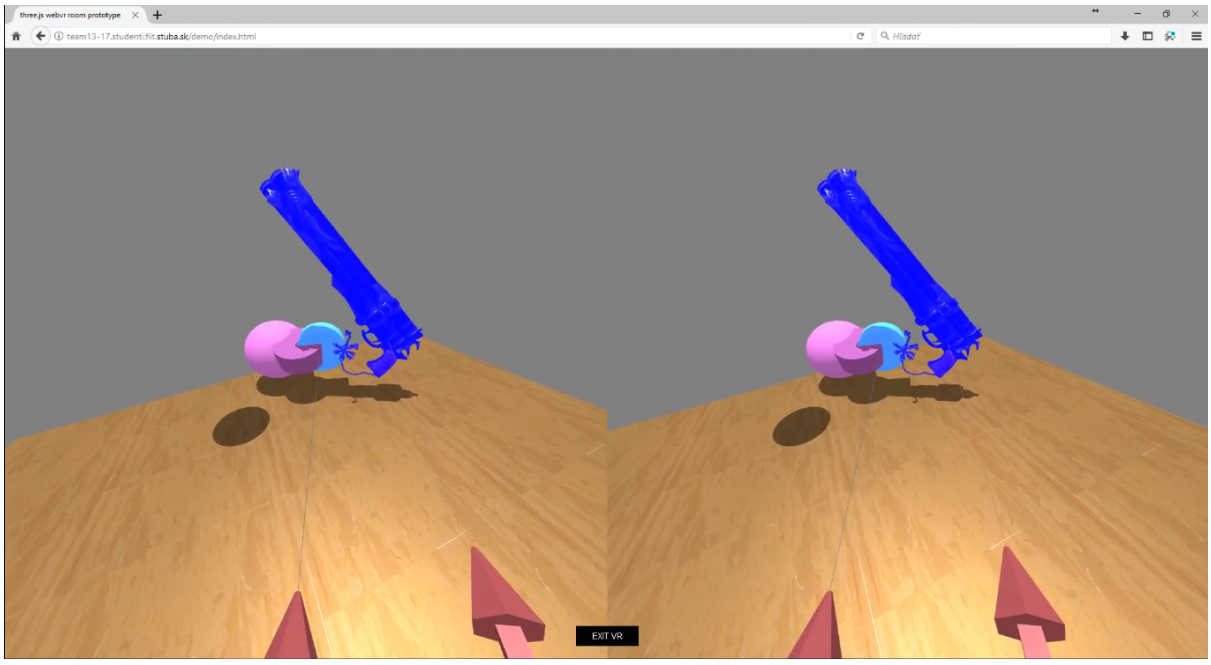
11 Implementácia

Ako bolo zistené v analýze, multiplatformovosť najlepšie dosiahneme ak budeme používať technológiu webVR umožňujúce tvorbu webových aplikácii pre virtualnú realitu. Základom projektu je podporovaný webový prehliadač a naša webová aplikácia.

11.1 Prvý prototyp – 3D Room

Prvý prototyp aplikácie bol implementovaný pomocou knižnice Three.js dostupnej na nasledovnej adrese: https://threejs.org/ Knižnica výrazne pomáha pri tvorbe webgl 3D scén pomocou jazyka JavaScript. V posledných verziách bola vylepšená podpora pre virtuálnu realitu, čo nám pomohlo pri tvorbe našej prvej verzie aplikácie. Prototyp bol určený hlavne na využitie s virtuálnou realitou HTC Vive.

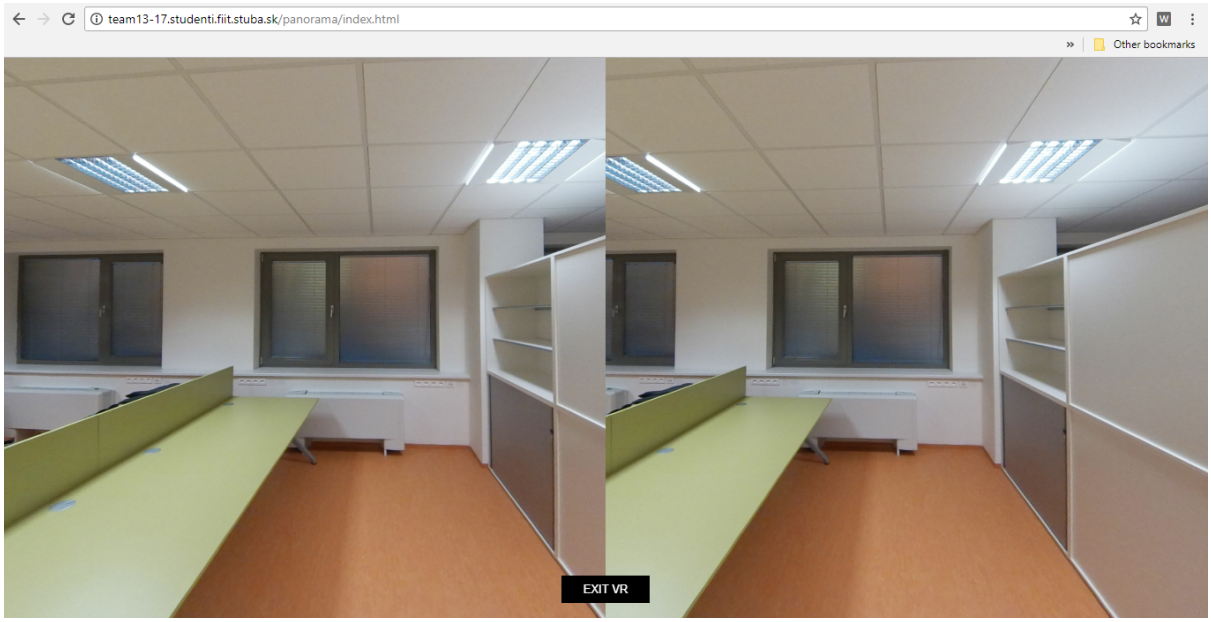
Implementovali sme základnú podporu pohybových ovládačov a aj pohyb kamery podľa pohybu človeka v reálnom priestore. Ovládače sú v prototype reprezentované pomocou šipiek so svetelným lúčom. Pomocou lúča vieme vyberať objekty v scéne a pohybovať s nimi podľa pohybu ovládača. Na obrázku č. 6 je znázornená scéna a ovládače nášho prototypu.



Obrázok č. 6: Scéna a ovládače v prototype 3D Room

11.2 Druhý prototyp – Panorama

Druhý prototyp aplikácie bol implementovaný v čistom java skripte využitím knižnice Three.js čo umožňuje veľmi dobrý výkon a teda realistický dojem. Implementovali sme 360° panoramu. používateľ sa nachádza v 3D sfére, ktorá je z vnútornej strany otexturovaná jednou alebo viacerými fotkami čí videom. Používateľ sa v prototype môže otáčať. Pohybovanie v tomto prototype nie je umožnené. Na obrázku č. 7 je ukážka z prototypu.



Obrázok č. 7: Ukážka z prototypu Panorama

12 Testovanie

Testovanie vykonávame priamo s HMD v 3D laboratóriu. Testuje sa výhradne podľa spísaných testovacích scenárov. Vždy testuje osoba, ktorá sa nepodieľala na implementácii funkcionality daného testovacieho scenára. Testovacie scenáre sú vo forme tabuliek. V prípade potreby, môže tester ešte dostať ďalšie informácie ohľadom testovania.

Testovací scenár prehrávania udalosti	
Test Case ID:	01
Use Case:	UC05
Programátor:	Meno programatora
Tester:	Meno testera
Vstupné požiadavky:	Používateľ je už vo vizualizovanom priestore Používateľ má k dispozícií ovládač HTC VIVE. Používateľ si spustí demo z /tests/test1.html
Výstupné požiadavky:	Objekt ostane na novej pozícií

číslo	**Akcie používateľa**	**Očakávaný výstup systému**	**Komentár**
1.	Namierte na pištoľ	Pištoľ sa zvýrazní	
2.	Podržte tlačítko a urobte niekoľko ľubovoľných pohybov	Systém bude podľa pohybov ruky meniť pozíciu pištole	
3.	Poustíte tlačítko	Systém preruší presúvanie pištole	

Testovací scenár prehrávania udalosti	
Test Case ID:	02
Use Case:	UC01
Programátor:	Meno programatora
Tester:	Meno testera
Vstupné požiadavky:	Používateľ je už vo vizualizovanom priestore Používateľ má k dispozícii ovládač HTC VIVE. Používateľ si spustí demo z /tests/test2.html
Výstupné požiadavky:	Používateľ sa ocitne uprostred prehrávaného deja, v ktorom sa môže obzerať vôkol seba.

číslo	**Akcie používateľa**	**Očakávaný výstup systému**	**Komentár**
1.	Kliknite na zelenú guľu	Zobrazí sa nový vizualizovaný priestor	
2.	Otočte sa okolo svojej osi a pozrite sa hore a dole.	Systém zobrazuje plynule prehrávaný dej v celom priestore.	
3.	Používateľ klikne na červenú guľu	Systém preruší prehrávanie deja a zobrazí používateľa na pôvodnej pozícii	

Testovací scenár zobrazenia podrobností o objekte	
Test Case ID:	03
Use Case:	UC02
Programátor:	Meno programatora
Tester:	Meno testera
Vstupné požiadavky:	Používateľ je už vo vizualizovanom priestore Používateľ má k dispozícii ovládač HTC VIVE. Používateľ si spustí demo z: /tests/test3.html
Výstupné požiadavky:	Zobrazenie popisu objektu

číslo	**Akcie používateľa**	**Očakávaný výstup systému**	**Komentár**
1.	Namierte na pištoľ	Vo vizualizovanom priestore sa zobrazí okno s popisom	
2.	Namierte mimo pištole	Systém skryje okno s popisom	

Testovací scenár pohybovania sa vo vizualizovanom priestore	
Test Case ID:	04
Use Case:	UC03
Programátor:	Meno programatora
Tester:	Meno testera
Vstupné požiadavky:	Používateľ je už vo vizualizovanom priestore Používateľ má k dispozícii ovládač HTC VIVE. Používateľ si spustí demo z /tests/test4.html
Výstupné požiadavky:	Premiestnenie používateľa na zvolenú pozíciu.

číslo	**Akcie používateľa**	**Očakávaný výstup systému**	**Komentár**
1.	Namierte na sivú podlahu	Zvýrazní sa úsek na zeleno	
2.	Namierte na drevenú podlahu	Nezvýrazni sa	
3.	Namierte na sivú podlahu a kliknite	Premiestni používateľa na namierenú pozíciu	

Príloha A – Metodiky

Metodika dokumentácie

1. Úvod

Dokumentovať kód je dôležité hlavne kvôli tomu, aby si ľudia, ktorí majú záujem používať naše riešenie, prípadne v ňom pokračovať, mali odkiaľ naštudovať, čo a ako funguje. Je dôležité mať miesto, kam sa dá pozrieť a vedieť si nájsť ako sa volá funkcia, ktorá má nejaký význam/účel a taktiež si pozrieť aké má vstupy a aké poskytuje výstupy. Takýto technický popis modelu nášho projektu poskytuje technická dokumentácia, ktorej pravidlá je dobré pre správne fungovanie tímového projektu zadefinovať. Taktiež je dôležité zadefinovať, akým spôsobom vytvárať iné druhy dokumentácií a to tie, ktoré nebudú generované z nami vytváraného kódu, ale tie, čo vytvárame samostatne. Na to, aby bola forma vždy jednotná, je potrebné určiť si základné pravidlá písania dokumentácie.

2. Štýl písania dokumentácie

V každom vzniknutom dokumente musí byť dodržaných niekoľko pravidiel.

1. Hlavné kapitoly majú nadpis typu 1
2. Podkapitoly majú nadpis typu 2
3. Číslovanie kapitol podľa úrovní
    a. Prvá úroveň - 1
    b. Druhá úroveň - 1.1
    c. Tretia úroveň - 1.1.1
4. Ďalší štýl, resp. úroveň nadpisov je na uvážení tvorcu dokumentácie
5. Typ písma a veľkosť písma normálneho textu je rôzna podľa typu dokumentu
6. Riadkovanie použijeme veľkosti 1.15

3. Šablóny dokumentov

Dokumenty, ktoré sú vytvárané pravidelne majú predpísané šablóny.

3.1. Šablóna zápisnice

ZÁPISNICA ZO STRETNUTIA TÍMU

zo dňa 25.9.2017

Miesto a čas konania:
miestnosť 3.38, 10:00 – 13:00

Účastníci stretnutia:

- Ing. Vincúr Juraj
- Bc. Andrejkovič Ivan
- Bc. Broniš Tadeáš
- Bc. Hózová Gabriela
- Bc. Janec Nikolas
- Bc. Lišiak Jaroslav
- Bc. Ofčarovič Tomáš
- Bc. Škuta Michal

Program stretnutia:

1.
2.
3.
4.
5.

Rozpísané body programu:

1.
2.
3.
4.
5.

3.2. Šablóna retrospektívy

RETROSPEKTÍVA ŠPRINTU ČÍSLO 1

Ciele šprintu

1.
2.
3.
4.
5.

Čo išlo podľa našich predstáv?

1.
2.
3.
4.
5.

Čo nešlo podľa našich predstáv?

1.
2.
3.
4.
5.

Čo plánujeme zlepšiť?

1.
2.
3.
4.
5.

Metodika písania kódu

1. Úvod

Nasledovná metodika stanovuje jasné pravidlá a odporúčania pre písanie zdrojových kódov v jazyku JavaScript pre náš projekt. Každý programátor by mal byť oboznámený s touto metodikou a vďaka tomu docielime prehľadný a zrozumiteľný kód pre každého člena tímu.

2. Konvencia pre názvy (naming conventions)

Názvy premenných ako aj celý kód píšeme v Anglickom jazyku a každú premennú alebo funkciu sa snažíme čo najviac výstižne pomenovať.

Entita	**Spôsob nápisu**
premenné	lowerCamelCase
funkcie/metódy	lowerCamelCase
triedy	UpperCamelCase
konštanty	UpperCamelCase

3. Konvencia štruktúry projektu

Cieľom projektu je vytvorenie webovej aplikácie bežiacej na strane klienta. Táto vlastnosť ovplyvňuje aj štruktúru kódu a umiestnenie súborov. V koreňovom priečinku sa vždy musí nachádzať *index.html* súbor obsahujúci štartovací bod našej aplikácie.

V priečinku *js* sa nachádzajú všetky JavaScript knižnice potrebné na fungovanie aplikácie.

Posledným priečinkom je *models* a obsahuje všetky potrebné grafické/multimediálne súbory zobrazené v aplikácii.

4. Odporúčania

- vkladať za každý príkaz znak **;**
- používať UTF-8 kódovanie pre zdrojové súbory
- vkladať jedno-riadkové komentáre na voľný riadok pred opisovanou časťou kódu alebo premennou
- je povolené používať viac-riadkové komentáre pomocou znakov /* a */
- používať klávesu Tab na odsadzovanie textu

- vkladať dostatočné množstvo medzerníkov pre prehľadnosť kódu, ako je znázornené na obrázku č. 1.

```
if ( a === 0 ) {

   // this is good
   return true;

}
```

Obrázok č. 1: Ukážka štruktúry if podmienky v zdrojovom kóde

Metodika komunikácie

1. Úvod

Účelom tejto metodiky je opísať a zadefinovať postupy komunikácie v tíme. Poskytuje informácie o správnom spôsobe komunikácie, určuje kanály pre rôzne typy komunikácie a uvádza postup pre správne použitie týchto kanálov. Metodika je určená pre všetkých členov tímu.

Komunikácia v tíme prebieha primárne prostredníctvom komunikačnej služby Slack, Skype, stretnutím tvárou v tvár a ďalej pomocou komentárov v zdrojovom kóde a komentárov v nástroji GitLab, ktorý je využívaný na manažment zadaných úloh.

2. Prehľad komunikačných kanálov

Nasledujúca tabuľka zobrazuje nástroje používané pri práci na projekte a ich využitie pri konkrétnych situáciách.

Činnosť	Slack	Skype	GitLab	Telefón	Osobné stretnutie	Email
Komunikácia s celým tímom	✓			✓	✓	
Manažovanie tímu	✓		✓	✓	✓	
Reportovanie o prebiehajúcej činnosti			✓		✓	
Reportovanie o dokončenej činnosti			✓		✓	
Pomoc pri riešení problému	✓			✓	✓	
Spresnenie podrobností o úlohe	✓		✓		✓	
Urgentný problém	✓			✓	✓	
Žiadosť o revíziu	✓		✓			
Vytvorenie novej úlohy			✓		✓	
Externá komunikácia						✓
Komunikácia s vedúcim tímu	✓				✓	
Komunikácia s neprítomným členom na stretnutí		✓				

3. Komunikačné kanály

Slack

V komunikačnom nástroji Slack sú vytvorené kanály na jednotlivé témy týkajúce sa projektu. Konkrétne sú to nasledovné:

#general - tento kanál sa využíva na komunikáciu všeobecných vecí týkajúcich sa tímového projektu, ktoré nespadajú do žiadneho iného kanálu

#dokumenty - kanál slúži na vladanie dokumentov ako zápisnice, retrospektívy a iné aby mohli prejsť revíziou ostatných členov tímu a následne mohli byť vložené na webovú stránku tímu

#fotky - do tohto kanálu sú vkladané fotografie, ktoré slúžia na vytváranie 3D modelov pomocou fotogrametrie

#tpcup - kanál je využívaný na komunikáciu týkajúcu sa súťaže TP CUP, do ktorej sme sa ako tím zapojili

#chaos - kanál chaos slúži na komunikáciu, ktorá nemusí byť priamo spojená s témou projektu

#kontakty - v kanáli kontakty sú vypísané kontaktné údaje členov tímu aby sme ich mali vždy po ruke a nemuseli ich nikde hľadať, kanál neslúži na inú komunikáciu

Skype

Ak sa nejaký člen tímu nemôže dostaviť na stretnutie, oznámi to dopredu ostatným členom tímu a následne ak je možné pripojí sa pomocou hovoru na Skype.

GitLab

Komunikácia v nástroji GitLab pozostáva z komentárov ku konkrétnym úlohám, kde členovia tímu riešia problémy alebo nejasnosti ohľadom úlohy. Taktiež pomocou priradenia značky “doing” oznamujú, že pracujú na úlohe a priradením značky “review” oznamujú, že úloha je dokončená a čaká na revíziu iným členom tímu.

Telefón

Telefonické spojenie je využívané v prípade, že nastal nejaký urgentný problém. Príkladom je napríklad oznámenie neúčasti na stretnutí ale taktiež aj problém pri plnení zadanej úlohy. Telefonická komunikácia zahŕňa hovor ale aj SMS.

Osobné stretnutie

Väčšina komunikácie prebieha počas osobného stretnutia, kedy je ideálne riešiť nejasnosti a problémy vzniknuté pri práci na úlohách. Na stretnutiach sú tak isto prezentované pokroky

na projekte a tým sa dostávajú vzniknuté zmeny na konkrétnych častiach projektu do povedomia ostatných členov tímu. Ak má člen tímu problém, ktorý nevie sám vyriešiť, dohodne sa s iným členom tímu na osobnom stretnutí a tam sa pokúšajú spoločnými silami tento problém vyriešiť.

Email

Emailová komunikácia pomocou tímového aliasu chaosfiit@gmail.com je využívaná na kontakt s okolím.

Metodika revízií

1. Úvod

Zdrojový kód ale aj dokumenty, ktoré vznikajú pri tvorbe projektu potrebujú byť skontrolované aj inými členmi tímu ako je autor. Kontrolu predstavuje revízia a v tejto metodike sú určené pravidlá a postupy pre revidovanie nášho projektu.

2. Revízia zdrojového kódu

Účelom tejto metodiky je sumarizácia pravidiel a vytvorenie štandardizovaných postupov pre revíziu kódu. Z dôvodu, že kód môže po otestovaní jeho vývojárom obsahovať logické chyby, či byť neprehľadný. Je potrebné dať kód do podoby, ktorá vyhovuje štandardom projektu, a preto je potrebne kód revidovať.

2.1. Roly

Na vytváraní kódu sa podieľa niekoľko nezávislých vývojárov, ktorých si môžeme rozdeliť do dvoch základných skupín:

- Autor kódu: osoba, ktorá vytvorila kód, čím sa určitou časťou podieľala na tvorbe aplikácie
- Posudzovateľ kódu: osoba zodpovedná za kontrolu kódu. Kontrola kódu môže prebiehať rôznymi spôsobmi, väčšinou však posudzovateľ kódu spíše pripomienky, ktoré sú následne posunuté autorovi kódu

2.2. Pravidlá revízie zdrojového kódu

<u>Posudzovateľ</u>

Posudzovateľ je osoba zodpovedná za kontrolu kódu a informovanie o chybách jeho autora. Za posudzovateľa je najlepšie zvoliť osobu, ktorá sa aktívnym spôsobom nepodielala na tvorbe kódu. Toto pravidlo je dobré uplatniť z dôvodu, aby nebola ovplyvnená doteraším priebehom vývoja, čím získame nezávislý názor na riešenú problematiku. Revíziu by však mala vykonávať osoba, ktorá má aspoň minimálne znalosti v danej problematike.

<u>Časová následnosť</u>

Posudzovateľ kontroluje kód, ktorý je funkčný a prešiel základným testovaním, čiže neobsahuje očividné logické chyby. Je to z dôvodu, aby posudzovateľ zbytočne nekontroloval kód, ktorý nefunguje. Jeho úlohou je chyby nájsť a následne reportovať autorovi kódu aj s možných riešením. Nemá chyby opravovať.

Sumarizácia

- Posudzovateľ musí mať minimálne znalosti o problematike
- Posudzovateľ sa aktívne nepodieľa na vývoji kontrolovaného kódu
- Posudzovateľ kontroluje funkčný kód
- Posudzovateľ neopravuje chyby v kóde, iba ich deteguje a píše pripomienky

Ako požiadať o code review

- Push
- Oznámenie v komunikačnom nástroji Slack
- Pridelenie úlohy členovi tímu, ktorý sa dobrovoľne prihlási

3. Revízia dokumentov

Účelom tejto metodiky je definovanie pravidiel a štandardov na revíziu dokumentácie. Dôvod revízie je, že dokumentácia môže obsahovať nedostatky, či skryté pravopisné chyby, nesprávnu štylizáciu textu, prípadne nesprávne formátovanie textu. Preto je nutné upraviť dokument podľa nasledujúcich pravidiel.

3.1. Roly

Na vytváraní dokumentu sa podieľajú všetci členovia tímu, ktorých môžeme rozdeliť do dvoch skupín:

- Autor dokumentu: Je osoba, ktorá je poverená spísaním dokumentu a zodpovedná dodržiavaním pravidiel písania dokumentácie
- Posudzovatelia dokumentu: členovia tímu, ktorí posudzujú dokument, ktorý napísal autor

3.2. Pravidlá revízie dokumentov

Časová následnosť

Revízia dokumentu prebieha pomocou nástroja Slack, kde autor zverejní dokument do komunikačného kanálu *#dokumenty*. Povinnosťou ostatných členov je posúdiť dokument podľa definovaných pravidiel a štandardov metodiky. V nástroji Slack sa následne diskutuje o stave dokumentu. Prípadné nájdené chyby sa prekonzultujú. Po nájdení nedostatkov je autor poverený opraviť tieto chyby, prípadne doplniť ďalšie nápady ostatných členov tímu. Autor je poverený spísať dokument do takej podoby, kedy sa zhodne nadpolovičná väčšina tímu, že dokument je možné zverejniť na webové sídlo nášho tímu.

Metodika verziovania

1. Úvod

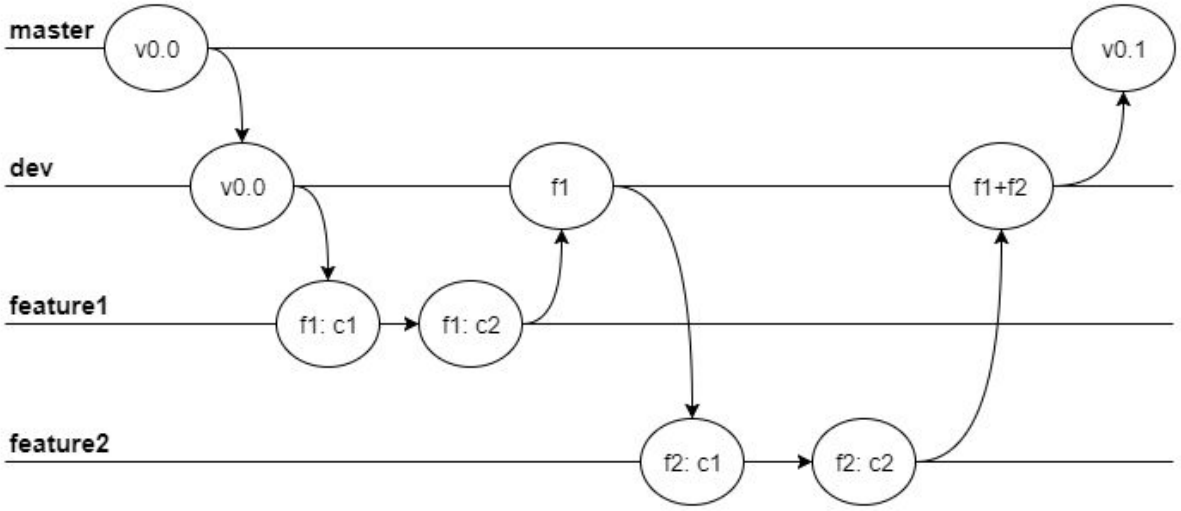
Cieľom metodiky je vyjasnenie si spôsobu práce so súbormi nášho projektu tak, aby každý programátor pracoval na správnom a najaktuálnejšom kóde. Ďalšou veľkou výhodou správneho verziovania je oddelenie práce rôznych autorov pracujúcich na rôznych úlohách (funkciach). Dobrým príkladom je, ak programátor vytvorí chybný kód, tak ostatní programátori pracujúci v iných vetvách nie sú daným chybným kódom zasiahnutí.

2. Slovník pojmov

Pojem	**Význam**
Branch (vetva)	Samostatná verzia projektu. Viaceré vetvy umožnujú paralelnú prácu
Push	Nahranie zmien súborov z lokálneho repozitára (lokálne commity) na vzdialený (remote repozitár)
Pull	Stiahnutie vzdialenej vetvy do lokálneho repozitára
Merge request	Akcia spájania jednej vetvy (napr. s novou funkciou) do inej vetvy
Commit	Zachytenie stavu súborov danej vetvy (Pridanie, odstránenie, zmena súboru)

3. Všeobecné vetvenie

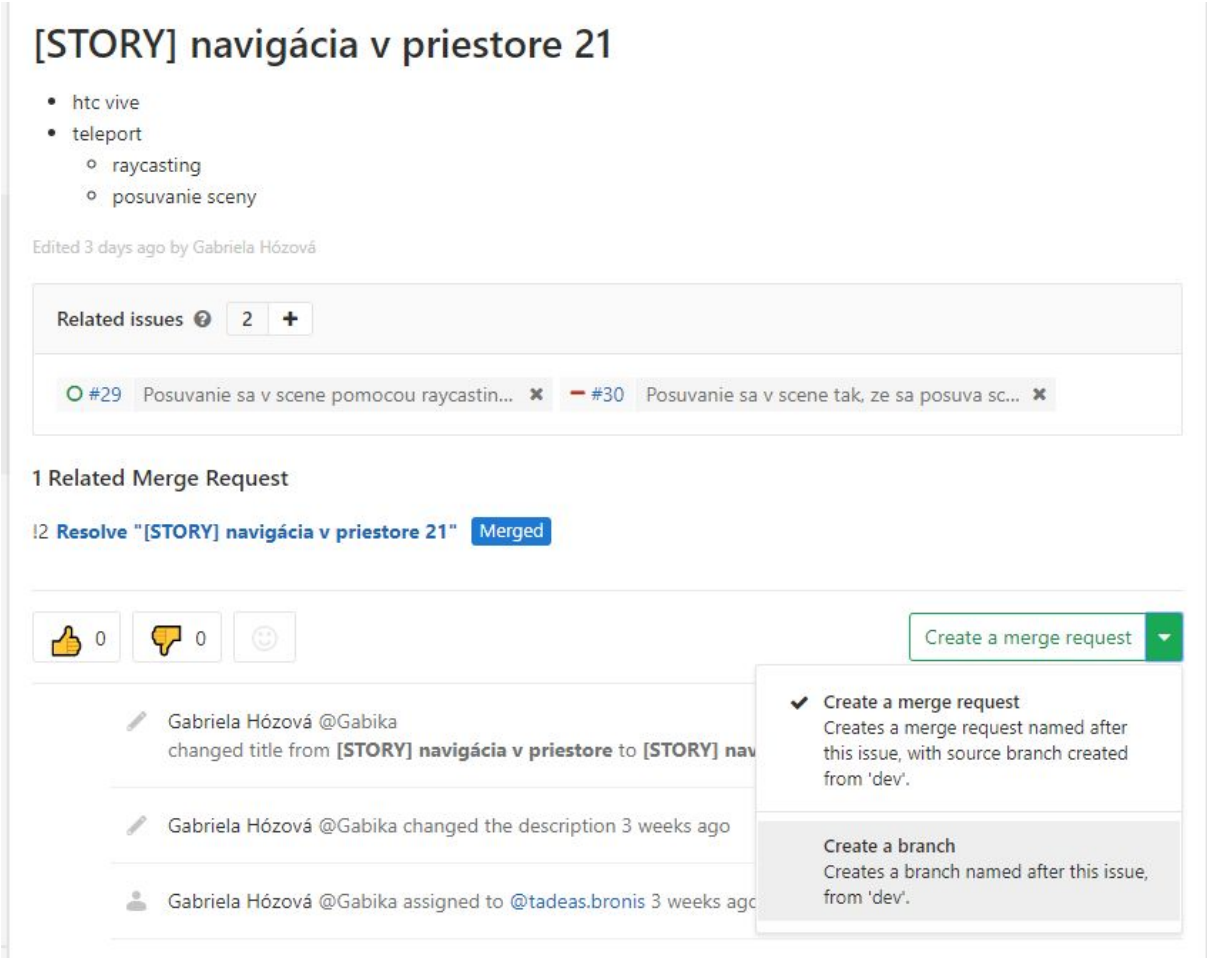
Všetky súbory projektu sú spravované nástrojom git s nadstavbou GitLab, ktorý ponúka prívetivé webové prostredie spájajúce úlohy projektu a správu vetiev projektu. Hlavný projekt je rozdelený na 2 stále vetvy dev a master. Vetva master predstavuje najstabilnejšiu verziu projektu, pričom vetva dev je určená na neustály vývoj. V určitých časových úsekov sa vetva dev bude spájať s hlavnou vetvou master a vznikne tak nová stabilná verzia. Okrem týchto 2 vetiev sa pre každú väčšiu (u nás nazývanú story úlohu) vytvorí osobitná vetva pomenovaná podľa úlohy. Na obrázku č. 1 je možná ukážka priebehu vývoja.



Obrázok č. 1: Príklad vytvárania vetiev

4. Vytváranie vetiev

Odporúča sa vytvárať nové vetvy od dev vetvy, zameranej na vývoj. Vytváranie vetiev úzko súvisí s úlohami pre daný šprint. Pre každú úlohu označenú ako story je potrebné vytvoriť novú vetvu pomocou nástroja GitLab podobne ako je na obrázku č. 2.



Obrázok č. 2: Vytvorenie vetvy

Užívatelia si môžu vytvárať aj svoje vlastné vetvy pre svoje menšie úlohy, ale daný postup sa neodporúča, aby sa zabránilo neprehľadnosti.

5. Pridávanie zmien do vetiev (Push)

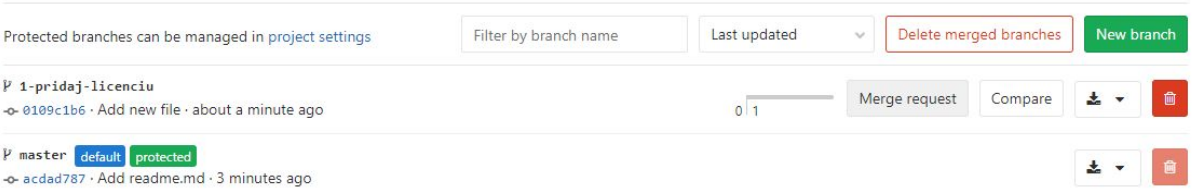
Každý člen tímu sa musí držať týchto pravidiel pri odovzdávaní zmien na vzdialený git repozitár:

- Nevytvárať jeden obrovský commit, ale preferovať viacej commitov podrobne opísaných
- Zmeny treba vždy krátko ale výstižne opísať v commit správe
- Aby sme mohli paralelne pracovať treba často a pravidelne nahrávať všetky zmeny projektu na vzdialený git repozitár
- Všetky zmeny, ktoré riešia určitú úlohu, musia byť nahraté do správnej vetvy, teda vetvy danej úlohy
- Neodporúča sa vytvárať priame zmeny na dev vetve, ale vytvárať merge requesty (spájať) odvodené vývojové vetvy

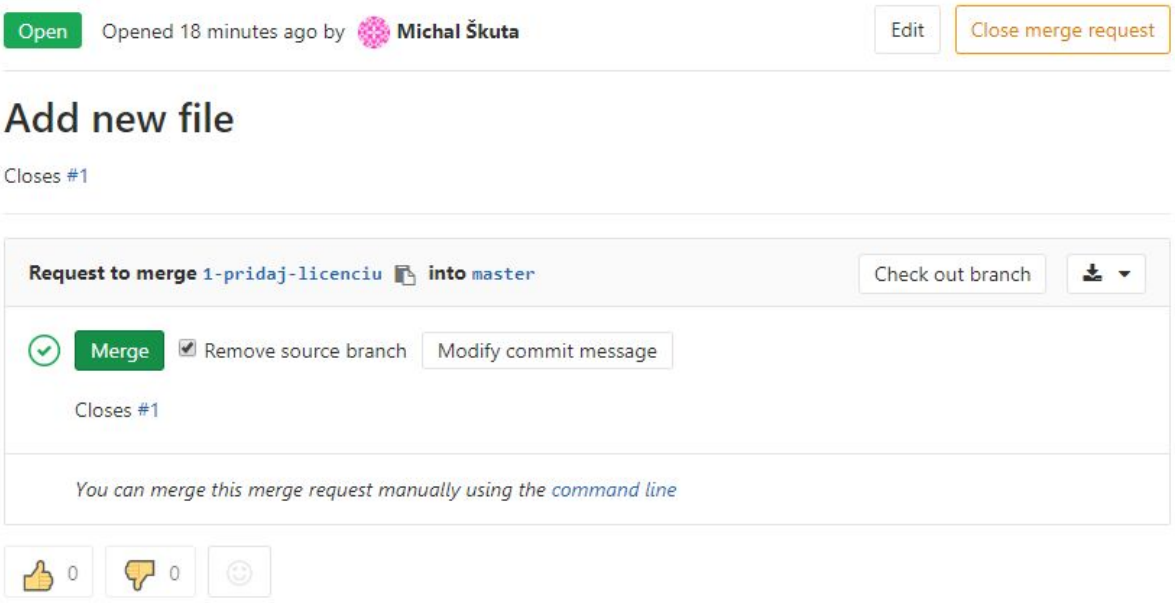
6. Spájanie vetiev (Merge request)

Na obrázku č. 3 je znázornené vytvorenie merge request pomocou webového nástroja GitLab, pričom uzavretie tohto merge request pomocou spájania vetiev je na obrázku č. 4.
Základné pravidlá:

- Odvodená vetva (napr. vetva určitej úlohy) sa môže spojiť z rodičovskou vetvou až keď daná úloha prejde revíziou a testovaním (podrobnejšie opísané v metodike manažmentu úloh)
- Vytvoriť merge request je možné až keď pridelený programátor dokončil programovanie a jeho zmeny sú prípravné na revíziu a testovanie
- v merge requeste je potrebné označiť alebo prideliť ľudí, ktorí budú vykonávať revíziu a testovanie
- Po úspešnom otestovaní sa môže merge request ukončiť spojením vetiev
- Pri akýchkoľvek problémoch pri revízií alebo testovaní je potrebné oznámiť nájdené problémy pridelenému programátorovi a spoločne sa snažiť čo najrýchlejšie problémy odstrániť



Obrázok č. 3: Vytvorenie merge requestu



Obrázok č. 4: Uzavretie merge requestu spojením vetiev

Metodika testovania

1. Úvod

V tejto metodike sa uvádza kedy a akým spôsobom testovať projekt. Cieľom tejto metodiky je ukázať akým spôsobom bude prebiehať testovanie projektu, spolu s konvenciami, ktoré by mali byť pri testovaní dodržané. Samotné testovanie bude realizované akceptačnými testami.

2. Pravidlá testovania

2.1. Roly

Na testovaní sa podieľajú osoby, ktoré môžeme zaradiť do dvoch kategórií:

1. Autor funkcionality: osoba, ktorá zodpovedná za vytvorenie testovanej funkcionality a je tvorcom akceptačných testov
2. Tester: osoba, ktorá sa nijakým spôsobom nepodieľala na tvorbe testovanej funkcionality. Jej úlohou je vykonať kroky definované v testovacom scenári, ktoré spísal autor funkcionality a následné nahlásenie reálnych výstupov aplikácie

2.2. Testovací scenár

Štruktúra testovacieho scenáru musí vyzerať ako v tabuľke č. 1. “Autor funkcionality” musí taktiež spísať kroky testovacieho scenáru podľa vzoru tabuľky č. 2. Je potrebné, aby sa zachovala definovaná konvencia, kvôli dodržaniu stanovených štandardov.

Názov testovacieho scenára	
Test Case ID:	XX
Use Case:	UCXX
Programátor:	Meno programatora
Tester:	Meno testera
Vstupné požiadavky:	Autor kódu v bodoch opíše vstupné podmienky testovania. Napr. Používateľ má k dispozícii HTC VIVE

Výstupné požiadavky:	Autor kódu opíše, aký je očakávaný výstup testovacieho scenáru. Napr. Používateľovi sa zobrazí popis k objektu

Tabuľka č. 1: Štruktúra testovacieho scenáru

Číslo	***Akcie používateľa***	***Očakávaný výstup systému***	***Komentár***
X.	*Namierte na pištoľ*	*Pištoľ sa zvýrazní*	

Tabuľa č. 2: Opis krokov testovacieho scenáru

Časová následnosť

Autor funkcionality napíše testovacie scenáre, aby zabezpečil otestovanie novej, pridanej funkcionality a zároveň aj tej pôvodnej. Následne tester porovná opis očakávaného výstupu s reálnym výstupom aplikácie. V prípade ak sa nezhodujú je nutné, aby reálny výstup popísal do stĺpca “Komentár”. Tester nenavrhuje riešenie pri nekorelujúcich výstupoch. Jeden testovací scenár môžu súčasne vykonávať viacerí testeri, pre zvýšenie kvality testovania.

Pridelenie testovacieho scenára

1. Nová funcionalita prešla revíziou a bola úspešne zlúčená s vetvou “dev”
2. “Autor funkcionality” vytvorí dostatočný počet testovacích scenárov, tak aby zaručil dôkladné otestovanie
3. “Autor funkcionality” vytvorí úlohu (task) v nástroji GitLab pre každý testovací scenár
4. “Autor funkcionality” informuje o vytvorení testovacích scenárov pomocou Slack-u v kanáli “#testovanie” (môže priamo navrhnúť “testera/ov”)
5. “Tester” informuje “autora funkcionality” o tom, že bude vykonávať konkrétny testovací scenár/e
6. “Tester” si priradí úlohu (task) v nástroji GitLab - testovací scenár

Metodika manažmentu úloh

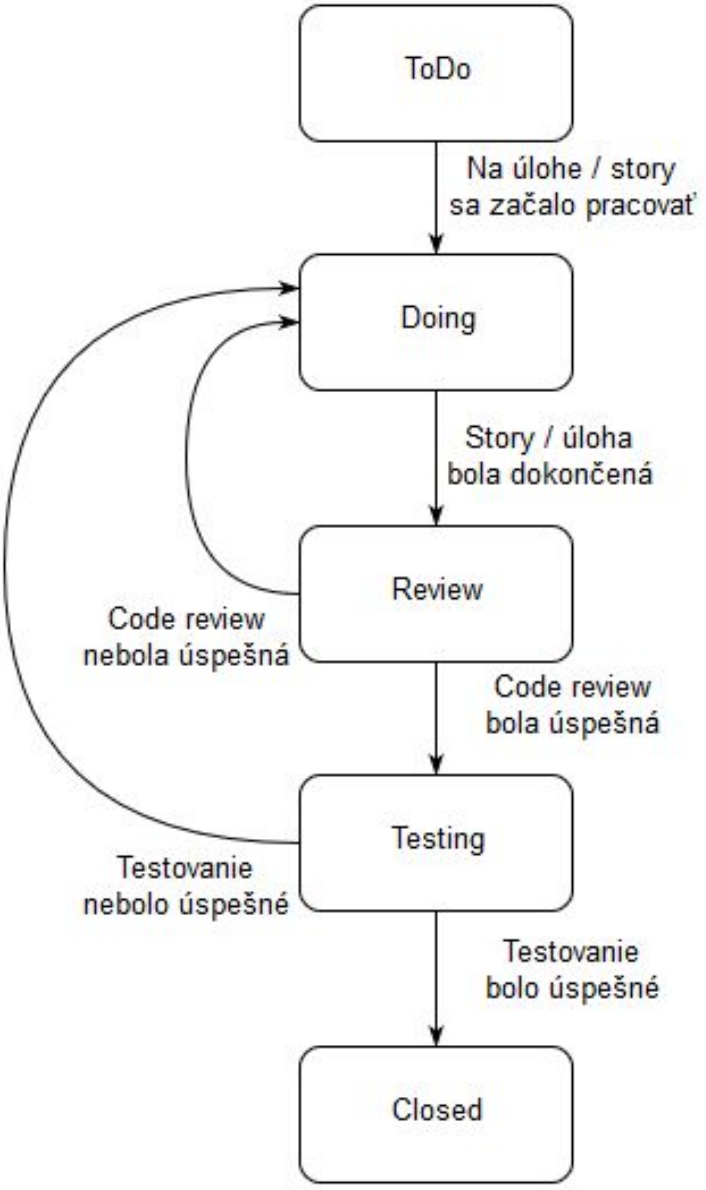
1. Úvod

Táto metodika zahŕňa pravidlá ohľadom správy a manažovania používateľských príbehov (user stories), príbehov (story) a úloh. Za používateľský príbeh sa považuje požiadavka určená zákazníkom, ktorá má byť vo finálnej verzii projektu splnená. Príbeh je časť používateľského príbehu, na ktorom sa pracuje po dobu trvania šprintu. Na začiatku každého šprintu sa vytvorí príbeh, ktorý sa ohodnotí. Bližší opis vytvrárania príbehu je opísaný nižšie. Úloha je najmenší prvok pri práci na projekte. Jeden príbeh sa skladá z niekoľkých úloh. Samotné úlohy sú prideľované členom tímu na vypracovanie počas šprintu. Používame nástroj GitLab, ktorý nám umožňuje určovať stav všetkých problémov (issues), ktoré je potrebné na projekte vykonať. GitLab nám taktiež umožňuje vytvárať určitú hierarchiu medzi používateľskými príbehmi, príbehmi a samotnými úlohami. V tejto kapitole si bližšie opíšeme, ako ich vytvárame, upravujeme, mažeme a nakoniec uzatvárame.

Zákazníkom zadané používateľské príbehy boli usporiadané podľa ich dôležitosti a boli uložené do backlogu. Na začiatku šprintu sa pri vytváraní príbehov rozhoduje na základe dôležitosti používateľských príbehov, pre ktorý je nový príbeh vytváraný. Takýto prístup nám zabezpečí, že sa už od začiatku bude pracovať na úlohách, ktoré od nás zákazník najviac požaduje. Tým, že najdôležitejšie úlohy sa vypracovávajú na začiatku máme dostatok času na konzultovanie všetkých problémov, ktoré by mohli počas vypracovávania nastať.

2. Životný cyklus problému

Na obrázku č. 1 môžeme vidieť stavový diagram príbehu / úlohy na ktorých sa v danom šprinte pracuje.



Obrázok č. 1: životný cyklus problému (príbehu, alebo úlohy)

Po vytvorení príbehu, daný príbeh nadobudne stav „Todo“. Príbeh sa nepovažuje za hotový, dokým úspešne neprejde všetkými krokmi potrebnými na jeho ukončenie. V prípade ak bol príbeh vypracovaný, bola vykonaná revízia a implementovaný kód bol otestovaný, môžeme daný príbeh považovať za uzatvorený.

2.1. Stavy problémov

Pri vytváraní príbehov, ktoré sa majú počas nasledujúceho šprintu vypracovať sa príbehy vytvárajú na základe nadradeného používateľského príbehu. Pri vytváraní príbehu sa pred názov príbehu vloží text „[STORY]“. To a zároveň štítky nám slúžia na odlíšenie príbehov od úloh. Všetky používateľské príbehy sa nachádzajú v kategórii „Backlog“. Počas vypracovávania jednotlivých príbehov je možné meniť ich stav vypracovania.
Na kontrolu stavov všetkých problémov v nástroji GitLab používame štítky, ktorými jednotlivé problémy označíme. V našom projekte používame nasledujúce štítky:

- UserStory - používateľský príbeh zadaný zákazníkom, sú zoradené podľa ich dôležitosti
- Story - samotný príbeh, na ktorom sa počas šprintu pracuje
- ToDo - stav úlohy naznačuje, že je potrebné danú úlohu počas šprintu vypracovať
- Doing - stav úlohy naznačuje, že je daná úloha riešená niektorým členom tímu
- Review - stav úlohy naznačuje, že daná úloha je už dokončená a čaká na kontrolu
- Testing - stav úlohy naznačuje, že daná úloha bola skontrolovaná a je potrebné ju otestovať

Finálnym stavom, do ktorého sa môže úloha dostať je „Closed“. Po úspešnom dokončení úlohy, jej skontrolovaní a otestovaní je úlohe pridelený štítok “Closed” a považuje sa za dokončenú.

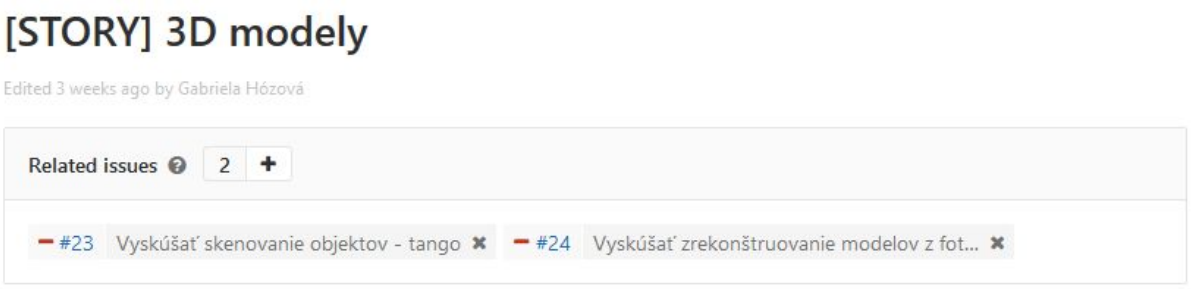
2.2. Hierarchia

Hierarchia vytvorených príbehov a úloh nám umožňuje prehľadnejšie orientovanie sa v systéme. Pri rozbíjaní príbehov na jednotlivé úlohy, popri tom ako pridávame jednotlivé úlohy do systému, ich zároveň aj spájame s nadriadeným príbehom. Takáto hierarchia nám umožňuje jednoduché sledovanie všetkých úloh vyplývajúcich z príbehu, na ktorom sa pracuje. Umožňuje nám jednoducho pozorovať stav všetkých úloh potrebných pre dokončenie daného príbehu. V kapitole nižšie je opísané samotné vytváranie úloh a ich spojenie s príbehom a vytváranie hierarchie.

2.3. Úlohy

2.3.1. Vytvorenie úloh

Vytváranie samotných úloh vyplýva z príbehu, pre ktorý sa dané úlohy vytvárajú. Na začiatku šprintu sa určí jeden, alebo viac príbehov, ktoré chceme počas daného šprintu vypracovať. Jednotlivé príbehy sa ďalej rozbíjajú na čo najmenšie časti, ktoré je možné vypracovať.
Nakoľko ako nástroj na manažment úloh používame GitLab, tak na vytvorenie požadovanej hierarchie používame pridávanie takzvaných súvisiacich problémov (related issue), kde prepojíme hlavný príbeh s jednotlivými úlohami, ktoré musia byť vypracované pre dokončenie daného príbehu.



Obrázok č. 2: Ukážka príbehu so súvisiacimi úlohami

Na obrázku č. 2 vidíme príbeh s názvom „3D modely“. V “related issues” vidíme 2 úlohy, ktoré súvisia s daným príbehom. Pre dokončenie tohto príbehu je potrebné, aby dané úlohy boli vypracované. Pred samotným názvom úlohy vidíme červený znak, ktorý nám symbolizuje, že dané úlohy už boli dokončené a uzatvorené.

2.3.2. Ohodnotenie úloh

Po vytvorení prvotných úloh všetci členovia tímu hlasujú pomocou planning poker kartičiek o náročnosti daných úloh. V prípade, že sa väčšina členov tímu zhodne, že niektorá z úloh je príliš náročná, tak sa daná úloha rozbije na menšie pod-úlohy. O týchto pod-úlohach sa znova hlasuje.

2.3.3. Pridelenie úloh

Prideľovanie úloh sa robí na základe záujmu člena tímu o danú úlohu a na základe jeho skúseností s vypracovaním danej úlohy. Úloha bude pridelená tomu členovi tímu, ktorý má o ňu najväčší záujem, má dostatok skúseností pre vypracovanie danej úlohy a časová náročnosť úlohy mu umožní dokončenie všetkých jemu pridelených úloh.

2.3.4. Ukončenie úloh

Podmienky na uzatvorenie úlohy sú opísané vyššie. Úloha musí splniť všetky kritéria pre jej uzatvorenie a až potom môže byť úloha považovaná za dokončenú, teda môže byť uzatvorená. V prípade ak sa úloha nestihne dokončiť do konca šprintu, je táto úloha presunutá do ďalšieho šprintu.

2.3.5. Úprava úloh

Úprava úloh slúži na zmenu textu úlohy, alebo jej názvu. Využíva sa to najmä v prípade, ak niektorá z už vytvorených úloh je príliš náročná a je potrebné ju rozbiť na menšie úlohy. Taktiež je možné upraviť príbeh. Napríklad v prípade ak vznikla nová úloha, je potrebné súvisiaci príbeh a novú úlohu prepojiť do hierarchie.