FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webová aplikace pro interaktivní myšlenkové mapy
Student:	Dominika Králiková
Vedoucí:	Mgr. Martin Podloucký
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Cílem bakalářské práce je vytvořit webovou aplikaci, která bude podporovat tvorbu interaktivních myšlenkových map.
Práce nejprve zhodnotí dostupný software pro tvorbu myšlenkových map. Na tomto základě vznikne webová aplikace, která nabídne uživatelsky přívětivé grafické rozhraní, které umožní tvorbu myšlenkové mapy, stylování jejích součástí pomocí tříd a nastavení interaktivních prvků. V rámci prezentačního režimu bude možné mapu prohlížet a zobrazovat přednastavená okna a informační boxy.
Hlavním smyslem aplikace je zamezit zahlcení informacemi při prvním pohledu na myšlenkovou mapu a zároveň zachovat přidanou informační hodnotu. Při vývoji bude použit klasický softwarový vývojový cyklus - analýza, design, implementace, testování.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 26. listopadu 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webová aplikace pro interaktivní myšlenkové mapy

Dominika Králiková

Katedra softwarového inženýrství
Vedoucí práce: Mgr. Martin Podloucký

13. května 2019

Poděkování

Ráda bych poděkovala vedoucímu své práce Mgr. Martinovi Podlouckému za spolupráci, dobré rady a přátelský přístup při její tvorbě. Také bych chtěla poděkovat své rodině, příteli a kamarádům za jejich bezmeznou podporu a trpělivost nejen při psaní této práce, ale také během celého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2019 ……

Odkaz na tuto práci

Králiková, Dominika. *Webová aplikace pro interaktivní myšlenkové mapy.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato bakalářská práce se zabývá vývojem webové aplikace pro tvorbu interaktivních myšlenkových map. Začátek práce je věnován definici základních pojmů a průzkumu existujících řešení. Hlavní náplní této práce je popis celého procesu vzniku aplikace – analýzy požadavků, návrhu uživatelského rozhraní, výběru technologií, vlastní implementace s využitím frameworku React a testování aplikace. Výstupem je funkční a uživatelsky přívětivá webová aplikace, která umožňuje vytvořit myšlenkovou mapu rozšířenou o interaktivní prvky a nastavit vzhled všech jejích součástí pomocí jednoduchého přiřazení tříd.

Klíčová slova myšlenková mapa, mentální mapa, interaktivní myšlenková mapa, diagram, interaktivní, prezentace, webová aplikace, interaktivní aplikace, React

Abstract

This bachelor thesis focuses on the development of a web application for creating interactive mind maps. At the beginning, the basic concepts are defined and existing software is evaluated. The main subject of this thesis is the description of the whole development process – requirements analysis, user interface design, technology selection, actual implementation with the help of the React framework and testing of the application. The output is a functioning and user-friendly web application that enables the user to create a mind map extended with interactive elements and to style all mind map elements with a simple class assignment.

Keywords mind map, interactive mind map, diagram, interactive, presentation, web application, interactive application, React

Obsah

Seznam obrázků

Seznam výpisů kódu

Úvod

Myšlenková mapa je nástroj, který podporuje kreativní myšlení. Má celou řadu využití – při učení, vymýšlení nových nápadů, ale také při prezentacích [1].

Uživatel může chtít, aby součástí prezentace pomocí myšlenkové mapy byly také nejrůznější obrázky, schémata, grafy, popisy apod. Z vlastní zkušenosti vím, že je potom problematické, aby i přes velké množství zahrnutých informací zůstala myšlenková mapa co nejpřehlednější. Posluchač by jinak mohl být zahlcen velkým množstvím informací ve spletitém a nepřehledném diagramu.

Dalším problémem, se kterým jsem se setkala při tvorbě myšlenkové mapy, je formátování a s ním spojený vzhled výsledku. Obvykle je potřeba rychle a jednotně nastavit vzhled velkého množství prvků. Zároveň je dobré mít možnost tento formát efektivně upravovat a vylepšovat.

V současné době existuje mnoho aplikací a programů, které se zabývají tvořením myšlenkových map, a celá řada z nich podporuje nějakou formu prezentačního režimu. Ani jeden z výše nastíněných problémů však není stávajícími dostupnými nástroji uspokojivě vyřešen.

V této práci proto na základě výsledků hodnocení existujícího software pro tvorbu myšlenkových map navrhnu a implementuji řešení těchto problémů – konkrétně prezentaci pomocí interaktivních prvků a formátování pomocí tříd.

Cíle práce

Cílem této bakalářské práce je vývoj webové aplikace, která uživateli umožní vytvořit myšlenkovou mapu rozšířenou o interaktivní prvky. Hlavním smyslem aplikace je zamezit zahlcení informacemi při prvním pohledu na myšlenkovou mapu, ale zároveň zachovat její přidanou informační hodnotu.

Aplikace dále nabídne možnost nastavení vzhledu prvků myšlenkové mapy pomocí tříd, stejně jako možnost prezentace mapy s využitím interaktivních prvků. Důraz bude též kladen na uživatelsky přívětivé grafické rozhraní aplikace umožňující tvorbu mapy se zaměřením na vizuální stránku výsledku.

Vzhledem k tomu, že cílem práce je především vývoj nové funkcionality, která se u současných dostupných nástrojů nevyskytuje, se práce nesoustředí na funkce, jež jsou u současných nástrojů běžné a zároveň nejsou pro základní fungování aplikace nutné. Mezi takové funkce patří například tvorba uživatelských účtů a ukládání map v databázi nebo některé podpůrné funkce využívané pří tvorbě mapy.

Členění práce

V kapitole 1 se budu věnovat definici pojmu „myšlenková mapa“, představím postup a principy tvorby myšlenkových map, jejich využití a základní pojmy s mapami spojené.

V kapitole 2 zhodnotím současné aplikace a programy pro tvorbu myšlenkových map. Nejprve uvedu programy, které jsou předmětem hodnocení, a dále pak definuji kritéria a postup hodnocení. Následně postupně podle kritérií popíši funkcionalitu současného software a zhodnotím ji.

Na začátku praktické části v kapitole 3 popíši funkční i nefunkční požadavky na aplikaci, stejně jako jednotlivé případy užití.

V kapitole 4 přistoupím k návrhu aplikace. Navrhnu uživatelské rozhraní, poté na základě informací z předešlých kapitol zvolím vhodné nástroje a technologie pro implementaci a navrhnu základní architekturu řešení.

V kapitole 5 popíši implementaci samotnou, podrobněji se zde budu věnovat také návrhu a funkcím jednotlivých částí aplikace. Zdůrazním i některé zajímavé body implementace.

Na závěr v kapitole 6 popíši techniky zvolené pro testování aplikace. Zhodnotím také výsledky testování.

Závěr věnuji výsledkům a vyhodnocení celé práce a jejím přínosům.

Kapitola 1

Myšlenková mapa

1.1 Definice myšlenkové mapy

Myšlenkové, nebo také mentální mapy vynalezl v 60. letech minulého století Tony Buzan, autor knih o myšlenkovém mapování, mozku a jeho použití [1, s. 11–13]. V knize Myšlenkové mapy spolu s Barrym Buzanem uvádějí následující definici: „*Myšlenková mapa je vizuální nástroj pro holistické, tedy celistvé myšlení, který podporuje všechny funkce mozku – především paměť, kreativitu, učení a veškeré přemýšlení.*“ [1, s. 42] Dále popisují také základní charakteristiky myšlenkové mapy:

1. obrázek ve středu myšlenkové mapy zachycuje hlavní téma mapy,
2. z centrálního obrázku vyrůstají větve – hlavní témata v přímé souvislosti s centrálním tématem, která se dále štěpí na další motivy,
3. každá větev je popsána klíčovým slovem nebo obrázkem. [1, s. 42]

V knize Mentální mapování Buzan [2, s. 15] dále uvádí, že myšlenkové mapy typicky využívají barvy, křivky, symboly, slova a zobrazení a mají paprskovitou strukturu, která se šíří z centra. To vše se řídí podle souboru jednoduchých a přirozených pravidel. Díky tomu je mentální mapa snadno zapamatovatelným schématem, které funguje v souladu s přirozeným pracováním lidského mozku.

1.2 Princip a přínosy myšlenkového mapování

Buzan uvádí, že: „*mentální mapa je nejsnadnějším prostředkem, jak dostávat informace do našeho mozku a jak z něj informace dostávat ven – je tvůrčím a efektivním způsobem dělání poznámek, který doslova ‚mapuje‘ naše úvahy.*“ [2, s. 14] Je tomu tak proto, že tvorba myšlenkových map respektuje mimo jiné také následující základní principy fungování mozku.

- **Vizuální vnímání** – přirozená schopnost mozku, obvykle si mnohem lépe pamatujeme informace podané obrázkovou formou. [2, s. 16]
- **Rozdělení mozku na pravý a levý** – mozková kůra rozděluje úkoly do dvou hlavních skupin:
  - **úkoly pravého mozku** – rytmus, prostorové vědomí, představivost, snění, barva, dimenze a úkoly vyžadující holistický přístup, tedy povědomí o dané věci jako o celku,
  - **úkoly levého mozku** – slova, logika, čísla, posloupnosti, přehledy, analýza.

  Spoléháním pouze na jednu hemisféru drasticky omezujeme potenciál mozku. Mentální mapy zaměstnávají obě strany mozku (zobrazení, barva a představivost pro pravou, slova, čísla a logika pro levou) a lépe tak využívají jeho možnosti. [2, s. 44, 45, 51]
- **Paprskovité myšlení** – mozek nemyslí lineárně, ale multilaterálně, paprskovitě. Přemýšlí organicky, ve strukturách podobných přírodním formám. Myšlenkové mapy opakují a napodobují paprskovité myšlení, které posiluje přirozené fungování mozku. Asociace a vztahy mezi myšlenkami pak pomáhají dělat velké pokroky ve snaze něco pochopit či vymyslet. [1, s. 27, 41, 42]

Platí, že čím kreativnější jsme při tvorbě myšlenkové mapy, tím lépe. Použití barev, obrázků a perspektivy pro vytvoření výtvarně povedeného výtvoru zvyšuje účinnost myšlenkové mapy. [1, s. 42]

Buzan [2, s. 15] také uvádí přínosy myšlenkových map, patří mezi ně například rozvoj komunikačních a organizačních schopností, rozvoj tvořivosti, úspora času, lepší zvládání problémů a lepší koncentrace, zlepšení paměti a efektivity učení.

1.3 Postup tvorby myšlenkové mapy

Buzan [2, s. 20] uvádí sedm kroků k vytvoření myšlenkové mapy. Při tvorbě bychom měli:

1. začít uprostřed čistého papíru – dáme mozku svobodu působit všemi směry a vyjadřovat se svobodněji a přirozeněji,
2. vyjádřit ústřední představu v obrázkem – názorné zobrazení má hodnotu tisíce slov a pomůže nám zapojit naši představivost, je zajímavější, udržuje soustředění a motivuje mozek k činnosti,
3. po celou dobu používat různé barvy – jsou pro mozek podnětné, mapy jsou živější a tvorba je zábavnější,

4. k centrálnímu obrázku připojit hlavní větve, k nim větve druhé úrovně, dále větve třetí úrovně atd. – mozek pracuje pomocí asociací, rád si spojuje věci dohromady, propojení usnadňuje zapamatování a porozumění,

5. větve zakreslovat jako křivky, ne jako přímky – rovné čáry jsou pro mozek nudné, organický tvar větví je pro mozek poutavější,

6. pro každou linku použít jen jedno klíčové slovo nebo slovní spojení – klíčová slova zvyšují účinnost a flexibilitu mentálních map,

7. používat různá vyobrazení na celé ploše mapy – obrázek má hodnotu tisíce slov.

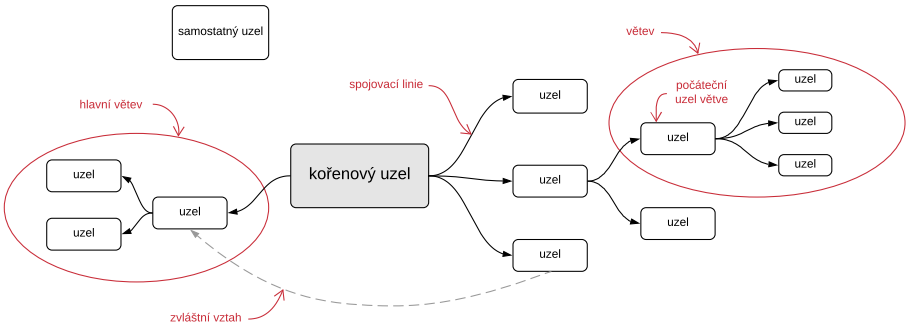
1.4 Použití myšlenkových map

Použití myšlenkových map je velmi široké. Podle [1, s. 89–113] spočívá základní využití v tréninku paměti, podpoře tvůrčího myšlení, usnadnění rozhodování a organizaci myšlenek jiných lidí (např. při tvorbě zápisků). Dále [1, s. 117–163] je možné využití myšlenkových map při sebeanalýze, plánování, studiu a učení, pracovních poradách, prezentacích, managementu.

1.5 Definice pojmů

Komplexní terminologie v oblasti myšlenkových map není pevně zavedená. Proto je v tomto místě nutné zadefinovat základní pojmy pro orientaci v rámci myšlenkové mapy, na které se budu v práci dále odkazovat.

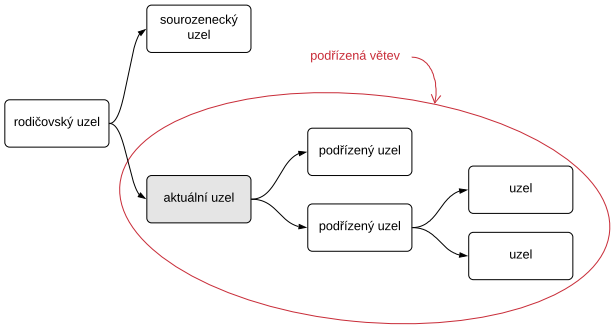
- pojmy související s obecnými prvky mapy (viz schéma na obrázku 1.1)
  - **uzel** (node) – jedna myšlenka nebo téma v rámci myšlenkové mapy
  - **spojovací linie** (line, linie, vztah) – vztah mezi dvěma uzly, který vyjadřuje buď hierarchii v rámci myšlenkové mapy, nebo vzájemnou souvislost dvou uzlů, které spolu nesousedí v rámci hierarchie (v tomto případě také „**zvláštní vztah**“)
  - **větev** (branch) – soubor uzlů a jejich vzájemných vztahů, začíná v libovolném uzlu myšlenkové mapy (tj. v „**počátečním uzlu větve**“) a zahrnuje celé rozvětvení pod ním (všechny uzly a jejich vztahy), „**hlavní větev**“ začíná v uzlu o jednu úroveň pod kořenovým uzlem
  - **úroveň** (level) – stupeň zanoření vzhledem ke kořenovému uzlu
  - **kořenový uzel** (root, central node, centrální uzel) – centrální myšlenka mapy, nemá rodičovský uzel



Obrázek 1.1: Obecné prvky mapy [vytvořeno autorem]

  - **samostatný uzel** (floating node, plovoucí uzel) – uzel, který nemá rodičovský uzel, ale zároveň není centrální myšlenkou mapy

- pojmy relativní vzhledem k vybranému uzlu (viz schéma na obrázku 1.2)
  - **aktuální uzel** – vybraný uzel
  - **rodičovský uzel** (parent, rodič, nadřazený uzel) – uzel o jednu úroveň nad aktuálním uzlem
  - **podřízený uzel** (child, potomek, přímý podřízený uzel) – uzel o jednu úroveň pod aktuálním uzlem
  - **sourozenecký uzel** (sibling) – uzel se stejným rodičem
  - **podřízená větev** (child branch) – větev s počátkem v aktuálním uzlu – tj. soubor aktuálního uzlu, všech jeho potomků a jejich podřízených větví (včetně vzájemných vztahů)



Obrázek 1.2: Relativní prvky mapy [vytvořeno autorem]

Kapitola 2

Zhodnocení existujících řešení

V rámci analýzy problému jsem provedla průzkum a zhodnocení dostupného software pro tvorbu myšlenkových map.

Za dostupný software považuji takový, který je k dispozici zcela zdarma nebo existuje jeho neplacená verze (hodnocení se pak týká pouze funkcí, které jsou součástí této neplacené verze). Průzkum se netýká placeného software, a to ani takového, který nabízí neplacenou časově omezenou zkušební verzi. Dostupným software může být jak desktopová, tak webová aplikace.

Důvodem pro omezení na neplacený software je velké množství existujících aplikací, jak placených, tak neplacených. Aplikace, která vznikne na základě tohoto průzkumu, je zamýšlena jako neplacená. Proto jsem se zaměřila na průzkum přímé konkurence, což mi umožnilo zúžit okruh zkoumaných aplikací.

2.1 Metodika vyhledání existujících řešení

Průzkum existujících řešení jsem provedla v rozmezí 20.11.2018 – 20.01.2019 pomocí vyhledávání na `google.com`. Konkrétně jsem prověřila vždy první stránku výsledků vyhledávání pro následující klíčová slovní spojení:

- mind maps online
- mind maps
- interactive mind maps
- interactive diagrams
- mind map tool
- mind map presentation

Do průzkumu jsem zařadila jak software přímo odkazovaný z výsledků, tak i software zmíněný v odkazovaných článcích (zejména v těch, které obsahovaly seznamy software pro tvorbu myšlenkových map).

2.2 Hodnocený software

Z vyhledávání vznikl soubor 21 dostupných nástrojů, jejichž funkcionalitu jsem prozkoumala a zhodnotila z několika hledisek. V seznamu níže je uvedeno jméno nástroje, odkaz na webovou stránku nástroje (pokud existuje) a klíčové slovní spojení, při jehož vyhledávání byl nástroj nalezen (pokud byl nástroj odkazován častěji, je uveden první odkaz).

- MindMup (`www.mindmup.com`) [3] – z vyhledávání „mind maps online“
- Coggle (`coggle.it`) [4] – z vyhledávání „mind maps online“
- bubbl.us (`bubbl.us`) [5] – z vyhledávání „mind maps online“
- GoConqr (`www.goconqr.com`) [6] – z vyhledávání „mind maps online“
- RealtimeBoard (`realtimeboard.com`) [7] – z vyhledávání „mind maps online“
- Lucidchart (`www.lucidchart.com`) [8] – z vyhledávání „mind maps online“
- MindMeister (`www.mindmeister.com`) [9] – z vyhledávání „mind maps online“
- Canva (`canva.com`) [10] – z vyhledávání „mind maps online“
- Venngage (`venngage.com`) [11] – z vyhledávání „mind maps online“
- FreeMind [12] – z článku ve vyhledávání „mind maps online“
- Text2MindMap (`tobloef.com/text2mindmap/`) [13] – z článku ve vyhledávání „mind maps online“
- Sketchboard (`sketchboard.me`) [14] – z článku ve vyhledávání „mind maps online“
- Creately (`creately.com`) [15] – z článku ve vyhledávání „mind maps“
- Mind42 (`mind42.com`) [16] – z článku ve vyhledávání „mind maps“
- Prezi (`prezi.com`) [17] – z vyhledávání „interactive mind maps“
- WiseMapping (`www.wisemapping.com`) [18] – z článku ve vyhledávání „interactive mind maps“
- Draw.io (`www.draw.io`) [19] – z vyhledávání „interactive diagrams“
- Mindmap Maker (`app.mindmapmaker.org`) [20] – z vyhledávání „mind map tool“

- Blumind [21] – z článku ve vyhledávání „mind map tool“
- Visual Understanding Environment (`vue.tufts.com`) [22] – z článku ve vyhledávání „mind map tool“
- MindMaster (`www.edrawsoft.com/mindmaster/`) [23] – z článku ve vyhledávání „mind map tool“

2.3 Hodnocení dostupných řešení

Funkcionalitu dostupného software jsem prozkoumala a zhodnotila z následujících hledisek:

- způsob tvorby kostry myšlenkové mapy
- možnosti nastavení vzhledu myšlenkové mapy
- možnosti prezentace hotové myšlenkové mapy
- možnosti zvýšení přehlednosti myšlenkové mapy
- přehlednost uživatelského rozhraní
- technologické řešení zobrazení myšlenkové mapy (pouze v případě webových aplikací)

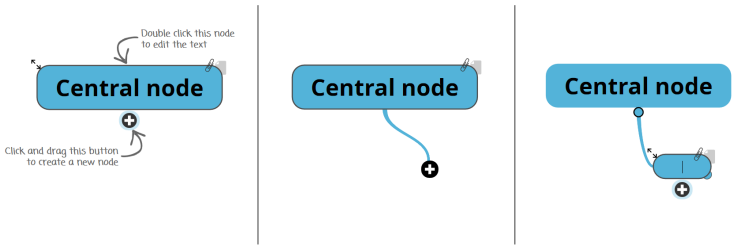
2.3.1 Způsob tvorby kostry myšlenkové mapy

V rámci tohoto hlediska zkoumám různé způsoby, jakými je možné myšlenkovou mapu vytvořit. Předmětem hodnocení je především metoda tvoření uzlů a jejich propojování a metoda jejich umisťování a přesouvání.

Tvoření uzlů Metoda tvoření uzlů je jeden z nejdůležitějších faktorů, které určují použitelnost nástroje. Volba vhodné metody závisí na tom, co uživatel od nástroje očekává – zda především rychlé vytvoření mapy, nebo spíše kreativní tvorbu se zaměřením na vzhled výsledku.

- metody umožňující tvorbu uzlu (sourozeneckého nebo podřízeného) spojené s automatickým rozmístěním nových uzlů (a případným posunem existujících uzlů) – vhodné pro rychlou tvorbu mapy
  - klávesové zkratky (MindMup [3], MindMeister [9], FreeMind [12])
  - klikání na tlačítka v menu (MindMup [3], MindMaster [23]) – uživatel může ztratit přehled o právě označeném uzlu
  - klikání na tlačítka přímo na uzlu (Coggle [4], RealtimeBoard [7], Mind42 [16])

- metody umožňující manuální rozložení uzlů po kreslící ploše – vhodné pro tvorbu mapy se zaměřením na vizuální stránku výsledku
  - tažení speciálního tlačítka z aktuálního uzlu, nový podřízený uzel vznikne na místě upuštění tlačítka, viz obrázek 2.1 (GoConqr [6], Mindmap Maker [20])
  - výběr tvaru v nabídce a jeho přetažení na kreslící plochu, následné vytvoření příslušných spojů mezi uzly (Lucidchart [8], Draw.io [19], Sketchboard [14])
  - prostý výběr různých tvarů, čar a textových polí v nabídce, jejich umístění na kreslící plochu a následná úprava do podoby myšlenkové mapy (Canva [10], Venngage [11], Prezi [17]) – velmi časové náročné
- automatické vytvoření celé mapy podle odsazení textu v textovém souboru, text je obsahem jednotlivých uzlů, uzly jsou rozloženy automaticky (Text2MindMap [13]) – vhodné pro rychlou tvorbu mapy



Obrázek 2.1: Postup tvoření nového uzlu v nástroji GoConqr [6, snímek autora]

Přesouvání uzlů Pokud je pro uživatele důležitá vizuální stránka výsledku, pak je přesouvání uzlů jedním z klíčových prvků celé tvorby myšlenkové mapy. Mezi nástroji, které jej podporují, se pak objevují různé přístupy k přesouvání podřízených uzlů.

- přesun celé podřízené větve relativně k přesunutému uzlu (MindMup [3], RealtimeBoard [7], Coggle [4]) – rychlejší varianta
- přesun pouze vybraného uzlu (GoConqr [6], Lucidchart [8])

2.3.2 Možnosti nastavení vzhledu myšlenkové mapy

Toto hledisko zkoumá různé možnosti nastavení vizuálních vlastností uzlů, spojovacích linií a textu – jaké vlastnosti je možné nastavit a na jaké hodnoty.

Dále zahrnuje také různé způsoby, které průběh tohoto nastavení ulehčují a zrychlují.

Obecné možnosti nastavení vzhledu Téměř všechny nástroje podporují nastavení obdobných vlastností, liší se pouze v drobných detailech. Z tohoto důvodu zde vlastnosti shrnuji obecně, bez uvedení příslušných nástrojů, které tuto funkcionalitu podporují.

V rámci nastavení vzhledu uzlu je samozřejmostí nastavení jeho barvy. Možné je rovněž nastavení barvy okraje uzlu a jeho tloušťky, případně stylu (tečkovaný, čárkovaný, plný…). Některé nástroje podporují nastavení tvaru uzlu (obdélník, elipsa, kruh…) a jeho velikosti. Občas je možné volit, jestli má být uzel reprezentován jako tvar nebo jako linie s popiskem.

Některé nástroje podporují nastavení vzhledu spojovacích linií. Je možné nastavit barvu, tloušťku, styl (tečkovaná, čárkovaná, plná…). Občas je možné přidat také šipku na libovolný konec. Nástroje umožňují volit mezi přímkou, křivkou nebo lomenou čárou. Některé přímo umožňují také nastavení tvaru linie. Možné bývá rovněž přidání popisku k linii.

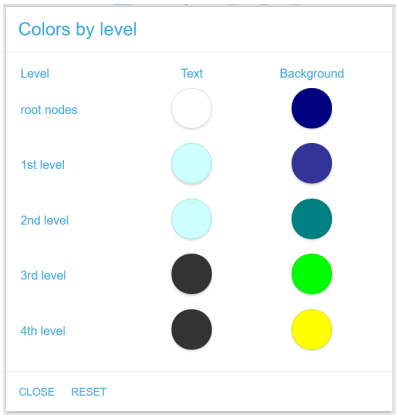
Možnosti nastavení vzhledu textu jsou většinou velmi široké. Uživatel může volit font, velikost písma, zarovnání, barvu. Je možné nastavit tučné písmo, kurzívu nebo podtržení.

Způsoby urychlení nastavení vzhledu Nastavení vzhledu by nemělo trvat příliš dlouho. Je proto nutné, aby jej aplikace umožňovala nějakým způsobem urychlit.

- použití existujících stylů – rychlé řešení, ale ne vždy bude nabídka odpovídat požadavkům uživatele
  - šablony a hotové naformátované prvky (Canva [10], Venngage [11], Prezi [17])
  - přednastavené globální styly – nástroje pak nastavují styly automaticky buď podle úrovní, nebo podle jednotlivých větví (MindMup [3], MindMaster [23])
  - přednastavené styly pro jednotlivé uzly (MindMeister [9])
- použití vlastních stylů – důležité, pokud má mít uživatel co největší volnost při nastavování vzhledu prvků
  - propagace změny uzlu celé podřízené větvi (Coggle [4], bubbl.us [5], Mindmap Maker [20])
  - výběr více uzlů naráz a jejich hromadné nastavení (Lucidchart [8], Draw.io [19]) – nutný manuální výběr uzlů
  - zkopírování stylu uzlu a jeho nastavení dalším uzlům (Lucidchart [8], Draw.io [19]) – styl nelze uložit

– vytvoření globálního stylu – určení výchozího vzhledu pro určité skupiny uzlů (jednotlivé větve, jednotlivé úrovně, centrální uzel a ostatní uzly), viz obrázek 2.2 (MindMup [3], Lucidchart [8], Blumind [21], Text2MindMap [13]) – ne vždy je formátování podle větví nebo podle úrovní dostačující

– vytvoření šablony stylu pro jednotlivé uzly – šablona je uložená a je možné ji aplikovat na libovolný uzel (FreeMind [12]) – po změně stylu se změna nepropaguje na již nastavené uzly



Obrázek 2.2: Nastavení globálního stylu v nástroji MindMup [3, snímek autora]

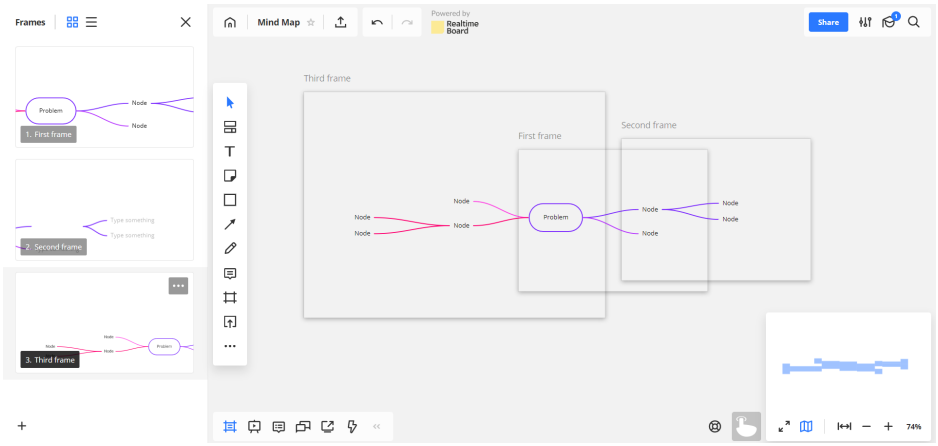
2.3.3 Možností prezentace hotové myšlenkové mapy

Z hlediska možností prezentace je zkoumáno, jestli nástroj nabízí nějaký způsob pohledu na myšlenkovou mapu, který může sloužit pro prezentaci mapy dalším osobám. Pokud nějaký takový pohled existuje, jsou hodnoceny jeho vlastnosti, případně složitost jeho použití.

- prezentace přímo v rámci nástroje není vůbec možná – nahrazení exportem (GoConqr [6], FreeMind [12], Text2MindMap [13])
- zobrazení celé mapy v režimu celé obrazovky – mapu je možné upravovat (MindMup [3], Draw.io [19]) – nevhodný přístup
- zobrazení celé mapy v režimu celé obrazovky – mapu není možné upravovat (Coggle [4], bubbl.us [5], Lucidchart [8]) – nemusí být dostačující, velká mapa může být nepřehledná, posluchači mohou mít problém se zorientovat
- interaktivní rozbalování a zabalování větví po jednotlivých úrovních (Coggle [4], MindMup [3]) – základní varianta, je interaktivní, je ale nutné rozbalit/zabalit všechny podřízené uzly naráz

- tvorba prezentace nad existující myšlenkovou mapou pomocí manuálního označení rámců (oblastí myšlenkové mapy), které jsou během prezentace zobrazeny a přiblíženy v předem definovaném pořadí, viz obrázek 2.3 (RealtimeBoard [7], MindMeister [9], MindMaster [23], Visual Understanding Environment [22]) – zajistí hladký průběh a vhodnou prezentaci informací, není interaktivní, prezentace musí být předem připravená a příprava může být složitá

- odkrytí doplňujícího textu a podřízených uzlů až po přiblížení uzlu v rámci prezentace (při pohledu na mapu tak jsou viditelné pouze nejdůležitější uzly), prezentace je doplněná o animace a odehrává se v předem daném pořadí (Prezi [17]) – velmi přehledná prezentace, která předá všechny potřebné informace, není problém s přehledností při celkovém pohledu na mapu, prezentace však není interaktivní, nástroj neslouží přímo k tvorbě myšlenkových map a práce s ním je náročná



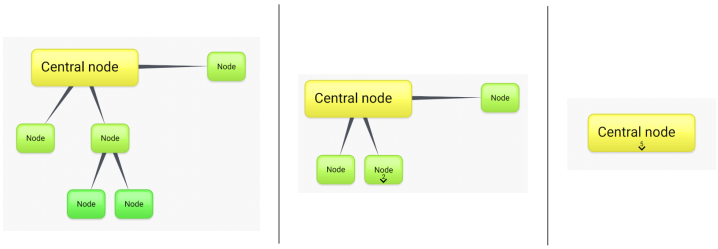
Obrázek 2.3: Nastavení rámců pro prezentaci v nástroji RealtimeBoard [7, snímek autora]

2.3.4 Možnosti zvýšení přehlednosti myšlenkové mapy

Přehlednost myšlenkové mapy závisí především na rozložení uzlů a nastavení jejich vzhledu (barvy, tvaru, velikosti). V rámci tohoto hlediska budou hodnoceny další možnosti, pomocí kterých je možné zvýšit přehlednost mapy.

- poznámky a komentáře k uzlům – s formátovatelným textem (Mindmap Maker [20], Blumind [21]) nebo nikoli (MindMup [3], GoConqr [6]) – nebývají viditelné během prezentace, není možné měnit vzhled poznámky jako celku

- rozbalení a zabalení větví po jednotlivých úrovních, viz obrázek 2.4 (MindMup [3], Coggle [4], bubbl.us [5]) – užitečné, ale může být poměrně chaotické, než se uživatel dostane do správné konfigurace rozbalených a zabalených částí
- vytvoření více vrstev a jejich zobrazování/schovávání (Lucidchart [8], Draw.io [19]) – tvorba časově náročná
- rozdělení informací do více stránek (Draw.io [19], Canva [10]) – ztráta přidané informační hodnoty diagramu
- speciální prezentační režim (Prezi [17], RealtimeBoard [7]) – bez schovávání informací nezpřehledňuje mapu jako takovou, se schováváním informací se blíží ideálnímu stavu



Obrázek 2.4: Postupné zabalení větve v nástroji bubbl.us [5, snímek autora]

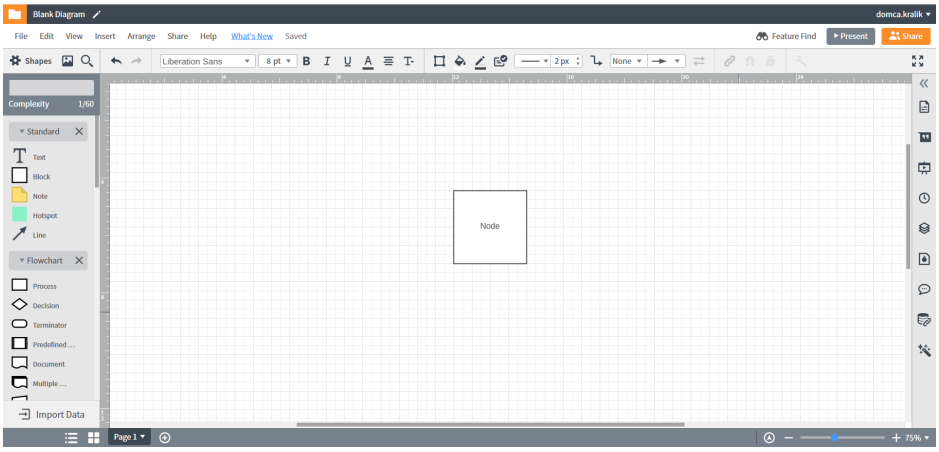
2.3.5 Přehlednost uživatelského rozhraní

V rámci tohoto hlediska jsou popsány a zhodnoceny přístupy k tvorbě grafického uživatelského rozhraní. Předmětem hodnocení je zejména přehlednost a intuitivnost výsledného rozhraní.

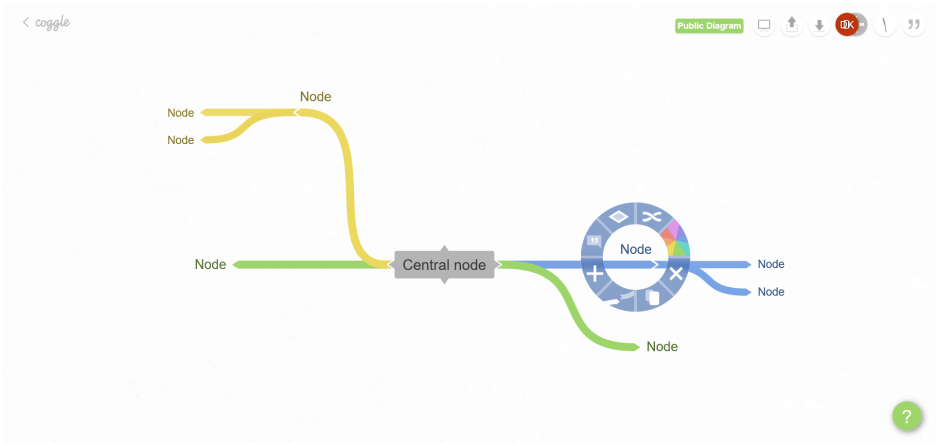
Každý nástroj má své vlastní rozhraní, nicméně z obecného hlediska se setkáváme s dvěma hlavními přístupy, které nástroje v různé míře kombinují.

- obsáhlé menu nad kreslící plochou nebo po jejích stranách, které ovládá obecnou funkcionalitu i prvky mapy, ovládání se vztahuje k prvku vybranému na kreslící ploše (MindMup [3], Lucidchart [8]) – má tendenci být nepřehledné, funkce jsou často schovány a orientace je náročnější, uživatel snadno ztratí přehled, jakého uzlu se úpravy týkají
- malá lokální menu u jednotlivých prvků mapy, která se zobrazí po kliknutí na prvek a ovládají pouze jeho funkce, jsou doplněná o malé globální menu s obecnou funkcionalitou (Coggle [4], bubbl.us [5]) – intuitivnější přístup, je patrné, kterého prvku se funkce týkají

Konkrétní příklady aplikací s kvalitním a uživatelsky přívětivým rozhraním jsou Lucidchart [8], Draw.io [19] a MindMaster [23] (přístup s obsáhlým hlavním menu, viz obrázek 2.5), případně Coggle [4] a Sketchboard [14] (přístup s lokálními nabídkami, viz obrázek 2.6).



Obrázek 2.5: Uživatelské rozhraní nástroje Lucidchart [8, snímek autora]



Obrázek 2.6: Uživatelské rozhraní nástroje Coggle se zobrazeným menu pro editaci uzlu [4, snímek autora]

2.3.6 Technologické řešení zobrazování myšlenkové mapy

Webové aplikace jsou zkoumány a hodnoceny také z hlediska jejich technologického řešení. Patří sem zejména řešení vykreslení celé myšlenkové mapy (zejména uzlů a spojovacích linií) – jestli jsou použity klasické HTML ele-

menty, jaké elementy to jsou a jestli použitá technologie nějak ovlivňuje aplikaci jako takovou.

- použití HTML elementů (především `div`, `svg` a `canvas`) – přístupy vzájemně ekvivalentní, pomocí každého přístupu je možné vytvořit dobře vypadající a správně fungující myšlenkovou mapu
  - reprezentace uzlů jako kombinace běžných HTML objektů (`div`, `span`, `p`), linie jako prvky jediného `svg` na pozadí (MindMup [3], Coggle [4], MindMeister [9]) nebo jako samostatné elementy `canvas` (bubbl.us [5], Mindmap Maker [20], Mind42 [16])
  - reprezentace celé myšlenkové mapy jako jednoho `svg` (GoConqr [6], Sketchboard [14], Draw.io [19], WiseMapping, [18])
  - reprezentace celé myšlenkové mapy jako jednoho elementu `canvas` (RealtimeBoard [7], Lucidchart [8], Text2MindMap [13], Prezi [17])
  - použití `svg` pro veškeré tvary a kombinace běžných HTML objektů (`div`, `p`, `span`) pro textová pole (Canva [10], Venngage [11])
- použití Adobe Flash (Creately [15]) – jediné řešení, které se nezdá být vhodné, aplikace s ním je pomalejší a uživatelské rozhraní je rozmazané

2.4 Zhodnocení výsledků průzkumu

Na základě výsledků průzkumu mohu nyní blíže specifikovat požadavky na vlastnosti vznikající aplikace a podrobněji rozvést a odůvodnit zvolené cíle práce

Tvorba mapy Z průzkumu vyplynulo, že pokud je hlavním cílem uživatele vzhled myšlenkové mapy spíše než rychlost její tvorby, pak jednou z nejlepších metod při vytváření podřízeného uzlu je tažení speciálního tlačítka z rodičovského uzlu. Musí být dále umožněno přesouvání uzlů – při přesunutí rodičovského uzlu by se relativně s ním měla přesunout také celá podřízená větev.

Nastavení vzhledu Kvůli zaměření na vizuální stránku musí aplikace nabídnout široké možnosti nastavení vzhledu všech prvků mapy. Zároveň je podstatné toto nastavení urychlit a zjednodušit.

Z průzkumu vyplynulo, že všechny zkoumané nástroje nastavují styl prvků pouze staticky – pokud formátování podle větví nebo úrovní není dostačující (to znamená, že globální styl není použitelný) a styl prvků je již jednou nastaven, není možné ho globálně změnit (například nastavit pozadí všech stejně naformátovaných žlutých uzlů na červené) a je nutné projít všechny dotčené

uzly a styl jim nějakým způsobem upravit. Celý tento proces může být při velkém počtu uzlů časově náročný – nikdy se nevyhneme nutnosti označit a projít všechny uzly, které chceme změnit.

Mnou navržená aplikace proto nabídne variantu dynamického nastavení stylu prvků, konkrétně pomocí tříd. Každá třída prvku bude mít svůj vlastní nastavený styl. Každému prvku bude možné přiřadit třídu nebo ho naformátovat samostatně. Pokud se styl třídy změní, bude zároveň aktualizován u všech prvků, kterým byla tato třída přiřazena. Aplikace umožní vytvářet nové třídy a měnit již existující třídy.

Prezentační režim Vznikající aplikace umožní zobrazit mapu v prezentačním režimu. V něm mapu nebude možné upravovat. Bude možné z něj zobrazit nastavené interaktivní prvky. Prezentace tak nebude probíhat v předem stanoveném lineárním pořadí (myšlenková mapa z principu také není lineární), ale bude do značné míry interaktivní, bude přehledná a využije přidané hodnoty myšlenkové mapy.

Iteraktivní prvky Některé informace není možné dobře prezentovat pomocí myšlenkové mapy (různé grafy, diagramy, postupy, děje…). Někdy je nutné udržet myšlenkovou mapu co nejjednodušší a nepřidávat ke každému uzlu množství detailních informací. Právě v takových chvílích je dobré tyto informace nějakým způsobem schovat a zobrazit je, až když jsou potřeba.

Současné nástroje tento problém řeší pomocí poznámek (které často nejsou formátovatelné ani viditelné z prezentačního režimu) a rozbalování podřízených větví (které ale nepomůže, pokud je nutné rozbalit pouze část potomků nebo pokud je potřeba popsat například děj knihy).

Aplikace proto umožní vytvořit myšlenkovou mapu, která bude rozšířena o interaktivní prvky v podobě stránek prezentace (tzv. slajdů). Tyto stránky bude možné přiřadit k uzlům, nastavit, naformátovat a zobrazit v prezentačním režimu přes celou obrazovku kliknutím na speciální ikonu. Tím bude zachována informační hodnota diagramu – bude viditelná hierarchie témat i jejich vzájemné souvislosti. Zároveň bude zamezeno zahlcení informacemi, protože informace, které v danou chvíli nejsou podstatné, budou skryty.

Uživatelské rozhraní Dalším z cílů práce je uživatelsky přívětivé grafické rozhraní. Ze zkoumaných nástrojů vyplynulo, že pro nastavení prvků mapy je nejintuitivnější používání lokálních nabídek, které slouží právě k nastavení konkrétního prvku a jsou zobrazeny po kliknutí na něj. Globální menu by nemělo být obsáhlé a mělo by obsahovat pouze globální funkcionalitu – nastavení tříd, spuštění prezentace.

Kapitola 3

Specifikace požadavků

V této fázi vývoje aplikace je nutné na základě cíle práce a výsledků průzkumu specifikovat, jaké požadavky musí budoucí aplikace splňovat, jaké má mít konkrétní vlastnosti a jaké mají být podporované funkcionality.

3.1 Funkční požadavky

Na aplikaci je několik obecných funkčních požadavků:

- režim tvorby mapy
- tvorba kostry mapy
- přidání interaktivních prvků
- formátování prvků mapy
- režim prezentace mapy
- export a import mapy

Každý z těchto požadavků má vlastní ID, název a popis a rozpadá se na vícero případů užití. Případy užití ilustrují konkrétní funkce aplikace, které může uživatel provádět, a jejich průběh. Mají ID, název, popis, vymezující kritéria a prioritu určenou systémem MoSCoW (must have, should have, could have, won't have).

Dále uvádím výčet požadavků a jednotlivých případů užití.

Režim tvroby mapy (F1) Aplikace zobrazí mapu v režimu tvorby. Z něj bude uživatel moci nastavit všechny prvky a vlastnosti myšlenkové mapy.

- vykreslení na kreslící ploše (F1.1) – must have
  - Mapa bude vykreslená na kreslící ploše. Tuto plochu bude možné libovolně posouvat všemi směry. Kreslící plocha bude mít stanovenou velikost, kterou bude možné změnit.
  - Minimální šířka i výška kreslící plochy bude 1 000 px. Maximální potom 10 000 px.
  - Uzly se budou moci nacházet mimo tuto kreslící plochu.
- přibližování mapy (F1.2) – should have
  - Během režimu tvorby bude možné mapu manuálně přiblížit nebo oddálit pomocí posuvníku na kreslící ploše.
- spuštění prezentačního režimu (F1.3) – must have
  - Uživatel bude mít možnost spustit prezentační režim.

Tvorba kostry mapy (F2) Aplikace umožní vytvořit kostru mapy – vytvořit uzly včetně jejich vzájemných vztahů a rozmístit je vhodným způsobem na kreslící ploše.

- tvoření uzlů (F2.1) – must have
  - Uživatel vytvoří nový uzel pomocí tažení speciálního tlačítka z vybraného uzlu. Toto tlačítko tažením umístí na libovolné místo na kreslící ploše. Na tomto místě vznikne nový podřízený uzel. Aplikace nebude podporovat žádnou variantu automatického rozložení uzlů ani tvorbu uzlů pomocí klávesových zkratek.
  - Uzly se budou moci překrývat – může dojít k vytvoření uzlu na místě již existujícího uzlu. Tlačítko musí být umístěno do okna prohlížeče, jinak bude akce neplatná.
- přesouvání uzlů (F2.2) – must have
  - Uživatel přesune existující uzel jeho přetažením na nové místo na kreslící ploše. Relativně k němu se přesune také celá podřízená větev.
  - Uzly se budou moci po přesunutí překrývat. Přetažením uzlu mimo okno prohlížeče bude akce zrušena a uzel vrácen na původní místo. Chybou není, pokud se po přesunu uzlu některá část jeho podřízené větve bude nacházet mimo okno prohlížeče.

- přidání plovoucího uzlu (F2.3) – could have
  - Přidání plovoucího uzlu bude možné pomocí tažení tlačítka z hlavního menu.
  - Plovoucí uzel musí být umístěn na viditelnou kreslící plochu, jinak bude akce neplatná.
- mazání uzlů (F2.4) – should have
  - Bude možné smazat libovolný uzel myšlenkové mapy. Společně s uzlem budou automaticky smazány všechny jeho podřízené uzly.

Přidání interaktivních prvků (F3) Aplikace nabídne jeden základní typ interaktivního prvku – vyskakovací okno v podobě stránky prezentace (slajdu). Tyto stránky budou zobrazeny přes většinu obrazovky.

- přidání stránky prezentace (F3.1) – must have
  - Bude možné přidat slajd k libovolnému uzlu. Přidaný slajd bude zařazen vždy jako poslední v posloupnosti slajdů u daného uzlu.
  - Počet slajdů přiřazených k jednomu uzlu nebude omezený.
- odebrání stránky prezentace (F3.2) – must have
  - Bude možné odebrat libovolný slajd.

Formátování prvků mapy (F4) Aplikace umožní přidat uzlům textový obsah a formátovat všechny prvky mapy (textové uzly, linie, interaktivní prvky). Pro vzhled prvků mapy bude možné vytvořit třídu stylu. Existující třídu bude možné modifikovat a přiřadit ji existujícím uzlům.

- přidání a modifikace obsahu uzlu (F4.1) – must have
  - Po aktivování módu modifikace textu v rámci uzlu bude možné libovolně měnit jeho textový obsah. Veškerý textový obsah bude na jednom řádku. Nebude možné přidávat do uzlů obrázkový obsah.
  - Délka textu bude omezena na 50 znaků, delší text nebude možné zadat. Text bude moci obsahovat všechny tisknutelné znaky. Šířka uzlu se bude při zadávání textu zvětšovat automaticky podle délky textu.
- formátování prvku (F4.2) – must have
  - Uživatel bude moci formátovat libovolný uzel (nastavení velikosti – F4.2.1, nastavení vzhledu – F4.2.2), spojovací linii (F4.2.3) a interaktivní prvek (F4.2.4).

  - Formátování bude možné dvěma základními způsoby – manuálně a pomocí přiřazení třídy. Platí, že po nastavení třídy manuálně formátovanému prvku bude manuální formát zahozen a prvek bude formátovaný podle vlastností třídy. Naopak, pokud uživatel manuálně modifikuje prvek nastavený pomocí třídy, budou k manuálnímu formátu zkopírovány vlastnosti třídy a poté bude provedena úprava – uzel tak bude formátovaný shodně jako třída (kromě manuálně provedené změny), ale už nebude nadále součástí této třídy. Dále, pokud uživatel odebere prvku nastavenou třídu, bude pro formátování prvku použit výchozí formát stanovený aplikací.

- změna velikosti uzlu (F4.2.1) – should have
  - Uživatel bude moci změnit velikost existujícího uzlu zadáním nové velikosti do formuláře.
  - Nebude kontrolováno, zda je po změně velikosti viditelný veškerý text v uzlu. Minimální šířka i výška uzlu bude 50 px, maximální bude 1 000 px.
- formátování uzlu (F4.2.2) – must have
  - Aplikace bude podporovat nastavení následujících vlastností uzlů:
    * tvar uzlu (obdélník, obdélník s kulatými rohy, kruh, elipsa atd.)
    * barva pozadí
    * okraj – barva, tloušťka, styl (plný, tečkovaný, dvojitý…)
    * text – font, barva, velikost, zarovnání (doprava, doleva, na střed), dekorace (tučné, kurzíva, podtržené)
- možnosti formátování linií (F4.2.3) – should have
  - Liniím bude moct uživatel nastavit barvu, šířku, styl (plná, čárkovaná, tečkovaná…), obecný tvar (křivka, přímka, lomená čára). Nebude možné měnit konkrétní tvar spojovací linie.
- možnosti formátování interaktivních prvků (F4.2.4) – must have
  - Pro každý interaktivní prvek bude moci uživatel vybrat z nabídky základní rozložení, které bude obsahovat různá textová či obrázková pole. Bude možné modifikovat obsah textových polí a nahrávat obrázky do obrázkových polí. Z hlediska formátování bude možné modifikovat poměr stran prvku, zaoblení rohů, barvu pozadí, okraj (barvu, šířku a styl) a obecné textové vlastnosti (font, barvu písma, velikost písma…). Další textové vlastnosti budou předdefinovanou součástí vybraného rozložení.
  - Velikost obrázku bude omezena na 1 MiB.

- vytvoření třídy stylu (F4.3) – must have
  - V nabídce tříd bude uživatel moci definovat vlastní třídu stylu pro libovolný formátovatelný prvek (uzel, linii, interaktivní prvek). Nastavení třídy proběhne pomocí vyplnění formuláře, který bude obsahovat nastavitelné vlastnosti. Po uložení bude třída zobrazena v nabídce tříd a bude moci být použita.
  - Nastavit bude možné všechny vlastnosti prvků zmíněné v F4.2.2, F4.2.3 a F4.2.4 s výjimkou rozložení interaktivních prvků – to znamená, že dva slajdy se stejným stylem budou moci mít různá rozložení.
- modifikace třídy stylu (F4.4) – must have
  - Již definovanou třídu bude možné dále upravovat pomocí změn v příslušném formuláři. Po uložení změn budou změny stylu propagovány ke všem prvkům, kterým byla daná třída přiřazena.
- přiřazení třídy stylu (F4.5) – must have
  - Uživatel bude moci hotovou třídu přiřadit libovolnému prvku daného druhu. Přiřadit třídu bude možné z lokálního menu u prvku.
- výchozí styly (F4.6) – could have
  - V nabídce tříd budou označeny a definovány výchozí styly pro všechny prvky (kořenový uzel, ostatní uzly, spojovací linie, interaktivní prvky). Tyto třídy budou automaticky přiřazeny nově vytvořeným prvkům. Výchozí styly bude možné upravit, ale nebude je možné smazat.
- smazání třídy (F4.7) – should have
  - Jakoukoli třídu stylu s výjimkou výchozí bude možné smazat. Uzly formátované touto třídou poté ztratí formát a budou formátovány výchozím formátem aplikace.

Prezentační režim mapy (F5) Mapu bude možné zobrazit ve specializovaném prezentačním režimu. V rámci tohoto režimu nebude možná jakákoli editace mapy. Mapa bude zobrazena přes celé okno prohlížeče. Uživatel bude moci zobrazovat interaktivní prvky a mapu přibližovat.

- zobrazení interaktivních prvků (F5.1) – must have
  - Uživatel bude moci zobrazit interaktivní prvky pomocí kliknutí na odpovídající ikonu u uzlu. Bude možné procházet sekvenci interaktivních prvků přiřazených k tomuto uzlu.

- přibližování mapy (F5.2) – should have
  - V rámci prezentačního režimu bude přibližování možné ve dvou modifikacích. Základní variantou bude manuální přibližování pomocí posuvného ukazatele. Rozšířením bude automatické přiblížení celé podřízené větve vybraného počátečního uzlu. Bude možné také automatické oddálení (o jednu úroveň nebo úplné).
- ukončení prezentačního režimu (F5.3) – must have
  - Bude možné ukončit prezentační režim a vrátit se zpět do režimu tvorby.

Export a import mapy (F6) Aplikace umožní exportovat hotovou mapu do souboru ve formátu JSON. Stejně tak ji bude možné ze stejného formátu také importovat.

- export do formátu JSON (F6.1) – must have
  - Uživatel bude moci rozpracovanou mapu uložit do souboru ve formátu JSON. Bude možné exportovat celou mapu včetně tříd stylů a nastavení kreslící plochy, nebo pouze samotné třídy (například pro použití pro další mapy).
- import z formátu JSON (F6.2) – must have
  - Uživatel bude moci do aplikace nahrát celý textový soubor ve formátu JSON nebo obsah souboru vložit do textového pole. Textový soubor může obsahovat buď mapu včetně tříd stylů a nastavení kreslící plochy, nebo pouze třídy stylů.
  - Nahraný soubor bude validován na přítomnost a správný datový typ všech potřebných atributů. Nebude validováno, zda atributy spadají do správné množiny hodnot či do správného rozsahu.

3.2 Nefunkční požadavky

Nefunkční požadavky popisují základní vlastnosti aplikace. Eviduje se u nich ID, popis a vymezující kritéria.

Některé z nefunkčních požadavků omezují zaměření aplikace (pro určité prohlížeče, určitá rozlišení, na určité funkce). V bakalářské práci se totiž dle cílů zaměřuji na vývoj nové funkcionality. Proto jsem se rozhodla věnovat se spíše právě novým funkcím aplikace, než například široké podpoře prohlížečů.

Dále uvádím výčet nefunkčních požadavků.

Uživatelské rozhraní (N1) Aplikace musí být maximálně uživatelsky přívětivá. Funkce musí být intuitivní a snadno dosažitelné. Aplikace bude využívat přístup s malým globálním menu s obecnou funkcionalitou (spuštění prezentace, export) a s malými lokálními menu s konkrétní funkcionalitou u každého prvku. Uživatelské rozhraní bude co nejjednodušší, nesmí dojít k zahlcení uživatele různými možnostmi a tlačítky.

Podporované prohlížeče (N2) Aplikace bude podporovat nejnovější verze prohlížečů Mozilla Firefox a Google Chrome. Nebude upravená pro podporu prohlížečů Internet Explorer, Microsoft Edge a Safari.

Responsibilita (N3) Aplikace bude uzpůsobena pro šířku obrazovky větší než 1 280 px. Nebude dimenzovaná na menší obrazovky ani na dotyková zařízení.

Persistence (N4) Aplikace v současné verzi nebude podporovat jakýkoli typ uživatelského účtu, nebude vyžadovat přihlášení a sama nebude ukládat mapy uživatelů. Persistence bude zajištěna skrz export do formátu JSON. Aplikace bude do budoucna počítat s možností rozšíření o podporu uživatelských účtů a ukládání map.

Pokrytí testy (N5) Logika a funkcionalita aplikace (obecná práce se strukturou myšlenkové mapy apod.) bude pokryta automatizovanými testy. Automaticky testován nebude vzhled. Uživatelské rozhraní bude testováno manuálně.

Kapitola 4

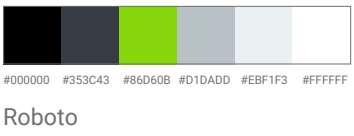
Návrh

Na základě požadavků specifikovaných v předchozí kapitole nyní přistoupím k návrhu aplikace. Konkrétně navrhnu uživatelské rozhraní, zvolím vhodné technologie pro implementaci a navrhnu základní architekturu aplikace.

4.1 Návrh uživatelského rozhraní

Základní vlastnosti rozhraní jsou definované v nefunkčním požadavku N1 – rozhraní musí být uživatelsky přívětivé, jednoduché a intuitivní. Globální menu bude z tohoto důvodu obsahovat pouze základní funkcionalitu, a veškeré vlastnosti prvků budou nastavovány z malých lokálních nabídek.

Barevná paleta a fonty Nejprve je nutné zvolit základní barevnou paletu pro prvky aplikace a také font (viz obrázek 4.1). Jako základní barvy jsem zvolila šedou a zelenou. Šedá bude použita pro pozadí, písmo a obecné prvky a zelená bude sloužit pro zdůraznění a akcent. Základním fontem aplikace bude *Roboto*, který je dle *Google Fonts* [24] open source a proto může být použit v rámci vznikající aplikace.



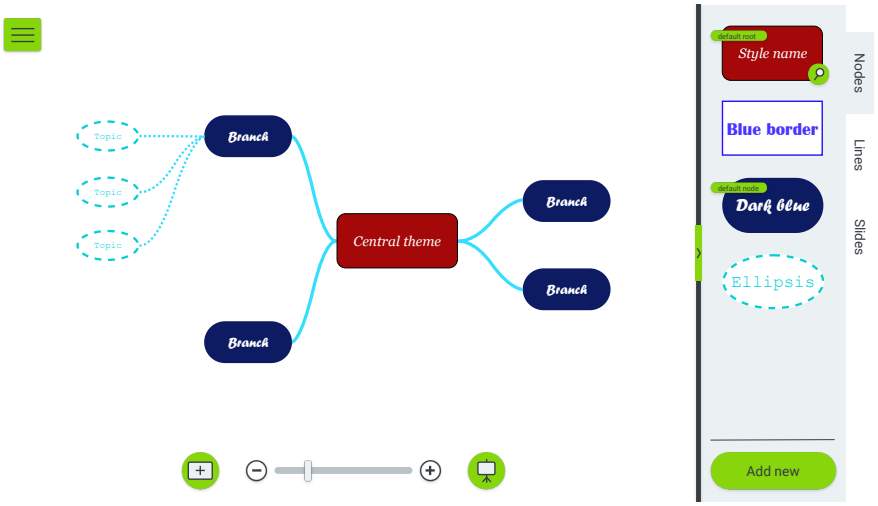
Obrázek 4.1: Barevná paleta aplikace a ukázka fontu

Základní rozložení První pohled na prostředí aplikace nesmí být pro uživatele matoucí. Z tohoto důvodu jsem se rozhodla pro co nejjednodušší návrh

s minimem viditelných tlačítek a možností (viz obrázek 4.2). Zřídka používané funkce jsou umístěny v rozbalovacích nabídkách. Tlačítka jsou často pro větší přehlednost označena snadno pochopitelnou ikonkou. Po najetí myší se ale zobrazí také textový popis funkcionality.

Většinu plochy obrazovky zabírá čistě bílá kreslící plocha. Ta musí být co největší, aby měl uživatel maximální přehled o mapě, kterou vytváří. V levé horní části je tlačítko rozbalovacího menu, ze kterého budou spustitelné základní funkce jako otevření čisté mapy, export, import a nápověda. V dolní části uprostřed je základní menu pro práci s mapou, které obsahuje častěji potřebné funkce – přidání nového samostatného uzlu, posuvník pro přiblížení a oddálení a spuštění prezentace.

V pravé části je panel tříd, ve kterém bude možné přidávat a upravovat třídy. Panel bude možné schovat a tím kreslící plochu ještě zvětšit.

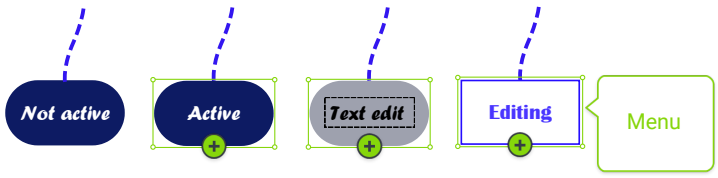


Obrázek 4.2: Základní rozložení aplikace

Editace uzlů a linií Uzly i linie mají více stavů, ve kterých se mohou nacházet. Tyto stavy jsou přepínány různými druhy kliknutí na daný prvek.

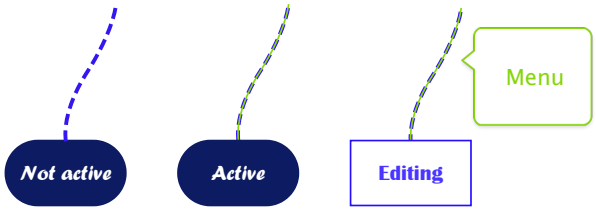
- různé stavy uzlu (viz obrázek 4.3):
  - neaktivní stav – výchozí stav uzlu, žádné funkce nejsou dostupné, do tohoto stavu se uzel vrací při kliknutí na jiné místo aplikace, neaktivní uzel je možné uchopit a přesunout
  - stav po kliknutí levým tlačítkem – tj. aktivní stav, zelený rámeček pro odlišení, tlačítko pro přidání podřízeného uzlu
  - stav po dvojkliku levým tlačítkem – tj. úprava obsahu, aktivuje se textové pole pro zadání či modifikaci textu uzlu, dále vše jako u aktivního stavu

  - stav po kliknutí pravým tlačítkem – tj. formátování, objeví se lokální menu pro úpravu vzhledu uzlu, dále vše jako u aktivního stavu
  - stav při tažení uzlu – stejný jako aktivní stav



Obrázek 4.3: Různé stavy uzlů

- různé stavy linií (viz obrázek 4.4):
  - neaktivní stav – výchozí stav linie, nejsou dostupné žádné funkce, do tohoto stavu se linie vrací při kliknutí na jiné místo aplikace
  - stav po kliknutí levým tlačítkem – tj. aktivní stav, zelená linie ve středu pro označení
  - stav po kliknutí pravým tlačítkem – tj. formátování, objeví se lokální menu pro editaci vzhledu linie, dále vše jako u aktivního stavu

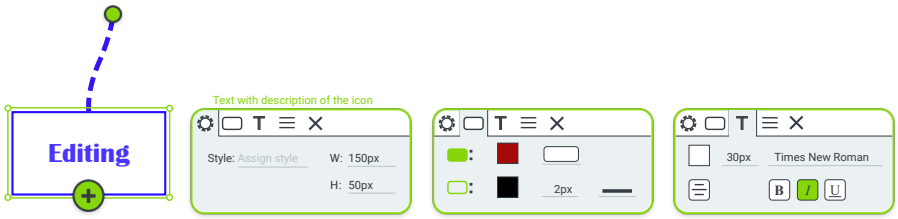


Obrázek 4.4: Různé stavy linií

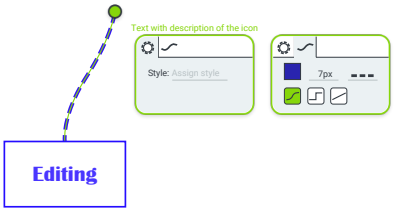
Editační menu prvků také musí být co nejpřehlednější. Proto jsem přistoupila k rozdělení funkcí po skupinách do záložek – nabídky pak nejsou na první pohled tolik obsáhlé a také jsou menší.

Editační menu uzlu (viz obrázek 4.5) je rozděleno do tří záložek – hlavní nastavení, editace celkového vzhledu a editace textu. Menu dále obsahuje tlačítka pro spuštění editace slajdů a pro smazání uzlu včetně podřízené větve.

Editační menu linie (viz obrázek 4.6) je rozděleno do dvou záložek – hlavní nastavení a nastavení vzhledu linie.



Obrázek 4.5: Editační menu uzlu



Obrázek 4.6: Editační menu linie

Interaktivní prvky Slajdy budou zobrazitelné pomocí ikonky v editačním menu uzlu. Zobrazen bude vždy první slajd, procházení mezi slajdy bude možné pomocí šipek. Za posledním slajdem bude následovat speciální strana s tlačítkem pro přidání nového slajdu na konec sekvence.

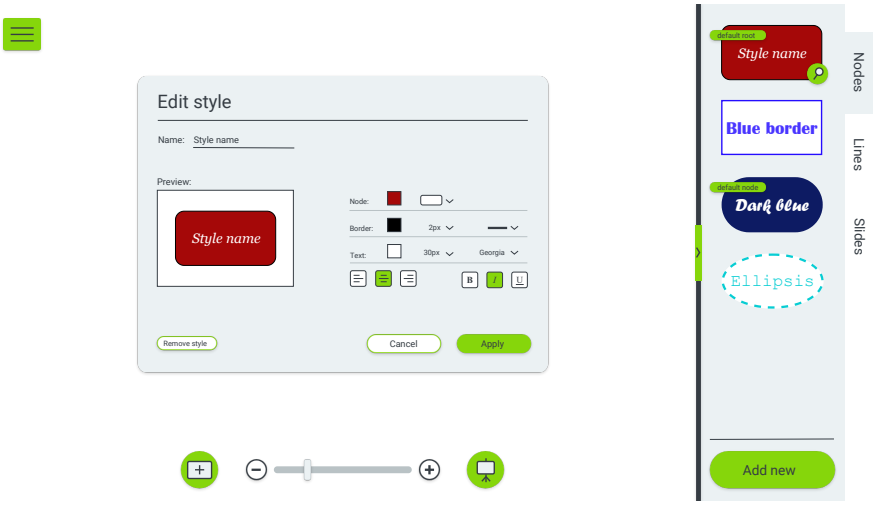
Po kliknutí pravým tlačítkem myši na slajd se zobrazí editační menu (viz obrázek 4.7). To je rozdělené do tří záložek – hlavní nastavení, editace celkového vzhledu a editace textu. Dále obsahuje také tlačítko pro odstranění prvku. Zobrazení slajdů bude ukončeno kliknutím na křížek v pravém horním rohu slajdu.

Správa tříd Nabídka tříd je rozdělena na tři záložky podle druhů formátovatelných prvků. Pomocí vlaječky jsou v ní označeny výchozí styly. Při najetí myší nad styl se objeví ikona lupy, pomocí které bude moci uživatel zobrazit editační formulář daného stylu.

Formulář pro editaci stylu bude obsahovat všechny možnosti nastavení vzhledu a náhled změn. Stejný formulář bude použit při definování nového stylu. Formulář uzlu je zobrazen na obrázku 4.8, formuláře linie a interaktivního prvku na obrázku 4.9.

Prezentační režim V prezentačním režimu bude přes celé okno prohlížeče zobrazena kreslící plocha. V dolní části bude viditelný posuvník pro přiblížení

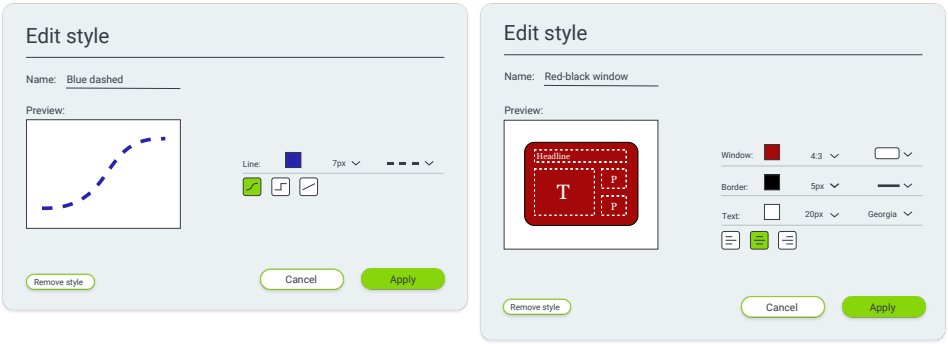
Obrázek 4.7: Nastavení interaktivního prvku



Obrázek 4.8: Nastavení stylu uzlu

a oddálení mapy a dvě tlačítka pro automatizované oddálení. V levém horním rohu bude tlačítko pro zavření prezentace. Zavření prezentace bude možné i klávesou `Escape`.

V prezentačním režimu nebude možná editace. Uzly i linie budou mít pouze jeden stav, shodný s neaktivním stavem v režimu tvorby. Při najetí myší nad uzel se objeví ikony pro otevření interaktivních oken a pro automatické přiblížení. Prezentace interaktivních oken bude otevřena po kliknutí na ikonu u uzlu a zavřena po kliknutí mimo prostor okna.



Obrázek 4.9: Nastavení stylu linie a interaktivního prvku

4.2 Výběr základních technologií

Důležitou volbou pro další vývoj aplikace je volba programovacího jazyka a eventuálně také příslušných frameworků.

Požadavky na aplikaci nevyžadují žádnou komunikaci s databází nebo s backendem, proto se bude jednat o čistě frontendovou aplikaci, jejíž veškerá funkcionalita bude prováděna přímo v prohlížeči uživatele.

React Pro vývoj aplikace jsem vybrala základní přístup s HTML, CSS a JavaScriptem, konkrétně jeho knihovnou React. Aplikace bude vysoce interaktivní, bude obsahovat velké množství opakujících se prvků (uzly, spojovací linie, nabídky), které se mohou nacházet v různých stavech, a bude muset pracovat s měnícími se vstupními daty. React se dle svých webových stránek [25] specializuje na tvorbu uživatelských rozhraní a usnadňuje zejména tvorbu interaktivních aplikací, dokáže efektivně vykreslovat různé prvky v závislosti na aktuálním stavu a na vstupních datech a umožňuje tvořit znovupoužitelné komponenty, které spravují svůj vlastní stav. Je to moderní a v současnosti často používaný framework s poměrně rozsáhlou základnou vývojářů, výbornou dokumentací a velkým množstvím již hotových komponent

Sass Styl hlavních komponent bude určen pomocí kaskádových stylů. Pro jejich zpracování budu využívat tzv. CSS preprocesor, konkrétně Sass. Dle webových stránek preprocesoru [26] umožňuje Sass mimo jiné například používání proměnných, zanořování, mixinů, dědění a podobně. Pro vývoj rozsáhlejší aplikace jsou tyto funkce velmi výhodné – proměnné mohou držet hlavní styl stránky (barevnou paletu, fonty, velikosti písma), který se pak snadno mění a aktualizuje na všech místech naráz, mixiny mohou pomoci při zajištění kompatibility mezi prohlížeči, zanořování, importy a dědičnost zase velmi zpřehledňují kód a umožňují lépe dodržovat DRY princip.

Jest a Enzyme Pro automatizované testování budu používat framework Jest a utilitu Enzyme. Jest je dle svých webových stránek [27] plně kompatibilní s Reactem, má jednoduché a snadno použitelné API, umožňuje snadné mockování a generuje přehledné statistiky pokrytí kódu testy. Je proto vhodný jako základ všech testovacích jednotek. Jako doplnění pro usnadnění vykreslování a procházení jednotlivých komponent v rámci testů a také pro simulaci událostí pak použiji Enzyme, který potřebné funkcionality podporuje [28].

4.3 Analýza problémové domény

V rámci analýzy problémové domény je nutné určit a pojmenovat jednotlivé entity, jejich atributy, vlastnosti a vztahy.

Zkoumanými problémovými entitami jsou zejména uzel (`Node`), linie (`Line`) a prezentační slajd (`Slide`). Celá problémová doména je shrnuta v doménovém modelu na obrázku 4.10.

Základním prvkem myšlenkové mapy je uzel, jehož vlastnostmi jsou `text` (popis myšlenky), `offset` (vyjádření pozice vůči rodičovskému uzlu resp. vůči kreslícímu plátnu) a `size` (velikost – výška a šířka v pixelech).

Každý uzel může mít libovolný počet podřízených uzlů – potomků. Uzel, který není samostatný ani kořenový, má také příchozí linii – vztah s rodičovským uzlem. Každý uzel pak může mít libovolný počet prezentačních slajdů. Základními vlastnostmi slajdu jsou jeho rozložení (`layout`) a jeho obsah (`content`).

Každý z těchto elementů má svůj určený styl se specifickou strukturou, která je dána možnostmi nastavení objektů. Styl může být přiřazený explicitně nebo skrze třídu stylu. Třída muže být sdílena více prvky daného typu. V rámci stávající implementace nebude možné využívat explicitní styl i třídu stylu současně (v takovém případě bude mít vždy přednost explicitní styl).

Každá třída má své jméno, které slouží zejména jako orientační prvek pro uživatele. Třída poskytuje elementům specifický objekt stylu.

V budoucí aplikaci bude celá tato struktura reprezentována pomocí jednoduchých JavaScriptových objektů. V každém okamžiku musí být možno strukturu co nejrychleji serializovat do souboru typu JSON (z důvodu budoucího předpokládaného napojení na server). Proto nebude struktura obsahovat žádné metody nebo funkce.

4.4 Návrh základní architektury aplikace

V rámci této sekce popíši strukturu a vzájemnou komunikaci hlavních komponent aplikace.

React tvoří aplikace pomocí skládání samostatných komponent, z nichž každá má svůj vlastní stav (`state`) a read-only sadu atributů od rodičovské

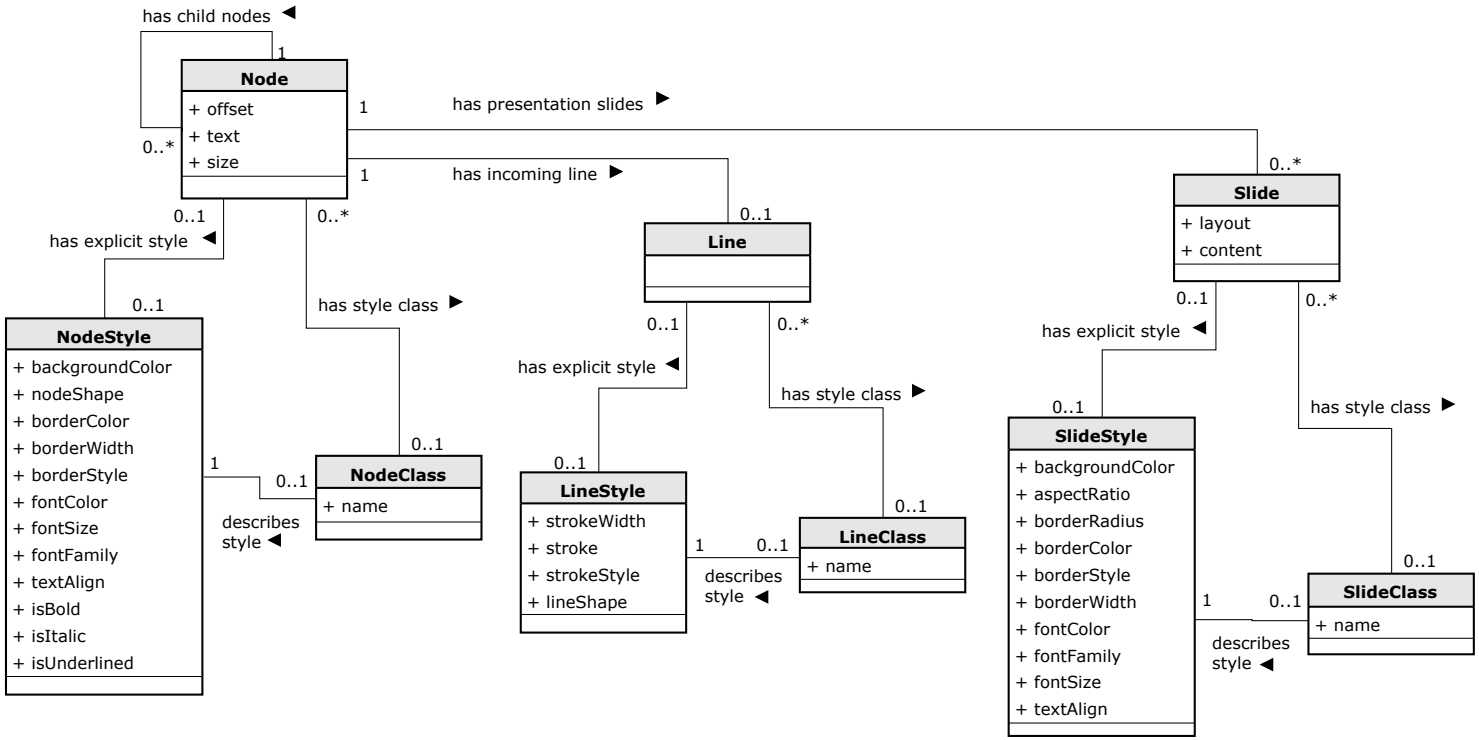
komponenty (`props`). Z hlediska návrhu je nejdůležitější zmapovat právě tyto vlastnosti, protože zabezpečují komunikaci mezi komponentami.

Návrh komponent je shrnutý v diagramu na obrázku 4.11. Pro popis využívám upravenou variantu UML notace třídy, ve které místo atributů a metod zachycuji podstatnější `state` a `props`. Návrh se skládá se z následujících komponent:

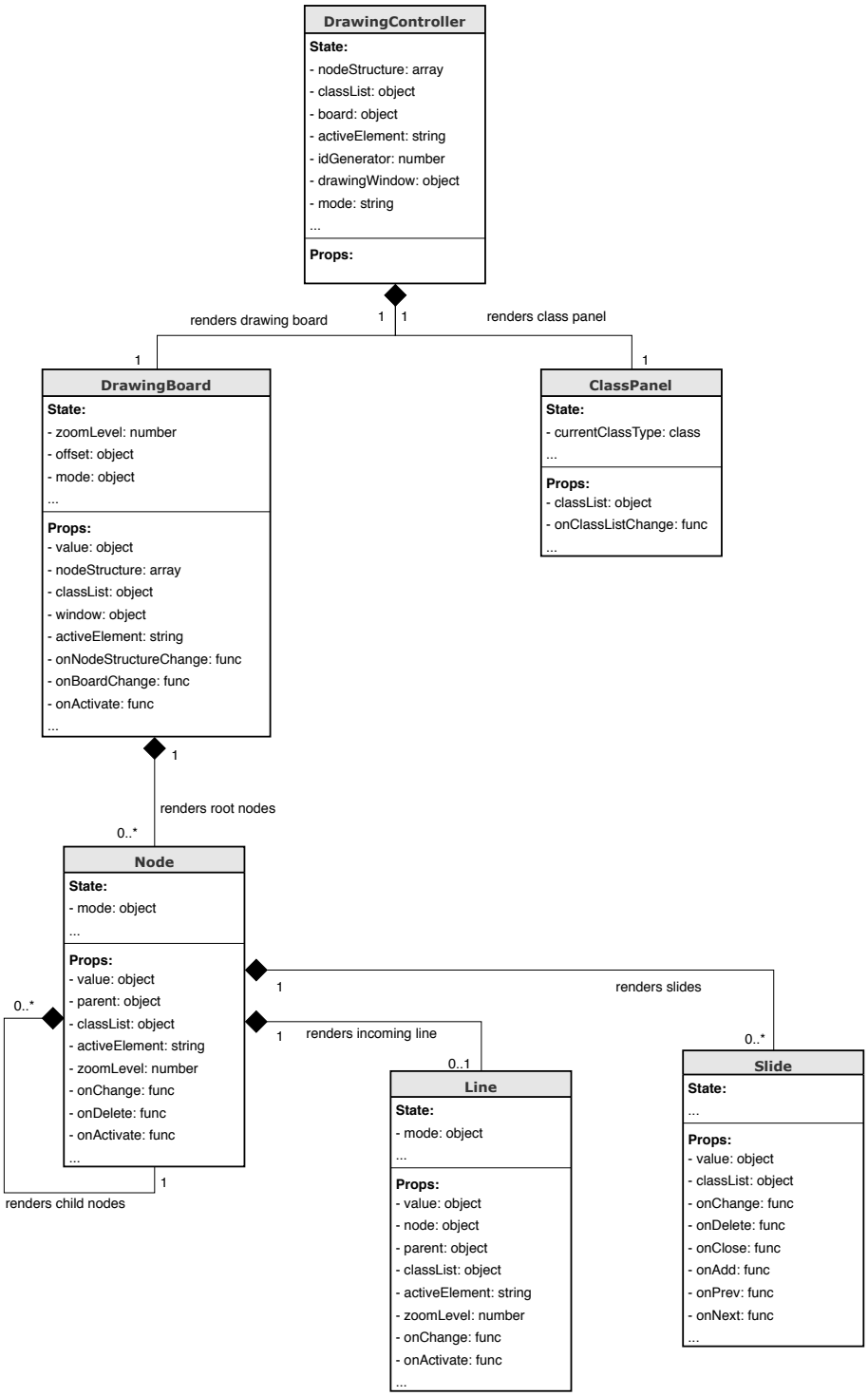
- `DrawingController` je hlavní řídící komponentou, která ve svém stavu drží informace o datové struktuře uzlů i o datové struktuře tříd.
- `ClassPanel` řídí vykreslení panelu pro správu formátování tříd a spravuje změny jednotlivých tříd.
- `DrawingBoard` reprezentuje kreslící plátno, na které se vykreslují jednotlivé uzly. Sama spravuje svoji vlastní pozici a přiblížení.
- `Node` neboli uzel se může nacházet v různých stavech. Vykresluje své podřízené uzly, svou příchozí linii a své slajdy.
- `Line` neboli linie má také svůj stav a vykresluje se na základě informací o vzájemné poloze uzlu a jeho rodiče.
- `Slide` zastupuje vyskakovací okno v podobě stránky prezentace.

Důležitým rozhodnutím v rámci návrhu bylo, zda uzly serializovat v rámci komponenty `DrawingBoard` a vykreslit všechny jako plochý seznam, či vykreslit pouze kořenové uzly jednotlivých stromů a nechat uzly, aby rekurzivně vykreslily své potomky. Z návrhu je patrné, že jsem zvolila druhou variantu, a to zejména z následujících důvodů:

- Umožňuje využít přirozeného rekurzivního vykreslování komponent Reactu. Není nutná umělá serializace a o rekurzivní průchod všemi uzly se postará sám framework, respektive rodičovské uzly.
- Umožňuje využít pro určení pozice uzlu absolutní pozici vůči rodičovskému uzlu (vlastnost CSS `position: absolute`), kterou je možné nastavit přímo z komponenty na požadovanou hodnotu. Uzel si proto musí držet pouze své odsazení vůči rodičovskému uzlu, nikoli pozici vůči kreslícímu plátnu.
- Přímo podporuje požadavek na přesouvání podřízených uzlů zároveň s rodičovským. Není nutné přepočítávat žádná odsazení, kromě odsazení právě posouvaného uzlu.
- Vznikající stromová struktura HTML elementů je přirozenější a více vypovídající, než prostý seznam.



Obrázek 4.10: Doménový model



Obrázek 4.11: Model komponent

Kapitola 5

Realizace

V rámci této kapitoly popíši detailněji návrh struktury aplikace a budu se věnovat účelu a funkcionalitám hlavních komponent. Zaměřím se také na některé zajímavé body implementace.

5.1 Základy vývoje v Reactu

Při tvorbě webové aplikace v Reactu je důležité znát principy vývoje v tomto frameworku a tzv. „best practises“ – doporučované postupy pro řešení nejrůznějších problémů. V této sekci tedy představím nejdůležitější z nich.

5.1.1 JSX

React dle své dokumentace [29] umožňuje pro popis vzhledu uživatelského rozhraní využívat speciální rozšíření syntaxe JavaScriptu zvané JSX. Dle [30] má JSX syntaxi podobnou HTML, nicméně se nejedná o kód, který by dokázaly zpracovat přímo prohlížeče – musí být nejprve kompilován. V dokumentaci Reactu [29] jsou dále uvedeny následující informace o syntaxi JSX (ilustrace použití viz výpis kódu 5.1):

- JSX element je zpracováván jako JavaScriptový výraz,
- pomocí složených závorek lze do JSX elementu vložit libovolný validní JavaScriptový výraz,
- je možné specifikovat atributy JSX elementů (např. `src` pro obrázek nebo `className` pro specifikaci CSS třídy),
- je možné tvořit vnořené JSX elementy.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

const element = <img src={user.avatarUrl} />;

const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

Výpis kódu 5.1: Ukázka použití JSX syntaxe [29, sloučeno více ukázek]

5.1.2 Komponenty

Dle dokumentace Reactu [29] jsou komponenty prostředkem pro rozdělení uživatelského rozhraní na vícero samostatných jednotek. Existují dva druhy komponent – funkční, definované jako JavaScriptové funkce, a třídní, definované jako JavaScriptové třídy. Komponenta většinou vykresluje React element (vytvořený pomocí JSX). V rámci funkční komponenty je tento obsah získán jako návratová hodnota celé funkce, v případě třídní komponenty jako návratová hodnota metody `render`. Definice obou typů komponent je ilustrována na výpisu kódu 5.2.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Výpis kódu 5.2: Ukázka rozdílu mezi třídní a funkční komponentou [29, sloučeno více ukázek]

[31] uvádí, že komunikace komponent mezi sebou je zajištěna pomocí tzv. `props`, tedy atributů předávaných například pomocí syntaxe JSX. V rámci `props` mohou být předávány také funkce. Žádná komponenta nesmí přímo editovat své `props`. Místo toho si komponenty mohou předat tzv. „callback“ – tedy funkci, která může být zavolána a která nese informaci o požadované změně. Tato vlastnost je nazývaná „jednosměrný datový tok“.

Podle [32] může mít třídní komponenta svůj vlastní stav, neboli `state`. Jedná se o objekt, který je atributem komponenty a je v ní přístupný skrze volání `this.state`. Jeho úpravy jsou prováděny pomocí asynchronní metody `this.setState()`.

Dle [33] je vhodné používat funkční komponenty, které nemají vlastní stav – jsou jednodušší a používají méně paměti.

5.1.3 Skládání a dědičnost

Dle dokumentace [29] upřednostňuje React pro sdílení kódu mezi komponentami použití skládání (composition) namísto dědičnosti. Dokumentace dále uvádí následující možnosti, jak skládání v Reactu správně využít.

- Jednoduchého skládání je možné docílit pomocí předání vnořených elementů, které jsou v komponentě dostupné skrz `props.children` (viz výpis kódu 5.3). Elementy je rovněž možné předávat v rámci `props`.
- Specializace může být dosažena pomocí generických komponent použitých v rámci těch specializovaných – generické komponenty mohou být nastaveny například pomocí `props`.
- Pro sdílení funkcionality, která není spojena s vykreslováním, je doporučeno extrahovat ji do separátního JavaScriptového modulu a importovat ji v těch komponentách, kde je potřeba.

```
function FancyBorder(props) {
  return (
    <div className={'FancyBorder FancyBorder-' + props.color}>
      {props.children}
    </div>
)}

function WelcomeDialog() {
  return (
    <FancyBorder color="blue">
      <h1 className="Dialog-title">Welcome</h1>
    </FancyBorder>
)}
```

Výpis kódu 5.3: Ukázka vykreslení vnořených elementů [29, sloučeno více ukázek, kód zkrácen]

5.2 Technologické řešení vykreslení myšlenkové mapy

Na úvod implementace bylo potřeba se rozhodnout, jakým způsobem bude vyřešeno vykreslení myšlenkové mapy. Vybírala jsem ze tří možností – vykreslit celou mapu na jednom elementu `canvas`, v rámci jednoho elementu `svg` nebo využít kombinace běžných elementů (`div`, `p`, `span`) pro uzly a jednoho ze speciálních elementů (`svg`, `canvas`) pro linie.

Jediný `canvas` je pro takto navrženou aplikaci s těmito technologiemi poměrně nevhodný. Dle [34] probíhá vykreslení objektů na elementu `canvas` skrze JavaScriptové metody a zpracování událostí je (v porovnání s ostatními variantami) komplikovanější. Celkově by tato varianta neumožňovala plně využít potenciál Reactu a JSX syntaxe.

Jediný element `svg` je vhodnější řešení – umožňuje využít JSX syntaxi [35] a umožňuje jednoduché zpracování událostí [36]. V rámci návrhu aplikace jsem se však rozhodla, že každá komponenta bude mimo jiné spravovat i své vlastní editační menu s případnými formulářovými prvky. Ačkoli je dle [36] možné do `svg` zakomponovat také běžné HTML elementy, nejedná se dle mého názoru o úplně přímočarý a přehledný přístup a jednodušší bude třetí varianta – kombinace běžných a speciálních elementů.

Kombinace běžných a speciálních elementů umožňuje vykreslit jak uzly (pro něž jsou běžné elementy zcela dostačující), tak také linie. Zároveň je možné tímto způsobem jednoduše a přehledně vykreslovat přímo z komponent editační menu s formulářovými prvky. Zpracování událostí je přímočaré a je možné využít syntaxi JSX.

5.3 Hlavní komponenty aplikace

V této sekci popíši hlavní komponenty aplikace. Jejich struktura odpovídá návrhu struktury komponent na obrázku 4.11.

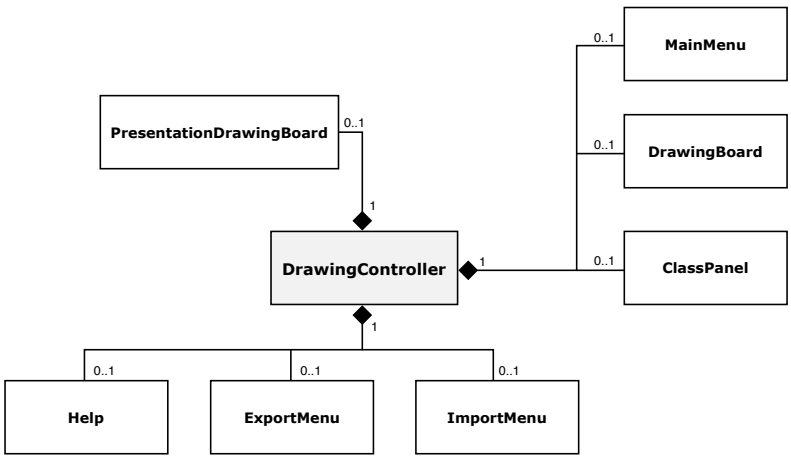
5.3.1 DrawingController

`DrawingController` je centrální komponentou aplikace. Ve svém `state` drží mimo jiné také všechny informace o aktuálním stavu aplikace – strukturu uzlů myšlenkové mapy (`nodeStructure`), seznam existujících tříd (`classList`), informace o kreslícím plátně (`board`).

Hlavní zodpovědností této komponenty je správa stavu aplikace, potažmo správa dat – pouze tato komponenta může upravit strukturu uzlů, seznam tříd nebo velikost plátna. Svým vnořeným komponentám proto poskytuje odpovídající callbacky, skrz které mohou o tuto úpravu zažádat.

Struktura komponenty `DrawingController` jako centrální komponenta aplikace zajišťuje vykreslení všech jejích součástí. Na základě stavu aplikace vykresluje následující prvky (celá struktura viz diagram na obrázku 5.1):

- `DrawingBoard` – kreslící plátno, více v sekci 5.3.2,
- `ClassPanel` – panel pro správu tříd, více v sekci 5.3.3,
- `MainMenu` – hlavní menu aplikace umožňující vytvořit novou mapu, importovat a exportovat mapu a zobrazit nápovědu,
- `ExportMenu` – vyskakovací okno umožňující uživateli stáhnout mapu nebo třídy,
- `ImportMenu` – vyskakovací okno umožňující uživateli nahrát do aplikace uloženou mapu nebo třídy,
- `Help` – jednoduché vyskakovací okno s nápovědou pro uživatele,
- `PresentationDrawingBoard` – obdoba běžného kreslícího plátna používaná v rámci prezentačního režimu.



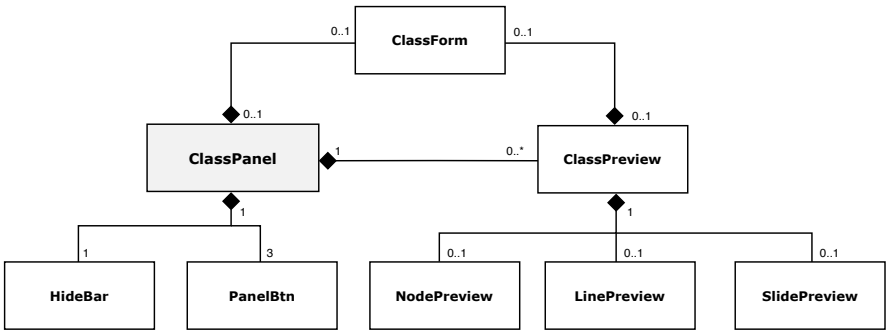
Obrázek 5.1: Model komponenty `DrawingController`

5.3.2 ClassPanel

Komponenta `ClassPanel` reprezentuje postranní panel, který slouží pro náhled, editaci a přidávání tříd stylu. Komponenta zobrazuje náhledy tříd stylů a zpracovává různé druhy uživatelských akcí (změnu třídy, přidání třídy a vymazání třídy).

Struktura komponenty `ClassPanel` vykresluje následující komponenty (viz diagram na obrázku 5.2):

- rozšířený okraj `HideBar` pro schování celého panelu,
- tlačítka `PanelBtn` k přepínání záložek,
- seznam elementů `ClassPreview` pro vykreslení jednotlivých náhledů tříd (`NodePreview`, `LinePreview` nebo `SlidePreview`) a zpracování změny třídy pomocí formuláře (`ClassForm`)
- formulář `ClassForm` pro přidání nové třídy.



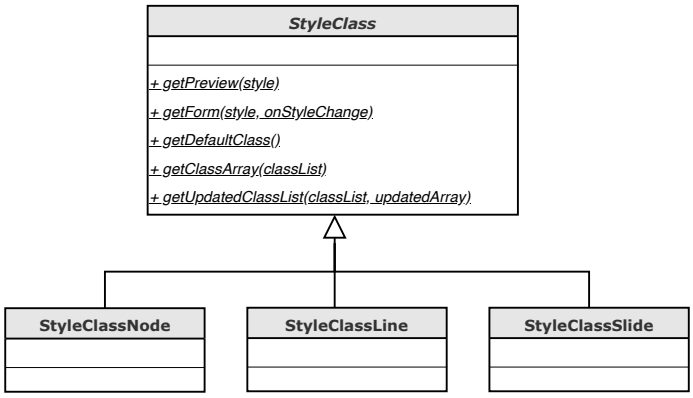
Obrázek 5.2: Model komponenty `ClassPanel`

Využití návrhového vzoru *strategy* Dle [37] umožňuje návrhový vzor *strategy* měnit chování algoritmu za běhu programu. V rámci návrhového vzoru je vytvořen kontext a jednotlivé strategie s odlišným chováním. Kontext pak na základě aktuální strategie mění své chování – strategie určuje spouštěný algoritmus.

Chování komponenty `ClassPanel` je specifické podle právě otevřené záložky (`Nodes`, `Lines` nebo `Slides`). Konkrétně se liší v tom, jaké typy náhledů se budou zobrazovat, co bude obsahem formulářů a jaké části seznamu tříd se bude týkat požadovaná úprava.

Tohoto rozdílného chování je dosaženo pomocí implementace tří různých strategií `StyleClassNode`, `StyleClassLine` a `StyleClassSlide`. Strategie jsou reprezentovány třídami se statickými metodami, které dokáží splnit požadované úlohy. Aktuální strategie je měněna v závislosti na měnící se aktivní záložce. Struktura strategií a jejich interface je patrný z diagramu na obrázku 5.3.

Použití návrhového vzoru umožňuje jednoduché přidávání nových formátovatelných prvků – je potřeba pouze vytvořit novou strategii, definovat její metody a vytvořit příslušné tlačítko a položku v seznamu tříd. Pokud by



Obrázek 5.3: Model strategií pro komponentu `ClassPanel`

strategie nebyla použita, bylo by nutné na všech místech, jejichž chování se v závislosti na strategii liší, přidat toto chování do struktury `if-else` případně do `switche`.

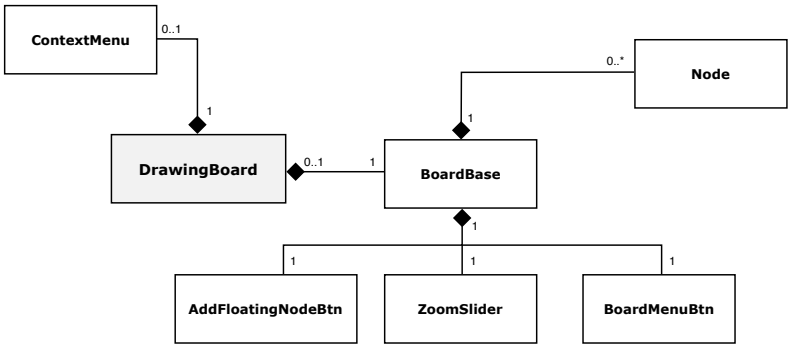
5.3.3 DrawingBoard

Komponenta `DrawingBoard` zastupuje kreslící plochu, na kterou se vykreslují všechny uzly. Sama komponenta si ve `state` drží a spravuje informace o úrovni přiblížení a o své vlastní pozici – obvykle je totiž jiné velikosti, než je okno prohlížeče a je tudíž nutné znát její absolutní polohu vůči levému hornímu rohu okna.

Kromě toho je hlavním účelem komponenty vykreslení kořenového a ostatních samostatných uzlů (které dále samy vykreslí své potomky) a zpracování jejich změn.

Struktura komponenty Struktura komponenty je patrná z diagramu na obrázku 5.4. Nejdůležitější komponenty z ní jsou:

- základová komponenta `BoardBase`, která má metody pro práci s přiblížením a polohou celé plochy,
- menu kreslící plochy `BoardMenu`, v rámci něhož jsou použity komponenty `AddFloatingNodeBtn` pro přidání nového samostatného uzlu metodou drag & drop, `ZoomBar` pro zpracování uživatelského vstupu týkajícího se přiblížení a `BoardMenuBtn` pro spuštění prezentace
- kořenové a samostatné uzly (více viz sekce 5.3.4),
- editační menu plochy `ContextMenu`.

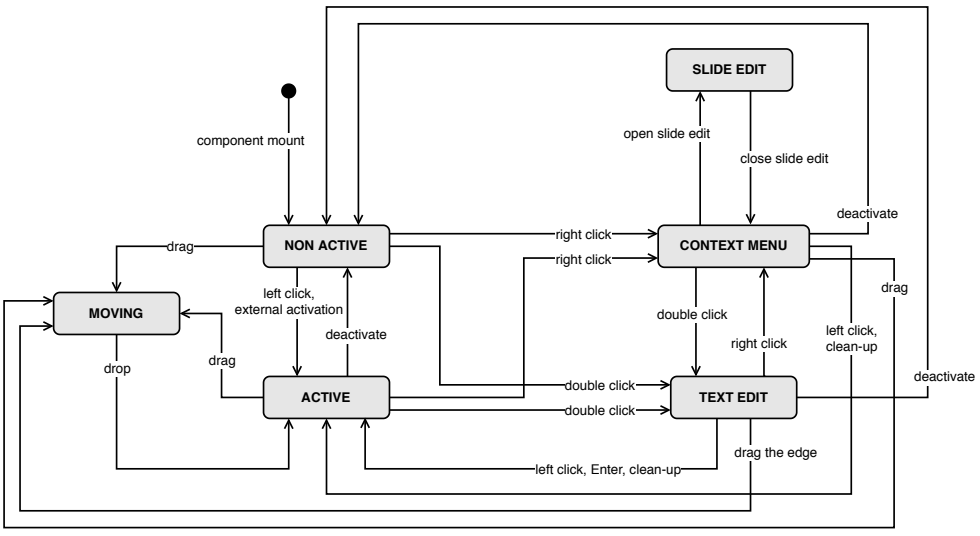


Obrázek 5.4: Model komponenty `DrawingBoard`

5.3.4 Node

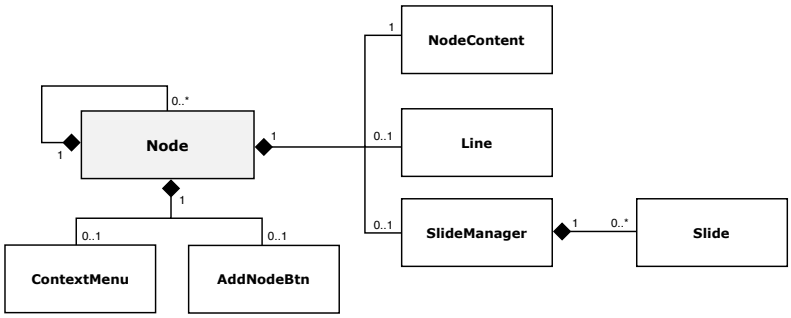
`Node` je hlavní komponenta myšlenkové mapy. Spravuje svůj vlastní stav a řídí vykreslování svého obsahu, svých podřízených uzlů, své příchozí linie a svých slajdů. Komponenta na základě informací v `props` určí polohu uzlu vůči jeho rodiči, jeho velikost, styl a textový obsah. Zpracovává jak vlastní změny, tak také změny veškerých vnořených komponent.

Uzel se může nacházet ve dvou základních stavech – aktivním a neaktivním. Aktivní uzel pak může mít buď otevřené editační menu, nebo může mít aktivní režim editace textu, nebo může mít otevřený slajd. Uzel také může být označen jako pohybující se. Detailní přehled stavů uzlu a přechodů mezi nimi je patrný z diagramu na obrázku 5.5.



Obrázek 5.5: Model stavů komponenty `Node`

Struktura komponenty Komponenta `Node` vykresluje celou řadu komponent (detailní struktura viz diagram na obrázku 5.6). Patří sem:



Obrázek 5.6: Model komponenty `Node`

- `NodeContent` – vlastní viditelný uzel včetně textového obsahu, zpracovává textový vstup od uživatele,
- `AddNodeBtn` – tlačítko pro přidání uzlu metodou drag & drop,
- `Line` – příchozí linie, pokud existuje (více viz sekce 5.3.5),
- `Node` – podřízené uzly,
- `ContextMenu` – editační menu, které zpracovává změny formátu,
- `SlideManager` – čistě logická komponenta, jejíž zodpovědností je správa slajdů (drží aktuálně zobrazený slajd a zpracovává informace o změnách slajdů) a zobrazení slajdů pomocí komponenty `Slide` (více viz sekce 5.3.6).

5.3.5 Line

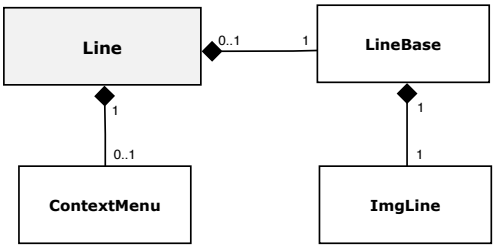
Komponenta `Line` slouží k vykreslení linie mezi dvěma uzly a správě jejího stavu. Styl linie určí komponenta na základě informací z `props`. Linie také sama zpracovává změny tohoto stylu.

V rámci svých `props` dále dostane linie informace o poloze uzlu vůči jeho rodičovskému uzlu a o výšce a šířce obou uzlů. Na základě těchto informací je určen směr linie (horizontální nebo vertikální), body začátku a konce linie a také odsazení linie vůči rodičovskému uzlu. Tyto hodnoty potom slouží k určení konkrétního tvaru linie.

Struktura komponenty Struktura komponenty je patrná z diagramu na obrázku 5.7 a sestává z následujících komponent:

- `LineBase`, která má metody pro výpočet tvaru linie,

- `ImgLine`, která na základě počátečního a koncového bodu vykreslí samotnou linii,
- `ContextMenu`, která představuje editační menu linie.



Obrázek 5.7: Model komponenty `Line`

Způsob vykreslení linie Linii bylo možné vykreslit buď pomocí elementu `canvas`, nebo pomocí elementu `svg`. Oba tyto elementy dle MDN [34, 36] podporují kreslení křivek, nicméně na rozdíl od elementu `canvas` může být `svg` formátováno pomocí CSS stylů a zpracování uživatelského vstupu (např. kliknutí) je jednodušší, protože umožňuje zachycovat běžné události.

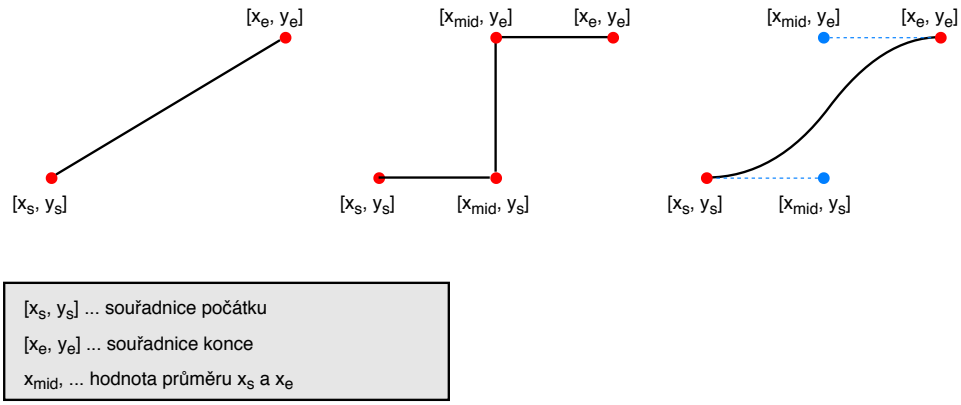
Pro vykreslení linie používám `svg` element `path`. Atributem, který určuje tvar linie, je `d`. Dle [36] používá tento atribut speciální syntaxi příkazů pro určení popisovaného tvaru. Pro požadované tvary jsou nejdůležitější příkazy:

- `M x,y` – přesun na bod daný souřadnicemi [x, y],
- `L x,y` – úsečka z aktuálního bodu do bodu určeného souřadnicemi [x, y],
- `C x1,y1 x2,y2 x,y` – kubická Bézierova křivka z aktuálního bodu do bodu určeného souřadnicemi [x, y], zakřivení v oblasti počátku je určeno kontrolním bodem [x1, y1], zakřivení v oblasti konce je určeno kontrolním bodem [x2, y2].

Rovná linie je tvořena pomocí jedné čáry, lomená čára je tvořena ze tří čar a hladká linie je tvořena jednou kubickou křivkou. Grafické znázornění význačných bodů pro horizontální linii je na obrázku 5.8. Body pro vertikální linie jsou obdobné.

5.3.6 Slide

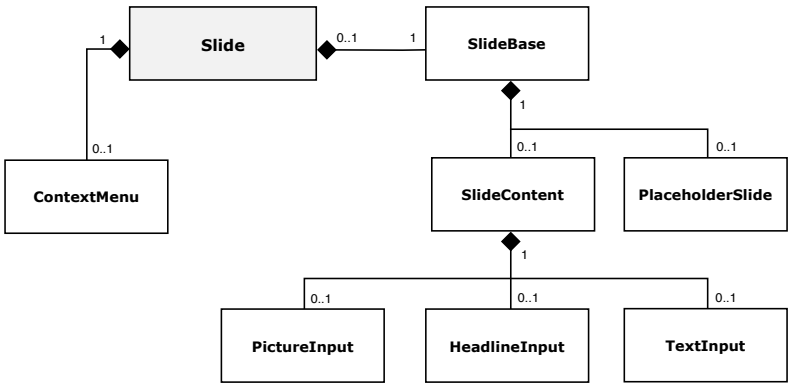
Komponenta `Slide` slouží k vykreslení slajdu a jeho obsahu. Slajd na základě informací z `props` určí své rozložení, formát i obsah. Zpracovává také změny všech těchto vlastností.



Obrázek 5.8: Význačné body linie a jejich souřadnice

Struktura komponenty Struktura komponenty je patrná z diagramu na obrázku 5.9. Nejdůležitější komponenty jsou:

- základová komponenta `SlideBase`, která vykresluje šipky pro procházení mezi slajdy, tlačítko pro zavření slajdu a samozřejmě taky vlastní obsah slajdu `SlideContent`, může vykreslit také `PlaceholderSlide`, který slouží pro přidávání nových slajdů,
- obsah slajdu `SlideContent`, který na základě rozložení vykreslí požadovanou strukturu slajdu včetně obsahu, k tomu využívá komponenty `HeadlineInput`, `TextInput` a `PictureInput`, které zobrazují aktuální obsah a zároveň zachycují a zpracovávají uživatelský vstup
- editační menu slajdu `ContextMenu`.



Obrázek 5.9: Model komponenty `Slide`

5.4 Zajímavé body implementace

5.4.1 Implementace pohybování objekty

Popis problému V aplikaci je několik objektů, kterými je možné pohybovat metodou drag & drop a které na toto tažení reagují různým způsobem. Jedná se o:

- uzel – část komponenty `Node`,
- kreslící plátno – část komponenty `DrawingBoard` (resp. `BoardBase`),
- tlačítko pro přidání podřízeného uzlu – `AddNodeBtn`,
- tlačítko pro přidání samostatného uzlu – `AddFloatingNodeBtn`.

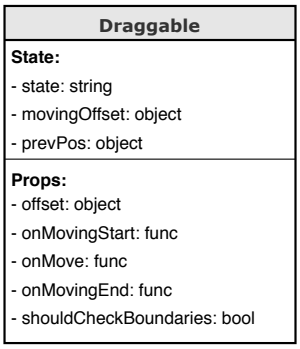
V reakci na tažení se všechny elementy chovají rozdílně – jinak zpracovávají svoji aktuální absolutní pozici, jinak reagují na začátek, průběh i ukončení tažení. Tažení samotné ale u všech komponent probíhá stejně v následujících krocích:

1. Na základě zaznamenání události `mousedown` začne sekvence tažení, je zaznamenána informace o poloze kurzoru.
2. Na základě zaznamenání informací z událostí `mousemove` jsou průběžně vypočítávány nové souřadnice pohybovaného objektu (z minulé a stávající pozice kurzoru), zaznamenává se nová pozice kurzoru.
3. Na základě události `mouseup` je ukončena sekvence tažení, jsou předány informace o nové poloze komponenty.

Řešení problému Pro vyřešení tohoto problému jsem se rozhodla využít doporučované skládání. Vytvořila jsem komponentu `Draggable`, která zachycuje události `mousedown`, `mousemove` a `mouseup`, odpovídajícím způsobem je zpracovává a pomocí callbacků informuje nadřazenou komponentu o změnách (předává novou polohu). Její struktura je patrná z diagramu na obrázku 5.10.

Tato komponenta musí obalovat celou viditelnou část elementu, se kterým má být pohybováno, aby mohla zachytit potřebné události. V rámci `props` dostává počáteční souřadnice elementu a tři callbacky `onMovingStart`, `onMove` a `onMovingEnd`, které volá na začátku, během a na konci tažení objektem. Všechny callbacky mají alespoň jeden parametr – změněné souřadnice.

Důležitým bodem na začátku tažení je registrace callbacků pro události `mousemove` a `mouseup` přímo k celému dokumentu. Během rychlého pohybu se může stát, že kurzor předběhne posouvaný objekt. Pokud by v tomto případě byly callbacky zaregistrovány pouze na konkrétní objekt, nedošlo by k zpracování informací o pohybu ani o upuštění a element by byl dále veden jako „pohybující se“, i když by zůstal na místě, na kterém ho kurzor opustil.



Obrázek 5.10: Model komponenty `Draggable`

5.4.2 Implementace prezentačních komponent

Popis problému Během prezentace se používají některé podobné komponenty jako během režimu tvorby. Chování prezentačních komponent se od běžných mírně liší – nemusí vůbec reagovat na změny myšlenkové mapy, jsou často vykresleny pouze staticky. Mají však i některé společné funkce a většinou také vypadají velmi podobně. Konkrétně se jedná o dvojice komponent:

- `DrawingBoard` a `PresentationDrawingBoard`
- `Node` a `PresentationNode`
- `Line` a `PresentationLine`
- `Slide` a `PresentationSlide`

Řešení problému V aplikaci tedy je pokaždé dvojice komponent, které se částečně chovají stejně, ale mají také každá své specifické chování. Tento problém je možné vyřešit skládáním – vytvořením základové generické komponenty, která bude sdružovat společné chování. Obě rozšiřující komponenty (běžná i prezentační) pak tuto komponentu použijí a nastaví ji na požadované hodnoty.

Tento postup má však i své nevýhody – pokud je rozdíl mezi běžnou komponentou a jejím prezentačním protějškem příliš velký, nebo pokud komponenta předává svým vnořeným komponentám mnoho hodnot a callbacků, potom může být nová struktura velmi nepřehledná – je tedy vždy nutné zvážit výhody a nevýhody u každé dvojice zvlášť.

Z tohoto důvodu byly komponenty `Node` a `PresentationNode` ponechány bez úprav. Tyto komponenty mají velmi málo společné logiky a jsou si podobné spíše strukturou metody `render` – tím, že obě vykreslují potomky, linii a slajdy. Jakýkoli pokus o sloučení v rámci této metody by však vedl pouze k znepřehlednění komponent a získané výhody by nebyly v podstatě žádné.

Pro společnou logiku proto byly vytvořeny externí funkce, které jsou importovány a používány oběma komponentami. Obě komponenty navíc používají pro vykreslení uzlu komponentu `NodeContent`.

U všech ostatních dvojic byla vytvořena odpovídající základová komponenta (`BoardBase`, `LineBase`, `SlideBase`), která sdružuje společnou funkcionalitu a je nastavitelná prostřednictvím `props`.

Kapitola 6

Testování

Hotová aplikace byla otestována dvěma základními způsoby. Automatizované testy pokrývají vnitřní logiku aplikace – sem spadají unit testy jednotlivých funkcí a integrační testy komponent. Testovací scénáře pak testují především vzhled a funkcionalitu uživatelského rozhraní.

6.1 Automatizované testy

Automatizované testy byly tvořeny již v průběhu vývoje. Zahrnují jak unit testy, které testují funkcionalitu každé komponenty v izolaci, tak i integrační testy, které testují komunikaci mezi komponentami. Tímto způsobem je otestována veškerá vnitřní logika aplikace, reakce na události i reakce na uživatelský vstup.

Každá komponenta má také automatizovaný „smoke test“, který kontroluje, zda je možné komponentu vykreslit v základní konfiguraci, resp. ve vícero konfiguracích, pokud je mezi nimi zásadní rozdíl.

Všechny automatizované testy při testování hotové aplikace dopadly bezchybně. Pokrytí řádek kódu testy je více než 90%.

6.1.1 Příklady testů

V této části uvedu několik reprezentativních příkladů automatizovaných testů a popíši, co testují a jak fungují.

Unit test komponenty Node Komponenta `Node` musí dokázat zpracovat změnu textu. K tomuto účelu má metodu `handleTextChange`, jejíž kód je na výpisu 6.1. Tato metoda dostane jako parametr nový text, vytvoří kopii aktuální hodnoty uzlu (tj. objektu se všemi vlastnostmi uzlu) a v ní změní hodnotu atributu `text`. Poté pomocí callbacku informuje o změně nadřazenou komponentu.

```
export default class Node extends Component {
  handleTextChange(newText) {
    let newValue = { ...this.props.value };
    newValue.text = newText;
    this.props.onChange(newValue);
  }
}
```

Výpis kódu 6.1: Metoda `handleTextChange` komponenty `Node` (kód zkrácen pro přehlednost)

V rámci unit testu této funkce tedy budu chtít otestovat, zda je callback skutečně zavolán se správnou hodnotou a zda je skutečně vytvořena kopie objektu aktuální hodnoty a není pouze měněn stávající objekt.

Kód unit testu je na výpisu 6.2. Nejprve je vykreslena komponenta `Node` se základními hodnotami `props`. Poté jsou zkontrolovány počáteční podmínky – callback nebyl zavolán. Následně je zavolána testovaná metoda a jsou zkontrolovány výsledky – zda byl callback zavolán, zda byl zavolán se správnou hodnotou a zda nedošlo ke změně hodnoty uzlu v `props` (které jsou read-only a mají být neměnné).

```
it('changes text', () => {
  const mockCallback = jest.fn(() => { })
  const wrapper = mount(
    <Node
      value={{ text: 'test', /*(...)*/ }}
      onChange={mockCallback}
      // (...)
    />
  );

  expect(mockCallback.mock.calls.length).toEqual(0);
  wrapper.instance().handleTextChange('abcd 123')
  expect(mockCallback.mock.calls.length).toEqual(1);
  expect(mockCallback.mock.calls[0][0]).toEqual(
    { text: 'abcd 123', /*(...)*/ }
  );
  expect(wrapper.props().value).toEqual(
    { text: 'test', /*(...)*/ }
  );
});
```

Výpis kódu 6.2: Unit test pro metodu `handleTextChange` komponenty `Node`

Integrační test komponenty ContextMenu pro Node Hlavní podstata integračních testů v pojetí této aplikace je otestovat, zda byly vnořeným komponentám předány správné callbacky. Jejich důležitost stoupá, pokud callbackem není přímo metoda komponenty, ale anonymní funkce, která tuto metodu volá.

Příkladem může být metoda `handleStyleChange` editačního menu uzlu (`ContextMenu` komponenty `Node`), která zpracovává změny v objektu manuálního stylu a informuje o nich komponentu `Node`. Tato metoda musí zpracovat změny různých atributů, proto jako parametr dostává objekt ve tvaru `{atribut: nováHodnota}`. Její kód je na výpisu 6.3. Na tomto výpisu je rovněž příklad předání této metody vnořené komponentě `ColorSelect` (jedná se o úryvek kódu z definice formuláře editačního menu).

```
export default class ContextMenu extends Component {
  handleStyleChange(changeObject) {
    const newStyle = { ...this.props.style, ...changeObject };
    this.props.onStyleChange(newStyle);
  }
}

<ColorSelect
  value={currentStyle.backgroundColor}
  onChange={(c) => this.handleStyleChange({ backgroundColor: c })}
/>
```

Výpis kódu 6.3: Metoda `handleStyleChange` komponenty `ContextMenu` a příklad jejího použití (kód zkrácen a upraven pro přehlednost)

V rámci integračního testu proto budu chtít otestovat, že pokud komponenta `ColorSelect` zavolá callback `onChange` se správným atributem, tak bude tato změna v komponentě `ContextMenu` korektně zpracována a bude zavolán callback s odpovídající hodnotou.

Kód integračního testu je na výpisu 6.4. Podobně jako u předchozího unit testu je nejprve vykreslena komponenta `ContextMenu` v základní konfiguraci. Poté jsou zkontrolovány počáteční podmínky. Následně je vybrán hledaný `ColorSelect` a uměle je z něj zavolán callback s příslušnou hodnotou (to, že komponenta v reakci na uživatelský vstup tento callback skutečně zavolá, je její zodpovědností, a proto je to součástí jejích testů, nikoli tohoto). Po zavolání jsou zkontrolovány výsledky – z `ContextMenu` byl zavolán callback s požadavkem na změnu stylu a jako parametr nesl správně upravenou kopii objektu stylu.

```
it('reacts correctly to form changes', () => {
  const mockCallbackStyle = jest.fn(() => { });
  const wrapper = mount(<ContexMenu
    style={{ backgroundColor: '#ffffff', /*(...)*/ }}
    onStyleChange={mockCallbackStyle}
    // (...)
  />);

  expect(mockCallbackStyle.mock.calls.length).toEqual(0);
  wrapper.setState({ openedPage: 1 });

  wrapper.find('ColorSelect').at(0).props().onChange('#df5618');
  expect(mockCallbackStyle.mock.calls.length).toEqual(1);
  expect(mockCallbackStyle.mock.calls[0][0]).toEqual(
    { backgroundColor: '#df5618', /*(...)*/ }
  );
});
```

Výpis kódu 6.4: Integrační test pro metodu `handleStyleChange` komponenty `ContextMenu` (kód zkrácen a upraven pro přehlednost)

6.2 Testovací scénáře

Testovací scénáře byly vytvořeny po dokončení aplikace a simulují chování uživatele při používání aplikace. Kontrolují zejména vzhled a funkčnost uživatelského rozhraní – tedy vlastnosti, které nepokrývají automatizované testy.

Kompletní seznam testovacích scénářů je v příloze D. Každý scénář obsahuje počáteční stav aplikace, seznam kroků testu a zhodnocení výsledku.

Testy podle scénářů jsem provedla na prohlížečích Mozilla Firefox a Google Chrome (dle požadavků). Během testů byly objeveny pouze drobné, především vizuální chyby, které nenarušovaly funkčnost aplikace. Tyto chyby byly opraveny.

Závěr

Naplnění cílů

V této práci jsem se zabývala tvorbou webové aplikace pro interaktivní myšlenkové mapy. Na základě výsledků hodnocení existujících aplikací vznikl soubor požadavků, který sloužil jako podklad pro návrh uživatelského rozhraní i struktury aplikace. Aplikaci jsem implementovala a otestovala – jak automaticky, tak z pohledu uživatele pomocí testovacích scénářů.

Hlavním cílem bylo vytvoření webové aplikace, která bude umožňovat obohacení myšlenkové mapy o interaktivní prvky. Tento cíl byl splněn díky přidání stránek prezentace (slajdů) k uzlům. Mezi vedlejší cíle patřilo formátování prvků pomocí tříd a prezentační režim s využitím interaktivních prvků. Oba tyto rozšiřující cíle byly rovněž splněny. Dle požadavků byl rovněž důraz kladen na přívětivé uživatelské rozhraní.

Možnosti dalšího rozšíření

Aplikace je plně funkční, přesto existuje celá řada možných vylepšení. Dle specifikace cílů jsem se během návrhu i implementace zaměřila na novou funkcionalitu a z té běžné jsem implementovala pouze nutné minimum. Proto by do budoucna bylo vhodné tuto funkcionalitu doplnit – jmenovitě sem spadá synchronizace se serverem, uživatelské účty a ukládání map, možnost vrátit zpět kroky při tvoření mapy, možnost přepojovat uzly, kopírovat je, možnost zabalovat a rozbalovat uzly atd.

Kromě této běžné funkcionality je ale možné přidat také další nové funkce, které u současných aplikací běžné nejsou. Je možné zejména rozšířit stávající systém formátování pomocí tříd o neúplně specifikované třídy, tvoření tříd na základě existujícího manuálního stylu, klonování stylů, vzájemné dědění stylů, možnost nastavit manuálně výchozí styly. Z hlediska interaktivních prvků by bylo vhodné do budoucna přidat více možných rozložení, možnost tvořit ta-

bulky, rozšířit možnosti formátování textu. Prezentační režim by bylo možné rozšířit o automatizované procházení uzlů například pomocí šipek na klávesnici.

Zhodnocení výsledků a využitelnost v praxi

Aplikace je velmi dobře využitelná pro tvorbu myšlenkových map se zaměřením na jejich budoucí prezentaci. Tvorba mapy je uživatelsky přívětivá a rychlá – vše usnadňuje formátování pomocí tříd. Prezentace samotná je přehledná, a to zejména díky využití interaktivních prvků pro rozšíření mapy o další obsah.

Literatura

1. BUZAN, Tony; BUZAN, Barry. *Myšlenkové mapy: probuďte svoji kreativitu, zlepšete svou paměť, změňte svůj život*. 2. vyd. Brno: BizBooks, 2012. ISBN 978-80-265-0030-8.
2. BUZAN, Tony. *Mentální mapování*. Praha: Portál, 2007. ISBN 978-80-7367-200-3.
3. SAUF POMPIERS LTD. *MindMup* [online software]. © 2013–2018 [cit. 2018-11-20]. Dostupné z: `https://www.mindmup.com/`.
4. COGGLEIT LIMITED. *Coggle* [online software]. 2018 [cit. 2018-11-21]. Dostupné z: `https://coggle.it/`.
5. LKCOLLAB, LLC. *bubbl.us* [online software]. © 2018 [cit. 2018-11-21]. Dostupné z: `https://bubbl.us/`.
6. EXAMTIME LTD. *GoConqr* [online software]. © 2018 [cit. 2018-11-21]. Dostupné z: `https://www.goconqr.com/`.
7. REALTIMEBOARD. *RaltimeBoard* [online software]. © 2018 [cit. 2018-12-03]. Dostupné z: `https://realtimeboard.com/`.
8. LUCID SOFTWARE INC. *Lucidchart* [online software]. © 2018 [cit. 2018-12-03]. Dostupné z: `https://www.lucidchart.com/`.
9. MEISTERLABS. *mindmeister* [online software]. © 2018 [cit. 2018-12-03]. Dostupné z: `https://www.mindmeister.com/`.
10. CANVA. *Canva* [online software]. © 2019 [cit. 2019-01-19]. Dostupné z: `https://canva.com/`.
11. VENNGAGE INC. *Venngage* [online software]. © 2011–2019 [cit. 2019-01-19]. Dostupné z: `https://venngage.com/`.
12. *FreeMind* [software]. © 2019 [cit. 2019-01-10]. Dostupné z: `http://freemind.sourceforge.net/wiki/index.php/Main_Page`.

13. LØFGREN, Tobias. *Text2MindMap* [online software]. © 2019 [cit. 2019-01-10]. Dostupné z: https://tobloef.com/text2mindmap/.

14. SKETCHBOARD. *Sketchboard* [online software]. © 2017 [cit. 2019-01-10]. Dostupné z: https://sketchboard.me/.

15. CINERGIX PTY. LTD. *Creately* [online software]. © 2008–2019 [cit. 2019-01-14]. Dostupné z: https://creately.com/.

16. SCHUSTER, Stefan. *Mind42* [online software]. © 2007–2019 [cit. 2019-01-14]. Dostupné z: https://mind42.com/.

17. PREZI INC. *Prezi* [online software]. © 2019 [cit. 2019-01-20]. Dostupné z: https://prezi.com/.

18. *WiseMapping* [online software]. 2019 [cit. 2019-01-12]. Dostupné z: http://www.wisemapping.com/.

19. JGRAPH LTD. *draw.io* [online software]. 2019 [cit. 2019-01-12]. Dostupné z: https://www.draw.io/.

20. RICHARD, David. *MindmapMaker* [online software]. © 2011 [cit. 2019-01-12]. Dostupné z: https://app.mindmapmaker.org/.

21. BLUMIND. *Blumind* [software]. 2019 [cit. 2019-01-15]. Dostupné z: https://blumind.en.softonic.com/.

22. TUFTS UNIVERSITY. *Visual Understanding Environment* [software]. © 2015 [cit. 2019-01-15]. Dostupné z: https://vue.tufts.edu/.

23. EDRAWSOFT. *MindMaster* [software]. © 2014–2019 [cit. 2019-01-18]. Dostupné z: https://www.edrawsoft.com/mindmaster/.

24. Roboto. *Google Fonts* [online] [cit. 2019-04-16]. Dostupné z: https://fonts.google.com/specimen/Roboto?selection.family=Roboto.

25. *React* [online]. Facebook Inc., © 2019 [cit. 2019-04-06]. Dostupné z: https://reactjs.org/.

26. *Sass basics* [online]. Sass, © 2006–2019 [cit. 2019-04-06]. Dostupné z: https://sass-lang.com/guide.

27. *Jest* [online]. Facebook Inc., © 2019 [cit. 2019-04-06]. Dostupné z: https://jestjs.io/.

28. *Enzyme* [online] [cit. 2019-04-06]. Dostupné z: https://airbnb.io/enzyme/.

29. *React Docs* [online]. Facebook Inc., © 2019 [cit. 2019-04-19]. Dostupné z: https://reactjs.org/docs/.

30. PATHY, Madhu. Learn Basics of React.js in 11 Minutes. *Medium* [online]. 2018 [cit. 2019-04-19]. Dostupné z: https://medium.com/@madhupathy/learn-basics-of-react-js-in-3-minutes-a94cbc6f02c8.

31. KOSTRZEWA, Denis. Is React.js the Best Javascript Framework in 2018? *Bejamas* [online]. 2018 [cit. 2019-04-19]. Dostupné z: https://bejamas.io/blog/react-best-javascript-framework-2018/.

32. BUNA, Samer. All the fundamental React.js concepts, jammed into this single Medium article. *Medium – Free Code Camp* [online]. 2017 [cit. 2019-04-19]. Dostupné z: https://medium.freecodecamp.org/all-the-fundamental-react-js-concepts-jammed-into-this-single-medium-article-c83f9b53eac2.

33. WEST, Donavon. Clean Code vs. Dirty Code: React Best Practices. *American Express Technology* [online]. 2017 [cit. 2019-04-19]. Dostupné z: https://americanexpress.io/clean-code-dirty-code/.

34. *Canvas API* [online]. MDN web docs, © 2005–2019 [cit. 2019-04-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API.

35. ABRAMOV, Dan. React v15.0 [online]. 2016 [cit. 2019-04-30]. Dostupné z: https://reactjs.org/blog/2016/04/07/react-v15.html.

36. *SVG: Scalable Vector Graphics* [online]. MDN web docs, © 2005–2019 [cit. 2019-04-20]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/SVG.

37. Design Patterns - Strategy Pattern. *Tutorials Point* [online] [cit. 2019-04-20]. Dostupné z: https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm.

Dodatek A

Seznam použitých zkratek

API Application Programming Interface

CSS Cascading Style Sheets

DRY Don't Repeat Yourself

HTML Hypertext Markup Language

JSON JavaScript Object Notation

JSX JavaScript XML

SVG Scalable Vector Graphics

UML Unified Modeling Language

Dodatek B

Instalační příručka

B.1 Použití online verze

Aplikace je dostupná na webové adrese `http://mindmaps.9e.cz/`, odkud může být přímo spuštěna a používána.

B.2 Spuštění z přiložených souborů

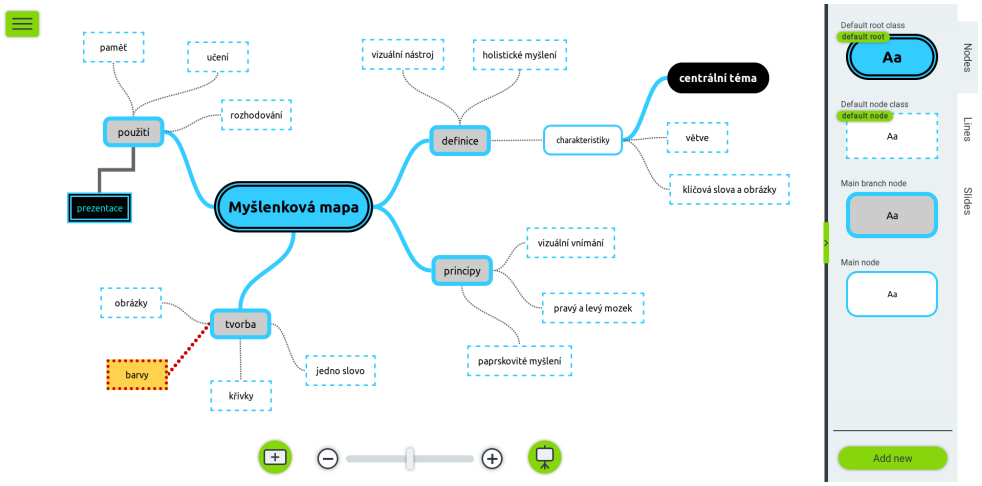
Pro spuštění aplikace z přiložených souborů je třeba mít nainstalovaný *Node.js* společně se správcem balíčků *npm*. Poté je možné spustit aplikaci v následujících krocích.

1. V terminálu se přesuňte do kořenového adresáře zdrojového kódu.
2. Zadejte příkaz `npm install`, který nainstaluje potřebné balíčky.
3. Zadejte příkaz `npm start`, který zkompiluje a spustí aplikaci na lokální adrese `http://localhost:3000`.

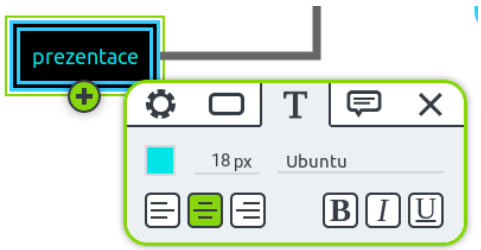
Dále je možné spustit testy příkazem `npm test`, případně ověřit pokrytí kódu testy příkazem `npm test -- --coverage`.

Dodatek C

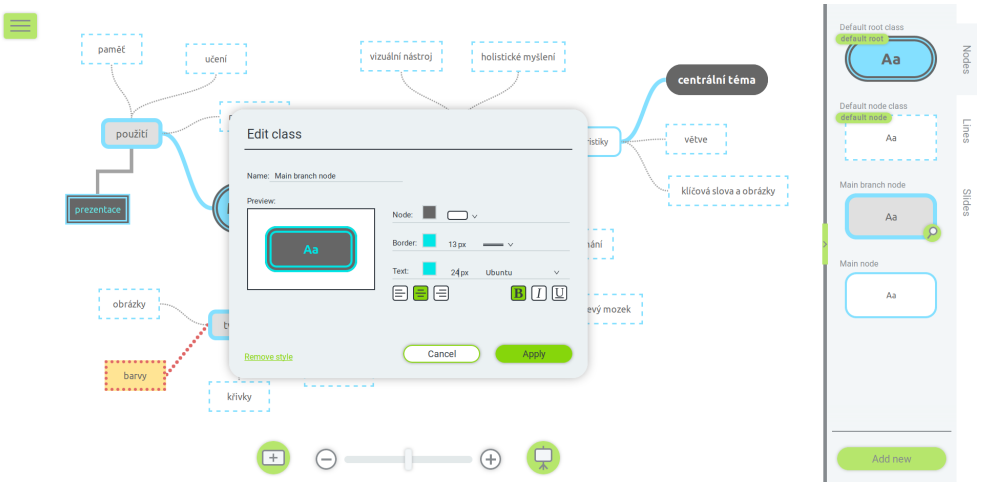
Ukázka výsledné aplikace



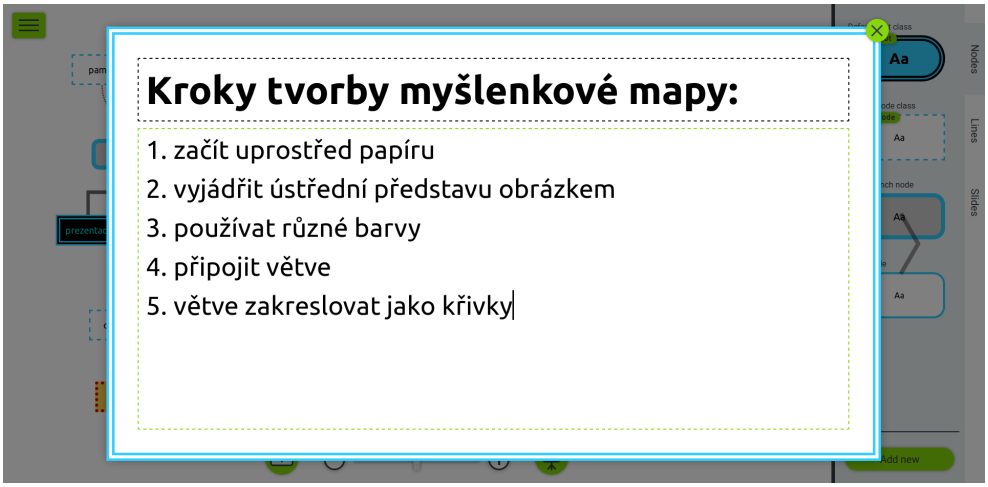
Obrázek C.1: Aplikace s nakreslenou myšlenkovou mapou (obsah inspirován poznatky o myšlenkových mapách z kapitoly 1)



Obrázek C.2: Editační menu uzlu



Obrázek C.3: Formulář pro editaci třídy stylu



Obrázek C.4: Úprava obsahu slajdu (obsah inspirován poznatky o myšlenkových mapách z kapitoly 1)

Dodatek D

Testovací scénáře

D.1 Tvorba kostry mapy

D.1.1 Tvoření uzlů

Výchozí stav: nově načtená aplikace

1. Na obrazovce je jeden aktivní uzel – má zelený rámeček, je viditelné plus tlačítko pro přidání podřízeného uzlu.

2. Tažení a upuštění plus tlačítka volně do prostoru kreslící plochy – během tažení spojuje tlačítko a rodičovský uzel přerušovaná zelená čára, po upuštění vznikne na místě nový uzel, nový uzel je označený jako aktivní, kořenový uzel není aktivní.

3. Kliknutí levým tlačítkem myši na kořenový uzel – kořenový uzel je označen jako aktivní, potomek není aktivní.

4. Tažení a upuštění plus tlačítka mimo okno prohlížeče – nový uzel není vytvořen, kořenový uzel zůstává aktivní.

5. Vytvoření stromu myšlenkové mapy s alespoň deseti uzly, alespoň čtyři úrovně, alespoň jeden uzel se třemi a více potomky, z nichž každý má také alespoň jednoho potomka, všechny uzly je možné aktivovat, vždy aktivní pouze jeden uzel.

6. Tažení a upuštění tlačítka pro přidání plovoucího uzlu z menu volně do prostoru kreslící plochy – na místě je vytvořen nový plovoucí uzel, uzel je označen jako aktivní, původní kořenový uzel není aktivní.

7. Tažení a upuštění tlačítka pro přidání plovoucího uzlu z menu mimo okno prohlížeče – nový uzel není vytvořen, žádný uzel není aktivní.

8. Vytvoření stromu myšlenkové mapy s alespoň deseti uzly pod plovoucím uzlem, alespoň čtyři úrovně, alespoň jeden uzel se třemi a více potomky, z nichž každý má také alespoň jednoho potomka, všechny uzly je možné aktivovat, vždy aktivní pouze jeden uzel.

Výsledek: OK
Poznámka: v prohlížeči Mozilla Firefox nebyl nově vytvořený plovoucí uzel označen jako aktivní, chyba byla opravena

D.1.2 Pohybování s uzly

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem

1. Označení libovolného potomka jako aktivního – pouze jeden aktivní uzel.
2. Tažení aktivního uzlu na jiné libovolné místo na kreslící ploše – při tažení se uzel přesouvá zároveň s myší a zůstává aktivní, po upuštění zůstane na novém místě, je označen jako aktivní.
3. Tažení neaktivního uzlu na jiné libovolné místo na kreslící ploše – při tažení se uzel přesouvá zároveň s myší a stává se jediným aktivním uzlem, po upuštění zůstane na novém místě, je označen jako aktivní.
4. Tažení libovolného uzlu s potomky na jiné místo na kreslící ploše – potomci se pohybují zároveň s přesouvaným uzlem, ostatní uzly se nepohybují.
5. Tažení kořenového uzlu na jiné místo na kreslící ploše – celý strom se pohybuje zároveň s uzlem, plovoucí uzel se nepohybuje.
6. Tažení libovolného uzlu mimo okno prohlížeče – po upuštění se uzel vrátí zpět na původní místo.

Výsledek: OK

D.1.3 Mazání uzlů

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem

1. Označení libovolného listu jako aktivního – pouze jeden aktivní uzel.
2. Kliknutí pravým tlačítkem myši na aktivní uzel – otevře se editační menu, uzel zůstane aktivní.
3. Kliknutí na tlačítko smazání uzlu – uzel je smazán, žádný uzel není aktivní.

4. Kliknutí pravým tlačítkem myši na neaktivní uzel s potomkem – otevře se editační menu, uzel označen jako aktivní.

5. Kliknutí na tlačítko smazání uzlu – uzel je smazán včetně potomka.

6. Podobným stylem je možné smazat kořenový uzel – uzel je smazán včetně všech potomků.

7. Podobným stylem je možné smazat plovoucí uzel – kreslící plocha je prázdná.

Výsledek: OK

D.2 Práce s kreslící plochou

D.2.1 Pohyb s kreslící plochou

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem

1. Tažení kreslící plochy na libovolné místo uvnitř prohlížeče – všechny uzly se posouvají, žádný uzel není aktivní.

2. Tažení kreslící plochy na libovolné místo mimo okno prohlížeče – v momentě, kdy myš opustí okno, se pohyb plochy zastaví, po puštění tlačítka zůstane kreslící plocha v této pozici.

3. Tažení plochy tak, že je vidět její okraj, pokus o posunutí dále stejným směrem – kreslící plocha se už dále neposouvá, zůstává v pozici, kdy je její okraj na kraji okna pro tvorbu mapy.

Výsledek: OK

D.2.2 Přibližování a oddalování kreslící plochy

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem, je vyplněný textový obsah uzlů

1. Přiblížení kreslící plochy pomocí posuvníku – žádný uzel není aktivní, vše se zvětšuje se správnými proporcemi, centrum přiblížení je uprostřed stránky.

2. Oddálení kreslící plochy pomocí posuvníku – vše se zmenšuje se správnými proporcemi, centrum oddálení je uprostřed stránky, v případě, že je okno prohlížeče větší než oddálená kreslící plocha, je plocha uprostřed okna.

3. Přiblížení a oddálení plochy pomocí tlačítek po stranách posuvníku – vše se správně proporčně zvětšuje/zmenšuje.

4. Vytvoření potomka kořenového uzlu a plovoucího uzlu při maximálním přiblížení mapy – uzly se vytvoří na správném místě a s očekávanými rozměry a stylem, uzlům je možné vyplnit text

Výsledek: OK

D.2.3 Změna velikosti kreslící plochy

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem

1. Maximální oddálení kreslící plochy – jsou viditelné okraje, plocha je vycentrovaná.

2. Kliknutí pravým tlačítkem myši na kreslící plochu – otevře se editační menu.

3. Zadání nových rozměrů do menu – změní se velikost plochy, odsazení uzlů od levého horního rohu se nemění, plocha je stále vycentrovaná.

Výsledek: OK

D.2.4 Vycentrování obsahu na obrazovce

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a s alespoň jedním plovoucím uzlem, pohled není vycentrovaný, plocha je maximálně přiblížená

1. Kliknutí pravým tlačítkem myši na kreslící plochu – otevře se editační menu.

2. Kliknutí na tlačítko vycentrování na obrazovce – upraví se poloha kreslící plochy a její přiblížení tak, aby se celá mapa vešla na obrazovku a pokrývala její maximální část.

Výsledek: OK

D.2.5 Otevření nové mapy

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem, s alespoň jedním plovoucím uzlem, se změněnou definicí stylů a se změněnou velikostí kreslící plochy, pohled není vycentrovaný

1. Kliknutí na tlačítko `New map` v hlavním menu – existující mapa zmizí, je vytvořen pouze jeden počáteční uzel, který je vycentrován na stránce, nachází se uprostřed kreslící plochy, která má opět počáteční velikost, seznam tříd zůstává upravený.

Výsledek: OK

D.2.6 Otevření nápovědy

Výchozí stav: nově načtená aplikace

1. Kliknutí na tlačítko `Help` v hlavním menu – otevře se nápověda překrývající celé okno, text nápovědy je možné scrollovat.
2. Kliknutí na tlačítko `Cancel` – zavře se nápověda.

Výsledek: OK

D.3 Interaktivní prvky

D.3.1 Přidávání, procházení a odebírání slajdů

Výchozí stav: nově načtená aplikace

1. Kliknutí pravým tlačítkem myši na kořenový uzel – otevře se editační menu.
2. Kliknutí na tlačítko spravování interaktivních prvků – spuštění režimu editace slajdů, zobrazen slajd s tlačítkem pro přidání dalšího slajdu.
3. Kliknutí na tlačítko pro přidání slajdu – přidán a zobrazen nový slajd.
4. Kliknutí na šipku doprava – zobrazení slajdu s tlačítkem.
5. Přidání dalšího slajdu – zobrazení nového slajdu.
6. Kliknutí na šipku doprava – zobrazení slajdu s tlačítkem.
7. Kliknutí na šipku doleva – zobrazení druhého slajdu.
8. Kliknutí pravým tlačítkem kamkoli do prostoru slajdu – otevře se editační menu.
9. Kliknutí na tlačítko vymazaní slajdu – slajd je smazán, je otevřen první slajd.
10. Kliknutí na tlačítku pro ukončení režimu editace slajdů – slajd se zavře, kořenový uzel je stále aktivní.

Výsledek: OK

D.3.2 Editace obsahu slajdu

Výchozí stav: nově načtená aplikace

1. Kliknutí pravým tlačítkem myši na kořenový uzel – otevře se editační menu.
2. Kliknutí na tlačítko spravování interaktivních prvků – spuštění režimu editace slajdů, zobrazen slajd s tlačítkem pro přidání dalšího slajdu.
3. Kliknutí na tlačítko pro přidání slajdu – přidán a zobrazen nový slajd.
4. Kliknutí pravým tlačítkem kamkoli do prostoru slajdu – otevře se editační menu.
5. Změna rozložení slajdu na poslední v nabídce (obsahuje jedno obrázkové a dvě textová pole) – změna rozložení.
6. Kliknutí na textové pole, zadání textu – text se zapíše do pole.
7. Dvojklik na obrázkové pole – otevře se nabídka pro nahrání obrázku.
8. Výběr obrázku – obrázek je nahrán do obrázkového pole, obrázek je zmenšen tak, aby se do pole vešel celý.
9. Kliknutí na ikonku pro smazání obrázkového obsahu – obrázek je smazán.

Výsledek: OK

D.4 Formátování prvků

D.4.1 Editace textu uzlu

Výchozí stav: aplikace s kořenovým uzlem s alespoň jedním potomkem

1. Dvojklik na aktivní kořenový uzel – aktivace režimu úpravy textu.
2. Napsání delšího textu (cca 30 znaků) na klávesnici – text se zapíše do uzlu a uzel se rozšíří tak, aby byl všechen text viditelný.
3. Kliknutí doprostřed stávajícího textu a napsání textu na klávesnici – text se zapíše na dané místo, uzel se dále rozšiřuje.
4. Smazání několika znaků pomocí klávesy `Backspace` – znaky jsou smazány, zbytek textu beze změny, šířka uzlu se nemění.
5. Označení celého textu a jeho smazání – text se smaže, šířka uzlu se nemění.

6. Zapsání nového textu do uzlu – text se zapíše, šířka uzlu s nemění, pokud to není potřeba.
7. Zmáčknutí klávesy `Enter` – ukončení režimu úpravy textu, kořenový uzel zůstává aktivní, text se nemění.
8. Dvojklik na neaktivního potomka – potomek je aktivní se zapnutým režimem úpravy textu.
9. Zapsání textu do potomka – text je možné zapsat.
10. Kliknutí na kořenový uzel – kořenový uzel aktivní, ukončení režimu úpravy textu potomka, text zůstal zapsaný v potomkovi.

Výsledek: OK

D.4.2 Změna velikosti uzlu

Výchozí stav: nově načtená aplikace

1. Kliknutí pravým tlačítkem myši na aktivní kořenový uzel – objeví se editační menu.
2. Změna šířky i výšky v daných polích (pomocí šipek, nebo pomocí napsání a zmáčknutí `Enter`) – změna pomocí šipek se projevuje okamžitě, změna pomocí napsání nové velikosti se projeví až po zmáčknutí `Enter`.

Výsledek: OK

D.4.3 Změna manuálního formátu prvků

Výchozí stav: aplikace s kořenovým uzlem s alespoň jedním potomkem

1. Kliknutí pravým tlačítkem myši na aktivní kořenový uzel – objeví se editační menu, k uzlu je přiřazena výchozí třída stylu.
2. Změna všech vlastností v záložkách pro formát uzlu a formát textu – změny se okamžitě projevují na editovaném uzlu, k uzlu není přiřazena žádná třída.
3. Kliknutí pravým tlačítkem na linii mezi uzly – zmizí editační menu uzlu, objeví se editační menu linie, k linii je přiřazena výchozí třída stylu.
4. Změna všech vlastností v záložce pro formát linie – změny se okamžitě projeví na editované linii, k linii není přiřazena žádná třída.
5. Otevření správy interaktivních prvků, přidání slajdu k libovolnému uzlu, zapsání textu do libovolného pole – zmizí editační menu linie, text je zapsán.

6. Kliknutí pravým tlačítkem myši na slajd – objeví se editační menu, ke slajdu je přiřazena výchozí třída stylu.

7. Změna všech vlastností v záložkách pro formát slajdu a formát textu – změny se okamžitě projeví na editovaném slajdu, ke slajdu není přiřazena žádná třída.

Výsledek: OK

Poznámka: v prohlížeči Mozilla Firefox bylo editační menu linie příliš úzké a mělo proto nevhodné rozložení, chyba byla opravena

D.4.4 Přidávání tříd stylu

Výchozí stav: nově načtená aplikace

1. Kliknutí na tlačítko `Add new` v panelu tříd – otevře se formulář pro přidání třídy stylu pro uzel.

2. Změna všech hodnot ve formuláři – všechny změny formátu se ukazují v náhledu stylu.

3. Potvrzení změn kliknutím na tlačítko `Apply` – formulář se zavře, třída se objeví v panelu, náhled odpovídá nastaveným hodnotám.

4. Kliknutí na tlačítko `Add new` v panelu tříd – otevře se formulář pro přidání třídy stylu pro uzel.

5. Kliknutí na tlačítko `Cancel` – formulář se zavře, třída se neobjeví v panelu.

6. Kliknutí na záložku `Lines` v panelu – objeví se seznam tříd stylu pro linie.

7. Přidání nové třídy stylu linie stejným způsobem jako u uzlu – všechny změny formátu se ukazují v náhledu, po potvrzení změn se třída objeví v seznamu, náhled odpovídá nastaveným hodnotám.

8. Kliknutí na záložku `Slides` v panelu – objeví se seznam tříd stylu pro slajdy.

9. Přidání nové třídy stylu slajdu stejným způsobem jako u uzlu – všechny změny formátu (kromě změny poměru stran) se ukazují v náhledu, po potvrzení změn se třída objeví v seznamu, náhled odpovídá nastaveným hodnotám.

Výsledek: OK

D.4.5 Mazání tříd stylu

Výchozí stav: aplikace s alespoň jednou nevýchozí třídou stylu pro uzel, linii i slajd, panel tříd na záložce `Nodes`

1. Kliknutí na ikonu lupy u nějakého nevýchozího stylu – otevře se formulář pro úpravu stylu, formulář obsahuje i link pro smazání stylu.
2. Kliknutí na link `Remove style` – formulář se zavře, styl zmizí z nabídky.
3. Kliknutí na záložku `Lines` v panelu – objeví se seznam tříd stylu pro linie.
4. Smazání třídy stylu linie stejným způsobem jako u uzlu – formulář se zavře, třída zmizí z nabídky.
5. Kliknutí na záložku `Slides` v panelu – objeví se seznam tříd stylu pro slajdy.
6. Smazání třídy stylu slajdu stejným způsobem jako u uzlu – formulář se zavře, třída zmizí z nabídky.
7. Postupné otevření a zavření formulářů pro všechny výchozí styly uzlů, linií i slajdů – formulář neobsahuje link pro smazání stylu.

Výsledek: OK

D.4.6 Změna tříd stylu

Výchozí stav: Aplikace s kořenovým uzlem s jedním potomkem, uzly mají vyplněný text, kořenový uzel má jeden vytvořený slajd s textem, vše formátováno pomocí výchozích tříd stylu, panel tříd na záložce `Nodes`

1. Kliknutí na ikonu lupy u výchozího stylu pro kořenový uzel – otevře se formulář pro úpravu stylu.
2. Změna všech hodnot ve formuláři – všechny změny formátu se ukazují v náhledu stylu.
3. Potvrzení změn kliknutím na tlačítko `Apply` – formulář se zavře, náhled třídy v panelu odpovídá nastaveným hodnotám, zároveň se odpovídajícím způsobem změní styl kořenového uzlu.
4. Kliknutí na ikonu lupy u výchozího stylu pro běžný uzel – otevře se formulář pro úpravu stylu.
5. Změna všech hodnot ve formuláři – všechny změny formátu se ukazují v náhledu stylu.

6. Zrušení změn kliknutím na tlačítko `Cancel` – formulář se zavře, náhled třídy v panelu se nezmění, styl potomka se také nezmění.

7. Kliknutí na záložku `Lines` v panelu – objeví se seznam tříd stylu pro linie.

8. Úprava výchozího stylu linie stejným způsobem jako u uzlu – všechny změny formátu se ukazují v náhledu stylu, po potvrzení se změní náhled třídy v panelu a změní se také styl linie mezi uzly.

9. Kliknutí na záložku `Slides` v panelu – objeví se seznam tříd stylu pro slajdy.

10. Úprava výchozího stylu linie stejným způsobem jako u uzlu – všechny změny formátu (kromě změny poměru stran) se ukazují v náhledu stylu, po potvrzení se změní náhled třídy v panelu a změní se také styl slajdu u kořenového uzlu.

Výsledek: OK
Poznámka: při některých poměrech stran byly šipky pro přepínání mezi slajdy umístěny příliš blízko k slajdu, chyba byla opravena

D.4.7 Přiřazení, změna a odebrání tříd stylu

Výchozí stav: aplikace s kořenovým uzlem s jedním potomkem, kořenový uzel má jeden vytvořený slajd, uzly i slajd mají vyplněný text, vše formátováno pomocí manuálních stylů, alespoň dvě třídy stylu pro uzel, linii i slajd

1. Kliknutí pravým tlačítkem myši na kořenový uzel – otevře se editační menu, uzel nemá přiřazenou žádnou třídu.

2. Přiřazení jedné z tříd stylu – změní se formát uzlu tak, aby odpovídal třídě stylu, hodnoty v záložkách pro formát uzlu i formát textu odpovídají hodnotám třídy stylu.

3. Přiřazení jiné třídy stylu – změní se formát uzlu tak, aby odpovídal třídě stylu, hodnoty v záložkách pro formát uzlu i formát textu odpovídají hodnotám třídy stylu.

4. Odebrání třídy stylu – uzel je formátován jednoduchým výchozím způsobem, hodnoty v záložkách pro formát uzlu i formát textu odpovídají výchozímu formátu aplikace.

5. Kliknutí pravým tlačítkem myši na linii mezi uzly – otevře se editační menu, linie nemá přiřazenou žádnou třídu.

6. Zopakování stejného postupu testování jako pro uzel (postupné přiřazení jedné a druhé třídy stylu, odebrání třídy stylu) – výsledky odpovídají očekáváním (viz uzel).

7. Otevření správy interaktivních prvků u kořenového uzlu, kliknutí pravým tlačítkem myši na slajd – otevře se editační menu, slajd nemá přiřazenou žádnou třídu.

8. Zopakování stejného postupu testování jako pro uzel (postupné přiřazení jedné a druhé třídy stylu, odebrání třídy stylu) – výsledky odpovídají očekáváním (viz uzel).

Výsledek: OK

D.5 Prezentační režim

D.5.1 Spuštění a ukončení prezentačního režimu, zobrazení prvků v prezentačním režimu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a alespoň jedním plovoucím uzlem, s několika slajdy u kořenového uzlu a s manuálním formátem u některých prvků, uzly i slajdy mají vyplněný text, alespoň jeden slajd obsahuje i obrázek

1. Kliknutí na tlačítko spuštění prezentačního režimu – spustí se prezentační režim, mapa je vycentrovaná na obrazovce, vzhled uzlů i linií odpovídá nastaveným hodnotám.

2. Najetí myši nad kořenový uzel – po straně uzlu se objeví dvě tlačítka (přiblížení a spuštění prezentace slajdů).

3. Kliknutí na tlačítko pro spuštění prezentace slajdů – otevření slajdů, slajdy je možné procházet pomocí šipek, jejich vzhled odpovídá nastaveným hodnotám.

4. Kliknutí na plochu slajdu – nic se nestane.

5. Kliknutí mimo plochu slajdu – zavření slajdu.

6. Najetí myši nad jiný uzel – po straně uzlu se objeví pouze jedno tlačítko (přiblížení).

7. Tažení kreslící plochy libovolným směrem – plocha se pohybuje, po upuštění zůstane na novém místě.

8. Kliknutí na tlačítko ukončení prezentačního režimu – ukončení prezentačního režimu, mapa je vycentrovaná na obrazovce.

9. Kliknutí na tlačítko spuštění prezentačního režimu – spustí se prezentační režim.

10. Zmáčknutí klávesy `Escape` – ukončení prezentačního režimu.

Výsledek: OK

D.5.2 Pohyb s plochou v prezentačním režimu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a alespoň jedním plovoucím uzlem, se spuštěným prezentačním režimem

1. Tažení kreslící plochy na libovolné místo uvnitř prohlížeče – všechny uzly se posouvají.

2. Tažení uzlu na libovolné místo uvnitř prohlížeče – všechny uzly se posouvají, pohybuje se celá kreslící plocha.

3. Tažení linie na libovolné místo uvnitř prohlížeče – všechny uzly se posouvají, pohybuje se celá kreslící plocha.

4. Tažení kreslící plochy na libovolné místo mimo okno prohlížeče – v momentě, kdy myš opustí okno, se pohyb plochy zastaví, po puštění tlačítka zůstane kreslící plocha v této pozici.

5. Tažení plochy tak, že je vidět její okraj, pokus o posunutí dále stejným směrem – kreslící plocha se už dále neposouvá, zůstává v pozici, kdy je její okraj na kraji okna prohlížeče.

Výsledek: OK

D.5.3 Manuální přibližování a oddalování plochy v prezentačním režimu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a alespoň jedním plovoucím uzlem, se spuštěným prezentačním režimem

1. Přiblížení kreslící plochy pomocí posuvníku – vše se zvětšuje se správnými proporcemi, centrum přiblížení je uprostřed stránky.

2. Oddálení kreslící plochy pomocí posuvníku – vše se zmenšuje se správnými proporcemi, centrum oddálení je uprostřed stránky, v případě, že je okno větší než oddálená kreslící plocha, je plocha uprostřed okna.

3. Přiblížení a oddálení plochy pomocí tlačítek po stranách posuvníku – vše se správně proporčně zvětšuje/zmenšuje.

Výsledek: OK

D.5.4 Vycentrování obsahu na obrazovce v prezentačním režimu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem, a s alespoň jedním plovoucím uzlem, se spuštěným prezentačním režimem, pohled není vycentrovaný a mapa je maximálně přiblížená

1. Kliknutí na tlačítko vycentrování na obrazovce – upraví se poloha kreslící plochy a její přiblížení tak, aby se celá mapa vešla na obrazovku a pokrývala její maximální část.

Výsledek: OK

D.5.5 Automatické přibližování a oddalování plochy v prezentačním režimu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem (alespoň 3 úrovně) a alespoň jedním plovoucím uzlem vzdálenějším od hlavního stromu, se spuštěným prezentačním režimem

1. Najetí myší nad kořenový uzel, zmáčknutí tlačítka pro přiblížení – celý strom pod kořenovým uzlem je přiblížen a vycentrována na obrazovce.

2. Najetí myší nad jednoho z potomků kořenového uzlu, zmáčknutí tlačítka pro přiblížení – větev potomka je přiblížena a vycentrována na obrazovce.

3. Najetí myší nad jednoho z potomků stávajícího přiblíženého uzlu, zmáčknutí tlačítka pro přiblížení – větev potomka je přiblížena a vycentrována na obrazovce.

4. Kliknutí na tlačítko oddálení o jednu úroveň – návrat do předchozí přiblížené pozice (nad potomka kořenového uzlu).

5. Kliknutí na tlačítko oddálení o jednu úroveň – návrat do předchozí přiblížené pozice (nad kořenový uzel).

6. Kliknutí na tlačítko oddálení o jednu úroveň – návrat do pozice s vycentrovanou celou mapou.

7. Najetí myší nad kořenový uzel, zmáčknutí tlačítka pro přiblížení – celá mapa pod kořenovým uzlem je přiblížena a vycentrována na obrazovce.

8. Najetí myší nad jednoho z potomků kořenového uzlu, zmáčknutí tlačítka pro přiblížení – větev potomka je přiblížena a vycentrována na obrazovce.

9. Kliknutí na tlačítko vycentrování na obrazovce – návrat do pozice s vycentrovanou celou mapou.
10. Kliknutí na tlačítko oddálení o jednu úroveň – bez změny pozice.

Výsledek: OK

D.6 Import a export

D.6.1 Export a import tříd

Výchozí stav: aplikace s nově vytvořenými třídami pro styl uzlu, linie a slajdu a s upravenými všemi stávajícími třídami

1. Kliknutí na tlačítko `Export map` v hlavním menu – zobrazí se menu pro export.
2. Uložení souboru s třídami stylu – stáhne a uloží se soubor ve formátu JSON.
3. Kliknutí na tlačítko `Cancel` – zavření menu pro export, hlavní menu zůstává otevřené.
4. Restart aplikace.
5. Vytvoření alespoň pěti nových uzlů, změna velikosti kreslící plochy.
6. Kliknutí tlačítko `Import map` v hlavním menu – zobrazení menu pro import.
7. Nahrání souboru s třídami, potvrzení importu tříd kliknutím na tlačítko `Import` – zavření menu pro import, v panelu tříd se zobrazí nové a upravené třídy, struktura uzlů se nezmění, velikost kreslící plochy také ne, mapa je vycentrovaná na kreslící ploše, žádný prvek není aktivní.
8. Vytvoření dalších pěti uzlů – vše funguje, je možné použít importované třídy pro formátování prvků.

Výsledek: OK

D.6.2 Export a import celé mapy

Výchozí stav: aplikace s nově vytvořenými třídami pro styl uzlu, linie a slajdu a s upravenými všemi stávajícími třídami, dále s vytvořeným stromem pod kořenovým uzlem a alespoň jedním plovoucím uzlem, s několika slajdy u různých uzlů, s manuálním formátem u některých prvků i formátem pomocí nových tříd stylu, se změněnou velikostí kreslícího plátna a s vyplněným obsahem uzlů i slajdů

1. Kliknutí na tlačítko `Export map` v hlavním menu – zobrazí se menu pro export.
2. Uložení souboru s celou mapou – stáhne a uloží se soubor ve formátu JSON.
3. Kliknutí na tlačítko `Cancel` – zavření menu pro export, hlavní menu zůstává otevřené.
4. Restart aplikace.
5. Vytvoření alespoň pěti nových uzlů a několika tříd.
6. Kliknutí tlačítko `Import map` v hlavním menu – zobrazení menu pro import.
7. Nahrání souboru s mapou, potvrzení importu mapy kliknutím na tlačítko `Import` – zavření menu pro import, v panelu tříd se zobrazí nové a upravené třídy, na kreslící ploše se objeví struktura uzlů stejná jako před exportem, uzly mají správně přiřazené slajdy, kreslící plocha má správnou velikost, mapa je vycentrovaná na kreslící ploše, žádný prvek není aktivní.
8. Vytvoření dalších pěti uzlů – vše funguje, je možné použít importované třídy pro formátování prvků.

Výsledek: OK

D.6.3 Používání textových polí při importu a exportu

Výchozí stav: aplikace s vytvořeným stromem pod kořenovým uzlem a několika novými třídami

1. Kliknutí na tlačítko `Export map` v hlavním menu – zobrazí se menu pro export.
2. Uložení textu v textovém poli s exportem celé mapy.
3. Kliknutí na tlačítko `Cancel` – zavření menu pro export, hlavní menu zůstává otevřené.
4. Restart aplikace.
5. Kliknutí tlačítko `Import map` v hlavním menu – zobrazení menu pro import.
6. Pokus o import prázdného textového pole – nelze, zobrazí se chybová hláška.

7. Napsání textu se špatnou syntaxí JSONu do textového pole, pokus o import – nelze, zobrazí se chybová hláška.
8. Napsání textu reprezentujícího validní JSON, ale s nevyhovující strukturou, pokus o import – nelze, zobrazí se chybová hláška.
9. Vložení textu popisujícího mapu do textového pole, pokus o import – import proběhne v pořádku.

Výsledek: OK

Dodatek E

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF