Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Tímový projekt
Dokumentácia

Názov tímu: Prostredie na vizualizáciu mikrogridu [GridBox]
Vedúci tímu: Ing. Marek Lóderer
Číslo tímu: 6
Členovia: Martin Činčurák, Michal Ostrodický, František Ďurana, Dávid Pavelka, Peter Pavlík, Richard Mocák, Matej Procházka
Vypracovanie: 2018/19

Obsah

Riadenie projektu

Úvod

Tento dokument opisuje výsledok práce tímového projektu číslo 6 s názvom "Prostredie na vizualizáciu mikrogridu" počas zimného semestra. Dokument je rozdelený na dve veľké časti a to riadenie projektu a inžinierske dielo.

Prvá časť sa skladá z krátkeho predstavenia členov tímu a ich rolí, aplikovaného manažmentu, popisu jednotlivých absolvovaných šprintov a vytvorených retrospektív i metód riadenia, ktoré sme sa postupne naučili používať. Jej súčasťou je i motivačný dokument vytvorený na začiatku semestra. Túto časť uzatvárajú metodiky, ktoré sme v rámci projektu vytvorili.

Druhá časť ponúka pohľad na globálne ciele projektu pre obdobie zimného semestra, celkový pohľad na systém a jeho architektúru i podrobnejšie rozobrané moduly, ich význam a zacelenie do jedného systému. Poslednou časťou tohto dokumentu je príloha obsahujúca zoznam exportovaných úloh zo systému TFS, každá so zodpovedným členom tímu.

Role členov tímu a podiel práce

Náš tím sa skladá zo 7 členov, vedúcim tímu je Ing. Marek Lóderer a produktovým vlastníkom je spoločnosť Sféra a.s. Členovia tímu sú (viac o nás sa dozviete v časti Motivačný dokument):

Martin Činčurák - Serverový master
Manažérske úlohy: Manažment verzií zdrojového kódu
Martin vypracoval v dokumentácií časti: Architektúra systému, Simulačný server

Michal Ostrodický - Databázový špecialista
Manažérske úlohy: Manažment komunikácie
Michal vypracoval v dokumentácií časti: Metodika databáz, Modul databáza, Export prílohy

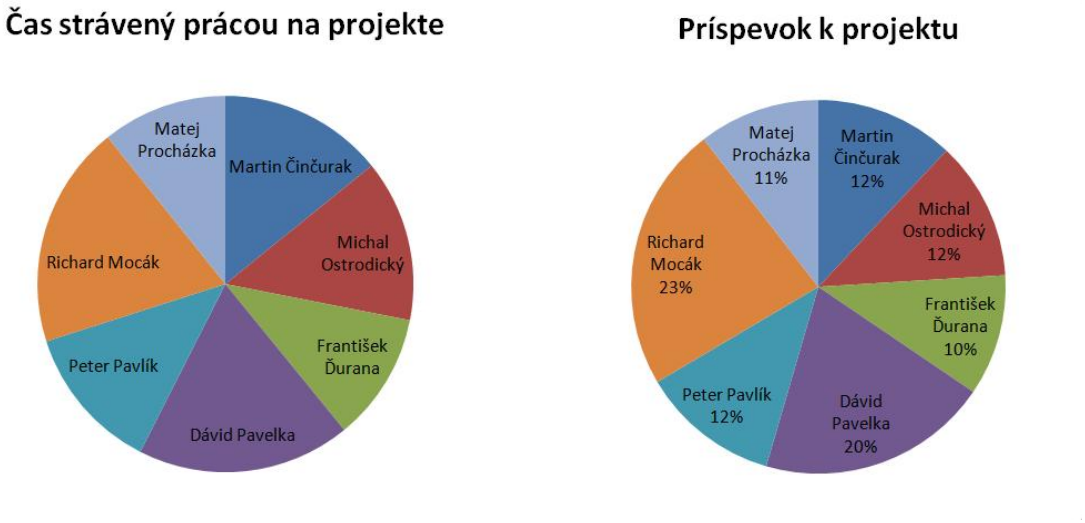
František Ďurana - Majster dokumentarista a podpora
Manažérske úlohy: Manažment dokumentácie
František vypracoval v dokumentácií časti: Úvod, Big Picture, Sumarizácie šprintov

Dávid Pavelka - JavaScript expert + Scrum Master
Manažérske úlohy: Manažment úloh
Dávid vypracoval v dokumentácií časti: Globálna retrospektíva, Aplikácie manažmentov

Peter Pavlík - Dátový majster
Manažérske úlohy: Manažment testovania
Peter vypracoval v dokumentácií časti: Dátový model, Role členov tímu a podiel práce

Richard Mocák - Front-End guru
Manažérske úlohy: Manažment technológií
Richard vypracoval v dokumentácií časti: Webová aplikácia, Automatizovaná integrácia

Matej Procházka - Profesionál na energetiku
Manažérske úlohy: Manažment prehliadok kódu
Matej vypracoval v dokumentácií časti: Code review, Prototyp



Aplikácie manažmentov

V rámci manažmentu využívame viacero podporných programov, konkrétne sa jedná o:

Slack

Slack je kolaboratívny nástroj pre organizácie, ktorý umožňuje:

- Vytváranie špecializovaných kanálov
- Súkromné miestnosti pre chat
- Odosielanie správ, ich formátovanie a vyhľadávanie medzi nimi
- Prispôsobiteľné notifikácie
- Manažment súborového systému

Tento nástroj používame v našom projekte ako hlavný spôsob komunikácie medzi jednotlivými členmi tímu a vedením. Vytvorenie konkrétnych kanálov pre jednotlivé časti manažmentu projektu urýchľuje vyhľadávanie a zvyšuje prehľadnosť dát. Zároveň využívame funkciu notifikácií, aby sme vedeli operatívne riešiť prípadné problémy v tíme. Súkromné miestnosti a vlákna umožňujú výmenu dát medzi členmi tímu, ktorí pracujú na rovnakej časti projektu bez rozposielania týchto informácií ostatným členom tímu.

Team Foundation Server

Team Foundation Server (ďalej už len ako TFS) je systém, ktorý nám umožňuje manažovanie zdrojových kódov, vytváranie reportov, manažovanie projektových činností a automatický build a nasadenie. V rámci nášho projektu je TFS hlavný nástroj pre vytváranie user story s ich cieľmi a opisom. V rámci nich následne vytvárame úlohy, ktoré vedú k naplneniu user story. Platí pri tom, že na každú úlohu aj na každú user story je vždy priradený práve jeden človek, ktorý nesie zodpovednosť za jej splnenie. Identifikované a opísané user story s úlohami sú na začiatku plánovania šprintu zaradené do nového šprintu podľa priority udelenej vlastníkom projektu. Nástroj nám umožňuje označovať aktuálny stav úloh a čas potrebný na ich dokončenie, vďaka čomu vieme rýchlo pozorovať aktuálny stav šprintu.

V rámci TFS servera využívame aj Wiki, kde sa nachádzajú všetky potrebné dokumenty, tie sú tak dostupné na jednom mieste pre všetkých členov tímu. Konkrétne sa tu nachádzajú zápisnice vytvorené na stretnutiach, retrospektívy, metodiky a dokumentácia k častiam samotného vyvíjaného systému.

Sumarizácie šprintov

Náš tímový projekt je rozdelený do šprintov, z ktorých každý pokrýva časové obdobie dvoch týždňov. Vzhľadom na dobu vypracovania dokumentu je za nami prvých 5 šprintov, ktorými sme ukončili práci na tímovom projekte počas zimného semestra.

Šprint 1 - Vanilla

Začiatok šprintu: 8. októbra 2018
Koniec šprintu: 22. októbra 2018

Prvý šprint, ktorý bol zameraný najmä na analýzu dostupných JavaScript knižníc, ktoré v projekte využívame na modelovanie a vizualizáciu mikrogridu. Identifikovali sme 4 vhodné knižnice: Go.js, Mxgraph, Vis.js a D3. Najpodstatnejšie parametre pri výbere boli:

- možnosti perzistencie pozícií (fixná podpora niektorých prvkov v modeli)
- možnosti importu a exportu dát (fungovanie a formát vstupných a výstupných údajov)
- prítomnosť simulačného modulu (či obsahuje výpočtové jadro)
- možnosti vykresľovania kriviek (zobrazovanie kriviek z dátových typov)
- schopnosť prepínania medzi mapovým a schématickým zobrazením
- možnosti agregácie (zgrupovanie prvkov pri zoomovaní)

Na základe analýzy sme vybrali dve vhodné knižnice Go.js a Mxgraph. Tie sme potom podrobili ďalšiemu prerokovaniu s produktovým vlastníkom. Okrem toho sme inicializovali pracovné nástroje a vytvorili prvotné metodiky pre náš projekt. V šprinte sa nám podarilo úspešne splniť všetky úlohy a šprint bol úspešne dokončený.

Sprint 2 - The Burning Crusade

Začiatok šprintu: 22. októbra 2018
Koniec šprintu: 12. novembra 2018

Na základe výsledkov predchádzajúceho šprintu sme sa nakoniec rozhodli využívať knižnicu Go.js. V tomto šprinte sme vytvorili prototyp, ktorý otestoval jej kľúčové vlastnosti pre využitie v rámci nášho projektu. Bolo vytvorené základné používateľské rozhranie s drag-and-drop menu, krivkami a atribútmi prvkov. Okrem toho sme špecifikovali prvotný návrh architektúry systému a dátového modelu projektu. Aj v tomto šprinte sa nám podarilo splniť všetky zadané úlohy a šprint bol úspešne dokončený.

Sprint 3 - Wrath of the Lich King

Začiatok šprintu: 12. novembra 2018
Koniec šprintu: 26. novembra 2018

V treťom šprinte sme pokračovali s rozširovaním prototypu o ďalšiu funkcionalitu, okrem vizualizácie sme implementovali simulačnú a dátovú vrstvu. Simulačná vrstva obsahovala základný model batérie a simulačný server. Pre tento server boli vytvorené API rozhrania ako príprava pre ďalší šprint. Čo sa týka používateľského rozhrania, vytvorili sme GUI pre režim simulácie. Zároveň boli zapracované požiadavky produktového vlastníka po prezentácií prototypu. Pre proces simulácie sme vypracovali návrh architektúry simulačného jadra. Vzhľadom na časové zaťaženie aj zo strany ostatných predmetov sme nestihli úspešne dokončiť všetky úlohy. Šprint z toho pohľadu môžeme označiť ako neúspešný.

Sprint 4 - Cataclysm

Začiatok šprintu: 26. novembra 2018
Koniec šprintu: 10. decembra 2018

Štvrtý šprint pridal mnoho nových funkcionalít z front-endu aj back-endu.
Front-endové zmeny a úpravy:

- Grafické používateľské rozhranie bolo upravené podľa požiadaviek produktového vlastníka
- paleta GO.js prvkov bola nahradená HTML prvkami
- pridali sme hornú lištu a kontextové menu
- zobrazenie simulácie sa rozrástlo aj o agregovanie výsledkov
- vytvorili sme formulár pre zadávanie cenníka pre cenu energie v konkrétnych hodinách

Back-endové zmeny a úpravy:

- Všetky komponenty systému boli navzájom prepojené do jednotnej aplikácie
- Proces simulácie umožňuje odoslanie požiadavky z webovej aplikácie, uloženie simulácie v databáze a následne aj zobrazenie výsledku jednoduchej simulácie vo webovom rozhraní
- Boli pridané obmedzenia na veľkosť simulácie v databáze z dôvodu bezpečnosti a integrity
- Integrácia webovej aplikácie s databázou umožňuje nasledujúce operácie: vytvoriť nový projekt, zmazať projekt, uložiť projekt, načítať projekt, načítať typy elementov a načítať SVG ikony elementov

Napriek tomu, že sme počas tohto šprintu dodali veľké množstvo nových funkcionalít, nepodarilo sa nám splniť všetky zadané úlohy. Na obdobie dvoch týždňov sme si naplánovali príliš veľa činností,

ktoré popri ostatným predmetom nebolo v našich silách dokončiť. Šprint je teda klasifikovaný ako neúspešný.

Sprint 5 - Mists of Pandaria

Začiatok šprintu: 10. decembra 2018
Koniec šprintu: 17. decembra 2018

Posledný šprint, ktorý trval len jeden týždeň bol zameraný na dokončovanie predchádzajúcich úloh, finalizáciu projektov dokumentácie a zapracovanie pripomienok produktového vlastníka.

Globálna retrospektíva

Po skončení šprintu prebiehala na stretnutí retrospektíva šprintu, kde postupne všetci členovia tímu uviedli, čo bolo v šprinte hodnotené ako kladné, s čím naopak prestať a ako vyriešiť vzniknuté problémy.

Šprint 1 (08.10.2018 - 22.10.2018)

V rámci prvého šprintu sme sa oboznámili s technikami a vytvárali sme metodológie. Vznikli však určité problémy, ktoré sme identifikovali a ďalej sa snažili im predchádzať.

S čím prestať?

- Prestať pracovať na cudzích úlohách bez upovedomenia osoby, ktorá je za danú úlohu zodpovedaná.

S čím pokračovať?

- So spoluprácou na komplexnejších úlohách a využití techniky pair programming, vďaka ktorej sa nám podarilo aj pri nedostatočných skúsenostiach niektorých členov úspešne splniť úlohy.

Čo zlepšiť?

- Podrobnejšie špecifikovať user story a úlohy.
- Začať presnejšie trackovať odpracované hodiny jednotlivých členov tímu (spoločný excel sheet)
- Testovanie vytvorených častí systému

Šprint 2 (22.10.2018 - 12.11.2018)

Šprint bol úspešný, tiež sa objavili menšie problémy, ktoré sme v tíme identifikovali s cieľom ich ďalšieho minimalizovania.

S čím pokračovať?

- Vytváranie úloh tak, aby medzi sebou nemali žiadne závislosti
- Komunikácia v tíme, nástroj sa Slack sa osvedčil ako vhodný komunikačný nástroj

Čo zlepšiť?

- Medzišprintovou analýzou ďalších úloh môžeme dopredu vedieť lepšie určiť náročnosť úloh, ktoré nás čakajú

Šprint 3 (12.11.2018 - 26.11.2018)

Na základe získaných poznatkov sme sa v tejto retrospektíve zamerali najmä o zlepšenie stretnutí s projektovým vedúcim, v ktorých sme identifikovali niektoré problematické časti.

S čím prestať?

- Prestať s rozoberaním nedôležitých častí projektu do príliš veľkých detailov, čo nás spomaľuje na stretnutiach

S čím pokračovať?

- Spolupráca na komplexnejších úlohách a rozdelenie členov na úlohy
- Vytváranie nezávislých úloh, aby nemuseli členovia tímu na seba vzájomne čakať

Čo zlepšiť?

- Vytvorenie agendy pre stretnutie s vedúcim projektu a produktovým vlastníkom a dodržiavanie stanovených časových rozhraní
- Sledovanie času a smerovania diskusie počas stretnutí
- Prestávka počas stretnutia po väčšom bloku na oddych
- Asertívnejšie komunikovať s produktovým vlastníkom a ihneď vyjasniť prípadné nezrovnalosti

Šprint 4 (26.11.2018 - 10.12.2018)

S čím prestať?

- Prestať s vytváraním príliš veľkých User Story

S čím pokračovať?

- Rozčlenenie členov tímov podľa ich skúseností pomohlo zvýšiť efektivitu
- Rýchla spätná reakcia zo strany tímu (napríklad spoločné hovory) pomohla pri rýchlom riešení vzniknutých problémov a nejasností
- Časový time boxing stretnutí nám pomohol, aby sa stihlo všetko, čo bolo potrebné vyriešiť ešte priamo na stretnutí

Čo zlepšiť?

- Podrobnejšie si definovať úlohy bez závislosti od ostatných úloh

- Nasadzovanie nových verzií riešiť postupne, pridať jednu a potom ju otestovať, potom zase ďalšiu a pod.
- Dôležité zmeny v projekte komunikovať tak, aby boli čo najskôr dostupné všetkým členom tímu
- V story si definovať aké úlohy sú pre backend a pre frontend
- Definovať maximálny počet úloh na jednu User Story, v prípade presiahnutia tohto počtu rozbiť Story na 2 menšie Story
- Push aj pre menšie zmeny, v prípade veľkých je často komplikované odhaliť, v čom nastala chyba
- Dedikovať url pre plne funkčnú verziu a vlastnú url pre vývojovú verziu
- Vykonávať podrobnejšie code review kódov ostatných členov tímu
- Story si rozdeliť na menšie, aby sa nestávalo, že na jednej robí viacero ľudí naraz

Šprint 5 (10.12.2018 - 17.12.2018)

Zhrnutie retrospektív

Vzhľadom na to, že sme nemali skúsenosti s takouto prácou v projekte, mali sme zo začiatku problémy s definovaním úloh, ich rozdeľovaním a minimalizáciou závislostí. Tento problém sme sa snažili postupne riešiť po retrospektívach na jednotlivých stretnutiach. Postupne sa nám darilo tvoriť menej závislé úlohy, definovať ich a zodpovednejšie ich prideľovať. Problémom, ktorý sme mali bolo aj nepresné odhadovanie náročnosti jednotlivých úloh, postupne sme ich začali rozbíjať na menšie tak, aby bolo jednoduchšie určiť, čo všetko sa za danou úlohou skrýva. Pomohla nám aj medzišprintová analýza úloh pre nasledujúci šprint.

Mentoring, ktorý sme absolvovali nám pomohol pri lepšej organizácií stretnutí s projektovým vedúcim, ktoré sme začali časovo ohraničovať a plánovať jednotlivé úlohy. Na začiatku sa nám totiž často stávalo, že sme kvôli absencií plánu museli predlžovať stretnutia, aby sme stihli potrebné náležitosti. Zároveň sme si postupne osvojili aj komunikačné nástroje a pridali k nim spoločné hovory s cieľom minimalizácie straty času medzi členmi tímu.

Motivačný dokument

Tento motivačný dokument bol odovzdaný počas 1. týždňa semestra.

Tímový mail: tp_06_2018@googlegroups.com

Náš tím je zložený z Martina Činčuraka, Františka Ďuranu, Richarda Mocáka, Michala Ostrodického, Dávida Pavelku, Petra Pavlíka a Mateja Procházku. Všetci sme absolventmi bakalárskeho štúdia na FIIT STU a spolupracovali sme už na mnohých školských zadaniach.

Traja členovia tímu (Ostrodický, Pavlík, Procházka) sa pri riešení bakalárskej práce venovali problémom z oblasti energetiky, článok P. Pavlíka zaoberajúci sa predikciou spotreby elektrickej energie bol ocenený na tohtoročnom IIT.SRC a bude prezentovaný aj na medzinárodnej konferencii. Okrem toho sa ostatní členovia venovali vo svojich prácach aj interakcii gestami v rozšírenej realite (Pavelka), počítačovému videniu (Mocák, Ďurana) a analytike dát z oblasti biomedicíny (Činčurak), takže projekty z tejto oblasti by taktiež nemali predstavovať pre náš tím väčší problém.

Dávid P. a Peter P. boli počas bakalárskeho štúdia členmi výskumne orientovanej skupiny, kde získali znalosti z oblasti vývoja webových aplikácií a dátovej analytiky nad rámec bežného štúdia. Za zmienku stoja najmä skúsenosti získané pri vytváraní webových aplikácií v Ruby on Rails, z ktorých jedna je naďalej používaná na uľahčenie výmeny a evidencie učebníc na základnej škole.

Všetci členovia tímu majú základné skúsenosti s počítačovou grafikou a vytvorením základných konceptov hry, ktoré sme získali počas štúdia s využitím rozhrania OpenGL. Vývoj interaktívnych médií je v súčasnosti vďaka voľne dostupným herným enginom ako Unity alebo Unreal Engine dostupnejší ako kedykoľvek predtým a vidíme v tomto smere potenciál do budúcnosti. Aj keď naše znalosti nepresahujú vypracovanie jednoduchých tutoriálov, radi by sme počas tímového projektu získali skúsenosti aj v tejto oblasti.

Medzi programovacie jazyky, ktoré členovia tímu používajú a majú nadobudnutú znalosť patria Java, Javascript, Python, C++, C#, R, PHP a už spomínaný Ruby. Najviac skúseností majú členovia v jazyku Java, ktoré čerpali zväčša zo školských projektov, P. Pavlík, F. Ďurana a M. Procházka aj z práce popri škole. Programovacie jazyky Python a R sme používali hlavne v bakalárskych prácach so zameraním na dátovú analytiku (Činčurak, Ostrodický, Pavlík, Procházka). Pred vypracovaním praktickej časti títo členovia nemali žiadne skúsenosti s týmito jazykmi. Poznatky sme získali priebežnou prácou, čo ďalej dokazuje naše schopnosti rýchlo sa adaptovať a naučiť na nové, doposiaľ nepoužívané technológie.
F. Ďurana, R. Mocák a D. Pavelka sa vo svojom voľnom čase venujú aj tvorbe webových aplikácií. Používajú pri tom technológie Angular a React pre vývoj používateľského rozhrania prostredníctvom komponentov a Springboot pre reprezentáciu stavu prostredníctvom REST rozhraní.

Motivácia 16: Prostredie na vizualizáciu mikrogridu [GridBox]

Téma číslo 16. vzbudila pozornosť s myšlienkou na pokračovanie a prehlbovanie znalostí v doméne elektriny. Je to dynamická doména, v ktorej vidíme obrovský potenciál na zlepšenie súčasných riešení a najmä príležitosť.

V nástroji Cisco Packet Tracer všetci členovia tímu pracovali a máme predstavu, ako by sme vedeli podobný nástroj implementovať pre potreby energetiky. Plne sa stotožňujeme s cieľom implementovať softvérový nástroj, ktorý pomôže zjednodušiť návrh a optimalizáciu existujúcich sietí. Navyše, vidíme pridanú hodnotu v spolupráci s externou firmou Sféra a.s, ktorá sa špecializuje na inteligentnú grafickú komunikáciu a informačné systémy. Vzájomná spolupráca môže byť pre obe strany prospešná.
Problém, ktorý sa rieši je reálny a umožňuje vytvorenie vlastného originálneho riešenia, čo zvyšuje záujem o túto tému v našom tíme. Toto tvrdenie potvrdzuje aj fakt, že téma skončila v našom výbere medzi najžiadanejšími.
Zo spomenutých technológií máme reálne skúsenosti s jazykom javascript a niektorí z nás aj v jazyku C#. Naše znalosti modelovacieho rámcu Unity nie sú siahodlhé, no na druhú stranu ponúkame zanietenosť a vôľu naučiť sa niečo nové, ktoré považujeme za kľúčové v projekte väčšieho rozsahu.

Motivácia 18: Škola hrou vo virtuálnej realite [VREducation]

Dnes už deti majú prístup k IT zariadeniam a ich obľúbenou činnosťou je v takom veku hranie sa. Aj my sme tak začínali a okrem hrania je skvelé hru aj tvoriť, preto je nám táto téma už na prvý pohľad sympatická. Téma zároveň oproti ostatným ponúka aj hlbší aspekt spoločenského pôsobenia so zameraním práve na mladú generáciu. To je vec, ktorá zaujala na druhý pohľad. Pôsobiť tak, aby sa deti hrali s ich obľúbenými prostriedkami a zároveň mali z toho pridanú hodnotu. Používať nové technológie, s ktorými sme doposiaľ nemali veľké skúsenosti sme si už zvykli, či už v škole alebo v pracovnom živote. S tvorbou hier má každý z nás základné skúsenosti, nadobudnuté počas štúdia na fakulte.

Motivácia 9: Vnímanie neviditeľného [Holographic Eyes]

Pomocou technológie Microsoft Hololens sa dá zlepšiť kvalita mnohých každodenných ľudských aktivít a preto od samého vzniku vzbudila v nás veľký záujem a vyvolala záujem o prácu s ňou.
Každý z nás má potenciál vytvoriť niečo, čo by pomohlo ostatným ľuďom zlepšiť ich životy. Téma taktiež ponúka možnosť pracovať s mnohými modernými technológiami, o ktorých si myslíme, že ich poznanie prinesie veľkú pridanú hodnotu. Väčšina z nás už pracovala s technológiami, ktoré sa dajú využiť na tomto projekte, ako sú neurónové siete (TensorFlow,Python), s počítačovým videním alebo aj s programovacím jazykom C#. Umelá inteligencia je oblasť, s ktorou každý z nás pracoval a chcel by prehĺbiť svoje vedomosti. Pracovanie na projekte, ktorý by mal pomôcť slabozrakým a nevidiacim, nám umožní vidieť svet inými očami.

Priorita tímových projektov:

Prostredie na vizualizáciu mikrogridu [GridBox]	1
Škola hrou vo virtuálnej realite [VREducation]	2
Vnímanie neviditeľného [Holographic Eyes]	3
Podpora výskumu behaviorálnej biometrie [behametrics-learn]	4
Vizualizácia softvéru vo virtuálnej a rozšírenej realite (Remake) [VizReal]	5
Analýza správania sa vozidiel v meste [SmartMobility]	6
Monitoring antisociálneho správania [MonAnt]	7
Vyhľadávanie pomocou obrázkov [ImageSearch]	8

Monitorovanie a vyhodnocovanie fyziologických procesov človeka [BioMonitor]	9
3D simulovaný robotický futbal [3D futbal]	10
IoT systém monitorovania osôb [Breyslet]	11
Generátor 3D priestoru [3DSpaceGen]	12
Inteligentný importér verejných dát [Importer]	13
Prostredie pre inteligentnú analýzu textov [TxtEnv]	14
Databanka otázok a úloh [FIIT - DU]	15
3D UML, improved version [3D-UML]	16
Automatické testovanie v prostredí Internetu vecí [IoTesting]	17
In-memory databáza s využitím GPU [In-memory-DB]	18
Identifikácia entít – spracovanie textu [SK-CZ-TEXT]	19
WiFi Funtoro [WFuntoro]	20

Rozvrh tímu:

https://docs.google.com/spreadsheets/d/10UypZCxbQ4x6elKdQ36uX_BYegYnC82euGqpW648YR8/edit?usp=sharing

Metodiky

Táto časť obsahuje vytvorené metodiky v rámci nášho tímu.

Metodika komunikácie

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, standup volania v google hangouts, na spoločnom stretnutím tímu v stredu, pomocou komentárov v zdrojovom kóde v nástroji TFS(Team Foundation Server), ktorý je využívaný na manažment zadaných úloh.

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
- zapisnice - 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
- random - kanál random slúži na komunikáciu, ktorá nemusí byť priamo spojená s témou projekt
- tfs-notifications- kanál automaticky generuje správy z TFS týkajúce sa pull requestov témou projektu
- teambuilding- v kanáli sa riešia stretnutia členov tímu za účelom otužovania kolektívu

TeamFoundationServer

Komunikácia v nástroji TeamFoundationServer pozostáva z komentárov ku konkrétnym úlohám, kde členovia tímu riešia problémy alebo nejasnosti ohľadom úlohy. Taktiež sa riešia aj komentáre ku code review na konkrétnom pull requeste.

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 v stredu, 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 tp_06_2018@googlegroups.com je využívaná na kontakt s okolím.

Metodika dokumentácie

Počas práce na projekte vznikajú viaceré formy sprievodnej dokumentácie, ktorých jednotnú štruktúru a konzistentnosť zabezpečuje dodržiavanie tejto metodiky. Nasledovanie nižšie uvedených pravidiel je povinné pre všetkých členov tímu. V rámci dokumentácie rozoznávame generovanú technickú dokumentáciu, ktorá vzniká zo zdrojových kódov a úzko súvisí s Metodikou písania zdrojového kódu. Dokumentácia, ktorá vzniká samostatne musí dodržiavať nasledujúce pravidlá:

Štýl vytvárania dokumentácie:

Pri tvorbe sprievodných dokumentov musia byť dodržané nasledujúce pravidlá:

- Pre hlavné kapitoly dokumentácie používať Nadpis 1
- Pre podkapitoly dokumentácie používať Nadpis 2
- Pre menšie obsahové jednotky používať Nadpis 3
- Pre technické časti a ukážky kódov obsiahnuté v dokumentácií používať sivé podfarbenie
- Pre parametre jednotiek používať kurzívu
- Typ a veľkosť písma musí zodpovedať typu vytváraného dokumentu

Zverejňovanie dokumentácie:

Všetka vytvorená dokumentácia musí byť dostupná na Wiki záložke TFS servera (https://tfs.fiit.stuba.sk:8443/tfs/StudentsProjects/Gridbox/_wiki/wikis), kde majú k nej prístup všetci členovia tímu. Pri vytváraní dokumentácie na Wiki je vyžadovaný formát Markdown, ktorý je potrebné dodržať podľa tak aby boli splnené vyššie uvedené pravidlá. Súbor musí byť vecne pomenovaný, aby z jeho názvu bolo jasné, aký obsah sa v ňom nachádza. Dokument je potrebné umiestniť do jedného z pripravených adresárov, podľa toho, akou problematikou sa zaoberá. Každý autor je povinný predtým ako nahrá svoj text si ho skontrolovať a opraviť gramatické chyby a preklepy.

Revízia dokumentácie:

V prípade, že je niektorý z vytvorených dokumentov potrebné revidovať, vykoná zodpovedný člen tímu revíziu dokumentu, pričom musí stále dodržiavať pravidlá pre dokument. V prípade, že revízia obsahuje dôležité zmeny, upovedomí o tom aj ostatných členov tímu (konkrétny spôsob a typ komunikácie podľa Metodiky komunikácie.

Formát dokumentácie:

V prípade rozsiahlejšieho dokumentu (ktorého rozsah presiahol 10, prípadne menej ak si to vyžaduje konkrétny dokument) strán je nutné na mať na začiatku vygenerovaný obsah jednotlivých častí dokumentu. Ak sa v dokumentácií vyskytujú skratky alebo pojmy, ktorých význam nie je jasný, je potrebné uviesť ich vysvetlenie.

Šablóny opakovane vytváraných dokumentov

Počas tímového projektu prebieha opakované vytváranie niektorých dokumentov. Tie musia dodržiavať šablóny uvedené nižšie.

Šablóna pre zápisnice zo stretnutí z vedúcim:

Zápisnica zo stretnutia s vedúcim dd.mm.rrrr, tím 06

Čas stretnutia: od - do vo formáte (hh:mm)

Miesto stretnutia: miesto, kde prebehlo stretnutie

Téma stretnutia: Téma stretnutia

Zápisnicu vypracoval: meno člena tímu, ktorý zápisnicu vypracoval

Prítomní:

zoznam prítomných členov tímu na stretnutí

Neprítomní:

zoznam neprítomných členov tímu na stretnutí

Zápisy zo stretnutia:
jednotlivé zápisy zo stretnutia vymenované v bodoch

Vytvorená zápisnica je pomenovaná ako Zápisnica X - dd.mm.rrrr (kde X reprezentuje číslo zápisnice a dd.mm.rrrr dátum stretnutia, z ktorého daná zápisnica vznikla).

Šablóna retrospektívy

X. Retrospektíva dd.mm.rrrr, tím 06

Prítomní:

zoznam prítomných členov tímu na stretnutí

Neprítomní:

zoznam neprítomných členov tímu na stretnutí

S čím prestať?

zoznam toho, s čím je potrebné skončiť

S čim pokračovať?

zoznam dobrých zvykov a činností, v ktorých chceme pokračovať

Čo zlepšiť?

zoznam toho, čo je potrebné zlepšiť

Vytvorená retrospektíva je pomenovaná ako Retrospektíva X - dd.mm.rrrr (kde X reprezentuje číslo retrospektívy a dd.mm.rrrr dátum stretnutia, z ktorého daná retrospektíva vznikla).

Metodika úloh

Dedikácia

Táto metodika je určená pre každého člena tímového projektu, vrátane produktového vlastníka. Definuje roly a zodpovednosti v rámci manažmentu úloh.

Vymedzenie pojmov a definícií

Product Backlog

Je nástroj na definovanie požiadaviek produktového vlastníka voči vývojovému tímu. Obsahuje zoznam požiadaviek a úloh usporiadaných podľa priority doručenia produktového vlastníka. Požiadavky/úlohy sú spravidla v jednej z troch úrovní a to Feature, User Story, a Task opísané nižšie.

Feature

Je high level požiadavka produktového vlastníka opisujúca črty doručovaného produktu. Jedna Feature spravidla obsahuje viac User Stories. Dĺžka trvania Feature nie je časovo ohraničená.

User Story

Je úloha, ktorej cieľom je naplniť časť požiadaviek jej nadradeniej Feature. Jedna User Story spravidla prislúcha jednej Feature a zároveň obsahuje viac Taskov. Rozsah User Story by mala byť definovaná tak, aby ju bolo možné doručiť v rámci jedného šprintu.

Task

Je konrétna úloha, ktorej cieľom je napĺňať akceptačné kritériá jednej User Story. Task spravidla prislúcha jednej User Story. Task by mal byť definovaný tak, aby ho bolo možné dokončiť za jeden pracovný deň.

Definition of done

- Feature: Ak je dodaná všetka funkcionalita opísaná vo Feature a všetky User Stories k nej prislúchajúce sú hotové, po dohode s produktovým vlastníkom možno Feature uzavrieť.
- User Story:

1. Všetky prislúchajúce Tasky sú v stave Done
2. Sú splnené všetky akceptačné kritériá
3. Je otestovaná
4. Je akceptovaná produktovým vlastníkom

- Task:

1. Úlohy definované v Tasku sú splnené.
2. Prislúchajúce pull requesty sú mergnuté v master vetve

Roly a zodpovednosti

Tím

Skladá sa zo všetkých členov tímového projektu dodávajúcich produkt.

- Povinnosti:

1. Aktívne sa zapájať do stretnutí s produktovým vlastníkom
2. Aktívne sa podieľať na došpecifikovaní User Stories
3. Ohodnocovať došpecifikované User Stories
4. Vytvárať Tasky pre User Stories
5. Efektívne si rozdeľovať úlohy
6. Plniť stanovené úlohy
7. Diskutovať a seba-reflektovať výkonnosť v rámci retrospektív

- Zodpovedný za:

1. Dodávaný produkt

SCRUM Master

- Povinnosti:

1. Sledovať a vynucovať dodržiavanie metodík SCRUM-u
2. Viesť stretnutia tímu s produktovým vlastníkom
3. Zapisovať požiadavky/úlohy počas stretnutí do manažovacieho nástroja

- Zodpovedný za:

1. Harmonickú kolaboráciu tímu a produktového vlastníka

Product Owner

- Povinnosti:

1. Definovať Features dodávaného produktu
2. Vytvárať a diskutovať Uset Stories
3. Prioritizovať Uset Stories
4. Schvaľovať User Stories a časti dodávaného produktu

- Zodpovedný za:

1. Jasnú špecifikáciu požiadaviek

Metodika integrácie a dodávania softvéru

Integrácia dodávania softvéru opisuje proces integrácie a dodávania softvéru zo zdrojových kódov v repozitári do bežiacej aplikácie prístupnej pre cieľových používateľov.

Dedikácia

Metodika je určená pre všetkých členov tímu, špeciálne pre Technického vedúceho a technických správcov, ktorí sa aktívne podieľajú na procese integrácie a dodávania softvéru.

Roly

- **Technicky vedúci (Tech leader)**
  Zodpovednosti: Administrácia konfigurácie v nástroji, Jenkins, Administrácia konfigurácie nástroja Docker, poveruje Technických správcov, rokuje o riešeniach integrácie a dodávania softvéru s produktovým vlastníkom, Aktualizuje a spravuje Metodiku integrácie a dodávania softvéru.
- **Technický správca**
  Člen tímu poverený Technickým vedúcim, ktorý má práva pre prístup a zmenu konfigurácie pre vedúcim určený komponent projektu. Zodpovednosti - Po dohode s Technickým vedúcim upravuje konfiguráciu integrácie a nasadenia, ak nie je proces integrácie automatiovaný , tak po dohode s Technickým vedúcim vykoná ručne proces integrácie a nasadenia pre daný komponent projektu.

Vymedzenie pojmov a definícií

Komponent projektu - Softvérový produkt Gridbox pozostáva z 3 komponentov (WebApp - webová aplikácia a používateľské rozhrania, DB Server - databázový server s REST službami pre perzistenciu dát, Simulation Server - simulačný server s REST rozhraniami pre výpočty a simuláciu). Architektúra systému je detailne opísaná v dokumente Architektúra. Systém využíva architektonický štýl Mikroslužieb a tieto komponenty sa nachádzajú v samostatných repozitároch. Každý komponent je samostatne spustiteľný a nasaditeľný do ďalšej fázy nasadenia.

Fáza nasadenia - Každý komponent projektu prechádza fázami nasadenia:

- *Nasadené vo vývoji*: Počas vývoja na komponente projektu je komponent nasadení v Vývojovom prostredí na vývojovom serveri. Slúži na demo pri stretnutiach s produktovým vlastníkom, vývoj novej funkcionality projektu a integračnom testovaní komponentov projektu.
- *Testovanie pred produkciou*: Po dohode s produktovým vlastníkom je verzia komponentu projektu presunutá do fázy testovania pred produkciou, kde je komponent projektu nasadený na server produktového vlastníka, alebo je po dohode s produktovým vlastníkom určené inak. Testovanie prebieha v réžii produktového vlastníka. Cieľový používateľ softvérového produktu nemá prístup k produktu v tejto fáze nasadenia.
- *Nasadenie v produkcií*: Po úspešnom ukončení fázy testovania pred produkciou je komponent projektu nasadený do produkcie. Jedine v tejto fáze má Cieľový používateľ prístup k softvérovému produktu.

Využívané technológie

Jenkins - nástroj pre konfiguráciu integrácie a nasadenia jednotlivých komponentov projektu, umožňuje rozšírenie o prepojenie s verziovaním softvéru (git), kontajnerizáciu komponentov (docker) ale aj počúvanie na prichádzajúce zmeny (Team Foundation Server)

Docker - nástroj pre kontajnerizáciu softvérových komponentov vhodný pre vývoj aplikácií s mikroslužbovou architektúrou

Linux Ubuntu - operačný systém pre aplikačné serveri (vo vývoji, test pred produkciou a aj produkciu)

Integrácia a dodávanie pre jednotlivé komponenty projektu:

WebApp

Pre tento komponent je vytvorený proces automatizovanej kontinuálnej integrácie v nástroji Jenkins. Konfigurácia procesu je definovaná v súbore Jenkinsfile, ktorý sa nachádza v koreňovom priečinku repozitáru. Konfigurácia využíva rozšírenie *Jenkins Pipeline*, ktoré umožňuje definovať automatizovanú integráciu ako postupnosť krokov. Integrácia je delená do fáz (a angl. Stage), ktoré spájajú spolu súvisiace kroky integrácie (z angl. Steps). Spustenie automatizovanej integrácie nastane po zaregistrovaní akcie *push* pre špecifikované vetvy z repozitára z TFS. Kontinuálna integrácia zahŕňa aj vytvorenie Docker kontajnerov, ktorých konfigurácia sa nachádza v súbore Dockerfile v koreňovom priečinku repozitára.

Fázy kontinuálnej integrácie pre komponent WebApp:

1. Fetch dependencies - načítanie nevyhnutných knižníc pre build Angular aplikácie do výstupných súborov.
2. Compile - kompilácia súborov z TypeScript na JavaScript a build Angular aplikácie do výstupných súborov s produkčnou konfiguráciou.
3. Build and Push Docker Image - kroky pre vytvorenie Docker kontajneru pre tento komponent a pridanie novej značky s verziou pre tento kontajner a odoslanie do registra kontajnerov pre komponent WebApp.

Proces nasadenia komponentu aktuálne je potrebné vykonať manuálne nasadenie vykonávané na určenom serveri podľa fázy nasadenia (Vo vývoji, Testovanie pred produkciou alebo Produkcia).

Kroky manuálneho nasadenia:

1. Stiahnutie kontajneru z registra podľa požadovanej verzie komponentu (značka kontajneru)
2. Ak aktuálne beží kontajner webovej aplikácie na serveri je potrebné zastaviť a odstrániť bežiacu inštanciou Docker kontajneru.
3. Spustenie novej inštancie Docker kontajneru z novo stiahnutej verzie.

DB Server

Momentálne pre tento komponent neexistuje automatizovaná metóda a je potrebné manuálna integrácia a proces nasadenia komponentu je potrebné vykonať manuálne na určenom serveri podľa fázy nasadenia (Vo vývoji, Testovanie pred produkciou alebo Produkcia).

Kroky manuálnej integrácie:

1. Vytvorenie Docker kontajneru pre tento komponent a pridanie novej značky s verziou pre tento kontajner
2. Odoslanie vytvoreného kontajneru do registra kontajnerov pre komponent DB Server.

Kroky manuálneho nasadenia:

1. Stiahnutie kontajneru z registra podľa požadovanej verzie komponentu (značka kontajneru)
2. Ak aktuálne beží kontajner webovej aplikácie na serveri je potrebné zastaviť a odstrániť bežiacu inštanciou Docker kontajneru.
3. Spustenie novej inštancie Docker kontajneru z novo stiahnutej verzie.

Simulation Server

Momentálne pre tento komponent neexistuje automatizovaná metóda a je potrebné manuálna integrácia a proces nasadenia komponentu je potrebné vykonať manuálne na určenom serveri podľa fázy nasadenia (Vo vývoji, Testovanie pred produkciou alebo Produkcia).

Kroky manuálnej integrácie:

1. Vytvorenie Docker kontajneru pre tento komponent a pridanie novej značky s verziou pre tento kontajner
2. Odoslanie vytvoreného kontajneru do registra kontajnerov pre komponent Simulation Server.

Kroky manuálneho nasadenia:

1. Stiahnutie kontajneru z registra podľa požadovanej verzie komponentu (značka kontajneru)
2. Ak aktuálne beží kontajner webovej aplikácie na serveri je potrebné zastaviť a odstrániť bežiacu inštanciou Docker kontajneru.
3. Spustenie novej inštancie Docker kontajneru z novo stiahnutej verzie.

Metodika testovania

Testovanie vytvoreného softvéru je nevyhnutnosťou pre akýkoľvek rozsiahly softvérový projekt. Každá zmena softvéru má potenciál ovplyvniť aj funkcionalitu ostatných súvisiacich komponentov a tak je nevyhnutné, aby bola správnosť funkcionality softvéru testovaná po akejkoľvek zmene a úpravy boli schválené až po takomto overení.

Vymedzenie pojmov

Jednotkový test - test najmenšej možnej zložky softvéru, spravidla vstupu a výstupu jednej metódy.

Integračný test - test spoločného fungovania niekoľkých komponentov softvéru.

Štruktúra testov

Všetky testy sú uložené v zložke /tests príslušného komponentu, odkiaľ sa dajú skupinovo spúšťať jedným príkazom (npm test pre webapp a pytest pre servery). Testy pre webovú aplikáciu využívajú test runner Karma, musia byť uložené v zložke tests a dodržiavať menovaciu konvenciu *.spec.js. Testy pre webovú aplikáciu využívajú balík PyTest, musia byť taktiež uložené v zložke tests a dodržiavať menovaciu konvenciu test_*.py.

Testy slúžiace na pokrytie jednej funkcionality by mali byť súčasťou samostatného súboru (testovacieho skriptu), ktorý môže obsahovať niekoľko testovacích scenárov slúžiacich na overenie správnosti pri rôznych vstupoch, postupnostiach akcií a rôznych okrajových prípadoch.

Pokyny pre autorov zmien

Po vykonaní akýchkoľvek zmien v zdrojovom kóde komponentu sa od autora vyžaduje, aby spustil všetky dostupné testy pre daný komponent. Pre databázový a simulačný server to obnáša spustenie integračných (pytest) a jednotkových testov (unittest), pre webovú aplikáciu iba unit testy (karma). Až po úspešnom behu testov môže vývojár vytvoriť pull request. Ak vykonané zmeny spôsobili zastaranie starých testov, mali by byť aktualizované tak, aby zodpovedali správnemu správaniu po vykonaní zmien. Ak testy zlyhávajú na funkcionalite zdanlivo nesúvisiacej s nedávnymi zmenami, je potrebné o tom informovať ostatných členov tímu, od ktorých sa očakáva, že pomôžu s vyšetrením zlyhaní zabraňujúcich mergu zmien.

Od autora zmien sa zároveň očakáva, že spolu s nimi rozšíri testovaciu sadu o nové testy, ktoré budú overovať správne fungovanie rozšírení kódu. Nie je to však nutnosťou v každom prípade (napr. pri časovej tiesni alebo netriviálnosti testovania) a je na zvážení reviewera, či po zdôvodnení umožní zmeny merge-núť. V takomto prípade by mal však byť o chýbajúcom pokrytí upovedomený manažér testovania, ktorý si potrebu testu zapíše a vytvorí úlohu na doimplementovanie.

Code conventions

Pomenovanie premenných a funkcií

- Názov premenných, funkcií vyskytujúcich sa v kóde písané **v angličtine**
- Viac slovné premenné a funkcie sú písané štýlom camelCase (napr. defaultZoom)
- Všetky premenné majú začínať písmenami
- Globálne premenné pomenované UPPERCASE
- Konštanty pomenované UPPERCASE - PI

Pravidlá písania if statemets

- Jednoduché (jednoriadkové) telo podmienky je tiež uzatvorené v zátvorkách { }
- Otváracia zátvorka { na konci prvého riadku
- Medzera pred otváracou zátvorkou {
- Medzera pred otváracou zátvorkou (

```
// Example: If statements
if (condition) {
  singleLineCommand();
}
```

Objektové premenné

- Otváracia zátvorka nech je v rovnakom riadku ako je názov objektu
- Použitie dvojbodky a medzery medzi key a value v objekte
- Zatváracia zátvorka objektu v novom riadku
- Páry key a value sú oddelené novým riadkom a čiarkou je pridaná iba ak je to nevyhnutné
- Po zatváracej zátvorke bodkočiarka
- `const` pre premenné bez reinicializácie
- `let` pre premenné s reinicializáciou

```
// Example: Object variables
const bicycle = {
  color: 'red',
  size: 'big'
}

// OK
bicycle.size = small

// WRONG
bicycle = {
  wheel: false
}


let car = {
  model: 'BMV'
}

// OK
car.model = 'Alfa Romeo'
```

```
// OK
car = {
  model: 'Mercedens-Benz'
}
```

Komentáre

- Jednoduché zrozumiteľné
- Používanie riadkových komentárov nie blokových
- Zarovnané vždy doľava
- Samotný obsah komentára je oddelený od značky komentára medzerou

Maximálna dĺžka riadku

- Každý riadok kódu ma maximálne dĺžku 100 znakov
- Presahujúci kód je potrebné presunúť do nového riadku
- Ak riadok presahuje maximálnu dĺžku, tak vstupné argumenty budú oddelené čiarkou a novým riadkom a otváracia zátvorka (sa nachádza hneď za volaním a záverečná zátvorka) sa bude tiež nachádzať v novom riadku

```
// Example: Max line length
print(
  firstArgument,
  secondArgument,
  thirdArgument,
  fourthArgument,
  fifthArcgument,
);
```

Názvy súborov

- Slová v názve súboru sú oddelené čiarkou
- Názov obsahuje aj typ podľa obsahu a kódu v súbore (pr. component, model, service, ...)
- Názov, typ obsahu a formát sú oddelené bodkou (pr. `right-panel.component.ts`, `right-panel.component.html` a `project.model.ts`)

Ostatné

- V aritmetických operáciách sú znamienka oddelené medzerami
- Ukončiť riadok kódu / príkaz bodkočiarkou
- Na riadok maximálne jeden riadok kódu

Metodika manažovania verzií (Git)

Účelom tejto metodiky je zadefinovať pravidlá podľa ktorých sa riadi nástroj na manažovanie verzií zdrojového kódu. Definuje pravidlá vytvárania a zlučovania vetiev. Metodika je záväzná pre všetkých členov tímu, ktorý pracujú so zdrojovým kódom. Na manažovanie verzií sa používa nástroj Git.

Zdrojové kódy sú rozdelené do troch repozitárov:

- WebApp
- SimulationEngine
- DBServer

Repozitár WebApp obsahu zdrojové kódy webovej aplikácie. Zdrojové kódy simulačného serveru sa nachádzajú v repozitáre SimulateEngine. Repozitár DBServer obsahuje zdrojové kódy databázového serveru.

Práca s nástrojom na manažovanie verzií

Každý repozitár ma vetvu master do ktorej sa pridávajú otestované funkcionality z ostatných vetiev. Pre každú novú funkcionalitu sa vytvára nová vetva s menom v tvare **feature/newfeature**. V tejto vetve sa vyvíja nová funkcionalita. Po implementácii a otestovaní novej funkcionality vytvoríme pull request. Ak je pull request schválený zlúčime vetvy.
Ak zistíme chybu v master vetve, vytvoríme novú vetvu s menom v tvare fix/newbug. Podobne ako pri implementácii novej funkcionality, po odstránení anomálii produktu a po schválení pull requestu, vetvu zlúčime s master vetvou.

Počas práce s Gitom musime dodržiavať nasledovné pravidlá:

- Počas vytváraní komitov správy sa píšu v anglickom jazyku.
- Do master vetvy nikdy nerobíme zlučovanie bez toho aby bol schválený pull request.
- Taktiež je zakázané komitovať do master vetvy.
- Nevytvárať jeden obrovský commit, ale preferovať viacej menších commitov podrobne opísaných.

Ak identifikujeme problémov s Gitom, na spoločných stretnutiach sa rozoberá problém a definujú nové pravidlá, aby v budúcnosti neprišlo k rovnakému alebo podobnému problému.

Code review

Pokyny pre autora

Kód môže byť posunutý na prehliadku kódu (code review) až vtedy, keď autor urobil prvotnú kontrolu, prešli všetky testy a skontrolujte si dodržiavanie konvencií pre písanie kódu.
Autor je povinný poskytnúť aspoň základný prehľad urobených zmien a k nim dôvod, prečo sú potrebné.
Recenzent (reviewer) sa bude vyberať najmä na základe znalosti danej oblasti kódu. Ak nie je nikto s prednostnou znalosťou danej oblasti, druhým hlavným faktorom je dostupnosť. Prednosť má člen, ktorý aktuálne nie je recenzentom, prípadne má pridelených najmenej úloh na prehliadku.

Pokyny pre Recenzenta

Každý recenzent je povinný skontrolovať následné časti:

- podrobná kontrola zmenenej logiky systému
    - správnosť
    - efektívnosť
- dodržiavanie konvencií
    - názvy premenných, funkcií
    - zarovnanie
    - písanie if-statement
- čistota kódu
    - čitateľnosť
    - zrozumiteľnosť

Primárna je kontrola logiky, ktorá musí byť najpodrobnejšia a mala by sa vykonávať prvá. Kontrola konvencie a čistoty kódu sú však taktiež dôležité.
Všetky pripomienky budú zaznamenané v rámci pull request-u v systéme TFS, ku konkrétnemu riadku, ktorého sa daná pripomienka týka (ak ide o väčší logický celok, pripomienka bude uvedená na začiatku tohto celku, napr. pri definícií funkcie).
Zaznamenávajú sa ako pripomienky k logike tak aj k dodržiavaniu konvencií a čistote kódu.
V prípade neistoty recenzent najprv kontaktujte autora a až po overení zadá pripomienka v systéme TFS.
Ak je niektorá z pripomienok chybná a nepotrebuje žiadnu reakciu zo strany autora, autor poskytne vysvetlenie k pripomienke v systéme TFS.
V prípade, že je kód príliš ťažký na pochopenie, resp. je príliš nepriehľadný (nízka čistota kódu) recenzent môže zamietnuť daný pull request a vrátiť ho autorovi na úpravu.

Kontrolný zoznam pre recenzenta:

- Úspešné spustenie danej časti kódu
- Úspešný výsledok všetkých testov týkajúcich sa danej časti kódu
- Pochopenie upravovanej logiky
- Evidencia pripomienok v systéme TFS
- Kontrola dodržiavania konvencií a čistoty kódu
- Evidencia pripomienok k čistote kódu v systéme TFS

Database conventions

Všeobecné pokyny

- vždy používať medzeru za dvojbodkou
- prvky v riadkoch oddelené čiarkou
- používanie tabulátora
- nepoužívať podčiarkovník v strede názvu

Názvy kolekcií

- používať plurály a camelCase napr. projects namiesto project
- spájať viac slovné spojenia do jedného - napr. first_name do firstName

Názvy databáz

- názov databázy by nemal mať viac ako 64 znakov
- názov typu camelCase
- nemala by obsahovať niektorý z nasledovných znakov “/, \, ., “, *, <, >, :, |, ?, $,

Názvy polí

- camelCase
- nepoužívať znak _, jediné pole, ktoré môže mať tento znak je id
- názvy by nemali obsahovať bodku, nullové znaky a nemôžu začínať znakom $

Inžinierske dielo

Big picture

Táto časť dokumentu obsahuje informácie o systéme vytvorenom v rámci témy "Prostredie na vizualizáciu mikrogridu". Zdokumentovaná je celková architektúra ale i jednotlivé moduly, z ktorých sa systém skladá.

Globálne ciele projektu na zimný semester

Cieľom projektu vypracovaného počas zimného semestra je vytvorenie prvotnej verzie systému na modelovanie, vizualizáciu a simulovanie mikrogridu. Používateľ bude so systémom pracovať vo webovom rozhraní s využitím knižnice Go.js. Okrem komponentu pre vizualizáciu využívame aj databázu pre správu dát, pričom je používateľovi umožnené načítať vstupy aj zo súborov v jeho zariadení. Aplikácia ponúka dva základné režimy modelovania a to mapové a schématické. Mapové rozhranie umožňuje modelovať nad reálnym mapovým podložím, schématické zase nad mriežkou. Vytvorený model môže používateľ odoslať na simuláciu, ktorej výsledkom sú grafy, vypočítané ceny a spotreby pre celý mikrogrid ako aj pre jednotlivé prvky modelu, ktoré si používateľ pred simuláciou zvolil.

Základná funkcionality je nasledovná:

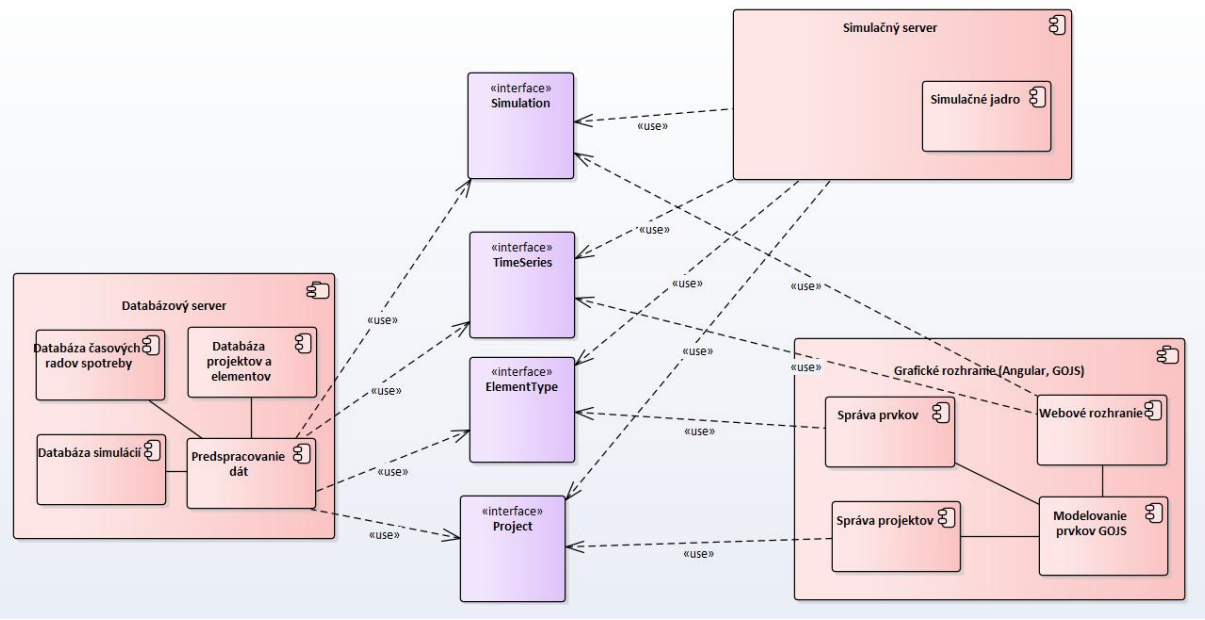
- Modelovanie mikrogridu z dostupných prvkov vo webovom rozhraní
- Implementovanie jednoduchého simulačného servera pre simulovanie mikrogridu
- Implementovanie a prepojenie dátovej vrstvy pre ukladanie výsledkov simulácie a vstupných údajov pre simuláciu ako samostatný komponent systému

Celkový pohľad na systém:

Systém je zložený z viacerých komponentov, ich význam a fungovanie sú podrobnejšie opísané v častiach nižšie.

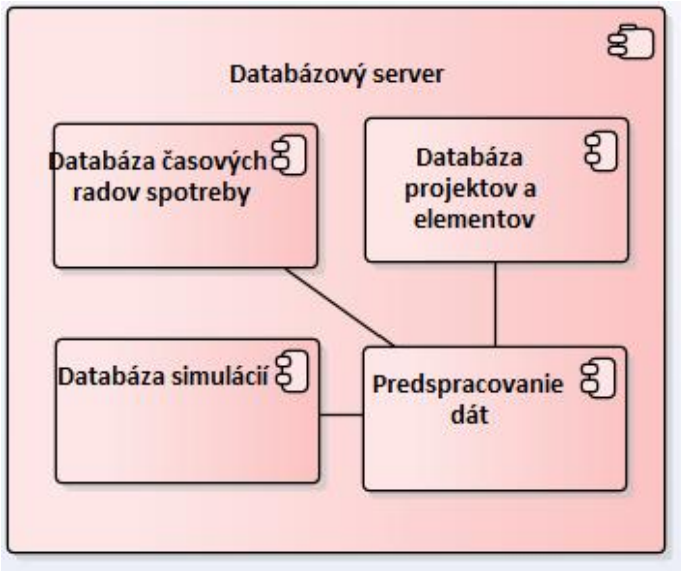
Architektúra systému

Architektúru systému tvoria 3 hlavné časti a to simulačný server, databázový server a samotná aplikácia (grafické rozhranie).



Obrázok 1 Architektúra systému

Komponent: Databázový server:

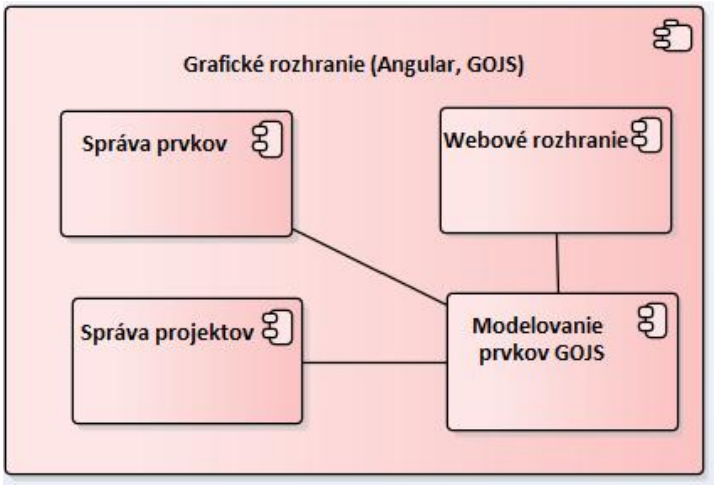


Databáza sa delí na štyri časti:

- Databáza projektov a elementov
- Časové rady spotreby a výroby elektrickej energie uzlov
- Databáza simulácií
- Predspracovanie dát

Časť databázy projekty a elementy obsahuje všetky vytvorené projekty spolu so všetkými elementami, ktoré môžu vystupovať v projektoch (elementy na spotrebu, výrobu a sklad elektrickej energie). V druhej časti databázy budú uložené údaje spotreby alebo výroby energie pre uzly, ktoré tieto aktivity vykonávajú.
Riadenie projektusimulácii uchováva všetky vykonané simulácie na daných projektoch.
Časť databázového serveru je aj predspracovanie dát. Dáta sa predspracovávajú kvôli jednoduchému uchovávaniu a manipulácií s dátami.

Komponent: Grafické rozhranie



Grafické rozhranie pozostáva z 4 častí:

- Správa elementov
- Správa projektov
- Modelovanie prvkov GOJS
- Webové rozhranie

Správa elementov
Tento komponent má na starosti správu elementov.
Komponent podporuje funkcionality:

- Vytvoriť nový element
- Zmazať element (aj hromadne)
- Exportovať knižnicu elementov do súboru (všetky/vybrané užívateľom)
- Importovať knižnicu elementov zo súboru

Základné parametre elementu:

- uniqueid (kvôli opakovanými importom pre aktualizáciu prvku)
- názov
- kategória (prvok/spoj)
- obrázok (ideálne aj možnosť svg)
- rozšírené užívateľské parametre (meniteľné pre každú inštanciu v rámci modelu)
    - názov
    - kategória
    - default hodnota
    - simulačné parametre

Komponent komunikuje s databázou pomocou rozhrania ElementType.

Správa projektov (modelov)
Tento komponent spravuje modely. Zastrešuje nasledovné funkcionality:

- Vytvoriť nový model
- Zmazať model
- Upraviť model
- Skopírovať model ako nový model
- Exportovať model do súboru
- Importovať model zo súboru

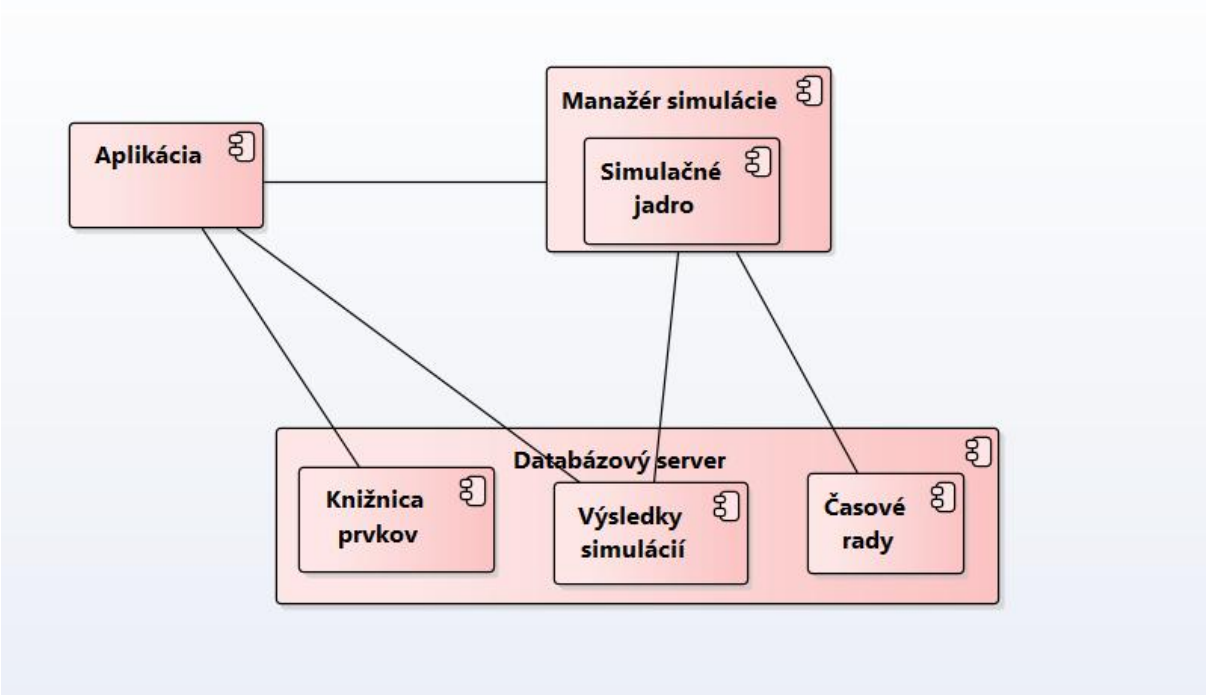
Komponent si z databázy vypýta projekt, následne ho upravuje a posiela webovému rozhraniu na zobrazenie.
Komunikuje pomocou rozhrania Project.

Webové rozhranie (Angular, GOJS)
Webové rozhranie interaguje s používateľom, zobrazuje modely a dovoľuje úpravy.
Modelovacia časť
Modelovacia časť spravuje prvky a prepojenia v grafe. Používa dátový objekt Project, v ktorom vykonáva zmeny a posiela upravený objekt webovému rozhraniu na zobrazenie.
Úpravy modelu:

- Vyčistiť model
- Vložiť prvok na plochu
- Odstrániť prvok z plochy (aj hromadne)
- Presunúť prvok
- Prepojiť dva prvky
- Skopírovať prvok
- Zamknúť polohu prvku na ploche (aj hromadne)
- Odomknúť polohu prvku na ploche (aj hromadne)
- Nastaviť parametre prvku

Proces simulácie

Náš systém ponúka okrem vizualizácie mikrogridu aj možnosť simulácie jeho správania. Táto časť je koncipovaná tak, aby bolo možné vymeniť simulačné jadro za externé jadro s definovanými rozhraniami a zároveň používať ľubovoľnú knižnicu prvkov. Tieto dve požiadavky pramenia z požiadaviek produktového vlastníka na vytváraný systém.



Obrázok 2 Proces simulácie

Postupnosť krokov v rámci simulácie je nasledovná:

1. Aplikácia pošle schému modelu s parametrami do Manažéra simulácie
2. Simulačné jadro si postupne pre dané Id prvkov získajú ich časové rady v databáze
3. Prijaté časové rady sa odošlú do Manažéra simulácie, do Simulačného jadra
4. V Simulačnom jadre prebehne samotný výpočet simulácie, ktorý sa uloží do Databázy pre Výsledky simulácií
5. Manažér simulácie zároveň odošle Id výsledku simulácie do Aplikácie
6. Aplikácia pošle žiadosť do databázy s Id výsledku simulácie
7. Databáza následne odošle výsledné časové rady do aplikácie

Simulačný server

Server sa používa na simuláciu toku elektrickej energie v mikrogride. Simulácia si vypýta údaje o mikrogride a vráti spotrebu a výrobu elektrickej energie v čase. Funkcionality simulácie:

- Nastaviť parametre simulácie
- Nastaviť cenu energie v časovom pásme (pracovné dni, víkendy, sviatky, hodinový raster)
- Spustiť simuláciu za obdobie
- Zobraziť priebeh toku energie v danom prvku
- Možnosť nastaviť limitné hodnoty prvku vrátane alarmového systému (maximálna kapacita)
- Vytvoriť návrh
- Najnižšia cena za energiu (akumulácia za účelom odberu v najnižšej cenovej tarife)
- Najnižší vstupný výkon (akumulácia za účelom konštantného vstupu)

Časť simulačného serveru je simulačné jadro, ktoré zabezpečuje fyzikálny výpočet interakcie medzi prvkami v mikrogride.

Architektúra simulačného servera
Simulačný server obsahuje dva moduly:

- Elements
- SimulationKernel

Modul Elements obsahuje implementované správanie elementov vyskytujúcich sa v mikrogride. Druhá časť simulačného serveru je simulačné jadro, ktoré zaobstaráva fyzikálny výpočet simulácie.
Simulačný server je implementovaný v jazyku Python pomocou knižnice Flask. Knižnica Flask je webový framework. Umožňuje vytvoriť webovú aplikáciu.
Simulačný server komunikuje s databázovým serverom pomocou REST rozhrania, rovnako ako aj s webovou appkou. Pre vykonanie simulácie potrebuje server časové rady spotreby elektrickej energie uzlov, ktoré sa nachádzajú v mikrogride.

Výsledok simulácie pozostáva z:

- Výsledná spotreba elektrickej energie
- Náklady v eurách pre elektrickú energiu
- Kapacita baterky
- Sledované elementy

Položka sledované elementy je zložená z nákladov a spotreby pre každý sledovaný element.
Každá položka pozostáva z časového radu, ktorý je vo formáte:

```
{
"ts": "2018-01-05T01:00:00+01:00",
"value": 0.22831666666666697
}
```

Dátový model

Dátová štruktúra pozostáva zo štyroch modelov:

- model projektov (Project)
- model elementov (ElementType)
- model simulácie (Simulation)
- model časových radov (TimeSeries)

Všetky dátové štruktúry sú napísané v jazyku JSON. V rovnakom tvare sa aj ukladajú do databázy.

Model Element (*ElementType*) definuje typy prvkov obsiahnuté v projekte. Z tohto zoznamu sú zároveň vyberané prvky do bočného menu, z ktorého sa dajú pridávať do schémy.

```
"types": {
    "nodes": [
        {
            "id": "SOLAR_PANEL",
            "name": "Solar panel",
            "icon": "./icons/solar-panel.svg"
            "attributes": { "power": "integer" }
        }
    ]
```

Typy definujú okrem unikátneho id a mena ("*name*") aj výzor prvku alebo prepojenia ("*icon"/"color*"). Každý typ prvku obsahuje aj svoje špecifické parametre, ktoré sa nachádzajú v položke "*attributes*".

Údaje o projekte sú uložené vo forme JSON-u. Rozdeleného na 2 časti - metadáta o projekte a samotné údaje reprezentujúce graf mikrogridu.
Formát projektu:

```
{
  "id": "42c93284-adc9-4452-bafe-3cebacb43667",
  "name": "Project 1",
  "createdAt": "timestamp",
  "updatedAt": "timestamp",
  "graph": {
      "groups": [
          {
              "id": 1,
              "name": "Zapadne Slovensko",
              "group": null,
              "separateSchema": false
          }
      ],
      "nodes": [
```

```
            {
                "id": 123,
                "type": "SOLAR_PANEL",
                "name": "Producent 1 - solarny panel",
                "latlong": [48.774947, 16.765878],
                "schemaPos": [-12.05, 0.54],
                "group": [
                    1
                ],
                "attributes": {}
            },
            {
                "id": 321,
                "type": "HOUSE",
                "name": "Konzument 1 - 4-clenna rodina",
                "latlong": [47.774947, 15.765878],
                "schemaPos": [-5.55, 4.21],
                "group": [],
                "attributes": {}
            }
        ],
        "links": [
            {
                "type": "COPPER",
                "from": 123,
                "to": 321
            },
            {
                "type": "COPPER",
                "from": 123,
                "to": 321
            }
        ]
    }
}
```

Metadáta obsahujú ID projektu (unikátne UUID), názov projektu a časové pečiatky. Po nich nasledujú samotné zoznamy prvkov, prepojení a skupín grafu (`"graph"`).
Skupiny (`"groups"`) majú vlastné `id`, na ktoré sa odkazujú ich prvky, meno (`"name"`) a zoznam nadradených skupín, v ktorých je táto skupina združená. `separateSchema` atribút v group je zahrnutý kvôli požiadavke zobraziť separátnu schému, ktorá by sa mala zobrazovať v samostatnom grafe po jej rozkliknutí (napr. schéma bytovky). Skupina, ktorá bude mať tento atribút kladnej hodnoty by mala byť na mapovom zobrazení reprezentovaná iba jedným bodom, ktorý predstavuje všetky jej elementy.
Každý prvok má vlastné `id` (integer), identifikátor prislúchajúci jednému z definovaných typov, meno, pozíciu na mapovom zobrazení (`"latlong"`), pozíciu na schematickom zobrazení (`"schemaPos"`) a id jednej alebo viacerých skupín, do ktorých patrí (`"group"`).
Uzly ďalej obsahujú vnorený objekt `"attributes"`, ktorý môže obsahovať údaje špecifické pre daný typ prvku.
Prepojenia (`"links"`) majú rovnako ako prvky svoj typ (`"type"`), počiatočný a koncový bod (`"from"`, `"to"`) a prípadné dodatočné atribúty (`"attributes"`).

Dátový model simulácie pozostáva z parametrov pre simuláciu, ktorú chceme simulovať na simulačnom serveri.

V tejto štruktúre sa uchovávajú dátumy, pre ktoré obdobie chceme výpočet (dateFrom, dateTo). Tiež sa nachádza odkaz na cenník elektriny (priceID). Posledná položka dátového modelu simulácie je zoznam uzlov, pre ktoré chceme výsledné ceny pre dané obdobie zistiť.

```
"simulation":{
                "priceID":"5bf52ee9fb6fc0561ffdca9c",
                "dataFrom":"2018-08-16T02:00:39+0100",
                "dateTo":"2019-08-16T02:00:39+0100",
                "returnNodesID":[123,254]
        }
```

Webová aplikácia

Webová aplikácia je nástrojom pre modelovanie, simulovanie budúcnosti a vizualizáciu gridu. Aplikácia tiež umožňuje správu projektov, prvkov a tvorbu modelu.

Technológie

Webová aplikácia je implementovaná vo webovom programovacom rámci Angular verzie 6, ktorá je písaná v programovacom jazyku TypeScript, ktorý je exkluzívna nadmnožina programovacieho jazyku JavaScript. Okrem silnej typizácií ponúka dedenie, rozhrania a polymorfizmus podobne ako objektovo orientované jazyky. V čase kompilácie sa však TypeScript transformuje do jazyku JavaScript.
Programovací rámec Angular uľahčuje vývoj webových, mobilných ale aj desktopových aplikácii vďaka deklaratívnym HTML šablónam, reaktivnemu programovaniu a integrácii overených praktík pre vývoj aplikácii. Angular využíva NodeJS. Programovací rámec je vyvýjaný firmou Google od roku 2010 a je OpenSource. Viac informácií o Angulari nájdete tu .

Externé knižnice

V aplikácii sme využili tieto dostupné knižnice:

- **GoJS** - Knižnica pre manipuláciu a vizualizáciu komplexných grafov a diagramov.
- **Leaflet.js** - Knižnica poskytuje komponenty pre zobrazenie voľne dostupných máp. Tiež umožňuje integráciu máp a diagramov knižnice GoJs.
- **Chart.js** - Knižnica pre dynamickú vizualizáciu grafov z dát.

Repozitár

Repozitár obsahuje iba zdrojové súbory a konfiguračné súbory, ktoré sú nevyhnutné pre kompiláciu projektu a boli vygenerované nástrojom konzolovou aplikáciou `@angular/cli`. Všetky externé knižnice a závislosti je pri vývoji potrebné lokálne nainštalovať (viac v sekcii Pokyny pre vývoj).
Zdrojové súbory aplikácie sa nachádzajú v priečinku `src`.
Konfiguračné súbory:

- `angular.json` - Konfiguračný súbor pre Angular.
- `package.json` - Konfigurácia pre NodeJS a zoznam externých knižníc pre vývoj a produkciu.
- `Jenkinsfile` - Konfigurácia pre CI Jenkins a automatizovaný build aplikácie.
- `Dockerfile` - Konfigurácia kontajnerizácie samotnej webovej aplikácie prostredníctvom nástroja docker.
- Priečinok `src` obsahuje tri podpriečinky:
- `app` - Zdrojové súbory samotnej Angular aplikácie.

- `assets` - Statické súbory potrebné pri behu aplikácie (obrázky, fonty, ikony ...)
- `environments` - Obsahuje súbory potrebné pre konfiguráciou premenných pre rôzne prostredia.

V priečinku `src` sa ešte nachádzajú dôležité súbory `index.html` a `styles.scss`. Do týchto súborov Angular vloží skompilovaný kód aplikácie.
Podpriečinok `app` má nasledovnú štruktúru:

- `components` - Obsahuje komponenty, ktoré sú zdieľané naprieč viacerými obrazovkami a inými komponentami.
- `model` - Obsahuje rozhrania a triedy definujúce všetky doménové objekty z reálneho sveta.
- `pages` - Obsahuje iba komponenty zodpovedajúce obrazovkám aplikácie.
- `pipes` - Obsahuje filtre (pipe), ktoré umožňuje transformáciu dát v aplikácii.
- `services` - Obsahuje implementáciu služieb v ktorých sa uchováva logika, ktorá je zdieľaná medzi komponentami v aplikácii.

Automatizovaná integrácia

Keďže sme sa rozhodli pre distribuovanú mikroslužbovú architektúru, tak sa s ňou úzko spája aj automatizovaná integrácia (Continuous Integration).
Z celkového pohľadu na architektúru systému sme identifikovali tieto 3 mikroslužby:

- Grafické rozhranie (Angular)
- Simulačný server
- Databázový server

Repozitáre

Kedže každá mikroslužba bude nezávislá od ostatných služieb tak vzniknu 3 repozitáre, pre každú službu osobitný repozitár.
Každá služba bude osobitne nasadená do produkcie a preto zmeny v jednom repozitáry nemôžu ovplyvniť inú mikroslužbu.

Jenkins

Pre automatizovanie buildu, testov a kompilácie jednotlivých mikroslužieb, nainštalovali sme na našom vývojovom serveri nástroj Jenkins. Jenkins umožňuje automatizovať jednotlivé kroky prostredníctvom pluginu Pipelines. Automatizovaný pipeline vieme konfigurovať v repozitári prostredníctvom `Jenkinsfile` konfiguračného súboru. V tomto konfiguračnom súbore vieme nakonfigurovať Etapy (stage) a jednotlivé kroky (step), ktoré môžu napríklad predstavovať príkaz. Pri spustení tejto automatizovanej integrácie, Jenkins vykoná jednotlivé nakonfigurované kroky. Aby bola aplikácia úspešne skompilovaná musí úspešne prejsť všetkými etapami a krokmi.

Docker

Jedným z posledných krokov automatizovanej integrácie je kontajnerizácia prostredníctvom nástroja docker. Docker umožňuje rýchlejší vývoj, nasadenie a spustenie aplikácií vďaka kontajnerizácií. Kontajnerizácia umožňuje vývojárovi naplniť kontajner všetkými nevyhnutnými závislosťami pre aplikáciu ako sú knižnice, nástroje a ostatné závislosti. Ak aplikácia pobeží v docker kontajneri, tak je zabezpečené, že pobeží aj na akomkoľvek inom stroji, na ktorom je nainštalovaný docker. Aplikácia sa odosiela do produkcie ako jeden kontajner.
Samotný kontajner sa vytvorí z docker obrazu (docker image). Tento obraz je definovaný prostredníctvom konfiguračného súboru `Dockerfile`, ktorý sa nachádza v každom repozitári. Do

obrazu môžeme vložiť konfiguráciu už iného docker obrazu, čo podporuje a uľahčuje nasadenie aplikácií s mikroslužbovou architektúrou.

Databázový server

Server sa nachádza na webovej adrese http://80.241.209.214:5050 .
Na databázových server sa pripája pomocou REST služieb, kde sú definované rozhrania API. K dispozícii sú štandardné metódy GET, POST, PUT a DELETE. Databázový server pristupuje do databázy, upravuje formát dát pred a po volaní do DB. Databáza je na freehostingu na webovej adrese https://mlab.com/databases/gridbox/. Používame NoSQL databázu, do ktorej sa vkladajú "dokumenty", ktoré obsahujú JSON. V databáze sa nachádza 6 tabuliek:

- **projects** - obsahuje všetky vytvorené projekty vo webovej aplikácii
- **time_series** - tabuľka obsahuje časové rady elementov v gride
- **simulations** - uchováva výsledky simulácie na daných projektoch
- **element_types** - všetky elementy, ktoré môžu vystupovať v projekte
- **prices** - cenník elektrickej energie pre každú hodinu v týždni a dňa sviatku
- **icons** - ikony uložené v kódovaní base64 pre každú ikonu

V ďalších šprintoch projektu plánujeme migráciu na vlastný server, ktorý bude mať jediný prístup do databáz, a na ktorý sa bude naša webová aplikácia a simulačný server dopytovať.

Rozhrania API Servera

V tejto časti sú popísané rozhrania na server pomocou, ktorých sa dá dopytovať do databázy.

Kolekcia Project

Operation	HTTP method	path	returns
List projects	GET	/projects	JSON of project ids + metadata
Create project	POST	/projects	id
Load project	GET	/projects/id	project JSON
Save project	PUT	/projects/id	200
Delete project	DELETE	/projects/id	204
Copy project	POST	/projects/id	id

Kolekcia Simulation

Operation	HTTP method	path	returns
List simulations without results	GET	/simulations	JSON of simulations

Operation	HTTP method	path	returns
Create simulation	POST	/simulations	id
Load simulation	GET	/simulations/id	simulation JSON
Save simulation	PUT	/simulations/id	200
Delete simulation	DELETE	/simulations/id	204

Kolekcia Element_types

Operation	HTTP method	path	returns
Load all elements	GET	/elements	JSON of all elements_list
Load single elements	GET	/elements/id	element list JSON

Kolekcia Prices

Operation	HTTP method	path	returns
Load single price list	GET	/prices/id	price list JSON

Kolekcia Icons

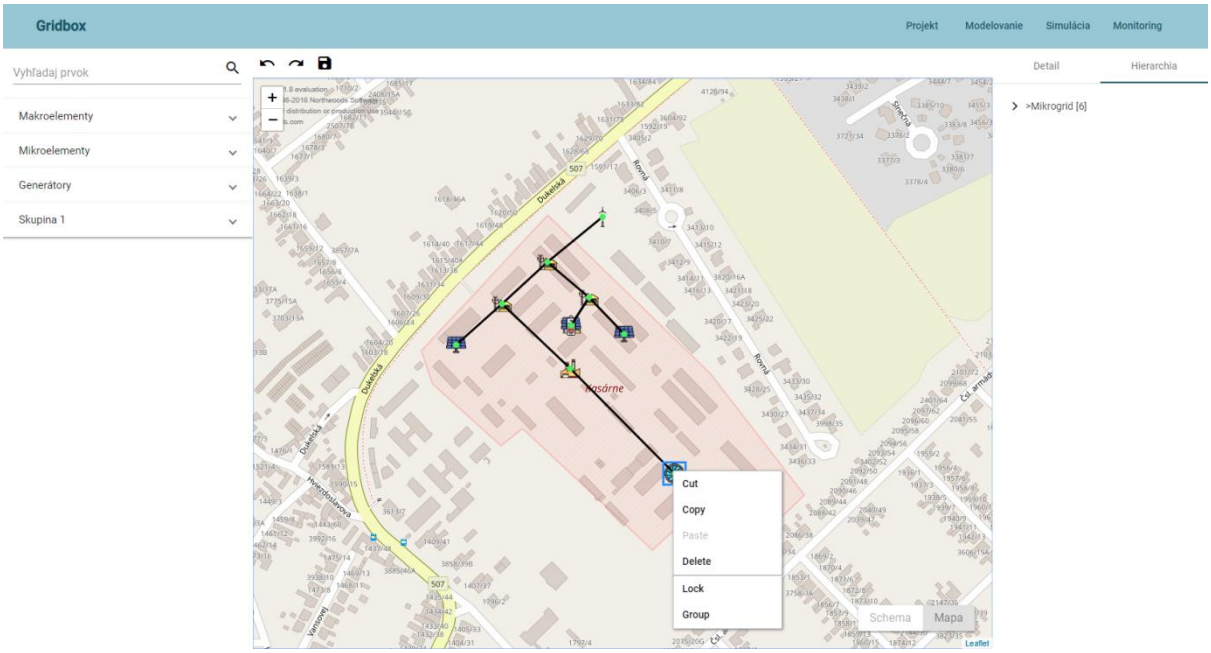
Operation	HTTP method	path	returns
Load icon	GET	/icons/id	SVG data

Prototyp

Prototyp, ktorý vznikol na konci zimného semestra obsahuje nasledovnú funkcionalitu a služby.

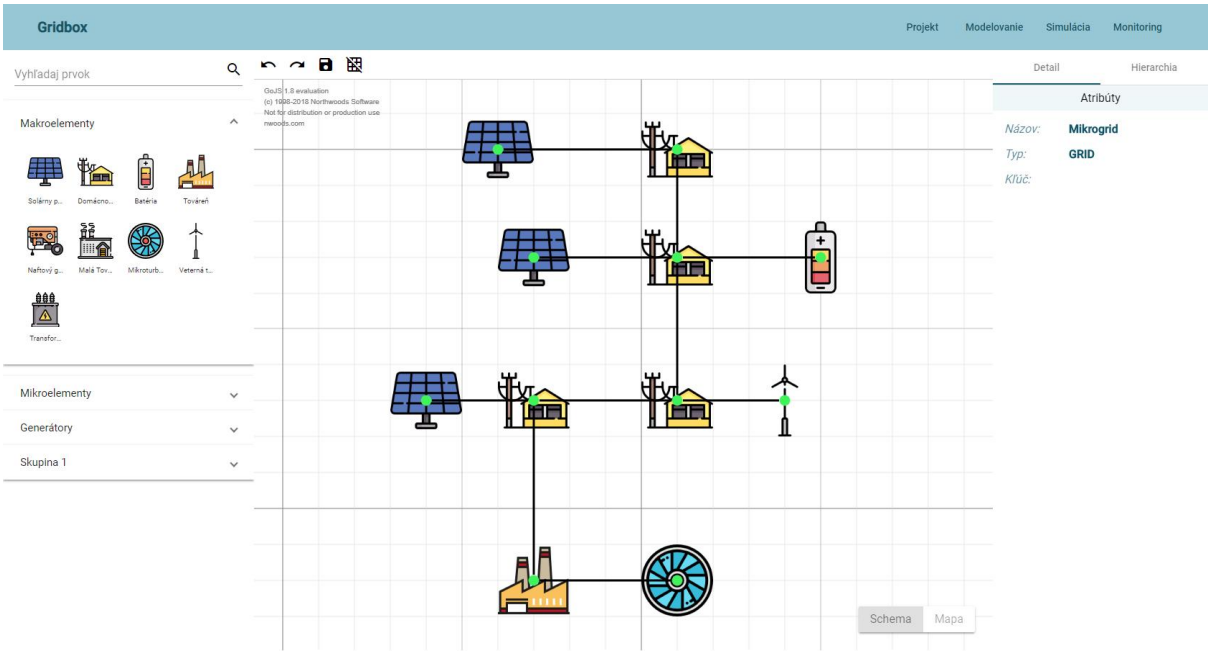
Modelovanie

V rámci modelovania má používateľ v ľavom paneli možnosť vybrať si spomedzi 9 prepripravených makroelementov a tie umiestniť na modelovaciu plochu. Tu môže medzi nimi vytvárať prepojenia, ktoré vychádzajú zo stredu jednotlivých elementov. Pomocou pravého kliknutia na element si môže zvoliť vystrihnúť prvok, skopírovať prvok, prilepiť skopírovaný prvok, vymazať prvok alebo si vybrané prvky označiť ako skupinu.



Obrázok 3 GUI pre modelovanie s mapou

Na pravej strane obrazovky má používateľ možnosť vidieť tri základné atribúty aktuálne zvoleného prvku: jeho názov, typ a id. Zároveň si môže prekliknúť a zobraziť si hierarchiu vytvoreného modelu. Používateľ na má výber medzi mapovým a schématickým zobrazením. V prípade schématického modelovania si môže vypnúť mriežku pod modelovaním, ostatné operácie sú spoločné pre oba režimy (vrátane kroku späť, kroku dopredu a uloženia).



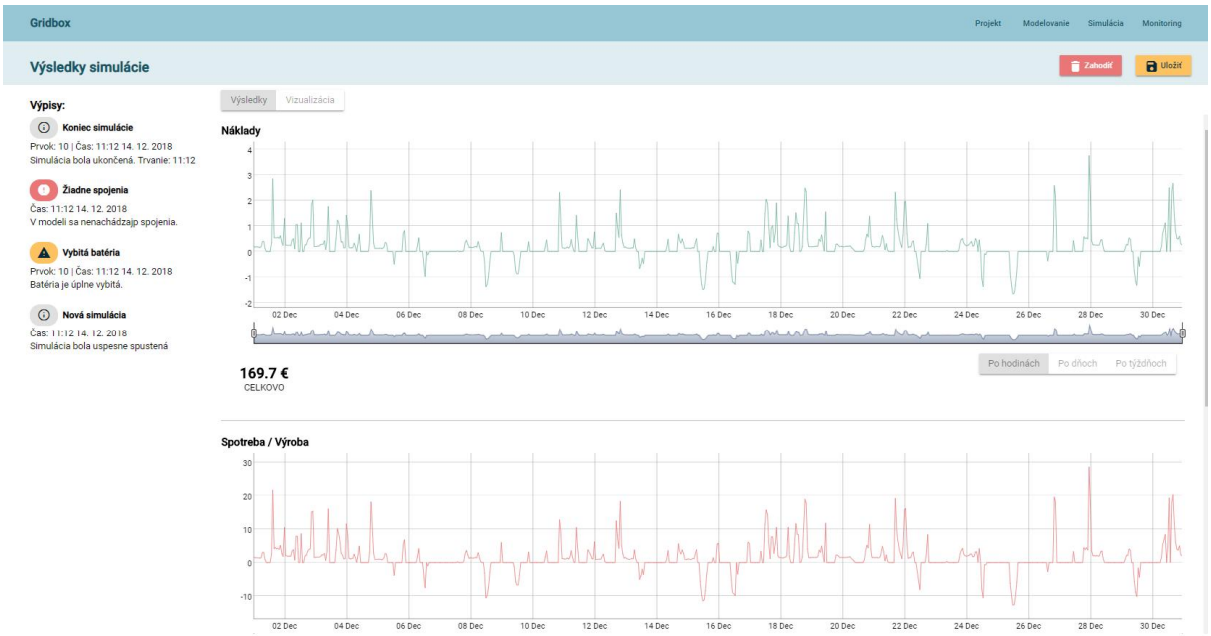
Obrázok 4 GUI pre modelovanie v schéme

Simulácia

V menu simulácie sú zobrazené uložené simulácie. Používateľ si tu môže upraviť zadaný cenník alebo vytvoriť novú simuláciu. V prípade voľby novej simulácie si zadá jej názov, dátum začiatku a konca simulácie a z hierarchie prvkov si vyberie prvky, ktoré chce pri simulácií sledovať.

Obrázok 5 GUI pre parametre simulácie

Vo výslednom okne simulácie sa používateľovi zobrazia v grafoch náklady (ich zobrazovanie je možné upravovať: po hodinách, po dňoch, po týždňoch) vrátane celkovej sumy nákladov v eurách. Pod touto časťou sa nachádza graf pre spotrebu a výrobu (platí to čo je uvedené aj pri nákladoch), vrátane celkového odberu a celkového prebytku. Vľavo sú zobrazené výpisy (tie sú zatiaľ natvrdo zobrazené nezávislé od zvolenej simulácie).



Obrázok 6 GUI pre výsledky simulácie

Príloha A - Export evidencie úloh

Export úloh vygenerovaný zo systému TFS.

Šprint 1

ID	Work Item Type	Title	Assigned To
9082	Product Backlog Item	Analýza možnosti perzistencie pozícií	Bc. Matej Prochazka
9118	Task	Preskúmať správanie pozícií pri načítaní a vymazaní po exporte [go.js]	Bc. Matej Prochazka
9092	Task	Preskúmať správanie pozícií pri načítaní a vymazaní po exporte [mxgraph]	Bc. Matej Prochazka
9119	Task	Preskúmať správanie pozícií pri načítaní a vymazaní po exporte [vis.js]	Bc. Matej Prochazka
9120	Task	Preskúmať správanie pozícií pri načítaní a vymazaní po exporte [d3]	Bc. Matej Prochazka
9079	Product Backlog Item	Analýza dátového modelu	Bc. Peter Pavlik
9087	Task	Overiť Import/Export dát [mxgraph]	Bc. Peter Pavlik
9122	Task	Overiť Import/Export dát [vis.js]	Bc. Peter Pavlik
9124	Task	Overiť Import/Export dát [go.js]	Bc. Peter Pavlik
9126	Task	Overiť Import/Export dát [d3]	Bc. Peter Pavlik
9088	Task	Overiť reprezentáciu dátového modelu grafu/vizualizácie [mxgraph]	Bc. Peter Pavlik
9127	Task	Overiť reprezentáciu dátového modelu grafu/vizualizácie [go.js]	Bc. Peter Pavlik
9128	Task	Overiť reprezentáciu dátového modelu grafu/vizualizácie [vis.js]	Bc. Peter Pavlik
9129	Task	Overiť reprezentáciu dátového modelu grafu/vizualizácie [d3]	Bc. Peter Pavlik
9080	Product Backlog Item	Analýza možnosti simulácie	Bc. David Pavelka
9089	Task	Zistiť prítomnosť simulačného modulu(výpočtového jadra) [mxgraph]	Bc. David Pavelka
9130	Task	Zistiť prítomnosť simulačného modulu(výpočtového jadra) [go.js]	Bc. David Pavelka
9131	Task	Zistiť prítomnosť simulačného modulu(výpočtového jadra) [vis.js]	Bc. David Pavelka
9132	Task	Zistiť prítomnosť simulačného modulu(výpočtového jadra) [d3]	Bc. David Pavelka
9081	Product	Analýza možnosti vykreslovania grafov	Bc. Michal

	Backlog Item		Ostrodicky
9090	Task	Overiť vykreslovanie krviek/grafov [mxgraph]	Bc. Michal Ostrodicky
9133	Task	Overiť vykreslovanie krviek/grafov [go.js]	Bc. Michal Ostrodicky
9134	Task	Overiť vykreslovanie krviek/grafov [vis.js]	Bc. Michal Ostrodicky
9135	Task	Overiť vykreslovanie krviek/grafov [d3]	Bc. Michal Ostrodicky
9091	Task	Mouseover možnosť vykreslenia vlastného obsahu (graf, tabulka, obrázok...) [mxgraph]	Bc. Michal Ostrodicky
9136	Task	Mouseover možnosť vykreslenia vlastného obsahu (graf, tabulka, obrázok...) [go.js]	Bc. Michal Ostrodicky
9137	Task	Mouseover možnosť vykreslenia vlastného obsahu (graf, tabulka, obrázok...) [vis.js]	Bc. Michal Ostrodicky
9139	Task	Mouseover možnosť vykreslenia vlastného obsahu (graf, tabulka, obrázok...) [d3]	Bc. Michal Ostrodicky
9083	Product Backlog Item	Analýza mapového zobrazenia (transformácie zo schématického)	Bc. Martin Cincurak
9094	Task	Overiť možnosť dvoch zobrazení - preklik medzi nimi - [mxgraph]	Bc. Martin Cincurak
9140	Task	Overiť možnosť dvoch zobrazení - preklik medzi nimi - [go.js]	Bc. Martin Cincurak
9141	Task	Overiť možnosť dvoch zobrazení - preklik medzi nimi - [vis.js]	Bc. Martin Cincurak
9142	Task	Overiť možnosť dvoch zobrazení - preklik medzi nimi - [d3]	Bc. Martin Cincurak
9084	Product Backlog Item	Analýza možnosti agregácie	Bc. Richard Mocak
9096	Task	Overiť vytvorenie agregovaného prvku [mxgraph]	Bc. Richard Mocak
9146	Task	Overiť vytvorenie agregovaného prvku [go.js]	Bc. Richard Mocak
9147	Task	Overiť vytvorenie agregovaného prvku [vis.js]	Bc. Richard Mocak
9098	Task	Overiť zoom optimalizáciu [mxgraph]	Bc. Richard Mocak
9149	Task	Overiť zoom optimalizáciu [go.js]	Bc. Richard Mocak
9099	Task	Overiť možnosť dopytovania (GraphQL) [mxgraph]	Bc. Richard Mocak
9155	Task	Overiť možnosť dopytovania (GraphQL) [go.js]	Bc. Richard Mocak
9085	Product Backlog Item	Analýza architektúry	Bc. Frantisek Durana
9103	Task	Preskúmať architektúru knižnice a jej modulov [mxgraph]	Bc. Frantisek Durana
9143	Task	Preskúmať architektúru knižnice a jej modulov [go.js]	Bc. Frantisek Durana
9144	Task	Preskúmať architektúru knižnice a jej modulov [vis.js]	Bc. Frantisek Durana

9145	Task	Preskúmať architektúru knižnice a jej modulov [d3]	Bc. Frantisek Durana

Šprint 2

9517	Product Backlog Item	Vytvoriť GUI pre prototyp	Bc. Richard Mocak
9518	Task	Vytvoriť základné GUI	Bc. Richard Mocak
9519	Task	Podpora pre dve základné zobrazenia (mapové a schéma)	Bc. David Pavelka
9520	Task	Aspoň základné drag-and-drop menu s prvkami	Bc. Matej Prochazka
9521	Task	GUI by malo obsahovať komponenty pre zobrazenie grafu a pre zobrazenie vlastností prvku	Bc. Frantisek Durana
9522	Product Backlog Item	Vytvorenie architektonického návrhu	Bc. Martin Cincurak
9523	Task	Prvotný architektonický návrh	Bc. Martin Cincurak
9514	Product Backlog Item	Vytvorenie dátového modelu projektu	Bc. Peter Pavlik
9516	Task	Definovať štruktúru projektu	Bc. Peter Pavlik
9524	Task	Analýza možností optimalizovaného verziovania	Bc. Michal Ostrodicky
9525	Product Backlog Item	Definovanie metodológií	Bc. Michal Ostrodicky
9526	Task	Definovať metodológiu pre používanie Git-u	Bc. David Pavelka
9527	Task	Definovať metodológie pre písanie kódu	Bc. Richard Mocak

Šprint 3

9927	Product Backlog Item	Rozšírenie GUI	Bc. David Pavelka
9928	Task	GUI pre simuláciu	Bc. Richard Mocak
9931	Product Backlog	Testovanie	Bc. Richard

	Item		Mocak
10042	Task	Jenkins pre CI	Bc. Richard Mocak
10043	Task	Build jenkins pipeline pre Angular applikaciu	Bc. Richard Mocak
9922	Product Backlog Item	Implementácia simulačnej vrstvy	Bc. Martin Cincurak
9923	Task	Vytvorenie modelu batérie	Bc. Matej Prochazka
9925	Task	Vytvoriť server	Bc. Martin Cincurak
9926	Task	Analýza a návrh architektúry simulačného jadra	Bc. Frantisek Durana
9930	Task	Vytvorenie API pre simulačný server	Bc. Martin Cincurak
9918	Product Backlog Item	Implementácia dátovej vrstvy	Bc. Peter Pavlik
9919	Task	Vytvorenie databázy	Bc. Michal Ostrodicky
9920	Task	Aktualizovanie dátového modelu	Bc. Peter Pavlik
9921	Task	API pre dopyty do DB	Bc. Michal Ostrodicky

Šprint 4

9927	Product Backlog Item	Rozšírenie GUI	Bc. David Pavelka
9929	Task	Zapracovanie pripomienok z review	Bc. David Pavelka
10260	Task	GOJS paleta prvkov na HTML paletu	Bc. David Pavelka
10261	Task	Pridanie hornej lišty pri modelovaní	Bc. David Pavelka
10263	Task	Pridanie kontextového menu	Bc. David Pavelka
10332	Task	Uchovavanie stavu pri prepinani rezimov	Bc. Richard Mocak
10453	Task	Agregácia výsledkov zo simulácie	Bc. Peter Pavlik
10056	Product Backlog Item	Integrácia simulačného servera	
10230	Task	Nasadenie simulačného servera	Bc. Martin Cincurak
10231	Task	Prijatie výsledku simulácie web aplikáciou	Bc. Martin Cincurak
10232	Task	Odoslanie požiadavky na simuláciu z webapp	Bc. Richard Mocak
10233	Task	Nastaviť limit generovaných dát na DB	Bc. Michal Ostrodicky
10234	Task	Uloženie simulácie v DB [BE]	Bc. Michal Ostrodicky
10235	Task	Zobrazenie výsledkov simulácie vo	Bc. Richard

		webapp	Mocak
10050	Product Backlog Item	Implementácia jednoduchej simulácie prvkov	Bc. Martin Cincurak
10236	Task	Pridanie parametrov do jednoduchej simulácie	Bc. Martin Cincurak
10062	Product Backlog Item	Integrácia cenníka do simulácie	
10237	Task	Vytvorenie formuláru na zadanie cenníka	Bc. Richard Mocak
10045	Product Backlog Item	Integrácia webovej aplikácie a DB	Bc. Frantisek Durana
10238	Task	Vytvoriť nový projekt	Bc. Michal Ostrodicky
10239	Task	Zmazanie projektu z DB	Bc. Frantisek Durana
10241	Task	Uložiť projekt do DB	Bc. Michal Ostrodicky
10242	Task	Načítať projekt z DB	Bc. Peter Pavlik
10246	Task	Načítanie typov elementov z DB	Bc. Peter Pavlik
10360	Task	Pridanie DB endpointu pre SVG	Bc. Frantisek Durana
10361	Task	Aktualizácia enpointu pre typy elementov	Bc. Peter Pavlik

Šprint 5

9927	Product Backlog Item	Rozšírenie GUI	Bc. David Pavelka
10262	Task	Zapracovať jednotné ovládanie pomocou klávesových skratiek	Bc. David Pavelka
9931	Product Backlog Item	Testovanie	Bc. Richard Mocak
10464	Task	Integračné testy pre rozhrania DB servera	Bc. Richard Mocak
10465	Task	Template pre integračné testy rozhraní	Bc. Peter Pavlik
10466	Task	Integračné testy pre rozhrania Simulačného serveru	Bc. Martin Cincurak
10056	Product Backlog Item	Integrácia simulačného servera	
10247	Task	Načítanie simulácie z DB	Bc. Richard Mocak
10469	Task	Zobrazenie uložených simulácii vo webapp	Bc. Richard Mocak
10470	Task	Načítanie uložených simulácií z DB	Bc. Michal Ostrodicky
10045	Product Backlog Item	Integrácia webovej aplikácie a DB	Bc. Frantisek Durana

10240	Task	Skopírovať projekt ako nový	Bc. Peter Pavlik
10471	Product Backlog Item	Editácia atribútov	Bc. David Pavelka
10244	Task	Možnosť zmeny atribútov prvkov so základnou validáciou	Bc. David Pavelka
10472	Task	Zmena názvu projektu	Bc. David Pavelka