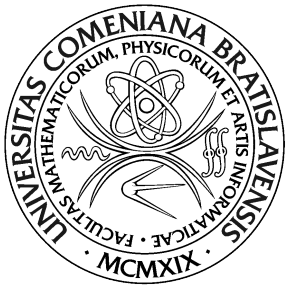
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

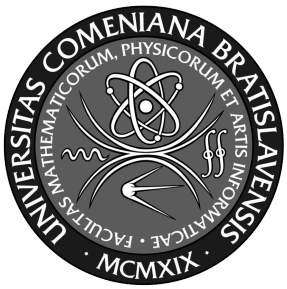


KLABORATÍVNY GRAFICKÝ EDITOR PRE MEDIAWIKI

Diplomová práca

2018 Bc. Martin Krasňan

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



KLABORATÍVNY GRAFICKÝ EDITOR PRE MEDIAWIKI

Diplomová práca

Študijný program:	Aplikovaná informatika
Študijný odbor:	2511 Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	doc. RNDr. Zuzana Kubincová, PhD.
Konzultant:	Mgr. Ján Kľuka, PhD.

Bratislava, 2018 Bc. Martin Krasňan

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Krasňan
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Klaboratívny grafický editor pre MediaWiki
Collaborative graphics editor for MediaWiki

Cieľ: Navrhnúť a implementovať grafický editor pre MediaWiki určený pre žiakov umožňujúci kolaboratívne kreslenie a úpravu obrázkov. Vytvorený editor integrovať s wiki.matfyz.sk.

Vedúci: doc. RNDr. Zuzana Kubincová, PhD.
Konzultant: Mgr. Ján Kľuka, PhD.
Katedra: FMFI.KZVI - Katedra základov a vyučovania informatiky
Vedúci katedry: doc. RNDr. Zuzana Kubincová, PhD.

Dátum zadania: 06.10.2016

Dátum schválenia: 13.10.2016 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne vyhlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením doc. RNDr. Zuzany Kubincovej, PhD., s použitím zdrojov uvedených v zozname použitej literatúry.

.....

Bratislava, 2018 Bc. Martin Krasňan

Poďakovanie

...podakovanie...

Abstrakt

KRASŇAN, Martin: *Klaboratívny grafický editor pre MediaWiki* [Diplomová práca]. - Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. - Školiteľ: doc. RNDr. Zuzana Kubincová, PhD.. Bratislava: Bratislava, 2018. ?? strán.

Text abstraktu...

Kľúčové slová: klucove, slova, sk, ...

Abstract

KRASŇAN, Martin: *Collaborative graphics editor for MediaWiki* [Master thesis]. - Comenius University in Bratislava. Faculty of Mathematics, Physics and Informatics; Department of Applied Informatics. - Supervisor: doc. RNDr. Zuzana Kubincová, PhD.. Bratislava, 2018. ?? pages.

Text of abstract

Keywords: keywords, en, ...

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

...

1

Motivácia

...

2

Prehľad problematiky

2.1 Základné pojmy

V nasledujúcej kapitole sa budeme venovať popisu základných pojmov, vysvetlíme si princípy práce s 2D počítačovou grafikou a priblížime si jednotlivé technológie, pomocou ktorých budeme implementovať zadanie tejto diplomovej práce.

2.1.1 Počítačová grafika

Pojem počítačová grafika je z technického hľadiska odbor informatiky, ktorý sa zaoberá vykreslením a spracovaním obrazových informácií, za pomoci výpočtovej techniky. Tieto obrazové informácie môžu byť získavané napríklad pomocou digitálneho fotoaparátu, kde svetlo prechádzajúce sústavou šošoviek dopadá na optický snímač. Ten pozostáva z miliónov buniek citlivých na svetlo. Tieto bunky prevádzajú farbu a intenzitu dopadajúceho svetelného lúča na elektrický signál, ktorý je následne uložený do digitálnej formy pomocou zodpovedajúceho čísla. Výstupom takto zachytených obrazových informácií je v podstate veľmi dlhý text, pozostávajúci z čísel popisujúcich jednotlivé body zachyteného obrazu. Počet týchto bodov je závislý od počtu svetlo-citlivých buniek obrazového snímača fotoaparátu, tzv. pixelov. Zachytený obraz je možné ďalej spracovávať počítačovou technikou, za pomoci grafických softvérov.

Tento druh počítačovej grafiky zaraďujeme medzi takzvanú rastrovú grafiku. Ako sme si už popísali, grafická informácia je tu uložená pre každý bod vykresľovaného obrazu a tým pádom je výsledná kvalita limitovaná počtom týchto bodov.

Ďalší druh počítačovej grafiky je vektorová grafika. V tomto prípade sú obrazové informácie ukladané do jednotlivých grafických objektov. Ich základom sú body, pozostávajúce zo súradníc. Pri plošnom zobrazení sú to výška a šírka. Pomocou bodov definujeme jednotlivé geometrické objekty ako priamku (vektor), krivku, štvorec, polygón, elipsu a iné. Môžeme im taktiež definovať rôzne atribúty, na základe ktorých vieme určiť konkrétne vlastnosti ako farba výplne, šírka a farba obtiahnutia a ďalšie.

Vyššie opísané dva druhy počítačovej grafiky zaraďujeme medzi plošnú (2D) počítačovú grafiku. Ďalším spôsobom zobrazenia grafickej informácie je priestorové (3D) zobrazenie.

Toto zobrazenie je definované podobne ako pri vektorovej grafike, ale je rozšírené o tretí rozmer, ktorým je hĺbka. Vďaka tomu vieme definovať objekty zobrazené v priestore ako guľa, kváder, ihlan a iné. Trojrozmerná počítačová grafika sa primárne využíva v hernom, strojárenskom priemysle alebo na tvorbu filmových efektov. Využíva sa na vytvorenie virtuálneho modelu, ktorý sa snaží čo najdôvernejšie priblížiť objektu v reálnom svete.

Priestor, v ktorom zobrazujeme objekty 3D počítačovej grafiky sa nazýva scéna. Je to virtuálna reprezentácia sveta, definovaná tromi rozmermi, ktoré tvoria globálny súradnicový systém. Objekty zobrazované v scéne sú zdroj svetla, pozícia kamery a samotné 3D modely. 3D model je objekt zložený zo vzájomne susediacich polygónov. Tieto polygóny sú tvorené vrcholmi, hranami medzi nimi a tvoria tak sieť výsledného objektu. Jednotlivé vrcholy sú v rámci objektu, do ktorého patria, definované v lokálnom súradnicovom systéme. Pre prevod súradníc vrcholov z lokálneho súradnicového systému do globálneho súradnicového systému scény využívame transformačnú maticu, ktorou sú násobené všetky vrcholy daného modelu. Tvorí ju súčin matice posunutia, rotácie a skálovania.

2.1.2 Kolaboratívna práca vo webových aplikáciách

Kolaboratívna práca [LCL04] je proces, pri ktorom je obsah dokumentu vytváraný skupinou niekoľkých používateľov. Rozdeľujeme ju na synchrónnu a asynchrónnu.

Pri synchrónnej kolaborácií pracujú používatelia na zadanej úlohe súčasne, pričom každá akcia vykonaná jedným používateľom sa v tom istom čase prejaví všetkým ostatným spolupracovníkom. Výhodami synchrónnej kolaborácie sú:

- **Rýchlosť** - vďaka akciám prejavovaným v reálnom čase je pre účastníkov umožnená ich okamžitá reakcia
- **Kvalita** - práca viacerých používateľov na jednej téme spoločne prináša širší uhol pohľadu a tým pádom aj lepšiu informovanosť

Asynchrónne vytváranie obsahu môže byť organizované čiastkovými podúlohami, ktoré jednotliví používatelia riešia samostatne a po dokončení sa tieto podúlohy spoja do jedného celku. Najčastejšie využitie je pri tvorbe textových dokumentov alebo programovaní zdrojových súborov aplikácií. Tento prístup sa uplatňuje najmä vo verziovacích systémoch, akými sú napríklad GIT alebo SVN. Taktiež sa využíva v systéme MediaWiki.

2.2 Použité technológie

2.2.1 HTML

HTML (**H**yper **T**ext **M**arkup **L**anguage) je značkovací jazyk určený na tvorbu statických webových aplikácií. Využíva štruktúru XML jazyka, kde sú informácie zaobalené do tagov. Internetové prehliadače na základe týchto tagov určujú vzhľad a formát akým je informácia reprezentovaná používateľovi.

2.2.2 JavaScript

JavaScript je multiplatformový objektovo orientovaný skriptovací jazyk, určený predovšetkým na tvorbu interaktívnych webových aplikácií. Najčastejšie je používaný na strane klienta, čo znamená že funkcionalita je najskôr odoslaná do webového prehliadača klienta, kde je následne vykonaná. V moderných webových aplikáciách sa jazyk JavaScript využíva aj na serverovej strane, kde sa funkcionalita spúšťa v runtime prostredí Node a klientovi sú odoslané iba výsledky spracovania.

2.2.3 PHP

PHP je skriptovací jazyk navrhnutý na tvorbu dynamických webových stránok. Funkcionalita je vyhodnocovaná na strane servera pomocou PHP runtime a klientovi sú odosielané iba výsledky spracovania. Môžeme ho nasadiť na väčšinu webových serverov pracujúcich na operačných systémoch ako Unix, Windows alebo MacOS. PHP najčastejšie kooperuje s databázovými systémami typu MySQL, PostgreSQL alebo Microsoft SQL, v ktorých sú uchovávané dáta využívané aplikáciou.

2.2.4 MediaWiki

MediaWiki je voľne šíriteľný CMS systém určený na kolaboratívnu tvorbu stránok. Je primárne naprogramovaný v jazyku PHP a dáta sú uchovávané v relačnej databáze typu MySQL. Využívajú ho viaceré veľké aplikácie ako Wikipedia, Wikitionary, Wikibooks a množstvo ďalších. Wiki sa stali popu-

lárnym nástrojom pre kolaboráciu na internete. Primárnym účelom wiki stránok je zhromažďovať, udržiavať a zdieľať informácie jednoduchým spôsobom [KVV06]. Na tvorbu obsahu týchto informácií sa používa špeciálna syntax určená pre wiki systémy s názvom wiki-text. Ten primárne pozostáva z obyčajného textu s niekoľkými špeciálnymi značkovacími elementami. Napríklad odkaz na inú stránku v rámci MediaWiki systému zapíšeme tak, že názov danej stránky zaobalíme do dvojitých hranatých zátvoriek `[[Názov stránky]]`. Podobný zápis sa využíva aj na vkladanie súborových príloh ako napríklad obrázok, kde je syntax zápisu nasledovná: `[[File:NazovSuboru.jpg]]`. Ďaľšie informácie ohľadom spôsobu syntaxe wiki-textu je možné nájsť v dokumentácií MediaWiki systému [Med17a]. Obsah stránky je možné zapisovať priamo pomocou tohoto značkovacieho jazyka, alebo pomocou editora formátovaného textu (rich-text editora).

Hlavnými výhodami MediaWiki systémov sú nasledujúce vlastnosti:

- *Kolaboratívny aspekt:* všetky informácie sú okamžite dostupné pre každého a každá zmena v článku je publikovaná a viditeľná.
- *Jednoduchosť vytvárania dokumentov a prepojení medzi nimi.*
- *Otvorenosť pre čítanie a úpravu:* informácie sú dostupné pre čitateľov, rovnako ako editorov.
- *Otvorenosť pre experimenty:* je dosiahnuteľná vďaka histórií modifikácií pre všetky informácie.
- *Rozčlenenie informácií:* vďaka možnosti jednoduchého odkazovania na iné články, je možné členiť informácie do samostatných stránok, pre každú tému, výrok alebo samotné slovo. Tieto informácie sú následne

jednoducho dostupné v rôznych kontextoch, čo zabezpečuje ich znovapoužiteľnosť.

- *Refaktorizácia:* jednoduchosť vytvárania dokumentov a verziovanie podporuje ich následnú refaktorizáciu. Ak sa dokument stáva príliš veľkým, je možné ho rozdeliť do menších častí a tým ho značne sprehľadniť.

2.2.5 WebSocket

WebSocket je sieťový protokol definujúci spôsob komunikácie medzi serverom a klientom vo webovom prostredí. Zachováva vlastnosti HTTP protokolu pre webové aplikácie (URL adresy, HTTP zabezpečenie, jednoduchší dátový model založený na správach a vstavanú podporu pre text). *WebSocket*, tak ako aj TCP protokol, je asynchrónny a môže byť použitý ako transportná vrstva iných protokolov. Umožňuje komunikáciu v reálnom čase, vďaka čomu sa hodí na tvorbu četovacích protokolov alebo odosielanie serverových notifikácií klientom. Pripojenie je realizované vždy zo strany klienta na server pomocou HTTP dopytu nazývaného *handshake*[WSM13]. Následne prebieha komunikácia obojsmerne, pokiaľ je komunikačný kanál otvorený.

2.3 Použité knižnice

Vyššie opísané technológie sú síce kľúčovými pre tvorbu interaktívnych webových aplikácií, avšak existujú viaceré knižnice, vďaka ktorým je možné s týmito technológiami pracovať jednoduchšie a efektívnejšie. V nasledujúcej časti si stručne opíšeme tie, ktoré sú pre túto diplomovú prácu najpodstat-

nejšie.

2.3.1 JavaScriptová knižnica Less.js

LESS je dynamický štýlovací jazyk, ktorý môže byť skompilovaný do CSS kaskádových štýlov, upravujúcich základný vzhľad komponentov webových aplikácií. Narozdiel od klasických kaskádových štýlov, tento jazyk umožňuje definovanie:

- premenných - špecifikovanie často používaných hodnôt na jednom mieste a následné referencovanie tejto hodnoty kdekoľvek v zdrojových súboroch
- mixinov - umožňujú vložiť všetky vlastnosti jednej triedy do inej
- vnorených pravidiel - namiesto vytvárania selektorov s dlhým názovm špecifikujúcich dedičnosť používame vnorenie selektorov do iných, vďaka čomu je zápis prehľadnejší a kratší
- funkcií a výpočtov - hodnoty závislé od proporcií iných elementov je možné vypočítavať za pomoci základných aritmetických operácií sčítania, odčítania, násobenia alebo delenia

Zdrojové súbory v jazyku Less sú kompilované pomocou viacerých možných spôsobov. Napríklad priamo v prehliadači používateľa, v prostredí Node, PHP, .Net a ďalších. Výsledkom kompilácie je vytvorenie súboru s kaskádovými štýlmi vo formáte css.

2.3.2 JavaScriptová knižnica Node.js

NodeJS je open-source multiplatformové JavaScriptové runtime prostredie, ktoré vykonáva funkcionalitu naprogramovanú jazykom JavaScript na strane servera. Toto prostredie beží na V8 JavaScript engine navrhnutom spoločnosťou Google. Je určené na tvorbu vysoko škálovateľných internetových aplikácií, s využitím asynchrónnych I/O operácií [TV10] pre minimalizáciu réžie a maximalizáciu výkonu.

To znamená, že hlavný proces aplikácie spracováva požiadavky postupne, pričom každá časovo zložitejšia podúloha je vykonávaná asynchrónne. Hlavné vlákno teda nečaká na vykonanie pomalých operácií, ale vykoná svoju funkcionalitu, následne spustí pomalú operáciu a v zápätí začne spracovávať ďalšiu požiadavku. Po vykonaní pomalej asynchrónnej operácie, NodeJS vykoná funkcionalitu, ktorá je reakciou na jej ukončenie a ďalej sa venuje iným úlohám. Je to teda jedno vlákno, ktoré sa rýchlo a na krátku dobu prepína medzi rôznymi úlohami.

Takýto spôsob spracovania dopytov má teda dve hlavné výhody:

- nízke nároky na pamäť servera
- nevzniká potreba komplexnosti z paralelného vykonávania

NodeJS obsahuje sadu API umožňujúcu vykonávanie serverových operácií do ktorej patria:

- HTTP - umožňuje spracovávať HTTP serverové dotazy
- I/O - práca so súbormi pomocou Streams a Buffers

- DNS/URL - pomocné API na prácu s DNS a URL
- Crypto - kryptografické operácie
- Processes - interakcia s operačným systémom
- Cluster - spúšťanie NodeJS v clustri, možnosť fungovania vo viacerých vláknach

2.3.3 JavaScriptový framework Angular.js

AngularJS je JavaScriptový framework, ktorý sa zameriava na rozšírenie HTML jazyka do čitateľnejšej a dynamickejšej podoby [JBM15]. Umožňuje pridávať do zdrojového HTML kódu vlastné tagy a atribúty, ktoré sú synchronizované s funkcionalitou napísanou v jazyku JavaScript. Tie sú vyhodnocované až po načítaní obsahu stránky do DOM, čo má nasledujúce výhody:

- bezproblémová integrácia do existujúcich aplikácií
- možnosť pracovať s Angular-om priamo v HTML dokumente bez nutnosti spúšťania webservera alebo kompilovania
- jednoduchá rozšíriteľnosť pomocou direktív, ktoré určujú ako sa majú jednotlivé elementy vykresľovať v internetovom prehliadači a akú funkcionalitu majú poskytovať

Dynamickosť aplikácií naprogramovaných pomocou knižnice AngularJS zabezpečuje taktiež mapovanie (angl.: dat-binding) dát na jednotlivé elementy HTML šablóny. To znamená, že pri zmene hodnôt modelu, sa tieto zmeny automaticky odzrkadľujú vo výstupnom vzhľade šablóny aplikácie. Okrem

toho je tento proces obojsmerný, takže je možné zmenou hodnôt v šablóne aplikácie taktiež ovplyvňovať samotný model. *AngularJS* sa tým pádom stará o synchronizáciu modelu a šablóny bez potreby programovania setter alebo getter funkcií.

Táto knižnica začleňuje základné princípy programovania pomocou návrhového vzoru **M**odel-**V**iew-**C**ontroller na tvorbu klientskej časti webových aplikácií.

Model

Model, v prípade aplikácie využívajúcej túto knižnicu, sú dáta s ktorými aplikácia pracuje. Sú to obyčajné JavaScriptové objekty. Medzi model môžeme taktiež zaradiť základný `$scope` objekt. Ten poskytuje jednoduché API navrhnuté na detekciu a odoslanie informácie o zmene svojho stavu. K funkciám a premenným tohoto objektu je možné pristupovať priamo z HTML šablón.

Controller

Controller je zodpovedný za prvotné nastavenie stavu aplikácie a následné rozširovanie `$scope` objektu o metódy na riadenie správania aplikácie.

View

View je HTML, ktoré je vyprodukované po tom, ako AngularJS rozparsoval a skompiloval pôvodnú šablónu, v ktorej boli definované jednotlivé mapovania, atribúty a elementy.

2.3.4 JavaScriptová knižnica Fabric.js

V jazyku HTML existuje element `canvas`, do ktorého je možné dynamicky vykresľovať grafické objekty vo webovej aplikácií pomocou jazyka JavaScript. Funkcionalita vstavaného API je žiaľ veľmi obmedzená [CGM+14]. Pokiaľ máme záujem o vykreslenie jednoduchých tvarov a následne s nimi nepotrebujeme nič viac robiť, tak funkcionalita API bude postačovať. Avšak, akonáhle je potrebná ďalšia interakcia s vykresleným objektom, prípadne vykreslenie zložitejšieho objektu, nastáva tu problém.

Fabric je nadstavba nad toto natívne API, poskytujúce jednoduchý avšak veľmi efektívny a výkonný model objektu. Stará sa o udržiavanie stavu `canvas`-u, prekresľovanie grafickej plochy a dovoľuje nám pracovať s objektami priamo.

Objekty

Pri vytváraní základných geometrických útvarov pomocou tejto knižnice sa vytvárajú JavaScriptové objekty definované pod `fabric` namespaceom. Každý z nich je dedený od základného `fabric` objektu typu `fabric.Object`. Konkrétne ide o tieto útvary a objekty, ktoré definujú ich vlastnosti a metódy:

- `fabric.Circle` *(Kruh)*
- `fabric.Ellipse` *(Elipsa)*
- `fabric.Line` *(Úsečka)*
- `fabric.Polygon` *(Polygón)*

- `fabric.Polyline` *(Krivka)*
- `fabric.Rect` *(Obdĺžnik)*
- `fabric.Triangle` *(Trojuholník)*
- `fabric.Path` *(Cesta)*
- `fabric.IText` *(Textové pole)*
- `fabric.Textbox` *(Blok textu)*

Každý z nich má definované nasledovné premenné:

- `left, top` - pozícia objektu v canvas elemente
- `width, height` - šírka a výška objektu
- `fill, opacity, stroke, strokeWidth` - fatba výplne, priehľadnosť, farba obtiahnutia a šírka obtiahnutia
- `scaleX, scaleY, angle` - skálovanie a uhol rotácie
- `flipX, flipY` - horizontálne a vertikálne prevrátenie
- `skewX, skewY` - horizontálne a vertikálne zošikmenie

V prípade, že potrebujeme zmeniť nejakú vlastnosť niektorého z objektov, využijeme metódu `set` s JSON parametrom premenných, ktoré chceme zmeniť (Zdrojový kód 2.1).

```
1 var rect = new fabric.Rect();
2 rect.set({ width: 10, height: 20, fill: '#f55', opacity:
     0.7 });
```

Zdrojový kód 2.1: Vytvorenie objektu typu obdĺžnik pomocou knižnice FabriJS a zmena jeho základných vlastností

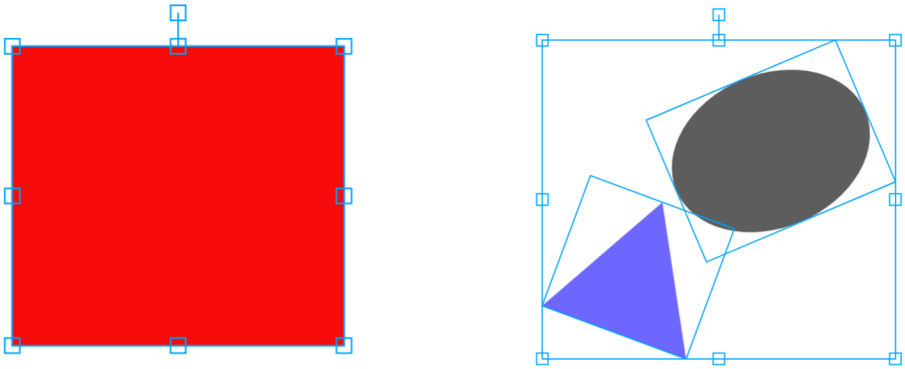
Canvas

Inicializácia Fabric knižnice pozostáva z vytvorenia `fabric.Canvas` objektu zaobaľujúceho grafickú plochu editora. Ten je zodpovedný za spravovanie všetkých fabric objektov konkrétneho canvs-u. Vstupom objektu je parameter s hodnotou `id` pre HTML element typu `<canvas>`. Výstupom je inštancia `fabric.Canvas` objektu. Do konštruktora však môžeme vložiť taktiež parameter typu JSON, pozostávajúci z konfiguračných nastavení pre vytváranú inštanciu (Zdrojový kód 2.2).

```
1  var canvas = new fabric.Canvas('c', {
2    backgroundColor: 'rgb(100,100,200)',
3    selectionColor: 'blue'
4    // ...
5  });
6
7  // alebo
8
9  var canvas = new fabric.Canvas('c');
10 canvas.setBackgroundColor('rgb(100,100,200)');
11 canvas.setSelectionColor('blue');
12 // ...
```

Zdrojový kód 2.2: Inicializácia Fabric canvas wrappera

Jednou z unikátnych vstavaných vlastností knižnice Fabric je interakčná



(a) Manipulácia s objektom (b) Manipulácia skupiny objektov

Obr. 2.1: Manipulácia s objektami pomocou vstupných zariadení s použitím knižnice Fabric

vrstva nad grafickou plochou canvas elementu, obsahujúceho objektový model. Objektový model existuje aby umožnil programový prístup a manipuláciu s objektami canvas-u. Z pohľadu používateľa je však možné manipulovať s objektami pomocou počítačovej myši, prípadne dotykov pri zariadeniach s dotykovou obrazovkou (Obrázok 2.1a). Ihneď po inicializácií pomocou `new fabric.Canvas('...')`, je možné vybrať objekt, posúvať ho ťahaním, škálovať, rotovať alebo zoskupiť viacero objektov do skupiny a manipulovať so všetkými súčasne (Obrázok 2.1b).

Event

Aplikácie a frameworky postavené na architektúre riadenej udalosťami poskytujú veľkú flexibilitu a výkonnosť. Knižnica Fabric nie je výnimkou a poskytuje rozšíriteľný systém udalostí (*event*-ov), začínajúc od jednoduchých (spúšťaných akciami počítačovej myši), až po zložité objektové udalosti. Dovoľujú nám odchytávať rôzne akcie odohrávajúce sa v *canvas* grafickej ploche.

API na prácu s udalosťami je veľmi jednoduché na obsluhu. Je podobné štandardom implementovaným v jQuery, Underscore.js alebo iných populárnych JavaScriptových knižniciach. Na inicializovanie event listener-u slúži metóda `on` (Zdrojový kód 2.3) a na prípadné odstránenie použijeme metódu `off`.

```
1 var canvas = new fabric.Canvas('...');
2 canvas.on('mouse:down', function(options) {
3   console.log('Klikli ste na pozíciu: ', options.e.clientX
      , options.e.clientY);
4   if (options.target) {
5     console.log('Klikli ste na objekt: ',
        options.target.type);
6   }
7 });
```

Zdrojový kód 2.3: Ukážka programovej implementácie na prácu s eventami

Ako je možné vidieť z ukážky programového kódu vyššie, metóda `on` obsahuje dva parametre. Prvým je názov udalosti ktorú chceme odchytávať, tým druhým je metóda spracovávajúca danú udalosť nazývaná *event handler*. Tá prijíma objekt *options*. V tomto objekte sa nachádzajú dve premenné:

- `e` - originálny JavaScriptový event
- `target` - objekt na ktorý bolo kliknuté alebo hodnota `null`

Existuje viacero kategórií udalostí vyvolávaných nasledovnými akciami:

- akcie počítačovej myši - `mouse:up`, `mouse:down`, `mouse:move`, `mouse:dblclick`, `mouse:wheel`, `mouse:over`, `mouse:out`

- akcie zmeny označenia objektu - `before:selection:cleared`, `selection:cleared`, `selection:created`, `selection:updated`
- akcie objektu - `object:added`, `object:removed`, `object:modified`, `object:moving`, `object:scaling`, `object:rotating`, `object:skewing`
- akcie grafickej plochy - `after:render`

Udalosti sú vyvolávané automaticky v rámci programového kódu knižnice Fabric. V prípade že potrebujeme manuálne vyvolať niektorú z udalostí, docielime to pomocou metódy `canvas.trigger(...)` (Zdrojový kód 2.4)

```
1 canvas.trigger('object:modified', {target: object});
```

Zdrojový kód 2.4: Manuálne vyvolanie udalosti v knižnici Fabric

2.3.5 JavaScriptová knižnica Socket.io

Socket.IO je JavaScriptová knižnica vytvorená pre realtime webové aplikácie. Socket.IO umožňuje real-time obojstrannú komunikáciu medzi webovým klientom a servermi. Socket.IO má dve časti:

- *klientská strana* bežiaca vo webovom prehliadači používateľa
- *serverová strana* pre Node.js aplikáciu riadiacu správanie všetkých klientov

Obe časti majú takmer identické API. Podobne ako node.js je socket.io "*event-driven*", teda funkcionalita je vyvolávaná pomocou definovaných funkcií pri určitých akciach používateľa alebo systému.

2.4 Rozšírenia MediaWiki (Extensions)

MediaWiki, tak ako aj množstvo ďalších systémov na správu obsahu stránok, poskytuje spôsob, pomocou ktorého je možné zmeniť správanie aj vzhľad MediaWiki aplikácie. V závislosti od potreby existuje niekoľko kategórií rozšírení (angl.: *extension*):

- *Parser extension*: rozšírenie wiki-textu o vlastné značky a príkazy.
- *Special page extension*: pridanie funkcionality na reportovanie a administratívu.
- *User interface extension*: zmena vzhľadu MediaWiki aplikácie.
- *Authentication and Authorisation extension*: rozšírenie zabezpečenia a overenia používateľov pomocou vlastných mechanizmov.

Môžu byť do systému inštalované iba s administrátorským prístupom do súborového systému na serveri. Inštalácia pozostáva zo skopírovania zdrojových súborov do adresára `$IP/extensions/nazov_rozsirenia/`. Následne ho je možné inicializovať v koreňovom adresári MediaWiki inštalácie, v súbore `LocalSettings.php`, pomocou globálnej funkcie `wfLoadExtension('nazov_rozsirenia')`.

Okrem existujúcich oficiálnych rozšírení spravovaných komunitou vývojárov, je možné nainštalovať aj rozšírenia tretích strán. Môže tu však nastať viacero problémov. Dané rozšírenie môže spôsobovať chyby, prípadne úplnú nefunkčnosť aplikácie, nemusí byť viac podporované vývojármi alebo môže byť nekompatibilné s aktuálnou verziou MediaWiki inštalácie. Ak rozšírenie

ovplyvňuje štruktúru databázy, je dobré pred inštaláciou previesť proces jej kompletného zálohovania. Pri vývoji vlastného rozšírenia je preto potrebné myslieť na tieto faktory a snažiť sa dodržiavať konvencie programovania pre systém MediaWiki [Med17b].

Pre správne fungovanie rozšírenia je potrebné zabezpečiť niekoľko kľúčových vlastností, pozostávajúcich z týchto krokov:

- Zaregistrovať akýkoľvek media handler, parsovaciu funckiu, špeciálne stránky, vlastné XML značky a premenné použité vašim rozšírením.
- Definovať a zvalidovať každú konfiguračnú premennú.
- Pripraviť triedy použité vo vašom rozšírení pre autonačítanie.
- Rozhodnúť ktoré časti inštalačného nastavenia majú byť vykonané okamžite a ktorá majú čakať pokiaľ sa inicializuje jadro MediaWiki.
- Definovať dodatočné hooky potrebné pre rozšírenie.
- Vytvoriť / skontroluje všetky nové databázové tabuľky nutné pre rozšírenie.
- Nastaviť lokalizáciu rozšírenia.

2.4.1 ResourceLoader

Vytváraním rozšírení existujúcich systémov vzniká množstvo súborov, odosielaných zo servera na jednotlivých klientov. Aby sa predišlo dlhému načítaniu stránky, je potrebné nejakým spôsobom riadiť, kedy a komu sa majú odoslať zdrojové súbory. Systém MediaWiki na to využíva takzvaný *delivery system*

označovaný taktiež ako *ResourceLoader*. Jeho úlohou je odosielať zdrojové súbory, ktoré klientská časť aplikácie aktuálne potrebuje a ktoré podporuje internetový prehliadač klienta. Využíva pri tom nasledujúce procesy:

- *Minifikácia a spájanie* zdrojových súborov optimalizuje ich veľkosť a tým pádom aj redukuje čas ich načítania.
- *Dávkové načítavanie* viacerých modulov (rozšírení) súčasne s použitím predchádzajúceho procesu minifikácie a spájania redukuje počet dopytov na server.
- *Data URI embedovanie* zdrojových obrázkov v kaskádových štýloch taktiež redukuje počet potrebných dopytov na server a tým aj jeho čas odozvy.

3

Predchádzajúce riešenia

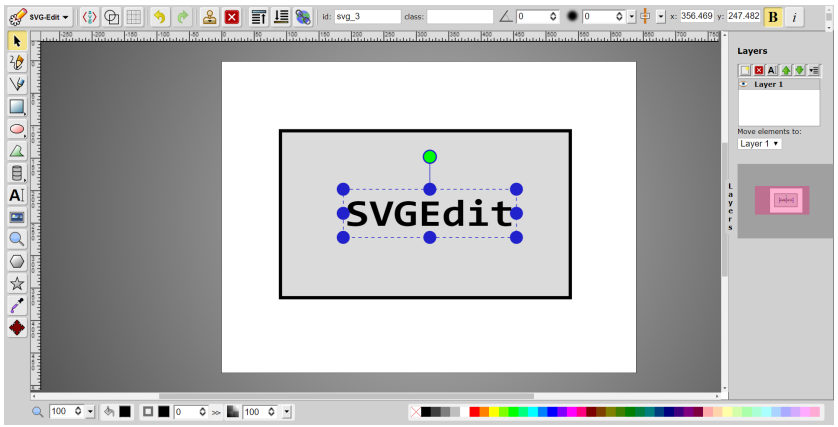
V nasledujúcej kapitole opíšeme existujúce riešenia, zameriame sa na ich výhody a prípadné nevýhody.

3.1 SVGEdit

SVGEdit je jediné oficiálne rozšírenie MediaWiki existujúce v dobe písania tejto diplomovej práce. Aktuálne ja v stave experimental, čo naznačuje, že

nie všetka funkcionalita bude bezproblémová. Poskytuje možnosti vytvárania a editovania súborov vektorovej grafiky vo formáte svg. Využíva open source SVG-edit widget, ktorý poskytuje štandardnú funkcionalitu vektorového editora, s relatívne vysokou kvalitou spracovania.

Nevýhodou tohoto riešenia je, že samotný editor je vložený formou iframe widgetu zo stránok vývojárov tohoto widgetu. To spôsobuje nemožnosť upraviť vzhľad alebo funkcionalitu editora. Prepojenie s MediaWiki systémom je zabezpečené pomocou asynchrónnych ajax volaní. Editor taktiež nepodporuje kolaboratívnu úpravu jedného súboru viacerými používateľmi. Ďalšou nevýhodou je jeho zložité ovládanie nehodiace sa pre cieľovú skupinu finálnej aplikácie, ktorou sú študenti základných a stredných škôl.



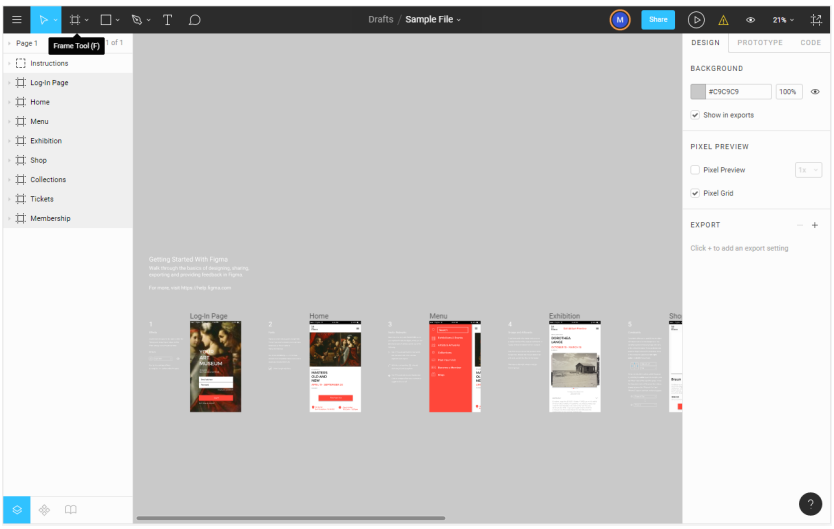
Obr. 3.1: Prostredie editora SVG-edit

3.2 Figma

Figma je profesionálny grafický editor s možnosťou kolaboratívnej práce viacerých používateľov. Je určený predovšetkým pre ľudí zaoberajúcich sa grafickým návrhom mobilných, webových alebo desktopových aplikácií. Jeho

hlavnou výhodou je intuitívne ovládanie, real-time komunikácia medzi spolupracujúcimi používateľmi a bohatá funkcionalita.

Keďže sa však jedná o samostatnú webovú aplikáciu, nie je možné ju implementovať do MediaWiki systému.



Obr. 3.2: Prostredie kolaboratívneho editora Figma

3.3 Draw.IO

Draw.IO je online webová aplikácia slúžiaca na kolaboratívnu tvorbu vektorovej grafiky. Využitie tejto aplikácie je možné nájsť najmä pri tvorbe:

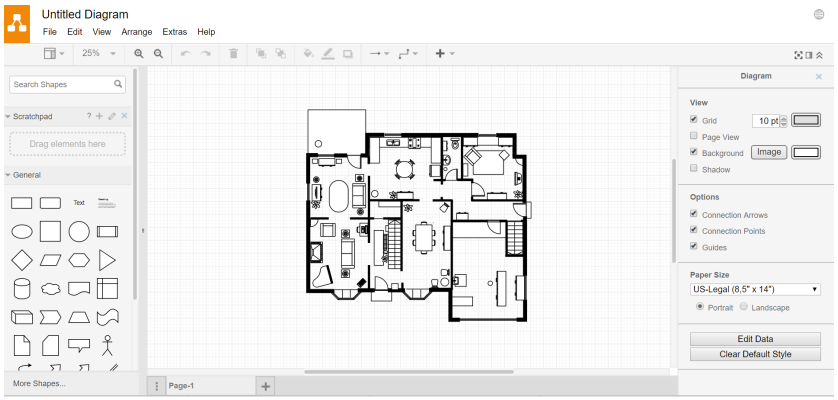
- *Diagramov* na lepšie pochopenie funkčnosti aplikácie, postupov vývoja alebo dátového modelu. Podporuje tvorbu *UML*, *BPMN*, *stromových*, *sieťových*, *sekvenčných*, *elektrotechnických* a množstva ďalších diagramov.
- *Obrysových (wireframe) modelov* na plánovanie rozloženia elementov

pri návrhu mobilných, počítačových alebo webových aplikácií.

- *Vývojových (mockup) modelov* na rýchle prototypovanie vzhľadu aplikácie.
- *Grafov* - Ganttov graf používaný pri plánovaní projektov, zobrazujúci časovú závislosť jednotlivých podúloh.
- *Architektonických plánov* podlaží budov.
- *Infografiky* na lepšiu zapamätateľnosť informácií.

Umožňuje bezplatné prepojenie s viacerými cloudovými riešeniami ako Google Drive, OneDrive alebo možnosť integrovať do vývojových kolaboratívnych softvérov Confluence a Jira. Aplikáciu je možné používať aj pomocou desktopových aplikácií na všetkých bežných operačných systémoch (Windows, macOS, Linux, Chrome OS). Ďalšími výhodami sú jednoduché a intuitívne ovládanie, dobrá skálovateľnosť pre projekty s veľkým množstvom elementov, možnosť bezplatného používania a voľná prístupnosť zdrojových súborov aplikácie pre vývojárov.

Nevýhodou je že aplikáciu nieje možné prepojiť so systémom MediaWiki a nepodporuje voľné kreslenie objektov. Tie môžu byť importované formou obrázka.



Obr. 3.3: Prostredie kolaboratívneho editora Draw.IO

4

Návrh modelu

V tejto kapitole si opíšeme cieľ diplomovej práce, zameriame sa na návrh modelu všetkých častí aplikácie a definujeme si požiadavky na ich funkcionalitu.

Riešenie diplomovej práce bude pozostávať z dvoch hlavných častí. Prvou časťou bude navrhnutie synchronizačného servera, ktorý bude zabezpečovať výmenu informácií medzi klientmi. V druhej časti navrhneme samotný grafický editor obrázkov. Vysvetlíme si spôsob integrácie editora do systému MediaWiki a popíšeme si jeho požadované vlastnosti.

4.1 Cieľ práce

- Navrhnúť prostredie grafického editora
- Implementovať grafický editor do prostredia MediaWiki pomocou rozšírenia
- Navrhnúť a implementovať spôsob ukladania verzií výstupných súborov grafického editora
- Navrhnúť a implementovat komunikačný protokol na synchronizáciu grafického editora pre viacero nezávislých používateľov
- Integrovať rozšírenie s webovou stránkou fakulty http://wiki.matfyz.sk

4.2 Synchronizačný server

Synchronizačný server bude pozostávať z aplikácie naprogramovanej v jazyku JavaScript. Spúšťaný bude v runtime prostredí NodeJS. Primárnou úlohou bude zaradiť pripojeného používateľa do miestnosti zodpovedajúcej editovanému súboru, vďaka čomu zabezpečíme synchronizáciu údajov medzi všetkými používateľmi pracujúcimi s týmto súborom. Využijeme pri tom knižnicu Socket.IO, konkrétne jej serverovú verziu, ktorá bude slúžiť ako hlavný komunikačný protokol postaveného na báze WebSocket protokolu. Aplikácia synchronizačného servera bude pozostávať z niekoľkých entít (objektov), ktoré si postupne popíšeme v ďalších častiach tejto diplomovej práce.

4.2.1 Popis entít synchronizačného servera

Entita používateľ

Základnou entitou bude objekt *Používateľ*, ktorý definuje vlastnosti pripojeného používateľa. Medzi informácie uchovávané v tomto objekte budú patriť *meno*, jeho zodpovedajúca *farba* kurzora, pod ktorou bude rýchlo rozpoznateľný inými používateľmi a *jedinečný identifikátor* získaný po pripojení na server.

Entita správa

Editor poskytne možnosť textovej komunikácie medzi používateľmi. Kvôli tomu je potrebné navrhnúť objekt reprezentujúci takúto správu. Medzi jeho vlastnosti bude patriť informácia, kto správu odoslal, pre koho je určená, čas kedy bola odoslaná a taktiež textový obsah správy. Pre potreby lepšej informovanosti o stave pripojených používateľov je možné odosielať takzvanú systémovú správu, určenú pre všetkých používateľov.

Entita objekt

Informácie o objektoch grafickej plochy budú synchronizované pomocou JavaScriptového objektu typu *Object*. Je to objekt dynamickej štruktúry s niekoľkými základnými vlastnosťami. Tými sú *jedinečný identifikátor objektu*, informácia o *stave uzamknutia* a v prípade uzamknutého objektu taktiež informácia o *používateľovi, ktorý daný objekt uzamkol*. Ďalšie informácie o objekte sú závislé od jeho dátového typu generovaného na klientskej strane

grafického editora.

Entita miestnosť

Každý používateľ bude po pripojení na synchronizačný server automaticky zaradený do miestnosti zodpovedajúcej editovanému súboru. Táto miestnosť bude reprezentovaná objektom Room s vlastnosťami popisujúcimi jej aktuálny stav. Bude sa tu nachádzať zoznam pripojených používateľov, zoznam odoslaných správ, zoznam grafických objektov editora, informáciu o formáte editovaného súboru a vlastnostiach grafickej plochy.

Entita správca miestností

Aby sa predišlo zbytočnému udržiavaniu stavov miestností v ktorých sa nenachádzajú žiadni používatelia, musíme navrhnúť entitu, ktorá bude riadiť dynamické vytváranie a odstraňovanie miestností. Pre tento účel vytvoríme objekt RoomManager, ktorý bude obsahovať zoznam všetkých miestností a funkcie na vytvorenie miestnosti pri pripojení prvého používateľa a jej vymazanie v prípade že sa z nej odpojil posledný používateľ.

4.2.2 Funkčné požiadavky synchronizačného servera

- **Pripojenie používateľa** - vytvorí sa entita typu `User` a nastavia sa mu vlastnosti na základe zaslaného dopytu.
- **Overenie pripojeného používateľa** - po pripojení nového používateľa sa overuje správnosť bezpečnostného tokenu. Ak bezpečnostného

token nie je správny, používateľ nebude mať umožnené pripojenie do zvolenej miestnosti.

- **Vytvorenie miestnosti** - v prípade, ak sa pripojený používateľ má zaradiť do miestnosti ktorá aktuálne neexistuje, takáto miestnosť je vytvorená a následne je do nej používateľ zaradený. V prípade že daná miestnosť už vytvorená je, používateľ sa do nej zaradí automaticky a táto informácia sa odošle všetkým ostatným používateľom nachádzajúcim sa v miestnosti. S pripojením prvého používateľa je potrebné načítať informácie o editovanom obrazovom súbore. Docielime to HTTP dopytom na rozhranie API (**A**plication **P**rograming *Interface*). Návratové hodnoty dopytu v prípade existujúceho súboru rozdelíme do príslušných vlastností aktuálnej miestnosti a následne ich odošleme formou odpovede pripájanému používateľovi.

- **Synchronizácia editora** - po pripojení a zaradení používateľa do miestnosti zodpovedajúcej editovanému súboru, bude server schopný prijímať synchronizačné správy o zmenách vykonaných používateľom v prostredí editora.

  Pri *vytvorení nového grafického objektu* editora sa odosiela požiadavka s informáciami o tomto objekte na synchronizačný server. Ten požiadavku spracuje, pridá objekt do zoznamu objektov na základe jeho jedinečného identifikátora a odošle informáciu o novom objekte všetkým ostatným používateľom pripojeným do miestnosti.

  S *modifikáciou existujúceho grafického objektu* editora bude taktiež odoslaný dopyt o tejto akcii synchronizačnému serveru. Ten ho spracuje, zmení zodpovedajúci objekt na základe prijatých informácií a odošle správu o zmene objektu spolu so zmenenými údajmi všetkým ostat-

ným používateľom.

Ak je *objekt v grafickom editore odstránený*, táto akcia bude taktiež odoslaná na synchronizačný server, spolu s jedinečným identifikátorom objektu. Ten ho spracuje, odstráni zodpovedajúci objekt zo zoznamu a následne odošle ostatným používateľom informáciu o jeho vymazaní.

Aby sa predišlo konfliktom pri zmenách vlastností objektov, je potrebné navrhnúť proces ich automatického uzamykania. To zabezpečíme tak, že pri zvolení aktívneho objektu v grafickom editore bude odoslaná požiadavka na jeho uzamknutie daným používateľom na server. Táto akcia je znova synchronizovaná so zvyšnými editormi používateľov. V prípade že objekt už uzamknutý bol, odošle sa odpoveď o zamietnutí uzamknutia objektu.

- **Odpojenie používateľa** - pri tejto akcií je potrebné vykonať niekoľko úkonov. Najskôr musia byť odomknuté všetky objekty, ktoré boli uzamknuté odpájaným používateľom a následne sa odstráni z miestnosti. V prípade, že miestnosť zostane prázdna, vymaže sa. Inak je odoslaná informácia o odpojení používateľa všetkým zvyšným pripojeným klientom.

- **Odstránenie miestnosti** - ako sme spomenuli vyššie, miestnosť je automaticky odstránená pri odchode posledného používateľa. Tento proces zabezpečí entita `RoomManager`.

4.3 Editor obrázkov

Grafický editor implementujeme formou MediaWiki rozšírenia, naprogramovaného primárne v jazyku JavaSrcipt, s použitím frameworku AngularJS (kap.: 2.3.3). Na komunikáciu so synchronizačným serverom použijeme JavaScriptovú knižnicu Socket.IO (kap.: 2.3.5), konkrétne jej klientskú verziu. Na vykresľovanie a udržiavanie stavu grafickej plochy použijeme knižnicu Fabric (kap.: 2.3.4).

4.3.1 Funkčné požiadavky grafického editora

- **Otvorenie grafického editora** a v prípade upravovania existujúceho súboru jeho automatické načítanie.
- **Výber nástroja** - grafický editor poskytne viacero pracovných nástrojov, na vytváranie rôznych typov grafických objektov. Tie môžu byť zvolené kliknutím na tlačidlo reprezentujúce daný nástroj.
- **Vytvorenie objektu** - po zvolení nástroja môžeme pridať do grafickej plochy nasledujúce objekty:
  - **Obdĺžnik**, **elipsu**, **trojuholník**, **úsečku** a **textové pole** pomocou kliknutia na grafickú plochu a ťahaním kurzora sa bude nastavovať ich veľkosť.
  - **Polygón**, ktorého vrcholy sa budú pridávať postupným klikaním na grafickú plochu.
  - **Obrázok** môže používateľ pridať zvolením nástroja na pridávanie obrázkov. Zobrazí sa ponuka na nahratie lokálneho súboru.

  - **Voľne kreslená cesta** vytvorená kliknutím a ťahaním kurzora, s možnosťou nastavenia šírky a typu cesty, farby a veľkosti rozmazania.

- **Výber aktívneho objektu / skupiny objektov** - používateľ vyberie aktívny objekt a následne sa zobrazí panel s možnosťou zmeny jeho vlastností.

- **Modifikácia vlastností aktívneho objektu** v závislosti od jeho typu. Základné vlastnosti ako výška, šírka, pozícia, uhol rotácie a skosenie možu byť nastavované pomocou transformácie obálky objektu. Tieto a ďalšie vlastnosti bude možné zmeniť aj pomocou zadávania hodnôt do zodpovedajúcich textových polí, prípadne tlačidlami a inými ovládacími prvkami.

- **Odstránenie objektu** z grafickej plochy.

- **Zmena hĺbky** ovplyvňujúca prekrývanie jednotlivých objektov v grafickej ploche.

- **Lokálne uzamknutie** zamedzí používateľovi možnosť pohybu, rotácie, a zmeny veľkosti objektu.

- **Zmena viditeľnosti** - každý grafický objekt môže byť skrytý z grafickej plochy editora a následne znova zobrazený.

- **Kopírovanie, vystrihnutie, vloženie a duplikovanie** objektu.

- **Priblíženie grafickej plochy** - používateľ môže proporcionálne zväčšiť alebo zmenšiť grafickú plochu zmenou hodnoty skálovania(zoom).

- **Zmena vlastností grafickej plochy** - okrem skálovania môže používateľ nastaviť aj jej rozmery, od čoho sa odvíja veľkosť výsledného súboru a tým aj jeho kvalita a taktiež farbu pozadia.
- **Zobrazenie zoznamu používateľov**
- **Režim celej obrazovky**, kedy sa skryjú ovládacie prvky systému MediaWiki a grafický editora sa zväčší na maximálnu dostupnú veľkosť okna internetového prehliadača používateľa.
- **Vycentrovanie grafickej plochy** na vertikálny a horizontálny stred okna grafického editora.
- **Chat** - používatelia si môžu vymieňať informácie formou krátkych textových správ.
- **Uloženie revízie súboru** v systéme MediaWiki.
- **Zatvorenie editora** a následné presmerovanie používateľa na stránku súboru.

4.3.2 Integrácia do sytému MediaWiki

Ako sme si opísali v časti 2.4, pre systém MediaWiki existuje viacero typov rozšírení. Pre naše riešenie sme si zvolili rozšírenie typu *special page extension*. Učinili sme tak z dôvodu, že ide o najlepšiu voľbu z hľadiska vytvárania vlastného používateľského rozhrania a jeho špecifickej funkcionality.

V systéme MediaWiki má každý multimediálny súbor svoju vlastnú podstránku. Aby sme používateľom umožnili otvoriť multimediálne súbory pomocou nášho editora, je potrebné integrovať tlačidlo na otvorenie súboru v

editore. To umiestnime medzi tlačidlá hornej navigačnej lišty, kde sa nachádzajú odkazy na úpravu alebo vytvorenie stránky, zobrazenie histórie zmien, zmazanie, presunutie a zamknutie súboru. Používatelia môžu taktiež vytvoriť nový súbor, za pomoci odkazu na špeciálnu stránku editora. Keďže používateľ vytvára nový súbor, musí byť vyzvaný na zadanie jeho názvu.

Systém MediaWiki umožňuje modifikovať položky hlavného menu a pridávať odkazy na rôzne stránky, vrátane špeciálnych stránok systému alebo rozšírení. Administrátor vďaka tomu môže jednoduchým spôsobom pridať odkaz na grafický editor do hlavného menu.

4.3.3 Návrh ovládacích prvkov editora

Väčšina prostredí grafických editorov, či už sú to vektorové, rastrové alebo 3D modelovacie editory využívajú rozdelenie ovládacích prvkov na 4 hlavné komponenty. Zoznam nástrojov zobrazený pri ľavom okraji okna editora, hlavné menu pri vrchnom a nastavenia nástrojov pri pravom okraji okna, s grafickou plochou uprostred týchto ovládacích komponentov. V našom prípade sme sa rozhodli použiť podobné rozloženie, kvôli zavedeným zvykom používateľov a tým pádom vyššej intuitívnosti pri používaní.

Rozloženie komponentov bude taktiež pozostávať zo 4. hlavných častí (Obrázok 4.1):

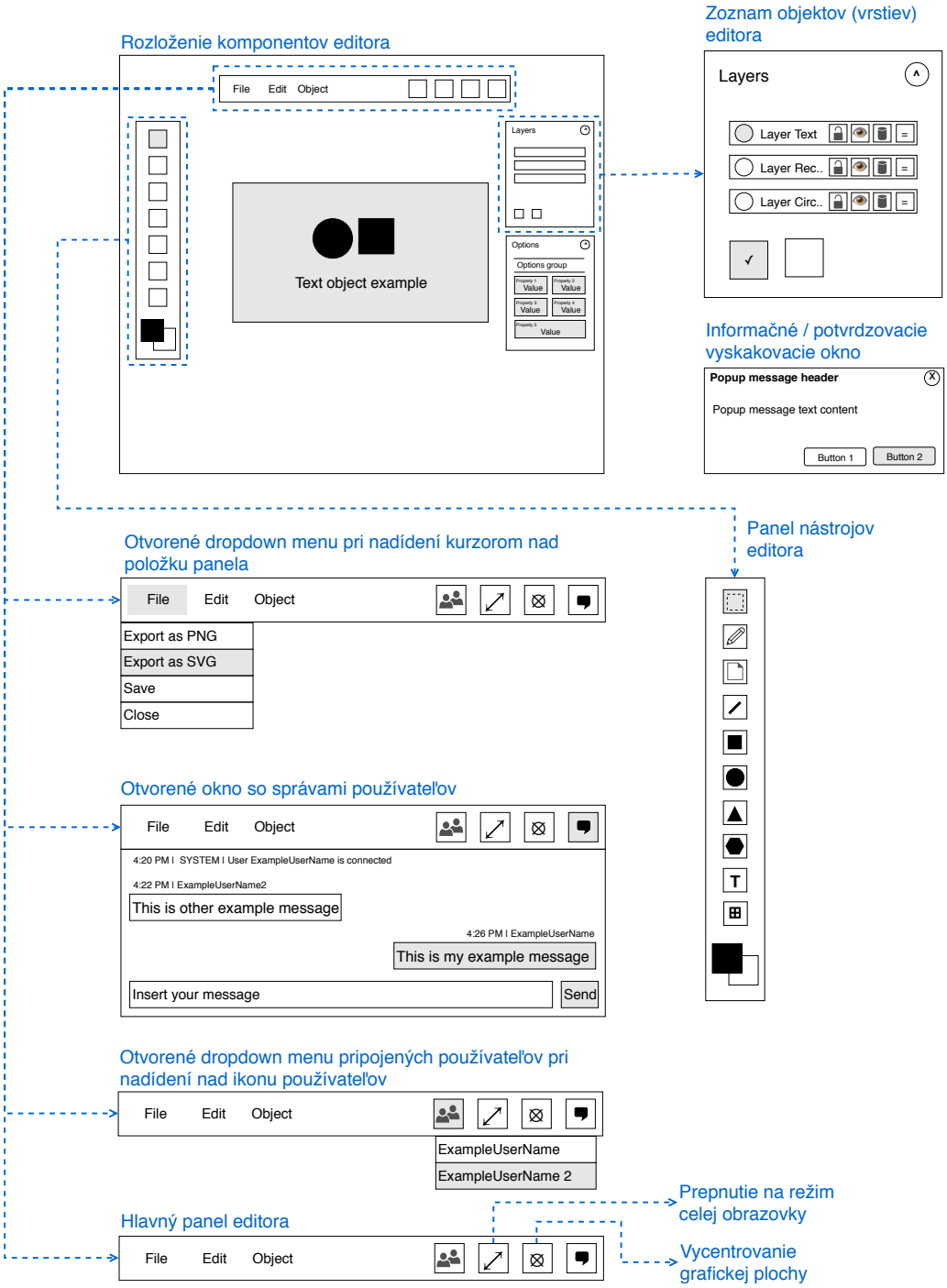
- **Panel nástrojov** s tlačidlami všetkých nástrojov a tlačidlami pre výber farby výplne a obtiahnutia objektu. V paneli sa taktiež nachádza tlačidlo na zapnutie prichytávania objektov k mriežke grafickej plochy.
- **Hlavný panel** obsahujúci tlačidlo na prepnutie zobrazenia do režimu

celej obrazovky, tlačidlo na vycentrovanie grafickej plochy a rozbaľovací zoznam správ chatu. Okrem týchto tlačidiel tu nájdeme aj 3 rozbaľovacie (dropdown) tlačidlá *Súbor*, *Upraviť* a *Objekt*. Rozbalením tlačidla *Súbor* sa zobrazí ponuka na exportovanie grafickej plochy do viacerých formátov, uloženie revízie a zatvorenie editora. Po rozbalením tlačidla *Upraviť* sa sprístupnia možnosti na kopírovanie, vystrihnutie, vloženie a duplikovanie objektu. Pod posledným tlačidlom *Objekt*, sa zobrazia možnosti operácií s aktívnym objektom na zmenu hĺbky objektu v grafickej ploche.

- **Panel nastavení** tvorený dvoma komponentmi. Komponent vrstiev zobrazujúci zoznam všetkých grafických objektov s možnosťou ich rýchleho uzamknutia, skrytia. Vrstvy môžu byť preusporiadané pomocou ťahania, čím sa bude meniť ich hĺbka v grafickej ploche editora. Druhý komponent obsahujúci nastavenia aktuálneho kontextu editora bude dynamicky zobrazovať položky na nastavenie vlastností aktívneho objektu, grafickej plochy alebo nastavenia voľného kreslenia.

- **Grafická plocha**, v ktorej budú vykresľované jednotlivé grafické objekty.

Okrem týchto panelov je potrebné informovať používateľa o niektorých vykonávaných akciách formou vyskakovacích (*popup*) správ, s možnosťou potvrdenia alebo prípadného zrušenia akcie. Obsah vyskakovacích okien je potrebné dynamicky vytvárať a modifikovať funkcionalitu potvrdzovacích tlačidiel.



Obr. 4.1: Návrh rozloženia komponentov používateľského prostredia editora

5

Implementácia

V tejto kapitole sa zameriame na riešenie popísanej problematiky, implementácii algoritmov a funkcií potrebných na správnu funkčnosť a dosiahnutie cieľov zadania diplomovej práce. Implementácia pozostáva z dvoch hlavných častí. Vytvorenie synchronizačného servera a klientskej aplikácie grafického editora formou rozšírenia systému MediaWiki. Popíšeme si ich dátové modely a ich funkcionalitu.

Pri implementácií využívame verziovací systém git. Obe hlavné časti sú umiest-

nené vo vlastných repozitároch na serveri GitHub.

5.1 Synchronizačný server

Synchronizačný server pozostáva z JavaScriptovej aplikácie naprogramovanej v jazyku JavaScript. Aplikácia je spúšťaná v runtime prostredí NodeJS servera, konkrétne vo verzii 7.6 a vyššej. Ten je nainštalovaný formou kontajnerového systému docker. Aplikácia pracuje pomocou inštancie HTTP servera, využívajúceho port s číslom `8080` a lokálnu adresou servera `0.0.0.0`.

Knižnice servera sú inštalované pomocou balíčkovacieho manažéra NPM (**N**ode **P**ackage **M**anager). Ich potrebné verzie sú zapísané v súbore `package.json`. Nachádzajú sa tu taktiež konfiguračné premenné prostredia:

- `api_endpoint` - adresa API koncového bodu systému MediaWiki, v ktorom je nainštalované naše rozšírenie. V našom prípade má hodnotu `"https://wiki.matfyz.sk/api.php"`.
- `api_token` - tajný token slúžiaci na overenie pripájaných klintov. Jeho hodnota sa musí zhodovať s hodnotou na strane rozšírenia.

5.1.1 Dátový model

Pre implementáciu serverovej časti aplikácie využívame dátovú štruktúru JavaScriptových objektov.

Názov	Typ	Popis
name	string	Meno používateľa prijaté v dopyte pri pripojení klienta na server
color	string	Automaticky generovaná farba pomocou funkcie `getRandomColor()` v HEX formáte `"#ffff00"`
verified	boolean	Príznak, či bol korektne overený bezpečnostný token
token	string	Privátna premenná s hodnotou bezpečnostného tokenu

Tabuľka 5.1: Zoznam triednych premenných objektu User

Názov	Popis
setToken(token)	Nastavenie bezpečnostného tokenu
getToken()	Vráti hodnotu bezpečnostného tokenu
verifyUser()	Overí hodnotu bezpečnostného tokenu a nastaví triednu premennú `verified`

Tabuľka 5.2: Zoznam metód objektu User

User

Trieda `User` reprezentuje používateľa pripojeného na synchronizačný server. Jej vlastnosti určujú triedne premenné a metódy opísané v tabuľkách 5.1 a 5.2.

Message

Objekt `Message` reprezentuje správu odoslané používateľmi pomocou chatu v editore. Popis vnútornej reprezentácie vlastností objektu môžeme vidieť v tabuľke 5.3

Názov	Typ	Popis
from	User	Odosielateľ správy
to	User	Prijímateľ správy
text	string	Textový obsah správy
type	string	Typ správy nastavovaný ak ide o systémovú správu
time	string	Dátum a čas odoslania správy

Tabuľka 5.3: Zoznam triednych premenných objektu Message

Názov	Typ	Popis
users	array<User>	Zoznam používateľov
messages	array<Message>	Zoznam správ
objects	array<object>	Zoznam objektov grafickej plochy
canvas	object	Dynamický objekt s nastaveniami grafickej plochy
format	string	Formát editovaného súboru (jpg, png, svg)
loaded	boolean	Príznak či je miestnosť plne vyinicializovaná
file	string	Názov editovaného súboru

Tabuľka 5.4: Zoznam triednych premenných objektu Room

Room

Trieda `Room` slúži na udržiavanie stavu miestnosti pripojených používateľov, spracovanie požiadaviek odosielaných zo strany editora, udržiavanie štruktúry objektov, odoslaných správ a funkcionality na načítanie informácií o súbore. Popis jej vnútornej štruktúry a poskytovanej funkcionality môžeme vidieť v tabuľkách 5.4 a 5.5.

Názov	**Popis**
loadFromWiki()	Načíťanie obsahu editora pri inicializácii miestnosti. Načítanie prebieha pomocou HTTP dopytu na koncový bod API systému MediaWiki, v ktorom je rozšírenie nainštalované.
createUser(user, socket)	Pripojí používateľa do zvolenej miestnosti na základe editovaného súboru.
removeUser(user, socket)	Odstáni používateľa zo zoznamu používateľov a odpojí ho z miestnosti.
createMessage(text, from, to, type)	Spracováva požiadavku odoslania novej správy pomocou chatu.
modifyCanvas(properties, socket)	Spracováva požiadavku na zmenu grafickej plochy editora.
createObject(object, socket)	Spracovanie požiadavky na vytvorenie objektu grafického editora.
modifyObject(object, socket)	Spracovanie požiadavky na zmenu objektu grafického editora.
removeObject(id, socket)	Spracovanie požiadavky na odstránenie objektu grafického editora.
setSelectable(id, selectable, user, socket)	Spracovanie požiadavky na zmenu uzamknutia objektu grafickej plochy.
deselectAll()	Metóda nastaví všetky objekty grafickej plochy na neuzamknuté.

Tabuľka 5.5: Zoznam metód objektu Room

Názov	Popis
isEmpty()	Metóda vracajúca true/false hodnotu v závislosti od toho, či je v danej miestnosti pripojený nejaký používateľ.
createRoom(name)	Metóda overí či existuje miestnosť s daným názvom a v prípade že neexistuje, vytvorí ju. Návratovou hodnotou je objekt typu `Room` zodpovedajúci danému názvu miestnosti.
getRoom(name)	Metóda vráti objekt typu `Room` v prípade že existuje miestnosť s daným názvom v zozname miestností.
removeRoom(name)	Metóda vymaže miestnosť so zadaným názvom zo zoznamu miestností.

Tabuľka 5.6: Zoznam metód objektu RoomManager

RoomManager

Trieda menežéra miestností `RoomManager` slúži na spravovanie existujúcich miestností, ich dynamické vytváranie a vymazávanie. Obsahuje triednu premennú `rooms`. Je to premenná typu asociatívne pole, obsahujúca objekty typu `Room`. Poskytuje funkcie na prácu s miestnosťami popísané v tabuľke 5.6.

5.1.2 Pripojenie klienta

Pri vytvorení inštancie Socket.IO objektu s názvom `io`, vkladáme do jeho konštruktora inštanciu vytvoreného HTTP servera. Po jej vytvorení je server pripravený prijímať správy s udalosťami takzvané *socket*-y. Príchod pripájacej správy s názvom udalosti `'connection'` je odchytený event-listenerom `io.on('connection', function(socket){...})`, v ktorom sa volá funkcia spracujúca túto udalosť. Informácie o pripájanom klientovi je do nej poslaná

parametrom socket. Pomocou tohoto parametra následne server odchytáva správy ďalšie správy, odosielané zo strany klienta. Parameter v sebe taktiež nesie objekt `socket.handshake`. V tomto objekte sú uložené informácie o nadviazanom spojení, medzi ktorými je aj premenná query. V tejto premennej sú uložené informácie o názve editovaného súboru, bezpečnostný token a meno pripájaného používateľa.

Po úspešnom nadviazaní spojenia vytvoríme nový objekt používateľa typu `User` s menom získaným z premennej `socket.handshake.query['name']` a nastavíme mu bezpečnostný token funkciou `setToken(socket.handshake.query['secret'])`. Pri nastavení tokenu sa aplikuje jeho automatická validácia a nastavenie premennej `verified`.

Inicializácia miestnosti

Ak je je používateľský token vyhodnotený ako správny, nasleduje inicializácia miestnosti. Táto operácia sa vykonáva pomocou metódy `createRoom(socket.handshake.query['file'])` triedy `RoomManager`. V prípade že miestnosť s požadovaným názvom neexistuje, vytvorí sa jej nová inštancia. Používateľ sa zaradí do zoznamu používateľov miestnosti a je mu odoslaná notifikácia o tejto udalosti spolu s objektom používateľa.

Súčasne sa vykoná načítanie editovaného súboru pomocou asynchrónneho HTTP dopytu na koncový bod API systému MediaWiki. V prípade že API nájde zadaný súbor, odpoveďou je objekt tohoto súboru, s informáciami o jeho rozmeroch, MIME formáte.

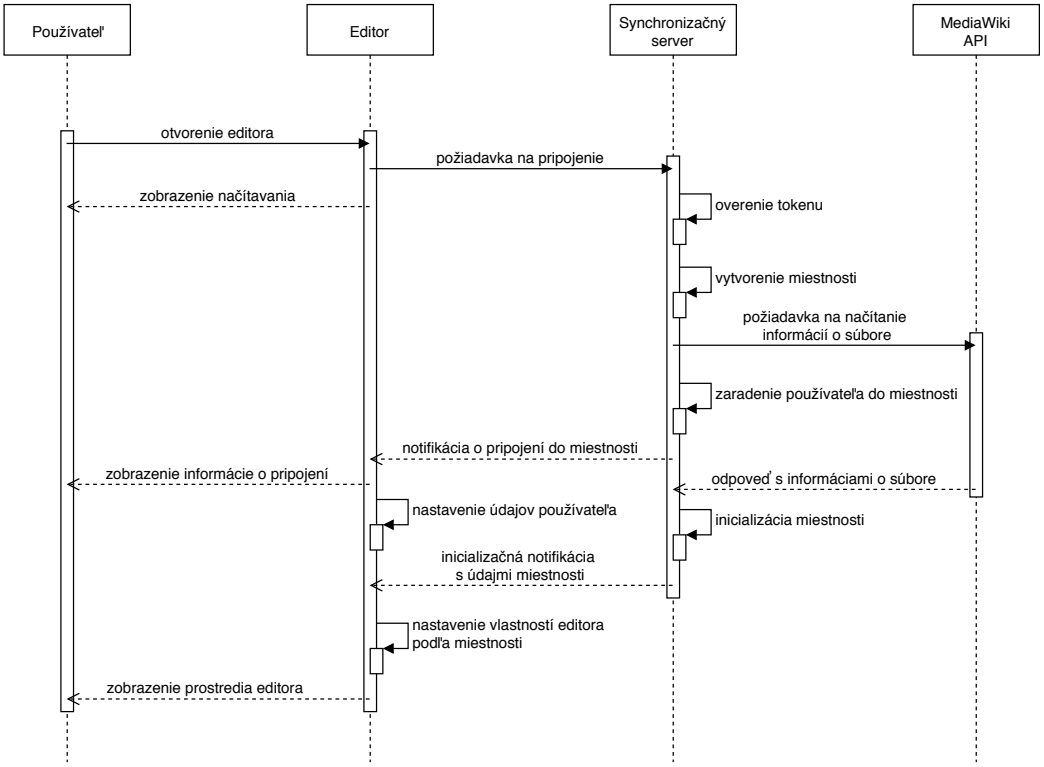
Ak sa jedná o súbor, ktorý už bol editovaný pomocou nášho rozšírenia, vnú-

torná reprezentácia editora je uložená vo forme textového reťazca v metadátach súboru. Tento reťazec obsahujúci vlastnosti grafickej plochy a objektov v nej umiesnených dekódujeme do formátu JSON a nastavíme potrebné vlastnosti triednych premenných miestnosti.

Ak načítavaný súbor ešte nebol upravovaný našim editorom, vytvoríme nový grafický objekt editora typu obrázok a pridáme ho do zoznamu objektov grafickej plochy miestnosti. Pri neexistujúcom súbore sú ponechané pôvodné vlastnosti miestnosti.

Po inicializovaní vlastností miestnosti je odoslaná používateľovi notifikácia s jej inicializovanými dátami, na základe ktorých sa mu nastaví prostredie editora (Obrázok 5.1).

Pripojením používateľa do existujúcej miestnosti, proces pripojenia pozostáva iba zo zaradenia používateľa do zoznamu používateľov, odoslania notifikácie o úspešnom pripojení spolu s odoslaním notifikácie všetkým ostatným používateľom. Pripájanému používateľovi sa taktiež odosiela inicializačná notifikácia s aktuálnym stavom grafického editora, vytvorenými objektami a pripojenými používateľmi.



Obr. 5.1: Sekvenčný diagram úspešného pripojenia používateľa na synchronizačný server

5.1.3 Spracovanie udalostí grafického editora

S pripojením používateľa do miestnosti získava možnosť odosielať a prijímať synchronizačné správy. Synchronizačný server prijíma nasledujúce názvy udalostí:

- `'canvas-modified'` - *Zmeny grafickej plochy* spracováva triedna metóda `room.modifyCanvas(properties, socket);`, ktorá ej parametrami sú objekt `properties` obsahujúce vlastnosti zmenenej grafickej plochy editora a `socket` objekt odosielateľa. Po zmene vlastností plochy v rámci servera je odoslaná udalosť so zmenami všetkým ostatným používate-

ľom.

- `'object-created'` - *Vytvorenie objektu* je spracované metódou `room.createObject(object, socket);`. Tá vytvorí nový záznam v zozname objektov editora, indexovaný podľa jedinečného identifikátora objektu a odošle udalosť na vytvorenie objektu všetkým ostatným používateľom.

- `'selection-changed'` - *Zmena uzamknutia objektu* sa vyhodnotí metódou `room.setSelectable(data.id, data.selectable, user, socket);`. Parametrami sú jedinečný identifikátor objektu, stav či má byť objekt uzamknutý alebo odomknutý a používateľ ktorý vykonáva túto zmenu. V prípade že objekt nemôže byť uzamknutý z dôvodu že už je uzamknutý iným používateľom, odosielateľ obdrží udalosť o zakázaní označenia daného objektu. Inak sa nastaví hodnota uzamknutia a odošle sa ostatným používateľom.

- `'object-modified'` - *Modifikácia objektu* je spracovaná pomocou metódy `room.modifyObject(object, socket);`, ktorá vyhľadá objekt na základe jedinečného identifikátora v zozname objektov a nastaví jeho nové vlastnosti podľa vstupného parametra `object`. V prípade že objekt v zozname neexistuje, vytvorí ho. Funkcia ďalej odošle udalosť na modifikáciu objektu všetkým ostatným používateľom.

- `'object-removed'` - *Odstránenie objektu* zabezpečuje triedna metóda `room.removeObject(id, socket);`. Tá odstráni objekt s jedinečným identifikátorom podľa vstupného parametra zo zoznamu objektov miestnosti a odošle udalosť na odstránenie objektu ostatným používateľom.

- `'message-created'` - *Vytvorenie textovej správy* v chate editora je spra-

covaná metódou `room.createMessage(message.text, message.from, message.to);`. Parameterami sú údaje prijaté v požiadavke, nesúce informácie o obsahu, odosielateľovi a prijímateľovi správy. Táto správa sa následne rozdistribuuje medzi všetkých používateľov pripojených do miestnosti.

5.1.4 Odpojenie klienta

Proces odpojenia používateľa pozostáva z jeho odstránenia zo zoznamu používateľov v miestnosti, zrušenia príznaku uzamknutého objektu všetkých objektov, ktoré mal používateľ pri odchode uzamknuté, a odoslania udalosti všetkým zvyšným používateľom. Trieda `RoomManager` overí počet používateľov v miestnosti a pokiaľ zostala miestnosť prázdna, vymaže ju zo svojho zoznamu pomocou metódy `removeRoom(name);`.

5.2 Grafický editor obrázkov

Druhou časťou implementácie tejto diplomovej práce je naprogramovanie grafického editora obrázkov. Editor je implementovaný formou *special page* rozšírenia, integrovaného do systému MediaWiki. Na vykresľovanie objektov grafickej plochy a udržiavanie jej stavu využívame knižnicu Fabric. Dátový model aplikácie je preto z veľkej časti závislý od dátového modelu tejto knižnice. Kolaboratívna funkcionalita je docielená použitím klientskej verzie Socket.IO knižnice, pomocou ktorej zabezpečujeme komunikáciu so synchronizačným serverom. Na riadenie klientského rozhrania používame knižnicu AngularJS. Komponenty používateľského prostredia sú naprogramované v šablónovacom

jazyku HTML a ich vzhľad je naprogramovaný pomocou štýlovacieho jazyka LESS.

5.2.1 MediaWiki rozšírenie

Pri tvorbe rozšírení pre MediaWiky je potrebné dodržať základné návrhové konvencie stanovené tvorcami tohoto systému. Prvoradou vlastnosťou je jednoduchá inštalácia a nastavenie rozšírenia. To je docielené pomocou viacerých faktorov. Prvoradé je používanie správnej súborovej štruktúry.

Rozšírenie musí byť umiestnené vo vlastnom priečinku medzi ostatnými MediaWiki rozšíreniami. V našom prípade je to priečinok `/extensions/ImageEditor`, kde sa nachádzajú zdrojové súbory a priečinky kategorizované podľa ich obsahu. V priečinku i18n sa nachádzajú php súbory s prekladmi textov použitých v prostredí rozšírenia, nazvané skratkou jazyka v ktorom sú preložené. Pre slovenčinu je to `sk.php`.

Zdrojový súbor `ImageEditor.php` zabezpečuje možnosť načítania rozšírenia pomocou príkazu `wfLoadExtension( 'ImageEditor');` v konfiguračnom súbore `LocalSettings.php`.

Súbor `ImageEditor.hooks.php` obsahuje funkcie, pomocou ktorých vieme odchytiť niektoré akcie, vykonávané jadrom systému, a ovplyvniť tak jeho základnú funkcionalitu. V tomto súbore sa nachádza trieda `ImageEditorHooks` poskytujúca nasledujúce statické metódy:

- `onFileUpload` - funkcia odchytávajúca nahrávanie nového multimediálneho súboru alebo jeho novej revízie, pomocou ktorej uložíme obsah

editora do metadát súboru. Metadáta sú uchovávané v databáze pomocou blob databázového dátového typu, ktorého obsahom je serializované pole jazyka php. Naše rozšírenie pri ukladaní revízie zapíše do tohoto poľa svoju dátovú štruktúru serializovanú do JSON textového formátu, ktorá je odosielaná pri ukladaní súboru pomocou HTTP POST dotazu, pod indexom `imageEditorContent`.

- `onSkinTemplateNavigation` - metóda odchytáva proces vykreslenia navigačných tlačidiel pri vykreslení šablóny stránok. Na základe overenia práv používateľa a typu vykresľovanej stránky sa pridá tlačidlo na otvorenie súboru v našom editore obrázkov.

- `onSkinTemplateNavigation_SpecialPage` - podobná metóda ako `onSkinTemplateNavigation` s výnimkou že táto udalosť je volaná pri vykresľovaní šablóny špeciálnych stránok.

- `onResourceLoaderGetConfigVars` - metóda slúži na pridanie konfiguračných premenných posielaných do JavaScriptového rozhrania systému MediaWiki. Vďaka tomu môžeme pristupovať k premenným definujúcim bezpečnostný token, port a adresu synchronizačného servera z klientskej aplikácie naprogramovanej v jazyku JavaScript pomocou premenných

  - `mw.config.values.wgImageEditor.secret`
  - `mw.config.values.wgImageEditor.host`
  - `mw.config.values.wgImageEditor.port`

V súbore `templates/ImageEditorTemplate.php` sa nachádza php trieda `ImageEditorTemplate` dedená od triedy `QuickTemplate`, obsahujúca jednu me-

tódu s názvom `execute()`. Volaním tejto metódy sa vykreslí HTML šablóna nášho rozšírenia.

Hlavná logika načítania rozšírenia sa však odohráva v triede `SpecialImageEditor` rozširujúcej triedu `SpecialPage`, umiestnenej v súbore `specials/SpecialImageEditor.php`. V metóde `execute()` sa vykoná overenie práv na upravovanie stránok a overí sa či je vybraný súbor ktorý chceme upravovať. Ak používateľ nemá práva na úpravu, je presmerovaný buď na prihlásenie (neprihlásený používateľ) alebo chybovú stránku.

Ak používateľ vytvára nový súbor pomocou adresy `https://wiki.matfyz.sk/Special:ImageEditor` zobrazí sa mu formulár so výzvou na zadanie názvu súboru. Po jeho zadaní je presmerovaný do prostredia editora.

Zdrojové kódy editora jazyka JavaScript sú umiestnené v priečinku `resources/scripts`. Pre ich lepšiu prehľadnosť sme ich rozdelili do súborov podľa zamerania funkcií ktoré obsahujú (Tabuľka 5.7). Kaskádové štýly používateľského prostredia editora sa nachádzajú v priečinku `resources/styles` a taktiež sú rozdelené podľa funkcionality (Tabuľka 5.8). Použité knižnice sa nachádzajú v priečinku `resources/vendor` a zdrojové obrázky spolu s písmami v priečinku `resources/assets`.

Pri tvorbe rozšírenia pre systém MediaWiki je veľmi dôležitý súbor `extension.json` nachádzajúci sa v jeho koreňovom priečinku. Slúži na definovanie umiestnenia súborov, závislostí na iných rozšíreniach a ich automatického načítavania pomocou *resource loadera*.

Názov	Popis
ext.imageEditor.utils.js	Definície globálnych funkcií a prototypovanie vlastných funkcií do pre knižnicu Fabric
ext.imageEditor.init.databinding.js	Definície funkcií na prístup k premenným objektov knižnice Fabric pomocou Angular controllera
ext.imageEditor.init.tools.js	Definície funkcií na prácu s používateľským prostredím editora
ext.imageEditor.init.events.js	Definície funkcií na riadenie udalostí (event-handlers) grafickej plochy a synchronizačných správ
ext.imageEditor.init.keybindings.js	Definícia funkcie na spracovanie klávesových skratiek
ext.imageEditor.init.history.js	Príprava pre funkcionalitu ukladania histórie zmien (undo, redo)
ext.imageEditor.init.app.js	Inicializácia Angular controllera, služby poskytujúcej prístup k funkciám knižnice Socket.IO a definícia direktív Angular modulu pre vytvorenie vlastných HTML elementov a atribútov.

Tabuľka 5.7: Zoznam JavaScriptových zdrojových súborov editora

Názov	Popis
ext.imageEditor.vars	Definície premenných a mixinov a dedičných tried
ext.imageEditor.colorpicker	Štýly pre komponent vyberača farieb
ext.imageEditor.icons	Definície premenných a tried pre ikony používateľského prostredia
ext.imageEditor.less	Definície štýlov používateľského prostredia editora

Tabuľka 5.8: Zoznam LESS zdrojových súborov kaskádových štýlov

Názov	Popis
bindValueTo	Direktíva na dynamické mapovanie get/set funkcií nastavujúcich hodnoty modelu Fabric objektov. Zabezpečuje zmenu modelu pri zmene hodnoty HTML vstupných elementov vykreslených šablónou editora.
onEnter	Direktíva, ktorá pridaním svojho atribútu do HTML vstupného elementu zabezpečuje volanie funkcií pri stlačení klávesy enter.
filesInput	Direktíva poskytuje funkciu načítania obrazového súboru, a následné mapovanie jeho hodnoty v textovom dataURI formáte do zvolenej premennej modelu aplikácie.
socket	Direktíva poskytujúca prístup k službám knižnice Socket.IO pomocou funkcie `$scope.socket.on('event-name', callback)`, odchytávajúce udalosti prichádzajúce zo synchronizačného servera, a funkcie `$scope.socket.emit('event-name', {...})`, odosielajúcej udalosti na synchronizačný server.

Tabuľka 5.9: Zoznam direktív AngularJS modulu ImageEditor

5.2.2 Funkcie editora

Aplikácia grafického editora je naprogramovaná v jazyku JavaScript pomocou frameworku AngularJS. Všetky jej funkcie sú naprogramované formou modulu s názvom `ImageEditor`. Modul poskytuje direktívy pre šablónu, ktoré zabezpečujú dynamické vykresľovanie niektorých používateľských komponentov, pridávajú im dodatočnú funkcionalitu, upravujú spôsob mapovania dát alebo poskytujú prístup k službám knižnice Socket.IO (Tabuľka 5.9). Okrem direktív modul obsahuje taktiež controller s názvom `ImageEditorController`, ktorý riadi funkcionalitu a správanie grafického editora. Kvôli dostupnosti funkcií priamo v šablóne, je potrebné všetky funkcie a premenné definovať

do `$scope` objektu.

Mapovanie dát

Riadenie udalostí (Event-handler)

Klávesové skratky

6

Výsledky

...

7

Záver

...

Literatúra

[CGM+14] Rik Cabanier, Eliot Graff, Jay Munro, Tom Wiltzius, and I Hickson. Html canvas 2d context. *W3C Candidate Recommendation (work in progress)*, 21, 2014.

[CS16] W. Chansuwath and T. Senivongse. A model-driven development of web applications using angularjs framework. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6, June 2016.

[Fou17] Node.js Foundation. Docs | node.js [online], Máj 2017. `https://nodejs.org/en/docs/`.

[Goo17] Inc. Google. Angularjs api docs [online], Máj 2017. `https://docs.angularjs.org/api`.

[JBM15] Nilesh Jain, Ashok Bhansali, and Deepak Mehta. Angularjs: A modern mvc framework in javascript. *International Journal of Global Research in Computer Science (UGC Approved Journal)*, 5(12):17–23, 2015.

[KAW+14] M. Kuhara, N. Amano, K. Watanabe, Y. Nogami, and M. Fukushi. A peer-to-peer communication function among web browsers for web-based volunteer computing. In *2014 14th International Symposium on Communications and Information Technologies (ISCIT)*, pages 383–387, Sept 2014.

[KNGR13] S. Kode, K. Nagaraju, L. Gollapudi, and S. K. Reddy. Using mediawiki to increase teaching expertise in engineering colleges. In *2013 IEEE Fifth International Conference on Technology for Education (t4e 2013)*, pages 204–205, Dec 2013.

[KVV06] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic mediawiki. In *International semantic web conference*, pages 935–942. Springer, 2006.

[LCL04] Paul Benjamin Lowry, Aaron Curtis, and Michelle René Lowry. Building a taxonomy and nomenclature of collaborative writing to improve interdisciplinary research and practice. *The Journal of Business Communication (1973)*, 41(1):66–99, 2004.

[Med17a] MediaWiki.org. Help:formatting mediawiki - mediawiki [online], Máj 2017.
`https://www.mediawiki.org/wiki/Help:Formatting`.

[Med17b] MediaWiki.org. Manual:coding conventions - mediawiki [online], Máj 2017.
`https://www.mediawiki.org/wiki/Manual:Coding_conventions`.

[Med17c] MediaWiki.org. Manual:developing extensions - mediawiki [online], Máj 2017.

`https://www.mediawiki.org/wiki/Manual:Developing_extensions`.

[Med17d] MediaWiki.org. Mediawiki docs [online], Máj 2017. `https://doc.wikimedia.org/mediawiki-core/master/php/`.

[Pee17] Peerjs.com. Peerjs documentation [online], Máj 2017. `http://peerjs.com/docs/#api`.

[Rai13] Rohit Rai. *Socket. IO Real-time Web Application Development*. Packt Publishing Ltd, 2013.

[tea17] Fabric.js team. Fabricjs doc [online], Máj 2017. `http://fabricjs.com/docs`.

[TV10] S. Tilkov and S. Vinoski. Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14:80–83, 11 2010.

[VSB16] S. Vashishth, Y. Sinha, and K. H. Babu. Addressing challenges in browser based p2p content sharing framework using webrtc. In *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pages 850–857, March 2016.

[W3S17] W3Schools.com. Angularjs tutorial [online], Máj 2017. `https://www.w3schools.com/angular/`.

[WSM13] Vanessa Wang, Frank Salim, and Peter Moskovits. *The WebSocket Protocol*, pages 33–60. Apress, Berkeley, CA, 2013.