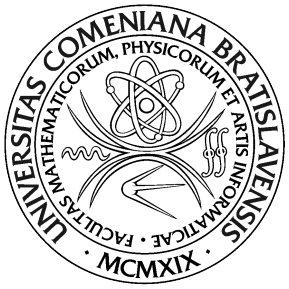
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

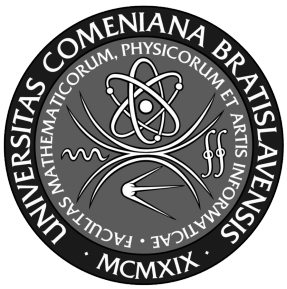


KOLABORATÍVNY GRAFICKÝ EDITOR PRE MEDIAWIKI

Diplomová práca

2018 Bc. Martin Krasňan

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



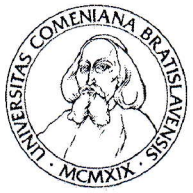
KOLABORATÍVNY GRAFICKÝ EDITOR PRE MEDIAWIKI

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: Kolaboratívny grafický editor pre MediaWiki
Collaborative graphics editor for MediaWiki

Anotácia: Tímová a kolaboratívna práca je v súčasnosti považovaná za dôležitú súčasť pracovných i vzdelávacích aktivít. Jedným z nástrojov, ktoré možno pri kolaboratívnej práci využiť, je wiki. Tento webový nástroj však nemá integrovaný grafický editor, ktorý by umožňoval kolaboratívnu tvorbu a úpravu obrázkov.

Cieľ: Cieľom práce je navrhnúť a implementovať grafický editor pre MediaWiki vhodný aj 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

Ďakujem doc. RNDr. Zuzane Kubincovej PhD. za čas a trpezlivosť, ktorú mi venovala pri konzultáciách, za cenné rady a profesionálne vedenie pri vypracovaní diplomovej práce. Moja veľká vďaka patrí taktiež Mgr. Jánovi Kľukovi, PhD. za ochotu a pomoc pri riešení problémov spojených s touto prácou. V neposlednom rade sa chem poďakovať mojim rodičom, starým rodičom a sestre za ich podporu.

Abstrakt

KRASŇAN, Martin: *Kolaboratívny 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. 85 strán.

Pri súčasnom rozvoji používania internetu a webových aplikácií je vítanou možnosťou kolaboratívne vytváranie informácií, ktoré sú na internete vyhľadávané a udržiavané. Jedným z používaných nástrojov umožňujúcich tento proces je systém MediaWiki. Cieľom diplomovej práce bolo navrhnúť a implementovať grafický editor pre systém MediaWiki, umožňujúci kolaboratívne kreslenie a úpravu obrázkov. Vytvorený editor bolo potrebné integrovať do stránky wiki.matfyz.sk. Nami implementované riešenie pozostáva z dvoch častí. Prvú časť tvorí implementácia grafického editora vektorových obrázkov. Integrácia do systému MediaWiki bola realizovaná formou rozšírenia. V druhej časti sme vytvorili komunikačný protokol a synchronizačný server, slúžiaci na výmenu informácií o obsahu grafickej plochy editorov spolupracujúcich používateľov.

Kľúčové slová: kolaboratívny grafický editor, webová aplikácia, rozšírenie MediaWiki

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. 85 pages.

At a current rate of the Internet and web application expansion, it is a welcome opportunity to collaboratively cerate and share information that is searched and maintained on the Internet. One of the tools used to enable this process is MediaWiki. The aim of the diploma thesis was to design and implement a graphic editor for MediaWiki, enabling collaborative drawing and editing of images. The created editor needed to be integrated into wiki.matfyz.sk. The solution implemented by us consists of two parts. The first part consists of implementing a graphic editor of vector images. Integration into MediaWiki has been implemented as an extension. In the second part, we have created a communication protocol and a synchronization server to exchange information about the content of the collaborative user's graphics area.

Keywords: collaborative graphics editor, web application, MediaWiki extension

Obsah

Úvod

Vďaka rozvoju informačných technológií vzrastá aj dopyt po vytváraní multimediálneho obsahu. Informácie sa už neudržiavajú iba vo fyzickej podobe formou kníh alebo obrazov, ale sú vytvárané online a udržiavané v digitálnej podobe na dátových úložiskách rozličných internetových aplikácií a serverov. Preto je potrebné tieto informácie jednoduchým a efektívnym spôsobom spravovať.

Na zhotovenie textových alebo grafických dokumentov dnes slúžia rôzne nástroje. Tie môžu byť realizované formou počítačových alebo webových aplikácií dostupných na rôznych zariadeniach. Informácie uchovávané online sú väčšinou prístupné pre široké spektrum používateľov, ktorí sa môžu spoločne zapájať do ich tvorby. Spoločné vytváranie informácií prináša viaceré pohľady na danú problematiku s cieľom lepšej výpovednej hodnoty výsledného obsahu. Medzi najznámejšie zdroje overených informácií patrí internetová encyklopédia Wikipedia. Táto stránka postavená na systéme *MediaWiki* umožňuje voľnú úpravu informatívnych článkov, v ktorých sa môžu nachádzať rozličné typy multimediálnych súborov. Multimediálne súbory a obrázky je nutné vytvárať a upravovať pomocou externých nástrojov a výsledok do

systému importovať. Preto vzniká potreba integrácie nástroja slúžiaceho na dynamické vytváranie multimediálneho obsahu priamo v prostredí používaného systému. Umožnenie spolupráce viacerým používateľom súčasne, zabezpečuje rýchlejšie a kvalitnejšie spracovanie.

V našej diplomovej práci sa zameriame na navrhnutie a implementáciu grafického editora obrázkov, pomocou ktorého bude možné kresliť obrazové súbory priamo v prostrediach webových aplikácií, vytvorených systémom *MediaWiki*. Ten umožní spoluprácu viacerých používateľov na spoločnom zadaní, pričom zmeny vykonávané každým účastníkom budú v reálnom čase viditeľné a reflektované všetkým ostatným spolupracovníkom.

V prvej kapitole s názvom *"Prehľad problematiky"* si vysvetlíme pojmy, ktoré budeme využívať pri implementácii nášho riešenia a popíšeme si využité technológie. Následne navrhneme a vysvetlíme princípy implementácie grafického editora pomocou rozšírenia pre systém MediaWiki. V závere práce sa zameriame na spôsob testovania vytvorenej aplikácie a dosiahnuté výsledky.

1 Prehľad problematiky

1.1 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.

1.1.1 Počítačová grafika

Pojem počítačová grafika je z technického hľadiska odbor informatiky, ktorý sa zaoberá vykreslením a spracovaním obrazových informácií za pomoci výpočtovej techniky. Obrazové informácie môžu byť získavané napríklad pomocou digitálneho fotoaparátu, kde svetlo prechádzajúce sústavou šošoviek dopadá na optický snímač. Optický snímač 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 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 pomocou grafických softvérov.

Tento druh počítačovej grafiky zaraďujeme medzi rastrovú grafiku. Grafická informácia je tu teda 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 je napr. 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 pri hernom a strojárenskom priemysle, alebo na tvorbu filmových efektov. Využíva sa na vytvorenie virtuálneho modelu, ktorý sa snaží čo najdôvernejšie priblížiť objektu v reálnom svete.

1.1.2 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 účastníkom umožnená ich okamžitá reakcia,
- **Kvalita** – práca viacerých používateľov na jednej téme spoločne prináša širší uhol pohľadu a tým pádom aj lepšiu informačnú hodnotu.

Asynchrónne vytváranie obsahu môže byť organizované čiastkovými podúlo-

hami, 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.

1.2 Použité technológie

1.2.1 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.

1.2.2 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.

1.2.3 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 Unix, Windows alebo MacOS. PHP najčastejšie kooperuje s databázovými systémami typu MySQL, PostgreSQL alebo Microsoft SQL, v ktorých sú uchovávané dáta využívané aplikáciou.

1.2.4 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 populárnym 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 obrázok, kde je syntax zápisu nasledovná: `[[File:NazovSuboru.jpg]]`. Ďalšie informácie ohľadom spôsobu syntaxe wiki-textu je možné nájsť v dokumentácií MediaWiki systému [Med17a]. Obsah stránky je možné zapisovať priamo po-

mocou 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 histórii 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.

1.2.5 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.

1.3 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 najpodstatnejšie.

1.3.1 JavaScriptová knižnica Less.js

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

- premenných – špecifikovanie často používaných hodnôt na jednom mieste a následné referencovanie tejto hodnoty kdekoľvek v zdrojových súbo-

roch,

- 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 viacerými možnými spôsobmi, napríklad priamo v prehliadači používateľa v prostredí Node, PHP, .Net a ďalších. Výsledkom kompilácie je vytvorenie súboru s kaskádovými štýlmi vo formáte css.

1.3.2 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 nečaká na vykonanie pomalých operácií, ale vykoná svoju funkciona-

litu, 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 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.

1.3.3 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.

1.3.4 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, 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` – farba 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 1.1).

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

Zdrojový kód 1.1: Vytvorenie objektu typu obdĺžnik pomocou knižnice FabriJS a zmena jeho základných vlastností

Canvas

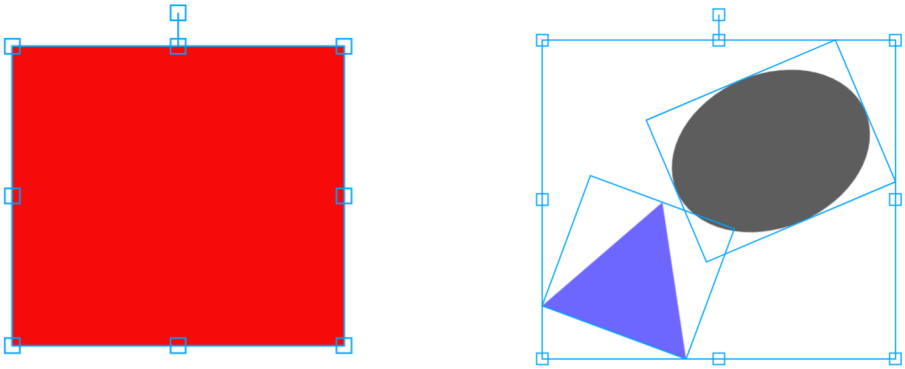
Inicializácia Fabric knižnice pozostáva z vytvorenia `fabric.Canvas` objektu zaobaľujúceho grafickú plochu editora. Ten je zodpovedný za spravovanie

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 1.2).

```
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 1.2: Inicializácia Fabric canvas wrappera

Jednou z unikátnych vstavaných vlastností knižnice Fabric je interakčná vrstva nad grafickou plochou canvas elementu, obsahujúceho objektový model. Objektový model existuje, aby umožnil programový prístup a manipuláciu s objektami canvas elementu. Z pohľadu používateľa je však možné manipulovať s objektami pomocou počítačovej myši, prípadne dotykov pri zariadeniach s dotykovou obrazovkou (Obrázok 1.1a). Ihneď po inicializácií pomocou `new fabric.Canvas('...')`, je možné vybrať objekt, posúvať ho ťahaním, skálovať, rotovať alebo zoskupiť viacero objektov do skupiny a manipulovať so všetkými súčasne (Obrázok 1.1b).



(a) Manipulácia s objektom (b) Manipulácia so skupinou objektov

Obr. 1.1: Manipulácia s objektami pomocou vstupných zariadení s použitím knižnice Fabric

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 1.3) 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 1.3: 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 1.4)

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

Zdrojový kód 1.4: Manuálne vyvolanie udalosti v knižnici Fabric

1.3.5 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 akciách používateľa alebo systému.

1.4 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.

Rozšírenia 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 je možné inicializovať ho 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 ktoré majú čakať pokiaľ sa inicializuje jadro MediaWiki.
- Definovať dodatočné hooky potrebné pre rozšírenie.
- Vytvoriť / skontrolovať všetky nové databázové tabuľky nutné pre rozšírenie.
- Nastaviť lokalizáciu rozšírenia.

1.4.1 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.

2

Predchádzajúce riešenia

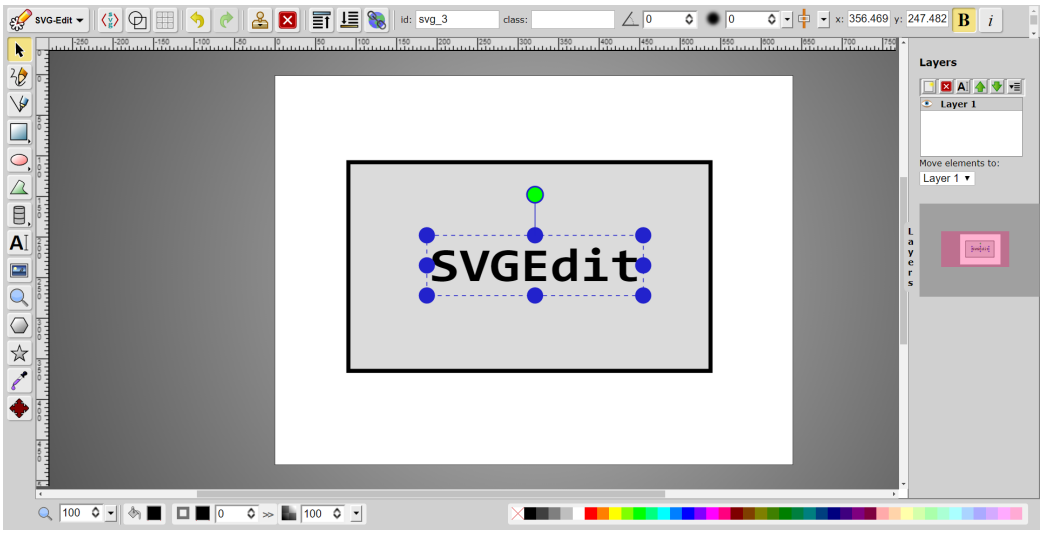
V nasledujúcej kapitole opíšeme existujúce riešenia, v ktorých sa zameriame na ich výhody a nevýhody.

2.1 SVGEdit

SVGEdit je jediné oficiálne rozšírenie MediaWiki existujúce v dobe písania tejto diplomovej práce. Aktuálne je 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.

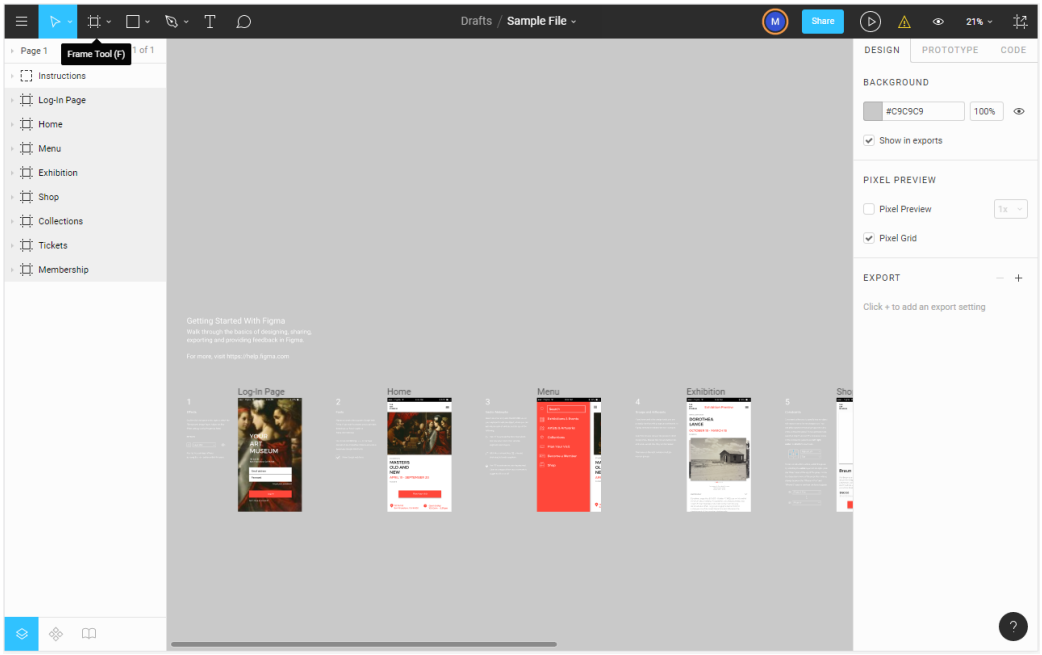


Obr. 2.1: Prostredie editora SVG-edit

2.2 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. 2.2: Prostredie kolaboratívneho editora Figma

2.3 Draw.IO

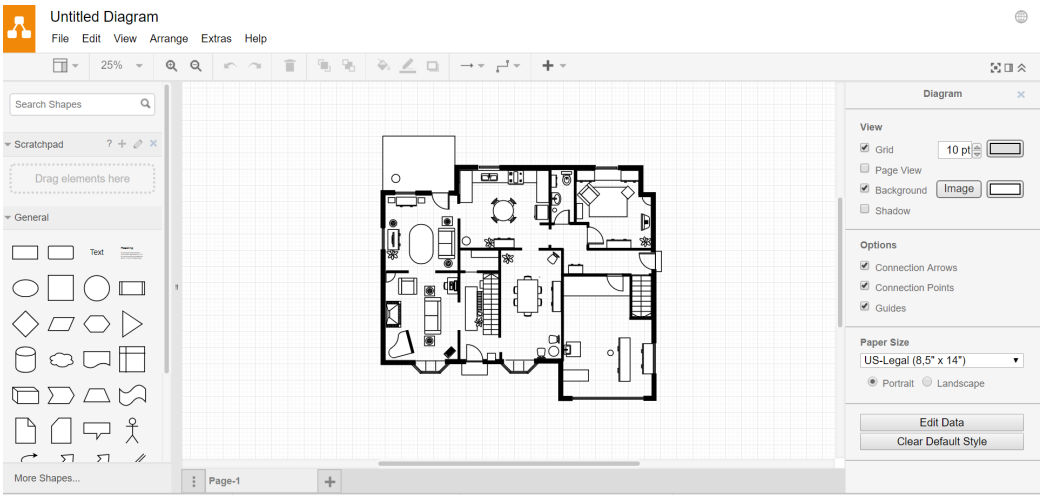
Draw.IO je online webová aplikácia slúžiaca na kolaboratívnu tvorbu vektorovej grafiky. Využitie tejto aplikácie je možné nájsť najmä pri tvorbe:

- *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 škálovateľnosť pre projekty s veľkým množstvom

elementov, možnosť bezplatného používania a voľná prístupnosť zdrojových súborov aplikácie pre vývojárov.

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



Obr. 2.3: Prostredie kolaboratívneho editora Draw.IO

3

Návrh modelu

V tejto kapitole podrobne rozoberieme 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 návrh 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.

3.1 Cieľ práce

Hlavným cieľom diplomovej práce bolo navrhnúť a implementovať kolaboratívny grafický editor pre MediaWiki a integrovať ho s portálom matfyz.sk. Na dosiahnutie tohto cieľa sme si stanovili nasledujúce podciele:

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

3.2 Synchronizačný server

Synchronizačný server bude aplikácia naprogramovaná v jazyku JavaScript. Spúšťaný bude v runtime prostredí NodeJS. Jeho 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 postavený na báze WebSocket protokolu. Aplikácia synchronizačného servera bude pozostávať z niekoľkých entít (objektov), ktoré si postupne popíšeme v ďalších častiach tejto diplomovej práce.

3.2.1 Popis entít synchronizačného servera

Entita používateľ

Základnou entitou bude objekt *Používateľ*(`User`), 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.

3.2.2 Funkčné požiadavky synchronizačného servera

- **Pripojenie používateľa** – vytvorí sa entita typu `User` a nastavia sa jej vlastnosti na základe zaslaného dopytu.
- **Overenie pripojeného používateľa** – po pripojení nového používateľa sa overuje správnosť bezpečnostného tokenu. Ak bezpečnostný 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 sa vytvorí 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 **I**nterface). 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 je-

dinečného 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 ostatným 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 na server 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 akcii 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 pro-

ces zabezpečí entita `RoomManager`.

3.3 Editor obrázkov

Grafický editor implementujeme formou MediaWiki rozšírenia, naprogramovaného primárne v jazyku JavaSrcipt, s použitím frameworku AngularJS (kap.: 1.3.3). Na komunikáciu so synchronizačným serverom použijeme JavaScriptovú knižnicu Socket.IO (kap.: 1.3.5), konkrétne jej klientskú verziu. Na vykresľovanie a udržiavanie stavu grafickej plochy použijeme knižnicu Fabric (kap.: 1.3.4).

3.3.1 Funkčné požiadavky grafického editora

Navrhovaný grafický editor by mal umožňovať nasledovné operácie:

- **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 škálovania (zoom).
- **Zmena vlastností grafickej plochy** – okrem škálovania 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 editor 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.

3.3.2 Integrácia do sytému MediaWiki

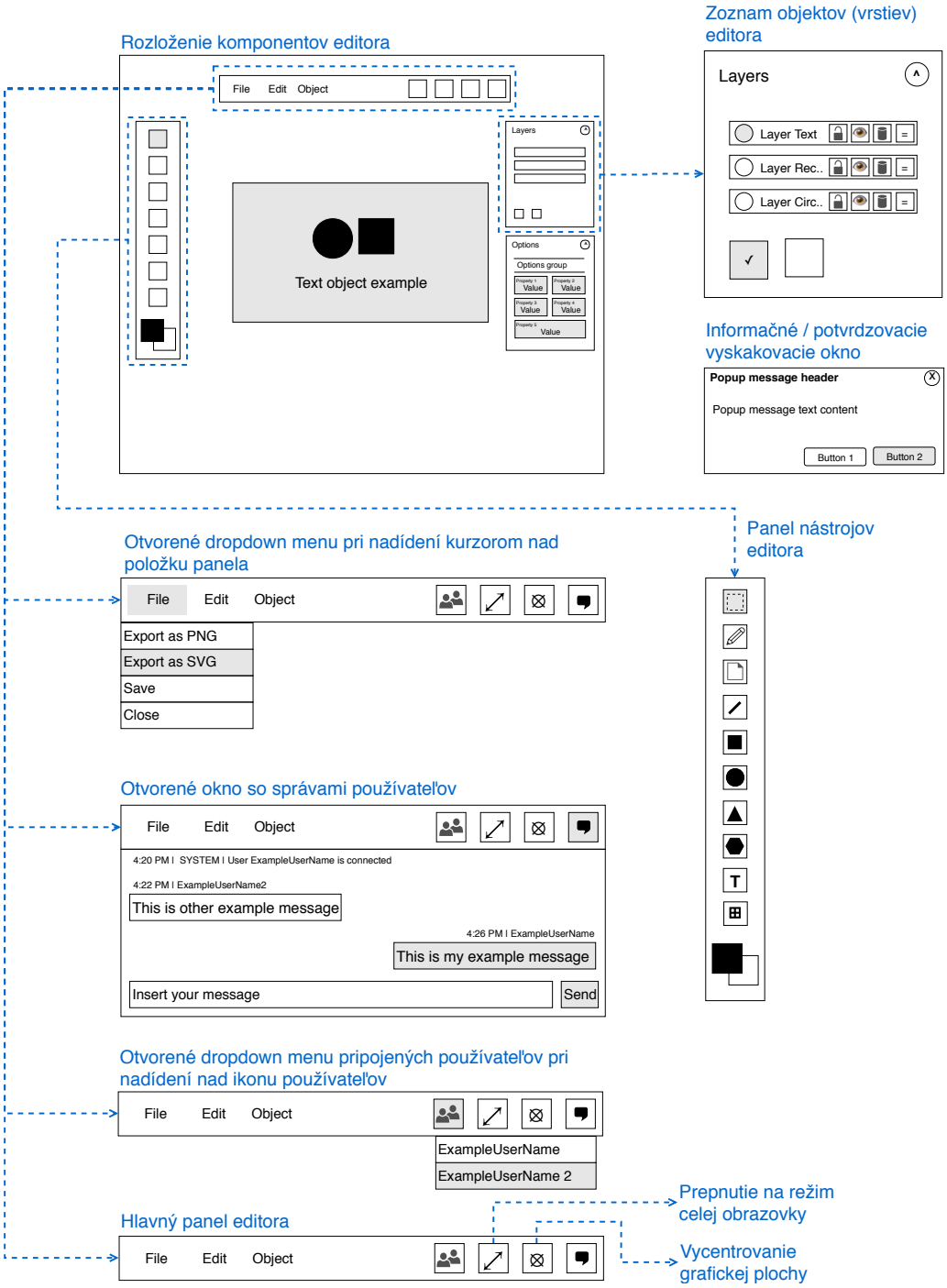
Ako sme si opísali v časti 1.4, pre systém MediaWiki existuje viacero typov rozšírení. Pre naše riešenie sme si zvolili rozšírenie typu *special page extension*. Učinili sme tak z dôvodu, že ide o najlepšiu voľbu z hľadiska vytvárania vlastného používateľského rozhrania a jeho špecifickej funkcionality.

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. Zobrazovanie tohoto tlačidla bude podmienené tým, či je používateľ prihlásený do systému a súčasne má práva na úpravu alebo vloženie multimediálnych súborov. 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.

3.3.3 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 (Obrázok 3.1). V našom prípade sme sa rozhodli použiť podobné rozloženie, kvôli zavedeným zvykom používateľov a tým pádom vyššej intuitívnosti pri používaní.



Obr. 3.1: Návrh rozloženia komponentov používateľského prostredia editora

4

Implementácia

V tejto kapitole sa zameriame na riešenie popísanej problematiky, implementáciu 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 funkcionalitu.

Pri implementácií využívame verziovací systém GIT. Obe hlavné časti sú

umiestnené vo vlastných repozitároch na serveri GitHub.

4.1 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.

4.1.1 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 4.1: 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 4.2: Zoznam metód objektu User

User

Trieda `User` reprezentuje používateľa pripojeného na synchronizačný server. Jej vlastnosti určujú triedne premenné a metódy opísané v tabuľkách 4.1 a 4.2.

Message

Objekt `Message` reprezentuje správy odoslané používateľmi pomocou chatu v editore. Popis vnútornej reprezentácie vlastností objektu môžeme vidieť v tabuľke 4.3.

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 4.3: 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 4.4: 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 4.4 a 4.5.

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 4.5: 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 4.6: Zoznam metód objektu RoomManager

RoomManager

Trieda manažéra 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 4.6.

4.1.2 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. Informácia o pripájanom klientovi je do nej poslaná

parametrom `socket`. Pomocou tohoto parametra následne server odchytáva 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.

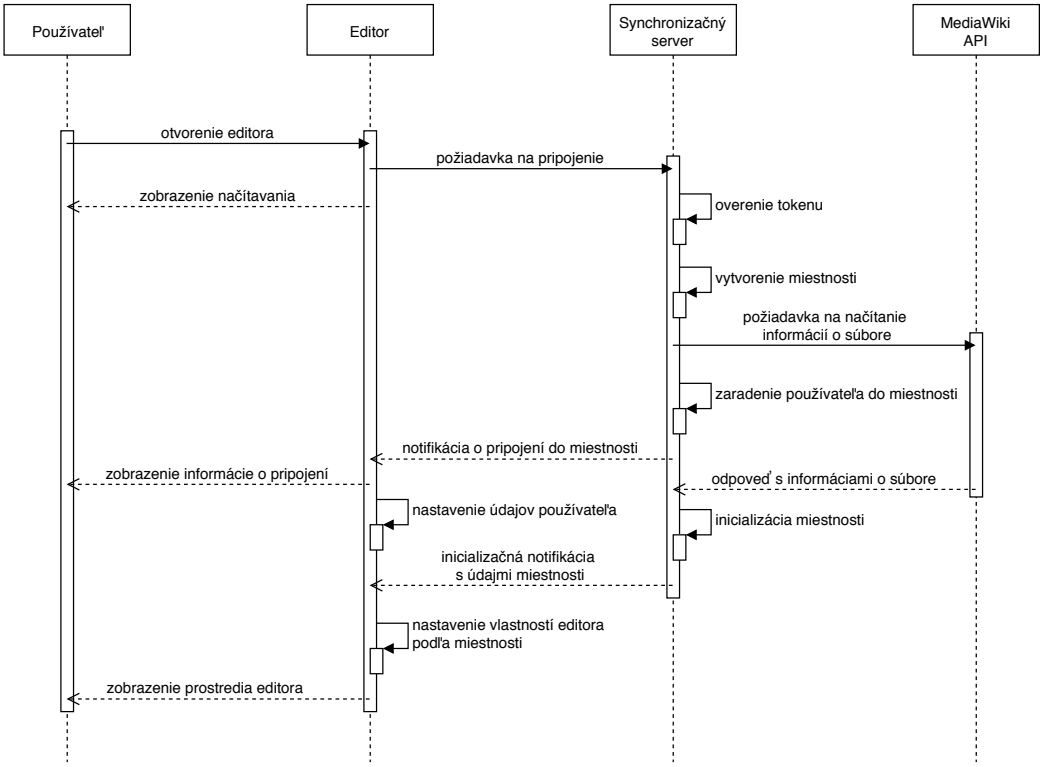
Ak ide o súbor, ktorý už bol editovaný pomocou nášho rozšírenia, vnútorná

reprezentácia editora je uložená vo forme textového reťazca v metadátach súboru. Tento reťazec obsahujúci vlastnosti grafickej plochy a objektov v nej umiestnených 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 4.1).

Pripojením používateľa do existujúcej miestnosti pozostáva proces pripojenia iba zo zaradenia používateľa do zoznamu používateľov a 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. 4.1: Sekvenčný diagram úspešného pripojenia používateľa na synchronizačný server

4.1.3 Spracovanie udalostí grafického editora

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

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

- `'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 bol 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 spracovaná 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.

4.1.4 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);`.

4.2 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.

4.2.1 MediaWiki rozšírenie

Pri tvorbe rozšírení pre MediaWiki je potrebné dodržať základné návrhové konvencie stanovené tvorcami tohoto systému. Prvoradou vlastnosťou je jednoduchá inštalácia a nastavenie rozšírenia. To je docielené pomocou viacerých faktorov. Dôležité 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, do ktorého 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 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 for-

mou databázového dátového typu `blob`. 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 POST premennej HTTP 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 (Obrázok 4.3).

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

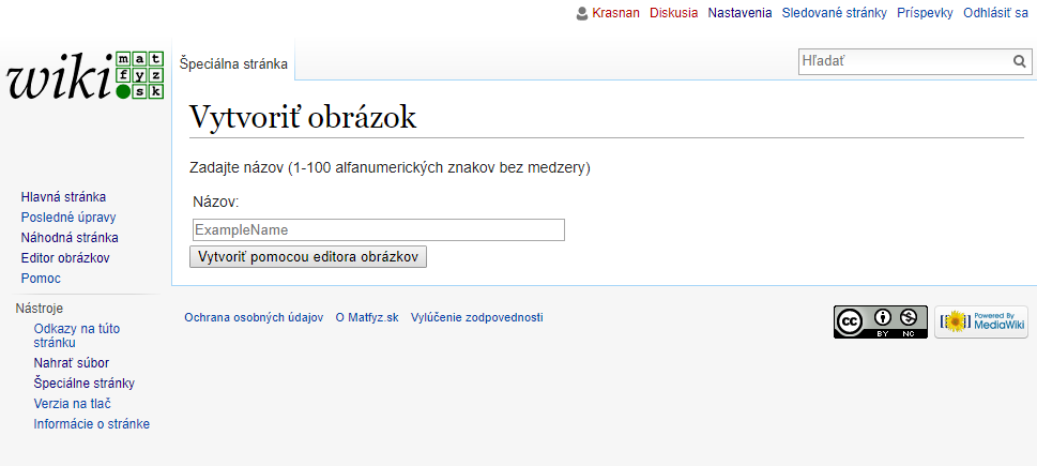
- `onResourceLoaderGetConfigVars` – metóda slúži na pridanie konfiguračných premenných posielaných do JavaScriptového rozhrania systému MediaWiki. Vďaka tomu môžeme pristupovať k premenným definujúcim bezpečnostný token, port a adresu synchronizačného servera z klientskej aplikácie naprogramovanej v 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 metódu s názvom `execute()`. Volaním tejto metódy sa vykreslí HTML šablóna nášho rozšírenia.

Hlavná logika načítania rozšírenia sa však odohráva v triede `SpecialImageEditor`, 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 s výzvou na zadanie názvu súboru (Obrázok 4.2). Po jeho zadaní je presmerovaný do prostredia editora.



Obr. 4.2: Používateľské prostredie na zadanie názvu nového obrázku.

Obr. 4.3: Integrácia do natívneho prostredia systému MediaWiki.

Zdrojové kódy editora naprogramované v jazyku JavaScript sú umiestnené v priečinku `resources/scripts`. Pre ich lepšiu prehľadnosť sme ich rozdelili do súborov podľa zamerania funkcií, ktoré obsahujú (Tabuľka 4.7). Kaskádové štýly používateľského prostredia editora sa nachádzajú v priečinku `resources/styles` a taktiež sú rozdelené podľa funkcionality (Tabuľka 4.8). 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 loaderu*.

Názov	Popis
ext.imageEditor.utils.js	Definície globálnych funkcií a prototypovanie vlastných metód knižnice Fabric.
ext.imageEditor.init.databinding.js	Definície funkcií na prístup k premenným objektov knižnice Fabric pomocou Angular controlleru.
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 4.7: Zoznam JavaScriptových zdrojových súborov editora

Názov	Popis
ext.imageEditor.vars	Definície premenných, 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 4.8: 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 4.9: Zoznam direktív AngularJS modulu ImageEditor

4.2.2 Aplikácia grafického editora

Aplikácia grafického editora je naprogramovaná v jazyku JavaScript pomocou frameworku AngularJS. Všetky jej funkcie sú naprogramované v module 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 4.9).

Okrem direktív modul obsahuje taktiež controller s názvom `ImageEditorController`, riadiaci funkcionalitu a správanie grafického editora.

Kvôli dostupnosti funkcií priamo v šablóne je potrebné všetky funkcie a premenné definovať do `$scope` objektu.

Mapovanie dát

Ako sme si vysvetlili v sekcii 1.3.3, Angular je navrhnutý tak, aby dokázal dynamicky zobrazovať hodnoty modelu v šablóne aplikácie pomocou dátového mapovania. Aby sme boli schopní mapovať dátový model Fabric canvas objektu, je potrebné naprogramovať metódy na prístup k jeho hodnotám. Musíme tak učiniť z dôvodu, že hodnoty modelu nie vždy zodpovedajú hodnotám, ktoré potrebujeme zobrazovať v používateľskom prostredí. Zdrojový súbor `ext.imageEditor.init.databinding.js` obsahuje funkcie určené na túto činnosť. Okrem funkcií popísaných v Tabuľke 4.10, sú v tomto súbore taktiež `set` a `get` funkcie definované pre každú premennú Fabric objektov [tea17], ku ktorej je potrebné pristupovať z používateľského prostredia.

Riadenie udalostí (Event-handler)

Naša aplikácia využíva princípy programovania riadeného udalosťami (event-driven). Dokáže dynamicky vyhodnocovať niektoré z udalostí, ktoré sa v nej vykonávajú, bez potreby volania funkcií spracovávajúcich danú akciu do častí kódu, kde je daná akcia vykonávaná. Namiesto toho sa použije funkcia `$scope.canvas.trigger('event-name', ...)`, pomocou ktorej sa vyvolá udalosť s názvom podľa prvého parametra a dátami uloženými v druhom parametri tejto funkcie. Následne je možné tieto udalosti jednoducho zachytiť v našej aplikácií.

Názov	Popis
`getActiveStyle(styleName, object)`	Funkcia na prístup k hodnote premennej štýlov `styleName` objektu `object` alebo aktívneho objektu.
`setActiveStyle(styleName, value, object)`	Funkcia na nastavenie premennej štýlu `styleName` hodnotou `value` objektu `objekt` alebo aktívneho objektu.
`getActiveProp(name)`	Funkcia na prístup k hodnote `name` premennej aktívneho objektu.
`setActiveProp(name, value)`	Funkcia na nastavenie premennej `name` hodnotou `value` aktívneho objektu.

Tabuľka 4.10: Zoznam základných funkcií nastavujúcich hodnoty modelu Fabric objektov

Okrem akcií grafickej plochy spracovávame taktiež udalosti vyvolávané synchronizačnými správami servera, poskytované službami knižnice Socket.IO. Zoznam a popis udalostí, ktoré naša aplikácia dokáže spracovať, môžeme vidieť v tabuľkách (Tabuľka 4.11) a (Tabuľka 4.12).

Názov	**Popis**
object:created	Vytvorenie nového grafického objektu. Odosiela sa požiadavka synchronizačnému serveru na vytvorenie nového objektu.
object:modified	Zmena grafického objektu. Odosiela sa požiadavka synchronizačnému serveru na modifikáciu objektu.
object:removed	Odstránenie grafického objektu. Odosiela sa požiadavka synchronizačnému serveru na vymazanie objektu.
object:moving	Pohyb objektu pomocou kurzora používateľa. Ak je zapnuté prichytávanie k mriežke, hodnoty pozície pohybovaného objektu sa nastavia na zaokrúhlené číslo v závislosti od veľkosti priblíženia grafickej plochy.
path:created	Vytvorenie nového grafického objektu počas voľného kreslenia. Odosiela sa požiadavka synchronizačnému serveru na vytvorenie nového objektu.
canvas:modified	Zmena vlastností grafickej plochy. Odosiela sa požiadavka synchronizačnému serveru na zmenu grafickej plochy.
selection:created, selection:updated, selection:cleared	Zmena označenia aktívneho objektu. Odosielajú sa požiadavky synchronizačnému serveru na zmenu uzamknutia objektov, ktorých sa zmena týka.
mouse:over	Nadídenie kurzorom nad objekt. Ak je objekt uzamknutý, zobrazí sa v grafickej ploche meno používateľa, ktorý ho uzamkol.
mouse:out	Odídenie kurzorom z objektu grafickej plochy. Ak je zobrazené meno používateľa nad objektom, skryje sa.

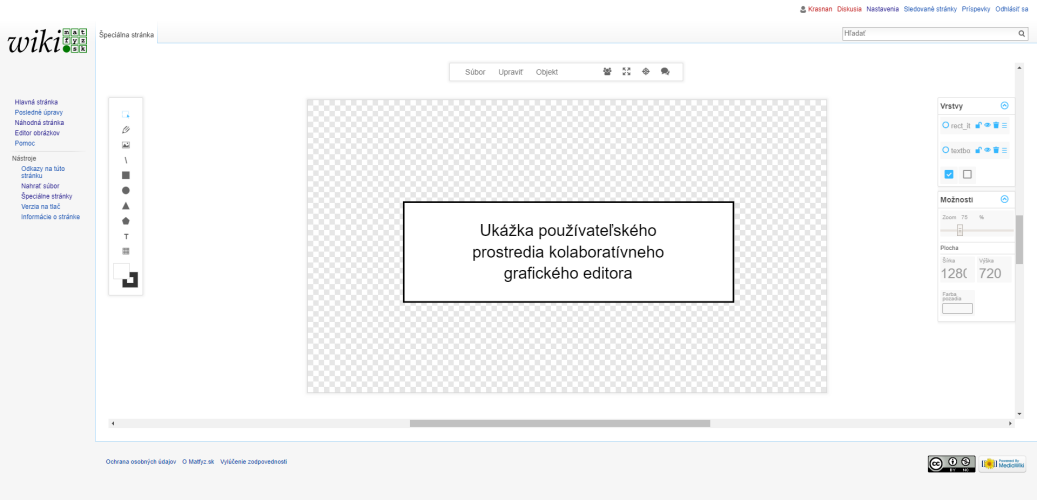
Tabuľka 4.11: Zoznam udalostí grafickej plochy

Názov	Popis
connected	Pripojenie na synchronizačný server. Nastaví sa objekt pripojeného používateľa a skryje sa správa o pripájaní v loaderi.
init	Inicializačná správa s informáciami, podľa ktorých sa nastavuje editor. Pridajú sa všetky objekty do grafickej plochy, vytvorí sa zoznam pripojených používateľov, načíta sa zoznam správ a nastavia sa vlastnosti grafickej plochy.
user-created	Pripojenie nového používateľa do miestnosti. Používateľ je pridaný do zoznamu všetkých používateľov.
user-removed	Odpojenie používateľa z miestnosti. Používateľ je odstránený zo zoznamu všetkých používateľov.
message-created	Prijatá nová správa. Nastaví sa príznak na zobrazenie notifikácie o novej správe a obsah správy sa zaradí do zoznamu správ.
selection-changed	Zmena uzamknutia objektu. Nastaví sa hodnota `selectable` a `selectedBy` podľa prijatej správy, čím sa zamedzí možnosti označiť daný objekt.
object-created	Vytvorenie objektu. Objekt prijatý v správe sa pridá do grafickej plochy. Ak objekt s daným identifikátorom existuje, proces sa ukončí.
object-modified	Modifikácia objektu. Na základe vlastností objektu prijatého v správe sa nastavia vlastnosti objektu s relevantným identifikátorom v grafickej ploche editora. Ak objekt s daným identifikátorom neexistuje, vytvorí sa, čím zabezpečíme nápravu prípadnej straty alebo výrazného oneskorenie správy o vytvorení objektu.
object-removed	Odstránenie objektu. Na základe prijatého identifikátora sa odstráni objekt z grafickej plochy.
canvas-modified	Zmena vlastností grafickej plochy. Nastavia sa vlastnosti na základe prijatých informácií.

Tabuľka 4.12: Zoznam udalostí synchronizačného servera

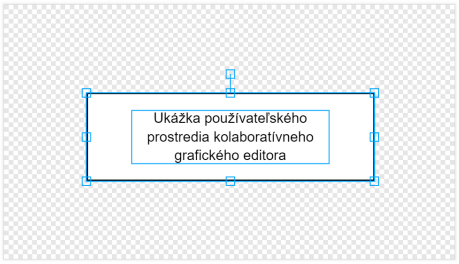
Používateľské prostredie

Používateľské prostredie editora je nadizajnované tak, aby bola každá funkcionalita prístupná pokiaľ možno na jedno kliknutie používateľa. Pozostáva zo 4. hlavných komponentov.

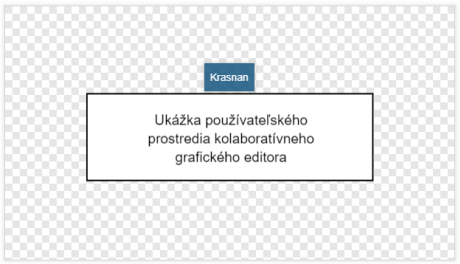


Obr. 4.4: Používateľské prostredie grafického editora obrázkov.

Grafická plocha, v ktorej sú vykresľované jednotlivé grafické objekty. Ak je niektorý z objektov uzamknutý iným používateľom, prístup k tomuto objektu bude zamietnutý. Pri nadídení kurzora počítačovej myši sa zobrazí meno používateľa používajúceho daný objekt (Obrázok 4.5b).



(a) Označenie aktívnych objektov

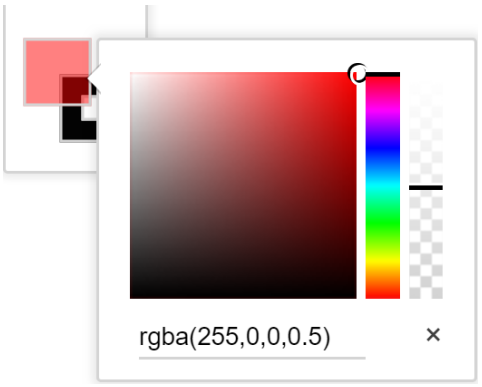


(b) Zobrazenia používateľa, ktorý uzamkol objekt

Obr. 4.5: Grafická plocha edit

Panel nástrojov s tlačidlom pre každý nástroj a tlačidlami pre výber farby výplne a obtiahnutia objektu. V paneli sa taktiež nachádza tlačidlo na zapnutie prichytávania objektov k mriežke grafickej plochy.

(a) Zoznam nástrojov grafického editora



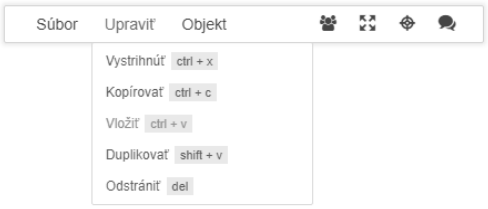
(b) Vyberač farby výplne a obtiahnutia objektu grafickej plochy

Obr. 4.6: Panel nástrojov

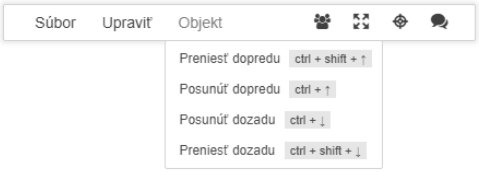
Hlavné menu obsahujúce tlačidlo na prepnutie zobrazenia do režimu celej obrazovky, tlačidlo na vycentrovanie grafickej plochy a rozbaľovací zoznam správ chatu (Obrázok 4.7e). Okrem týchto tlačidiel tu nájdeme aj 3 rozba-

ľovacie (dropdown) tlačidlá *Súbor* , *Upraviť* a *Objekt*. Rozbalením tlačidla *Súbor* (Obrázok 4.7a) sa zobrazí ponuka na exportovanie grafickej plochy do formátov PNG a SVG, uloženie revízie a zatvorenie editora. Rozbalením tlačidla *Upraviť* (Obrázok 4.7b) sa sprístupnia možnosti na kopírovanie, vystrihnutie, vloženie, duplikovanie a odstránenie objektu. Pod posledným tlačidlom *Objekt* (Obrázok 4.7c), sa zobrazia možnosti operácií s aktívnym objektom na zmenu hĺbky objektu v grafickej ploche.

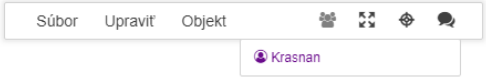
(a) Rozbaľovacie menu pre tlačidlo *Súbor*



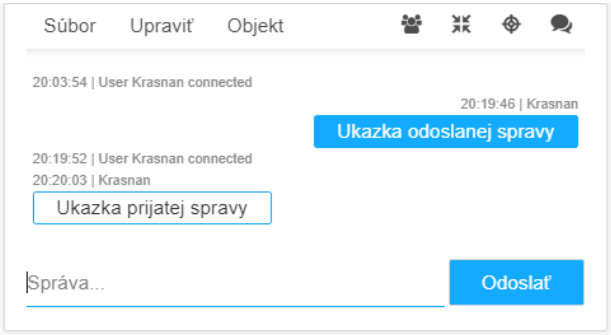
(b) Rozbaľovacie menu pre tlačidlo *Upraviť*



(c) Rozbaľovacie menu pre tlačidlo *Objekt*



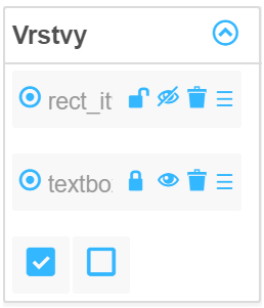
(d) Rozbaľovacie menu zoznamu pripojených používateľov



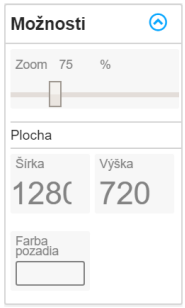
(e) Prostredie pre odosielanie textových správ

Obr. 4.7: Zobrazenie hlavného menu

Panel nastavení tvorený dvoma komponentmi. Komponent vrstiev zobrazujúci zoznam všetkých grafických objektov s možnosťou ich rýchleho uzamknutia, skrytia a odstránenia. Vrstvy môžu byť preusporiadané pomocou ťahania kurzora, čím sa mení ich hĺbka v grafickej ploche editora. Druhý komponent obsahujúci nastavenia aktuálneho kontextu editora dynamicky zobrazuje položky na nastavenie vlastností aktívneho objektu, grafickej plochy, alebo nastavenia voľného kreslenia (Obrázok 4.8).



(a) Panel vrstiev



(b) Panel nastavení grafickej plochy

(c) Panel nastavení voľného kreslenia

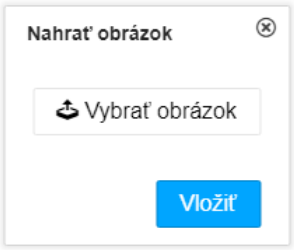
(d) Panel nastavení textového objektu

(e) Panel nastavení objektu

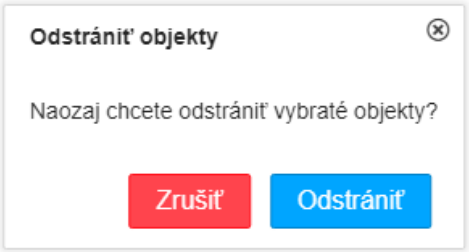
Obr. 4.8: Ukážka panelov s nastaveniami

Okrem týchto panelov je potrebné informovať používateľa o niektorých vykonávaných akciách formou dialógových (*popup*) správ s možnosťou potvrdenia alebo prípadného zrušenia akcie. Obsah dialógových okien je potrebné dy-

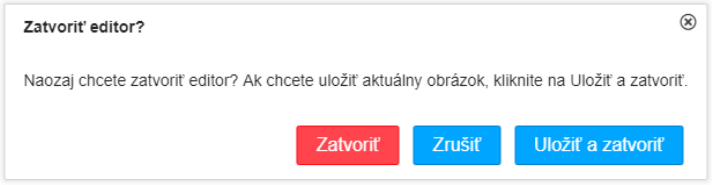
namicky vytvárať a modifikovať funkcionalitu potvrdzovacích tlačidiel. Využijeme ich na upozornenie používateľa pri odstraňovaní objektu z grafickej plochy (Obrázok 4.9b) a zatváraní okna editora, kde bude používateľ vyzvaný na uloženie súboru alebo ukončenie bez ukladania zmien (Obrázok 4.9c). Pomocou dialógového okna je taktiež riešené nahratie obrázku z lokálneho úložiska používateľa (Obrázok 4.9a).



(a) Dialógové okno na nahratie súboru



(b) Dialógové okno na potvrdenie vymazania grafického objektu



(c) Dialógové okno zobrazené pri zatváraní editora

Obr. 4.9: Dialógové okná

5

Výsledky

V tejto kapitole si priblížime metodiku testovania nášho riešenia, opíšeme si testovacie scenáre a zosumarizujeme výsledky, ktoré boli dosiahnuté. Taktiež sa zameriame na opis možných vylepšení a návrh prípadnej budúcej práce na rozšírení grafického editora.

5.1 Testovanie

Počas aktívneho vývoja aplikácie bolo zorganizované skupinové záťažové testovanie. Zúčastnili sa na ňom študenti prvého ročníka vysokej školy v počte 6. ľudí. V čase testovania bola aplikácia v stave tesne pred dokončením a obsahovala niekoľko chýb, o ktorých boli testujúci účastníci uvedomení. Testovanie prebiehalo v dvoch krokoch. V prvom kroku testovala aplikáciu skupina 3. používateľov, ktorí pracovali na zadaní popísanom v testovacom scenári nižšie. V druhom kroku bol použitý záťažový test, kedy všetci používatelia testovali aplikáciu súčasne, s veľkým počtom operácií za účelom zistenia stability synchronizácie editora.

5.1.1 Testovací scenár

Zadaním testovacieho scenára bolo *vytvoriť jednoduchý grafický návrh webovej aplikácie* vo forme blogu. Grafický návrh musel pozostávať z 3. podstránok.

1. *Hlavná stránka*, obsahujúca zoznam náhľadov jednotlivých článkov blogu s nadpisom, dátumom, obrázkom, autorom a tlačidlami stránkovania zoznamu.
2. *Stránka konkrétneho článku* s nadpisom, úvodným obrázkom a ľubovoľným obsahom.
3. *Profil autora blogu* s jeho aktivitou.

5.1.2 Výsledky testovania

Pri testovaní sa neprejavili žiadne zásadné nedostatky alebo chyby aplikácie okrem tých, na ktoré boli účastníci upozornení. V jednom prípade sa nepodarilo načítať prostredie grafického editora, čo bolo spôsobené zastaralou verziou internetového prehliadača.

Zúčastnená vzorka používateľov bola po testovaní vyzvaná na vyplnenie krátkeho dotazníka, kde bolo potrebné zhodnotiť prácu s editorom a popísať jeho prípadné nedostatky.

Intuitívnosť ovládania zhodnotili používatelia pozitívne. Synchronizácia aplikácií editorov jednotlivých používateľov prebiehala taktiež bez problémov. Za najväčší nedostatok bolo popísané chýbajúce tlačidlo na návrat vykonanej operácie (undo). Ostatné nedostatky boli po testovaní zapracované alebo opravené.

5.2 Návrhy na vylepšenie a budúcu prácu

Ako sa prejavilo pri testovaní, najväčším nedostatkom editora je návrat vykonaných zmien používateľom. Ide však o veľmi kľúčovú funkcionalitu, ktorú pri aktuálnom spôsobe implementácie nie je možné jednoducho zapracovať a vyžadovala by si dlhší čas potrebný na zapracovanie. Preto je to jedno z možných budúcich vylepšení.

Ďalším prípadným vylepšením je lepšie rozčlenenie zdrojových súborov do samostatných tried. To by zlepšilo prehľadnosť zdrojového kódu a jeho jed-

noduchšiu rozšíriteľnosť.

Knižnica Fabric poskytuje funkcionalitu na aplikovanie grafických filtrov pre objekty typu obrázok. Zapracovanie tejto funkcionality do grafického editora by taktiež značne zvýšilo jeho použiteľnosť.

6

Záver

Neustálym rozvojom webových technológií narastá aj dopyt ľudí po informáciách, ktoré je možné nájsť na internete. Pre ich jednoduché a rýchle sprístupňovanie širokej verejnosti, vzniká množstvo webových aplikácií slúžiacich na vytváranie a spravovanie online dokumentov. Jedným z najpoužívanejších systémov na ich tvorbu je MediaWiki. Obsah často pozostáva nie len z textu, ale taktiež z obrázkov a iných multimediálnych súborov. Tie je možné vytvárať či už pomocou počítačových, alebo formou webových aplikácií dostupných na internete a následne ich vložiť do dokumentu. Značným uľahčením tohoto

procesu by bola možnosť kreslenia obrázkov súvisiacich so spracovávanými informáciami priamo v systéme MediaWiki. Existujú rozšírenia, ktoré to v čase písania tejto diplomovej práce čiastočne umožňujú, sú však závislé od aplikácií tretích strán. Integrácia grafického editora priamo do systému MediaWiki je preto vítanou vlastnosťou, ktorá môže značne uľahčiť vytváranie obsahu dokumentov.

V našej diplomovej práci sme popísali popísali východiská pre návrh a vývoj kolaboratívneho grafického editora – pojem počítačovej grafiky a princípy kolaboratívnej práce viacerých používateľov na jednom dokumente. Urobili sme analýzu potrieb a vytvorili sme prehľad technológií potrebných na implementáciu grafického editora obrázkov pomocou webovej aplikácie, integrovanej do prostredia systému MediaWiki. Naprogramovali sme komunikačný protokol na synchronizáciu grafického editora pre viacerých používateľov spoločne pracujúcich na jednom dokumente. Vďaka tomu vidí každý používateľ zmeny vykonávané inými spolupracujúcimi používateľmi v reálnom čase. Implementované riešenie bolo integrované do MediaWiki systému používaného na adrese `https://wiki.matfyz.sk`. Serverovú časť sme nasadili pomocou balíčkovacieho systému Docker na server matfyz.sk. Z vykonaného testovania vyplynulo že ovládanie grafického editora je dostatočne intuitívne a poskytované funkcie sú dostačujúce pre kolaboratívne kreslenie a úpravu obrázkov počítačovej grafiky.

Zoznam obrázkov

Zoznam tabuliek

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. Fu-

kushi. 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.

Príloha A - obsah elektronického média

- súbor s textom diplomovej práce vo formáte pdf,
- priečinok **obrázky**, v ktorom sú všetky obrázky použité v texte diplomovej práce,
- priečinok **zdrojové súbory rozšírenie**, obsahuje implementáciu rozšírenia pre systém MediaWiki,
- priečinok **zdrojové súbory server**, obsahuje implementáciu synchronizačného servera