VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Virtuální sdílená tabule

Whiteboard Online Sharing Tool

Adam Šárek

Bakalářská práce

Vedoucí práce: Ing. Svatopluk Štolfa, Ph.D.

Ostrava, 2021

Abstrakt

Cílem této práce je vytvoření moderní, responzivní webové aplikace sdílené virtuální tabule. Práce se zaměřuje na optimalizaci pro plynulý chod aplikace a umožňuje tak spolupráci více uživatelů. Důraz je kladen také na podporu více typů zařízení. Práce se inspiruje několika existujícími weby, které dále rozšiřuje. Aplikace nabízí jednoduché nástroje, které umožňují její využití v širokém spektru oblastí. Každá tabule má dva základní režimy přístupu (úpravy a zobrazení), z nichž každý má svůj vlastní odkaz. Výsledkem práce je funkční řešení pro vytváření a sdílení virtuálních tabulí.

Klíčová slova

API; Canvas API; Fetch API; WebSockets; MySQL; JSON; Web workers; HTML; JavaScript; PHP

Abstract

The goal of this work is to create a modern, responsive web application for a shared virtual whiteboard. The work focuses on optimization for the smooth running of the application and thus allows the cooperation of multiple users. Emphasis is also placed on supporting multiple types of devices. The work is inspired by several existing sites, which it further extends. The application offers simple tools that allow its use in a wide range of areas. Each whiteboard has two basic access modes (editing and viewing), each with its own link. The result is a functional solution for creating and sharing virtual whiteboards.

Keywords

API; Canvas API; Fetch API; WebSockets; MySQL; JSON; Web workers; HTML; JavaScript; PHP

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
JSON	–	JavaScript Object Notation
HTML	–	Hyper Text Markup Language
DOM	–	Document Object Model
URL	–	Uniform Resource Locator
PHP	–	PHP: Hypertext Preprocessor

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Motivací práce bylo vytvořit moderní responzivní webovou aplikaci pro sdílení virtuální tabule, která bude umožňovat spolupráci více uživatelů na různých typech zařízení. Virtuální tabule slouží k zaznamenávání a sdílení návrhů a ilustrací pomocí jednoduchých nástrojů, díky kterým je možné ji využít v práci, škole či v osobním životě. Samotný obsah tabule je průběžně synchronizován, takže jsou změny jednotlivých uživatelů vidět přímo bez nutnosti stránku aktualizovat. Zároveň jsou také veškeré úpravy ukládány do databáze, která slouží jako trvalé uložiště dat a také záloha při případném výpadku serveru či spojení.

První kapitola se věnuje výběru jednotlivých stávajících řešení nástrojů pro sdílení virtuální tabule. Kapitola uvádí nástroje a technologie vybraných webů a porovnává je s vypracovaným řešením.

Druhá kapitola se zaměřuje na rozbor požadavků, analýzu problémů a možných řešení a také na architekturu systému. Kapitola přibližuje vnitřní strukturu systému na kterém je celé řešení postaveno.

Třetí kapitola pak pojednává o implementaci samotného řešení a jeho jednotlivých součástí. Nejprve je zmíněno vykreslování, jeho optimalizace pomocí vláken a také zpracování barev v rámci barevných motivů. Dále se kapitola věnuje výměně dat mezi klientských zařízením a serverem pomocí WebSockets. Následně je popsána správa dat, která na lokální či serverové úrovni zajišťuje využití dat pro konkrétní potřeby uživatele. Zmíněno je mimo jiné využití MySQL databáze, LocalStorage a Fetch API a také jednotlivé možnosti exportování tabule do rastrové či nativní podoby. V poslední řadě se kapitola věnuje uživatelskému rozhraní z pohledu responzivity, podpory více jazyků včetně jejich zakomponování do sdílení tabule a také jednotlivým stránkám a jejich podobě ve finálním řešení.

Kapitola 2

Inspirace řešení

Virtuální sdílená tabule je natolik užitečný nástroj, že je na internetu již spousta různých variant řešení. Odlišit se tedy není snadné, a proto je práce inspirována několika již existujícími weby, které v rámci funkcí rozšiřuje o určité nové nápady.

2.1 Analýza vybraných webů

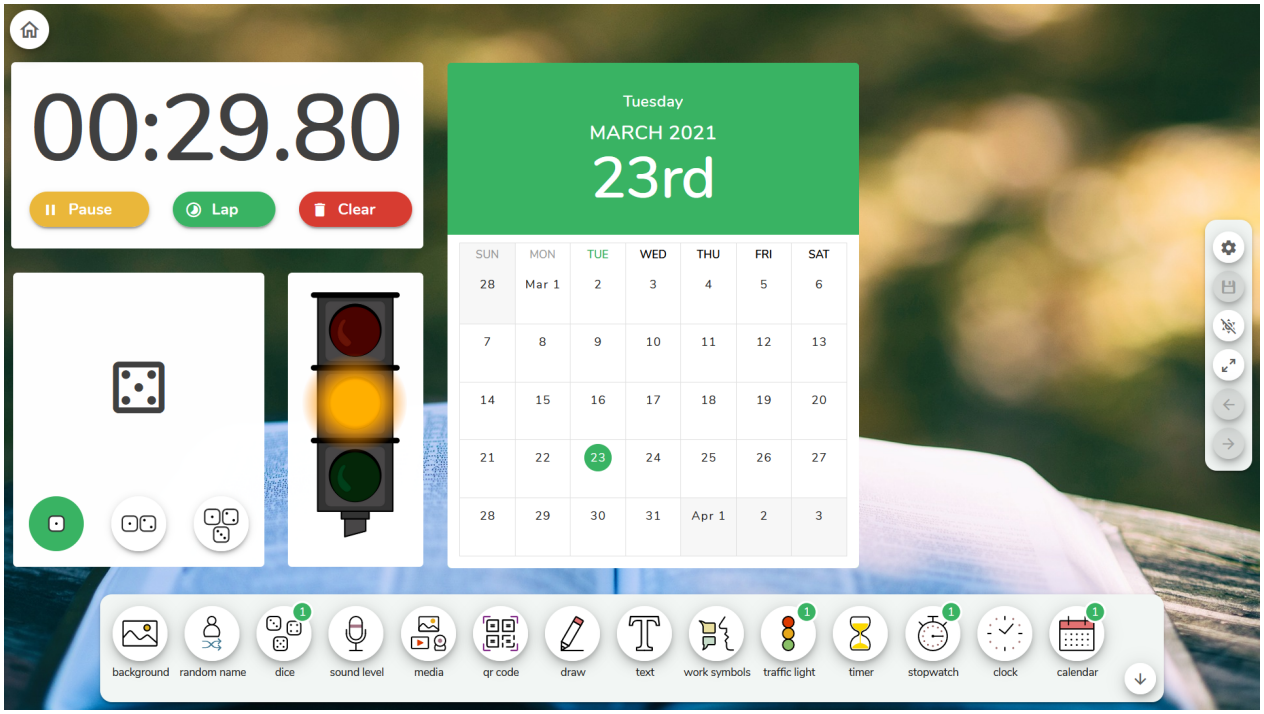
Weby použité pro inspiraci bylo nejdříve potřeba analyzovat jak z hlediska uživatelských nástrojů a rozhraní, tak z hlediska použitých technologií pro dosažení výsledné funkčnosti. K identifikaci použitých technologií bylo využito standardně zabudovaných vývojářských nástrojů webového prohlížeče, konkrétně v tomto případě prohlížeče Mozilla Firefox ve verzi 86. Pro inspiraci bylo vybráno celkově pět webů, které jsou níže vypsány od nejméně po nejvíce autorově subjektivně vnímané ideální řešení nástroje pro sdílení virtuální tabule.

2.2 Inspirace vybranými weby

2.2.1 Inspirace - classroomscreen.com

Jako první proběhla analýza webu classroomscreen.com. Tato stránka již na první pohled vypadá spíše jako sdílená obrazovka než jako sdílená tabule, což ostatně napovídá i její název. Obrazovka zobrazuje simulaci jednotlivých oken, které slouží jako kalendář, stopky, generátor hodu kostkou a další. Jednotlivé okna mají velmi odlišnou funkčnost a působí tedy podobným dojmem jako počítačové aplikace, což umožňuje tento web využít pro velmi různorodé účely. Rozhraní obsahuje několik základních tlačítek nástrojů pro vytvoření již zmíněných instancí oken. Na pozadí je od prvního spuštění vykreslen náhodný obrázek většinou s motivem přírody či nějaké kulturní památky, který je možné kdykoliv změnit v nastavení. Web má jasně vymezené okraje použitelné části plochy, které jsou dané velikostí uživatelova okna prohlížeče. Není tedy možné se pohybovat libovolně mimo

viditelnou oblast. Web není určen pro spolupráci více uživatelů a je tedy vhodný především pro menší projekty či prezentace. Jedinou technologií, která stojí za zmínku je zde Canvas API, která je použita k vykreslování grafického výstupu. Inspirací k bakalářské práci byl tento web pouze díky nástroje pro úpravu poznámek. Poznámka v bakalářské práci je stejně jako na tomto webu tvořena pomocí Canvasu překrytého HTML elementem `<div>` využívajícím atribut `contenteditable="true"`, který umožňuje úpravu vnitřního textu a díky čemuž tato poznámka působí celistvým dojmem.

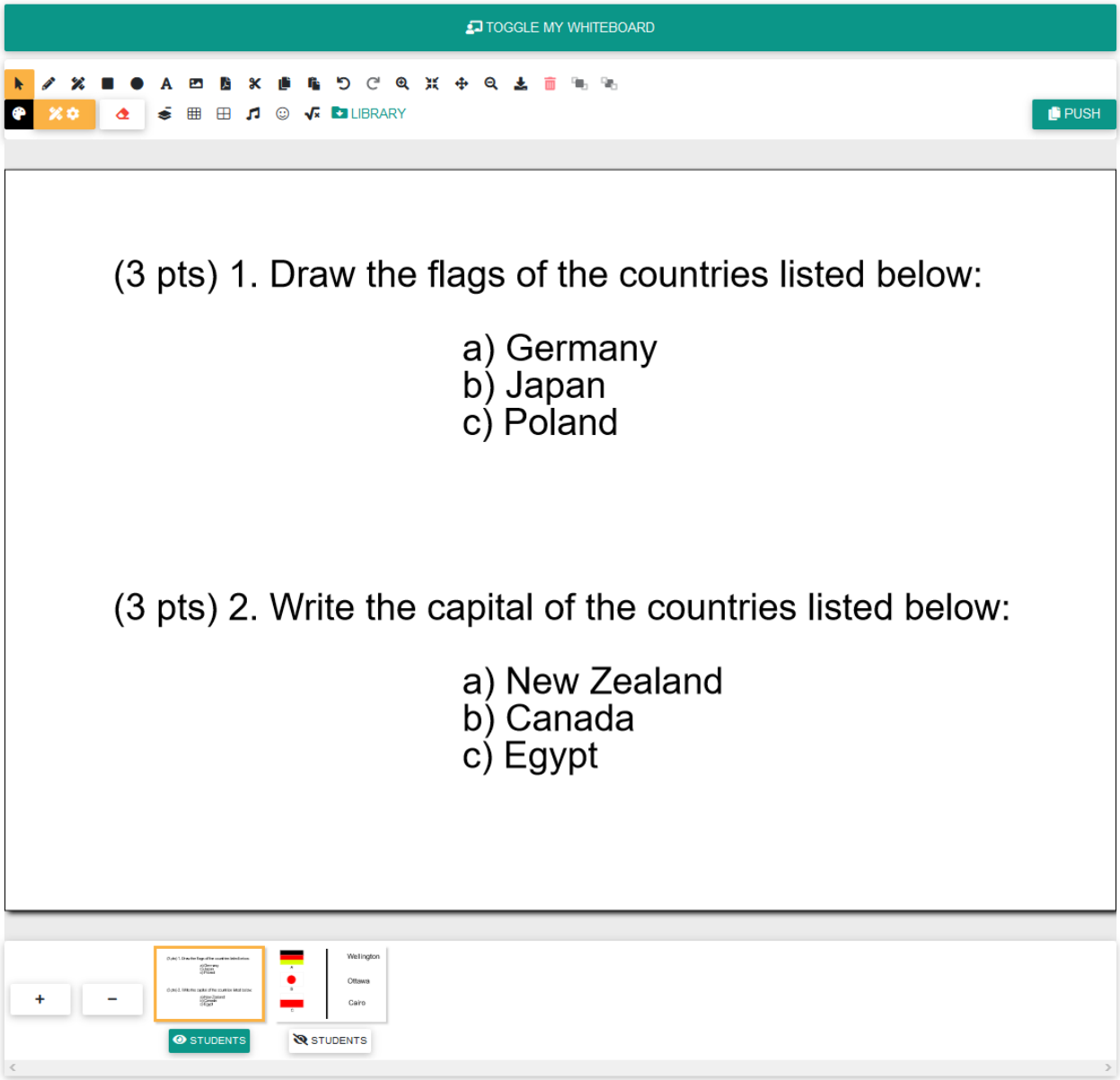


Obrázek 2.1: Ukázka webu classroomscreen.com

2.2.2 Inspirace - whiteboard.fi

Dále byla zanalyzována stránka whiteboard.fi. Tento web registrovaný pod finskou národní doménou je orientovaný především pro využití ve školním prostředí. Po vytvoření místnosti je zobrazeno plátno o předem dané velikosti, které se ostatním uživatelům během úpravy neaktualizuje. Pro odeslání aktuálního stavu plátna je potřeba stisknout tlačítko Push. V rámci místnosti je možné vytvářet více pláten a libovolně mezi nimi přepínat a upravovat je. Web obsahuje základní nástroje jako kreslení, přidání textu, obrázku či speciální nástroje jako notový zápis a další a jak již bylo zmíněno, jeho využití je tedy především ve škole. Stránka rovněž využívá Canvas API pro vykreslení grafiky, ale navíc používá protokol WebSocket, díky kterému podporuje spolupráci více uživatelů.

Hlavní inspirací u tohoto webu bylo přepínání hustoty mřížky na pozadí plátna, díky které je možné lépe umístit či vyměřit určité objekty.



Obrázek 2.2: Ukázka webu whiteboard.fi

2.2.3 Inspirace - whiteboardfox.com

Třetím webem pro inspiraci byl whiteboardfox.com. Tento web oproti předchozím mnohem více funguje na myšlence virtuální sdílené tabule.

Velikost plátna tabule není omezena velikostí okna prohlížeče a veškeré úpravy tabule jsou automaticky distribuované všem uživatelům. Inspirace těmito dvěma vlastnostmi jsou ostatně hlavním důvodem, proč je zde tato stránka zmíněna.

Uživatelské rozhraní však nepůsobí příliš moderně, není responzivní a orientace v něm je relativně

složitá, jelikož většina nástrojů je schována pod tlačítkem nastavení, což pro mnoho uživatelů nemusí být intuitivní. Opět se zde objevuje využití Canvasu a také WebSocket, který zmíněný web používá ke sdílení dat a díky čemuž je tento web určen pro spolupráci více uživatelů.

2.2.4 Inspirace - collboard.com

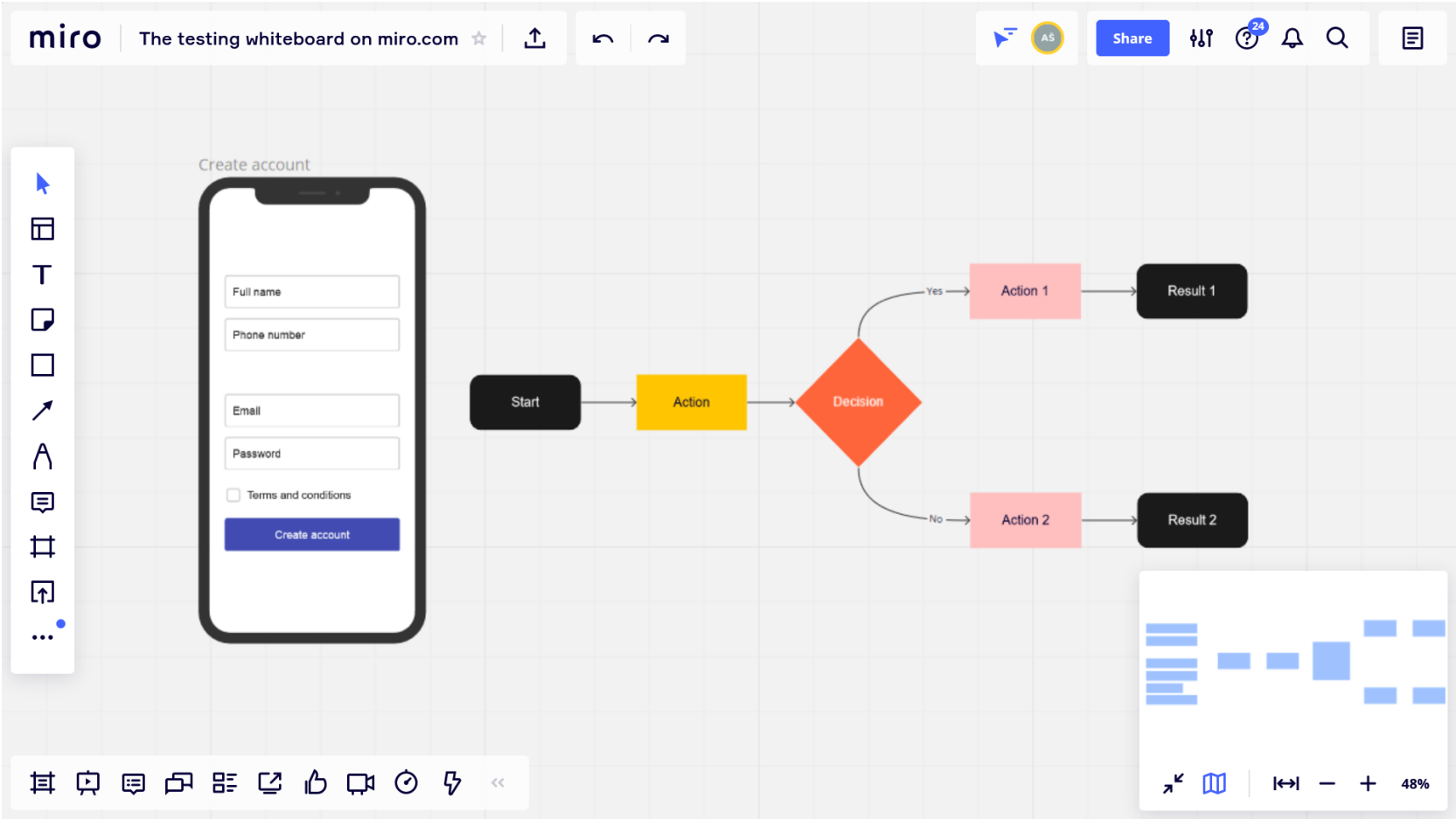
Předposledním vybraným řešením virtuální tabule je stránka collboard.com. Tato stránka vylepšuje nedostatky předchozího webu co se týče náročnosti používání uživatelského rozhraní a doplňuje jej o celou řadu možností úprav jednotlivých objektů, jako např. změna tloušťky čáry u kreslení či změna řezu písma u textu a další. Veškeré objekty lze libovolně vybírat a přesouvat či dále měnit. Web dále umožňuje přidávat různé pluginy, které využitelnost celého řešení ještě více rozšiřují. U tabule je možné zadat také její název, což se může zdát jako drobnost, nicméně tento detail může být užitečný pro identifikaci sdílené tabule např. v emailové pozvánce. Na místě je také tlačítko vrácení na počáteční pozici. Může se totiž lehce stát, že se uživatel při pohybu po tabuli ztratí. Použité technologie se oproti předchozímu řešení nijak neliší.

Tento web byl velmi výraznou inspirací pro finální řešení bakalářské práce, jelikož nabízí opravdu moderní, responzivní prostředí, plné užitečných nástrojů a velmi dobře promyšlených řešení případných uživatelských problémů. Jedná se tak o velmi užitečný nástroj pro sdílení virtuální tabule.

2.2.5 Inspirace - miro.com

Nejrozsáhlejší a nejpropracovanější stránkou z výběru je miro.com. Oproti collboard.com je tento web rozšířen o další spoustu nástrojů vhodných pro větší projekty. Poskytuje vlastní API pro tvorbu pluginů a také integraci s aplikacemi třetích stran jako Microsoft Teams či Google Disk. Nabízí funkci chatu, seznam poznámek, zobrazuje umístění jednotlivých uživatelů. Přidávat lze mimo základní nástroje také šablony, tedy např. myšlenkové mapy či vývojové diagramy viz. obrázek 2.3 a obsahuje také opravdu detailní konfiguraci jednotlivých objektů. V rámci technologií kromě Canvasu a WebSocket používá navíc Web worker. Web worker slouží k rozdělení operací do více vláken, čímž se sníží nároky na hlavní vlákno, které je dle specifikací povinné reagovat na uživatelský vstup. [1] Touto optimalizací je hlavní vlákno schopno dosáhnout vyššího výkonu, což ve výsledku znamená plynulejší běh celého webu.

Největší inspirací byl vizuální styl uživatelského rozhraní a také způsob sdílení tabule prostřednictvím emailu a rozdělení tabule do různých uživatelských režimů - úpravy a zobrazení.



Obrázek 2.3: Ukázka webu miro.com

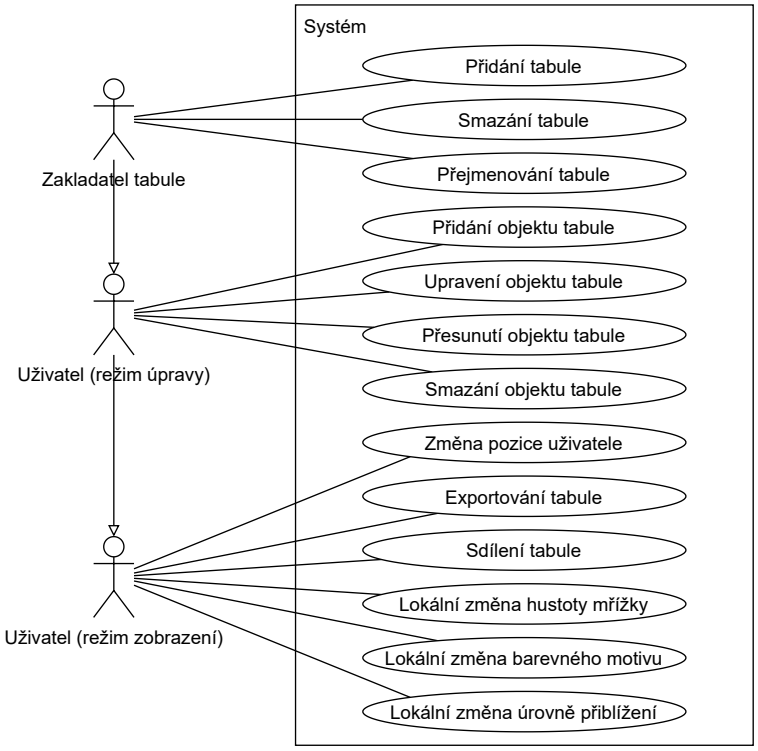
Kapitola 3

Požadavky, analýza a návrh

3.1 Požadavky

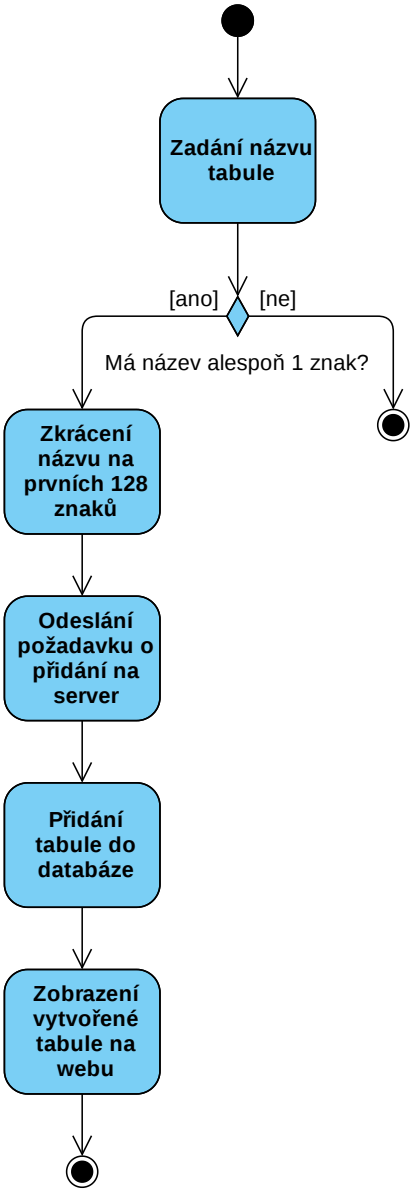
3.1.1 Funkční požadavky

3.1.1.1 Diagram případů užití



Obrázek 3.1: Diagram případů užití

3.1.2 Aktivitní diagram - přidání tabule

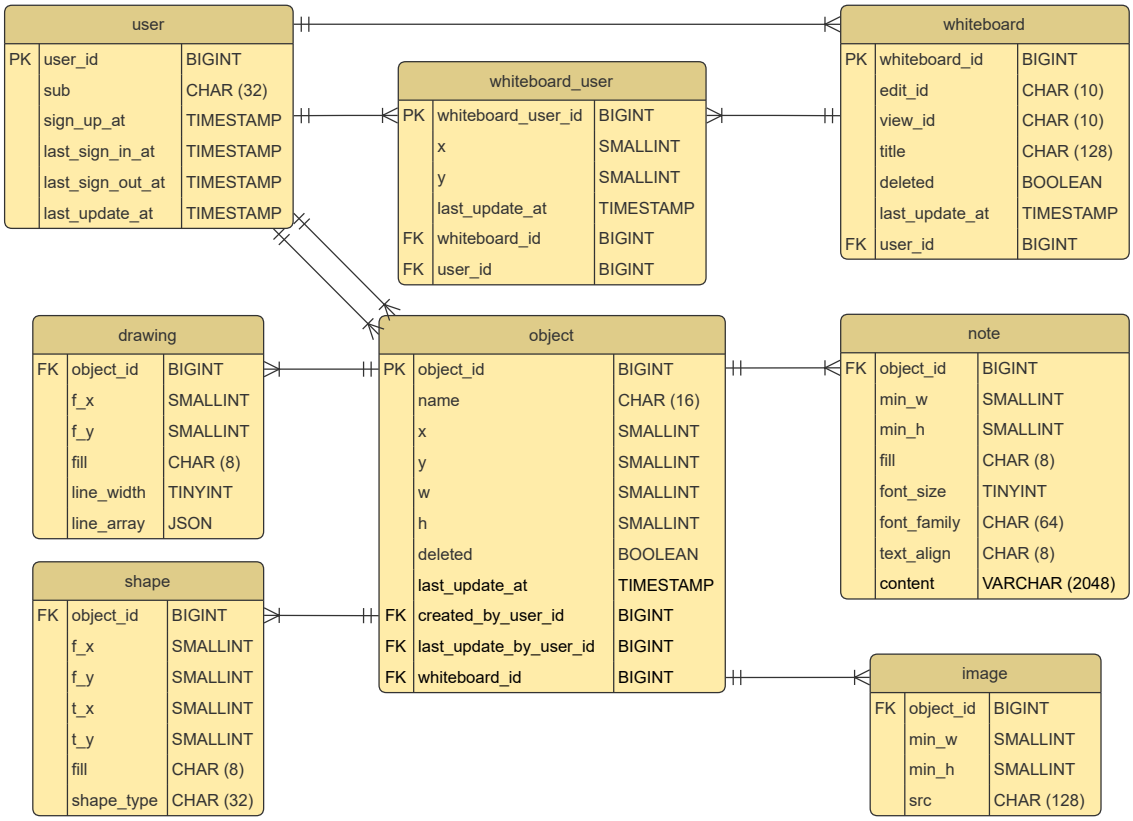


Obrázek 3.2: Aktivitní diagram - přidání tabule

3.2 Analýza

3.2.1 Datová analýza

3.2.1.1 Relační datový model databáze



Obrázek 3.3: Relační datový model databáze

3.2.2 Stavová analýza

Definujeme tyto stavy tabule:

- **Smazaná** – tabule, která byla uživatelem smazána
- **Nesmazaná** – tabule, která dosud nebyla uživatelem smazána

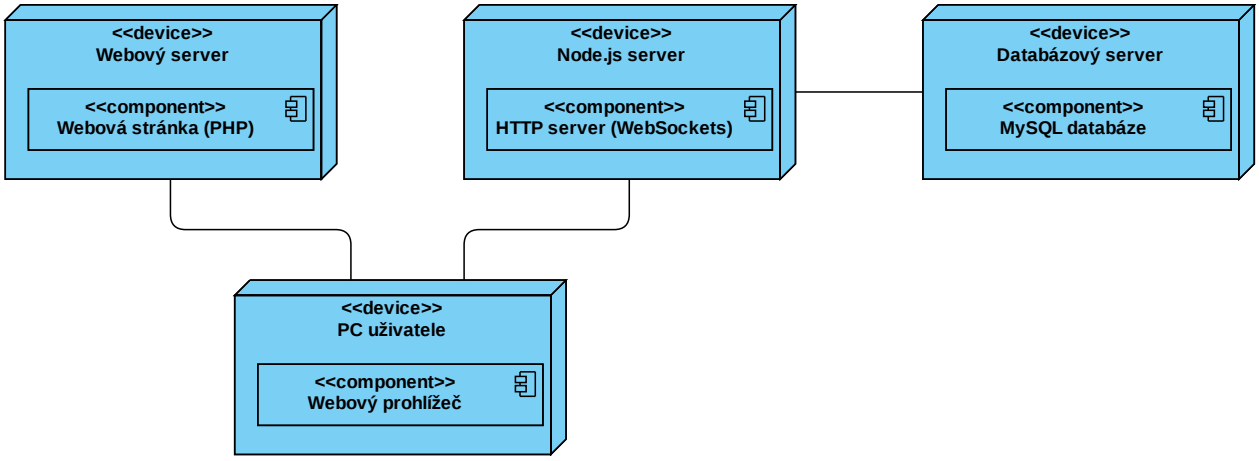
Definujeme tyto stavy objektu tabule:

- **Smazaný** – objekt tabule, který byl uživatelem smazán
- **Nesmazaný** – objekt tabule, který dosud nebyl uživatelem smazán

3.3 Návrh

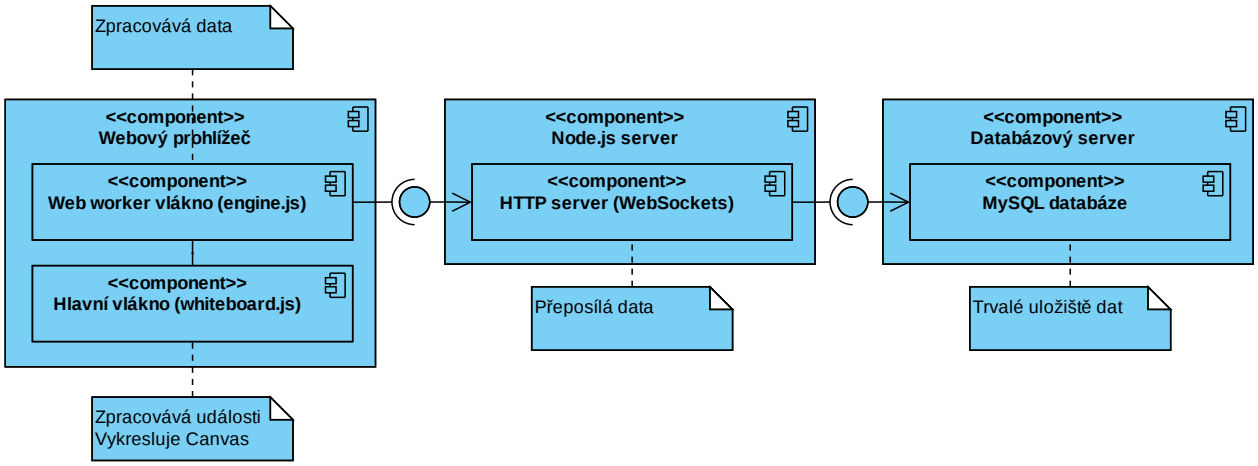
3.3.1 Architektura systému

3.3.1.1 Diagram nasazení



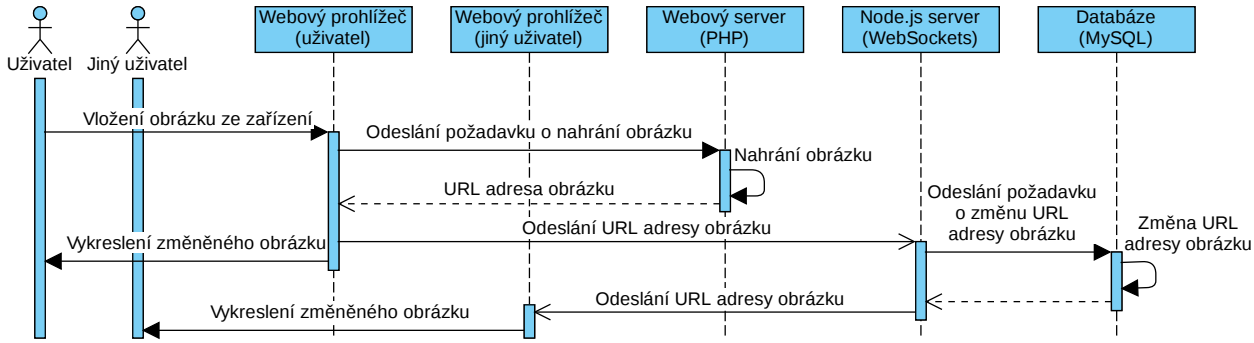
Obrázek 3.4: Diagram nasazení systému

3.3.1.2 Diagram komponent - datová komunikace



Obrázek 3.5: Diagram komponent - datová komunikace

3.3.2 Sekvenční diagram - změna obrázku objektu tabule



Obrázek 3.6: Sekvenční diagram - změna obrázku objektu tabule

Kapitola 4

Implementace

4.1 Vykreslování grafického výstupu

4.1.1 Zavedení Canvasu

Pro vykreslování grafického výstupu je využíván HTML element `<canvas>`. V JavaScriptu je objekt tohoto elementu možné získat pomocí DOM (objektový model dokumentu). [2] Získaný objekt `HTMLCanvasElement` je použit v rámci vlastní vytvořené třídy `WhiteboardCanvas`, která byla vytvořena za účelem optimálního vykreslování všech artefaktů tabule. Nad objektem `HTMLCanvasElement` je zavolána funkce `getContext("2d")`, která vrací instanci rozhraní `CanvasRenderingContext2D`. Tato instance je pak dále nastavena všem již přidaným objektům tabule, které díky ní mohou začít vykreslovat.

4.1.2 Vykreslovací cyklus

Vykreslovací cyklus probíhá v rámci již zmíněné třídy `WhiteboardCanvas`. Cyklus začíná v metodě `linkHtml(canvas)` po získání instance `CanvasRenderingContext2D` zavoláním funkce `redraw()`.

```
redraw() {
   if(this.redrawRequested) {
      this.clear();
      this.draw();
      this.redrawRequested = false;
   }

   requestAnimationFrame(this.redraw.bind(this));
}
```

Listing 4.1: Metoda redraw() třídy WhiteboardCanvas

Funkce začíná podmínkou, která kontroluje, zda je požadováno překreslení aktuálního stavu plátna. Požadavek je při spuštění nastaven na hodnotu `true` právě z důvodu, aby počáteční vykreslení proběhlo bez nutnosti požadavek dále nastavovat.
Pokud je podmínka splněna, dojde k vyčištění plátna a následnému vykreslení objektů přidaných funkcí `setObjects(objects)`. Dále je pak požadavek nastaven na `false`, jelikož po aktuálním překreslení již další nejsou potřeba, dokud nedojde k nějaké změně, v rámci které bude tento požadavek vyvolán.
Ať už podmínka byla nebo nebyla splněna, po jejím průchodu je, díky `requestAnimationFrame()`, funkce `redraw()` pokaždé znovu volána ve frekvenci odpovídající obnovovací frekvenci monitoru uživatele. Díky omezení frekvence nemůže dojít k vyšším nárokům stránky (např. v módu kreslení), než které je koncové zařízení schopno zvládnout.

4.1.2.1 Odeslání požadavku o vykreslení

Pokud dojde ke změně kterékoliv vlastnosti jakéhokoliv objektu tabule nebo se změní její barevný motiv je potřeba vyvolat požadavek o překreslení plátna skrze `requestRedraw()`.

```
requestRedraw() {
   this.redrawRequested = true;
}
```

Listing 4.2: Metoda requestRedraw() třídy WhiteboardCanvas

Nastavením proměnné `redrawRequested` na hodnotu `true` dochází v rámci vykreslovacího cyklu ke splnění podmínky, ve které následně dojde k překreslení plátna tabule.

4.1.3 Využití více vrstev pláten

V rámci optimalizace byly místo jednoho Canvasu vytvořeny dva, jeden pro mřížku na pozadí a druhý pro samotný obsah tabule. Důvod je jednoduchý. Plátno s mřížkou se aktualizuje méně často než hlavní vlákno. Pokud by tedy byla mřížka s hlavním obsahem v jednom canvasu, musela by se pokaždé znovu překreslit i tato mřížka a zbytečně by tak byl snížen výkon webu. Plátno s vlastním obsahem tabule má průhledné pozadí, aby byla mřížka na pozadí viditelná.

4.1.4 Využití vláken

Součástí optimalizace řešení je využití současné práce více vláken. JavaScript umožňuje kromě hlavního vlákna, které slouží především pro vykreslování a event handling, využít také další vlákna pomocí web workerů. Tyto workery nemají přístup ke všem objektům, ke kterým má přístup hlavní vlákno, takže je nutné určité potřebné data poslat jiným způsobem. To však není problém, jelikož workery mají přímo zabudovaný event `onmessage` pro příjem zpráv a funkci `send()` pro odesílání zpráv.

Vedlejší vlákna jsou vhodná především pro větší výpočty či práci s daty, které nutně nemusí být v hlavním vláknu. Čím méně instrukcí musí hlavní vlákno řešit, tím lépe zvládá činnosti, které je povinné zajišťovat. Výsledkem je tedy plynulejší vykreslování a rychlejší reakce na jednotlivé události.

V této práci je využíváno pouze jedno vedlejší vlákno, jelikož je pro optimální funkčnost dostačující, nicméně je možné, že s pomocí více vedlejších vláken by šlo určité operace provádět současně a tedy efektivněji.

4.1.4.1 Výpočet souřadnic

Nejvíce výpočtů v práci souvisí s výpočtem souřadnic a je tedy vhodné tyto výpočty provádět v rámci vedlejšího vlákna. Souřadnice jsou důležitou součástí zaznamenávání a zobrazování objektů tabule. Práce k výpočtu souřadnic využívá několik faktorů, které ovlivňují, kde se zrovna uživatel nachází, jaká část tabule je viditelná a zda je vůbec samotný výpočet potřeba.

Výchozím referenčním bodem tabule je bod [0;0]. Tento bod označuje místo, kde se objevují noví uživatelé při prvním spuštění tabule a nachází se uprostřed uživatelovy obrazovky, ať už na počítači či mobilním zařízení.

Výhodou umístění bodu na střed je to, že je možné uprostřed vytvořit obsah, který má být pro nové uživatele na první pohled viditelný bez nutnosti se přepínat mezi záchytnými body tabule či se po tabuli přesunovat ručně. Je to také místo, kde bude chtít začít většina uživatelů pracovat, takže bude výsledek viditelný ihned jednoduše pro všechny.

Dalším důvodem výběru je také to, že pokud by byl výchozí bod sice uprostřed tabule, ale tabule by souřadnice počítala pouze v kladných hodnotách, tak by objekty okolo středu měly zbytečně celkově vyšší hodnoty souřadnic. Může se to zdát jako zbytečnost, nicméně při vyšším počtu tabulí, uživatelů a jejich dat toto může být důležitým opatřením pro udržení nižší datové náročnosti jak pro provozující server, tak pro internetové připojení klienta. K tomuto přispívá nejen bod [0;0], ale také fakt, že veškeré souřadnice tabule jsou zaznamenávány v celých číslech.

Dalším rozhodnutím pro nižší datový přenos je omezení maximální velikosti tabule. Stávající maximální velikost tabule byla stanovena na rozlišení 32K z toho důvodu, že se dnes ještě nejedná o běžně dosažitelné rozlišení a tedy není snadné jednoduše vyčerpat celý prostor tabule. Zároveň se však jedná o stále relativně nízké rozlišení pro exportování tabule do obrázku.

4.1.4.1.1 Faktor pozice uživatele

Aby bylo možné vykreslit mřížku či libovolný objekt tabule, je nutné znát aktuální pozici uživatele. Při prvním spuštění je tato pozice v bodě [0;0] nicméně díky možnosti se po tabuli pohybovat se tato hodnota v průběhu používání mění. Pozice uživatele je vždy dána souřadnicemi bodu tabule uprostřed vykreslovaného plátna. Při každé změně pozice je nutné souřadnice všech objektů přepočítat a znovu vykreslit aktualizovaný obsah.

4.1.4.1.2 Faktor přiblížení

Nejen pozice uživatele, ale i všech artefaktů tabule je ovlivněna také mírou přiblížení. Přiblížit je možné až na úroveň 300 % a oddálit pak na 30 % standardní míry zobrazení. S každým přiblížením či oddálením je rovněž nutné znovu přepočítat a vykreslit změny.

4.1.4.1.3 Faktor velikosti obrazovky

Velikost Canvasu ovlivňuje souřadnice událostí pracujících s pozicí kurzoru či prstu na obrazovce. Vždy je nutné od aktuální pozice ukazatele odečíst polovinu velikosti Canvasu, čímž dojde k vycentrování ukazatele na střed a k dalšímu využití momentálních souřadnic uživatele.

Canvas také svojí velikostí udává viditelnou oblast tabule, kterou je nutné vykreslit, čímž je možné spoustu objektů tabule vynechat a nutné výpočty provádět pouze u zobrazených objektů. Aby byl objekt považován za viditelný, je nutné aby byl alespoň jeden roh objektu ve viditelné oblasti. Možným vylepšením současného stavu projektu je doplnění této podmínky o kontrolu skutečné vykreslené oblasti objektu. Aktuálně se totiž může stát, že pokud např. objekt volného kreslení v daném rohu neobsahuje žádnou čáru, nicméně tento roh je součástí viditelné oblasti, tak se s tímto objektem počítá jako s viditelným a je nutné u něj vypočítat souřadnice i když ve výsledku na tabuli není vidět.

4.1.4.2 OffscreenCanvas

OffscreenCanvas je experimentální technologie, která nově přináší vykreslování Canvasu pomocí Web workeru. Kontext plátna není přímo závislý na DOM, což umožňuje jeho použití k vykreslování mimo hlavní vlákno. [3] Přenesením renderovacích příkazů na vedlejší vlákno se eliminuje sekání způsobené vyčkáváním hlavního vlákna na dokončení vykreslovacích operací, které při plném využití výkonu znemožňují další interakci s webem.

Tato technologie je s použitím 2D kontextu zatím mezi nejrozšířenějšími prohlížeči podporovaná pouze prohlížeči postavenými na projektu Chromium a dále prohlížečem Samsung Internet, které globálně používá okolo 70% uživatelů. [4]

Jelikož podpora nezahrnuje prohlížeč Mozilla Firefox a další široce zastoupené značky, tak bylo po dohodě s vedoucím práce rozhodnuto tuto technologii neimplementovat a pouze její existenci zmínit v rámci tohoto dokumentu.

Podpora technologie OffscreenCanvas je však již nyní mezi vývojáři velmi žádaná a bude důležitou součástí webů využívajících Canvas.

4.1.5 Vlastní třídy pro vykreslování

Standardní metody rozhraní CanvasRenderingContext2D jsou plně postačující pro vykreslení několika jednoduchých tvarů. Avšak pro využití ve větším počtu je v rámci zachování efektivity příkazů

a přehlednosti zdrojového kódu lepší tyto funkce sjednotit. Ke sjednocení byly použity třídy, které JavaScript podporuje od verze standardu ECMAScript 6.

Seznam níže uvádí třídy, které byly v bakalářské práci vytvořeny za účelem zjednodušení či zefektivnění vykreslování jednotlivých artefaktů tabule:

WhiteboardCanvasPolyline – Třída slouží k vykreslování lomené čáry. Tento typ čáry je využit k vykreslení mřížky a také v rámci nástroje kreslení, kde jedna lomená čára vyobrazuje jeden souvislý tah.

V metodě `draw()` třída shlukuje příkazy `CanvasRenderingContext2D.moveTo()` a `CanvasRenderingContext2D.lineTo()` v rámci cyklu, který prochází zadané pole počátečních a koncových souřadnic čar. Cyklus čáry přidává do jedné cesty uvedené voláním `CanvasRenderingContext2D.beginPath()`, což zajišťuje efektivnější vykreslování, než kdyby byla každá čára vykreslována samostatně. [5]

WhiteboardCanvasRoundedRectangle – Třída slouží k vykreslování zaobleného obdélníku, který je použit jako pozadí poznámky. Výběr zaobleného obdélníku místo hranatého je čistě stylizační rozhodnutí.

Třída v metodě `draw()` obsahuje, po volání `CanvasRenderingContext2D.beginPath()`, sekvenci 4 příkazů `CanvasRenderingContext2D.arc()`, které tvoří kruhové oblouky využité na zaoblené hrany obdélníku. [6] Oblouky jsou následně spojeny zavoláním funkce `CanvasRenderingContext2D.closePath()`. [7] Spojení oblouků pomocí příkazu `closePath()` je efektivnější oproti samostatnému vykreslování zbylých čar na každé straně obdélniku. Celá nově vzniklá oblast je, dle nastavení v `CanvasRenderingContext2D.fillStyle`, vyplněna metodou `CanvasRenderingContext2D.fill()`. [8]

WhiteboardCanvasText – Třída slouží k vykreslování textu poznámky.

Canvas neumožňuje vykreslovat text na více řádků, takže je zdrojový HTML kód zpracován a rozdělen na jednotlivé řádky. Zarovnání dále nabízí pouze od daného bodu, takže je nutné pro zarovnání doprava a na střed tento výchozí bod posunout doprava resp. na střed poznámky. Jednotlivé řádky textu jsou poté zobrazeny pomocí metody `CanvasRenderingContext2D.fillText()`.

WhiteboardCanvasNote – Třída slouží k vykreslování poznámky. Obsahuje objekty tříd `WhiteboardCanvasRoundedRectangle` a `WhiteboardCanvasText`, díky kterým je možné poznámku vykreslit a obsahuje metody, jako např. `setPosition(x, y)` nebo `setSize(w, h)`, které oběma objektům nastaví potřebné vlastnosti. Cílem bylo zapouzdřit jednotlivé části poznámky do jedné třídy, pro snadnější úpravu libovolných parametrů.

ImageWrapper – Třída slouží k indikaci stavu načtení obrázku a umožňuje nastavit vlastní události `onerror(fn)` a `onload(fn)`. Dále obsahuje metodu `isReady()`, která říká, zda je obrázek

načten (platí i při chybě při načítání) a `canDraw()`, která vrací `true` pouze v případě, pokud byl obrázek načten v pořádku.

WhiteboardCanvasImage – Třída slouží k vykreslování obrázku. Využívá objekt třídy `ImageWrapper`, jehož metody `isReady()` a `canDraw()` určí aktuální stav obrázku, který je pak následně zobrazen pomocí `CanvasRenderingContext2D.drawImage()`. Obrázek může nabývat následujících stavů:

- **Obrázek se načítá** – je zobrazeno tmavě šedé pozadí s neutrálním piktogramem obrázku
- **Obrázek se nepodařilo načíst** – je zobrazeno tmavě červené pozadí s piktogramem rozbitého obrázku
- **Obrázek byl úspěšně načten** – je zobrazen samotný obrázek

WhiteboardCanvasShape – Třída slouží k vykreslování geometrického tvaru.

Tabulka 4.1: Výpis tvarů a jejich volaných funkcí CanvasRenderingContext2D

Tvar	Volané funkce CanvasRenderingContext2D
Obdélník	`beginPath()`, `rect()`, `fill()`
Elipsa	`beginPath()`, `ellipse()`, `fill()`
Pravoúhlý trojúhelník	`beginPath()`, `moveTo()`, `lineTo()`, `closePath()`, `fill()`
Rovnoramenný trojúhelník	`beginPath()`, `moveTo()`, `lineTo()`, `closePath()`, `fill()`
Kosočtverec	`beginPath()`, `moveTo()`, `lineTo()`, `closePath()`, `fill()`

Tvary mají společné vlastnosti jako pozici, rozměry, pozadí a liší se pouze samotným typem tvaru. Jejich vykreslovací funkce tedy mohou být jednoduše rozděleny na základě typu tvaru přímo v rámci jedné třídy. Tímto se eliminuje duplikování totožných metod při vytváření samostatných tříd pro každý tvar zvlášť.

4.1.6 Barevné motivy tabule

Dnešním trendem moderních webů a aplikací je podpora tmavého motivu. Zatímco světlý motiv je vhodné využít především za ostřejšího denního světla, tak tmavý se hodí spíše v šeru večerních či nočních hodin, kdy je pro oči šetrnější.

Právě s denní dobou počítá ve výchozím stavu také vypracované řešení, kdy vždy v 18:00 se dle uživatelova lokálního času přepne web do tmavého motivu a ráno během 6:00 se zase opět vrátí do světlého motivu. Toto automatické přepínání jde samozřejmě kdykoli tlačítkem přepnout přímo na výběr konkrétního motivu či vrátit zpátky na automatickou volbu.

4.1.6.1 Barevný kontrast

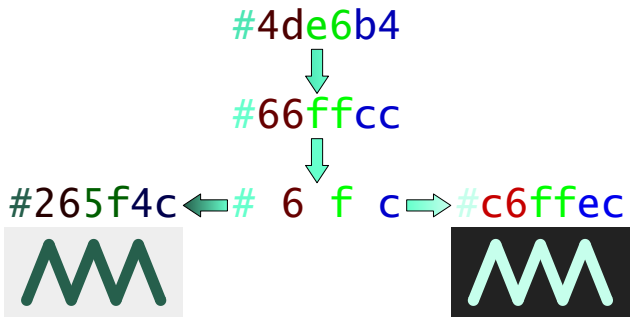
Vzhledem k tomu, že tmavé objekty na tmavém motivu a stejně tak světlé objekty na světlém motivu jsou špatně viditelné, tak se nabízí několik variant, jak tento problém vyřešit.

Jedním ze způsobů je zvolení daného motivu při vytváření tabule s tím, že by byl daný motiv nastaven pro všechny uživatele tabule a bez možnosti jej dále přepínat. Tento způsob však není příliš ideální, jelikož si tímto uživatel de facto vyřadí právě jednu z funkcí, které web nabízí.

Dále je poté možné implementovat pouze světlý (nebo případně pouze tmavý) motiv. Tento přístup lze pozorovat u téměř všech webů, kterými se práce inspirovala. V rámci řešení však byla snaha najít vhodnou variantu, jak by se dalo implementovat oba barevné motivy a zároveň zachovat určitý barevný kontrast samotných objektů.

Při implementaci řešení tedy došlo k rozhodnutí, že aby byl obsah plátna dobře viditelný ať už na tmavém či na světlém motivu, které bude možné libovolně přepínat, tak bude potřeba zavést určité barevné odstíny.

4.1.6.1.1 Zpracování barevného kontrastu



Obrázek 4.1: Příklad zpracování barvy #4de6b4

Jednotlivé objekty v rámci jejich vytváření či editace nabízí možnost změny barvy. Pro výběr barvy je využit HTML element `<input type="color">`, který barvu vrací v hexadecimálním formátu #rrggbb o barevné hloubce 24 bitů.

Pro zesvětlení či ztlumení dané barvy je vrácený řetězec nejdříve převeden do 12 bitové barevného hloubky. Tento krok je proveden převedením jednotlivých segmentů R, G a B do desítkové soustavy, vydělením 51, aritmetickým zaokrouhlením, vynásobením 3 a následně převedením zpět do hexadecimální soustavy. Výsledkem těchto operací je hexadecimální řetězec, jehož 1.–2., 3.–4. a 5.–6. pozice obsahují stejný znak, a proto je možné jej zkrátit do formátu #rgb.

Zkrácená varianta umožňuje vykreslit pouze 4 096 rozdílných barev, což je oproti původním 16 777 216 barvám veliký rozdíl. Pokud by se jednalo o nástroj pro profesionální práci s grafikou, tak by tato barevná ztráta byla absolutně nepřípustná. V tomto řešení však umožňuje právě již

zmíněné zesvětlení či ztlumení barvy tím, že do zkrácené verze řetězce se opět, do každé ze 3 barev, přidá vždy zleva jeden znak. Přidávané znaky zachovávají poměr jednotlivých barev od 0 do 5 a jsou případně zvýšené o požadovanou hodnotu od 0 do 10 tak, aby byly dostatečně viditelné v rámci daného motivu. Obrázek 4.1 graficky popisuje celý proces zpracování barev na uvedeném příkladu.

4.1.6.1.2 Kontrast jednotlivých typů objektů

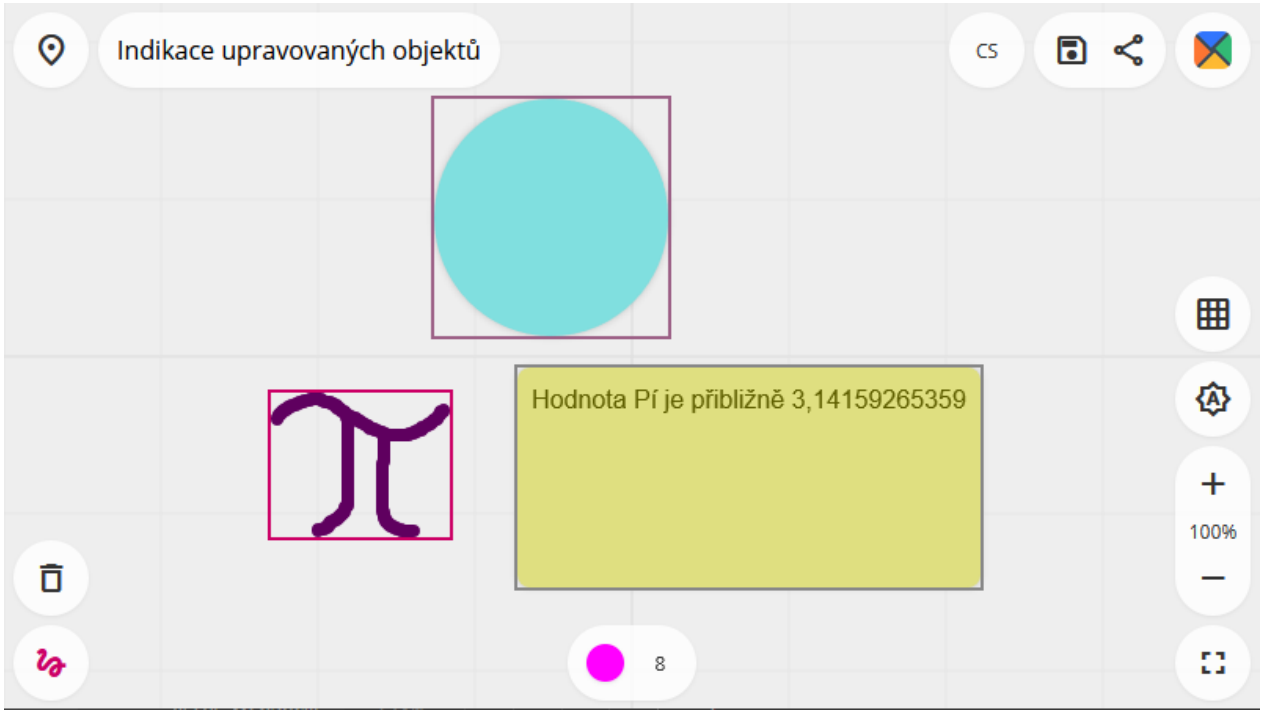
Barevný odstín textu poznámky je stejný jako u vykreslovaných čar volného kreslení. To stejné platí i o pozadí poznámky a pozadí geometrického tvaru.

Barevný odstín textu poznámky se pohybuje ve stejném rozmezí jako vykreslované lomené čáry u volného kreslení. Důvodem stejného rozmezí je především charakter daných artefaktů, jelikož oba jsou velmi úzké a proto na tmavém pozadí by měly být světlejší a zároveň na světlém pozadí tmavší.

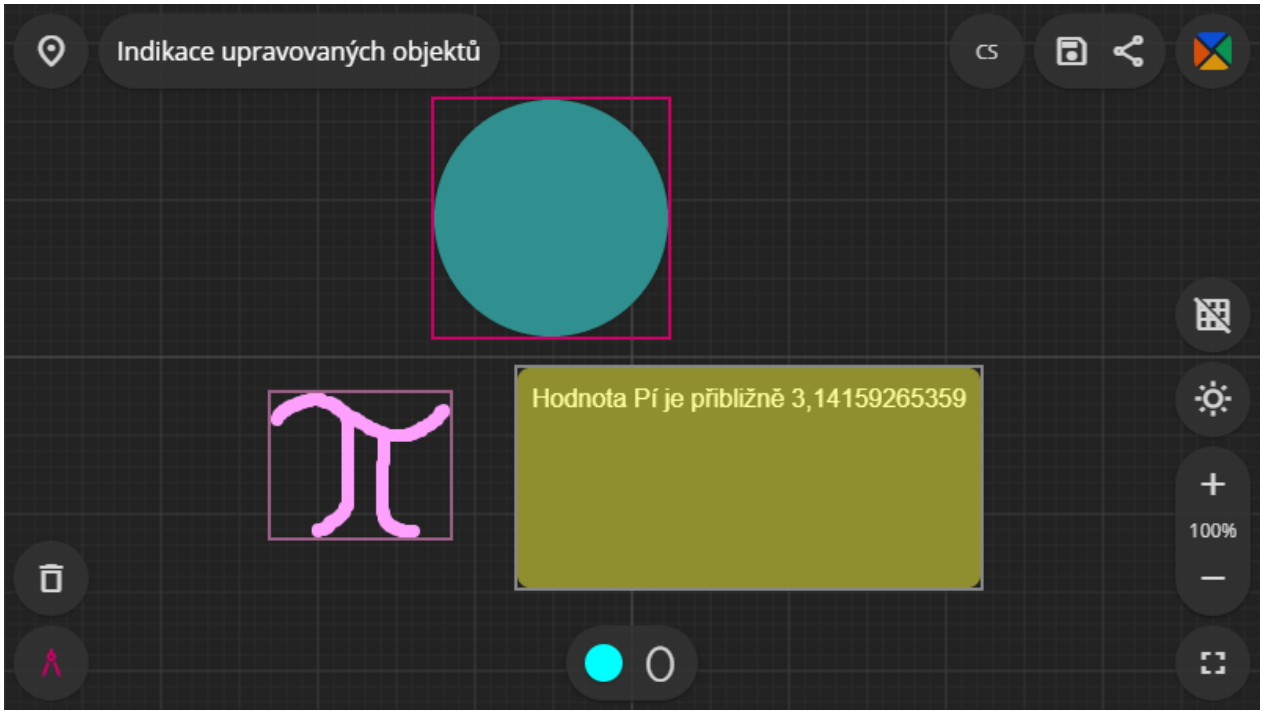
Pozadí poznámky i geometrického tvaru má rovněž stejné rozmezí, což je dáno tím, že zabírají relativně větší plochu a navíc u poznámky je nutné vzít v potaz také kontrast nejen vůči plátnu, ale také vůči samotnému textu poznámky.

4.1.7 Indikace upravovaných objektů

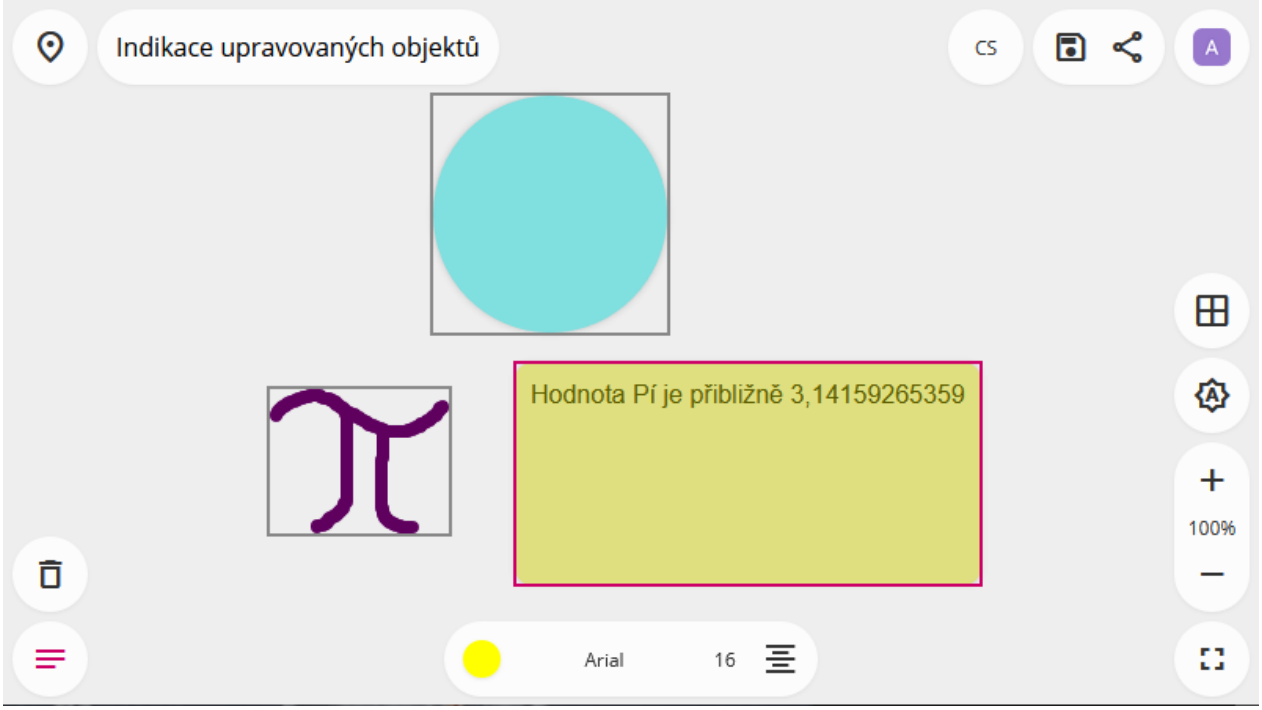
Aktuálně upravované objekty tabule lze jednoduše rozeznat díky tomu, že jsou během úpravy barevně ohraničeny. Barva ohraničení se liší na základě několika faktorů. Pokud se jedná o objekt upravovaný aktuálním uživatelem v aktuálním okně, je ohraničení sytě růžové. V případě, že jej sice upravuje aktuální uživatel avšak v jiném okně, je ohraničení zbarveno do šedě růžové barvy. Šedé je pak ohraničení v případě, že daný objekt edituje úplně jiný uživatel. Vykreslení barevného rámečku je čistě kosmetická záležitost, avšak samotný výběr objektu uživatelem je důležitá součást integrity tabule, která je dále podrobněji vysvětlena v následující kapitole.



Obrázek 4.2: Pohled uživatele X v okně A (poloviční mřížka, světlý motiv, Mozilla Firefox)



Obrázek 4.3: Pohled uživatele X v okně B (plná mřížka, tmavý motiv, Microsoft Edge)



Obrázek 4.4: Pohled uživatele Y v okně C (žádná mřížka, světlý motiv, Mozilla Firefox)

4.2 Datová komunikace se serverem

Důležitou součástí webu je možnost spolupráce více uživatelů zároveň v reálném čase. Sdílení dat je umožněno WebSocket serverem, který přijímá aktualizace jednotlivých detailů tabule a dále je rozesílá všem aktivně připojeným uživatelům kromě uživatele, který tuto změnu inicioval. Součástí práce serveru je také ukládání dat do databáze či do proměnných o čemž pojednává následující sekce 4.3.

4.2.1 Součásti WebSocket serveru

4.2.1.1 Node.js

Pro vývoj serverové části práce bylo vybráno prostředí Node.js. Jedná se o asynchronní runtime JavaScriptu, který právě díky asynchronnímu přístupu umožňuje nenáročný vývoj škálovatelných síťových aplikací. [9] Node.js aplikace se programují v jazyce JavaScript a tyto aplikace je možné díky aktivní Node.js komunitě obohatit o již vytvořené balíčky, které celý vývoj zjednodušují. Práce využívá ke svému fungování 4 balíčky uvedené níže:

- **google-auth-library** – určen k ověření klientského identifikačního čísla získaného po přihlášení k uživatelskému účtu Googlu
- **http** – slouží ke spuštění HTTP serveru na určitém síťovém portu, na který se poté klient připojuje
- **mysql** – umožňuje navázat komunikaci s MySQL databází
- **ws** – zajišťuje komunikaci s klientským zařízením pomocí protokolu WebSocket

4.2.1.2 WebSocket

Technologie WebSocket byla vybrána z důvodu dlouhodobého spolehlivého spojení skrze plně duplexní komunikační kanál. [10] Protokol vychází ze standardu RFC 6455 z roku 2011 a jeho aktuální podpora se pohybuje okolo 97–98 % všech uživatelů webových prohlížečů. [11][12]

4.2.1.3 JSON

WebSocket protokol umožňuje přenos binárních či textových dat (kódování UTF-8). [11] K výměně dat práce používá datového formátu JSON, který pro přenos převádí na text pomocí příkazu `JSON.stringify()`. JSON byl zvolen z důvodu snazšího debugování a vyšší přehlednosti výsledného kódu.

Nižší velikosti odesílaných a přijímaných dat by bylo možné dosáhnout využitím binárních dat, nicméně jejich využití je vhodné po ujasnění výsledné formy všech posílaných dat. JSON umožňuje rychleji a jednodušeji provádět změny struktury posílaných dat.

4.2.2 Připojení uživatele

Po vytvoření a spuštění je server připraven navázat spojení a komunikaci s webovou stránkou, pokud tato stránka splní určité přístupové kritéria.

4.2.2.1 Validace údajů

4.2.2.1.1 Lokální část

Prvním krokem před samotným pokusem o spojení je validace přístupových dat. Nejprve je provedena kontrola URL adresy, tedy konkrétně, zda je uživatel připojen přes zabezpečený HTTPS protokol a zda celé doménové jméno odpovídá očekávaní. Dále je pak ověřen řetězec přístupového režimu, který v adrese nabývá hodnot *user*, *edit* či *view* a v případě posledních dvou také desetimístný unikátní identifikátor tabule pro daný režim. Identifikátor se skládá z velkých a malých písmen anglické abecedy, číslic od 0 do 9 a dále pomlčky a podtržítka a je ověřen pomocí regexu `/^[A-Za-z0-9\-\_]{10}$/`. Jako poslední je také provedena kontrola, zda se uživatel přihlásil a zda po tomto přihlášení byl získán potřebný token k dalšímu ověření na serveru.

Toto základní ověření dat zajišťuje ochranu serveru před nežádoucími pokusy o připojení, především v případě možné interní chyby aplikace. Pokud byly všechny výše uvedené podmínky splněny, webová stránka zažádá o připojení na WebSocket server a do připojované URL adresy zahrne GET požadavky s uvedenými kontrolními daty.

4.2.2.1.2 Serverová část

Na WebSocket server se může připojit prakticky kdokoliv, kdo zná jeho adresu, která je na webu lehce dohledatelná ve vývojářských nástrojích. Nikdy tedy není možné předpokládat, že se uživatel připojuje pouze z konkrétního webu a také, že posílá smysluplné data. Je tedy vhodné všechny vstupy ověřovat a při detekci jakékoliv anomálie uživatele odpojit od serveru buď úplně či případně s možností se znovu automaticky připojit v případě nějaké chyby aplikace. Na serveru je tedy nejprve potřeba zkontrolovat původ připojovaného klienta, tedy opět zda se připojuje skrze HTTPS, z očekávaného webu, s použitelným režimem přístupu a v případě identifikátoru také jeho formátová správnost. Token vrácený po přihlášení uživatele je následně pomocí balíčku `google-auth-library` verifikován a pokud je tato akce úspěšná, tak je získán unikátní identifikátor uživatelského účtu Google tzv. `sub`. Po verifikaci je `sub` uložen do databáze (pouze v případě nového uživatele). Poslední kontrola se týká pouze přístupových režimů edit a view a zjišťuje, zda se unikátní identifikátor tabule nachází v databázi.

4.2.2.2 Režimy přístupu

4.2.2.2.1 Režim uživatelského profilu

- Přidání/editace/smazání více tabulí

4.2.2.2.2 Režim editace tabule

- Editace parametrů tabule (pouze její autor)
- Výběr objektu - editace, pohyb, smazání

4.2.2.2.3 Režim zobrazení tabule

- Osekaná verze režimu editace

4.2.3 Udržování spojení

- Ping/Pong

4.2.4 Důvody a řešení odpojení uživatele

4.2.4.1 Odpojení na základě chybné komunikace

4.2.4.2 Odpojení spojené s problémovým připojením k serveru

- Výpadek serveru
- Výpadek internetového připojení uživatele

4.3 Správa dat

4.3.1 Správa dat na serveru

4.3.1.1 MySQL databáze

4.3.1.2 Dočasné proměnné serveru

4.3.2 Lokální správa dat

4.3.2.1 Export tabule v nativní podobě formátu JSON

4.3.2.2 Nahrání obrázku pomocí Fetch API a PHP

4.3.2.3 Uložení uživatelských nastavení

4.3.2.3.1 LocalStorage

4.3.2.3.2 PHP Session

4.4 Uživatelské prostředí

4.4.1 Uživatelské rozhraní

4.4.1.1 Řešení responzivity

4.4.1.2 Barevné motivy GUI

4.4.1.3 Jazyky GUI

4.4.2 Jednotlivé stránky webu

4.4.2.1 Stránka uživatelského účtu

- Umožňuje přidání více tabulí...
- U každé tabule má tlačítka s odkazy na režim editace a zobrazení a smazání...

4.4.2.2 Stránka s tabulí

- Tlačítko pro přesouvání na důležité místa
- Tlačítka pro export a sdílení
- Tlačítko pro přepnutí na uživatelskou stránku či odhlášení
- Ovládání přiblížení tabule - zoom in/out/reset, fullscreen
- Tlačítka uživatelských nastavení - mřížka, barevné motivy, jazyk
- Tlačítka nástrojů tabule - kreslení, poznámka, obrázek, geometr. tvar, nativní podoba
- Rozdíly uživatelského rozhraní režimu editace a zobrazení

4.4.2.3 Chybová stránka

- Zobrazuje chyby 403, 404 a další...

4.4.3 Sdílení tabule

- Velmi krátká kapitola pouze o sdílení tabule, které nesedí do jiných kapitol
- Možnost zkopírování odkazu (bez dalších kroků, nejedná se o adresní řádek, ale o výběr možnosti v <select>)

4.4.3.1 Sdílení emailem

- Sdílení předem vytvořené zprávy v nastaveném jazyku - automaticky pomocí serveru (PHP funkce mail())
- Sdílení předem vytvořené zprávy v nastaveném jazyku - ručně uživatelem (mailto:)

Kapitola 5

Závěr

Příloha A

Projekt

Jednotlivé sekce přílohy byly vytvořeny za účelem doplnění zbývajících bodů požadavků předmětu Elektronické publikace. Ostatní požadavky již byly splněny v rámci samotného textu práce.

A.1 Zvýrazňování textu

Text psaný kapitálkami

A.2 Výčtová prostředí

1. První
2. Druhý
3. Třetí

Část I: Úvod

Část II: Vypracování

Část III: Závěr

A.3 Komentáře

A.4 Vkládání vzorců

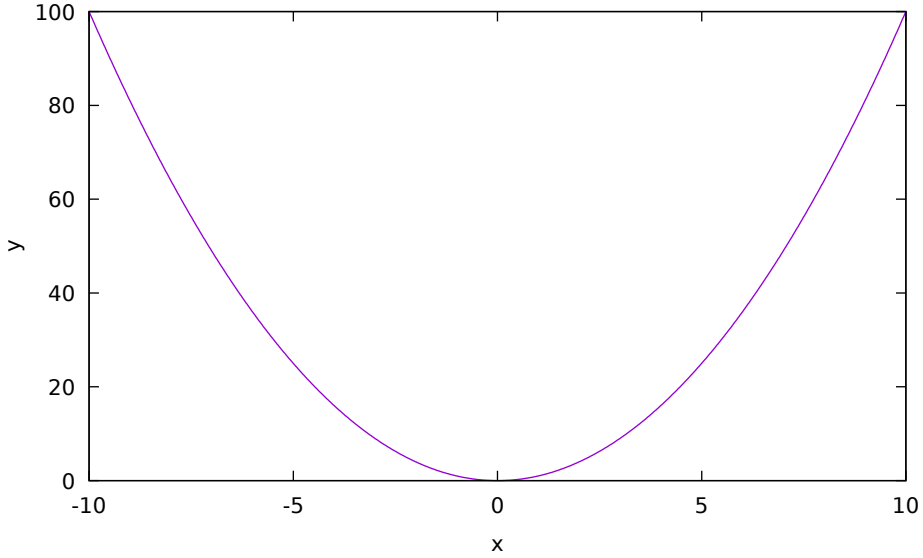
$$E = mc^2 \tag{A.1}$$

A.5 Sazba indexu

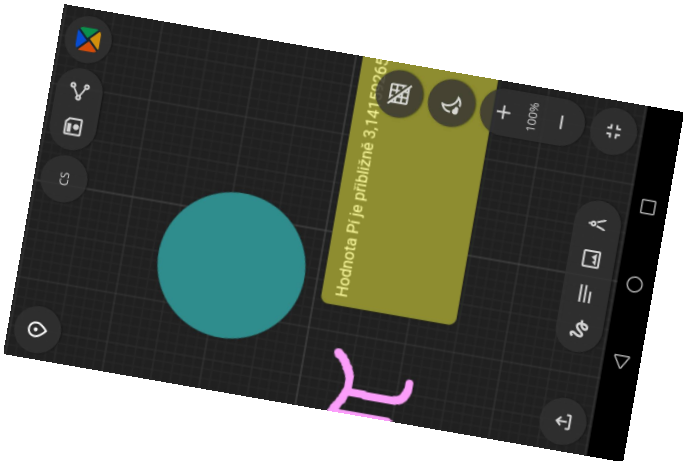
Tento text je napsán v jazyce LaTeX, který slouží k sazbě dokumentu formy bakalářské práce a také projektu do předmětu Elektronická publikace.

A.6 Vkládání grafů z GNUplot

Graf matematické funkce x^2:

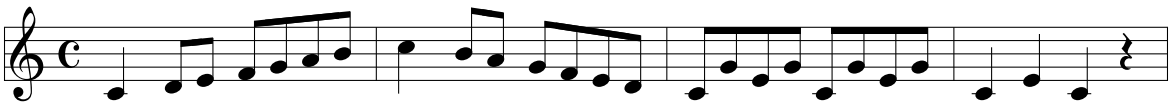


A.7 Kreslení pomocí balíčku graphicx



Obrázek A.1: Obrázek otočený o 80%

A.8 Sazba not



A.9 Sazba exotických druhů písma

- *Písmo Calligra*
- Písmo Punk
- **PISMO FOBEKFONT**

Literatura

1. *Main thread* [online] [cit. 2021-03-24]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Glossary/Main_thread`.
2. *Introduction to the DOM* [online] [cit. 2021-03-24]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction`.
3. *OffscreenCanvas* [online] [cit. 2021-04-02]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/OffscreenCanvas`.
4. *OffscreenCanvas* [online] [cit. 2021-04-02]. Dostupné z: `https://www.caniuse.com/offscreencanvas`.
5. *Improving HTML5 Canvas Performance* [online] [cit. 2021-03-26]. Dostupné z: `https://www.html5rocks.com/en/tutorials/canvas/performance/#toc-batch`.
6. *CanvasRenderingContext2D.arc()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/arc`.
7. *CanvasRenderingContext2D.closePath()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/closePath`.
8. *CanvasRenderingContext2D.fill()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fill`.
9. *About Node.js®* [online] [cit. 2021-04-05]. Dostupné z: `https://nodejs.org/en/about/`.
10. LOMBARDI, Andrew. *WebSocket*. Sebastopol: O'Reilly, 2015. ISBN 978-1449369279.
11. *The WebSocket Protocol* [online] [cit. 2021-04-08]. Dostupné z: `https://tools.ietf.org/html/rfc6455`.
12. *Web Sockets* [online] [cit. 2021-04-08]. Dostupné z: `https://caniuse.com/websockets`.

Rejstřík