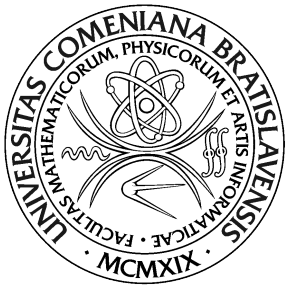
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

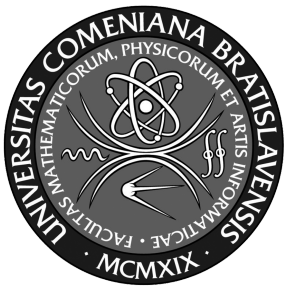


KLABORATÍVNY GRAFICKÝ EDITOR PRE MEDIAWIKI

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 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 napríklad

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 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 najpodstatnejšie.

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 });
```

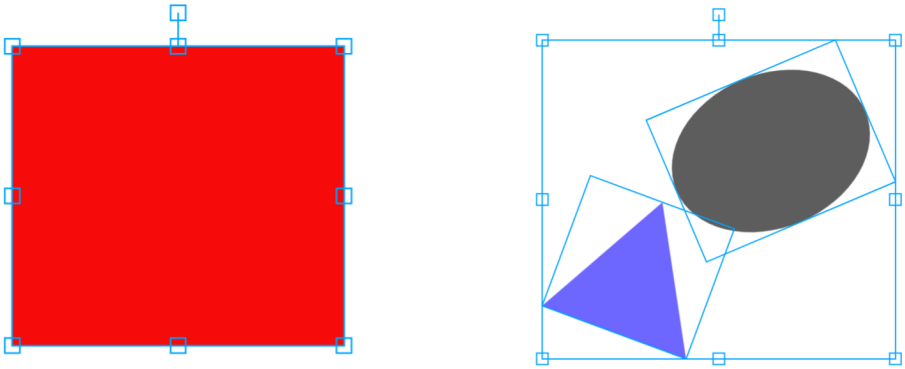
Zdrojový kód 2.1: Vytvorenie objektu typu obdĺžnik pomocou knižnice FabriJS a zmena jeho základných vlastností

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



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

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

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-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
- *serverovú 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 Me-

diaWiki 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

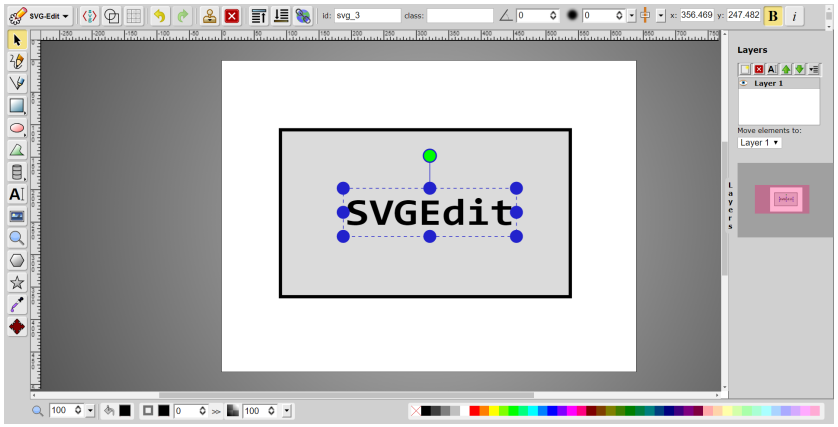
V nasledujúcej kapitole opíšeme existujúce riešenia, zameriame sa na ich výhody a prípadné nevýhody.

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.



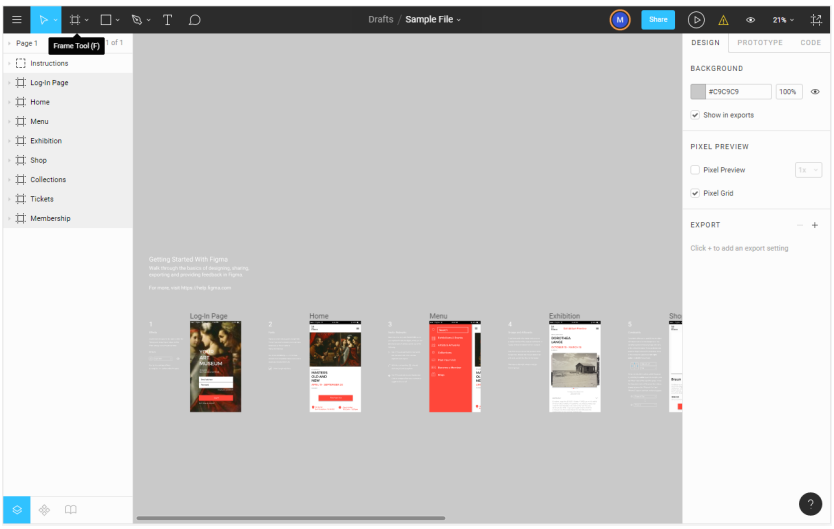
Obr. 3.1: Prostredie editora SVG-edit

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

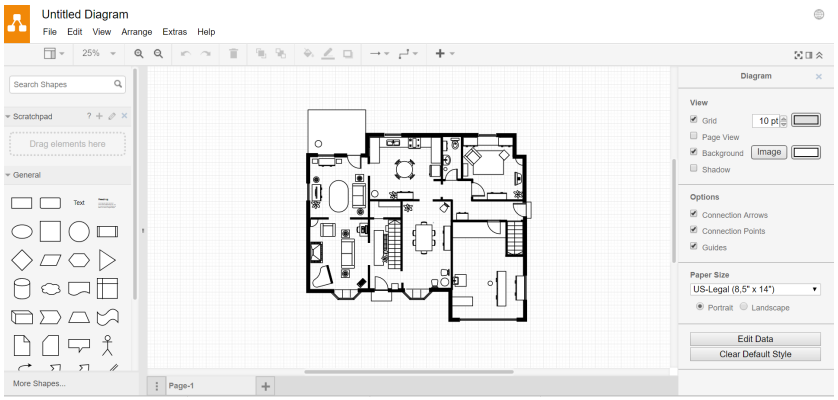
Draw.IO je online webová aplikácia slúžiaca na kolaboratívnu tvorbu vektorovej grafiky. Využitie tejto aplikácie je možné nájsť najmä pri tvorbe:

- *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.

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 Návrh riešenia

Riešenie diplomovej práce bude pozostávať z dvoch hlavných častí. Prvo 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.

4.2.1 Požiadavky na 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.

Popis základných entít synchronizačného servera

Entita User

Základnou entitou bude objekt *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 Message

Editor bude poskytovať 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 bude možné odosielať takzvanú systémovú správu, ktorá bude určená pre všetkých používateľov.

Entita Object

Informácie o objektoch grafickej plochy budeme synchronizovať pomocou JavaScriptového objektu typu *Object*. Bude 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 Room

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 RoomManager

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.

Funkčné požiadavky synchronizačného servera

- **Pripojenie používateľa** - vytvorí sa objekt typu `User` a nastavia sa mu vlastnosti na základe zaslaného dopytu.
- **Overenie pripojeného používateľa** - po pripojení používateľa sa overí správnosť overovacieho tokenu. Ak overovací 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 je vytvorená, používateľ sa do nej zaradí automaticky a

táto informácia sa odošle všetkým ostatným používateľom nachádzajúcim sa v miestnosti.

- **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.

  Modifikáciou existujúceho grafického objektu editora bude taktiež odoslaný dopyt o tejto akcii synchronizačnému server. 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 vymazaní tohoto objektu.

  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.

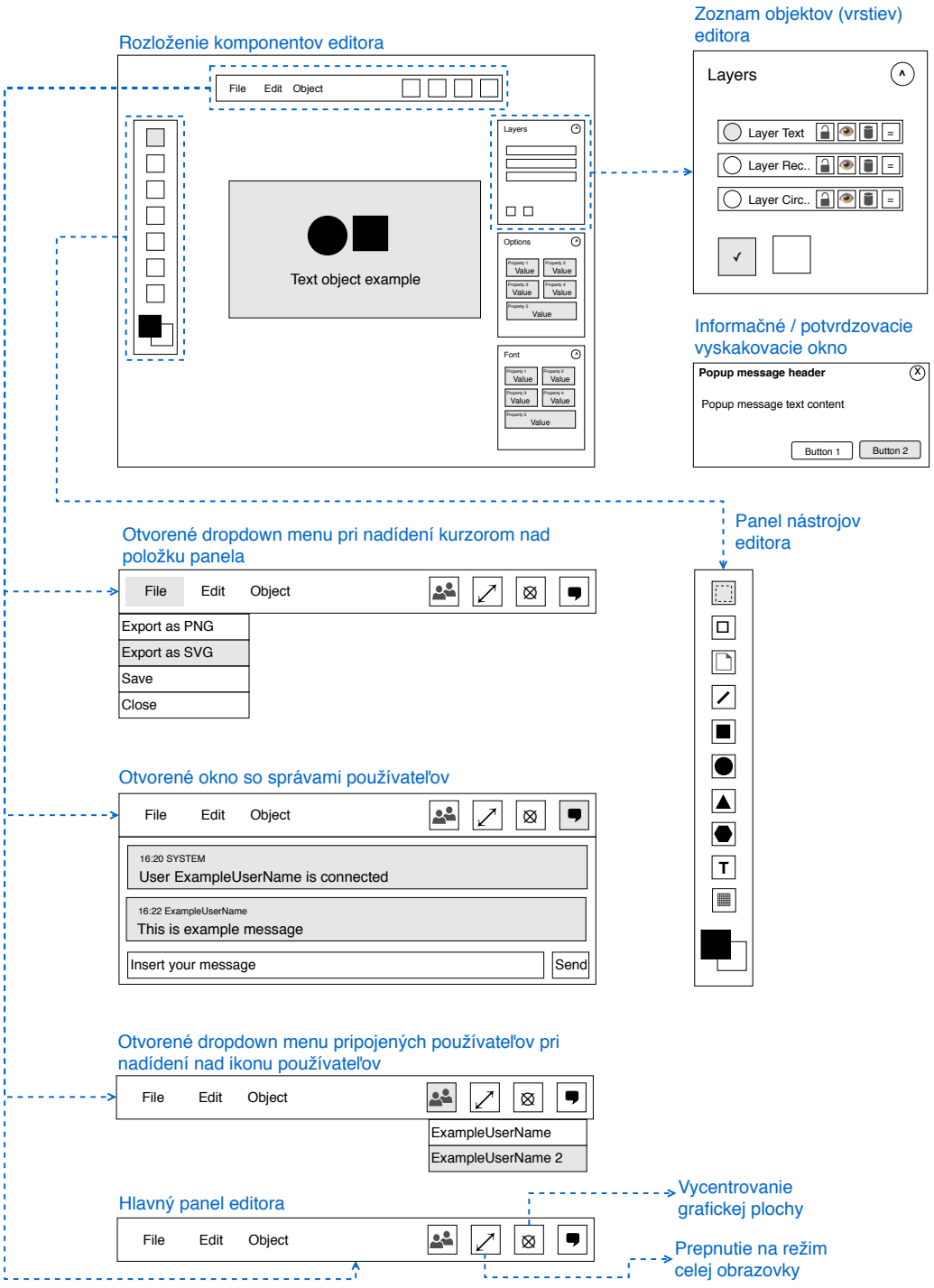
- **Odpojenie používateľa** - pri tejto akcií je potrebné vykonať niekoľko úkonov. Najskôr musia byť odomknuté všetky objekty, ktoré boli uzamknuté týmto používateľom a následne sa odstráni z miestnosti. V prípade že miestnosť zostane prázdna, vymaže sa. Inak je odoslaná informácia o odpojení používateľa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

- **Odstránenie miestnosti** - ako sme spomenuli vyššie, miestnosť bude automaticky odstraňovaná pri odchode posledného používateľa.

4.2.2 Editor obrázkov

Grafický editor bude implementovaný formou MediaWiki rozšírenia, naprogramovaného primárne v jazyku JavaSrcipt, s použitím frameworku AngularJS. Na komunikáciu so synchronizačným serverom použijeme JavaScriptovú knižnicu Socket.IO, konkrétne jej klientskú verziu.

Funkčné požiadavky grafického editora



Obr. 4.1: Návrh rozloženia komponentov používateľského prostredia editora

Integrácia do sytému MediaWiki

5

Implementácia

...

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. 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.