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 funkčností již existujících webů. 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; CSS; HTML; JavaScript; JSON; MySQL; PHP; WebSockets; Web workers

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 the functionality of existing websites. 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; CSS; HTML; JavaScript; JSON; MySQL; PHP; WebSockets; Web workers

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
HTML	–	Hyper Text Markup Language
JSON	–	JavaScript Object Notation
PHP	–	PHP: Hypertext Preprocessor
SQL	–	Structured Query Language
URL	–	Uniform Resource Locator

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 klientským 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, dále jednotlivé možnosti importování a exportování obsahu tabule v rastrové či nativní podobě a také jednotlivé možnosti sdílení tabule. V poslední řadě se kapitola věnuje uživatelskému rozhraní z pohledu responzivity, podpory více jazyků 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

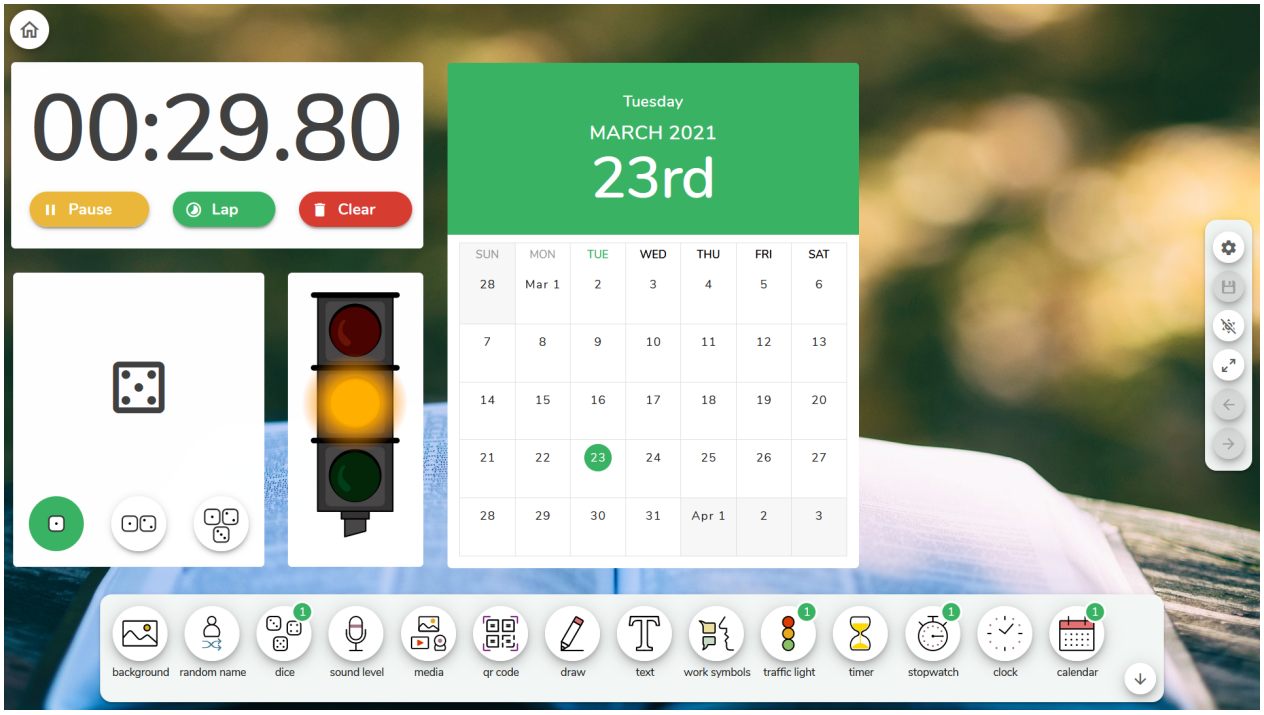
Weby použité pro inspiraci bylo nejdříve potřeba analyzovat jak z hlediska uživatelských nástrojů a rozhraní, tak z hlediska použitých technologií pro dosažení výsledné funkčnosti. K identifikaci použitých technologií bylo využito standardně zabudovaných vývojářských nástrojů webového prohlížeče, konkrétně 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.

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.

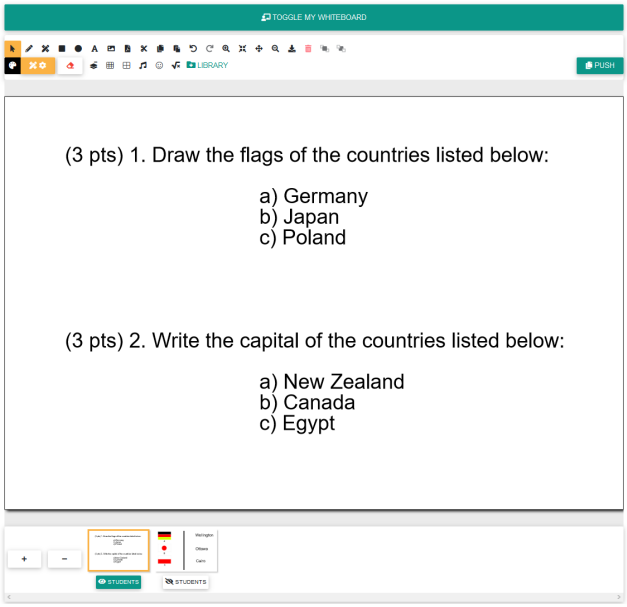


Obrázek 2.1: Ukázka webu classroomscreen.com

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

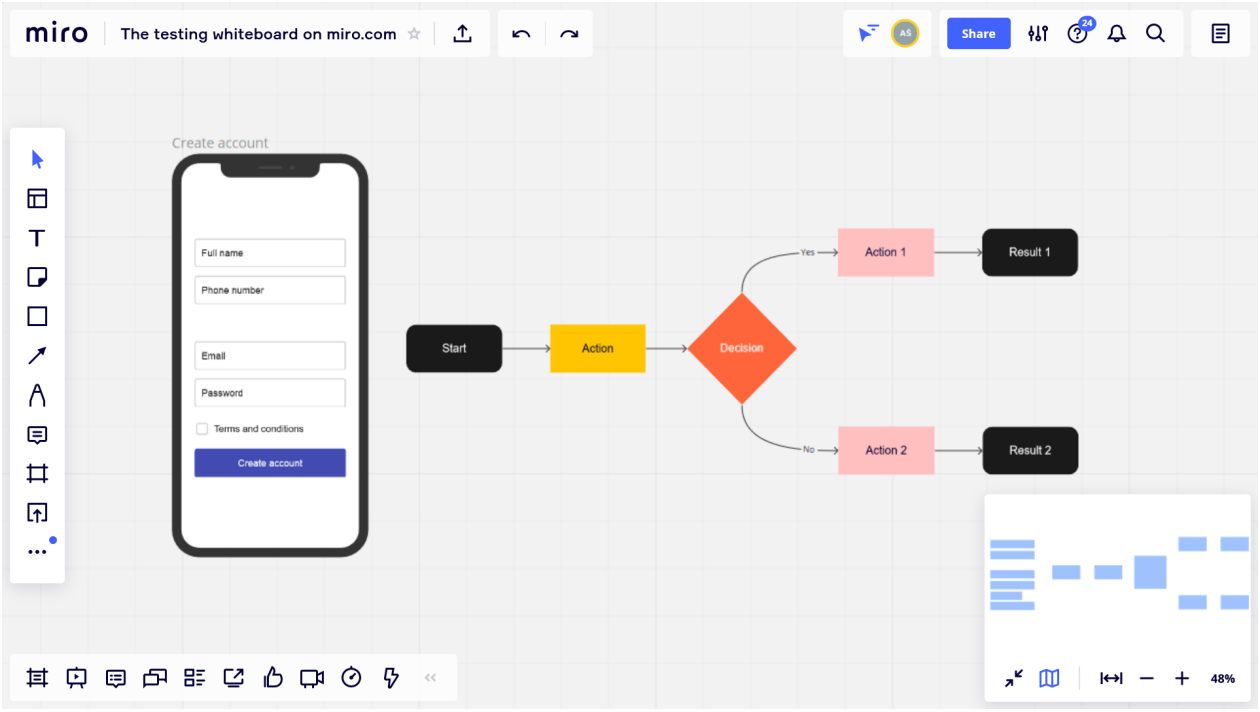
Předposledním vybraným řešením virtuální tabule je stránka collboard.com. Tato stránka vylepšuje nedostatky předchozího webu, co se týče náročnosti používání uživatelského rozhraní a doplňuje jej o celou řadu možností úprav jednotlivých objektů, jako např. změna tloušťky čáry u kreslení či změna řezu 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.



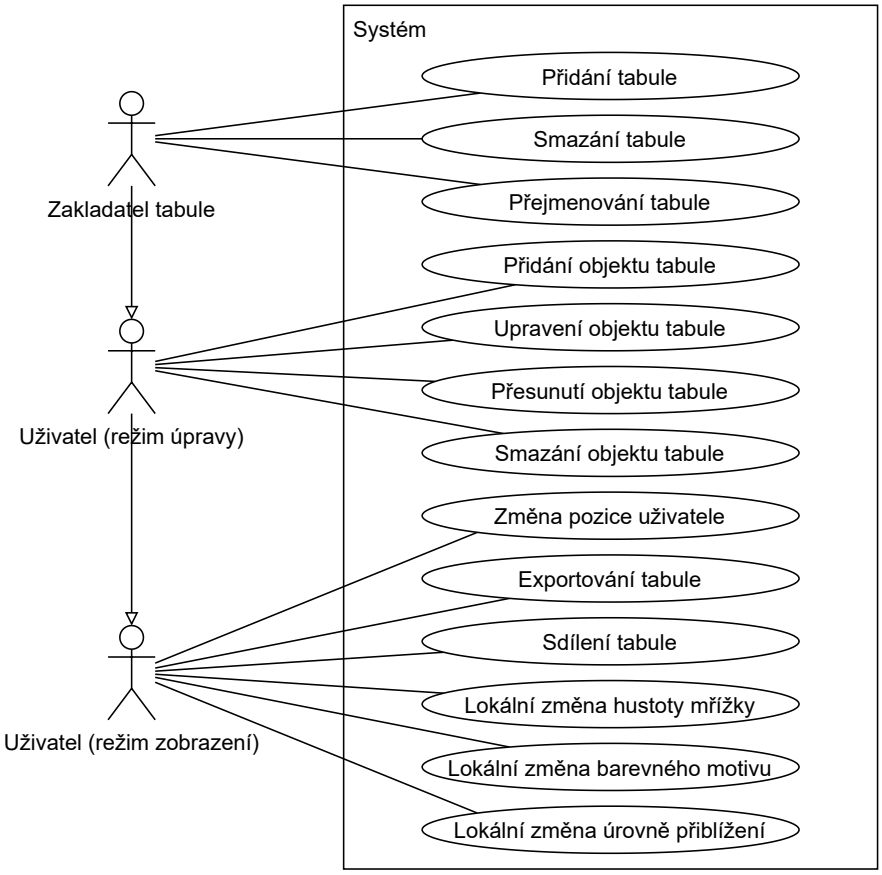
Obrázek 2.3: Ukázka webu miro.com

Kapitola 3

Požadavky, architektura a analýza

3.1 Požadavky

3.1.1 Využití řešení



Obrázek 3.1: Diagram případů užití

Systém počítá se třemi úrovněmi uživatelů, z nichž každý má o něco více možností využití systému. Jak je z diagramu na obrázku 3.1 patrné, nejméně pravomocí má uživatel v režimu zobrazení tabule. V tomto režimu uživatel obsah tabule pouze pasivně přijímá a nemá možnost do něj zasahovat. Dále systém počítá s uživatelem v režimu úpravy tabule, což je jakýkoliv uživatel kromě zakladatele tabule, kdo má přístup k odpovídajícímu odkazu tabule v daném režimu. V neposlední řadě má k systému přístup také zakladatel tabule, který může včetně všeho již zmíněného také navíc přidat, smazat či přejmenovat tabuli.

Více režimů přístupu k tabuli bylo zvoleno z toho důvodu, že ne každý při založení tabule chce, aby při jejím sdílení měli další uživatelé přístup k úpravě jejího obsahu. V režimu zobrazení tedy uživatelé nemají přístup k aktivní úpravě obsahu, což ale neznamená, že je pro ně tabule úplně nepoužitelná. Tabuli je v tomto režimu možné stále plnohodnotně zobrazit a také sdílet či exportovat.

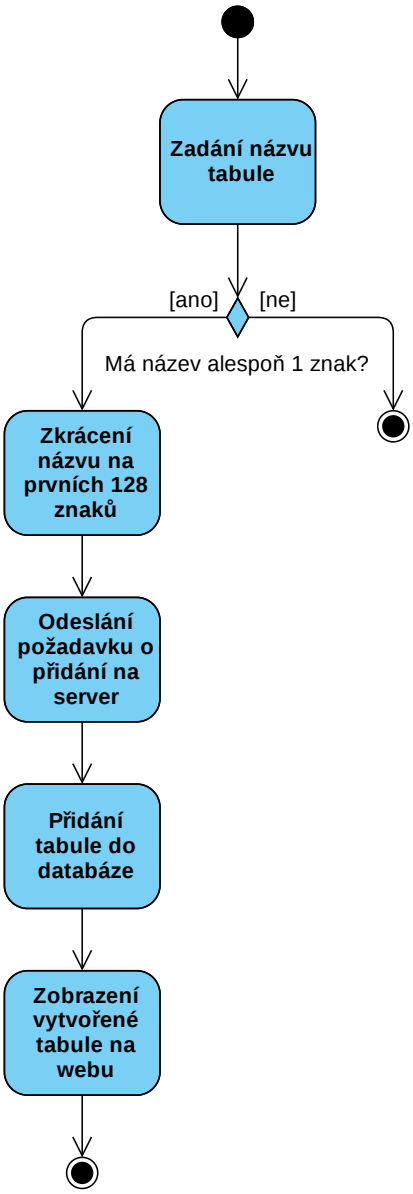
Jako příklad využití je možné uvést tuto situaci: aktuálně probíhá online výuka a učitel svůj výklad doplňuje o materiál vytvářený na sdílené tabuli. Vytvářený obsah chce zobrazit studentům, ale bojí se, že by mu do něj zasahovali, mazali jej a znemožňovali by pohodovou výuku, a proto jim pošle odlišný odkaz pouze pro zobrazení. Učitel tedy může v klidu vytvářet obsah, který se studentům automaticky v reálném čase aktualizuje. Součástí materiálu může být také domácí úkol, který bude pro jeho vypracování vyžadovat doplnění již připraveného nákresu. Pro vypracování však není nutné tabuli složitě fotit, ale je možné ji exportovat, nejlépe v nativní podobě pomocí zdrojového souboru, který je poté možné nahrát na nové tabuli, ve které student daný úkol vypracuje. Jako podklad ke studiu poté může student využít exportu do obrázku, který lze lépe zobrazit mimo web sdílené tabule, který není vytvořen za účelem procházení obrázků.

Zakladatel tabule by měl mít vždy o něco více pravomocí než jeden z mnoha možných upravujících uživatelů, především v přístupu k nastavením tabule, které se netýkají jejího obsahu. Jako ukázka výhody tohoto rozlišení je doplnění výše zmíněné situace: učitel si daný materiál připravuje společně s dalším kolegou, který mimo jiné externě učí na škole v Německu. Pokud by daný kolega změnil jméno společné tabule, mohl by učiteli způsobit chaos v podkladech. Kolega má však možnost si pro své zahraniční studenty obsah exportovat a dále využít dle potřeby na nové vlastní tabuli.

3.1.2 Vývoj aktivit vybraných funkcí

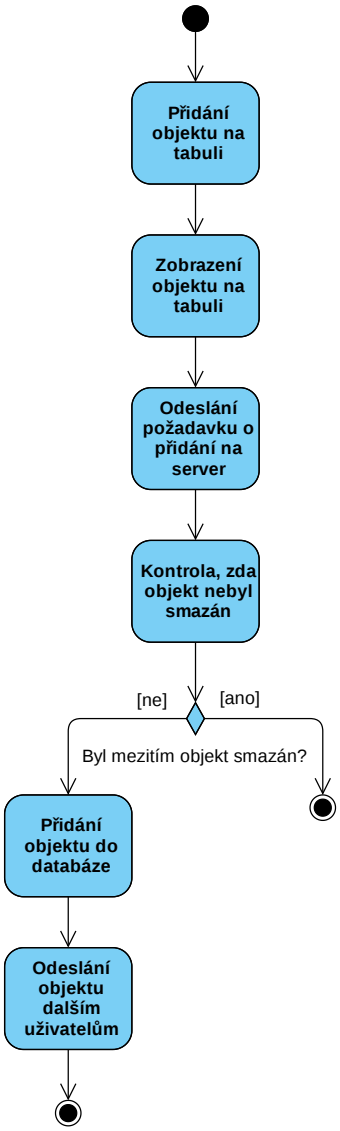
Systém k vykonávání požadovaných akcí využívá funkce, které provádí předem zadané úkony. Funkce má určitou vnitřní strukturu, která se skládá z jednotlivých po sobě jdoucích aktivit, které vedou k jednomu či více možným koncům.

Následující diagramy aktivit se zaměřují na vývoj vybraných funkcí pro přidání tabule a objektu tabule a mají vždy 2 možné konce, které jsou zajištěné validací uživatelova vstupu. Jednotlivé aktivity vybraných funkcí jsou poté dále podrobněji popsány.



Obrázek 3.2: Diagram aktivit – přidání tabule

První diagram se věnuje přidání nové tabule. Před přidáním uživatel zadává název tabule, který musí mít alespoň jeden znak, jinak je celý proces přidávání ukončen. Tabule by měla mít název, jelikož je podle něj možné identifikovat její obsah. Název by však neměl být ani příliš dlouhý, jelikož jeho účelem je pouze velmi stručné pojmenování tabule a pro přidání delšího textu je poté přímo v tabuli možné přidat poznámku, která může mít až 2 048 znaků. Jakkoliv dlouhý název, který projde úvodní kontrolou, je tedy poté zkrácen na maximální délku 128 znaků a dále skrze server přidán do databáze. Po přidání je tabule společně s vygenerovanými odkazy pro úpravu a zobrazení přidána na web uživatele bez nutnosti obnovení dané stránky.

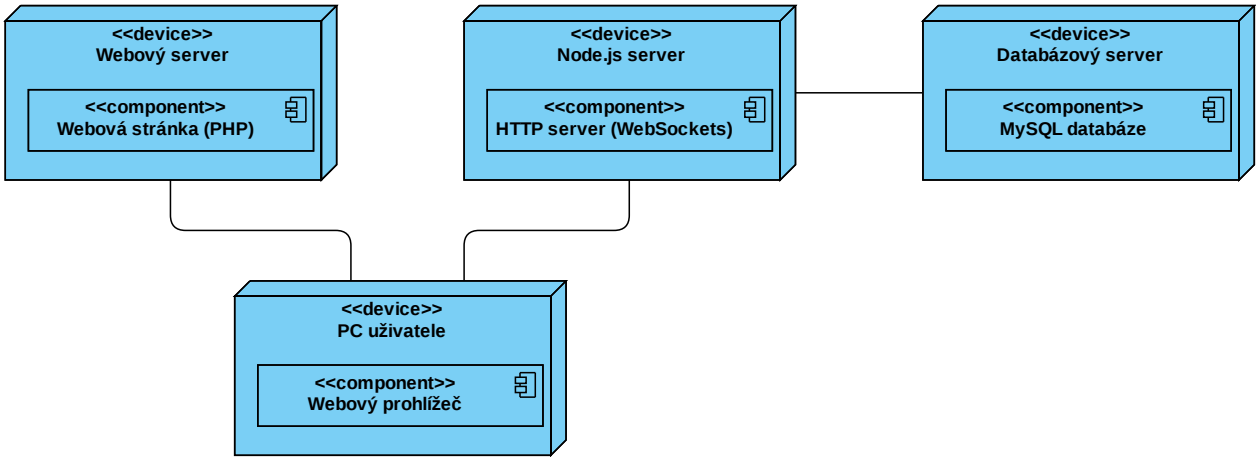


Obrázek 3.3: Diagram aktivit – přidání objektu

Druhý diagram je zaměřen na aktivity spojené s funkcí pro přidání objektu tabule. Nejprve uživatel daný objekt vytvoří, což okamžitě vede k jeho lokálnímu vykreslení k zajištění minimální latence mezi uživatelovým vstupem a výstupem webu. Poté je na server odeslán požadavek o přidání objektu, na který navazuje zpětný požadavek serveru na kontrolu toho, zda uživatel daný objekt mezitím již nesmazal. Tato kontrola je důležitá z toho důvodu, že objekt při jeho vytváření nemá přidělený žádný identifikátor, kterým by poté bylo možné daný objekt identifikovat např. při exportu tabule. Objekt tedy při vytváření posílá token, který je poté daným požadavkem serveru vyhledán a pokud není nalezen, tak je objekt považován za smazaný. Při nalezení tokenu je danému objektu přiřazeno id, pod kterým je poté objekt přidán do databáze. Po přidání je následně objekt odeslán také dalším uživatelům tabule.

3.2 Architektura systému

3.2.1 Propojení systémových zařízení a komponent

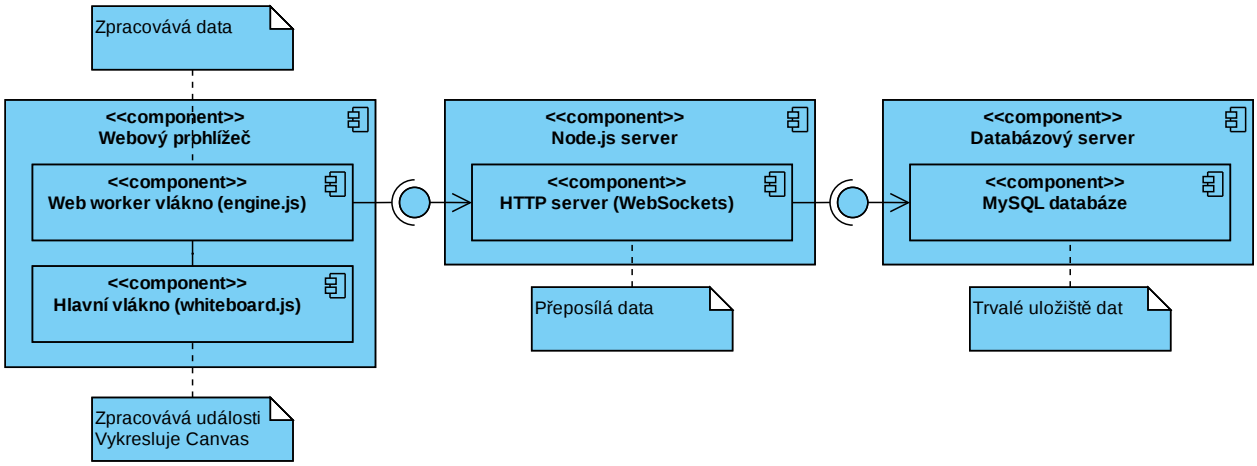


Obrázek 3.4: Diagram nasazení systému

Systémová infrastruktura se skládá ze 3 externích serverů, které komunikují s webovým prohlížečem uživatele a také mezi sebou navzájem. Na webovém serveru se nachází webová stránka, kterou uživatelské zařízení pomocí webového prohlížeče načítá do paměti zařízení. Webový prohlížeč poté komunikuje s WebSocket serverem, který data vyměňuje s databází, ale také s dalšími uživateli systému.

Důvod využití více serverů spočívá v tom, že každý se hodí na něco jiného. Webový server poskytuje statický obsah webu a není možné s ním udržovat dlouhodobé spojení skrze WebSocket protokol. Node.js server je zaměřen na dlouhodobou výměnu relativně malých dat a nehodí se tedy pro stahování webové stránky, zvlášť pokud web aktivně navštěvuje více uživatelů. Databázový server je poté určen pouze pro ukládání a následné načítání dat z databáze a není možné jej využít k jinému účelu.

Proč není databázový server připojen přímo k webovému prohlížeči? WebSocket server je zde jako prostředník potřeba, jelikož po odeslání požadavku jednoho uživatele může rovnou data rozeslat dále. Databázový server sám nemůže odeslat data na jiné zařízení a prohlížeč zase neví, kdy by o ně měl požádat. Pokoušet se data stahovat v nějakém pravidelném intervalu by bylo extrémně neefektivní a špatné řešení. WebSocket server tedy poskytuje velmi rychlou, nenáročnou a spolehlivou komunikaci mezi více uživateli a zároveň si také data průběžně ukládá do databáze, kde data zůstávají i při výpadku serveru.



Obrázek 3.5: Diagram komponent systému

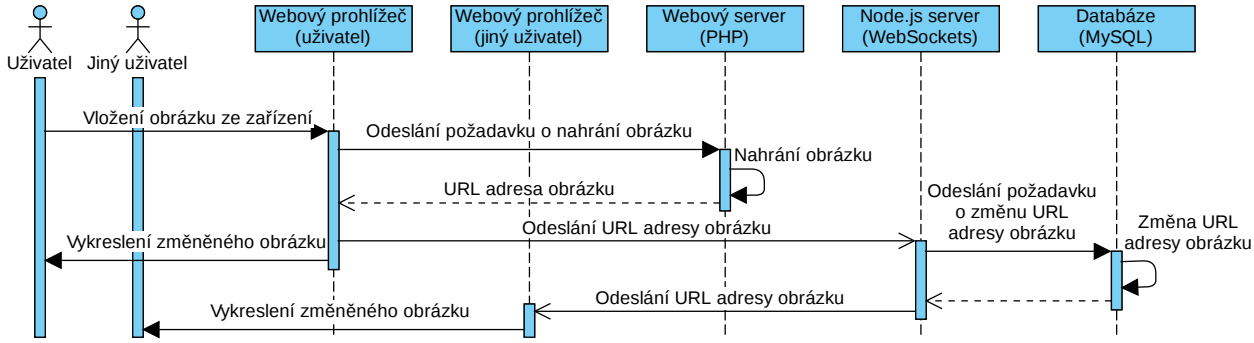
Instrukce webového prohlížeče jsou rozděleny do 2 vláken k čemuž je využito Web workers API. Web workers umožňují vytvořit vlákna, které mohou pracovat nezávisle na hlavním vláknu a hodí se tedy k současnému běhu několika instrukcí, které se navzájem neblokují. Hlavní vlákno se stará o zpracování událostí a vykreslování Canvasu, což aktuálně není možné provádět ve vedlejším vláknu, jelikož nemá přístup k DOM. Všechny akce nesouvisející s HTML ovšem lze v tomto vláknu provádět a je využito ke zpracování výpočtů a komunikaci s HTTP serverem pomocí WebSockets. Tyto operace mohou probíhat zároveň s vykreslováním Canvasu a detekcí pohybů myši, a tedy jednotlivé operace nečekají na dokončení vykreslení jednotlivých snímků či pohybů myši. Jak již bylo zmíněno na předchozí straně, WebSockets server slouží k výměně dat mezi jednotlivými uživateli a data dále ukládá do databáze.

3.2.2 Využití komponent ve vybraných funkcích

Systém ve svých funkcích využívá několik komponent, které plní různé úkoly a navzájem si poté mezi sebou posílají jejich výsledky. Komponenty jednotlivé operace provádí v určitém sekvenčním sledu, který je dán pořadím volání operací v kódu, ale také v případě asynchronních operací tím, která operace skončí dříve.

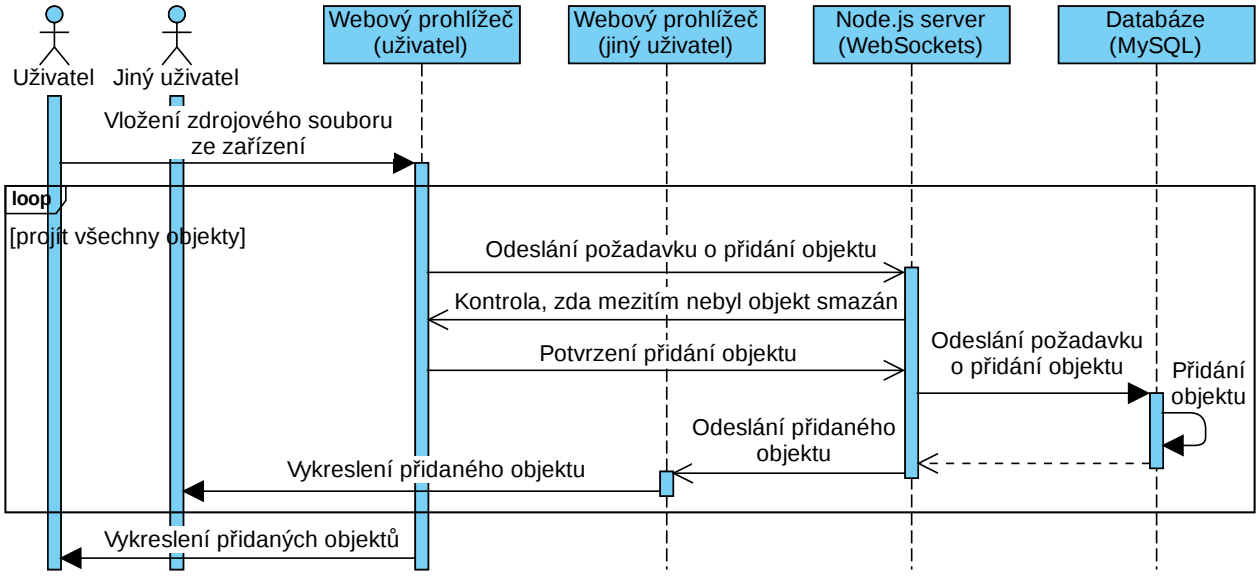
Každá komponenta systému je určena k jinému účelu. Webový prohlížeč slouží jako prostředník mezi uživatelem a servery, které dále zpracovávají a ukládají jeho data na externích zařízeních. Webový server především načítá statický obsah webové stránky, umožňuje však i jeho nahrání, které je dále popsáno na příkladu na obrázku 3.6. Node.js server se stará o komunikaci mezi jednotlivými uživateli a jejich data dále ukládá do databáze, ze které je poté také načítá. Každou komponentu je tedy potřeba využívat s vědomím jejich schopností, jelikož je vždy použitelná pouze v rámci určitých mezí daných použitými technologiemi.

Pro ukázku využití systémových komponent byly vybrány 2 funkce, které jsou pomocí sekvenčních diagramů vizuálně znázorněny a dále popsány z hlediska jejich konkrétního využití v řešení.



Obrázek 3.6: Sekvenční diagram – změna obrázku objektu tabule

Před začátkem dané funkce má uživatel vybrán objekt tabule, u kterého chce obrázek změnit. Samotná funkce začíná tím, že uživatel klikne na tlačítko pro změnu daného obrázku, načež se mu objeví dialog, ve kterém nový obrázek ze zařízení vybere. Po nahrání do paměti prohlížeče je obrázek odeslán na webový server, který jej nahraje (pokud obrázek na serveru ještě není) a vrací relativní cestu tohoto obrázku zpátky do prohlížeče uživatele. Nahrání obrázku na server a získání jeho odkazu je důležité pro zobrazení obrázku v budoucích relacích, jelikož je tento odkaz dále odeslán na Node.js server, který jej přeposílá do databáze, kde je nahrazen původní odkaz. Obrázek je mezitím již lokálně zobrazen uživateli, který jej přidal, jelikož je mu lokálně dostupný již před odesláním na server. Po změně v databázi je následně adresa odeslána do prohlížečů dalších uživatelů, které zajišťují její použití při vykreslení daného objektu.



Obrázek 3.7: Sekvenční diagram – nahrání zdrojového souboru

Funkce začíná tím, že uživatel klikne na tlačítko pro nahrání zdrojového souboru a v dialogu zařízení vybere požadovaný soubor. Po nahrání souboru probíhá cyklus, v němž jsou z poskytnutých dat získány jednotlivé objekty, které jsou poté po konci cyklu uživateli lokálně vykresleny. V cyklu jsou nejdříve odeslány požadavky o přidání na server, který daný objekt přidá do databáze ve stavu smazaného objektu, jelikož zatím není provedena kontrola, zda je stále přitomen na tabuli a po přidání je získáno jeho id. Dále Node.js server posílá požadavek o kontrolu, zda objekt nebyl mezitím smazán, jelikož předchozí operace nějakou dobu trvají, a tedy je možné, že se již na tabuli nenachází. Pokud však stále na tabuli je, tak je mu přiřazeno ono id a je na server odesláno potvrzení o přidání objektu. Server poté dále odesílá požadavek o přidání do databáze, ve které se daná položka změní ze stavu *smazaná* na *nesmazaná*. Po této změně je následně daný objekt odeslán také do prohlížečů všech dalších uživatelů tabule a následně je tento objekt vykreslen.

Pro upřesnění, lokální vykreslení přidaných objektů u uživatele probíhá mimo daný cyklus, avšak téměř okamžitě po odeslání veškerých prvotních požadavků o přidání. Tyto požadavky jsou asynchronní a neblokují systém během čekání na odpověď serveru, takže po jejich vykonání prohlížeč okamžitě pokračuje dalšími operacemi.

3.2.3 Odhad počtu současně pracujících uživatelů

Infrastruktura systému je koncipována pro současnou práci několika desítek uživatelů a je tedy vhodná pro menší firmy, školní výuku či pro osobní použití. Systémová databáze je ovšem připravena na mnohem vyšší počty uživatelů a je tedy po provedení určitých vylepšení této infrastruktury možné do budoucna počítat i s několika tisíci či miliony uživatelů.

3.2.4 Typy interakcí uživatelů a odhad jejich náročnosti

Uživatelé mají možnost přidávat, upravovat a mazat obsah tabule. Interakce jednotlivých uživatelů navzájem není nikterak neobvyklá a rovněž také není příliš náročná na hardware uživatele či systému. Vyšší náročnost se může projevit až s vyšším počtem uživatelů a je tedy potřeba při nasazení systému do provozu co nejrychleji reagovat na poptávku a další požadavky systému.

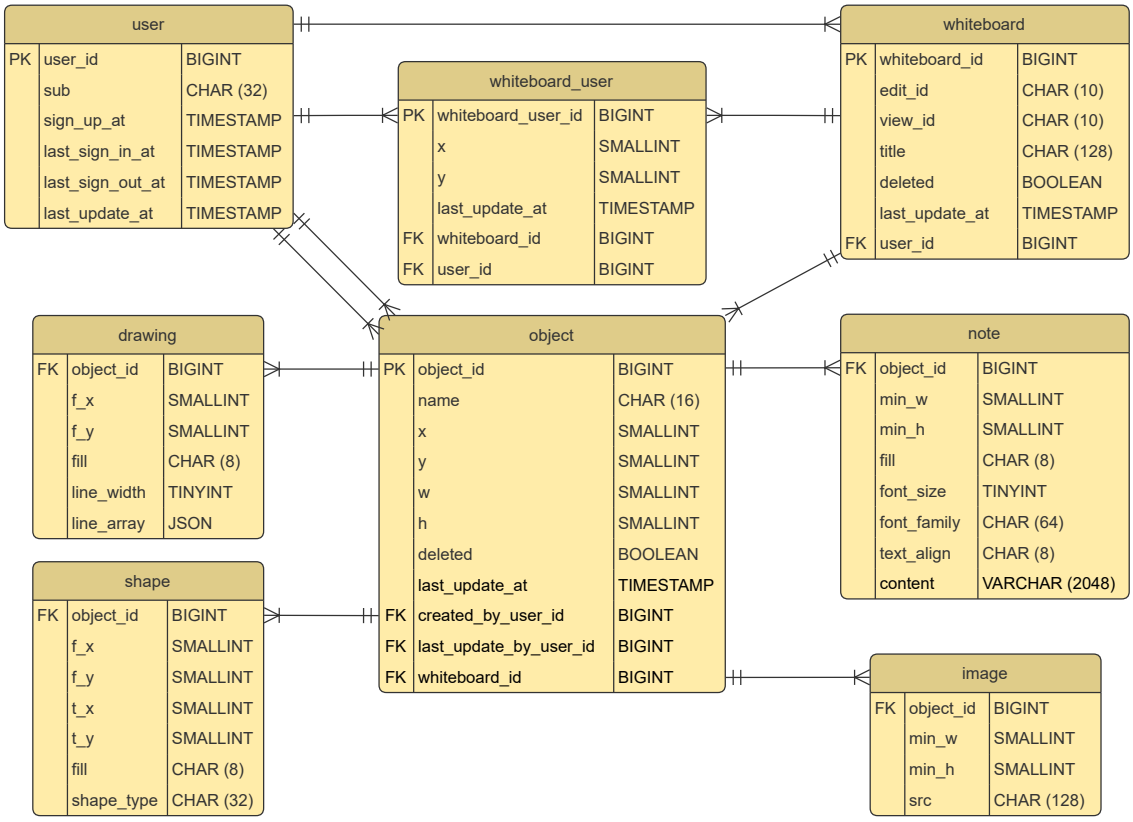
3.2.5 Zvolené technologie

Systém je určen k vývoji webové aplikace, jejíž design je vytvořen s pomocí HTML a CSS. Implementace systému probíhá v jazyce JavaScript s částečným zapojením PHP. Jako prostředník pro komunikaci mezi klientem a dalšími klienty, ale také databázi je využit Node.js server, na kterém běží HTTP server využitý pro běh WebSockets. MySQL databáze využívá SQL příkazů k ukládání a načítání dat.

3.3 Analýza a návrh

3.3.1 Datová analýza

3.3.1.1 Relační datový model databáze



Obrázek 3.8: Relační datový model databáze

3.3.1.2 Lineární zápis typů entit

Legenda: **tabulka**, primární klíč, *cizí klíč*, atribut

user(user_id, sub, sign_up_at, last_sign_in_at, last_sign_out_at, last_update_at)

whiteboard(whiteboard_id, edit_id, view_id, title, deleted, last_update_at, *user_id*)

whiteboard_user(whiteboard_user_id, x, y, last_update_at, *whiteboard_id*, *user_id*)

object(object_id, name, x, y, w, h, deleted, last_update_at, *created_by_user_id*, *last_update_by_user_id*, *whiteboard_id*)

drawing(*object_id*, f_x, f_y, fill, line_width, line_array)

image(*object_id*, min_w, min_h, src)

note(*object_id*, min_w, min_h, fill, font_size, font_family, text_align, content)

shape(*object_id*, f_x, f_y, t_x, t_y, fill, shape_type)

3.3.1.3 Datový slovník

Datový slovník doplňuje relační datový model databáze o informace o tabulce a jejích atributech, jejichž význam či další omezení nejsou z modelu zřejmé. U jednotlivých atributů je uveden jejich název, datový typ, maximální délka a klíč. Dále také informace, která říká, zda může být daný atribut v tabulce prázdný a jaký je jeho význam. Jedním ze základních prvků relační databáze je to, že jednotlivé tabulky mezi sebou vystupují v určitých vzájemných vztazích. Vztahy jsou v diagramu znázorněny propojením 2 tabulek a ve slovníku jsou poté u každé tabulky uvedené důvody tohoto propojení.

3.3.1.3.1 Tabulka user

Tabulka user slouží především k uchování identifikátoru externího účtu Google a k zaznamenání času registrace, přihlášení, odhlášení a poslední změny. Časové údaje v řešení nejsou nikde zobrazeny, avšak je dobré tyto údaje ukládat, především pro jejich možné budoucí využití. Uživatelské id je poté využito ať už při zakládání tabule jako její cizí klíč v tabulce whiteboard pro identifikaci zakladatele tabule či při vytváření a změně objektu v tabulce object. Dále také v tabulce whiteboard_user, která slouží pro ukládání pozice uživatele na tabuli.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
user_id	BigInt	20	PK	Ne	Identifikátor uživatele
sub	Char	32		Ne	Identifikátor uživatele účtu Google
sign_up_at	Timestamp			Ne	Čas registrace
last_sign_in_at	Timestamp			Ne	Čas posledního přihlášení
last_sign_out_at	Timestamp			Ano	Čas posledního odhlášení
last_update_at	Timestamp			Ne	Čas poslední aktualizace

Tabulka 3.1: Tabulka user

3.3.1.3.2 Tabulka whiteboard

Tabulka whiteboard slouží k ukládání informací ohledně samotné virtuální tabule. Obsahuje unikátní desetimístné identifikátory pro úpravu a zobrazení tabule a také její název a informaci o tom, zda byla tabule smazána či nikoliv. Nechybí ani čas poslední úpravy tabulky, který rovněž jako v případě tabulky user není nikde v řešení vidět i když by mohl být. Tabulka vystupuje pod cizím klíčem v tabulce whiteboard_user pro identifikaci, ke které tabuli vlastně uživatelova pozice patří a dále také v tabulce object, pro určení, na které tabuli se daný objekt nachází.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
whiteboard_id	BigInt	20	PK	Ne	Identifikátor tabule
edit_id	Char	10		Ne	Identifikátor módu úpravy tabule
view_id	Char	10		Ne	Identifikátor módu zobrazení tabule
title	Char	128		Ne	Název tabule
deleted	Bit	1		Ne	Je tabule smazaná?
last_update_at	Timestamp			Ne	Čas poslední aktualizace
user_id	BigInt	20	FK	Ne	Identifikátor zakladatele tabule

Tabulka 3.2: Tabulka whiteboard

3.3.1.3.3 Tabulka whiteboard_user

Tabulka whiteboard_user slouží k průběžnému zaznamenávání pozice uživatele na dané tabuli. Kromě souřadnic pozice x a y je opět zaznamenán také poslední čas úpravy. Tabulka pod žádným cizím klíčem v databázi nevystupuje.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
whiteboard_user_id	BigInt	20	PK	Ne	Identifikátor uživatele tabule
x	SmallInt	6		Ne	Souřadnice x pozice uživatele
y	SmallInt	6		Ne	Souřadnice y pozice uživatele
last_update_at	Timestamp			Ne	Čas poslední aktualizace
whiteboard_id	BigInt	20	FK	Ne	Identifikátor tabule
user_id	BigInt	20	FK	Ne	Identifikátor uživatele

Tabulka 3.3: Tabulka whiteboard_user

3.3.1.3.4 Tabulka object

Tabulka object slouží k zaznamenání obecných informací objektu, tedy jeho typu, pozice, velikosti a toho, zda je smazaný či nikoliv. Dále také nechybí poslední čas úpravy a cizí klíče odkazující na uživatele, který objekt vytvořil, naposledy upravil a také na tabuli, ve které se daný objekt nachází. Cizí klíč této tabulky se objevuje v tabulkách drawing, note, image a shape, jelikož tyto tabulky pouze rozšiřují informace uvedené v této tabulce.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
object_id	BigInt	20	PK	Ne	Identifikátor objektu tabule
name	Char	16		Ne	Typ objektu
x	SmallInt	6		Ne	Souřadnice x pozice objektu
y	SmallInt	6		Ne	Souřadnice y pozice objektu
w	SmallInt	6		Ne	Šířka objektu
h	SmallInt	6		Ne	Výška objektu
deleted	Bit	1		Ne	Je objekt smazaný?
last_update_at	Timestamp			Ne	Čas poslední aktualizace
created_by_user_id	BigInt	20	FK	Ne	Identifikátor tvůrce objektu
last_update_by_user_id	BigInt	20	FK	Ne	Identifikátor posledního editora
whiteboard_id	BigInt	20	FK	Ne	Identifikátor tabule

Tabulka 3.4: Tabulka object

3.3.1.3.5 Tabulka drawing

Tabulka drawing slouží k uchování informací ohledně objektu volného kreslení. Tato tabulka kromě záznamu o barvě výplně a tloušťce kreslené čáry zaznamenává také počáteční pozici kreslení a samotné raw data relativních pozic čar, které se odvíjí od již zmíněné počáteční pozice. Důvodem pro zvolení záznamu relativních pozic vůči dané pozici je fakt, že těchto dat o pozicích je opravdu hodně a je tedy vhodné, aby byly co nejmenší. Relativní data se budou vždy pohybovat okolo 0, což v případě absolutních pozic neplatí, jelikož tabule může být veliká i několik tisíc pixelů. Další výhodou je efektivní přesouvání tohoto objektu po tabuli, jelikož se vždy mění pouze absolutní počáteční pozice a jednotlivé body čáry tedy není nutné měnit.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
object_id	BigInt	20	FK	Ne	Identifikátor objektu tabule
f_x	SmallInt	6		Ne	Souřadnice x počáteční pozice kreslení
f_y	SmallInt	6		Ne	Souřadnice y počáteční pozice kreslení
fill	Char	8		Ne	Barevná výplň
line_width	TinyInt	3		Ne	Tloušťka čáry
line_array	JSON			Ne	Seznam čar

Tabulka 3.5: Tabulka drawing

3.3.1.3.6 Tabulka image

Tabulka image slouží k zaznamenání objektu obrázku, u kterého se uchovává jeho minimální velikost daná při jeho vytváření a relativní odkaz na zdroj obrázku na webovém serveru. Minimální šířka a výška obrázku je na webu nastavena na pevnou hodnotu 16 pixelů, což je také velikost ikony, která se objeví při načítání či chybě načítání daného obrázku. Odkaz na zdroj obrázku je uveden jako relativní odkaz, jelikož je daný obrázek uložen na webovém serveru, který rovněž tento obrázek načítá, jinak by však bylo vhodnější ukládat absolutní adresu pro univerzálnější budoucí použití.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
object_id	BigInt	20	FK	Ne	Identifikátor objektu tabule
min_w	SmallInt	6		Ne	Minimální šířka objektu
min_h	SmallInt	6		Ne	Minimální výška objektu
src	Char	128		Ne	Zdrojový odkaz

Tabulka 3.6: Tabulka image

3.3.1.3.7 Tabulka note

Tabulka note slouží k uložení informací poznámky, tedy její minimální velikosti, barevné výplně, velikosti a rodiny písma, zarovnání textu a textového obsahu. Minimální velikost poznámky je dána při jejím vytvoření, kdy uživatel navolí požadovanou velikost. Pokud by text poznámky přesáhl její minimální velikost, pak je poznámka odpovídajícím způsobem do oné konfliktní strany zvětšena. Zarovnání textu může nabývat následujících 3 hodnot – left (doleva), right (doprava) a center (na střed). Textový obsah je omezen na 2 048 znaků, což je dostatek na velmi dlouhý text a tento limit je určen především jako ochrana před posíláním a ukládáním až absurdně dlouhých zpráv. MySQL totiž u datového typu Varchar podporuje délku až 65 535 znaků a takto dlouhá zpráva je spíše pokusem o Denial of Service útok než o skutečnou smysluplnou poznámku.

Atribut	Dat. typ	Délka	Klíč	Null	Význam
object_id	BigInt	20	FK	Ne	Identifikátor objektu tabule
min_w	SmallInt	6		Ne	Minimální šířka objektu
min_h	SmallInt	6		Ne	Minimální výška objektu
fill	Char	8		Ne	Barevná výplň
font_size	TinyInt	3		Ne	Velikost písma
font_family	Char	64		Ne	Rodina písem
text_align	Char	8		Ne	Zarovnání textu
content	Varchar	2 048		Ano	Textový obsah

Tabulka 3.7: Tabulka note

3.3.1.3.8 Tabulka shape

Tabulka shape slouží k uchování informací o geometrickém tvaru, mezi které patří souřadnice počáteční a konečné pozice získané výběrem velikosti objektu při jeho vytváření. Dále také barevná výplň a typ geometrického tvaru. Počáteční a konečná pozice hrají velkou roli právě u různých typů tvarů, jelikož tyto tvary mohou být dle vzájemné pozice obou bodů otočeny o 90, 180 či 270 stupňů. Položka typu tvaru může nabývat následujících 5 hodnot – rectangle (obdélník), ellipse (elipsa), right_triangle (pravoúhlý trojúhelník), isosceles_triangle (rovnoramenný trojúhelník) a rhombus (kosočtverec).

Atribut	Dat. typ	Délka	Klíč	Null	Význam
object_id	BigInt	20	FK	Ne	Identifikátor objektu tabule
f_x	SmallInt	6		Ne	Souřadnice x počáteční pozice objektu
f_y	SmallInt	6		Ne	Souřadnice y počáteční pozice objektu
t_x	SmallInt	6		Ne	Souřadnice x konečné pozice objektu
t_y	SmallInt	6		Ne	Souřadnice y konečné pozice objektu
fill	Char	8		Ne	Barevná výplň
shape_type	Char	32		Ne	Typ tvaru

Tabulka 3.8: Tabulka shape

3.3.2 Stavová analýza

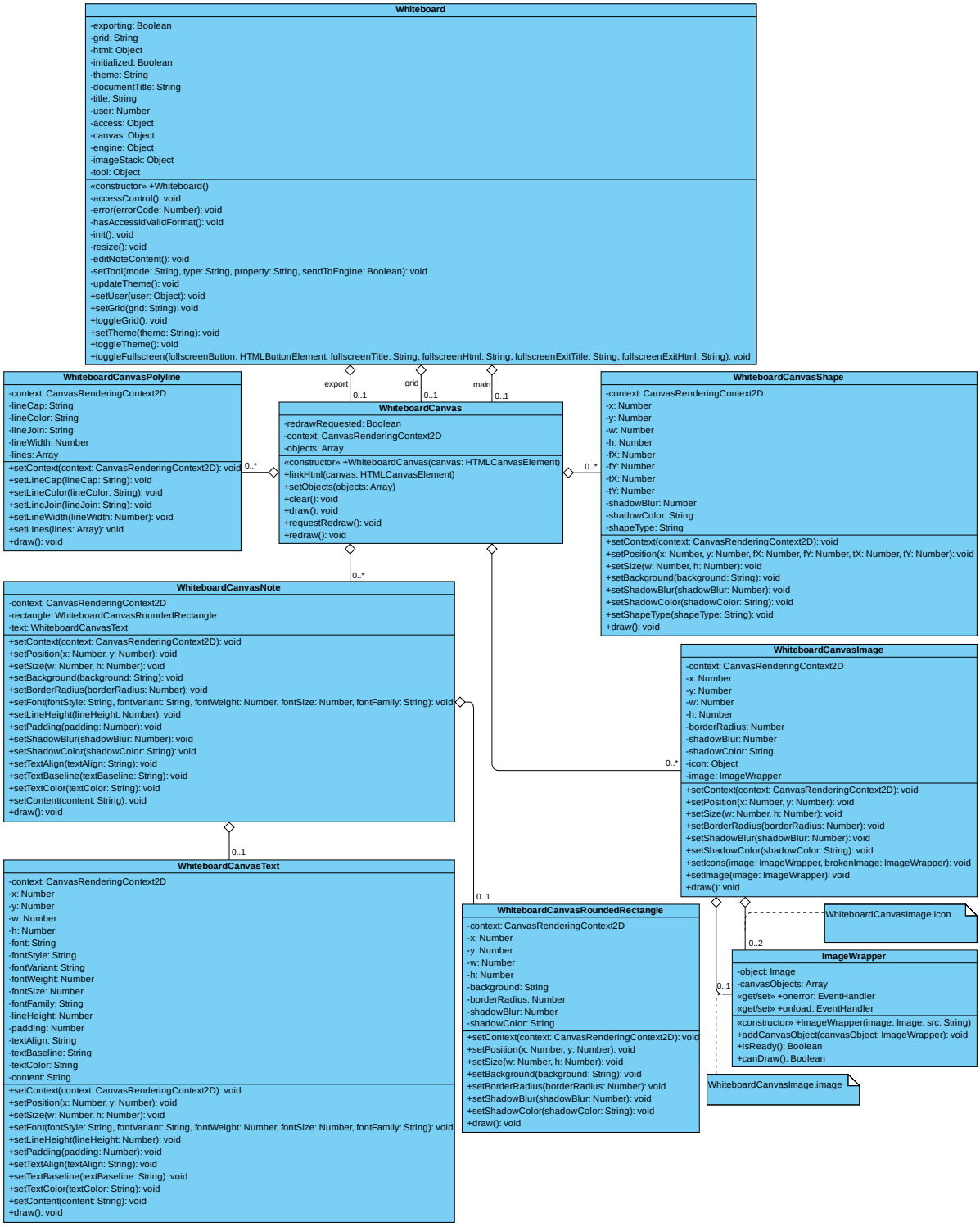
Definujeme tyto stavy tabule:

- **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

3.3.3 Návrh třídního modelu



Obrázek 3.9: Diagram tříd

3.3.3.1 Využití jednotlivých tříd

Systém k vykreslování obsahu používá JavaScriptové rozhraní CanvasRenderingContext2D, které nabízí základní metody pro práci s grafickým plátnem (tzv. Canvas). Pro vykreslení většího počtu objektů je ovšem vhodné dané operace volat efektivněji a jednodušeji, a proto jsou pro sloučení určitých operací vytvořeny vlastní třídy, které umožňují jednodušší a přehlednější práci s Canvasem. Seznam níže popisuje využití jednotlivých tříd:

- **Whiteboard** – Třída obsahuje objekty třídy WhiteboardCanvas, které využívá k vykreslení mřížky a samotného obsahu tabule a také k jejímu exportu. Dále se stará o komunikaci s vedlejším vláknem, které používá k většině výpočtů a výměně dat s WebSocket serverem. Třída také zajišťuje interaktivitu uživatelského rozhraní a reaguje na události s ním spojené.

- **WhiteboardCanvas** – Třída slouží k vykreslování přidělených objektů. Objekty mohou být třídy WhiteboardCanvasPolyline, WhiteboardCanvasNote, WhiteboardCanvasImage a WhiteboardCanvasShape, které jsou popsány v dalších bodech.
  Hlavní výhodou této třídy je její zaměření na co nejnižší frekvenci překreslování obsahu. Obsah je překreslován pouze při jeho změně s maximální frekvencí odpovídající obnovovací frekvenci obrazovky. Požadavek o překreslení je vyvolán metodou `requestRedraw()`.

- **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. [2]

- **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. [3] Oblouky jsou následně spojeny zavoláním funkce `CanvasRenderingContext2D.closePath()`. [4] 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()`. [5]

- **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.
  Řešení nabízí hned několik možných typů tvarů, konkrétně jde o obdélník, elipsu, pravoúhlý a rovnoramenný trojúhelník a kosočtverec. 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.

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). [6] 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, která daný požadavek vyvolá.
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 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 30720×17280px 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. [7] 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. [8]

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 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.

4.1.5.1 Barevný kontrast

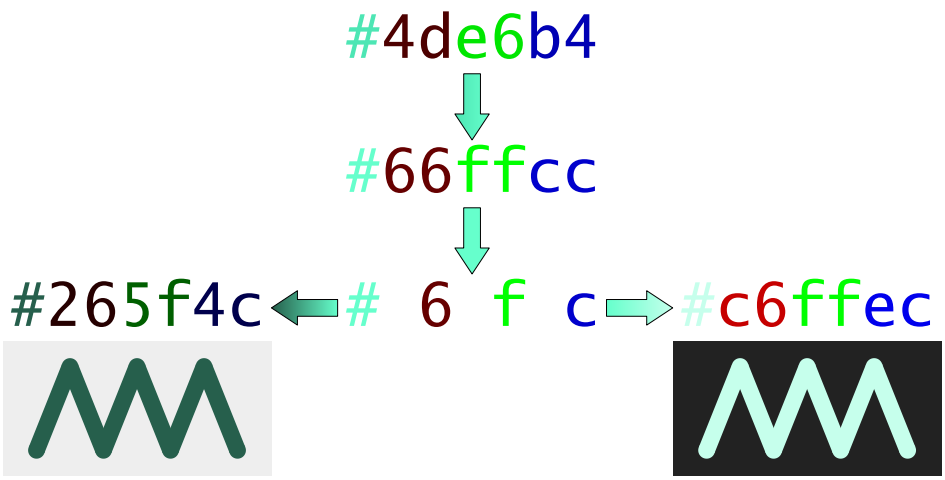
Vzhledem k tomu, že tmavé objekty na tmavém motivu a stejně tak světlé objekty na světlém motivu jsou špatně viditelné, tak se nabízí několik variant, jak tento problém vyřešit.

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.

4.1.5.1.1 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 12bitové 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.

4.1.5.1.2 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.

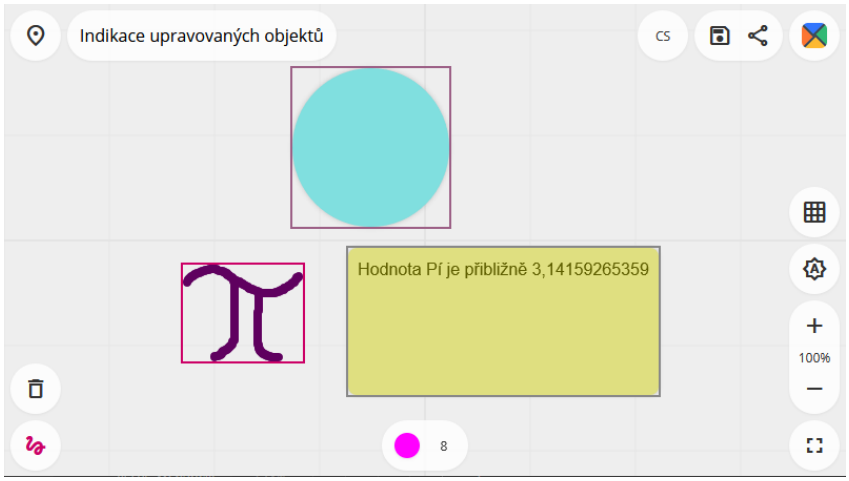
Pozadí poznámky i geometrického tvaru má rovněž stejné rozmezí, což je dáno tím, že zabírají relativně větší plochu, a navíc u poznámky je nutné vzít v potaz také kontrast nejen vůči plátnu, ale také vůči samotnému textu poznámky.

4.1.6 Indikace upravovaných objektů

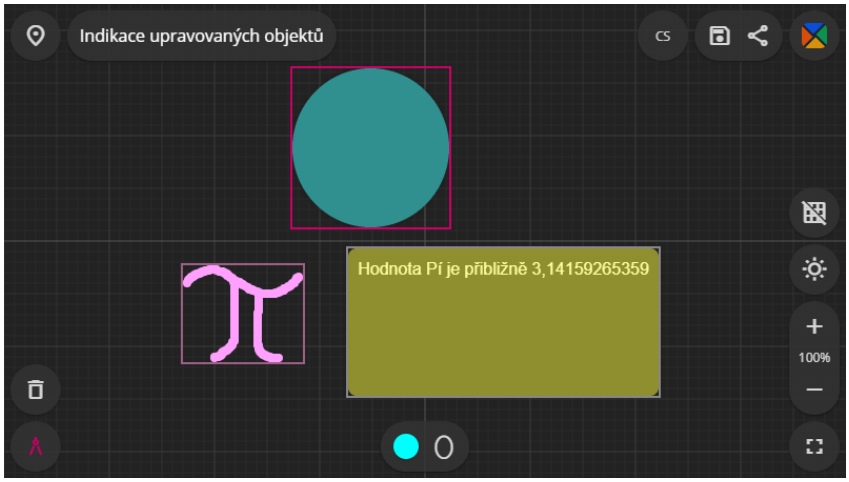
Aktuálně upravované objekty tabule lze jednoduše rozeznat díky tomu, že jsou během úpravy barevně ohraničeny. Barva ohraničení se liší na základě několika faktorů. Pokud se jedná o objekt upravovaný aktuálním uživatelem v aktuálním okně, je ohraničení sytě růžové. V případě, že jej sice upravuje aktuální 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 a uživatelské přívětivosti tabule. Vybraný objekt totiž žádný jiný uživatel nemůže vybrat do té doby, než původní uživatel svůj výběr zruší či se odpojí. Díky tomuto opatření tedy nemůže dojít k nečekaným výsledkům úpravy vybraného objektu tabule a to ani tím stejným uživatelem v jiném okně. Vybraný objekt je totiž spárován nejen se samotným uživatelem, ale také s danou instancí webu.

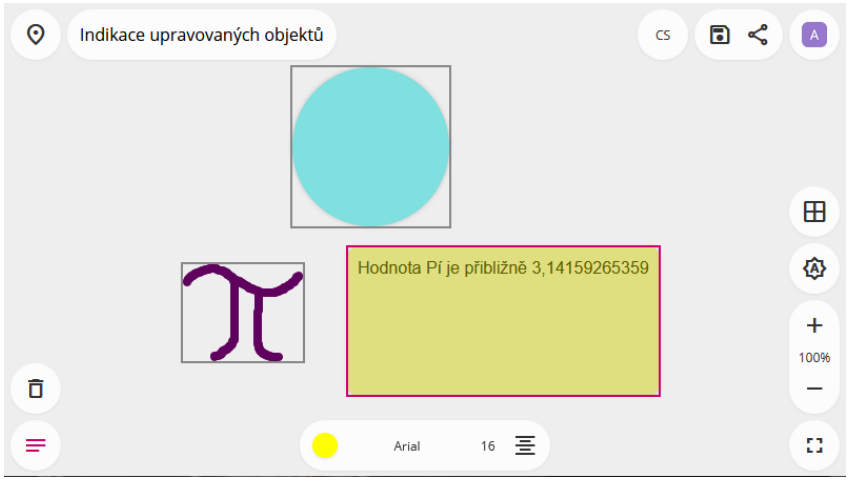
Následující obrázky 4.2–4.4 zobrazují pohled ze 3 různých instancí, z nichž první 2 pochází od stejného uživatele, pouze s rozdílem použitého prohlížeče a poslední poté pohled úplně jiného uživatele. Ve všech instancích je vždy vybrán alespoň jeden objekt.



Obrázek 4.2: Pohled uživatele X v okně A



Obrázek 4.3: Pohled uživatele X v okně B



Obrázek 4.4: Pohled uživatele Y v okně C

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. Data jsou poté na serveru průběžně ukládána do databáze o čemž dále důkladněji 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 UTF8). [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.

4.2.3 Udržování spojení

Pokud je spojení nečinné po dobu přibližně 50–60 vteřin, tak je spojení automaticky ukončeno. Tomu lze zabránit odesíláním `ping` požadavků, které poté automaticky vrací `pong` odpověď. Tímto způsobem je možné dlouhodobě udržovat spojení se serverem, tak aby nebylo nutné spojení kompletně obnovovat. [13]

Odesílání požadavků `ping` má smysl pouze při oboustranné nečinnosti v komunikaci. Pokud tedy uživatel či server s druhou stranou aktivně komunikují, není nutné přidávat další požadavky.

4.2.4 Důvody a řešení odpojení uživatele

Existuje několik možností, kvůli kterým může dojít k přerušení spojení. Pokud je přerušení očekávané a žádané, pak není požadováno ani opětovné spojení. V případě neočekávaného odpojení je však vhodné se snažit co nejdříve opět navázat komunikaci.

4.2.4.1 Odpojení na základě chybné komunikace

Po navázání připojení může klient na server odesílat libovolné data, které nemusí mít vždy očekávaný formát. Tyto data server může a nemusí zpracovat, je však vhodné jim věnovat pozornost.

K odeslání chybných dat může dojít jedině neznalostí jejich požadovaného formátu, což může být způsobeno tím, že vývojář klientské části aplikace nezná systém, se kterým pracuje, ale také může jít o pokus o podstrčení dat ze strany útočníka.

Pro zachování bezpečnosti je po přijetí těchto dat spojení s daným klientem přerušeno a dále jsou vypnuty také automatické pokusy o připojení. Klient nemá přístup zakázán úplně, ale vypnutím automatických pokusů o připojení jsou jednotlivé pokusy časově náročnější, což může případného útočníka odradit.

Pro zvýšení úrovně zabezpečení serveru by bylo možné uchovávat IP adresu či jiné identifikátory pro budoucí zamezení připojení určitým klientům. Toto řešení však přístupu útočníka nezabrání úplně, jelikož se toto opatření dá obejít smazáním cookies, změnou prohlížeče, IP adresy či zařízení.

4.2.4.2 Odpojení spojené s problémovým připojením k serveru

K přerušení spojení může dojít výpadkem internetu, proudu či operačního systému daného zařízení, a to jak na straně klienta, tak samotného serveru. Jelikož tento typ přerušení není způsoben žádným bezpečnostním opatřením serveru, má klient povoleno se automaticky co nejdříve opět připojit, jakmile to bude možné.

K automatickým pokusům o připojení dochází vždy v intervalu 5 vteřin. Pokud byl uživatel odpojen na základě ztráty připojení k internetu, tak k danému pokusu o spojení dochází okamžitě po obnovení připojení.

4.3 Správa dat

4.3.1 MySQL databáze

MySQL databáze je použita jako trvalé uložiště dat, které slouží mimo jiné jako záloha při výpadku spojení a obsahuje všechny data týkající se tabulí, objektů tabulí a také uživatelů. K zaznamenání všech potřebných dat využívá celkově 8 tabulek relačního datového modelu a jejich kompletní rozbor je popsán ve 3. kapitole v sekci 3.2 zaměřené na datovou analýzu systému.

Tabulky jsou v databázi vytvořeny serverem od jeho prvního spuštění a jsou dále využívány dle potřeby. Veškeré změny odeslané na server jsou následně pomocí Node.js balíčku `mysql` předány do databáze tak, aby vždy obsahovala aktuální stav, který distribuuje všem nově příchozím uživatelům. Po úspěšném provedení jedné či více operací nad danou databází obvykle následuje přeposlání souvisejících dat mezi další uživatele kromě (výjimečně také včetně) uživatele, který danou změnu inicioval.

Při prvním připojení klienta se do databáze ukládá unikátní řetězec Google účtu `sub` a dále, také pro všechny další připojení, čas přihlášení. Libovolné odpojení uživatele poté vede k uložení času odhlášení. Následující kroky se liší dle stránky, ze které se klient připojuje.

Pokud se jedná o stránku s tabulí, pak jsou z databáze získána data o dané tabuli, jejich objektech a pozici uživatele. Dále je možné přidávat, měnit nebo smazat objekty, měnit pozici uživatele či název tabule.

V případě stránky uživatele jsou získána data o všech tabulích uživatele, ale pouze jejich identifikátory a názvy. Následně je možné přidávat a smazat jednotlivé tabule či měnit jejich název přímo z této stránky. Přidání tabule je jediná operace, která zpětně posílá data také uživateli, který daný požadavek odeslal. Důvodem je to, že na stránce uživatele se u dané tabule objevují odkazy na možnost úpravy či zobrazení tabule a tyto odkazy obsahují řetězec, který je znám až po přidání tabule do databáze.

4.3.2 Importování obsahu tabule

4.3.2.1 Nahrání zdrojového souboru

Tabule umožňuje nahrát zdrojový soubor obsahující jednotlivé exportované objekty. Oblast těchto objektů je nejprve vycentrovaná na aktuální pozici uživatele a přidání objektů dále probíhá podobně jako v případě jednoho objektu. Objekty jsou po jednom odesílány na server a po přidání do databáze jsou tyto objekty distribuovány také zbylým uživatelům.

4.3.2.2 Nahrání obrázku

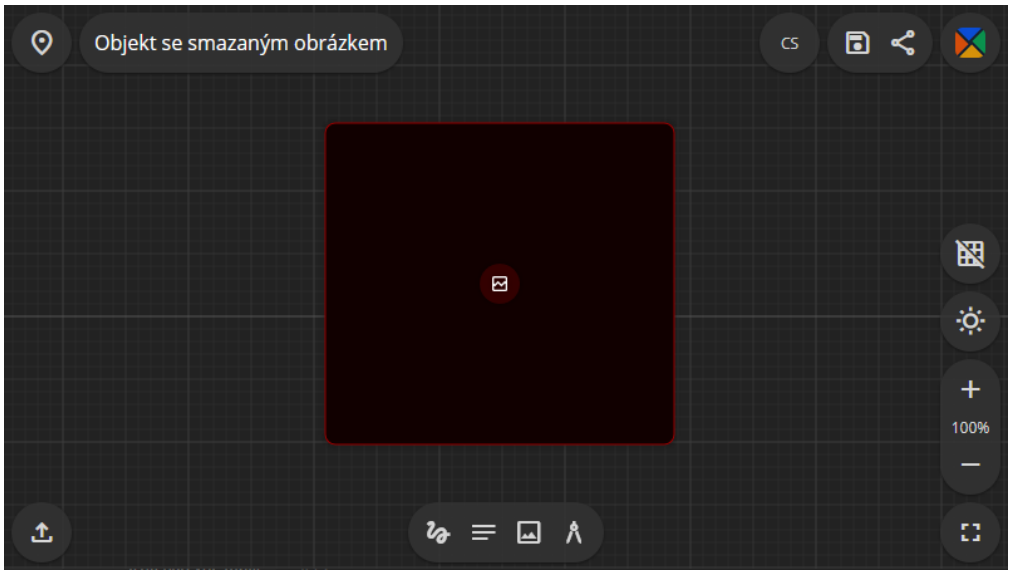
Pro nahrání obrázku je využita JavaScriptová Fetch API. Tato API umožňuje odesílat data na server mimo jiné ve formátu formulářových dat, které mohou zahrnovat také soubory. Serverem konkrétně v tomto případě je myšlený webový server, na kterém samotná webová stránka běží.

Před samotným odesláním požadavku je nutné zkontrolovat maximální velikost nahraného obrázku. Pokud přesahuje určitý limit (aktuálně stanoven na 10 MB), tak není umožněno jeho nahrání na server. V opačném případě dojde k zabalení souboru do formulářových dat a odeslání těchto dat na server pomocí POST požadavku.

Webový server poté zkontroluje, zda byly data odeslány ve správném formátu a opět také zda splňují daný velikostní limit. Následně provede kontrolu, zda se již daný soubor na serveru nenachází, tedy zda neexistuje soubor o stejné velikost a případně se stejným obsahem, který by na serveru již byl. Cílem je využít již uložených souborů na místo ukládání duplikátních souborů zabírajících cenné místo. V případě, že se daný soubor na serveru nenachází, dostane název náhodně vygenerovaného unikátního řetězce a je uložen do předem určené složky.

Výstupem serveru je poté relativní odkaz na umístění daného souboru, který je vrácen jako odpověď na požadavek odeslaný pomocí příkazu Fetch API. Tento odkaz je poté využit k vykreslení obrázku na tabuli a je také rozeslán pomocí WebSocket serveru dalším uživatelům tabule a uložen do databáze.

Objekty spojené s daným odkazem na tabuli zůstávají viditelné i po smazání daného obrázku. Tyto objekty však samotný obrázek nahrazují ikonou s piktogramem rozbitého obrázku, která indikuje jeho momentální nedostupnost. Tento stav je možné pozorovat na obrázku 4.5.



Obrázek 4.5: Ukázka objektu se smazaným obrázkem

4.3.3 Exportování obsahu tabule

Před exportováním obsahu tabule je okolo krajních bodů všech objektů ohraničená oblast, která je určena k exportu. Tato oblast je poté exportována ať už do zdrojového souboru či do obrázku ve výchozí úrovni přiblížení. Objekty jsou tedy zpracovány dle již zmíněných parametrů a jsou dále exportovány do vybraného typu souboru.

4.3.3.1 Exportování zdrojového souboru

Z již zpracovaných objektů jsou dále odebrány nepotřebné vlastnosti jako `token` určený ke kontrole přidávaného objektu a také `user` udávající id uživatele, který aktuálně objekt upravuje. Tyto vlastnosti v souboru ani při budoucím importu nejsou potřeba a jejich odebráním se do jisté míry zmenší velikost výsledného souboru.

Pro uložení do souboru ve formátu JSON jsou data převedena na řetězec pomocí příkazu `JSON.stringify()`. Následně je s pomocí tohoto řetězce vytvořen objekt Blob, který uchovává raw data a je možné se na něj odkázat pomocí příkazu `URL.createObjectURL()`, který vrací jeho URL adresu. [14] Tato adresa je poté vložena do atributu `href` HTML odkazu rovněž s atributem `download`, který umožňuje stáhnout libovolný soubor uvedený v atributu `href`. Pro vyvolání samotného dialogu uložení souboru je nad odkazem zavolána metoda `click()`, která simuluje kliknutí uživatele. Jelikož objekt Blob zůstává i po stažení stále v paměti, tak je pro jeho nepotřebnost z paměti uvolněn zavoláním `URL.revokeObjectURL()`.

4.3.3.2 Exportování obrázku

Pro export vybrané oblasti do obrázku je vytvořen speciální Canvas určený pouze pro tento účel. Tento Canvas není na stránce viditelný a slouží pouze k jednorázovému vykreslení obrázku. Po vykreslení vybrané oblasti je z tohoto Canvasu získán objekt Blob pomocí funkce `toBlob()`. Funkce přijímá mimo jiné parametr určující výsledný formát obrázku, tedy zda má být výsledný soubor s příponou bmp, jpg či png a dále také výslednou kvalitu obrázku v rozmezí 0–1. Výsledný objekt Blob je pak dále využit stejně jako v případě exportování zdrojového souboru, jehož postup je popsán výše.

4.3.4 Uložení uživatelských preferencí

Určité uživatelské nastavení není potřeba odesílat na server, jelikož toto nastavení může souviset se zařízením ze kterého toto nastavení provedl či může jít pouze o dočasnou volbu. Pro lokální uložení dat lze použít např. LocalStorage, který data ukládá lokálně přímo do prohlížeče nebo PHP session, které ukládá data na server, avšak pouze dočasně.

4.3.4.1 LocalStorage

LocalStorage je využit k uchování stavu mřížky a barevného motivu mezi jednotlivými návštěvami tabulí. Lokální uložení dat je v tomto případě vhodné z toho důvodu, že je platné pouze v daném prohlížeči a uživatel tedy může mít více instancí stejné stránky s jiným nastavením pro případné testování výsledné podoby tabule. Výhodou je také, že data zůstávají v prohlížeči do té doby, než je uživatel odstraní během čištění prohlížeče.

4.3.4.2 PHP Session

Ve výchozím stavu web poskytuje obsah v jazyce prohlížeče, avšak je možné si toto nastavení dočasně změnit. Konkrétně je na výběr mezi češtinou a angličtinou a každá změna je zaznamenána do PHP Session. Výhodou PHP Session je, že tyto data jsou opět platná pouze v daném prohlížeči, avšak oproti LocalStorage jsou tyto data platná jen po určitou dobu.

4.3.5 Sdílení tabule

Tabuli je kdykoli možné sdílet ať už pomocí samotného URL odkazu stránky či odesláním e-mailové zprávy. Web nabízí 2 možnosti sdílení pomocí e-mailu:

- **Vygenerovaná zpráva odeslaná serverem** – Dle jazyka ve kterém uživatel daný web aktuálně využívá, je vygenerována zpráva, kterou server odešle na zadaný e-mail. K odeslání e-mailu je využita PHP funkce `mail()`.
- **Manuálně psaná zpráva odeslaná uživatelem** – Zpráva je opět vygenerována dle jazyka uživatele, ovšem nyní je tato zpráva otevřena v e-mailovém klientu uživatele, skrze který má možnost tuto zprávu změnit a poté odeslat. K předání zprávy do e-mailového klienta je využitý odkaz s adresou obsahující řetězec `mailto:`.

Cílem je nabídnout uživateli snadnou a bezstarostnou variantu sdílení a pak také variantu s možností kompletní úpravy finální podoby e-mailu s určitou počáteční strukturou zprávy.

4.4 Uživatelské prostředí

4.4.1 Uživatelské rozhraní

4.4.1.1 Řešení responzivity

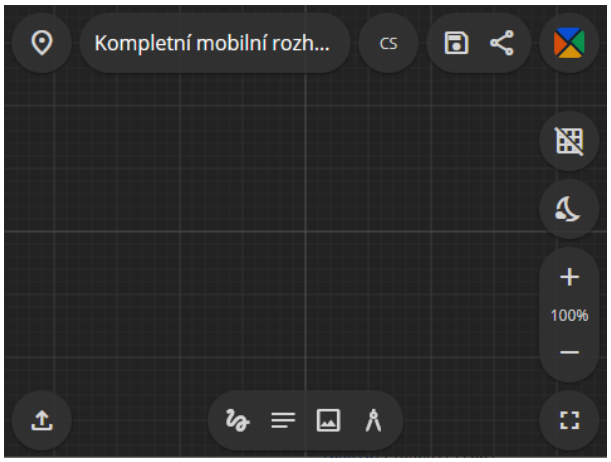
Responzivita je jednou ze základních součástí multiplatformního webu a v dnešní době je již považována za standard moderních webů. Aby se celý obsah webu vešel i na menší obrazovky je zapotřebí jej schovat za nějaké tlačítko či jeho část úplně vynechat.

V rámci řešení byly využity všechny výše uvedené možnosti. Rozhraní webu obsahuje povětšinou pouze ikony a jednoduché tlačítka, které po jejich stisknutí nabízí další funkce. V případě mobilních zařízení s opravdu malou obrazovkou poté úplně mizí prvky jako tlačítka přiblížení tabule či text s názvem tabule. Na dotykovém mobilu je nicméně možné zoomování pomocí tahu 2 prstů a text názvu tabule je pro takto malou obrazovku příliš veliký, takže by jeho zobrazení bylo spíše na škodu. Vynechání těchto 2 prvků však závisí pouze na velikosti samotného zařízení, a tedy dnes běžně používané telefony se mohou tomuto omezení úplně vyhnout.

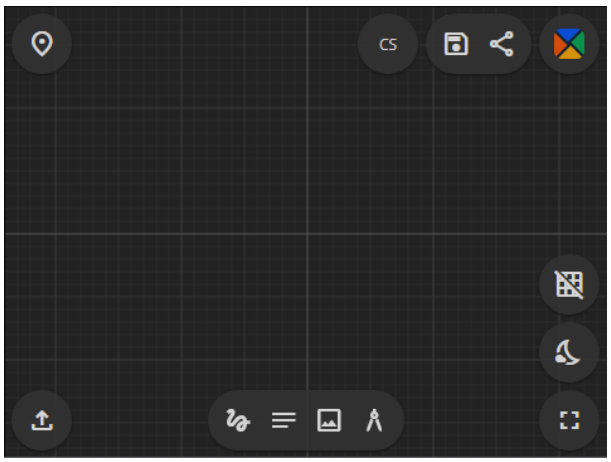
Odlišení prvků dle velikosti zařízení je implementováno pomocí CSS, konkrétně příkazem `@media`. Následující kód je využit přímo na webu a uvádí 2 třídy, které při splnění podmínek uvedených v parametrech příkazu `@media` schovají obsah, který je jimi označen.

```
@media only screen and (max-height: 359px) {
    .s-mobile-landscape--hidden {
        display: none;
    }
}
@media only screen and (max-width: 479px) {
    .s-mobile-portrait--hidden {
        display: none;
    }
}
```

Listing 4.3: CSS řešení responzivity webu



Obrázek 4.6: Kompletní mobilní rozhraní

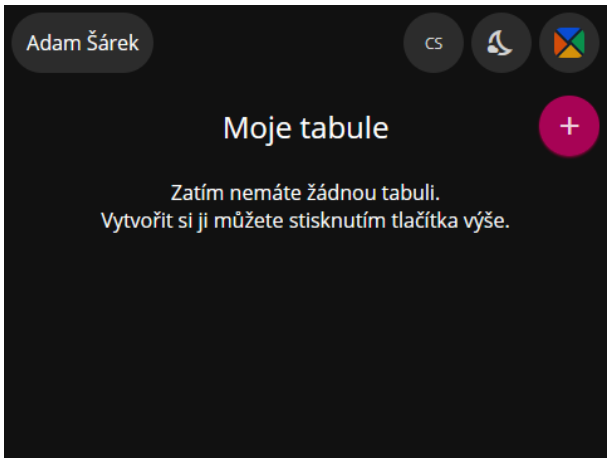


Obrázek 4.7: Částečné mobilní rozhraní

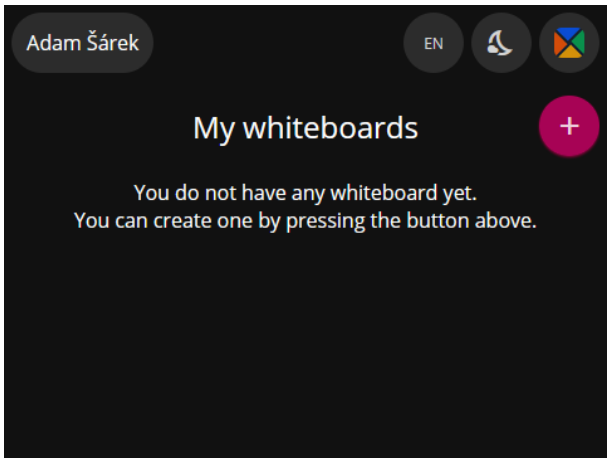
4.4.1.2 Podpora více jazyků

V rámci zpřístupnění webu zahraničním uživatelům je vhodné mít na webu i jinou než českou lokalizaci. Základem je podpora angličtiny, ale také dalších globálně nejpoužívanějších jazyků.

Načtení zdrojového souboru jazyka probíhá na webovém serveru, který daný jazyk přijímá buď z URL webu, PHP session záznamu nebo výchozího nastavení prohlížeče. Vybraný jazyk uživatelům nabízí kromě přeloženého uživatelského rozhraní také vygenerovanou zprávu při sdílení tabule. Obrázky níže zobrazují tu stejnou stránku nejprve v češtině a poté v angličtině.



Obrázek 4.8: Stránka v češtině



Obrázek 4.9: Stránka v angličtině

4.4.1.3 Jednotlivé stránky webu

4.4.1.3.1 Stránka uživatelského účtu

Stránka uživatelského účtu je určena pro spravování tabulí vytvořených daným uživatelem. Tabule je možné přidávat, měnit jejich název, a nebo je smazat. Každá tabule má vedle tlačítka na smazání také tlačítka odkazující na 2 různé režimy přístupu – editace a zobrazení.

Obrázek 4.10: Stránka uživatelského účtu

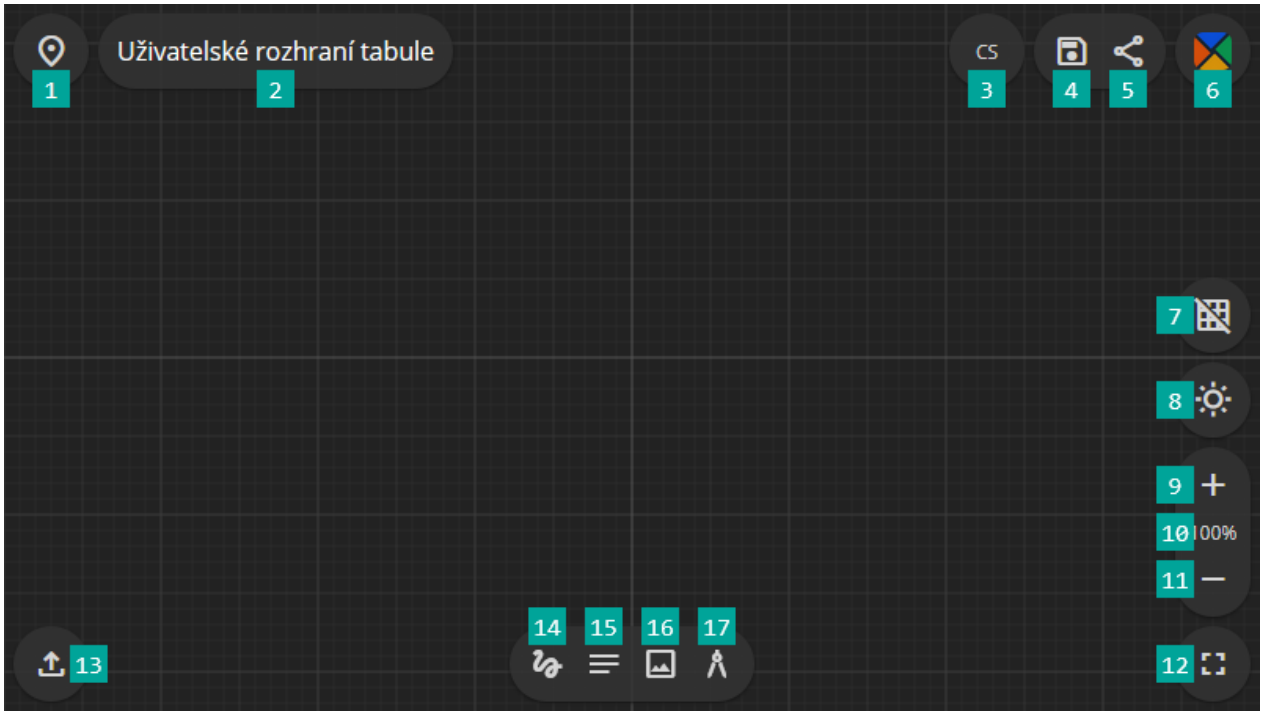
Přehled vyznačených prvků:

1. Jméno uživatele získané přihlášením do účtu Google
2. Tlačítko pro výběr jazyka – po stisknutí se objeví nabídka jazyků
3. Tlačítko pro přepnutí barevného motivu – po stisknutí se přepíná barevný motiv a na výběr je mezi tmavým, světlým a automatickým (světlý či tmavý dle lokálního času) motivem
4. Tlačítko pro zobrazení uživatelské nabídky – po stisknutí se objeví nabídka s možností přejít na stránku uživatelského účtu a dále také možnost odhlášení
5. Tlačítko pro přidání tabule – po stisknutí se objeví textové pole pro zadání názvu vytvářené tabule
6. Seznam vytvořených tabulí

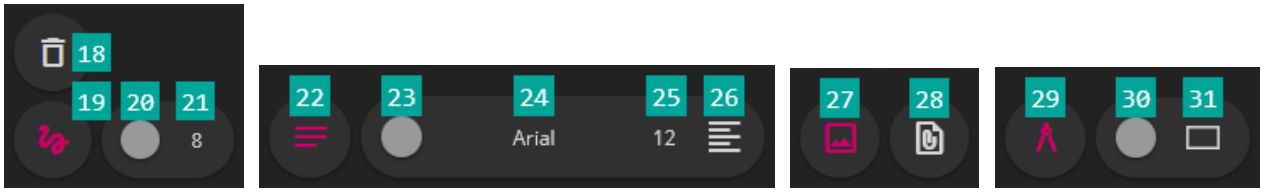
7. Název tabule – text lze po kliknutí upravit
8. Tlačítko odkazující na režim editace tabule
9. Tlačítko odkazující na režim zobrazení tabule
10. Tlačítko pro smazání tabule – po stisknutí se objeví dialog pro potvrzení smazání

4.4.1.3.2 Stránka tabule

Tato stránka je určena pro sdílenou práci uživatelů, kteří k tabuli přistupují ve dvou základních režimech. Režim editace nabízí možnost aktivní účasti na tvořeném obsahu tabule, zatímco režim zobrazení je určen pouze pro pasivní konzumaci vytvářeného obsahu. Režim zobrazení obsahuje téměř všechny prvky rozhraní, kromě prvků souvisejících s úpravou tabule.



Obrázek 4.11: Stránka tabule



Obrázek 4.12: Rozhraní úpravy kreslení, poznámky, obrázku a geometrického tvaru

Přehled vyznačených prvků – oba režimy:

1. Tlačítko pro přesun na určité místo tabule – po stisknutí se objeví nabídka významných míst tabule
2. Název tabule – text lze po kliknutí upravit
3. Tlačítko pro výběr jazyka – po stisknutí se objeví nabídka jazyků
4. Tlačítko pro export tabule – po stisknutí se objeví nabídka s možností exportu tabule do zdrojového souboru či obrázku
5. Tlačítko pro sdílení tabule – po stisknutí se objeví nabídka s možností zkopírování odkazu tabule či její sdílení e-mailem
6. Tlačítko pro zobrazení uživatelské nabídky – po stisknutí se objeví nabídka s možností přejít na stránku uživatelského účtu a dále také možnost odhlášení
7. Tlačítko pro přepnutí hustoty mřížky – po stisknutí se přepíná hustota mřížky a na výběr je mezi žádnou, poloviční a plnou mřížkou
8. Tlačítko pro přepnutí barevného motivu – po stisknutí se přepíná barevný motiv a na výběr je mezi tmavým, světlým a automatickým (světlý či tmavý dle lokálního času) motivem
9. Tlačítko pro přiblížení tabule
10. Tlačítko pro obnovení výchozí úrovně přiblížení tabule
11. Tlačítko pro oddálení tabule
12. Tlačítko pro zapnutí a zrušení režimu celé obrazovky

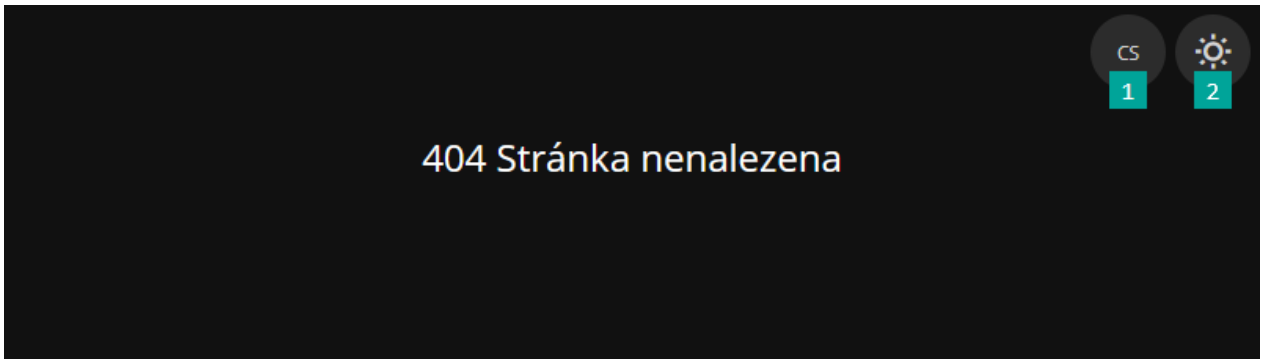
Přehled vyznačených prvků – režim úpravy:

13. Tlačítko pro nahrání zdrojového souboru
14. Tlačítko pro přidání objektu volného kreslení
15. Tlačítko pro přidání poznámky
16. Tlačítko pro přidání obrázku
17. Tlačítko pro přidání geometrického tvaru
18. Tlačítko pro smazání objektu
19. Tlačítko pro ukončení úpravy objektu volného kreslení

20. Tlačítko pro změnu barvy kreslené čáry
21. Tlačítko pro změnu tloušťky kreslené čáry
22. Tlačítko pro ukončení úpravy poznámky
23. Tlačítko pro změnu barvy písma poznámky
24. Tlačítko pro změnu rodiny písma poznámky
25. Tlačítko pro změnu velikosti písma poznámky
26. Tlačítko pro změnu zarovnání textu poznámky
27. Tlačítko pro ukončení úpravy obrázku
28. Tlačítko pro změnu obrázku
29. Tlačítko pro ukončení úpravy geometrického tvaru
30. Tlačítko pro změnu barvy geometrického tvaru
31. Tlačítko pro změnu typu geometrického tvaru

4.4.1.3.3 Chybová stránka

Stránka uvádí číslo a krátký popis chyby, která se při použití webu vyskytla. Řešení podporuje zobrazení chyb s číslem 400, 401, 403 a 404.



Obrázek 4.13: Chybová stránka

Přehled vyznačených prvků:

1. Tlačítko pro výběr jazyka – po stisknutí se objeví nabídka jazyků
2. Tlačítko pro přepnutí barevného motivu – po stisknutí se přepíná barevný motiv a na výběr je mezi tmavým, světlým a automatickým (světlý či tmavý dle lokálního času) motivem

Kapitola 5

Závěr

Cílem této práce bylo zanalyzovat existující řešení virtuální sdílené tabule, inspirovat se, navrhnout možné řešení a v rámci implementace vytvořit funkční webovou aplikaci, která bude podporovat současnou práci více uživatelů.

Stanovené cíle byly v bakalářské práci splněny. V první kapitole je detailně rozepsaná analýza vybraných webů, včetně bodů inspirace z hlediska postupů řešení a vybraných technologií. Další část je zaměřená na rozbor a návrh architektury systému, na které je celé řešení postaveno. Stanovení robustní škálovatelné architektury zajišťuje její dlouhodobou udržitelnost, jelikož je možné ji dále využít při rozšiřování aplikace. Samotná implementace poté pojednává o jednotlivých částech vývoje. Nejprve jsou zmíněny problémy a řešení spojené s vykreslováním tabule pro zachování plynulého a kontrastního obsahu. Důležitou součástí implementace je také propojení webové aplikace s infrastrukturou serveru a databáze, kde je mimo jiné zmíněné zabezpečení před nežádoucími daty a také řešení pro udržení dlouhodobého spojení mezi klientem a serverem. Další část se věnuje samotným datům a popisuje různé způsoby importu a exportu dat a následně také možnostem sdílení tabule. Na závěr je rozebráno uživatelské rozhraní včetně řešení responzivity pro podporu více typů zařízení a také obsah jednotlivých stránek webu pro jednodušší orientaci v dané aplikaci.

Aplikaci je možné dále rozšířit či vylepšit o nové technologie a postupy pro plynulejší a méně náročný běh a také vyšší bezpečnost pro další navyšování počtu uživatelů. Z hlediska optimalizace běhu aplikace je vhodné uvažovat o implementaci experimentální technologie OffscreenCanvas pro zajištění vyššího výkonu vykreslování. Pro náročnější operace je možné využít více vláken, které na sebe ve svých výpočtech nebudou čekat. Dalším možným vylepšením je změna formátu přenášených dat z JSON na binární, který má potenciál mít nižší velikost dat a díky čemuž by se snížily i minimální požadavky aplikace. V neposlední řadě je prostor také pro vyšší úroveň zabezpečení aplikace zavedením seznamu nepovolených IP adres, implementací kontroly CAPTCHA či dalších bezpečnostních opatření pro snížení rizika spojeného s potenciálním rozšiřováním uživatelské báze.

Literatura

1. *Main thread* [online] [cit. 2021-03-24]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Glossary/Main_thread`.
2. *Improving HTML5 Canvas Performance* [online] [cit. 2021-03-26]. Dostupné z: `https://www.html5rocks.com/en/tutorials/canvas/performance/#toc-batch`.
3. *CanvasRenderingContext2D.arc()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/arc`.
4. *CanvasRenderingContext2D.closePath()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/closePath`.
5. *CanvasRenderingContext2D.fill()* [online] [cit. 2021-03-26]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fill`.
6. *Introduction to the DOM* [online] [cit. 2021-03-24]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction`.
7. *OffscreenCanvas* [online] [cit. 2021-04-02]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/OffscreenCanvas`.
8. *OffscreenCanvas* [online] [cit. 2021-04-02]. Dostupné z: `https://www.caniuse.com/offscreencanvas`.
9. *About Node.js®* [online] [cit. 2021-04-05]. Dostupné z: `https://nodejs.org/en/about/`.
10. LOMBARDI, Andrew. *WebSocket*. Sebastopol (CA): 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`.
13. WANG Vanessa, Frank SALIM a Peter MOSKOVITS. *The Definitive Guide to HTML5 WebSocket*. Berlin: APress, 2013. ISBN 978-1430247401.
14. *Blob* [online] [cit. 2021-04-18]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/Blob`.