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 a letného semestra. Dokument je rozdelený na dve veľké časti a to riadenie projektu a inžinierske dielo podľa pokynov na stránke predmetu[1].

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

Druhá časť ponúka pohľad na globálne ciele projektu pre obdobie zimného i letného semestra, celkový pohľad na systém a jeho architektúru i podrobnejšie rozobrané moduly, ich význam a zacelenie do jedného systému. Poslednou časťou tohto dokumentu sú prílohy, obsahujúce používateľskú príručku, inštalačnú príručku a 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, Technická dokumentácia

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

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

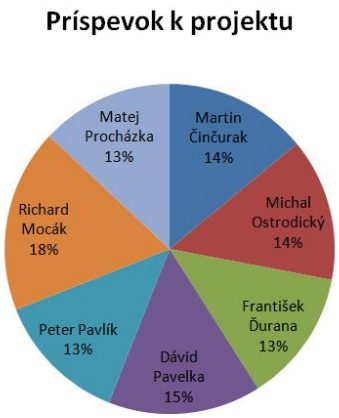
Peter Pavlík - Dátový majster
Manažérske úlohy: Manažment testovania

[1] http://www2.fiit.stuba.sk/~bielik/courses/tp-slov/tp-main.html#dokumentacia

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, Inštalačná príručka

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



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 štandardne časové obdobie dvoch týždňov. Vzhľadom na dobu vypracovania dokumentu sú tu sumarizované všetky šprinty, ktoré sme absolvovali počas zimného aj počas letného semestra.

Zimný semester

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.

Letný semester

Sprint 6 - Warlords of Draenor

Začiatok šprintu: 14. februára 2019
Koniec šprintu: 28. februára 2019

Prvý šprint v letnom semestri bol zameraný najmä na analýzu možností pre ďalšie smerovanie projektu počas celého letného semestra. Kľúčovou zložkou analýzy bolo pridávanie a úprava nových simulačných jadier. Okrem toho sme analyzovali možnosti logovania všetkých typov výpisov počas simulácie, zobrazovanie výsledkov simulácie a jej postupné prehrávanie.
Okrem toho sa zapracovali zmeny na základe pripomienok produktového vlastníka. Tie zahŕňali zmenu rozloženia modelovacej schémy s cieľom maximalizovať plochu modelovania (minimalizovanie ľavého panelu s prvkami a pravého panelu s atribútmi). Vzhľadom na vytváranie aplikácie v slovenčine bol upravený formát dátumu a času na slovenský štandard. Zmeny sa týkali aj vizuálnej stránky webovej aplikácie, vytvorili sme logo, landing page a optimalizovali menu v hornej časti.

Sprint 7 - Legion

Začiatok šprintu: 28. februára 2019
Koniec šprintu: 14. marca 2019

Na základe predchádzajúcej analýzy bol počas tohto šprintu vytvorený prototyp nového simulačného jadra. To sa od prvého jadra, ktoré sme používali doteraz líši najmä komplexnejšou logikou výpočtu simulácie nad daným modelom. Zároveň nám umožňuje pridať používateľovi možnosť voľby medzi simulačnými jadrami. Skutočnosť, že náš projekt má byť inicializačným bodom ďalších projektov nám predurčila potreby kvalitného dokumentovania používaných API rozhraní, na čo sme sa rozhodli využiť Swagger.
Vzhľadom na viac používaných typov grafov sme generalizovali výsledky simulácie tak, aby sa informácie o použitom grafe nachádzali priamo vo výstupe simulácie. Front-endová časť aplikácie bola upravená s cieľom zvýšenia používateľského zážitku a prehľadnosti jednotlivých funkcií ponúkaných aplikáciou.

Sprint 8 - Battle For Azeroth

Začiatok šprintu: 14. marca 2019
Koniec šprintu: 28. marca 2019

Počas tohto šprintu nás čakalo testovanie aplikácie potenciálnym používateľom, tomu boli prispôsobené aj tímové úlohy. Pre simulácie boli definované obmedzenia, ktoré musia byť splnené, aby bolo možné

využívať nové simulačné jadro. Zatiaľ nepodporovaná časť aplikácie "Monitoring" bola sprístupnená s informáciou, že sa na stránke pracuje. Na základe pripomienok sme opravili výber všetkých elementov pri kliknutí a skryli pravý panel aplikácie tak, aby sa zobrazoval len v prípade potreby. Pri vkladaní prvku sme upravili jeho pozíciu tak, aby bola presne na súradniciach zvolených používateľom aplikácie.

Sprint 9 - Cataclysm 2

Začiatok šprintu: 28. marca 2019
Koniec šprintu: 11. apríla 2019

Šprint 9 prebiehal v období pred TP-Cupom, čomu sme prispôsobili aj tímové úlohy. Hlavnú časť predstavovalo dokončenie možnosti správy knižníc a prvkov na oboch úrovniach aplikácie. Na front-endovej strane aplikácie to zahŕňalo pridanie novej podstránky do konfiguračnej stránky, vytvorenie komponentu pre nahrávanie svg ikony prvku z lokálneho úložiska a úpravu niektorých častí vizuálnej stránky aplikácie. Back-endová časť aplikácie sa rozrástla o nové rozhrania pokrývajúce funkcionalitu pridávania prvkov, knižníc a ich modifikácie. Okrem pridávania a správy knižníc sme pridali zobrazovanie postupného prehrávania simulácie v čase a zobrazovanie logov. Formálne náležitosti TP-Cupu vyžadovali vytvorenie prezentačného videa našej aplikácie, napísanie článku a vyplnenie dotazníka.

Gridbox Team Sprint 10 - Reign of Chaos

Začiatok šprintu: 11. apríla 2019
Koniec šprintu: 25. apríla 2019

Počas šprintu prebiehal TP-CUP, na ktorý sme pripravovali našu aplikáciu. Dokončili sme správu prvkov, pridali validáciu vstupu pre jej jednotlivé položky a pridávanie atribútov pre daný prvok. Táto časť sa rozrástla nie len o pridávanie nových prvkov ale aj úpravu už existujúcich prvkov. Tieto zmeny si však vyžadovali úpravu používaného dátového modelu.
Pre prezentačné účely sme vyčistili a pripravili produkčné prostredie so zálohou, prichystali sme si ďalšie dve knižnice prvkov pre modelovanie počítačových a cestných sietí, upravili stránky zatiaľ nepodporovaného monitoringu a upravili kontextové menu. Okrem toho sme sfinalizovali zvyšné požadované funkcie aplikácie a nakoniec aj úspešne odprezentovali náš projekt na IIT.SRC.

Sprint 11 - The Frozen Throne

Začiatok šprintu: 25. apríla 2019
Koniec šprintu: 10. mája 2019

Posledný šprint sme venovali finalizovaniu dokumentácie vytvoreného riešenia, doplnili a aktualizovali sme všetky jej časti tak, aby zodpovedali požiadavkám na stránke predmetu. Okrem toho sme na základe cennej spätnej väzby z TP-Cupu vykonali ďalšie zmeny aplikácie a vylepšili načítavanie ikony z lokálneho úložiska i opravili problém s vytvorením/uložením prvku.

Globálna retrospektíva za zimný semester

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

Zhrnutie retrospektív za zimný semester

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.

Globálna retrospektíva za letný semester

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 6 (14.2.2019 - 28.2.2019)

S čím prestať?

- Odkladanie časových odhadov zodpovedných osôb za úlohy

S čím pokračovať?

- Kvalitné a poctivé code review, ktoré pomáhajú odhaliť problémy a bugy, ktoré vznikli
- Komunikácia členov tímu medzi sebou o problémoch, ktoré vznikli priamo na stretnutiach

Čo zlepšiť?

- Lepšie rozdelenie členov tímu vzhľadom na to, že je front-end teraz najdôležitejšou časťou nášho systému
- Poriadnejšie zadeľovanie story tak, aby bola vždy jedna zodpovedná osoba pridelená na story
- Dohodnutie predbežného termínu ukončenia úlohy, aby sa to nemuselo dokončovať vo štvrtok v noci
- Definovanie odhadov na úlohy čo najskôr
- Rovnomernejšie rozdelenie práce medzi jednotlivých členov tímu aj vzhľadom na to, že skúsenosti a schopnosti členov tímu sú veľmi rozdielne
- Návrh a analýza s mock-up nechať a spraviť ich na prezentáciu produktovému vlastníkovi
- Vytvoriť úlohy aj pre také činnosti, ktoré nie sú priamo rozpísané vo vytvorených úlohách

Šprint 7 (28.2.2019 - 14.3.2019)

S čím pokračovať?

- Spoločná analýza problémov na stretnutiach a ich riešenie
- Pomáhanie členom tímu, ktorý nemajú s danou technológiou skúsenosti od skúsenejších kolegov namiesto zdĺhavého vyhľadávania riešenia na internete
- Spoločné definovanie rozhraní pomáha priamo vyriešiť nezrovnalosti
- Poctivé code review a riešenie konfliktov

S čím prestať?

- Dookola riešiť problémy, ktoré sami bez produktového vlastníka (alebo nejakého odborníka v oblasti) aj tak nevieme vyriešiť
- Výstupy z úloh dokončovať až tesne pred koncom šprintu
- Prílišné sústredenie sa na nepodstatné detaily projektu namiesto sústredenia sa na víziu

Čo zlepšiť?

- Detailnejšie a podrobnejšie výstupy z analytických úloh tak, aby bolo možné ich vhodne využiť do implementácie
- Riešenie pull requestov v skoršom termíne, nastavenie termínov na ich dokončenie
- Komunikácia členov tímu s pull requestami a notifikovanie čakajúcich členov tímu ak sa už dlho čaká
- Sústrediť sa na hlavnú víziu projektu a zjednotiť sa na cieľoch
- Experimentálne prostredie na serveri

Šprint 8 (14.3.2019 - 28.3.2019)

S čím prestať?

- Robiť na úlohách ovplyvňujúcich zvyšok, bez toho aby o tom vedeli ostatní

S čím pokračovať?

- Pomáhanie členom tímu, ktorý nemajú s danou technológiou skúsenosti

Čo zlepšiť?

- Častejšie spoločné hovory, ktoré budú štruktúrovanejšie a aj s výstupom pre tých, ktorí sa napríklad hovoru nezúčastnili
- Zadávať si úlohy hneď na začiatku šprintu, aby bolo už počas víkendu možné na nich začať pracovať
- Väčšia dôslednosť v práci tímu, dohodnutie sa na tom čo kto spraví, aby sa nestávalo že tesne pred koncom zostane nejaká úloha nepriradená
- Sústrediť sa na to podstatné, čo je cieľom nášho riešenia a nechať nedôležité aspekty bokom

Šprint 9 (28.3.2019 - 11.4.2019)

S čím prestať?

- Meškanie na tímové stretnutia

S čím pokračovať?

- Sústredenie sa na dôležité body projektu
- Rozdelenie členov do tímov podľa sekcií na ktorých pracujú
- Spoločná práca tímu na front-endovej časti aplikácie podľa dlhodobých cieľov

Čo zlepšiť?

- Pred vytvorením pull requestu na Front-endovej časti aplikácie najskôr spustiť build a presvedčiť sa či je možné aplikáciu vybuildovať
- Prezentovanie vytvorených výsledkov

Šprint 10 (11.4.2019 - 25.4.2019)

S čím prestať?

- Meškanie na tímové stretnutia

S čím pokračovať?

- Spoločná práca tímu na front-endovej časti aplikácie podľa dlhodobých cieľov
- Dokončenie úloh v časovom predstihu, aby sa stihli spracovať pull requesty
- Call v prípade potreby pre zrýchlenie komunikácie medzi všetkými členmi tímu

Čo zlepšiť?

- Zjednotenie prípadov použitia aplikácie
- Prezentovanie vytvorených výsledkov

Šprint 11 (25.4.2019 - 10.5.2019)

S čím prestať?

- Odkladanie prideľovania úloh šprintu na jednotlivých členov tímu

S čím pokračovať?

- Nasadenie v time a spolupráca
- Rozdelenie členov tímu na časti aplikácie podľa ich skúseností
- Vzájomná komunikácia tímu aj mimo tímových stretnutí
- Pomoc skúsenejších členov tímu pre tých, ktorí s danou technológiou nemajú také skúsenosti

Čo zlepšiť?

- Prezentovanie vytvorených výsledkov

Zhrnutie retrospektív za letný semester

Odrazovým mostíkom letného semestra boli pre nás skúsenosti nadobudnuté počas predchádzajúceho obdobia zimného semestra. Tie nám pomáhali v časovom rozplánovaní stretnutí tak, aby sme stihli všetko potrebné bez nutnosti ich predlžovania, pokračovali sme vo využívaní definovaných komunikačných nástrojov a vytvorených metodík. Aj v tomto semestri sa však postupne objavovali problémy a prekážky, ktoré sme museli riešiť. Na začiatku to bol hlavne problém s nedostatočnou dôslednosťou pri metodike úloh, ktorá nám spôsobovala problémy. Zaviedli sme si preto nové pravidlá.

Ďalšou problematickou časťou bola naša snaha venovať sa všetkým aspektom aplikácie bez presne definovaného cieľa, čo chceme počas tohto semestra vytvoriť, čo nie je dôležité a naopak čo má najvyššiu prioritu. Po konzultovaní s produktovým vlastníkom sme si vytvorili víziu projektu, ktorú sme využívali pri ďalšom určovaní dôležitosti vzniknutých úloh.

Veľkým pomocníkom bol pre nás postupne zavedený systém vzájomnej pomoci, kedy skúsenejší členovia tímu pomáhali menej skúseným, vďaka čomu dokázali rýchlejšie pochopiť danú technológiu a splniť svoje úlohy. Toto platilo najmä v prípade front-endovej časti aplikácie, ktorá bola počas letného semestra na prvom mieste. Po druhej polovici semestra sme s naším projektom absolvovali študentskú vedeckú konferenciu IIT.SRC, kde sme získali ďalšie cenné poznatky a skúsenosti.

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 reviziu 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 definícii 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" počas oboch semestrov. 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 funkcionalita 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 po zimnom semestri

Počas obdobia zimného semestra bol vytvorený prototyp aplikácie, ktorý spĺňal všetky ciele, ktoré boli naň na zvolené obdobie kladené minimálne v takom rozsahu, ako je uvedené vyššie v časti Globálne ciele projektu na zimný semester.

Globálne ciele projektu na letný semester

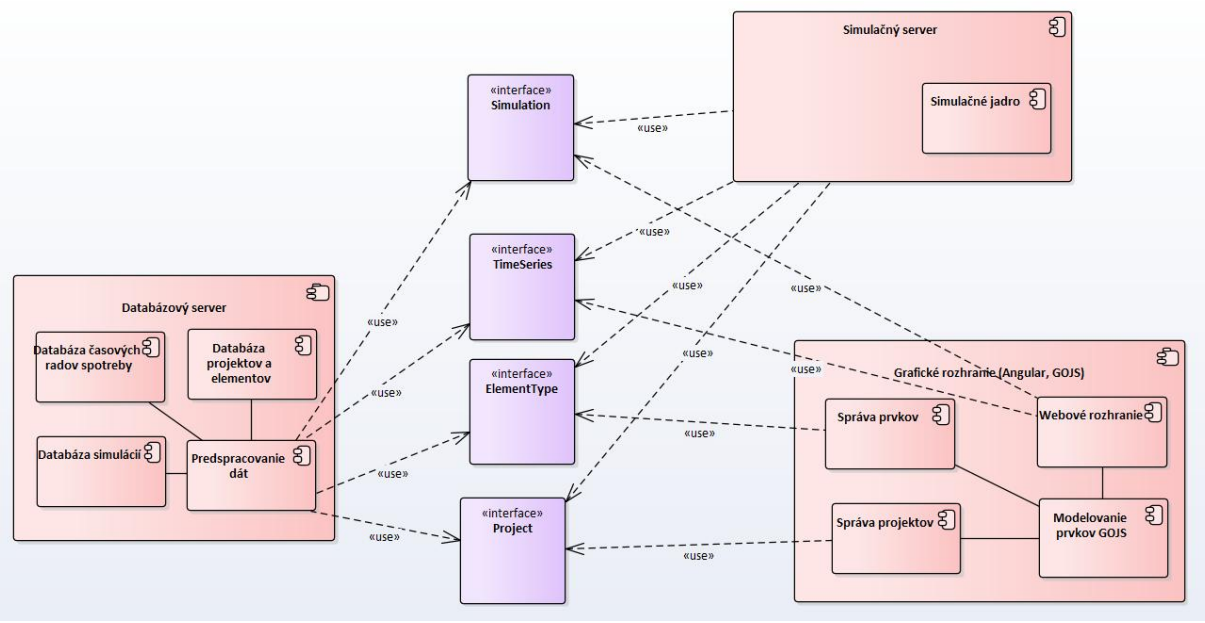
Hlavným smerovaním nášho tímu počas obdobia letného semestra je pokračovať vo vyvíjaní prototypu aplikácie vytvorenej počas zimného semestra. Cieľom je vytvorenie komplexného modelovacieho systému, ktorý bude umožňovať nielen prácu v doméne energetiky, ale s použitím ľubovoľných prvkov a vytvorením vlastnej knižnice modelovať ľubovoľnú doménovú oblasť. Okrem prvotného simulačného jadra vytvoreného počas zimného semestra plánujeme pridať nové, s komplexnejšou logikou výpočtu simulácie. Používateľ si však stále bude môcť vybrať ktorékoľvek z týchto jadier. Vzhľadom na to, že najväčší dôraz je kladený na front-endovú časť aplikácie je naším cieľom v tomto semestri ju vytvoriť používateľsky čo najpríjemnejšie, rešpektujúc štandardy v tejto oblasti a celú ju zabaliť do prezentačného kabátiku, ktorý používateľovi pri vstupe do aplikácie predostrie kľúčové body obsahu a fungovania aplikácie.

Celkový pohľad na systém:

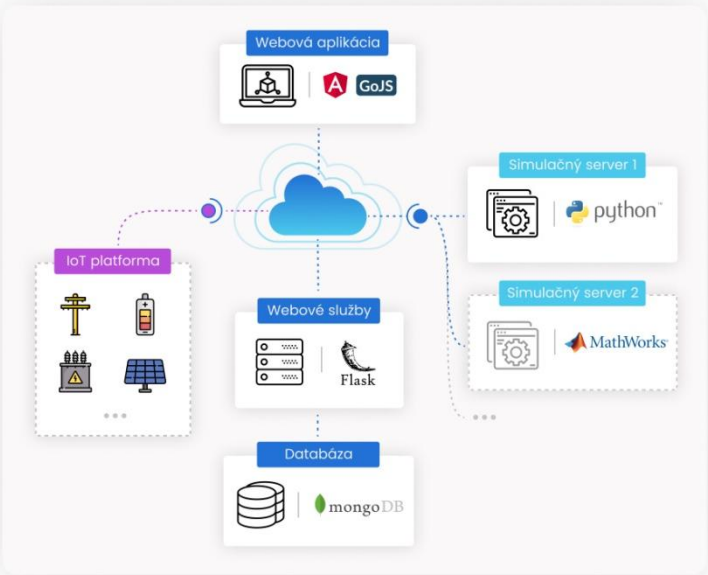
Systém je zložený z viacerých komponentov, ich význam a fungovanie sú podrobnejšie opísané v častiach nižšie.

Architektúra systému

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

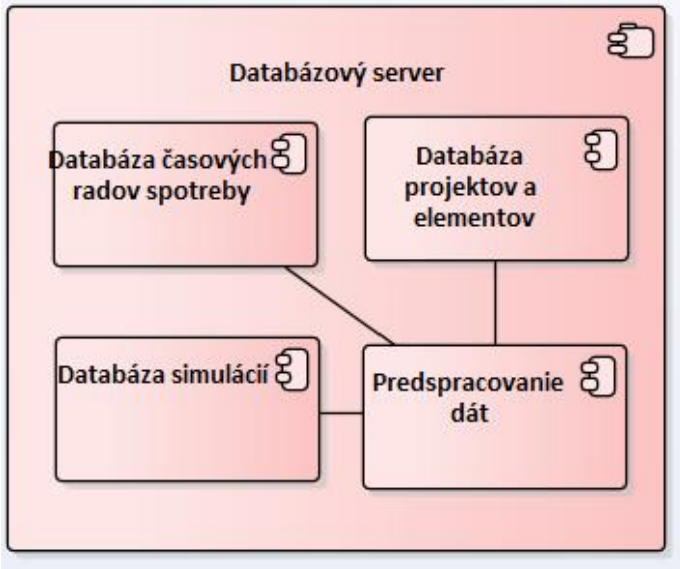


Obrázok 1 Architektúra systému



Obrázok 2 Platforma

Komponent: Databázový server:

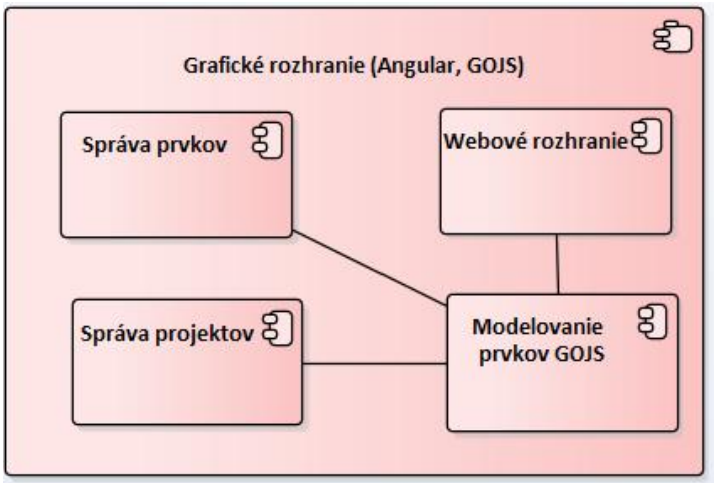


Databáza sa delí na štyri časti:

- 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 projektu simulácii 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.

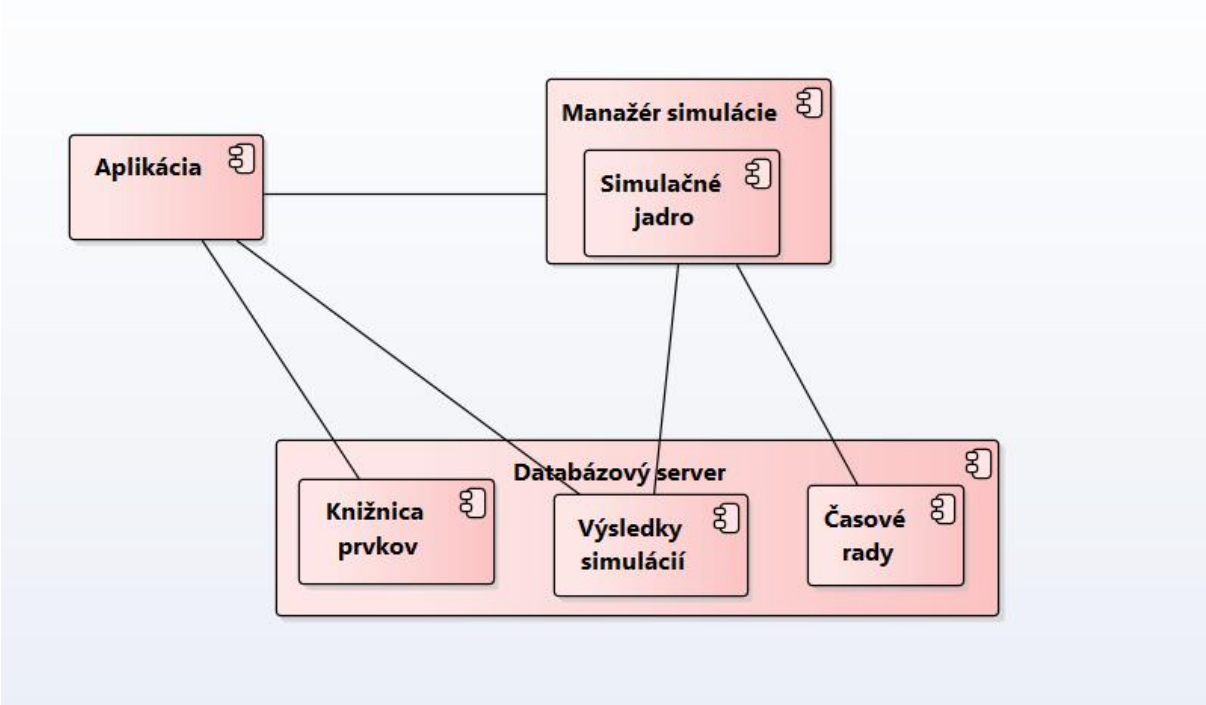
Webové rozhranie (Angular, GOJS)
Webové rozhranie interaguje s používateľom, zobrazuje modely a dovoľuje úpravy.
Modelovacia časť
Modelovacia časť spravuje prvky a prepojenia v grafe. Používa dátový objekt Project, v ktorom vykonáva zmeny a posiela upravený objekt webovému rozhraniu na zobrazenie.
Úpravy 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 3 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 vyžiada 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
- 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.

Algoritmus výpočtu v používanom simulačnom jadre:

1. Na začiatku sa skontroluje, či je v grafe práve jeden hlavný uzol a či graf obsahuje cykly
2. Prehľadávaním do hĺbky sa prechádzajú všetky uzly (začíname od začiatočného uzla, ktorým je hlavný transformátor)
3. Keď sa algoritmus dostane na uzol, ktorý je jeho označený vedúci, vytvorí sa nová podsieť a prehľadávanie pokračuje ďalej
4. Keď sa algoritmus dostane na posledný uzol (nemá ďalšie prepojenia), vypočíta sa spotreba pre daný uzol
5. Keď sa prehľadajú všetky prvky v podsieti, vypočíta sa výsledná krivka
6. Ak sieť obsahuje batériu, výstup siete sa transformuje funkciou batérie
7. Takto pokračuje prehľadávanie pre každú podsieť smerom naspäť k hlavnému uzlu (spotreba sa počíta najprv pre podsieť, ktorá je na najnižšej úrovni)

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": "value",
}

{
"2019-05-06T23:00:00+01:00":0.32945889166666625,
}
```

Rozhranie na vykonanie simulácie pozostáva z dvoch položiek:

- project
- simulation

Položka *Project* je celkový model mikrogridu vytvorený vo webovej aplikácii. Obsahuje uzly a prepojenia medzi nimi. *Simulation* obsahuje údaje o simulácii, ktorá sa má vykonať. V nej sú zaznamenané dátumy od kedy a do kedy simulácia predbieha, ktoré uzly majú byť sledované a cenník cien.

Vo výsledku simulácie sa nachádzajú údaje pre dva typy grafov, ktoré sa zobrazia vo webovej aplikácií:

- *Line Graph* - Spojitý graf
- *Pie Grapf* - Koláčový graf

Ďalej simulácia vracia výsledky pre sledované elementy a prepojenia. Taktiež sa vo výsledku nachádzajú aj logy ktoré boli vygenerované simulačným jadrom počas výpočtu výsledkov.

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.

Knižnica elementov (Element List)

Z tohto zoznamu (knižnice prvkov) sú vyberané prvky (elementy) do bočného menu, z ktorého sa dajú pridávať do schémy. Jeden projekt môže obsahovať prvky z viacerých knižníc za predpokladu, že sa "type" žiadneho prvku v nich nezhoduje.
Typy definujú okrem unikátneho identifikátora ("*type*") a mena ("*name*") aj výzor prvku alebo prepojenia ("*iconId*") a sekciu v menu ("*menuSection*"). Každý typ prvku obsahuje aj svoje špecifické parametre, ktoré sa nachádzajú v položke "*attributes*".

```
{
  "name": "Gridbox knižnica 2.0",
  "description": "Nasa produkcna kniznica.",
  "nodes": [
    {
      "elementType": "nodes",
      "type": "HOUSE",
      "name": "Domácnosť",
      "iconId": "2227777846c2080014ea205c",
      "menuSection": "MACROELEMENTS",
      "attributes": {
        "timeSeriesId": {
          "name": "Časový rad",
          "type": "text",
          "defaultValue": "HOUSE LARGE"
        }
      }
    },
    {
      "elementType": "nodes",
      "type": "BATTERY",
      "name": "Batéria",
      "iconId": "9937777846c2080014ea205c",
      "menuSection": "MACROELEMENTS",
      "attributes": {
        "capacity": {
          "name": "Kapacita",
          "unit": "kWh",
          "type": "number",
          "defaultValue": 100
        },
        "depthOfDischarge": {
          "name": "Pomer vybitia",
          "unit": "kWh",
          "type": "number",
          "defaultValue": 0.9
        },
        "maxCharge": {
          "name": "Maximálne nabitie",
          "unit": "kWh",
          "type": "number",
          "defaultValue": 10
        },
        "maxDischarge": {
          "name": "Maximálne vybitie",
          "unit": "kWh",
          "type": "number",
          "defaultValue": 10
        }
      }
    }
  ],
  "links": [
    {
      "elementType": "links",
      "type": "COPPER",
      "name": "Medené vedenie",
      "color": "#f9a339",
      "iconId": "3337777846c2080014ea205c",
      "menuSection": "CONNECTORS",
      "attributes": {
        "maxCurrent": {
          "name": "Maximálny prúd",
          "unit": "A",
          "type": "number",
          "defaultValue": 10
        }
      }
    }
  ],
  "id": "5c07db8bfb6fc038cbb3aa8f"
}
```

Projekt

Metadáta obsahujú ID projektu (unikátne UUID), názov projektu a časové pečiatky. Po nich nasledujú samotné zoznamy prvkov, prepojení a skupín grafu (`"graph"`).
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"`).

```
{
  "name": "Projekt domček",
  "graph": {
    "groups": [],
    "nodes": [
      {
        "id": -1,
        "type": "MAINTRANSFORMER",
        "name": "Hlavný Transformátor 1",
        "latlong": [
          48.14699721473494,
          17.123329639434818
        ],
        "schemaPos": [
          280,
          -240
        ],
        "attributes": {}
      },
      {
        "id": -2,
        "type": "HOUSE",
        "name": "Domácnosť 2",
        "latlong": [
          48.147018690270535,
          17.121226787567142
        ],
        "schemaPos": [
          -40,
          -240
        ],
        "attributes": {
          "timeSeriesId": {
            "name": "Časový rad",
            "type": "text",
            "defaultValue": "HOUSE_LARGE",
            "value": "HOUSE LARGE"
          }
        }
      },
      {
        "id": -3,
        "type": "SOLAR PANEL",
        "name": "Solárny panel 3",
        "latlong": [
          48.14810677231165,
          17.121720314025882
        ],
        "schemaPos": [
          -360,
          -240
        ],
        "attributes": {
          "timeSeriesId": {
            "name": "Časový rad",
            "type": "text",
            "defaultValue": "SOLAR PANEL LARGE",
            "value": "SOLAR_PANEL_LARGE"
          }
        }
      }
    ],
    "links": [
      {
        "from": -3,
```

```
        "to": -2,
        "id": "-3 -2"
      },
      {
        "from": -2,
        "to": -1,
        "id": "-2_-1"
      }
    ]
  },
  "elementLists": [
    "5c07db8bfb6fc038cbb3aa8f"
  ],
  "createdAt": "2019-04-16T20:22:57.647Z",
  "updatedAt": "2019-04-16T20:33:16.422Z",
  "id": "5cb6399d350f0200018a140a"
}
```


Simulácia

Výsledky simulácie sa ukladajú v databáze ako jeden JSON objekt. Má vlastné meno *name*, pole sledovaných prvkov *monitoredElements*, začiatok *to* a koniec *from* simulačného obdobia, a ďalšie metadáta o simulácii.

V *simulationResults* sú potom samotné dáta vrátené od simulačného servera. V *simulationResult* sú výsledky platné pre celú simuláciu, ktoré sa majú zobraziť hneď po otvorení simulácie. V *monitoredElements* sú zasa výsledky pre jednotlivé sledované prvky. Každý element môže obsahovať ľubovoľný počet výsledkov na zobrazenie, ktoré sú uložené v poli *results*. V *monitoredLinks* sa nachádzajú objekty s názvami prislúchajúcimi jednotlivým sledovaným časom. S týchto objektoch sa nachádzajú hodnoty prepojení, ktoré majú nadobúdať v týchto časoch pre potreby vizualizácie tokov. Pole výpisov *logs* obsahuje podobne objekty s názvami podľa sledovaných časov obsahujúce polia výpisov pre daný čas.

Objekt *projectVersion* obsahuje stav projektu v čase spúšťania simulácie


```
{
  "name": "Mala simulacia",
  "logs": [],
  "monitoredElements": [
    -1,
    -2
  ],
  "to": "2019-05-08T08:47:58.592Z",
  "from": "2019-05-07T08:47:58.592Z",
  "graph": null,
  "date": "2019-05-07T08:48:25.695Z",
  "healthStatus": "healhtStatusGood",
  "simulationResults": {
    "simulationResult": [
      {
        "title": "Náklady",
        "graphType": "LINE_GRAPH",
        "statistics": [
          {
            "name": "Priemer na hodinu",
            "value": 0.73,
            "unit": "€"
          },
          {
            "name": "Minimum na hodinu",
            "value": 0.28,
            "unit": "€"
          },
          {
            "name": "Maximum na hodinu",
            "value": 1.18797576,
            "unit": "€"
          }
        ],
        "timeseries": {
          "2019-05-06T23:00:00+01:00": 0.32,
          ...
          "2019-05-07T22:00:00+01:00": 0.28
        }
      },
      {
        "title": "Spotreba",
        "graphType": "LINE_GRAPH",
        "statistics": [
          {
            "name": "Priemer na hodinu",
            "value": 5.693945833333333,
            "unit": "kW/h"
          },
          {
            "name": "Minimum na hodinu",
            "value": 2.15053333333333,
            "unit": "kW/h"
          },
          {
            "name": "Maximum na hodinu",
            "value": 9.0616,
            "unit": "kW/h"
          }
        ],
        "timeseries": {
          "2019-05-06T23:00:00+01:00": 2.66,
          ...
          "2019-05-07T22:00:00+01:00": 2.15
        }
      },
      {
        "title": "Poplatky v tarifách",
        "graphType": "PIE GRAPH",
        "statistics": [
          {
            "name": "Maximálna spotreba v tarife",
            "value": 5.4,
            "unit": "€"
          }
        ],
        "labels": [
          "Ranná",
          "Denná",
          "Špička",
          "Nočná"
        ],
        "values": [
          4.35,
          5.4,
          3.02,
          4.97
        ]
      }
    ],
    "monitoredElements": [
      {
        "elementId": -1,
        "results": [
          {
            "title": "Náklady",
            "graphType": "LINE_GRAPH",
            "statistics": [
              {
                "name": "Priemer na hodinu",
                "value": 0.7386198701388889,
                "unit": "kW/h"
```

```
            },
            {
              "name": "Minimum na hodinu",
              "value": 0.2827951333333329,
              "unit": "kW/h"
            },
            {
              "name": "Maximum na hodinu",
              "value": 1.18797576,
              "unit": "kW/h"
            }
          ],
          "timeseries": {
            "2019-05-06T23:00:00+01:00": 0.32,
            ...
            "2019-05-07T22:00:00+01:00": 0.28
          }
        },
        {
          "title": "Spotreba",
          "graphType": "LINE_GRAPH",
          "statistics": [
            {
              "name": "Priemer na hodinu",
              "value": 5.693945833333333,
              "unit": "kW/h"
            },
            {
              "name": "Minimum na hodinu",
              "value": 2.15053333333333,
              "unit": "kW/h"
            },
            {
              "name": "Maximum na hodinu",
              "value": 9.0616,
              "unit": "kW/h"
            }
          ],
          "timeseries": {
            "2019-05-06T23:00:00+01:00": 2.66,
            ...
            "2019-05-07T22:00:00+01:00": 2.15

        }
      ]
    },
    {
      "elementId": -2,
      "results": [
        {
          "title": "Náklady",
          "graphType": "LINE GRAPH",
          "statistics": [
            {
              "name": "Priemer na hodinu",
              "value": 0.7386198701388889,
              "unit": "kW/h"
            },
            {
              "name": "Minimum na hodinu",
              "value": 0.2827951333333329,
              "unit": "kW/h"
            },
            {
              "name": "Maximum na hodinu",
              "value": 1.18797576,
              "unit": "kW/h"
            }
          ],
          "timeseries": {
            "2019-05-06T23:00:00+01:00": 0.32,
            ...
            "2019-05-07T22:00:00+01:00": 0.28
          }
        },
        {
          "title": "Spotreba",
          "graphType": "LINE GRAPH",
          "statistics": [
            {
              "name": "Priemer na hodinu",
              "value": 5.693945833333333,
              "unit": "kW/h"
            },
            {
              "name": "Minimum na hodinu",
              "value": 2.15053333333333,
              "unit": "kW/h"
            },
            {
              "name": "Maximum na hodinu",
              "value": 9.0616,
              "unit": "kW/h"
            }
```

```
          ],
          "timeseries": {
            "2019-05-06T23:00:00+01:00": 2.66,
            ...
            "2019-05-07T22:00:00+01:00": 2.15
          }
        }
      ]
    }
  ],
  "monitoredLinks": {
    "2019-05-06T23:00:00+0100": {
      "-2_-1": 2.66768333333333
    },
    ...
    "2019-05-07T22:00:00+0100": {
      "-2 -1": 2.15053333333333
    }
  },
  "logs": {
    "2019-05-06T23:00:00+01:00": [
      {
        "logType": "INFO",
        "logMessage": "Simulácia bola úspešne spustená."
      }
    ],
    "2019-05-07T22:00:00+01:00": [
      {
        "logType": "INFO",
        "logMessage": "Simulácia bola úspešne ukončená."
      }
    ]
  }
},
"projectVersion": {
  "name": "Small",
  "id": "5cd14627350f0200018a1f2a",
  "graph": {
    "groups": [],
    "nodes": [
      {
        "id": -1,
        "type": "HOUSE",
        "name": "Domácnosť 1",
        "latlong": [
          48.14591112222222,
          17.1201617
        ],
        "schemaPos": [
          -40,
          -40
        ],
        "attributes": {
          "timeSeriesId": {
            "name": "Časový rad",
            "type": "text",
            "defaultValue": "HOUSE_LARGE",
            "value": "HOUSE_LARGE"
          }
        }
      },
      {
        "id": -2,
        "type": "MAINTRANSFORMER",
        "name": "Hlavný Transformátor 2",
        "latlong": [
          48.14680001111111,
          17.1201617
        ],
        "schemaPos": [
          -40,
          -200
        ],
        "attributes": {}
      }
    ],
    "links": [
      {
        "from": -2,
        "to": -1,
        "id": "-2 -1"
      }
    ]
  },
  "elementLists": [
    "5c07db8bfb6fc038cbb3aa8f"
  ],
  "createdAt": "2019-05-07T08:47:35.427Z",
  "updatedAt": "2019-05-07T08:48:26.642Z"
},
"id": "5cd1465d350f0200018a1f67"
}
```


Webová aplikácia

Webová aplikácia je nástrojom pre modelovanie, simulovanie správania 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.
- **dygraphs** - Knižnica pre vizualizáciu grafov časových radov.

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. POST používame na vytváranie a PUT ako update. 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

Súbory v repozitári databázového servera sú usporiadané nasledovne. V adresári *app* sa nachádzajú python súbory, ktoré reprezentujú jednotlivé endpointy, na ktoré sa dopytuje webová aplikácia. Endpointy sú pomenované konvenciou **<Metóda><Typ kolekcie>.py** napr. PostElementList.py, GetAllProjects.py,...

V adresári *app* sa rovnako nachádza aj ďalší adresár **swagger_docs**, ktorý obsahuje yaml súbory rozdelené podľa metód. Tieto yaml súbory popisujú endpointy, z ktorých sa generuje aj Swagger. Ten je dostupný na webovej adrese http://80.241.209.214:5050/apidocs/ . V súboroch endpointov sa používa anotácia swag_from z knižnice *flasgger* na aktivovanie swaggera pre endpoint.

Súbor **MongoDB.py** obsahuje pripojenie na našu Mongo DB v Mlabe. Okrem connection sú v súbore definované všetky metódy, ktoré pristupujú do kolekcií cez funkcie insert, remove, find a pod. Endpointy sú tvorené spôsobom, že na začiatku sa získa ID objektu z URL (ak sa jedná o také rozhranie - napr. PUT, DELETE, GET). Následne sa zavolá príslušná metóda na kolekciu cez triedu MongoDB a overí sa či bola akcia úspešná. Ak áno, tak vráti result. V prípade POST je to id novopridaného elementu do kolekcie. PUT vracia status kód odpovede 200, ak sa úprava podarila. DELETE má definované 204 v prípade správnosti a GET vracia 200 s tým, že v response je JSON s výsledkom.

Template.yaml definuje dátový model, ktorý používame v aplikácii a zobrazujeme vo swaggeri. V adresári tests sa nachádzajú integračné testy na databázový server. V schemas sú definované schémy v JSON formáte pre element list, simulácie a project. Volania GET all sú validované voči týmto schémam v súbore test-responses.py.

V *util.py* sú pomocné metódy na úpravu ID v JSON. Webová aplikácia pracuje len s key:

```
"id": "5cd06765350f0200018a1acf"
```

"id", ale databáza má uchovaný identifikátor ako

```
"_id": {
      "$oid": "5cd06765350f0200018a1acf"
   }"
```

a preto databázový server cez metódy *deleteKeyFromJson* a *parseToId* vykonáva transformáciu týchto id. Ďalej je definovaná metóda na zisťovanie veľkosti JSON v byte. Zatiaľ sme definovali napevno limit na veľkosť 50MB, pri operáciách POST.

Rozhrania API Servera

V tejto časti sú popísané rozhrania na server pomocou, ktorých sa dá dopytovať do databázy. Príklady JSON na jednotlivé kolekcie a volania sú definované v swaggeri.

Kolekcia Project

Operation	HTTP method	path	returns
List projects	GET	/projects	JSON of project ids + metadata

Operation	HTTP method	path	returns
Create project	POST	/projects	id
Load project	GET	/projects/projectId	project JSON
Save project	PUT	/projects/projectId	200
Delete project	DELETE	/projects/projectId	204
Copy project	POST	/projects/projectId	id

Kolekcia Simulation

Operation	HTTP method	path	returns
List simulations without results	GET	/simulations	JSON of simulations
Create simulation	POST	/simulations	id
Load simulation	GET	/simulations/simulationId	simulation JSON
Save simulation	PUT	/simulations/simulationId	200
Delete simulation	DELETE	/simulations/simulationId	204

Kolekcia Element_types

Operation	HTTP method	path	returns
Load all elements	GET	/elements	JSON of all elements_list
Create element list	POST	/elements	id
Load single element list	GET	/elements /elementListId	element list JSON
Save element list	PUT	/elements/elementListId	200
Delete element list	DELETE	/elements/elementListId	204

Element

Element je súčasťou Element Listu a môže sa nachádzať v poli objektov nodes alebo links, podľa toho, čo daný element definuje.
Príkladom Elementu je napr. Batéria (node) alebo Medené vedenie (link).

Operation	HTTP method	path	returns
Check unique element	GET	/uniqueType	JSON with boolean
Create element	POST	/elements/elementListId	201
Save element	PUT	/elements/elementListId/element/type Parameter	200

Kolekcia Prices

Operation	HTTP method	path	returns
Load single price list	GET	/prices/pricelistId	price list JSON
Save price list	PUT	/prices/pricelistId	200

Kolekcia Icons

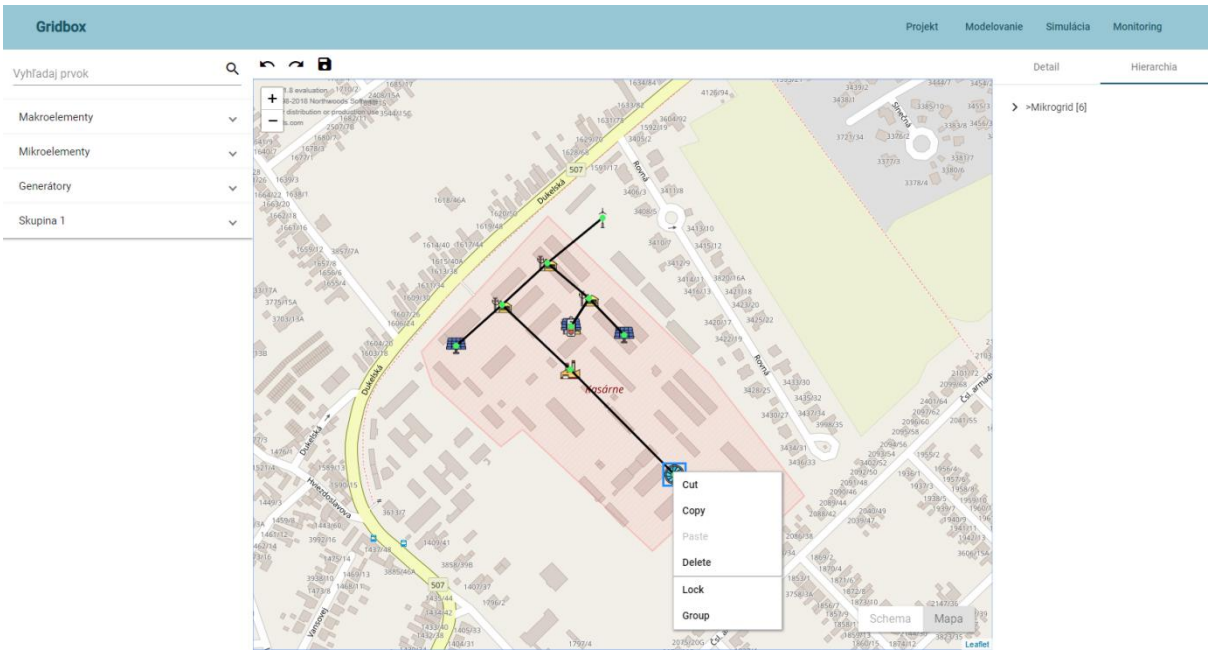
Operation	HTTP method	path	returns
Load icon	GET	/icons/iconId	SVG data
Create icon	POST	/icons	id
Save icon	PUT	/icons/iconId	200
Delete icon	DELETE	/icons/iconId	204

Prototyp

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

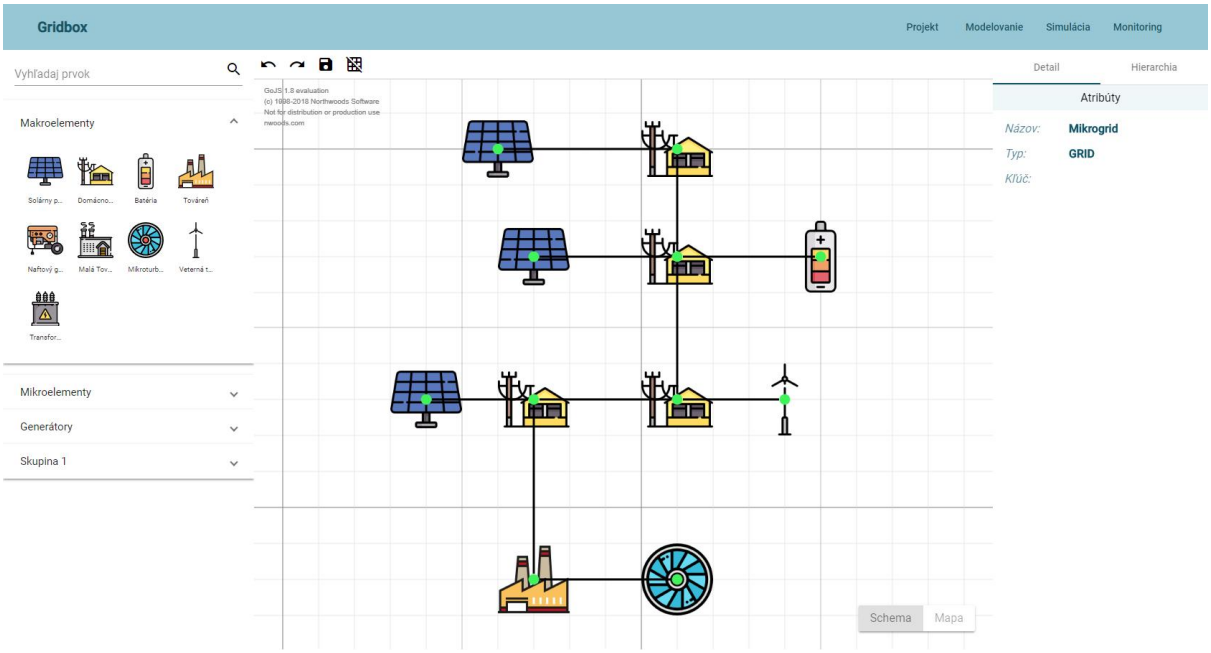
Modelovanie

V rámci modelovania má používateľ v ľavom paneli možnosť vybrať si spomedzi 9 predpripravených 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 4 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).



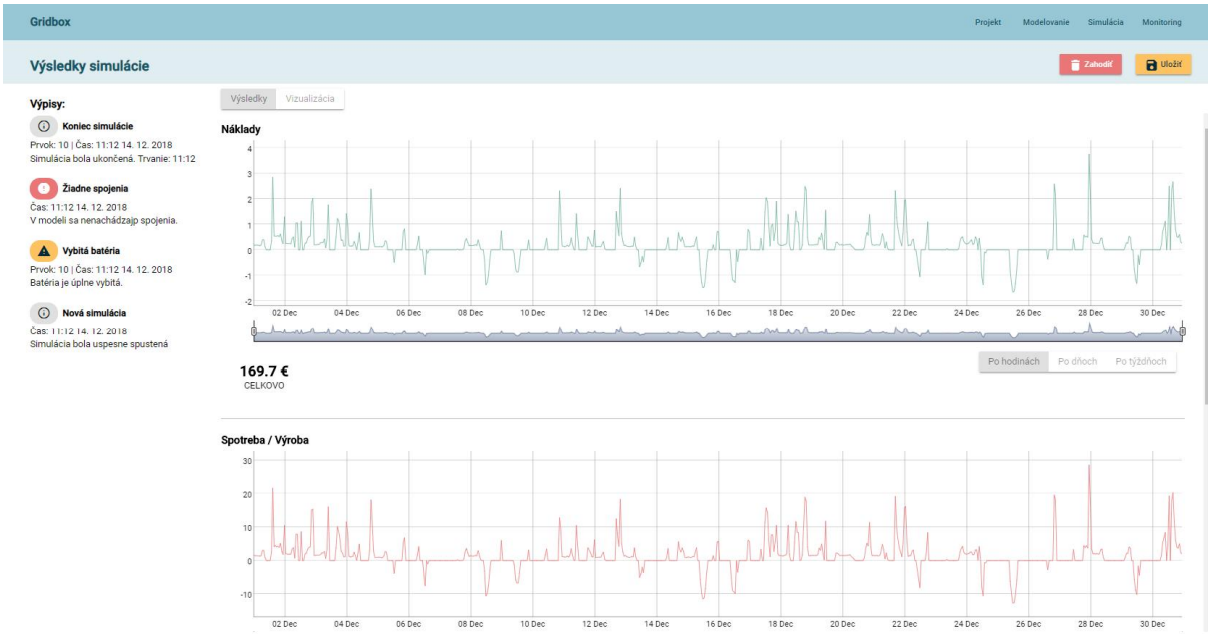
Obrázok 5 GUI pre modelovanie v schéme

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 6 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).



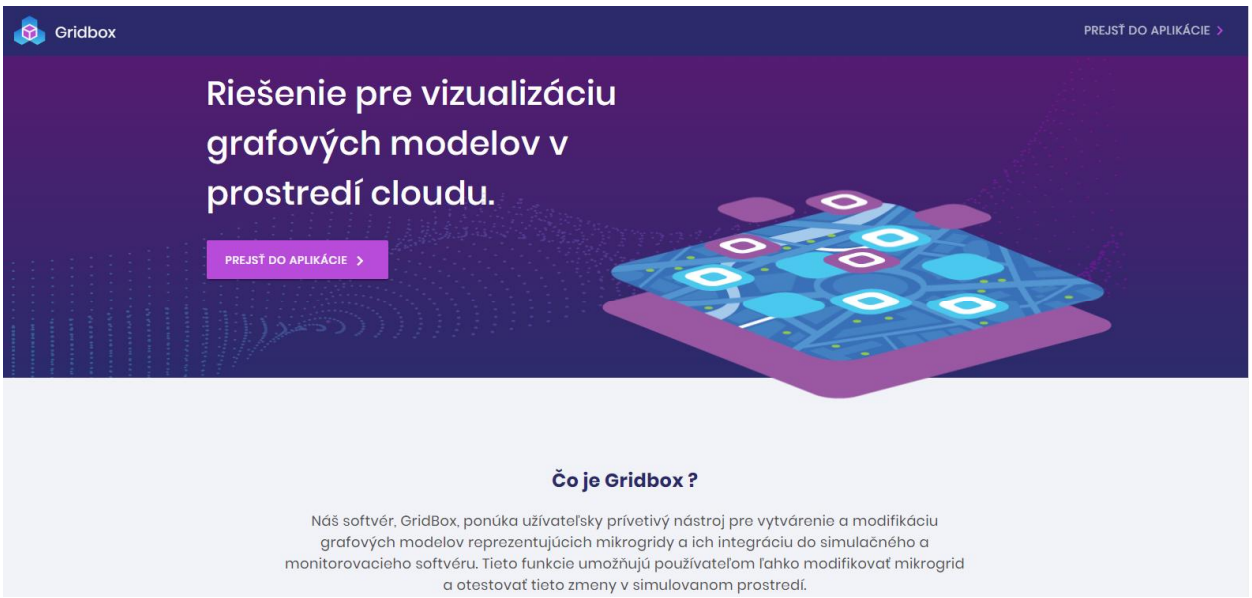
Obrázok 7 GUI pre výsledky simulácie

Príloha A - Používateľská príručka

Táto časť obsahuje používateľskú príručku pre vytvorenú aplikáciu.

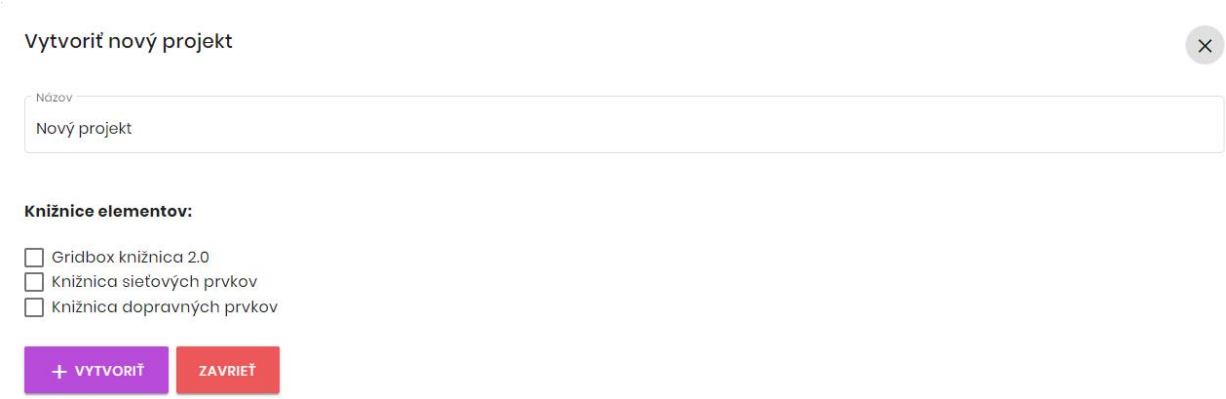
Ako začať

Používateľ začína prácu s aplikáciou na domovskej obrazovke. Tá ponúka základné informácie o vytvorenom riešení, základné prípady použitia, infraštruktúru a dokumentáciu k vytvorenej aplikácií. Pre začatie modelovania zvoľte možnosť "PREJSŤ DO APLIKÁCIE" v pravom hornom rohu alebo v pod hlavným nadpisom v ľavej časti obrazovky.

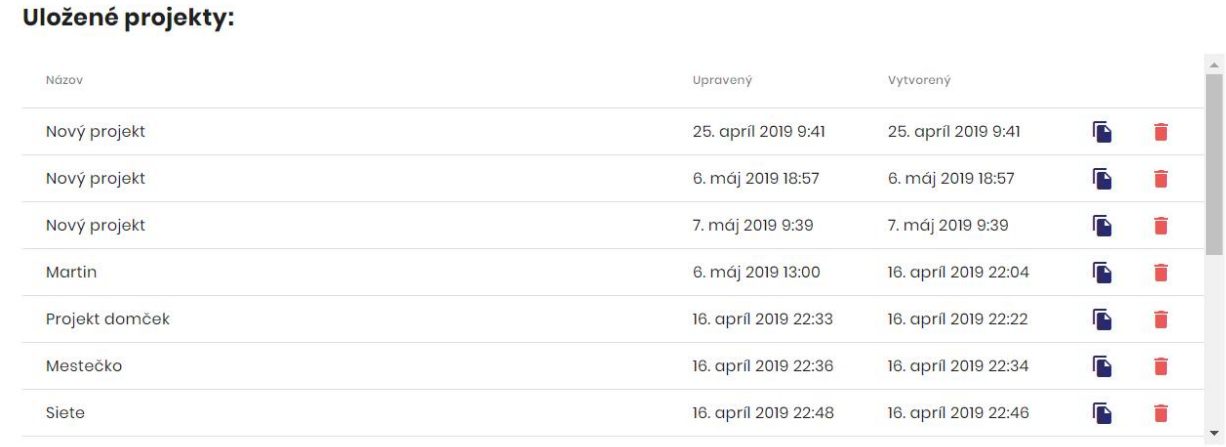


Po kliknutí sa zobrazí úvodná obrazovka, ktorá ponúka možnosť vytvoriť "nový projekt" alebo "otvoriť uložený projekt".

Možnosť "nový projekt" zobrazí vyskakovacie okno s možnosťami vytvárania projektu. Tu si používateľ zvolí názov projektu (preddefinovaný názov je Nový projekt) a knižnice, ktoré chce v projekte používať.



Možnosť "otvoriť uložený projekt" zobrazí vyskakovacie okno s existujúcimi projektami, informáciami o dátume ich vytvorenia a úpravy. Pre otvorenie projektu kliknite na jeho názov. Modrá ikona umožňuje duplikovanie projektu a červená ikona koša vymazanie projektu.



Modelovanie

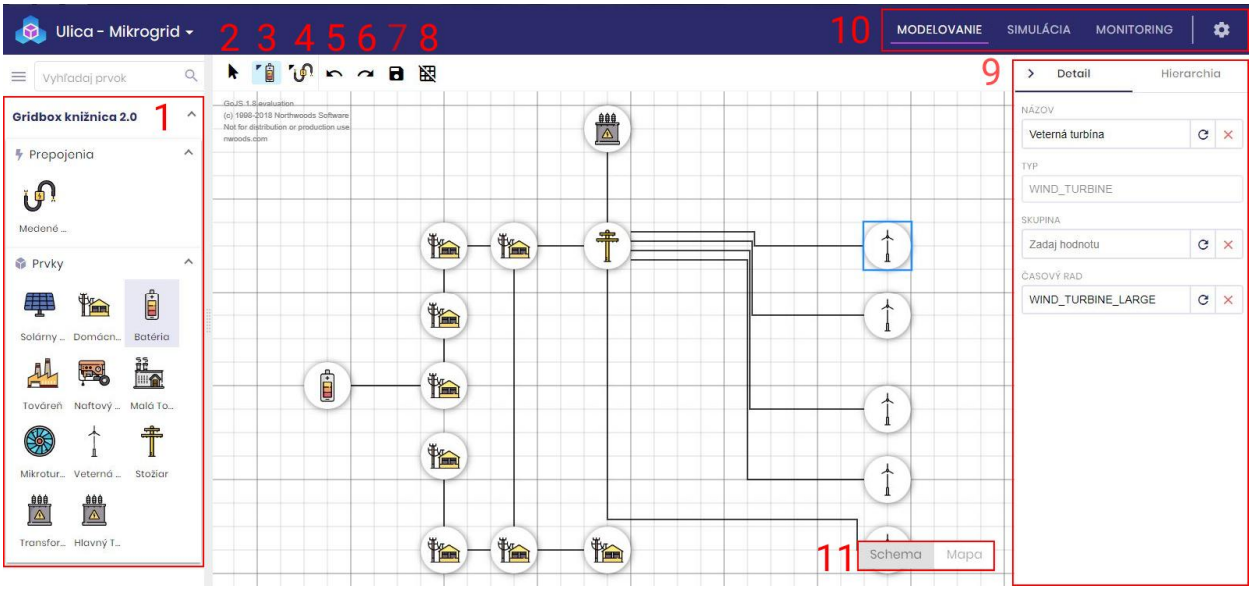
Hlavná časť modelovania sa vykonáva v okne modelovania. Význam jej jednotlivých položiek je nasledovný:

1 - Knižnica prvkov (obsahuje elementy a prepojenia, ktoré je možné pridať do vytváraného modelu)
2 - Návrat na štandardný kurzor
3 - Zvolený prvok
4 - Zvolené prepojenie
5 - Krok späť
6 - Krok dopredu
7 - Uložiť projekt
8 - Schovať/zobraziť mriežku
9 - Detail zvoleného prvku (ten je ohraničený modrým štvorcom), ktorý obsahuje jeho atribúty
10 - Hlavné menu
11 - Prepnutie medzi modelovaním na mape a schéme

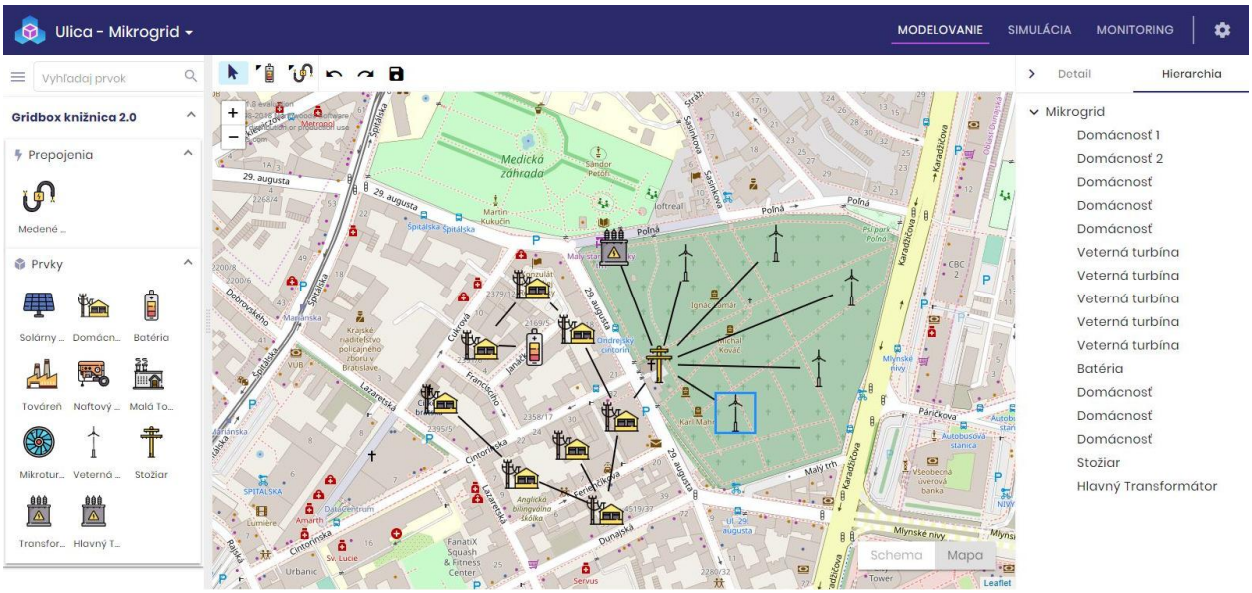
Význam zvyšných položiek:

V ľavom hornom rohu vedľa ikony aplikácie je zobrazený názov projektu (v tomto prípade Ulica - Mikrogrid)
Nižšie sa nachádza vyhľadávacie pole pre vyhľadávanie medzi prvkami
Centrálnu časť obrazovky zaberá samotná modelovacia plocha

Pravé kliknutie na jeden z elementov v modeli zobrazí modulárne okno s možnosťami Vystihnúť, Kopírovať, Prilepiť a Odstrániť.

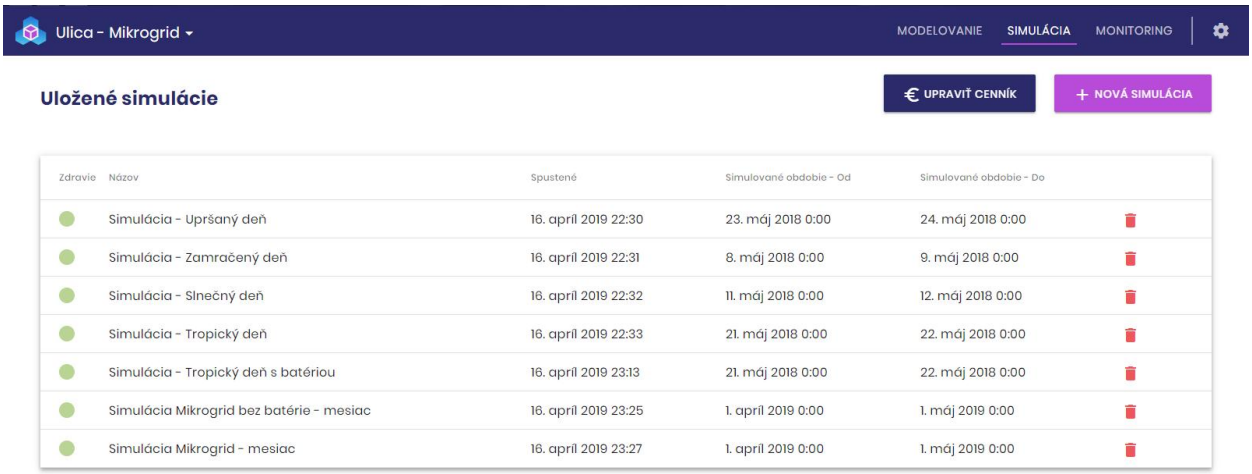


Vyššie opísané členenie platí aj pre modelovanie na mape, tu však chýba tlačidlo pre zobrazenie/skrytie mriežky. Na pravom paneli je okrem záložky "Detail" možné zvoliť záložku "Hierarchia", ktorá zobrazuje hierarchiu všetkých elementov v modeli.



Simulácia

Po zvolení možnosti "SIMULÁCIA" v hlavnom menu sa zobrazí okno "Uložené simulácie". To zobrazuje zoznam uložených simulácií, čas ich spustenia a časový úsek (Simulované obdobie - Od, Simulované obdobie - Do) a možnosť vymazať zvolenú simuláciu.
Tlačidlá:
UPRAVIŤ CENNÍK - úprava aktuálne používaného cenníka
NOVÁ SIMULÁCIA - vytvorenie novej simulácie nad zvoleným modelom



Po kliknutí na NOVÁ SIMULÁCIA sa zobrazí okno s atribútmi simulácie. Tie sú nasledovné:

Parametre simulácie:

- **Názov simulácie**: názov, pod akým bude simulácia označená (preddefinovaný názov je Nová simulácia - dnešný dátum)
- **Začiatok simulácie**: dátum začiatku simulácie v slovenskom formáte
- **Koniec simulácie**: dátum konca simulácie v slovenskom formáte
- **Simulačné jadro**: zvolené simulačné jadro, nad ktorým sa má vykonať simulácia

Restrikcie simulačného jadra

zoznam obmedzení, ktoré musí model spĺňať aby ho bolo možné simulovať na zvolenom simulačnom jadre

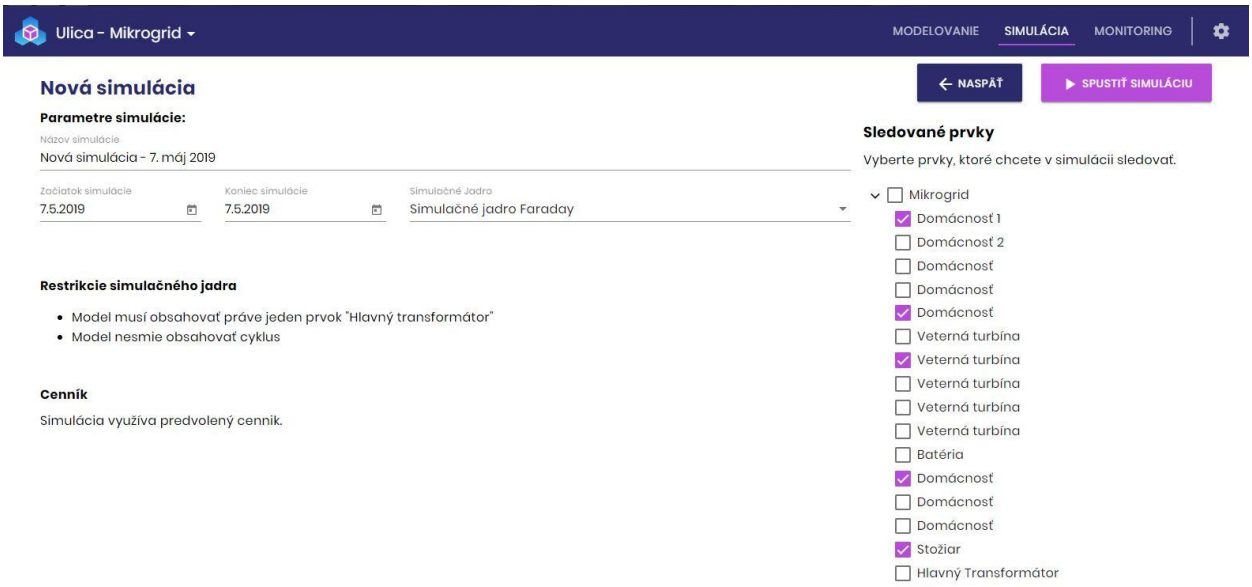
Cenník

informácia o cenníku, ktorý zvolená simulácia používa

Sledované prvky

zoznam prvkov, ktoré chce používateľ v simulácií sledovať (zakliknuté možnosti)

Tlačidlá:
NASPÄŤ - návrat na predchádzajúcu stránku
SPUSTIŤ SIMULÁCIU - spustenie simulácie so zvolenými parametrami

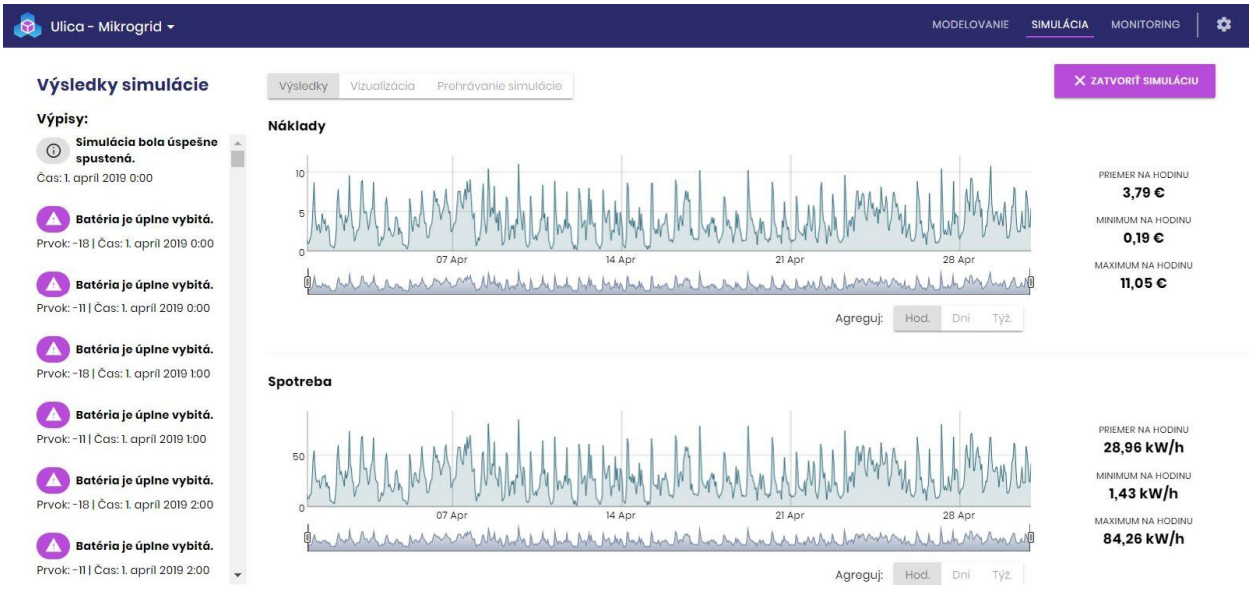


Výsledky simulácie

Po kliknutí na možnosť SPUSTIŤ SIMULÁCIU sa zobrazí okno "Výsledky simulácie".

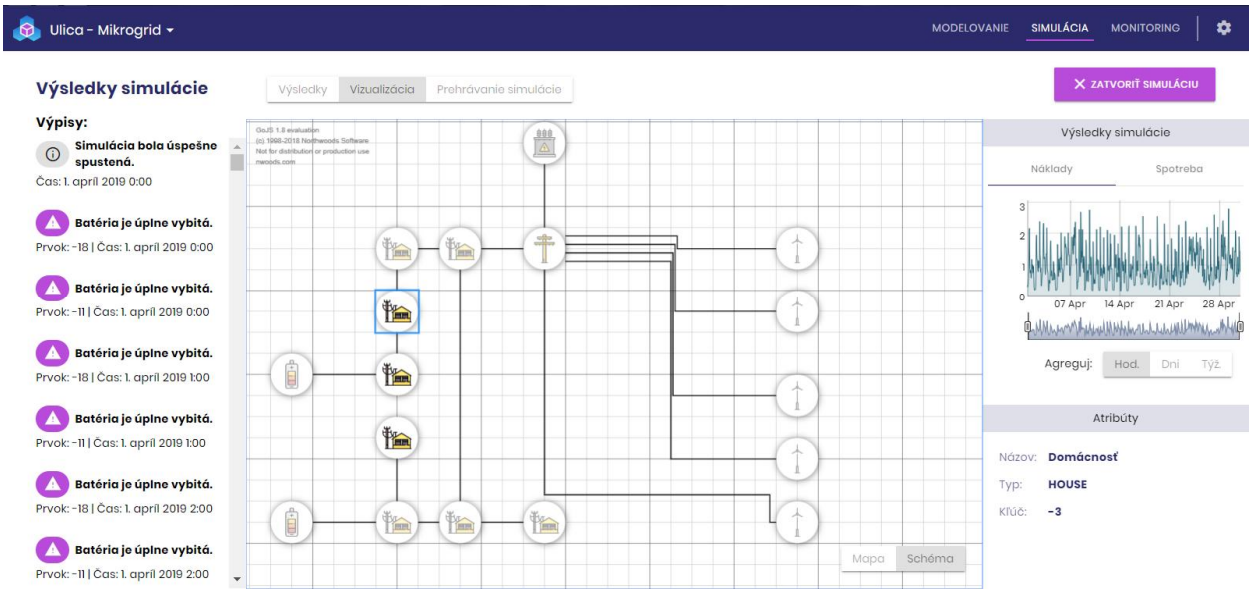
V ľavej časti sa nachádzajú výpisy zo simulácie, ktorá prebehla s informáciami o názve udalosti a čase, kedy daná situácia nastala. Jednotlivé výpisy sú podľa ich závažnosti kategorizované do skupín so samostatnom ikonou pre ich jednoduchšie odlíšenie. Usporiadanie je vzostupne zoradené podľa času. Centrálna časť obsahuje grafické zobrazenie nákladov, spotreby a nákladov v jednotlivých tarifách. Možnosť agreguj zobrazí graf agregovaný buď podľa hodín, dní alebo týždňov. Na pravej strane sú uvedené prídavné výsledky jednotlivých podkategórií simulácie.

ZATVORIŤ SIMULÁCIU - po kliknutí sa zavrie aktuálna simulácia a nastane návrat na obrazovku "Uložené simulácie".



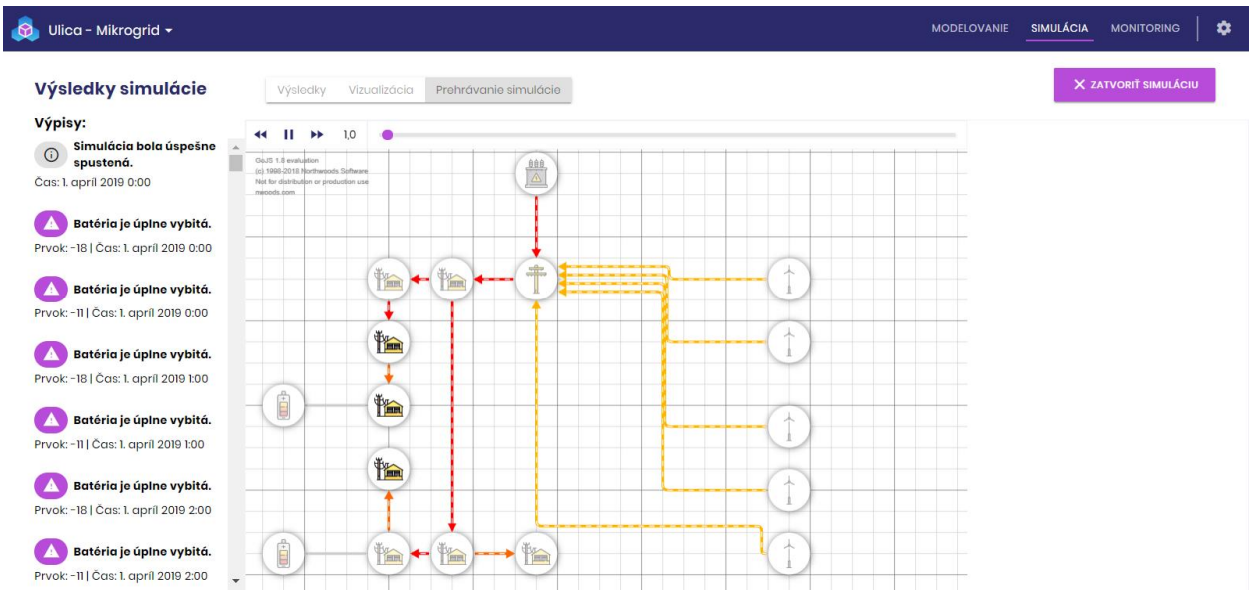
Vizualizácia

Slúži na vizualizovanie prvkov, ktoré používateľ zvolil v parametroch simulácie ako sledované. Dané prvky je možné rozkliknúť, pričom sa v pravom paneli zobrazia informácie zo simulácie analogicky ako v časti Výsledky.



Prehrávanie simulácie

Slúži na prehrávanie simulácie v čase. Farebné orientované prepojenia zobrazujú prúdenie elektriny v jednotlivých časových bodoch, v ktorých simulácia prebiehala.



Nastavenia

Okno nastavení sa zobrazí po kliknutí na ikonu ozubeného kolieska napravo v hlavnom menu.
Tu máte na výber tri kategórie nastavení:

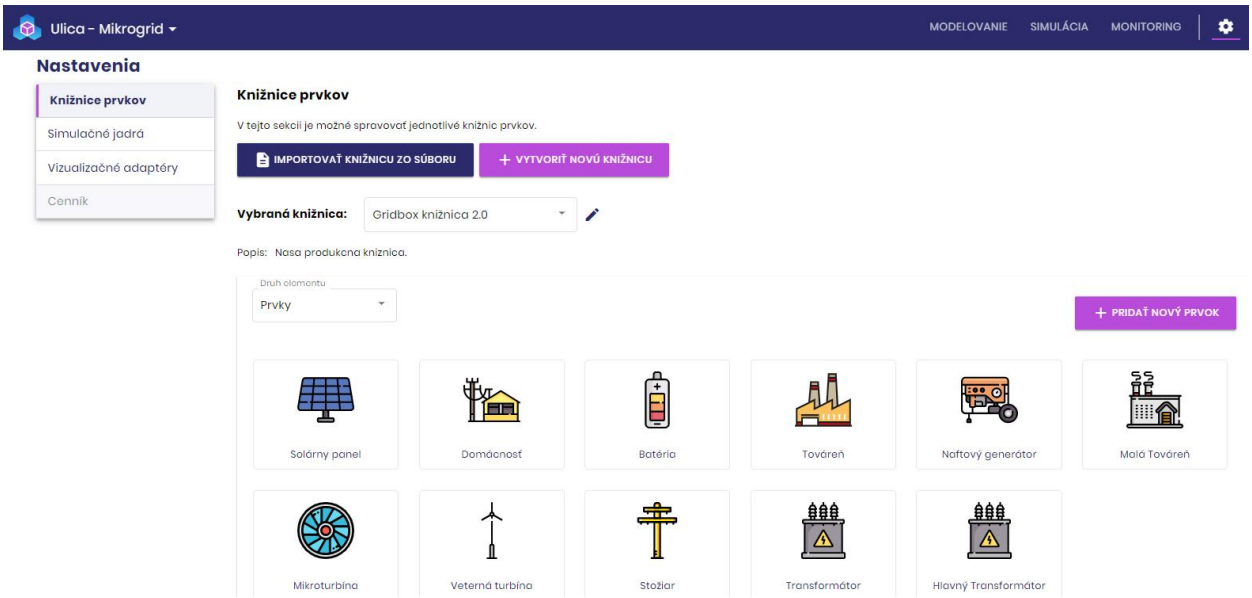
- Knižnice prvkov
- Simulačné jadrá
- Vizualizačné adaptéry

Knižnice prvkov

Tu máte možnosť konfigurácie existujúcej knižnice prvkov alebo vytvorenia novej knižnice prvkov. Význam jednotlivých položiek je nasledovný:

IMPORTOVAŤ KNIŽNICU ZO SÚBORU - načítanie existujúcej knižnice z JSON súboru na lokálnom disku používaného zariadenia
VYTVORIŤ NOVÚ KNIŽNICU - vytvorenie úplne novej knižnice prvkov

Nižšie sa nachádzajú zobrazené prvky alebo spojenia existujúcej knižnice prvkov. Tie je možné upravovať, prípadne pridať nový prvok do existujúcej knižnice prvkov.



Simulačné jadrá

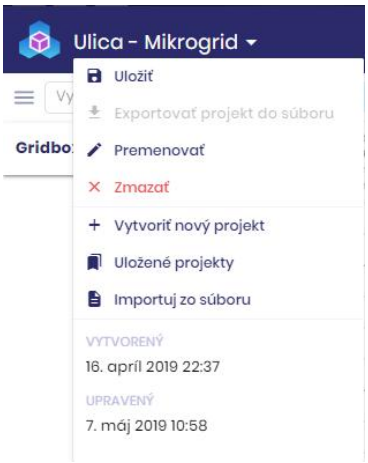
Tu je možné spravovať používané simulačné jadrá. Existujúce simulačné jadrá je možné vymazať, prípadne pridať nové simulačné jadro (je potrebné zadať Názov a IP adresu).

Vizualizačné adaptéry

Tu je možné spravovať používané vizualizačné adaptéry. Existujúce vizualizačné adaptéry je možné vymazať alebo pridať nový vizualizačný adaptér, stačí zadať jeho názov a potvdiť.

Nastavenia projektu

Po kliknutí na názov projektu sa zobrazí modulárne okno so správou daného modelu a informáciami o ňom.



Príloha B - Inštalačná príručka

V tejto príručke nájdete pokyny pre inštaláciu aplikácie a všetkých jej podsystémov.

Mikroslužby

Aplikácia je stavaná do architektúry Mikroslužieb, pričom pozostáva z 4 hlavných komponentov:

- Webová aplikácia
- Webové služby (REST)
- Databáza
- Simulačný server

Do aplikácie je možné pripojiť aj viaceré simulačné jadrá, ktorých napojenie je umožnené v konfigurácii prostredníctvom používateľského rozhrania vo Webovej aplikácii.

Kontajnerizácia - Docker

Pre zabezpečenie izolovaného prostredia a automatizáciu inštalácie sme využili princípy Kontajnerizácie a Docker. Pre každý hlavný komponent existujú Docker image, ktorý je dostupný na verejnom registri Docker image-ov (register nájdete na hub.docker.com/r/powerplayers.

Názvy Docker image-ov pre komponenty:

- Webová aplikácia - *powerplayers/gridbox-webapp*
- Webové služby (REST) - *powerplayers/gridbox-db*
- Simulačný server - *powerplayers/gridbox-simulation*

Spustenie

Pre spustenie kontajnerov je potrebné na výpočtový stroj nainštalovať Docker.

1. Stiahnutie docker image z registru:

```
docker pull powerplayers/<component-name>
```

2. Pre jednotlivé komponenty je potrebné spustiť Docker kontajner z Docker image:

- Webová aplikácia

```
docker run -d --name gridbox-webapp -p 4200:80 powerplayers/gridbox-webapp
```

- Webové služby

```
sudo docker run -d --name gridbox-db -p 5050:5000 powerplayers/gridbox-db
```

- Simulačný server

```
sudo docker run -d --name gridbox-simulation -p 5000:5000 powerplayers/gridbox-simulation
```

Webová aplikácia bude nasledovne nasadená na porte 4200 daného výpočtového stroja.

Príloha C - Export evidencie úloh

Export úloh vygenerovaný zo systému TFS obsahujúci všetky absolvované šprinty.

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 Backlog Item	Analýza možnosti vykreslovania grafov	Bc. Michal 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

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 Item	Testovanie	Bc. Richard 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 webapp	Bc. Richard 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	Integrácia webovej aplikácie a DB	Bc. Frantisek

	Backlog Item		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

Šprint 6

10723	**Product Backlog Item**	**Analýza a návrh - BE**	**Bc. Martin Cincurak**
10721	Task	Logovanie	Bc. Martin Cincurak
10762	Task	Dokumentácia pridania nového simulačného jadra	Bc. Frantisek Durana
10720	Task	Platformovosť - podpora viacerých sim. jadier	Bc. Michal Ostrodicky
10722	Task	Zohľadniť topologiu v simulácií	Bc. Martin Cincurak
10717	Product Backlog Item	Analýza a návrh - FE	Bc. David Pavelka
10718	Task	Výsledky simulácie	Bc. Michal Ostrodicky
10719	Task	Prehrávanie simulácie	Bc. Martin Cincurak
10711	Product Backlog Item	Úprava FE webovej aplikácie	Bc. Richard Mocak
10713	Task	Zmena rozloženia modelovacej schémy	Bc. Richard Mocak
10714	Task	Úprava panelu s prvkami	Bc. David Pavelka
10724	Task	Úprava vrchnej lišty - meno projektu	Bc. Richard Mocak
10716	Task	Upraviť formátovanie času	Bc. Richard Mocak
10725	Product Backlog Item	Zmena landing page	Bc. Richard Mocak
10645	Task	Logo aplikácie	Bc. Richard Mocak
10712	Task	Presun "Projekt" stránky do sub-menu	Bc. David Pavelka
10644	Task	Vyvtvoriť landing page	Bc. Richard Mocak
10047	Product Backlog	Implementácia chýbajúcich funkcií úpravy modelu	Bc. David

	Item		Pavelka
10245	Task	Pridanie vstupno-výstupného prvku trafa	Bc. Frantisek Durana
10715	Task	Pridanie nových prvkov	Bc. David Pavelka

Šprint 7

10931	**Product Backlog Item**	**Analýza a návrh**	**Bc. Peter Pavlik**
10932	Task	Univerzálna konfigurácia vizualizácie	Bc. Peter Pavlik
10936	Task	Generalizovať výsledky simulácie	Bc. Matej Prochazka
11002	Task	Swagger dokumentacia DB Servera	Bc. Michal Ostrodicky
10068	Product Backlog Item	Zložitejšia simulácia	Bc. Martin Cincurak
10927	Task	Analýza a návrh	Bc. Martin Cincurak
10928	Task	Prototyp druhého jadra	Bc. Martin Cincurak
10930	Task	Dátový model simulácei	Bc. Martin Cincurak
10933	Product Backlog Item	Úprava FE	Bc. David Pavelka
10766	Task	Fix select all pri simulácií	Bc. David Pavelka
10962	Task	Pridanie tlačidiel pre výber SJ pri vytváraní simulácie	Bc. David Pavelka
10963	Task	Pridanie tlačidla späť pri vytváraní simulácie	Bc. David Pavelka
10964	Task	Úprava digraph	Bc. Peter Pavlik
10967	Task	Config stránka FE	Bc. David Pavelka
10935	Product Backlog Item	Úprava dizajnu	Bc. Richard Mocak
10941	Task	Úprava panelu s prvkami	Bc. Richard Mocak
10942	Task	Úprava panelu s atribútmi	Bc. Richard Mocak
10943	Task	Prepínanie režimov kurzoru pri modelovaní	Bc. Richard Mocak
10944	Task	Rozšírené výsledky zo simulácie	Bc. Frantisek Durana

Šprint 8

11140	**Product Backlog Item**	**Simulácia**	**Bc. Martin Cincurak**
11316	Task	Cykli v grafe	Bc. Martin Cincurak
11317	Task	Hlavný prvok v grafe	Bc. Martin Cincurak
11318	Task	Nasadenie novej verzie serveru	Bc. Martin Cincurak
11202	Product Backlog Item	Príprava na PT	Bc. Frantisek Durana
11204	Task	Odlíšiť prepojenia	Bc. Richard Mocak
11205	Task	Monitoring WIP page	Bc. Frantisek Durana
11206	Task	Dátumy simulácie	Bc. David Pavelka
11208	Task	Vysledky ak nie su ziadne sledovane elementy + select all fix	Bc. Peter Pavlik
11235	Task	Pravý panel bez zvoleného prvku	Bc. Richard Mocak
10641	Product Backlog Item	Interné tímové úlohy	Bc. Matej Prochazka
10764	Task	Pridať footer ikonkám	Bc. Peter Pavlik
10819	Task	Zarovnanie prvku po vložení	Bc. David Pavelka

Šprint 9

11113	**Product Backlog Item**	**Správa prvkov FE**	**Bc. Richard Mocak**
11264	Task	Pridať správu prvkov do konfiguračnej stránky	Bc. Richard Mocak
11265	Task	Správa knižníc	Bc. Richard Mocak
11333	Task	Nahrávanie obrázku pre ikonu	Bc. Frantisek Durana
11602	Task	Oprava bugov	Bc.

			Frantisek Durana
11112	Product Backlog Item	Správa prvkov BE	Bc. Michal Ostrodicky
11143	Task	Vytvorenie novej kniznice prvkov	Bc. Michal Ostrodicky
11144	Task	Uprava existujucej kniznice prvkov	Bc. Michal Ostrodicky
11145	Task	Vytvorenie novej ikony	Bc. Michal Ostrodicky
11146	Task	Uprava existujucej ikony	Bc. Michal Ostrodicky
11147	Task	Swagger update	Bc. Michal Ostrodicky
11376	Task	End-point check unique name	Bc. Michal Ostrodicky
11377	Task	Put element	Bc. Michal Ostrodicky
11378	Task	POST Element	Bc. Michal Ostrodicky
11365	Product Backlog Item	Príprava na TP-Cup	Bc. Peter Pavlik
11366	Task	Príprava videa	Bc. Martin Cincurak
11369	Task	Článok na robime.it	Bc. Peter Pavlik
11371	Task	Dotazník	Bc. Matej Prochazka
11361	Product Backlog Item	Prehrávanie simulácie	Bc. Peter Pavlik
11319	Task	Hodnoty prepojení v čase	Bc. Martin Cincurak
11372	Task	Stránka pre zobrazenie prehrávania simulácie	Bc. David Pavelka
11373	Task	Service pre prehrávanie simulácie	Bc. David Pavelka
11374	Task	GoJS animácia pre prehrávanie simulácie	Bc. Richard Mocak
11375	Task	Zobrazenie detailov pri prehrávaní	Bc. Peter Pavlik
10067	Product Backlog Item	Logovanie udalostí v simulácii	Bc. Martin Cincurak
10945	Task	Vytvaranie logov zo simulacie	Bc. Martin Cincurak
10947	Task	Zobrazenie zoznamu logov vo FE	Bc. Matej Prochazka

Šprint 10

11113	**Product Backlog**	**Správa prvkov FE**	**Bc. Richard**

	Item		Mocak
11266	Task	Upraviť / vytvoriť nový prvok	Bc. Richard Mocak
11332	Task	Upraviť / vytvoriť nový atribút	Bc. Matej Prochazka
11574	Task	Vyber knižníc pri vytvorení nového projektu	Bc. Richard Mocak
11575	Task	Uprava zvolených knižníc v projekte	Bc. Richard Mocak
11628	Task	Úprava dátového modelu	Bc. Michal Ostrodicky
11365	Product Backlog Item	Príprava na TP-Cup	Bc. Peter Pavlik
11367	Task	Pripraviť poster	Bc. Richard Mocak
11621	Task	Príprava produkčného prostredia	Bc. Frantisek Durana
11623	Task	Príprava nových knižníc	Bc. Martin Cincurak
11624	Task	Záloha	Bc. Michal Ostrodicky
11625	Task	Prichystať dáta	Bc. Matej Prochazka
11627	Task	End-to-End test	Bc. Peter Pavlik
11742	Task	Úprava stránky Monitoring	Bc. David Pavelka
11743	Task	Context Menu fix	Bc. David Pavelka
11744	Task	Doplnit upozornenie na restrikcie sim. jadra	Bc. Martin Cincurak
10641	Product Backlog Item	Interné tímové úlohy	Bc. Matej Prochazka
11629	Task	Ošetriť cykly v modeli	Bc. David Pavelka
11630	Task	Upraviť vizualizácie výsledkov	Bc. Peter Pavlik
11641	Task	Schovať veľkosť projektu	Bc. Michal Ostrodicky

Šprint 11

11639	**Product Backlog Item**	**Dokumentácia**	**Bc. Frantisek Durana**
11848	Task	Riadenie projektu	Bc.

			Frantisek Durana
11901	Task	Priprava repozitarov na odovzdanie + dump DB	Bc. Richard Mocak
11902	Task	Update timovej stranky a priprava na odovzdanie	Bc. David Pavelka
11914	Task	Dokumentácia úvodnej časti Inžinierskeho diela	Bc. Frantisek Durana
11958	Task	Používateľská príručka	Bc. Frantisek Durana
11961	Task	Finalizácia dokumentácie	Bc. Frantisek Durana
11962	Task	Export evidencie úloh	Bc. Frantisek Durana
11996	Task	Tvorba dokumentácie Databázového servera	Bc. Michal Ostrodicky
12000	Task	Tvorba dokumentácie simulačného servera	Bc. Martin Cincurak
12001	Task	Aktualizácia dátového modelu v dokumentácií	Bc. Peter Pavlik
11113	Product Backlog Item	Správa prvkov FE	Bc. Richard Mocak
11817	Task	FIX: Vytvorenie / ukladanie prvku	Bc. Matej Prochazka
11818	Task	FIX: Nacitavanie ikony	Bc. Matej Prochazka

Príloha D - Technická dokumentácia

Táto časť obsahuje vygenerovanú technickú dokumentáciu pre API rozhrania.

Errors Open

Gridbox SWAGGER

Swagger documentation of Gridbox API

Version 0.2.0

Contact information
tp_06_2018@googlegroups.com (mailto:tp_06_2018@googlegroups.com)

Paths

`/elements/`

GET /elements/

Element list

Summary

GET all Element types list

Description

Endpoint for getting all element type lists from Collection Element types

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Element List {** description: string id: string links: ▸[] name: string nodes: ▸[] }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

POST /elements/

Element list

Summary

POST Element list

Description

Endpoint for creating new element list into Collection Element List

Parameters

Name	Located in	Description	Required	Schema
body	body	Element list that needs to be saved into Database	Yes	⇄ ▾ **Element List {** description: string id: string links: ▸[] name: string nodes: ▸[] }

Responses

Code	Description	Examples
201	CREATED. Returned id in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

/elements/{elementListId}

GET /elements/{elementListId}

Element list

Summary

GET single Element type list

Description

Endpoint for getting single element list from Collection Element Types

Parameters

Name	Located in	Description	Required	Schema
elementListId	path	Id of the element list that needs to be retrieved	Yes	⇄ string

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Element List {** description: string id: string links: ▸[] name: string nodes: ▸[] }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

POST /elements/{elementListId}

Element

Summary

POST Element

Description

Endpoint for creating new element into Element List

Parameters

Name	Located in	Description	Required	Schema
elementListId	path	Id of the Element list that needs to be changed	Yes	⇄ string
body	body	Creating new Element in Element list	Yes	⇄ ▾ **Element {** attributes: ▸ **Element_attributes { }** elementType: string iconId: string name: string type: string }

Responses

Code	Description
201	Element CREATED.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

PUT /elements/{elementListId}

Element list

Summary

PUT Element list

Description

Endpoint for changing element list from Collection Element list

Parameters

Name	Located in	Description	Required	Schema
elementListId	path	Id of the element list that needs to be changed	Yes	⇄ string
body	body	Element list that needs to be updated	Yes	⇄ ▾ **Element List {** description: string id: string links: ▸[] name: string nodes: ▸[] }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

DELETE /elements/{elementListId}

Element list

Summary

DELETE single Element list

Description

Endpoint for deleting element list from Collection Element types

Parameters

Name	Located in	Description	Required	Schema
elementListId	path	Id of the element list that needs to be deleted	Yes	⇄ string

Responses

Code	Description
204	No Content. Project was successfully deleted.
400	Bad Request
404	Not Found
405	Validation exception
409	Conflict
500	Internal Server Error

Try this operation

/elements/{elementListId}/element/{typeParameter}

PUT /elements/{elementListId}/element/{typeParameter}

Element

Summary

PUT Element

Description

Endpoint for changing Element in Element list

Parameters

Name	Located in	Description	Required	Schema
elementListId	path	Id of the Element list that needs to be changed	Yes	⇄ string
typeParameter	path	Unique type of the Element that needs to be changed	Yes	⇄ string
body	body	Element list with concrete Element that needs to be updated	Yes	⇄ ▾ **Element {** attributes: ▸ **Element_attributes { }** elementType: string iconId: string name: string type: string }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

/icons/

POST /icons/

Icon

Summary

POST Icon

Description

Endpoint for creating new icon into Collection Projects

Parameters

Name	Located in	Description	Required	Schema
body	body	Icon that needs to be saved into Database	Yes	⇄ ▾ Icon { id: string name: string svgData: string }

Responses

Code	Description	Examples
201	CREATED. Returned id in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

/icons/{iconId}

GET /icons/{iconId}

Icon

Summary

GET single Icon

Description

Endpoint for retrieving icon into app UI

Parameters

Name	Located in	Description	Required	Schema
iconId	path	Id of the icon that needs to be retrieved	Yes	⇄ string

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ Icon { id: string, name: string, svgData: string }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

PUT /icons/{iconId}

Icon

Summary

PUT Icon

Description

Endpoint for changing icon from Collection Icons

Parameters

Name	Located in	Description	Required	Schema
iconId	path	Id of the icon that needs to be changed	Yes	⇄ string
body	body	Icon that needs to be updated	Yes	⇄ ▾ Icon { id: string name: string svgData: string }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

DELETE /icons/{iconId}

Icon

Summary

DELETE single Icon

Description

Endpoint for deleting icon from Collection Icons

Parameters

Name	Located in	Description	Required	Schema
iconId	path	Id of the icon that needs to be deleted	Yes	⇄ string

Responses

Code	Description
204	No Content. Project was successfully deleted.
400	Bad Request
404	Not Found
405	Validation exception
409	Conflict
500	Internal Server Error

Try this operation

/prices/{priceListId}

GET /prices/{priceListId}

Price list

Summary

GET single Price list

Description

Endpoint for getting single price list from Collection Price list

Parameters

Name	Located in	Description	Required	Schema
priceListId	path	Id of the price list that needs to be retrieved	Yes	⇄ string

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Price {** Friday: ▸[] Holiday: ▸[] Monday: ▸[] Saturday: ▸[] Sunday: ▸[] Thursday: ▸[] Tuesday: ▸[] Wednesday: ▸[] id: string }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

/prices/{pricelistId}

PUT /prices/{pricelistId}

Price list

Summary

PUT Price list

Description

Endpoint for changing price list from Collection Price list

Parameters

Name	Located in	Description	Required	Schema
pricelistId	path	Id of the price list that needs to be changed	Yes	⇄ string
body	body	Price list that needs to be updated	Yes	⇄ ▾ **Price {** Friday: ▸[] Holiday: ▸[] Monday: ▸[] Saturday: ▸[] Sunday: ▸[] Thursday: ▸[] Tuesday: ▸[] Wednesday: ▸[] id: string }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

/projects/

GET /projects/

Project

Summary

GET all created projects

Description

Endpoint for getting all projects from Collection Projects

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Project {** createdAt: string elementLists: ▸[] graph: ▸ **Project_graph { }** id: string updatedAt: string }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

POST /projects/

Project

Summary

POST Project

Description

Endpoint for creating new project into Collection Projects

Parameters

Name	Located in	Description	Required	Schema
body	body	Project that needs to be saved into Database	Yes	⇄ ▾ **Project {** createdAt: string elementLists: ▸[] graph: ▸ **Project_graph { }** id: string updatedAt: string }

Responses

Code	Description	Examples
201	Returned JSON in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

/projects/{projectId}

GET /projects/{projectId}

Project

Summary

GET single Project

Description

Endpoint for getting single project from Collection Projects

Parameters

Name	Located in	Description	Required	Schema
projectId	path	Id of the project that needs to be retrieved	Yes	⇄ string

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Project {** createdAt: string elementLists: ▸ [] graph: ▸ **Project_graph { }** id: string updatedAt: string }	application/json ▸ Object
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

POST /projects/{projectId}

Project

Summary

POST Copy Project

Description

Endpoint for copying existing project

Parameters

Name	Located in	Description	Required	Schema
projectId	path	Id of the project that needs to be copied	Yes	⇄ string

Responses

Code	Description	Examples
201	Returned JSON in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

PUT /projects/{projectId}

Project

Summary

PUT Project

Description

Endpoint for changing project from Collection Projects

Parameters

Name	Located in	Description	Required	Schema
projectId	path	Id of the project that needs to be changed	Yes	⇄ string
body	body	Project that needs to be updated	Yes	⇄ ▾ **Project {** createdAt: string elementLists: ▸[] graph: ▸ **Project_graph { }** id: string updatedAt: string }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

DELETE /projects/{projectId}

Project

Summary

DELETE single Project

Description

Endpoint for deleting project from Collection Projects

Parameters

Name	Located in	Description	Required	Schema
projectId	path	Id of the project that needs to be deleted	Yes	⇄ string

Responses

Code	Description
204	No Content. Project was successfully deleted.
400	Bad Request
404	Not Found
405	Validation exception
409	Conflict
500	Internal Server Error

Try this operation

/simulations/

GET /simulations/

Simulation

Summary

GET all created simulations

Description

Endpoint for getting all simulations from Collection Simulations

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾**Simulation {** date: string from: string graph: string healthStatus: string id: string monitoredElements: ▸**[]** name: string projectVersion: ▸**Project { }** simulationResults: ▸ **Simulation_simulationResults { }** to: string }	application/j
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

POST /simulations/

Simulation

Summary

POST Simulation

Description

Endpoint for creating new simulation into Collection Simulations

Parameters

Name	Located in	Description	Required	Schema
body	body	Simulation that needs to be saved into Database	Yes	⇄

```
▾ Simulation {
    date:               string
    from:               string
    graph:              string
    healthStatus:       string
    id:                 string
    monitoredElements:  ▸[]
    name:               string
    projectVersion:     ▸Project { }
    simulationResults:  ▸
                        Simulation_simulationResul
                        { }
    to:                 string
}
```

Responses

Code	Description	Examples
200	Returned JSON in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

/simulations/{simulationId}

GET /simulations/{simulationId}

Simulation

Summary

GET single Simulation

Description

Endpoint for getting single simulation from Collection Simulations

Parameters

Name	Located in	Description	Required	Schema
simulationId	path	Id of the simulation that needs to be retrieved	Yes	⇄ string

Responses

Code	Description	Schema	Examples
200	Returned JSON in response body	⇄ ▾ **Simulation {** date: string from: string graph: string healthStatus: string id: string monitoredElements: ▸[] name: string projectVersion: ▸**Project { }** simulationResults: ▸ **Simulation_simulationResults { }** to: string }	application/j
400	Bad Request		
404	Not Found		
405	Validation exception		
500	Internal Server Error		

Try this operation

PUT /simulations/{simulationId}

Simulation

Summary

PUT Simulation

Description

Endpoint for changing simulation from Collection Simulations

Parameters

Name	Located in	Description	Required	Schema
simulationId	path	Id of the simulation that needs to be changed	Yes	⇄ string
body	body	Simulation that needs to be updated	Yes	⇄ ▾ **Simulation {** date: string from: string graph: string healthStatus: string id: string monitoredElements: ▸[] name: string projectVersion: ▸**Project { }** simulationResults: ▸ **Simulation_simulat { }** to: string }

Responses

Code	Description
200	Successful operation.
400	Bad Request
404	Not Found
405	Validation exception
500	Internal Server Error

Try this operation

DELETE /simulations/{simulationId}

Simulation

Summary

DELETE single Simulation

Description

Endpoint for deleting simulation from Collection Simulations

Parameters

Name	Located in	Description	Required	Schema
simulationId	path	Id of the simulation that needs to be retrieved	Yes	⇄ string

Responses

Code	Description
204	No Content. Simulation was successfully deleted.
400	Bad Request
404	Not Found
405	Validation exception
409	Conflict
500	Internal Server Error

Try this operation

/uniqueType/

GET /uniqueType/

Element

Summary

GET Unique name

Description

Endpoint for checking unique type in Element list

Parameters

Name	Located in	Description	Required	Schema
type	path	Name of Element type	Yes	⇄ string

Responses

Code	Description	Examples
200	Returned JSON in response body	application/json ▸ Object
400	Bad Request	
404	Not Found	
405	Validation exception	
500	Internal Server Error	

Try this operation

Models

Element

```
▾ Element {
    attributes:   ▸Element_attributes { }
    elementType: string
⇄   iconId:      string
    name:        string
    type:        string
  }
```

Element List

```
▾ Element List {
    description: string
    id:          string
⇄   links:       ▸[]
    name:        string
    nodes:       ▸[]
  }
```

Icon

```
▾ Icon {
    id:      string
⇄   name:    string
    svgData: string
  }
```

Price

```
▾ Price {
    Friday:    ▸[]
    Holiday:   ▸[]
    Monday:    ▸[]
    Saturday:  ▸[]
⇄   Sunday:    ▸[]
    Thursday:  ▸[]
    Tuesday:   ▸[]
    Wednesday: ▸[]
    id:        string
  }
```

Project

```
▾ Project {
    createdAt:    string
    elementLists: ▸[]
⇄   graph:        ▸Project_graph { }
    id:           string
    updatedAt:    string
  }
```

Simulation

```
▾ Simulation {
    date:              string
    from:              string
    graph:             string
    healthStatus:      string
    id:                string
⇄   monitoredElements: ▸[]
    name:              string
    projectVersion:    ▸Project { }
    simulationResults: ▸Simulation_simulationResults { }
    to:                string
  }
```

Element_attributes

```
▾ Element_attributes {
    defaultValue: number
⇄   name:         string
    unit:         string
}
```

Element List_links

```
▾ Element List_links {
    attributes:   ▸Element_attributes { }
    color:        string
    elementType: string
⇄   iconId:       string
    menuSection: string
    name:         string
    type:         string
}
```

Element List_attributes

```
▾ Element List_attributes {
    defaultValue: number
⇄   name:         string
}
```

Element List_nodes

```
▾ Element List_nodes {
    attributes:   ▸Element List_attributes { }
    elementType: string
    iconId:       string
⇄   menusection: string
    name:         string
    type:         string
}
```

Project_graph_links

```
▾ Project_graph_links {
    from: string
⇄   id:   string
    to:   string
}
```

Project_graph_nodes

```
▾ Project_graph_nodes {
    id:       string
    latlong:  ▸[]
⇄   name:     string
    schemaPos: ▸[]
    type:     string
  }
```

Project_graph

```
▾ Project_graph {
    groups: ▸[]
⇄   links:  ▸[]
    nodes:  ▸[]
  }
```

Simulation_simulationResults_logs_timestamp

```
▾ Simulation_simulationResults_logs_timestamp {
    linkId:  string
    message: string
⇄   nodeId:  string
    title:   string
    type:    string
  }
```

Simulation_simulationResults_logs

```
▾ Simulation_simulationResults_logs {
⇄   timestamp: ▸[]
  }
```

Simulation_simulationResults_statistics

```
▾ Simulation_simulationResults_statistics {
    name:  string
⇄   unit:  string
    value: number
  }
```

Simulation_simulationResults_timeseries

```
▾ Simulation_simulationResults_timeseries {
⇄   timestamp: number
  }
```

Simulation_simulationResults_results

```
▾ Simulation_simulationResults_results {
    graphType:  string
    statistics: ▸[]
⇄   timeseries: ▸Simulation_simulationResults_timeseries { }
    title:      string
  }
```

Simulation_simulationResults_monitoredElements

```
▾ Simulation_simulationResults_monitoredElements {
    elementId: string
⇄   results:    ▸[]
  }
```

Simulation_simulationResults_monitoredLinks_timestamp

```
▾ Simulation_simulationResults_monitoredLinks_timestamp {
⇄   linkId: number
  }
```

Simulation_simulationResults_monitoredLinks

```
▾ Simulation_simulationResults_monitoredLinks {
⇄   timestamp: ▸Simulation_simulationResults_monitoredLinks_timestamp { }
  }
```

Simulation_simulationResults

```
▾ Simulation_simulationResults {
    logs:              ▸Simulation_simulationResults_logs { }
    monitoredElements: ▸[]
⇄   monitoredLinks:    ▸Simulation_simulationResults_monitoredLinks { }
    simulationResult:  ▸[]
  }
```