FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Webová aplikace pro tvorbu grafického prototypu ve hře Agilně
Student:	Bc. Marek Mouček
Vedoucí:	Ing. Petra Pavlíčková, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce zimního semestru 2022/2023

Pokyny pro vypracování

Cílem diplomové práce je návrh a implementace webové aplikace pro modelování grafického prototypu ze hry Agilně. Cílem této aplikace je usnadnění tvorby prototypu a tím i zdokonalení samotné hry.

1. Seznamte se s hrou Agilně ve frameworku SCRUM.
2. Připravte návrh webové aplikace pro grafickou tvorbu prototypu včetně návrhu UI, formulujte požadavky a vyberte vhodné technologie.
3. Implementujte webovou aplikaci.
4. Proveďte testování aplikace.
5. Zhodnoťte dané řešení a navrhněte možná budoucí vylepšení.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 22. června 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Webová aplikace pro tvorbu grafického prototypu ve hře Agilně

Bc. Marek Mouček

Katedra softwarového inženýrství
Vedoucí práce: Ing. Petra Pavlíčková, Ph.D.

4. ledna 2022

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. ledna 2022 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Mouček, Marek. *Webová aplikace pro tvorbu grafického prototypu ve hře Agilně.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato práce se zabývá vytvořením webového grafického editoru určeného pro podporu průběhu hry Agilně. V práci jsou využity a popsány nejnovější technologie a standardy, včetně knihoven použitých k její tvorbě. Primárně práce využívá knihovny React a Konva. Jsou zde také porovnány výhody a nevýhody jednotlivých přístupů a celé řešení je nakonec řádně otestováno pomocí automatického i manuálního testování.

Klíčová slova webová aplikace, grafický editor, hra Agilně, React, Redux, Konva, Service Worker

Abstract

This thesis deals with the creation of a web-based graphical editor designed to support the Agile game flow. It uses and describes the latest technologies and standards, including the libraries used to create it. The implementation primarly uses the React and Konva libraries. The advantages and disadvantages of each approach are compared and the whole solution is properly tested through automated and usability testing.

Keywords web application, graphics editor, hra Agilně, React, Redux, Konva, Service Worker

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Hra Agilně byla vytvořena pro výzkumné účely v oblasti problematiky agilního přístupu k řízení projektů. Hra Agilně má za cíl rozšířit znalosti studentů převážně v agilním frameworku SCRUM a vyhodnotit na jakou roli se daný student nejlépe hodí. Hraje se v rámci jednoho cvičení a jejím výsledkem by měly být jednoduché grafické návrhy webové stránky, loga, vizitek a baneru.

Aktuálně se ve hře k vytvoření těchto grafických návrhů používají různé grafické editory, které je nutné nainstalovat, nebo jinak nastavit a navíc jsou pro hru zbytečně složité. Tento proces je pro účely krátké hry příliš komplikovaný.

Cílém této práce je usnadnění a zefektivnění tvorby tohoto grafického návrhu pomocí implementace aplikace pro tvorbu grafického prototypu. V první části je tedy zanalyzován průběh hry Agilně a jsou popsány použité technologie a knihovny. Dále jsou zformulovány funkční a nefunkční požadavky, případy užití a na jejich základě je navržena výsledná aplikace. Dle tohoto návrhu je aplikace implementována a řádně otestována pomocí automatických testovacích nástrojů. Následně je provedeno uživatelské testování aplikace s cílem budoucího vylepšení použitelnosti.

Kapitola 1

Rešerše

Následující kapitola tvoří souhrn základních informací, které byly použity při realizaci této diplomové práce. Kapitola popisuje hru Agilně, zaměřuje se na výběr platformy, technologií a standardů, které budou dále využívány. Popisuje fungování frameworku React a dalších technologií potřebných k vytvoření požadované funkcionality.

1.1 Hra Agilně

Hru Agilně velice dobře popisuje následující úryvek z úvodu její specifikace:

„Tato agilní hra byla vytvořena v rámci grantu IGA pro výzkumné účely v oblasti problematiky agilního přístupu k řízení projektů. Hra byla vytvořena převážně ve frameworku SCRUM, který patří dnes mezi nejrozšířenější pro řešení projektů v sektoru informačních technologií. Tato hra by měla sloužit k ověření, zda uživatel pochopil problematiku agilního řízení (především IT) projektů. Dále by tato hra měla být schopna vyhodnotit, na jakou týmovou roli v agilním týmu se hodí daný člověk dle jeho specifických povahových vlastností a chování.“

Hra Agilně tedy simuluje průběh běžného sprintu v softwarové firmě, kde existuje vztah mezi 2 subjekty (zákazník a dodavatel). V rámci hry se vytváří projekt, který má za cíl poskytnout levnější způsob online školení pro zaměstnance různých firem. Výsledný produkt však v tomto případě není software, nýbrž grafický prototyp, protože jinak by nebylo možné vše stihnout v čase jednoho cvičení (90 minut).

Níže je stručně popsáno zadání projektu. Toto zadání se podobně jako u reálného projektu v průběhu času hry mění, což je simulováno pomocí změnových požadavků od zákazníka. Výsledná dodávka by měla obsahovat:

- Webovou stránku s následujícími částmi:
  - úvodní stránka,

  - stránka s kurzy,
  - stránka s přehledem cen,
  - stránka s kontakty a lokací firmy,
  - možnost přihlášení, odhlášení a registrace.
- Mobilní aplikaci s následujícími částmi:
  - přihlašovací stránka,
  - domovská stránka,
  - knihovna kurzů uživatele,
  - oblíbené kurzy uživatele,
  - nastavení aplikace.
- Jednoduché logo, které by mělo být zajímavé, ale nepříliš složité. Toto logo bude použito v ostatních částech projektu.
- Vizitky s logem a motem firmy a údaji jednotlivých zaměstnanců. Vizitky nejsou součástí původního zadání, ale jsou změnovým požadavkem.
- Banner obdélníkového rozměru s nápisem „WORKSHOP“, který bude mít jednotlivá písmenka na červených visačkách a bude jej možné vložit na šířku na webovou stránku.

1.1.1 Realizace projektu

Projekt by měl být realizován dle agilní metodiky SCRUM. Jeho hlavní částí bude produktový backlog, který obsahuje dílčí celky zvané *Epic*, které se dále dělí na menší části zvané *User stories*. User stories se dále dělí na jednotlivé úkoly, které lze zadávat jednotlivým členům týmu.

User stories mají tyto dvě zásadní hodnoty, které je nutné zvážit při plánování sprintů:

- Business value, která představuje hodnotu pro podnikání zákazníka, který ji také stanoví.
- Prioritu, kterou stanoví Product Owner.

Součástí projektu je dále analýza rizik, která je předem připravená a jejím cílem je odhalit nežádoucí stavy či nebezpečí, která mohou při vývoji projektu nastat. Každý tým dostává na začátku hry omezený rozpočet a musí se rozhodnout, která rizika eliminuje, a která bude naopak ignorovat. Na konci hry se náhodně vytahují události představující jednotlivá rizika a je vyhodnoceno, zda se týmy na tyto rizika připravily nebo ne.

1.1.2 Herní tým

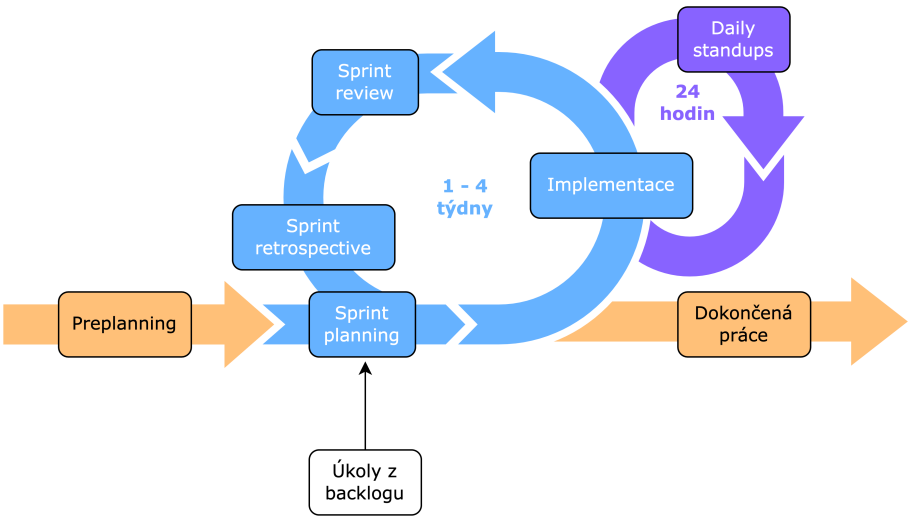
Herní tým reprezentuje dodavatelskou společnost projektu, kterou tvoří studenti dodávající design, kostru webu, mobilní aplikaci, vizitky, logo a banner. Tento tým se skládá z následujících třech hlavních rolí:

- Product Owner – je zoodpovědný za komunikaci se zákazníkem, dodání projektu a produktový backlog.
- Scrum Master – stará se o všechny činnosti a události v rámci agilní metodiky.
- Vývojový tým – pracuje na úkolech v rámci sprintů a dále se skládá z podrolí specifických dle projektu. V rámci hry Agilně definujeme tři až čtyři podrole: UX designer, senior designer, junior designer a nepovinný druhý junior designer.

1.1.3 SCRUM události

Scrum události jsou nezbytné pro zajištění hladké realizace projektu. Události jsou následující:

- Preplanning – událost zaměřená na prioritizaci prvotních User stories, se kterými by se měli všichni členové týmu předem seznámit.
- Sprint planning – důležitá událost před začátkem každého sprintu, při které se setká celý scrum tým a plánují se úkoly, které budou dodány v rámci příštího sprintu. User stories by se měly zaplánovávat do sprintu dle jejich priority. Planning je ukončen, pokud se všichni členové schodnou na tom, že je sprint zcela zaplněn.
- Sprint – ve sprintech se realizují jendotlivé User stories, kterým se tak v průběhu sprintu mění stav. Cílem sprintu je, aby na jeho konci byly všechny úkoly úspěšně odevzdány.
- Sprint review – událost na konci sprintu během které tým diskutuje, které úkoly byly dodány, a které je naopak nutné přesunout do dalších sprintů.
- Sprint retrospective – událost během které probíhá diskuze na téma co v daném sprint fungovalo dobře, a co by se naopak mělo zlepšit. Je to jedna z klíčových událostí, protože může zajistit postupné zvyšování kvality sprintů, a také zvyšování spokojenosti členů týmu.
- Daily standups – každodenní událost v průběhu sprintu během které každý člen v pár větách shrne na čem doposud pracoval, na čem bude pracovat a jaké problémy ho blokují.



Obrázek 1.1: SCRUM diagram

1.1.4 SCRUM v praxi

Protože se hra Agilně hraje v rámci 90 minutového cvičení, tak musely být z časových a organizačních důvodů upraveny některé aspekty reálného průběhu SCRUMu. Zásadní změnou je zkrácení délky událostí a sprintu. V praxi sprint obvykle trvá 1 až 4 týdny a v rámci něj jsou provozovány každodenní daily standups. Diagram reálného SCRUMu s posloupností jednotlivých událostí je znázorněn na obrázku 1.1.

Události SCRUMu obvykle trvají 1 až 3 hodiny (kromě daily standups, které jsou omezeny na 15 minut). Ve hře Agilně jsou všechny události zkráceny na 2 až 8 minut a jsou vynechány události sprint retrospective a daily standups, které nejsou pro průběh hry zásadní.

V praxi ve SCRUMu navíc existuje ještě nepravidelná ceremonie zvaná backlog grooming, kterou provádí Product Owner. Cílem této ceremonie je udržovat pořádek v produktovém backlogu. To lze zajistit pomocí:

- Doladění zadání položek nebo jejich rozdělení na více menších položek.
- Odstranění nerelevatních položek.
- Přidání nových položek.
- Prioritizace položek.

1.2 Výběr platformy pro aplikaci

Pro práci s grafickou aplikací je nejvhodnější stolní počítač nebo notebook. Předpokládáme, že studenti budou mít během cvičení tato zařízení k dispozici. Malá mobilní zařízení tedy nejsou brány v úvahu.

Při zacílení na osobní počítače je vhodné, aby aplikace podporovala alespoň nejnovější verze systémů MacOS, Windows a Linux. Linux má sice obecně menší zastoupení, ale na technických školách, na kterých se hraje tato hra, má obvykle o něco vyšší zastoupení než jinde.

Pokud nebudou brány v úvahu různé hybridní a multiplatformní typy aplikací, tak je rozhodováno primárně mezi nativní a webovou aplikací.

Výhody webové aplikace jsou následující: [1]

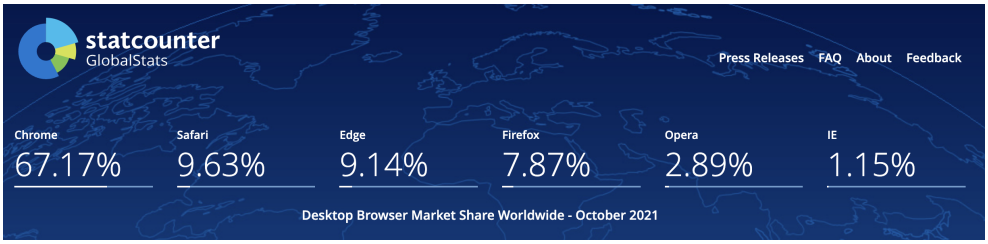
- Podpora více prohlížečů je snazší než podpora více operačních systémů, protože v jednotlivých prohlížečích jsou dnes již minimální rozdíly.
- Webové aplikace jsou přístupné z jakékoli lokace, která má přístup k internetu.
- Při využití vhodných technologií a podporovaného prohlížeče je webovou aplikaci možné nainstalovat do počítače a také používat offline, zároveň je však výhodou, že toto pro funkčnost aplikace není nutné a je možné ji používat kompletně online.
- Vývoj je snazší, protože není nutné udržovat více zdrojových kódů k jednotlivým platformám.

Výhody nativní aplikace jsou následující: [1]

- Rychlejší běh na cílové platformě a snažší optimalizace.
- Vyšší bezpečnost oproti webovým aplikacím.
- Snažší přístup k funkcím systémů (jako například k systému souborů).
- Instalace do zařízení a funkce offline bez nutnosti používat podporovaný prohlížeč.

Na základě tohoto porovnání byla pro tuto diplomovou práci vybrána webová aplikace, protože jednoduchá podpora mnoha zařízení a jednoduchost prvotního používání jsou klíčová kritéria pro její úspěch.

Dále bylo nutné vybrat, které prohlížeče bude aplikace podporovat. Zde bylo stanoveno kritérium, že prohlížeč musí používat více než 3% uživatelů. Pro porovnání prohlížečů byla vybrána stránka StatCounter [2], kde byla vyfiltrována data používání prohlížečů pro osobní počítače z října roku 2021. Různé statistické servery mohou mít mírně odlišná data, nicméně pro účely této práce je výše zmíněný zdroj dostačující.



Obrázek 1.2: Statistika používání prohlížečů

Výsledek této statistiky je viditelný na obrázku č. 1.2 a je z něj zřejmé, že by aplikace měla podporovat minimálně prohlížeče Chrome, Safari, Edge a Firefox.

1.3 React

React je JavaScriptová open-source knihovna, která je zaměřená na tvorbu uživatelského rozhraní a byla vytvořena společností Facebook. V současné době patří mezi jednu z nejoblíbenějších JavaScriptových knihoven pro tvorbu uživatelského rozhraní. [3] React má především tu výhodu, že se dá velmi snadno použít, protože není nutné používat velké množství boilerplate kódu. [4]

Základním pilířem Reactu je takzvaný jednosměrný datový tok, což znamená, že data se mohou zasílat pouze jedním směrem, a to od rodičovské komponenty do dceřiné komponenty. Data v dceřiné komponentě jsou určena pouze pro čtení. Pokud je potřeba, aby dceřiná komponenta změnila data, může rodičovská komponenta předat dceřiné komponentě pro tyto účely funkci určenou pro aktualizaci dat, čímž není porušen princip jednosměrného datového toku. [4]

Zásadní výhodou tohoto přístupu je zapouzdření dat a princip neměnnosti, který usnadňuje debugging a hledání chyb v aplikaci. Díky tomu je také jednodušší optimalizovat aplikaci, protože jednosměrný datový tok je snažší na pochopení a ladění kódu. [4]

1.3.1 Komponenty

Základní stavební prvek Reactu jsou komponenty. Každá webová aplikace napsaná v Reactu se skládá z množství komponent, které do sebe mohou být vnořené. Tyto komponenty lze v kódu použít velmi podobně jako HTML tagy, respektive se dá říct, že jsou to nové XML tagy. Tento způsob používání je pouze syntaktický cukr, protože na pozadí jsou komponenty vytvářeny jako běžné JavaScriptové funkce nebo třídy.

Komponenty přijímají takzvané *props*, které představují ve standardním zápisu atributy XML tagů, ale na pozadí jsou to běžné argumenty funkcí. Props mohou mít výchozí hodnotu a dá se s nimi pracovat jako s běžnými argumenty.

Komponenty jsou uzavřené a obvykle nezávislé a měly by to být takzvané čisté funkce bez postranních účinků, což znamená, že pro stejný vstup vracejí vždy stejný výstup, a nesmí změnit žádné globální proměnné.

1.3.2 Virtuální DOM

Důležitým konceptem v Reactu je takzvaný virtuální DOM, který představuje ideální reprezentaci UI uloženou v paměti, která je následně synchronizována s reálným DOM pomocí knihovny *ReactDOM*. [4]

Tento přístup je základním stavebním kamenem v Reactu díky kterému stačí, aby vývojář nastavil komponenty do takového stavu, v jakém chce aby byly. Zároveň může vývojář nad těmito komponentami naslouchat různým událostem. To vytváří jistou vrstvu abstrakce nad reálným DOM, a díky tomu odpadá nutnost ho přímo aktualizovat nebo v něm manuálně měnit atributy. Tato abstrakce navíc umožňuje, aby React optimalizoval způsob vykreslování reálného DOM a výsledná aplikace díky tomu bude rychlejší než při neopatrné manuální aktualizaci. [4]

V komponentě se při každém volání funkce `render()` vytváří nový strom elementů. V případě, že se změní stav nebo atributy, tak se vytvoří nový strom elementů. React nad tímto vytvořeným stromem provádí heuristiku, která vyhodnocuje jeho změny na základě dvou předpokladů: [4]

1. Dva elementy rozdílného typu budou produkovat různé stromy.

2. Vývojář může Reactu pomoci rozlišit, které dceřiné elementy se změnily pomocí `key` atributu.

1.3.2.1 Porovnání stromů

O porovnání stromů se v Reactu stará takzvaný *Diffing* algoritmus. Jeho přesné chování je závislé na typu výchozího elementu: [4]

- Elementy rozdílného typu – při změně typu výchozího elementu, jako například z `<a>` na `<div>`, React překreslí celý strom tohoto elementu (všechny elementy jsou zničeny a jsou vytvořeny nové elementy).

- Elementy stejného typu – React kontroluje atributy elementu a pokud se nějaký změní, tak se v DOM uzlu změní pouze tento konkrétní atribut. Stejně tak se při změně atributů kaskádových stylů změní pouze modifikované atributy.

```
function HelloComponent({name}) {
    return <p>Hello {name}!</p>;
}
```

Zdrojový kód 1.1: Komponenta jako funkce

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

Zdrojový kód 1.2: Komponenta jako třída

- Rekurze na dceřiných elementech – React iteruje nad všemi dceřinými elementy DOM uzlu (jako například `<ul>`) a provede změnu, pokud najde rozdíly. Například pokud se přidá nový element na začátek seznamu, tak se musí překreslit i celý zbytek seznamu. Toto nevyžádané chování lze eliminovat pomocí atributu `key`, který je doporučeno přidat všem elementům ze seznamu, protože dle něj React dokáže určit, co se přesně v seznamu změnilo. Pro tyto účely lze použít například id elementu z databáze, nebo je možné elementům generovat vlastní id.

1.3.2.2 Komponenty jako třídy nebo funkce

React umožňuje vytvářet komponenty jako třídy nebo jako funkce. Obě varianty jsou ekvivalentní jak z hlediska možností využití, tak z hlediska funkčnosti, nicméně ve funkcích lze používat takzvané *Hooks*, které usnadňují práci s Reactem. Tato kapitola vychází ze zdrojů [5], [6] a [7].

Funkční komponenty mají obvykle nižší počet řádků kódu a díky tomu jsou lépe pochopitelné a je snažší je ladit. Při použití React Hooks mají funkční komponenty dostupnou veškerou funkčnost tříd, což nebylo před zavedením Hooks možné. V současné době je tedy spíše doporučeno používat komponenty jako funkce.

Ukázka jednoduché komponenty jako funkce je viditelná ve zdrojovém kódu č. 1.1, ukázka komponenty jako třídy je viditelná ve zdrojovém kódu č. 1.2. Oba typy komponent lze použít v rámci ostatních komponent stejným způsobem, ukázka jejich použití ve funkční komponentě `App` je viditelná ve zdrojovém kódu č. 1.3.

```
funcion App() {
    return <HelloComponent name="World"/>
}
```

Zdrojový kód 1.3: Ukázka použití komponent

1.3.3 Hooks

React Hooks umožňují rozšířit funkce a komponenty o další funkcionalitu (například používat redux stav), bez nutnosti používat třídy. Jedny z nejdůležitějších Hooks v reactu jsou `useEffect` a `useState`. Vývojář může vytvářet nové Hooks, které používají jiné Hooks. Hooks mohou vracet libovolnou hodnotu, v tom nejsou odlišné od běžných funkcí. [4]

Pro jejich správné fungování je nutné dodržet následující pravidla: [4]

- Hooks by se měly volat vždy na začátku funkcí, ale neměly by se volat v podmínkách, smyčkách a vnořených funkcích.
- Hooks by se měly volat pouze z React komponent nebo z jiných Hooks, ale neměly by se volat z běžných JavaScriptových funkcí.

1.3.4 Context

Context je způsob předávání dat mezi rodičovskými a dceřinými komponentami bez nutnosti je předávat mezi props. Používá se typicky pokud jsou data potřebná hlouběji ve stromu komponent. Context funguje na principu publisher-subscriber. To znamená, že komponenta vytvoří provider komponentu, kde předává požadavanou hodnotu a komponenty níže ve stromu komponent se mohou přihlásit k odběru této hodnoty, což je možné buď pomocí speciální komponenty, nebo pomocí React Hooku `useContext`. [4]

Context lze použít například pro udržení přihlášeného uživatele nebo jazyka stránky. Výhodou je, že do Contextu lze ukládat i referenci, kterou není vhodné ukládat do stavu aplikace, protože není serializovatelná. Použití Contextu ve vhodných případech tedy značně zjednodušuje implementaci. [4]

1.4 Redux

Pozornému čtenáři je možná již zřejmá hlavní chybějící funkčnost v Reactu pro použití ve složitějších aplikacích, kterou je absence globálního stavu. Komponenty jsou čisté funkce, pokud tedy chceme předávat informace, tak je musíme posílat v props. V případě, že je komponent více, jsou do sebe vzájemně vnořené a navíc koncepčně existuje „globální“ stav aplikace, tak bude tento způsob předávání informací velmi neefektivní. [8]

```
// vytvoření akce
export const addPage = createAction('pages/add');

// nastavení výchozího stavu
const initialState = {
    byId: {},
    allIds: [],
    selectedPageId: null,
};

// vytvoření Reduceru
const pageReducer = createReducer(initialState, (builder) => {
    // zpracování akce addPage
    builder.addCase(addPage, (state, action) => {
        const payload = action.payload;
        // vytvoření a přidání nové stránky ze zaslasného objektu
        state.byId[payload.id] = payload;
        state.allIds.push(payload.id);
    });
});
```

Zdrojový kód 1.4: Ukázka použití Reduxu

Knihovna Redux slouží ke správě globálního stavu aplikace. Jednotlivé komponenty mohou k tomuto stavu přistupovat a upravovat jej pomocí takzvaných akcí, které lze definovat u jednotlivých stavů. Samotné akce jsou čisté funkce a neprovádí žádné vedlejší operace. [8]

Aplikace používající Redux jsou díky výše napsaným důvodům flexibilní a snadno laditelné, protože je možné se přesouvat do starých stavů pomocí reverzibilnosti akcí. To usnadňuje například implementaci funkce vrátit zpět. [8]

Vývojářům je pro usnadnění vývoje Redux aplikací a redukci repetitivních částí kódu doporučeno používat *Redux toolkit*, který byl vytvořen na základě doporučených pravidel používání Reduxu. Toolkit například standardizuje vytváření a volání akcí a přidává builder pro vytváření reducerů. [8]

Ukázka vytvoření akce a reduceru, pomocí Redux toolkitu je viditelná ve zdrojové kódu č. 1.4.

1.4.1 Akce

Akce jsou jednoduché JavaScriptové objekty, které definují svůj typ a mohou obsahovat *payload*, který podrobněji popisuje, co se v dané akci událo. Akce

si je možné představit jako události v aplikaci. Příkladem takové akce může být vytvoření nového objektu na plátně nebo výběr elementu. [8]

Tyto akce lze následně vyvolat z React komponentů nebo z React Hooks, čímž se modifikuje globální Redux stav aplikace. Akce jsou odbaveny pomocí takzvaných *Reducers*.

1.4.2 Reducers

Reducers jsou čisté funkce, které mají jako vstup aktuální Redux stav aplikace a akci. Vracejí nový stav vycházející z těchto argumentů. Tyto funkce by neměly editovat své argumenty nebo produkovat jiné vedlejší efekty, měly by pouze vracet nový upravený stav. [8]

1.4.3 Selectors

Selectors jsou funkce používané pro vyhledání dat v aplikačním stavu. Selectors mají jako argument Redux stav a vracejí zpět data vycházející z tohoto stavu. V selectoru lze provádět operace typu filtrace, hledání a řazení. Lze je pak použít ve více komponentách. [8]

1.5 Konva

Konva je JavaScriptová knihovna, která rozšiřuje HTML5 canvas o další funkcionalitu a usnadňuje jeho použití. Konva představuje abstrakci nad canvasem a její knihovna pro React přidává nové komponenty využívající tuto funkcionalitu. [9]

Knihovna Konva obohacuje React o následující prvky a funkce: [9]

- Přidává React komponenty pro základní tvary, které je možné na canvas vykreslovat jako je text, hvězda, obrázek, obdélník, linka a další. Tyto základní tvary mají množství nastavitelných parametrů jako je pozice, rotace, barva, tloušťka obrysu tvaru a další.
- Přidává komponenty a vlastnosti pro rozšířenou práci s canvasem jako jsou vrstvy, skupiny, řazení, možnost přibližování canvasu a další.
- S komponentami lze dále pracovat a přidávat jim různé animace, přechody, filtry nebo kešování.
- Rozšiřuje tyto komponenty s grafickými elementy o nové události jako je například klik myší.
- Přidává komponentům drag and drop funkcionalitu s událostmi, které lze zpracovat pro pokročilejší použití jako je omezení hranic pohybu nebo zpracování nového stavu komponenty.

- Přidává různé způsoby pro vyhledávání objektů na plátně jako je nalezení dle id, typu, nebo názvu objektu.

Nevýhoda této knihovny spočívá především v nutnosti při určitých operacích volat referenci na původně vytvořený hlavní objekt zvaný *Stage*. Dopady této nevýhody lze zmírnit pomocí vytvoření nového Contextu pro React, který bude tuto referenci udržovat. Díky tomuto Contextu nebude nutné předávat referenci v atributech komponentů a navíc ji bude možné snadno použít v React Hooks.

Ukázka použití knihovny Konva v kombinaci s Reactem je viditelná ve zdrojovém kódu č. 1.5.

1.5.1 Výkonnost knihovny

Další nevýhodou knihovny Konva je nižší výkonnost oproti HTML5 canvasu, což je cena za vyšší míru abstrakce. Konva nabízí tipy pro zlepšení výkonnosti, které vychází především z následujících pravidel: [9]

1. Snížení výpočtů na nezbytné minimum – každý výpočet zabere určitou dobu, je tedy nutné sledovat délku běhu JavaScriptového kódu, knihovny Konva a dalších vrstev, jako je například React. Účínky těchto vrstev se kumulují a mnoho operací může způsobit špatnou odezvu canvasu.

2. Co nejmenší počet volání vykreslení Konvy – každé vykreslení způsobuje zpomalení canvasu. Vykreslení se dělí na dvě kategorie – za prvé jsou to výpočty okolo vykreslení, které byly popsány v bodu 1, za druhé je to vykreslování objektů z paměti na obrazovku. Toto vykreslování je nutné co nejvíce minimalizovat.

Vzhledem k tomu, že je použita knihovna Konva pro React, tak je nutné také minimalizovat aktualizace React komponentů, protože ty způsobují volání vykreslení Konvy.

```
const KonvaShowcase = () => {
    return (
        // vytvoříme plátno o velikost 300x300 pixelů
        <Stage width={300} height={300}>
            // na plátně vytvoříme vrstvu
            <Layer>
                // do vrstvy umístíme obdelník na pozici
                // x = 10, y = 10 s velikostí 25 pixelů
                // a zeleným pozadím
                <Rect
                    x={10}
                    y={10}
                    width={25}
                    height={25}
                    fill="green"
                />
            </Layer>
        </Stage>
    )
}
```

Zdrojový kód 1.5: Ukázka použití knihovny Konva

Kapitola 2

Analýza a návrh

Tato kapitola se zabývá analýzou a návrhem aplikace, což je proces, který se zaměřuje na specifikaci požadavků a případů použití aplikace, čímž následně zjednodušuje její implementaci. Důkladné provedení této části šetří vývojový čas a také usnadňuje odhalení chyb uživatelského rozhraní před jeho vývojem.

2.1 Analýza požadavků

Tato část se zaměřuje na analýzu požadavků, které byly specifikovány ve spolupráci s týmem hry Agilně. Požadavky jsou obvykle rozděleny na tyto dvě kategorie: [10]

- Funkční požadavky, které se zaměřují na specifikaci funkčnosti aplikace. To může být například kalkulace, manipulace s daty, uživatelské interakce, nebo jakákoli jiná funkčnost. Tyto typy požadavků pomáhají specifikovat cílovou funkčnost výsledného systému.
- Nefunkční požadavky, které se zaměřují na specifikaci hranic kvality aplikace a řeší pojmy jako jsou bezpečnost, udržitelnost, výkonnost, responzivita a další. Tyto typy požadavků tvoří základ nutný pro vytvoření aplikace, která bude dobře použitelná i v reálných podmínkách.

2.1.1 Funkční požadavky

Při návrhu této aplikace byly identifikovány následující funkční požadavky:

1. Nastavení velikosti dokumentu a stránek – aplikace umožňuje nastavit velikost aktivního dokumentu a měla by obsahovat přednastavené varianty pro snadné použití ve hře Agilně.
2. Uložení a načtení dokumentu – aplikace podporuje uložení aktuálního dokumentu a načtení dříve rozpracovaného dokumentu ze souborového systému.

3. Export do obrázku – aplikace umožňuje exportovat aktuální stránku do obrázkového formátu.

4. Interaktivní prohlídka dokumentu – objektům na plátně lze nastavit odkaz na jinou stránku. Celý projekt lze následně spustit v režimu, ve kterém je možné projekt procházet podobně jako například reálnou webovou aplikaci s odkazy.

5. Podpora historie – aplikace bude udržovat dočasnou historii posledních provedených akcí a uživatel se bude moci pomocí tlačítek „vrátit zpět“ nebo „opakovat“ pohybovat v historii úprav dokumentu, aby bylo možné vrátit zpět nechtěné akce.

6. Vytváření stránek – každý dokument může mít více pojmenovaných stránek, ty lze používat při vytváření prototypů webů nebo aplikací a lze je mezi sebou prolinkovávat.

7. Předdefinované objekty – aplikace obsahuje několik předdefinovaných objektů, které lze použít pro UX návrh. Příklady takových objektů: blok textu, tlačítko, nadpis, odkaz a další.

8. Přidávání ikon – aplikace bude obsahovat objekt pro ikony, který bude mít alespoň 30 předem definovaných ikon, které lze snadno použít při návrhu.

9. Přidávání vlastní grafiky – na kreslící plátno bude možné pomocí grafického objektu přidávat také vlastní grafiku ve formě obrázku nebo vektorů.

10. Vytváření skupin – elementy a skupiny lze slučovat do dalších skupin, které lze snadno kopírovat na jiná místa v dokumentu.

11. Vlastnosti objektů na pláně – jednotlivým objektům lze změnit základní vlastnosti jako jsou barva, obrys nebo text. Další vlastnosti závisí na typu objektu.

12. Transformace objektů na plátně – objekty lze na plátně transformovat a přesouvat pomocí myši bez nutnosti manuálně upravovat tyto parametry v číselné podobě.

2.1.2 Nefunkční požadavky

Při návrhu této aplikace byly identifikovány následující nefunkční požadavky:

1. Podpora prohlížečů – aplikace bude podporovat nejnovější verze prohlížečů Chrome, Firefox, Safari a Edge.

2. Fungování bez připojení k internetu – aplikace bude fungovat i v případě absence internetového připojení, pokud ji uživatel již v minulosti alespoň jednou použil.

3. Progresivní aplikace – aplikace bude progresivní, což znamená, že si ji uživatel bude moci v podporovaných systémech a prohlížečích přidat na plochu a spouštět podobně jako nativní aplikace.

4. Aplikace se načte do 2 sekund na internetovém připojení s downloadem minimálně 20 Mbit/s.

5. Rozšiřitelnost – aplikace bude implementovaná takovým modulárním způsobem, aby ji bylo možné co nejsnadněji rozšiřovat o další funkcionalitu.

6. Responzivita – aplikace je určená pro počítače a notebook s klávesnicí a myší (nebo trackpadem) s minimálním rozlišením 1200 x 700 pixelů. Aplikace nebude podporovat mobilní telefony ani tablety.

7. Jazyk – aplikace bude pouze v českém jazyce.

2.2 Případy užití

Případy užití popisují, jak budou uživatelé provádět konkrétní úkoly v aplikaci. Každý use case představuje sérií jednoduchých kroků, které uživatel provádí s účelem splnění nějakého definovaného cíle. Jejich správná identifikace je klíčová pro zajištění užitečnosti a snadnosti použití, zároveň lze na jejich základně posoudit náročnost vývoje aplikace a vyhodnotit možné problematické části. [11]

Případy užití byly vytvořeny ve spolupráci s týmem hry Agilně, včetně cvičícího, který má zkušenosti s reálným průběhem této hry. Hlavním účelem této aplikace je podpora grafického procesu v této hře.

2.2.1 Seznam účastníků

Účastníci mohou nabývat následujících rolí, které se liší především způsobem jakým uživatel chce používat aplikaci, přičemž role neomezují přístup do jednotlivých sekcí aplikace:

1. editor, který připravuje grafické návrhy,

2. kritik, který obdrží grafické návrhy a hodnotí je.

2.2.2 Seznam případů užití

Při návrhu této aplikace byly identifikovány následující případy užití:

1. Vytvoření nového projektu – uživatel má možnost vytvořit nový projekt a to buď dle předdefinovaných rozměrů (web, vizika, mobilní aplikace, baner nebo logo) a nebo si může zvolit vlastní velikost dokumentu.
2. Načtení ze souboru – uživatel má možnost otevřít soubor s projektem ze svého disku a načíst ho do aplikace.
3. Uložení do souboru – uživatel má možnost aktuální projekt uložit na svůj disk.
4. Export do obrázkového formátu – uživatel může aktuálně otevřenou stránku exportovat do obrázkového formátu ve formátu png.
5. Pohyb v historii – uživatel má možnost vrátit zpět svých x posledních akcí a má možnost obnovit tyto akce, za předpokladu, že se v minulosti vrátil zpět, ale zároveň neudělal žádnou novou změnu.
6. Seznam stránek – uživatel má k dispozici seznam stránek v aktuálním dokumentu a může jednu ze stránek vybrat jako aktuální.
7. Smazání stránky – uživatel může smazat stránku ze seznamu.
8. Přidání stránky – uživatel může přidat novou stránku do projektu.
9. Editace stránky – uživatel může editovat vlastnosti vybrané stránky, jako je její výška nebo název.
10. Přidání objektu na plátno – uživatel si může vybrat objekt a přidat jej na kreslící plátno.
11. Selekce objektů – uživatel může na plátně vybrat jeden nebo více objektů, se kterými lze pak pracovat.
12. Editace objektů – uživatel může měnit vlastnosti objektů, které jsou různé dle jeho typu. Typické vlastnosti objektů jsou například barva pozadí, šířka a barva okrajů, velikost písma nebo řádkování u textových objektů.
13. Změna textu u objektů textového typu – uživatel může změnit text u objektů textového typu pomocí editace na kreslícím plátně.
14. Odkaz objektu na stránku – uživatel může nastavit jednotlivé objekty tak, aby odkazovaly na jiné stránky v projektu. Tyto odkazy se pak využijí v interaktivní prohlídce.

15. Změna velikosti, přesun a rotace objektu – uživatel může na plátně měnit základní vlastnosti objektu, jako je jeho velikost, pozice a rotace. Všechny tyto vlastnosti by měly být editovatelné i přímo na plátně.

16. Smazání objektu – uživatel může smazat jeden nebo více vybraných objektů.

17. Kopírování objektu – uživatel může jeden nebo více vybraných objektů zkopírovat na jinou stránku, nebo je duplikovat na aktuální stránce.

18. Přesun v hloubce plátna – uživatel může objekty přesouvat směrem do popředí nebo do pozadí.

19. Vytváření skupin – uživatel může seskupit několik objektů tak, aby se s nimi pracovalo podobně jako s jediným objektem.

20. Rozdělení skupin – uživatel může aktuální skupinu rozdělit na její jednotlivé objekty.

21. Dočasné otevření skupin – uživatel může aktuální skupinu dočasně otevřít, což mu umožní upravit individuální objekty.

22. Změna přiblížení plátna – uživatel může změnit přiblížení plátna.

23. Přichycení vůči ostatním objektům – uživatel může zapnout přichycení vůči ostatním objektům při posunu elementu, tak aby se objekt přichytil k okrajům a středu tohoto objektu.

24. Interaktivní prohlídka – uživatel může zapnout interaktivní prohlídku dokumentu, ve které budou aktivní odkazy objektů a dokument se tak bude chovat podobně jako reálná webová stránka.

2.3 Požadavky na dostupné objekty

Klíčovou částí grafického editoru jsou všechny jeho objekty, které lze vytvářet a následně editovat na kreslícím plátně. Jejich popisu je věnována tato sekce.

Aplikace by měla obsahovat alespoň následující typy objektů:

- obdélník,
- elipsa,
- text,
- obrázek,
- ikona (alespoň 30 různých typů),
- nadpis (alespoň 2 typy nadpisů),

- odstavec textu,
- tlačítko,
- input,
- odkaz.

Tento seznam objektů zajistí možnost vytvářet jednoduché UX prototypy a také některé další typy grafiky.

2.4 Návrh uživatelského rozhraní

Na základě procesů, požadavků a případů užití byl navržen vzhled uživatelského rozhraní. Při návrhu nebylo postupováno vytvořením grafiky v grafickém editoru a následně jeho implementací, ale aplikace byla navrhována přímo v prohlížeči v Reactu jako nefunkční prototyp.

Tento způsob návrhu považuji za optimální u tohoto typu funkčních aplikací. Níže jsou popsány jeho výhody a nevýhody a je vysvětlen důvod výběru tohoto přístupu a jeho předností pro tento projekt. Tato sekce vychází ze zdrojů [12] a [13].

Výhody tohoto přístupu jsou:

- Snadnější návrh – v některých případech je tento způsob návrhu rychlejší, protože není nutné ladit přesnost grafického návrhu, ale místo toho je možné vkládat komponenty přímo do stránky.
- Odpadá nutnost používat grafickou aplikaci.
- Snadnější testování – prototyp je spustitelný přímo na cílové platformě a navíc je možné některé funkce vyzkoušet a nebo alespoň nasimulovat, čímž je návrh realističtější a je zjednodušeno prvotní posouzení.

Nevýhody tohoto přístupu jsou:

- Složitější návrh – v některých případech je tento způsob návrhu zdlouhavý, například při vytváření nestandardních prvků nebo při vymýšlení nových postupů.
- Nutnost myslet na kvalitu kódu – při tvorbě grafického návrhu v prohlížeči se nutně musí grafik soustředit jak na kvalitu kódu (společně s nutností mít navržený základ aplikace), tak také na návrh, což může vést ke snížení kvality grafického návrhu.

- Technologické limitace – může se stát, že se bude navrhovat něco, co je jednoduše naprogramovatelné nebo něco, co je pro platformu běžné a ve výsledku tak budeme méně kreativní, protože v prvé řadě píšeme kód a až poté posuzujeme design. Tento bod může být zároveň výhodou, protože nejsou vymýšleny příliš komplexní prvky.

V případě této práce převažují výhody tohoto přístupu nad jeho nevýhodami. U grafické aplikace není nutné vytvářet velmi nestandardní grafické prvky a je žádoucí, aby návrh odrážel technologické limitace platformy. V případě této aplikace je navíc velmi důležité, abychom měli co nejdříve k dispozici prototyp na cílové platformě, což je jednou ze zásadních výhod tohoto přístupu. Pokud bychom požadovali vytvořit graficky nestandardní editor nebo jinak graficky výjimečnou aplikaci, tak by bylo výhodnější držet se běžnějšího způsobu návrhu v grafickém programu.

2.4.1 Rozložení editoru

První část návrhu je rozložení editoru, které se inspiruje rozložením ostatních grafických editorů. Na toto rozložení jsou uživatelé zvyklí a dá se očekávat, že výsledné rozhraní pro ně bude intuitivní a rychle akceptovatelné.

Rozložení editoru, viditelné na obrázku č. 2.1, je následující:

- V horní části editoru najdeme lištu nástrojů, která obsahuje funkce jako vrátit zpět, nový projekt, ukládání a načtení ze souboru, interaktivní prohlídku a další.
- V levé části je panel „Objekty a stránky“, který obsahuje stránky v aktuálním dokumentu a všechny objekty, které lze v dokumentu vytvářet.
- V pravé části je panel „Vlastnosti objektu“, ve kterém lze změnit vlastnosti aktuálně vybraného objektu.
- Uprostřed se nachází hlavní část editoru „Kreslící plátno“, která zobrazuje objekty a umožňuje s nimi pracovat. Tato část má upravené kontextové menu obsahující další funkce k aktuálně vybraným objektům.

Dalším použitým prvkem jsou modální okna pro vytváření projektu nebo výběr ikony a interaktivní prohlídku. Modální okno interaktivní prohlídky se zobrazuje jiným způsobem, protože je zaměřeno na prezentaci projektu bez rušivých elementů rozhraní editoru.

2.4.2 Lišta nástrojů

Lišta nástrojů je pomocí odsazení vizuálně rozdělená do třech sekcí. První sekce na levé straně obsahuje akce v historii, vytvoření nového dokumentu, export do obrázku a uložení a načtení ze souboru. Sekce uprostřed obsahuje

Obrázek 2.1: Rozložení editoru

ovládání přiblížení editoru, posun elementů v hloubce a nastavení přichycení elementů. Poslední sekce na pravé straně obsahuje pouze dominantní tlačítko pro vstup do interaktivní prohlídky.

2.4.3 Panel objekty a stránky

Tento panel se nachází v levé části a jeho hlavní funkcí je správa stránek v dokumentu a vytváření nových objektů na plátně. Níže jsou popsány obě tyto části.

2.4.3.1 Vytváření objektů

Klíčovou funkcí tohoto panelu je vytváření nových objektů, které jsou rozděleny do dvou hlavních kategorií, a to primitivní objekty a komplexní objekty.

Primitivní objekty představují základní objekty, u kterých předpokládáme, že si je uživatel přizpůsobí dle své potřeby. Komplexní objekty naproti tomu představují kompletně předpřipravené objekty s určitými asumpcemi o způsobu jejích používání a mají předem definované některé jejich vlastnosti, jako je barva, velikost a další. Tyto objekty se používají při přípravě rychlých UX prototypů, protože je stačí ve většině případů pouze přesunout na kreslící plátno bez dalších úprav. Všechny komplexní objekty jsou vytvořené tak, aby spolu graficky ladily.

Objekty je možné vytvářet buď pomocí přetažení na konkrétní místo na kreslícím plátně nebo pomocí kliknutí na objekt, které jej vytvoří do předem dané lokace na plátně.

2.4.3.2 Správa stránek

V této části panelu jsou viditelné všechny stránky v dokumentu, právě aktivní stránka a je zde možné přidat novou stránku nebo odstranit již existující stránku. Jednotlivá stránka představuje jednu stránku v mobilní nebo webové aplikaci, která vychází z designu vytvořeném v tomto editoru. Vlastnosti stránek je možné změnit v panelu vlastnosti objektu, pokud není vybraný žádný objekt na plátně.

Šířka stránky se definuje v projektu a je společná pro všechny stránky, protože se předpokládá, že u aplikací s více stránkami mají všechny stejnou šířku. Výška nově vytvořené stránky je dána výchozím nastavením projektu, ale stránkám lze výšku jednotlivě později změnit.

2.4.4 Panel vlastnosti objektu

Panel vlastností objektu se přizpůsobuje aktuálně vybranému objektu. Pokud není vybraný žádný objekt, tak se na panelu zobrazují editovatelné vlastnosti dokumentu a aktuální stránky. Při vybraném objektu se zde zobrazují editovatelné vlastnosti objektů jako je například pozice, rotace, barva pozadí, editace obrysu, vlastnosti textu nebo také nastavení odkazu na jiné stránky.

Hodnoty jednolivých vlastností jsou validovány v reálném čase, vstup s nesprávnou hodnotou se obarvuje červeným obrysem. Změny hodnot vlastností objektu se projevují již při jejich zadávání, protože v případě chyby lze vždy použít tlačítko zpět.

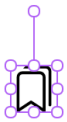
Validace a použití parametrů v reálném čase usnadňuje uživatelům používání aplikace, protože nemusejí hodnoty potvrzovat, ale hned je zřejmé, zda je hodnota validní či ne. Pokud uživatel vyplní nevalidní hodnotu a opustí vstupní pole, tak se do pole automaticky vrátí poslední použitá validní hodnota, díky čemuž se aplikace nenachází v nekomplentním stavu a uživatel má zároveň přehled jaká hodnota je aktuálně aplikována.

Další funkcí panelu vlastností objektu je otevření modálního okna pro výběr ikony, která se použije u objektu typu ikona. Panel také umožnuje nastavit objektům odkaz na jinou stránku, toto je funkce díky které lze pak testovat dokument pomocí interaktivní prohlídky.

2.4.5 Kreslící plátno

Kreslící plátno je hlavní prvek editoru, který zobrazuje obsah aktuální stránky a nabízí několik možností interaktivní editace objektů:

- Výběr objektů pomocí tažení myši – lze vybrat jeden nebo více objektů pomocí tažení myši.
- Transformace objektu – kolem vybraného objektu se zobrazuje transformační oblast, pomocí které lze prvek otáčet nebo škálovat. Škálování se



Obrázek 2.2: Transformační oblast

přizpůsobuje typu objektu, některým objektům se změní výška a šířka, jiným objektům se změní hodnota škálování. Jak vypadá transformační oblast je viditelné na obrázku č. 2.2.

- Přesun pomocí tažení myši – objekty lze na plátně přesouvat pomocí přetažení myši do jiné části plátna.
- Editace přes kontextové menu – při pravém kliku myší se zobrazí kontextové menu s nabídkou možných operací pro aktuálně vybrané objekty.
- Změna obsahu textového elementu přímo na plátně - obsah některých textových objektů je možné editovat přímo na plátně pomocí dvojitého kliku na takový objekt.

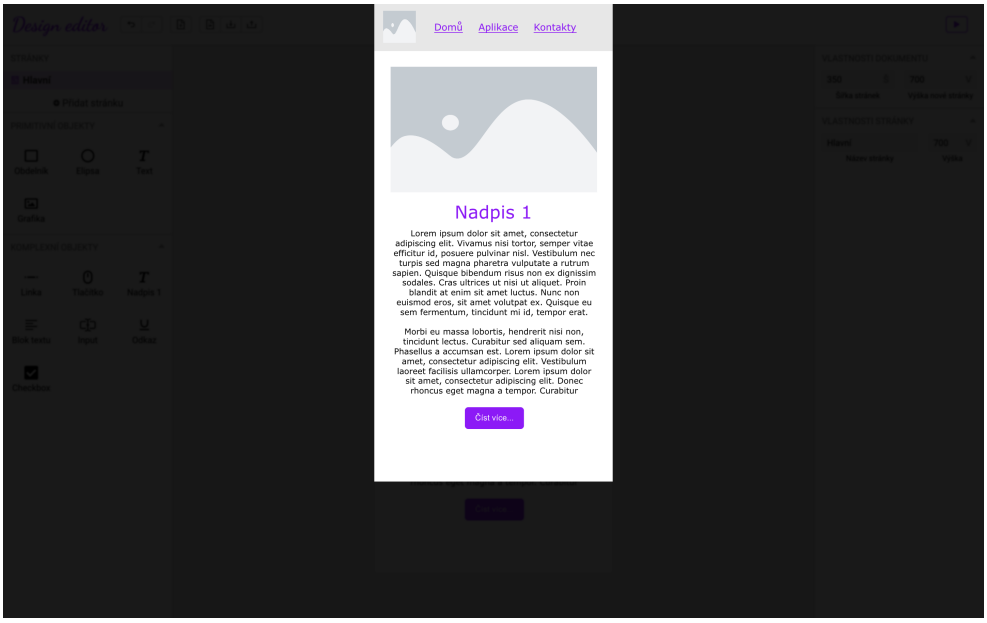
Pro zjednodušení organizace objektů na kreslícím plátně lze objekty seskupovat do skupin, které se pak chovají jako jeden větší objekt. Je možné, že uživatel bude potřebovat upravit jednotlivé objekty ze skupiny, a proto lze skupiny dočasně otevřít pomocí dvojitého kliknutí, které skupinu dočasně rozpadne na její dceřiné objekty a následně je lze individuálně upravovat. Po dokončení této operace se při výběru objektu mimo skupinu otevřená skupina opět zavře.

Další funkcí kreslícího plátna je přichycení k blízkým objektům. Při posunu objektu se vyhledávají okraje a středy blízkých objektů a ukládají se. Pokud je taková oblast dostatečně blízko, tak se v tomto místě na kreslícím plátně zobrazí vodítko, a pokud se původní objekt přiblíží do definované vzdálenosti, tak se k tomuto vodítku přichytí.

Tato funkce pomáhá uživatelům zarovnat objekty vůči sobě. Pokud by bylo potřeba objekt umístit těsně vedle vodítka nebo by uživateli tato funkce nevyhovovala, tak si ji může pomocí tlačítka nebo klávesové zkratky vypnout.

2.4.6 Interaktivní prohlídka

Po kliku na tlačítko interaktivní prohlídky v liště nástrojů se uživatel přesune do módu interaktivní prohlídky (viz obrázek č. 2.3). V módu interaktivní prohlídky jsou vypnuté všechny editační funkce, pozadí je ztmavené, čímž je vše zaměřeno na aktuální stránku zobrazenou uprostřed, ve které jsou aktivní odkazy z objektů na ostatní stránky.



Obrázek 2.3: Interaktivní prohlídka

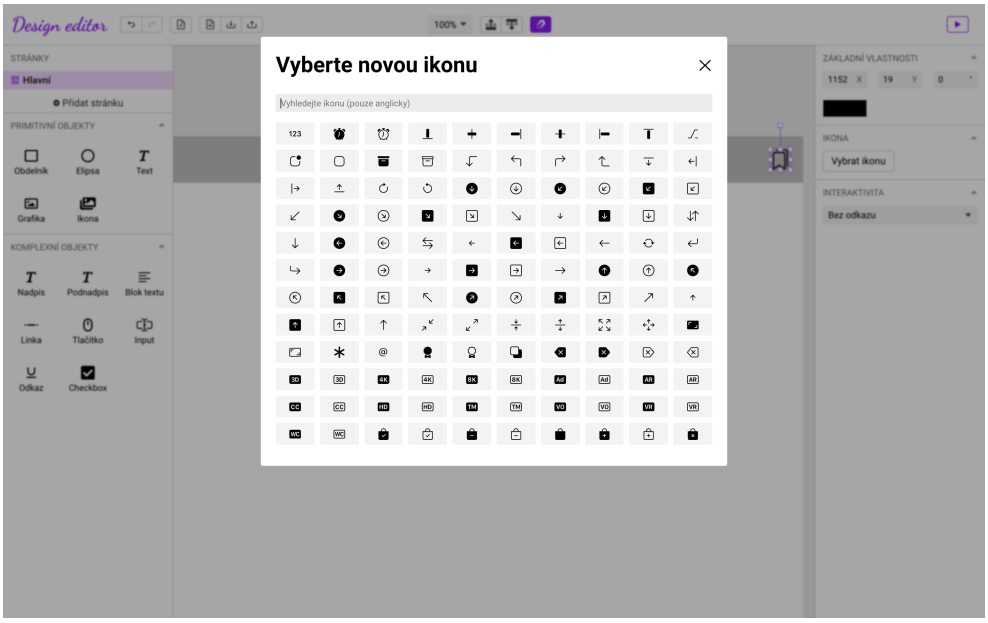
Uživatel nebo kritik má díky tomu možnost si v tomto prostředí nasimulovat interakci s reálnou webovou nebo mobilní aplikací. V tomto prototypu lze také provádět prvotní iteraci testování uživatelského rozhraní reálnými uživateli a vyhodnotit výslednou úspěšnost designu, s možností připravit seznam úprav, které bude nutné zapracovat pro zjednodušení jejího používání. Interaktivní prohlídka je tedy jeden z klíčových nástrojů UX designéra a přispívá ke zlepšení kvality grafického návrhu.

2.4.7 Modální okna

Hlavní výhodou modálních oken je, že uživatele nutí soustředit se na konkrétní úkol. Toho je dosaženo pomocí zobrazení modálního okna „nad“ zbytkem aplikace, a také pomocí vizuálního snížení důležitosti zbytku aplikace (například pomocí ztmavení). Uživatel je tak zaměřen na splnění úkolů, aniž by byl nucen opustit aktuální okno.

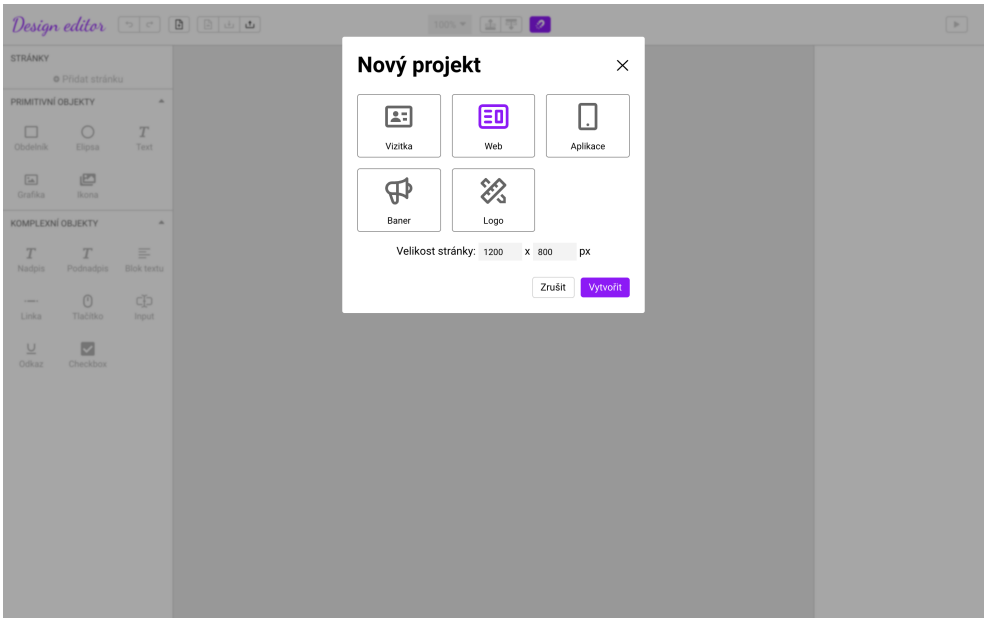
Jedno z hlavních modálních oken v aplikaci slouží k vytvoření nového projektu (viz obrázek č. 2.5). Dle zadání hry Agilně je zde uživateli nabídnuto pět předdefinovaných šablon pro vizitku, web, aplikaci, baner a logo. Pokud uživatel chce vytvořit dokument s jinými rozměry než mají tyto šablony, může rozměry snadno manuálně změnit.

Další modální okno v aplikaci, sloužící k výběru vzhledu ikony, se otevírá z panelu vlastností objektu při editaci objektu typu ikona (viz obrázek č. 2.4). Toto okno zobrazuje všechny dostupné ikony ve stylu tabulkového zobrazení a



Obrázek 2.4: Modální okno výběru ikony

je možné v něm vyhledávat dle anglických názvů ikon. Změna se automaticky provede po kliknutí na požadovanou ikonu, není nutné nic potvrzovat.



Obrázek 2.5: Modální okno pro vytvoření nového dokumentu

Kapitola 3

Realizace

Následující kapitola popisuje proces vývoje. Zaměřuje se na implementaci, strukturu kódu a modulů a také na pomocné nástroje použité při vývoji. Jsou zde dále popsány nejzajímavější části implementace.

3.1 Použité nástroje

Aplikace byla vyvinuta v prostředí WebStorm se studentskou licencí, které nabízí přímou podporu pro knihovnu React a usnadňuje orientaci v kódu a projektu. Pro správu balíčků aplikace je použit nástroj npm, React ovšem podporuje také yarn. Základní kostra projektu byla vytvořena pomocí oficiálního příkazu `create-react-app` (https://github.com/facebook/create-react-app).

3.2 Organizace kódu

Zdrojové kódy jsou rozděleny do následujících modulů, které jsou popsány dále v textu:

- componenets,
- contexts,
- fonts,
- HOC,
- redux,
- resources,
- utils.

3.2.1 Components

Modul components obsahuje všechny komponenty aplikace, mimo vstupní komponentu `App.jsx`. Komponenty jsou dále rozděleny do dalších modulů dle jejich použití a tyto moduly mohou obsahovat další podmoduly s komponentami, které tyto hlavní komponenty používají.

V modulu je také modul UI, který se používá pro vytváření stylizovaných elementů uživatelského rozhraní, které se v aplikaci používají opakovaně. Jsou to například tlačítka, tooltipy, modální okno a další.

Základem aplikace jsou následující hlavní komponenty, které tvoří rozložení a důležitou funkcionalitu:

- horní lišta,
- levá lišta stránek a vytváření objektů,
- pravá lišta vlastností objektů,
- kreslící plátno,
- interaktivní prohlídka.

Moduly komponentů mohou obsahovat lokální React Hooks, které se používají pouze v rámci jedné komponenty. Společné Hooks jsou součástí modulu `utils`, který bude popsán níže.

3.2.2 Contexts

Modul contexts obsahuje kontexty, které se používají v celé aplikaci. V případě této aplikace je to pouze `StageContext`, který pracuje s referencí na Konva objekt `Stage`, který není vhodné přidat do Redux stavu a zároveň s ním pracuje většina komponent nebo Hooků. Je to tedy vhodný adept na vytvoření kontextu.

Kontexty mají zásadní výhodu v tom, že předávají parametry celému stromu, bez nutnosti překreslit každou komponentu v tomto stromu. Překreslí se tak pouze consumer tohoto kontextu. To výrazně zvyšuje výkonnost aplikací, které potřebují pracovat se sdílenými parametry, které ale není vhodné přidat do Redux stavu.

3.2.3 Fonts

Modul fonts obsahuje lokální písma, která aplikace využivá. V tomto případě je v něm obsažen *Bootstrap* font s ikonami, které lze použít přímo na kreslícím plátně a dále obsahuje JSON soubor, který mapuje jednotlivé znaky v písmu na tyto ikony. Soubor se používá k dynamickému vytváření seznamu ikon v modálním okně výběru ikon a dále pro překlad názvu ikony na kód znaku

písma, který se používá pro zobrazení vybrané ikony. Ostatní písma v aplikaci jsou načtena pomocí balíčků npm.

3.2.4 HOC

Modul HOC obsahuje komponenty vyššího řádu, v případě této aplikace je to komponenta `WithContextMenu`, která nahrazuje základní kontextové menu na kreslícím plátně.

3.2.5 Redux

Modul Redux obsahuje nástroje pro správu aplikačního stavu pomocí balíčku Redux. Je rozdělena do dvou hlavních modulů:

- Reducers – obsahuje reducery a jejich akce.
- Selectors – obsahuje selectory, které lze použít nad reducery pro výběr konkrétních dat.

Redux stav je rozdělen na několik částí, z nichž hlavní je část `undoable`, která obsahuje všechny stavy, které lze vrátit zpět nebo načíst ze souboru. Dále stav obsahuje části `settings` a `stage`, které udržují nastavení aplikace nebo polohová data kompenenty `stage`.

3.2.6 Utils

Modul utils obsahuje pomocné nástroje, které se používají ve více komponentách. Typicky zde najdeme React Hooks nebo běžné JavaScriptové funkce.

Modul je dále rozdělen na moduly:

- app – nástroje používáne v celé aplikaci, obvykle s funkcí v rámci celé aplikace. Příkladem takových nástrojů je například Hook pro zpracování klávesových zkratek nebo Hooky pro stažení projektu.
- element – nástroje používané pro práci nad objekty jako je rekurzivní funkce pro vytvoření hluboké kopie celého elementu nebo skupiny, popřípadě Hooky pro odstranění elementu.
- project – nástroje pro práci nad projektem jako je Hook pro změnu velikosti kreslícího plátna nebo Hook pro počítání vlastností kreslícího plátna.
- undoable – nástroje pro práci nad funkcemi historie.

3.3 Kreslící plátno

Kreslící plátno je realizováno s pomocí knihovny Konva, která ale nabízí pouze základní rozhraní pro grafické elementy a pro plátno. Dále rozšiřuje React o komponenty, které s touto knihovnou umějí pracovat.

Plátno je poskládáno z několika komponent, přičemž každá mu přidává nějakou zásadní funkcionalitu. Hlavní komponenty, ze kterých se plátno skládá jsou následující:

- Canvas – hlavní vstupní komponenta,
- ScrollableCanvas – komponenta přidává funkcionalitu posuvání na plátně,
- SelectableCanvas – tato komponenta přidává funkcionalitu výběru elementů pomocí potáhnutí myši,
- SnappableCanvas – komponenta Snappable přidává funkcionalitu pro přichycení elementů k jiným prvkům,
- KonvaElement – rekurzivní komponenta pomocí které se vytváří jednotlivé elementy.

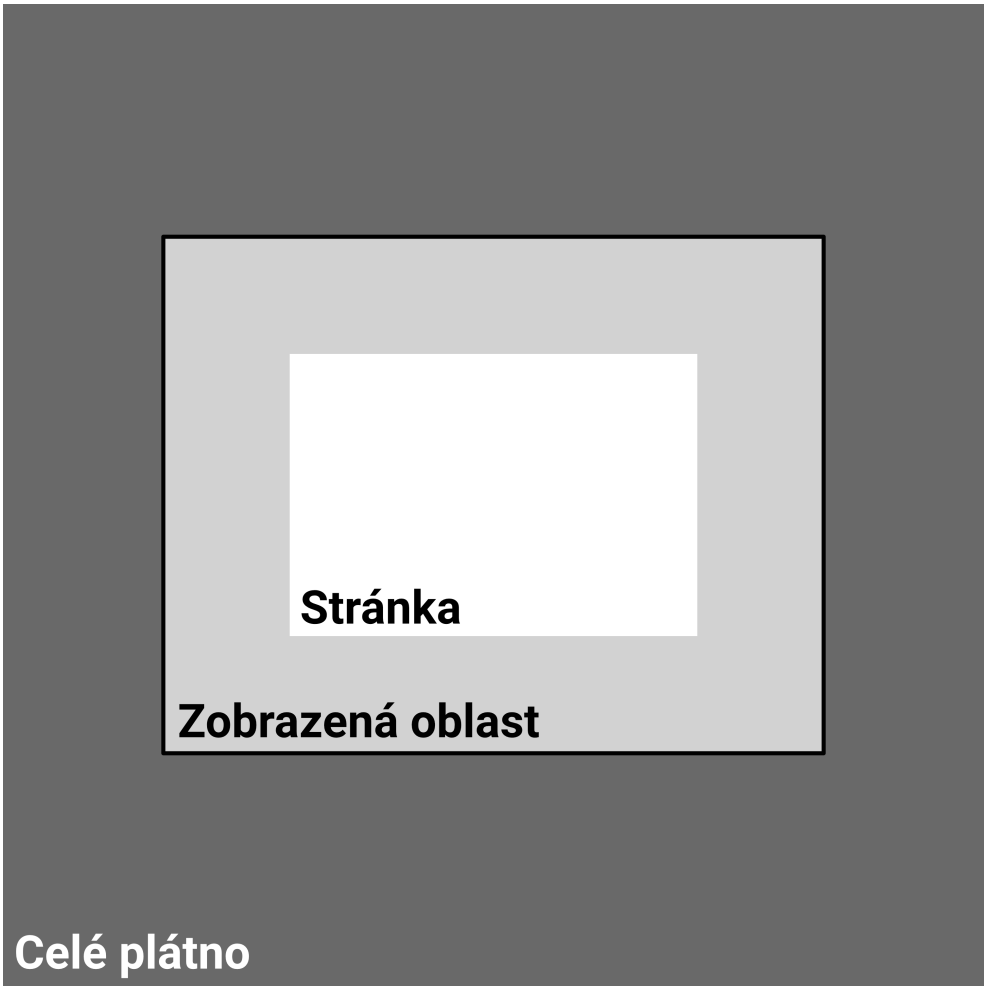
3.3.1 Canvas

Canvas je hlavní vstupní komponenta, která používá další rozšiřující komponenty. Zde stojí za zmínku především způsob zobrazení kreslícího plátna. Kreslící plátno je totiž nejen hlavní bílá oblast uprostřed, ale také neaktivní šedá oblast kolem. To umožňuje vkládat elementy částečně na stránku a částečně do neaktivní oblasti, a také to zjednodušuje navigaci na kreslícím plátně, protože lze pohodlně zobrazit i okraje hlavní oblasti. Oblasti plátna jsou viditelné na obrázku č. 3.1, reálně se ale vykresluje pouze viditelná oblast.

Výpočet velikostí všech oblastí má na starost React Hook, který kombinuje několik selektorů stavů pro stránky, stage a projekty. Dynamicky počítá velikost oblasti na základě velikosti zobrazené oblasti plátna v prohlížeči, která se mění v závislosti na velikosti prohlížeče, a snaží se co nejlépe optimalizovat jeho velikost, tak aby se v něm bylo možné efektivně pohybovat. Zároveň počítá odsazení bílé části plátna, tak aby byla vždy zarovnána na střed. Hook navíc vrací funkci, která umí převést souřadnice z prostoru celé fukční oblasti na prostor v hlavní bílé části.

3.3.2 ScrollableCanvas

ScrollableCanvas je komponenta, která má na starost zobrazení kreslícího plátna do viditelné oblasti v prohlížeči a jeho posun. Aby bylo dosaženo co nejlepšího výkonu, tak je posun plátna pouze simulován, což znamená že se



Obrázek 3.1: Rozložení kreslícího plátna

vždy zobrazuje pouze viditelná oblast rozšířená o okraje, které zajišťují plynulý posun bez problikávání. Díky tomu Konva nemusí vykreslovat elementy, které by nebyly viditelné, čímž je zvýšen výkon aplikace.

Simulovaný posun je realizován pomocí vytvoření Stage o velikosti zobrazené oblasti a přídáním okrajů tak, aby byl i rychlý posun plynulý. Následně je vytvořen `div` o velikosti celého plátna tak, aby se v prohlížeči zobrazily nativní posuvníky. Když se uživatel začne na plátně posouvat tak se Stage přemístí tak, aby byla opět celá viditelná a zároveň je změněn její offset tak, aby se obsah na plátně posunul v souladu s posuvníky. Pro uživatele je tím vytvořena iluze nativního posunu, ale s výhodou vyššího výkonu.

3.3.3 SelectableCanvas

SelectableElement je komponenta umožnující zobrazení oblasti pro výběr elementů. Oblast se vykresluje pomocí Konva obdélníku v rozdílné vrstvě než ostatní objekty. Na kreslícím plátně jsou zachyceny události kliku a pohybu myši a dle pozice těchto událostí se zobrazuje obdélník výběru.

Při uvolnění tlačítka myši se zavolá funkce pro výběr elementů, ve které jsou nalezeny všechny vybratelné objekty a pomocí knihovny Konva je zjištěno, zda mají průnik s oblastí výběru. Vybratelné objekty jsou rozlišovány pomocí speciálního parametru, který mají objekty přidaný do názvu. Například pokud není skupina otevřena pomocí dvojkliku, tak její objekty nejsou vybratelné zvlášť, ale vybírá se vždy jen celá skupina.

3.3.4 SnappableCanvas

SnappableCanvas je komponenta realizující zobrazení a výpočet vodítek, která slouží k zarovnání objektů mezi sebou. Nalezení vodítek probíhá následovně:

1. Nejprve jsou vybrány objekty na plátně, ke kterým je možné aktuální objekt přichytit, a jsou nalezeny pozice jejich okrajů a středů.
2. Dále se spočítají okraje a střed objektu, který chceme přichytit k vodítkům (tedy objekt se kterým aktuálně pohybujeme).
3. Následně se iteruje přes všechna vodítka z prvního kroku a dle jejich vzdálenosti od vodítek aktuálně vybraného objektu z druhého kroku je vyhodnoceno, zda se vodítko zobrazí. Zobrazení probíhá pomocí Konva komponenty `Line` s nekonečnou délkou. Při krátké vzálenosti aktuálního objektu od vodítka se objekt automaticky přesune na pozici tohoto vodítka.

3.3.5 KonvaElement

KonvaElement je komponenta, která vykresluje všechny uživatelem vytvořené objekty na plátně. Protože lze objekty seskupovat do skupin, tak je tato komponenta rekurzivní a může neomezeně vytvářet sebe sama. Skupiny mohou být do sebe bez omezení vkládány.

Skupiny jsou užitečné pro usnadnění práce s více objekty zároveň, nicméně v některých případech je užitečné mít možnost skupinu otevřít a upravit její objekty. Skupinu lze v této aplikaci otevřít pomocí dvojitého kliknutí. Následně je možné manipulovat s jejími elementy a pak ji kliknutím na jiné místo uzavřít. Funkce funguje rekurzivně, lze tak tímto způsobem otevřít libovolné množství do sebe vnořených skupin. Aplikace si ukládá otevřené skupiny a pokud uživatel dvakrát klikne na objekt patřící do stromu, který obsahuje nějakou neotevřenou skupinu, tak je otevřena první neotevřená skupina ve stromu skupin.

Komponenta umí také vytvořit komplexní objekty skládající se z více elementů, jako je například tlačítko, které se skládá z textu a zaobleného obdelníku. Komplexní objekty jsou definovány v modulu `complex-elements`.

3.3.5.1 Editace textu

Další funkcí této komponenty je editace textu přímo v kreslícím plátně, což je operace, pro kterou Konva nemá speciální objekt. Pokud uživatel edituje text na plátně, tak se místo Konva komponenty zobrazí nad objektem plátna HTML tag `input`, kterému jsou nastaveny stejné vlastnosti jako má textová komponenta. Uživatel má díky tomu pocit, že reálně edituje text přímo na kreslícím plátně. Po dokončení editace se texty inputu a textové komponenty synchronizují.

3.3.5.2 Kotvící body linek

V neposlední řadě je na kreslícím plátně možné upravovat délku linek pomocí dvou kotvících bodů, které jsou zobrazené na okrajích linky. Komponenta KonvaElement v tomto případě kontroluje, zda aktuálně vybraný element podporuje editaci přes kotvící body a pokud ano, tak se na okrajích objektu zobrazí dva kotvící body. Body jsou zobrazené jako kruhy v knihovně Konva. Při jejich přetažení na plátně se přepočítá editovaný objekt a tato změna se ihned projeví na plátně.

3.4 Šablony pro objekty

Pro zjednodušení vytváření objektů aplikace obsahuje jednoduchý systém šablon. Různé typy objektů jako je například text, tlačítko nebo ikona mají totiž odlišné vlastnosti a měly by mít různé výchozí nastavení.

Systém šablon obsahuje JSON šablony a funkci, která dle zadaného typu objektu vytvoří šablonu. Objekty mají určité základní vlastnosti, jako například pozici, rotaci a další, které se u většiny z nich opakují. Aby se nemusely tyto vlastnosti opakovaně zapisovat, tak byl do systému šablon implementován jednoduchý systém dědění, pomocí kterého lze zadávat názvy jiných objektů, ze kterých se budou dědit určité vlastnosti.

V komponentě `KonvaElement` se všechny vlastnosti ze šablon automaticky předávají jako arguementy do vytvořených komponent, což usnadňuje přidávání nových vlastností k objektům.

Objekty dále mají metadata, která slouží pro předání dalších informací pro interní využití. Metadata se používají například pro určení, zda je zapnutá editace textu přímo na plátně nebo zda lze objekt škálovat pomocí transformačního objektu.

3.5 Interaktivní prohlídka

Interaktivní prohlídka zobrazuje pouze bílou aktivní oblast bez okrajů s klikatelnými objekty. Vykresluje se podobně jako hlavní kreslící plátno pomocí knihovny Konva. Aby nebylo nutné duplikovat logiku mezi kreslícím plátnem a interaktivní prohlídkou, tak i interaktivní prohlídka využívá k vykreslení objektů stejnou komponentu jako kreslící plátno (KonvaElement).

Komponenta KonvaElement dostává argumenty, které vypínají editovatelný režim a zapínají odkazy na objektech mezi stránkami. Při kliku na objekt s odkazem se změní aktuální stránka na stránku odkazovanou.

3.6 Balíček ikon

Komplexní objekt ikona umožňuje na plátno vkládat sadu předdefinovaných ikon z balíčku ikon Bootstrap.

Problematickou částí implementace tohoto objektu je změna barvy, protože knihovna Konva nepodporuje u ikon dynamickou změnu barvy pozadí. Jedním z možných řešení je načíst obsah SVG, změnit jeho barvu v XML, a přidat tento objekt na plátno. Problémem tohoto řešení je, že při změně barvy pozadí dochází k problikávání objektu, protože Konva musí toto nově vytvořené SVG nejprve načíst.

Řešení, které bylo implementováno oproti tomu využívá speciální Bootstrap písmo, které obsahuje všechny ikony, a také soubor, který mapuje názvy ikon na kódy znaků. Tento soubor byl použit pro implementaci modálního okna s výběrem ikon. Ikony se poté na plátně zobrazují pomocí needitovatelného textového objektu, který umožňuje snadnou změnu barvy pozadí objektu. Uživatel nepozná, že se ikona vykresluje jako písmo, protože editor v panelu vlasností nezobrazuje nástroje pro úpravy textu.

3.7 Optimalizace

Součástí implementace aplikace byla také optimalizace výkonnosti. Základem optimalizace je pochopení způsobu vykreslování komponentů v Reactu, který je popsán v kapitole č. 1.3.2.

Najít problematické částí kódu vhodné k optimalizaci lze například profilováním za pomoci nástroje *React DevTools*. Tento nástroj zaznamená jak dlouho trvá vykreslení komponenty, proč se komponenta vykreslila, a další užitečné informace. Z nich pak lze vycházet při optimalizaci výkonnosti. [14]

Následující kapitola je zaměřena na těchto 5 hlavních metod optimalizace React aplikací: [14]

1. správný návrh stromu komponent,
2. memoizace komponentů a funkcí,

```
const memoizationShowcase = React.useCallback(
    () => {
        // tělo memoizované funkce, která nemusí mít
        // návratovou hodnotu
        updateSomething(x);
    },
    [x], // pole závislostí
);
```

Zdrojový kód 3.1: Ukázka memoizace pomoci useCallback

3. asynchronní načítání závislostí,
4. virtualizace oken a seznamů,
5. asynchronní náčítání obrázků.

3.7.1 Správný návrh stromu komponent

Při změně stavu rodičovské komponenty se překreslují i její dceřiné komponenty, což nemusí být žádoucí. Obecně je vhodné zajistit, aby se komponenty překreslovaly pouze, když to bude nezbytně nutné. To můžeme zajistit tak, že z dceřiných komponent, které nepoužívají stav rodičovské komponenty, uděláme sourozenecké komponenty. [14]

Tento princip se dá použít i jako obecné doporučení při návrhu stromu komponentů, ze kterých se aplikace skládá. Tato doporučení byla při implementaci dodržena.

3.7.2 Memoizace komponentů a funkcí

Memoizace je specifická forma kešování, která funguje na principu uložení návratové hodnoty funkce na základě vstupní hodnoty jejích argumentů. Pro stejné hodnoty argumentů je vrácen vždy stejný výsledek uložený v keši, kvůli tomu tato forma kešování funguje jen s čistými funkcemi, které nemají postranní účinky. [14]

Knihovna React obsahuje funkce pro memoizaci komponentů a funkcí, nicméně je nutné během implementace zvážit, kde je vhodné memoizaci použít, a kde naopak nikoli. Memoizace komponentů a funkcí je totiž způsob optimalizace, který funguje na principu výměny operační paměti za rychlejší běh aplikace, což v některých případech nemusí být žádoucí. [14]

V Reactu lze k memoizaci použít následující funkce: [14]

- `React.memo()` – funkce pro memoizaci komponenty na základě jejích vstupních argumentů, která funguje velmi dobře pro primitivní hodnoty

argumentů. Pro komplexní hodnoty, jako je objekt nebo pole, je nutné také naimplementovat vlastní logiku pro porovnání, jinak se bude komponenta překreslovat vždy.

- `React.useCallback()` – funkci se předává funkce, kterou chceme memoizovat a pole jejích závislostí. Pokud se hodnoty v poli závislostí změní, tak se zneplatní předchozí memoizace a bude vytvořena nová instance funkce. V opačném případě se použije instance z keše. Tímto způsobem lze memoizovat téměř jakoukoli funkci, včetně React Hooks, ukázka použití je viditelná ve zdrojovém kódu č. 3.1.

- `React.useMemo()` – funkce, která ukládá návratovou hodnotu na základě vstupních argumentů. Typicky se používá v případě složitých výpočtů polí nebo objektů. Podobně jako v předchozích případech se do této funkce předává funkce k memoizaci a pole jejích závislostí. Uložená hodnota bude přepočítána jen v případě změny hodnoty v poli závislostí.

Při implementaci této aplikace byla na mnoha místech použita memoizace pomocí funkce `React.useCallback()`. Použití ostatních způsobů memoizace zatím nebylo nutné.

3.7.3 Asynchronní načítání závislostí

Asynchronní načítání závislostí je jednou z důležitých technik optimalizace React aplikací. V základním nastavení se v prohlížeči při prvním vykreslení React aplikace načte najednou celý soubor obsahující celou aplikaci, včetně všech závislostí. Tento soubor je vygenerován sloučením všech zdrojových kódu a komponent, které aplikace potřebuje pro své spuštění. [14]

Výhoda tohoto přístupu je snížení počtu sítových requestů, nicméně pokud aplikace narůstá na velikosti, tak se zároveň zvětšuje velikost tohoto souboru. To má za následek zpomalení prvotního načtení stránky, čímž se může snižovat spokojenost uživatelů s rychostí aplikace. [14]

Knihovna React umožnuje využít takzvaný *lazy loading* komponent na vyžádání, který zajistí, že se komponenta načte až v případě, když ji bude potřeba zobrazit. Díky tomu může React rozdělit výchozí soubor na více částí, čímž se zvýší prvotní odezva aplikace, ale zároveň to znamená, že se některé části aplikace budou muset načíst později. [14]

Tento způsob optimalizace není pro tuto aplikaci vhodný, protože požadujeme, aby se celá aplikace načetla najednou. Aplikace neobsahuje více stránek a navíc na začátku načítáme všechny zdroje, aby mohla fungovat v režimu offline. Je však možné, že se v budoucnu tyto předpoklady změní.

3.7.4 Virtualizace oken a seznamů

Při vykreslování většího množství komponent, jako například v seznamech, se všechny komponenty vykreslí také v DOMu nehledě na to, zda jsou viditelné. To může mít negativní vliv na výkonnost aplikace. [14]

Pomocí virtualizace oken můžeme vykreslit pouze viditelnou část DOMu, a až ve chvíli, kdy uživatel posune stránku, vykreslíme nové komponenty a starší odstraníme. Tato technika pomáhá optimalizovat aplikace, které vykreslují velké množství nezobrazených komponent. K implementaci lze použít například knihovnu `react-window` nebo `react-virtualized`. [14]

Protože tato aplikace neobsahuje rozsáhlé seznamy, tak nebyla virtualizace oken nebo seznamů popsaná výše při implementaci použita. Nicméně optimalizace posunu na plátně funguje na podobném principu, protože kreslící plátno zobrazuje v daný čas pouze viditelné objekty, přičemž ostatní objekty se nevykreslují.

3.7.5 Asynchronní načítání obrázků

V případě, že aplikace obsahuje velké množství obrázků, tak je možné se vyhnout vykreslení všech obrázků najednou, což sníží dobu načtení stránky. Při využití asynchronního načítání obrázků se obrázek nenačte, dokud jej nebude potřeba zobrazit. [14]

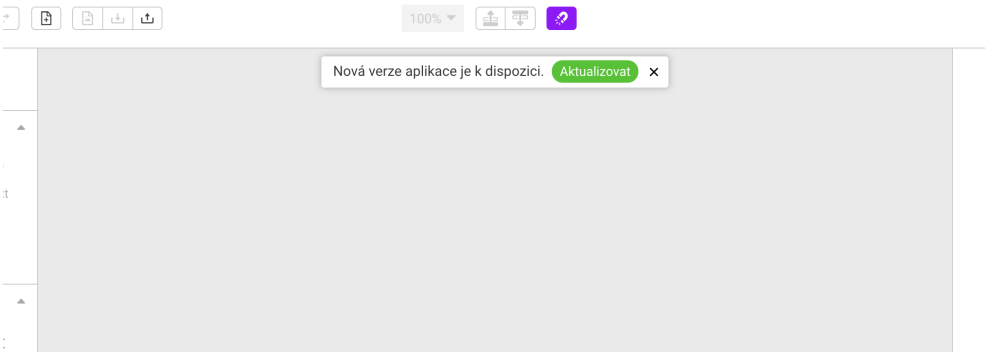
Asynchronní načítání obrázků vychází z podobného konceptu jako virtualizace oken a seznamů. K implementaci tohoto způsobu optimalizace lze použít například knihovnu `react-lazyload` a `react-lazy-load-image-component`. [14]

Tento způsob optimalizace nebyl při implementaci využit, protože aplikace neobsahuje velké množství obrázků.

3.8 Offline funkcionalita a instalovatelnost

Zásadní částí v návrhu aplikace je offline funkčnost a instalovatelnost, což dává uživateli pocit, že nepoužívá webovou, ale nativní aplikaci. Pro zajištění této funkce je nutné, aby webová aplikace splňovala následující podmínky: [15]

- Musí obsahovat *manifest* se základními vlastnostmi aplikace (jako je název aplikace, barevné schéma nebo způsob zobrazení) a různými verzemi ikon. Je nutné poznamenat, že se samozřejmě aplikace i při spuštění mimo prohlížeč vykresluje a spouští v jisté verzi prohlížeče určené k této funkčnosti.
- Musí obsahovat *Service Worker*, což je skript, který stránka nainstaluje do prohlížeče, a který následně může běžet na pozadí nezávisle na tom, zda je stránka spuštěná. Tím se vývojářům otevírají nové možnosti jako je příjem push notifikací a vytváření *middlewaru* pro requesty aplikace.



Obrázek 3.2: Aktualizace aplikace

Jednoduchý diagram principu Service Workeru je viditelný na obrázku č. 3.3.

- Musí být provozována na zabezpečené doméně (HTTPS).

V aplikaci tedy byl vytvořen základní manifest, sada ikon pro různá zařízení a Service Worker, který navíc umožňuje ovládat kešování requestů tak, aby aplikace běžela i bez přístupu k internetu. K tomu byl využit a nastaven oficiální Service Worker z šablony Reactu. [16]

3.8.1 Service Worker

Service Worker je rozdělen na registrační kód a samotný Service Worker. K jeho implementaci byla využita oficiální šablona Reactu pro vytváření projektu s aktivním Service Workerem, který byl doplněn o funkčnost popsanou níže.

Samotná registrace Service Workeru se provádí dle následujících scénářů:

1. Scénář v případě, že Service Worker této aplikace nebyl v minulosti nikdy nainstalován:
   a) Během načítání aplikace se do prohlížeče nainstaluje nejnovější verze Service Workeru.
   b) Aplikace přes Redux zaregistruje, že byl Service Worker poprvé nainstalován a zobrazí uživateli hlášku s informací, že byla aplikace uložena pro použití offline.
2. Scénář v případě, že již existuje nainstalovaný Service Worker, ale zároveň je k dipozici jeho novější verze:
   a) Na pozadí se zaregistruje nový Service Worker.

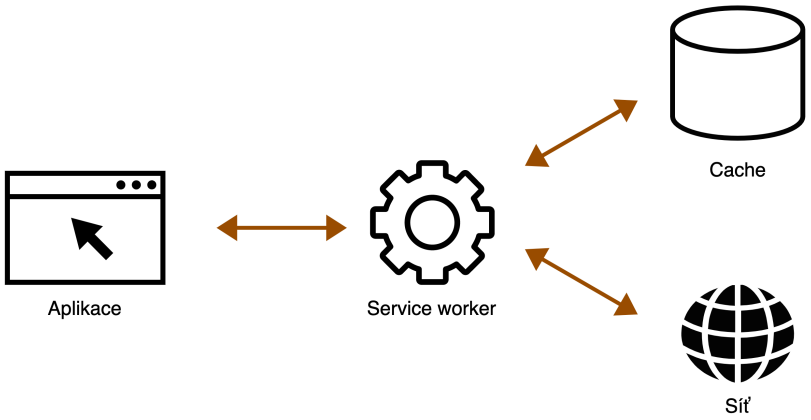
b) Aplikace přes Redux pozná, že byl zaregistrován nový Service Worker a zobrazí uživateli hlášku, že je k dispozici aktualizace aplikace s možností okamžité aktualizace (viz obrázek č. 3.2). Nový service worker je mezitím neaktivní, ve stavu „waiting“.

c) Pokud uživatel zvolí okamžitou aktualizaci, tak se do Service Workeru zašle událost „SKIP_WAITING“, která vyvolá okamžitou instalaci nového Service Workeru. Po dokončení instalace se aplikace automaticky přenačte, a již je aktivní nejnovější verze.

d) Pokud uživatel hlášku ignoruje, tak aplikace poběží ve staré verzi až dokud uživatel neukončí všechny její aktuální instance (karty v prohlížeči). Při následném otevření aplikace již bude automaticky používána nová verze.

Další funkčností Service Workeru je kešování statických zdrojů v projektové složce `src` pomocí knihovny Workbox. Zdroje z jiných míst, jako například z projektové složky `public` nebo ze síťových requestů, musí být kešovány explicitně. Pro jejich kešování lze použít následující strategie z Workboxu: [17]

- Stale-While-Revalidate – nejprve se kontroluje, zda není tento zdroj dostupný v keši. Pokud ano, tak se použije verze z keše a mezitím se na pozadí volá request na tento zdroj po jehož dokončení je zdroj aktualizován. Request na pozadí se provádí vždy, nezávisle na stáří keše.
- Cache First – strategie vhodná pro nekritické zdroje, které nejsou často aktualizovány. Je-li zdroj dostupný z keše, tak se vrátí tato verze a request na tento zdroj není zavolán. Pokud zdroj není k dispozici, tak se volá request a výsledek je uložen do keše.
- Network First – strategie vhodná pro zdroje, které jsou často aktualizovány. Vždy se nejprve volá request a zároveň je jeho výsledek uložen do keše. Jen v případě, že request selže, tak se vrací výsledek uložený v keši.
- Network Only – tato strategie vždy volá request a s keší vůbec nepracuje.
- Cache Only – tato strategie vždy vrací zdroj z keše a request není nikdy zavolán.



Obrázek 3.3: Princip Service Workeru

Kapitola 4

Testování

Testování je klíčová část vývoje softwaru, která přináší zvýšení kvality dodaného produktu. Obvykle testování rozdělujeme do těchto hlavních kategorií:

- Funkční testování – toto testování je zaměřeno na kontrolu všech funkčních aspektů softwaru, cílem je zjistit zda z funkcionality dostáváme námi očekáváný výsledek, čímž ověřujeme zda software funguje. [18]
- Nefunkční testování – jeho cílem je otestovat nefukční aspekty softwaru, jako například výkonnost, stabilitu, použitelnost a bezpečnost. Tento typ testování se nezaměřuje na to zda software funguje, ale na to jak dobře funguje. [18]
- Údržbové testování – jakmile je software nasazen do produkčního prostředí obvykle je nutné jej udržovat. Testování zaměřené na tuto fázi se nazývá údržbové testování. [19]

Pro účely této práce se zaměřím na dvě kategorie testování a to funkční a nefunkční testování. Z kategorie funkční testování jsem zvolil testování uživatelského rozhraní, které je pro tuto práci nejdůležitější. Z kategorie nefunkčního testování se zaměřím na testování použitelnosti, ostatní typy testů mohou samozřejmě být do aplikace doplněny později.

4.1 Testování použitelnosti

Testování použitelnosti je testovací metoda, která má za cíl odhalit problémy uživatelů při používání aplikace, získat data a zjistit míru spokojenosti uživatele s produktem. Při testování použitelnosti je velmi důležité mít konzistentní proces, abychom získali konzistentní a spolehlivé výsledky. Důvody proč toto testování provést: [20]

- Identifikace problémů při používání – při testování použitelnosti se obvykle odhalí největší problémy uživatelů při používání aplikace.

- Odhalení nových možností vylepšení aplikace.
- Zjištění preferencí uživatelů, které pomáhají v budoucnu upravit produkt pro vyšší míru spokojenosti uživatelů.

Testování použitelnosti má tři klíčové elementy: moderátora, testovací scénaře a testovaného uživatele. Moderátor provádí testovaného uživatele celým procesem, sleduje jeho chování, dělá si poznámky a na konci provádí s uživatelem rozhovor. Aby moderátor neovlinil výsledky testů, měl by do průběhu samotných testovacích scénářů zasahovat co nejméně. Moderátor by měl do scénáře zasáhnout pouze v případě úplného zaseknutí uživatele nebo při extrémně nepřiměřené délce jednoho testovacího scénáře. [20]

Dalším elementem jsou testovací scénáře, což jsou úkoly, které má uživatel provádět během testu. Scénáře mohou být otevřené nebo specifické. Je důležité dát si pozor na formulaci scénářů tak, aby neovlivnily průběh scénáře nebo neusnadnily uživateli jejich provedení. [20]

Posledním elementem je testovaný uživatel, který by měl patřit do cílové skupiny produktu. Takový uživatel už mohl v minulosti produkt používat a nebo se také může s produktem teprve seznamovat. Uživatel může být s jeho souhlasem i nahráván tak, aby se moderátor mohl k průběhu testování v budoucnu vrátit. [20]

Moderátor by se měl při sledování uživatele zaměřit na sledování následujích klíčových událostí, které mohou být použity k následnému vylepšení aplikace:

- Uživatel scénář sice splnil, ale jiným způsobem, než bylo očekáváno. To není nutně chyba, ale mělo by to být zaznamenáno a prověřeno ve vyhodnocovací fázi.
- Uživatel se v průběhu plnění scénáře dostane do nesprávného stavu, ale následně se vrátí do správného stavu.
- Uživatel v průběhu scénáře neví co má udělat, úplně se zastaví nebo proklikává nesouvisející prvky rozhraní. Zde je vhodné poznamenat co všechno a v jakém pořadí uživatel zkusil.
- Uživatel scénář sice splní, ale trvá mu delší dobu než je v rámci tohoto scénáře očekáváno.

4.1.1 Testovací scénáře

Pro účely uživatelského testování jsem připravil následujících 8 scénářů, které jsou popsány v tabulce č. 4.1. Scénáře budou provádět vždy všichni uživatelé.

Scénáře č. 1, č. 2 a č. 3 jsou zaměřeny na tvorbu projektu, vytvoření stránek, práci s nimi a jejích základní úpravy. Scénář č. 4 je zaměřen na samotnou

tvorbu designu. V rámci tohoto scénáře má uživatel za úkol vytvořit jednoduchý design, během kterého si vyzkouší funkce jako je přidávání elementů, posun, zarovnání a úpravy textu a škálování elementů. Scénář č. 5 je zaměřen na mazání elementů a vytváření skupin a scénář č. 6 na to navazuje kopírováním a úpravou skupin. Scénář č. 7 je zaměřen na tvorbu a použití interaktivní prohlídky a scénář č. 8 je zaměřen na export do obrázku a uložení projektu.

Tabulka 4.1: Testovací scénáře

Číslo scénáře	Název scénáře	Popis scénáře
1	Projekt	Vytvořte nový projekt v rozměru pro mobilní aplikaci.
2	Stránky	Vytvořte v projektu novou stránku, zmeňtě její výšku na 800 pixelů a pojmenujte ji kontakt.
3	Šířka stránek	Změňte šířku všech stránek na 400 pixelů.
4	Tvorba jednoduché stránky	Přejděte na stránku „Hlavní“ a vytvořte jednoduchou stránku, která v horní části bude mít menu (v tomto případě nám postačí pouze černý obdelník), následně bude obsahovat obrázek, nadpis, text a tlačítko „zobrazit vše“. Elementy a text by měly být zarovnané na střed.
5	Práce s elementy	Smažte ze stránky menu (černý obdelník) a z ostatních elementů vytvořte novou skupinu, kterou přesuňte výše na stránce.
6	Kopírování elementů a úpravu skupiny	Zkopírujte celou skupinu na stránku kontakt a následně změňtě text nadpisu na „Kontakt“.
7	Interaktivní prohlídka	Přidejte tlačítku „zobrazit vše“ odkaz na stránku „Hlavní“. Spusťe si aplikaci v interaktivní režimu.
8	Uložení a export	Exportujte si stránku „Hlavní“ do obrázku a celý projekt si uložte na disk.

4.1.2 Testovací protokol

Celý průběh testu je zaznamenán moderátorem do testovacího protokolu ve formě Google Forms, který je rozdělen do třech částí.

V první části se nachází základní údaje o testovaném uživateli, jako je jeho celé jméno, informace zda aplikaci v minulosti už používal a zda souhlasí s uvedením svého jména v diplomové práci.

V druhé části formuláře se nachází prostor pro sledování průběhu jednotlivých testovacích scénářu. Moderátor by měl u každého scénáře vyplnit, zda byl splněn, za jakou dobu byl splněn a doplnit případně další poznámky k průběhu scénáře. Například, zda se uživatel ztratil v nějaké části aplikace, nebo zda scénář provedl nestandardním způsobem.

V poslední části formuláře je uživatel požádán o subjektivní hodnocení aplikace, které se skládá z celkového hodnocení na stupnici 1 až 5, uvedením co se mu na aplikaci líbilo a co se mu nelíbilo. Uživatel by tuto část měl vyplnit upřímně, tedy zde je ideální pokud jej moderátor ujistí, že je klíčové, aby formulář vyplnil dle svých pocitů a aby aplikaci nedával vysoké hodnocení, pokud si to doopravdy nemyslí. Ukázka poslední části scénáře v Google Forms je vidět na obrázku č. 4.1.

4.1.3 Průběh testování

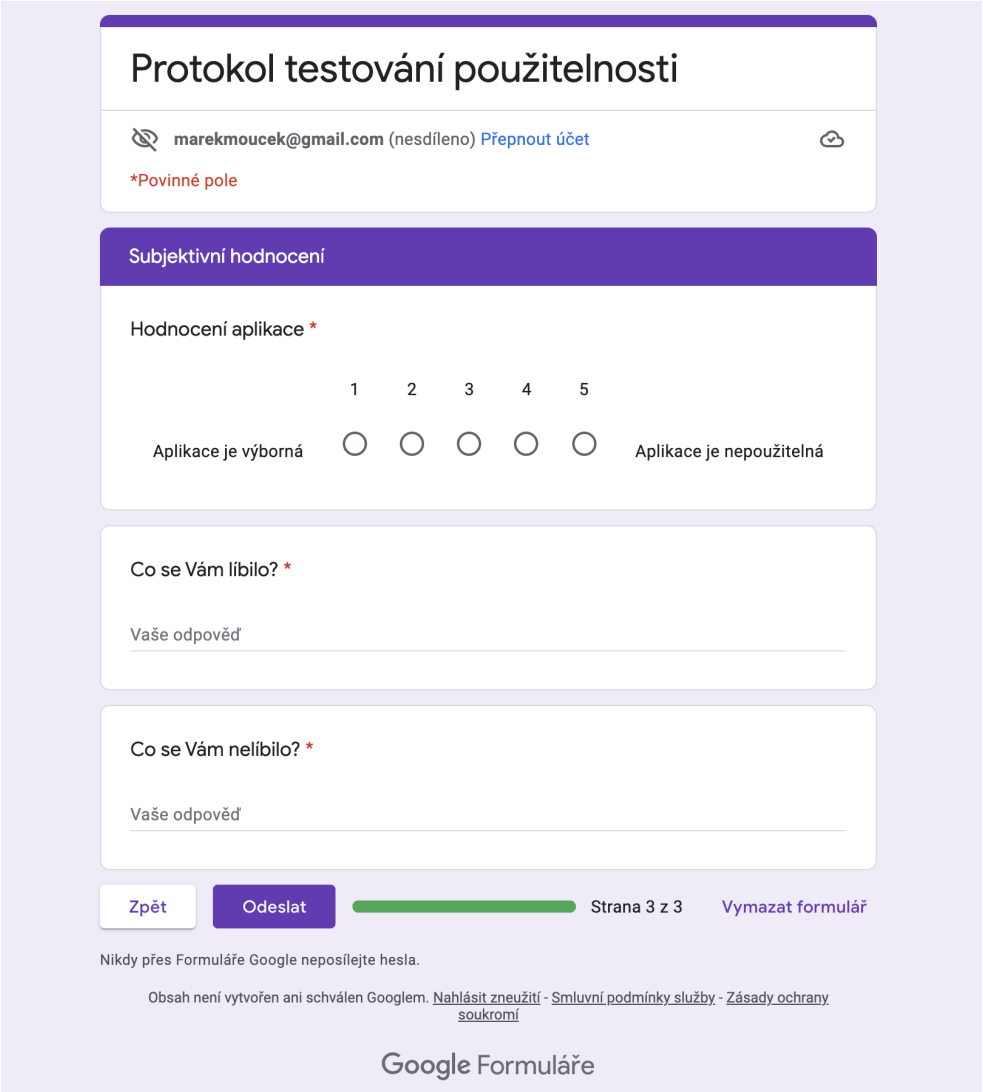
V této části je popsán průběh testování s šesti uživateli, se kterými byl prováděn test použitelnosti. Všichni uživatelé používali aplikaci poprvé a souhlasili s uvedením svého jména v této práci. Do testování byli zapojeni tito uživatelé:

- Bc. Lukáš Hepner – student ČVUT,
- Bc. Peter Hajtol – student ČVUT,
- Bc. Ondřej Vaniš – student ČVUT,
- Lucia Oravcová – studentka UK,
- Bc. Vojtěch Dalecký – student ČVUT,
- Bc. Tomáš Detko – student ČVUT.

Mezi uživateli je jedna studentka UK, která nepatří mezi primární skupinu uživatelů této aplikace, nicméně jsem ji pro diverzitu výsledků začlenil do testování.

4.1.3.1 Scénář č. 1 - Projekt

Tento scénář byl všemi uživateli splněn bez větších problémů. Pouze jeden uživatel se pokusil nejprve kliknout na neaktivní tlačítko „přidat stránku“, nicméně hned potom našel správné tlačítko.



Obrázek 4.1: Poslední část testovacího protokolu

4.1.3.2 Scénář č. 2 - Stránky

Scénář splnili všichni uživatelé, ale zároveň u všech kromě jednoho nastal problém se změnou výšky stránky, kdy všichni uživatelé změnili nejprve výšku nové stránky, následně si toho všimli, a místo toho změnili správně výšku aktuální stránky. Tři uživatelé se také pokusili stránku přejmenovat pomocí dvojkliku na stránku v levém panelu.

4.1.3.3 Scénář č. 3 – Šířka stránek

Scénář splinili všichni uživatelé bez problémů. Jeden uživatel zkontroloval po změně šířky všechny ostatní stránky, aby si ověřil, zda se u nich také změnila šířka.

4.1.3.4 Scénář č. 4 – Tvorba jednoduché stránky

Scénář byl splněn všemi uživateli, ale dva uživatelé hledali v rozhraní tlačítko pro hromadné zarovnání objektů. Někteří uživatelé vytvářeli nové objekty na plátně pomocí přetažení, a někteří pomocí kliknutí. Jeden uživatel se pokoušel upravit text tlačítka přímo v kreslícím plátně. Jeden uživatel se pokusil na plátně přesunout všechny objekty zároveň, aniž by vytvořil skupinu, což je funkce, kterou editor nepodporuje.

4.1.3.5 Scénář č. 5 – Práce s elementy

Tento scénář splnil tři uživatelé bez problémů, jednomu uživateli trval velmi dlouho, a dva uživatelé potřebovali radu moderátora. Problémy se scénářem byly následující:

- Některým uživatelům chvíli trvalo, než přišli na to, že plátno má kontextové menu.
- Dva uživatelé se pokusili objekty smazat pomocí klávesy *delete*, ale v aplikaci funguje pouze klávesa *backspace*.
- Dva uživatelé se místo vytvoření skupiny pokoušel přesunout více objektů zároveň, což je funkce, kterou aplikace aktuálně neumí.
- Jedna uživatelka nevěděla, jak vybrat více objektů zároveň.

4.1.3.6 Scénář č. 6 – Kopírování elementů a úpravu skupiny

Scénář byl splněn všemi uživateli, nicméně uživatelé postupovali různě. Čtyři uživatelé skupinu nejprve duplikovali přes kontextové menu, a potom se ji pokoušeli přesunout na jinou stránku. Očekáváný scénář je však kopírování pomocí klávesové zkratky. Dva uživatelé skupinu rozdělili, upravili text nadpisu, a potom skupinu opět sloučili místo toho, aby využili dočasné otevření skupinu pomocí dvojkliku.

4.1.3.7 Scénář č. 7 – Interaktivní prohlídka

Scénář byl všemi uživateli splněn bez problémů. Jeden uživatel se pokusil interaktivní prohlídku zavřít pomocí klávesy *escape*.

4.1.3.8 Scénář č. 8 – Uložení a export

Scénář byl všemi uživateli splněn dle očekávání.

4.1.4 Subjektivní hodnocení aplikace

Před subjektivním hodnocením aplikace byly uživatelům představeny všechny funkce aplikace a uživatelé dostali čas na vlastní prohlídku těchto funkcí. Dva uživatelé hodnotili aplikaci známkou 1 a čtyři uživatelé hodnotili aplikaci známkou 2, kde 1 je výborná aplikace a 5 je nepoužitelná aplikace.

Pět uživatelů se shodlo na tom, že aplikace se velmi snadno používá, je designově čistá a všechny hlavní prvky jsou na první pohled viditelné. To je pozitivní výsledek, protože to bylo také jedním z hlavních cílů aplikace. Tři uživatelé chválili mód interaktivní prohlídky. Dále se uživatelům líbila funkce přichytávání prvků, možnost používat klávesové zkratky a dále možnost otevřít skupinu pomocí dvojkliku.

Dva uživatelé se shodli na tom, že jim v uživatelském rozhraní chybělo tlačítko pro zarovnání objektů, a také na tom, že by bylo dobré kdyby aplikace podporovala více klávesových zkratek, a také že by možnost „zkopírovat“ měla být obsažena i v kontextovém menu. Další nápady uživatelů na vylepšení aplikace:

- Po vytváření textových objektů by se mohl rovnou editovat text.
- Měl by fungovat přesun více vybraných objektů zároveň.
- Bylo by dobré kdyby v komlexních objektech v levém menu bylo také rozbalovací menu a „radio button“.
- K tlačítku interaktivní menu by měl být umístěn i text.
- Barvy vodících linek pro zarovnání by mohly být dle kontextu různé (například pro zarovnání na střed by byla jiná barva vodící linky, než pro zarovnání vůči jinému objektu).
- Stránky by se mohly editovat společně vedle sebe na jedné obrazovce, místo rozdělení v editoru do více obrazovek.

4.1.5 Vyhodnocení testování a vylepšení aplikace

Přestože byly během testování nalezeny drobné problémy, tak uživatelé aplikaci hodnotili kladně a obecně se jim líbila designová čistota a snadná použitelnou. Nalezené problémy jsou hodnotné informace, na jejichž základě lze aplikaci dále vylepšovat.

V následujícím seznamu jsou popsány časté problémy během testování a rovnou provedné úpravy aplikace vedoucí k jejich vyřešení:

- Častým a důležitým problémem byla úprava výšky nové stránky, místo výšky aktuální stránky. Na základě toho byla tedy výška nové stránky z aplikace úplně odstraněna a při vytváření stránky se používá výška aktuální stránky. Výška nové stránky nebyla klíčovou, ani užitečnou funkcionalitou, která by musela v aplikaci zůstat na úkor jednoduchosti používání.

- Dalším problémem bylo kontextové menu. Některým uživatelům v něm chyběly funkce, a pro jiné uživatele bylo matoucí. Kontextové menu na plátně zobrazovalo vždy jen aktuálně dostupné možnosti, což mohl být jeden z důvodů zmatení, protože jsou uživatelé z ostatních aplikací zvyklí na to, že se v menu zobrazují všechny možnosti, ale některé jsou nedostupné. Do menu byly tedy přidány možnosti pro kopírování a vložení objektů, a také se nyní v kontextovém menu zobrazují vždy všechny položky. Nedostupné položky jsou zašedlé. To zvyšuje přehlednost kontextového menu, díky čemuž má uživatel na první pohled jasno jaké akce menu nabízí.

- Dále si uživatelé stěžovali na nemožnost zarovnání více objektů. Na základě toho byly do horního menu přidány tlačítka s různými možnostmi zarovnání.

- Posledním vylepšením bylo přidání podpory více klávesových zkratek. Pro smazání byla k aktuální klávesové zkrátce přidána také klávesová zkratka *delete*. Modální okna je nyní možné zavřít pomocí klávesy *escape*.

Na základě výsledků testování bych také v budoucnu doporučil implementovat následující vylepšení:

- Možnost přesouvat více vybraných objektů na plátně zároveň.

- Stránka by měla jít přejmenovat přímo v levém panelu se stránkami.

- Text by mělo být možné editovat přímo na plátně všem objektům, které obsahují text. Aktuálně je to možné pouze u čistě textových objektů, ale už ne u složených objektů jako je tlačítko.

- Vodící linky by měly mít různé barvy dle kontextu použití (například jinou barvu pro zarovnání na střed a jinou barvu pro zarování vůči jinému objektu).

4.2 Testování uživatelského rozhraní

Testování uživatelského rozhraní je důležitá fáze, která může být prováděna manuálně testerem nebo automaticky pomocí softwarových nástrojů. Níže se zaměřím na automatické testování, manuální testování bylo prováděno periodicky během vývoje.

Automatické testování uživatelského rozhraní je pro vývojáře klíčový nástroj, který přináší značné úspory prostředků a času. Při vývoji je totiž vysoká pravděpodobnost, že se do kódu dostanou chyby, které by narušovaly uživatelský požitek z aplikace. Díky testování mohou být chyby odhaleny dříve, čímž je snížen jejich dopad na produkt.

4.2.1 Nástroj Cypress

Mezi nejznámnější nástroje automatického testování uživatelského rozhraní webových aplikací patří *Cypress* a *Selenium*. Hlavní rozdíly mezi těmito nástroji jsou následující: [21]

- Cypress spouští testy přímo ve webovém prohlížeči, oproti tomu Selenium spouští testy mimo prohlížeč.
- Cypress na rozdíl od Selenia sám čeká na dokončení načtení DOMu, čímž elimunuje nutnost manuálně programovat čekací funkce. Cypress navíc automaticky stránku posouvá tak, aby byl vždy testovaný element na stránce viditelný.
- Na rozdíl od Selenia Cypress podporuje pouze JavaScript, využití jiných jazyků není možné.
- Vytváření testovacích scénářů v Seleniu je mnohem složitější, protože je nutné manuálně čekat na události jako je načtení stránky a elementů na stránce.
- Vytváření prvotního testovacího prostředí je v Seleniu složitější než v Cypressu.
- Selenium má lepší podporu prohlížečů, Cypress aktuálně podporuje verze prohlížečů Chrome, Edge, Electron a Firefox, Selenium podporuje navíc například Safari a Operu.

Na základě těchto rozdílů bylo vyhodnoceno, že pro tuto aplikaci je Cypress vhodnější nástroj než Selenium. Klíčovým faktorem je velká úspora času při

psaní testů a nastavení prostředí. Podpora pouze JavaScriptu není pro tento projekt problémem a podpora prohlížečů je dostačující.

4.2.1.1 Testování v Cypressu

Cypress spouští aplikace v simulovaném webovém prohlížeči a následně automaticky proklikává aplikaci dle kódu v testovacích souborech, který může přistupovat k HTML elementům na stránce. Cypress automaticky čeká na načtení stránky a elementů, psaní testů se tedy zaměřuje pouze na požadovanou funkcionalitu. [22]

Pro nalezení HTML elementů pro účely testování je v React komponentách přidán HTML atribut `data-cy`, což je jednoduché řešení, které zajistí unikátnost a zároveň žádným způsobem nenarušuje vývoj aplikace. Dalším možným řešením by bylo rozlišení pomocí tříd nebo id. To by ale mohlo kolidovat se styly v aplikaci, proto je řešení pomocí atributu považováno za vhodnější.

Ukázka použití Cypressu je viditelná ve zdrojovém kódu č. 4.1. Je zde znázorněno použití Cypress příkazů jako je `setupProject`, který provede úvodní nastavení projektu, dále obsahuje příkaz `customGet`, který získá elementy dle atributu `data-cy`, a také příkaz `mouseDrag`, který simuluje přetažení elementu pomocí myši. Je zde také použita metoda `canvasMousePos`, která převádí souřadnice z globálních na lokální vůči canvasu.

4.2.1.2 HTML canvas

Při použití automatických testovacích nástrojů je v této aplikaci problematické používání HTML canvasu, který je ovšem klíčový pro správnou funkci aplikace a je nutné jej v rámci testovacích scénářů řádně otestovat. Canvas je problematický, protože nevytváří žádné HTML elementy. Grafické objekty na něm tak nelze pomocí Cypressu najít. Problém s testováním canvasu by měl částečně řešit nástroj Percy.

4.2.2 Nástroj Percy

Nástroj Percy slouží k vizuální kontrole uživatelského rozhraní a lze jej použít v kombinaci s Cypress testováním. Pomocí knihovny Percy lze v určitých částech Cypress testování udělat screenshot, který se odesílá do aplikace https://percy.io/ a je porovnán vůči screenshotům z posledního testování. Pokud Percy najde vizuální odlišnosti, tak se screenshot označí k manuálnímu porovnání. Tester manuálně ověří odlišné screenshoty, a buď je schválí nebo ne, čímž je test dokončen.

Při úplně prvním testu ještě nejsou k dispozici screenshoty k porovnání, takže se označí ke kontrole všechny screenshoty. Rozhraní Percy je viditelné na obrázku č. 4.2.

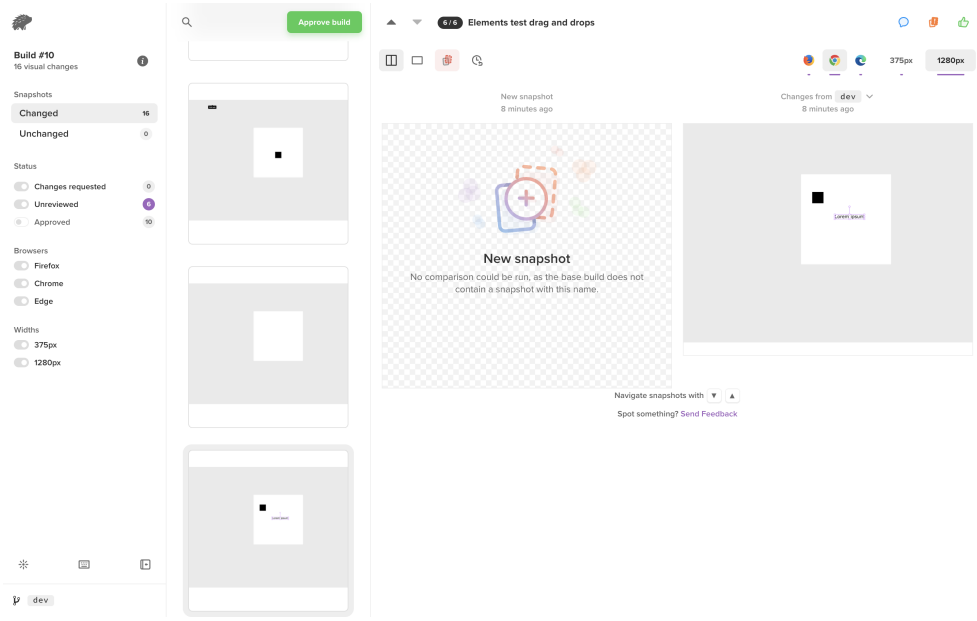
```
describe('History test', () => {
    it('undoes and redoes', () => {
        cy.setupProject();

        cy.customGet('history-undo').should('be.disabled');
        cy.customGet('history-redo').should('be.disabled');

        cy.customGet('create-rect').click();
        cy.customGet('canvas').mouseDrag(
            canvasMousePos(278, 246),
            canvasMousePos(350, 340)
        );

        /* ... zkráceno pro účely ukázky ... */
    });
});
```

Zdrojový kód 4.1: Ukázka Cypress testu



Obrázek 4.2: Nástroj Percy

4.2.2.1 Použití nástroje Percy

Pro účely této práce byl nástroj Percy použit pouze pro kontrolu objektů na canvasu, protože je nežádoucí, aby změny v uživatelském rozhraní, jako je přidání tlačítek nebo změna stylu grafiky, vyvolaly nutnost vizuální kontroly všech testovaných obrazovek. Zacílení nástroje pouze na canvas bylo dosaženo pomocí speciálního CSS pravidla pro Percy knihovnu `@media only percy`, pomocí kterého se v Percy screenshotech skrývají elementy uživatelského rozhraní tak, aby byl zobrazen pouze canvas.

Nástroj Percy lze používat zdarma až do výše 5 000 screenshotů měsíčně, což je pro účely této práce dostačují.

4.2.3 Testovací moduly

Automatické testování je rozděleno do následujících modulů (přičemž každý z nich se zaměřuje na určitou část aplikace):

- Element – testuje funkce prováděné s objekty, jako například drag and drop a smazání objektů.
- Transformer - testuje funkce transformeru elementu, který se zobrazuje nad objekty na canvasu.
- New project – testuje vytváření nového projektu a přidělování nejrůznějších atributů.
- Pages – testuje práci se stránkami v editoru, jako je vytváření, mazání a úprava stránky.
- History – testuje správnou funkci historie v aplikaci, jako je návrat zpět a opakování akce.
- Interactive play – testuje správnou funkci spuštění interaktivní prohlídky.
- Change Z index – testuje hloubkový posun objektu na canvasu, což je například posun objektu do popředí.

Do Cypressu bylo doplněno několik vlastních příkazů, které usňadňují testování této aplikace. Mezi nové příkazy patří příkaz pro základní nastavení projektu, příkaz pro realizaci tažení objektu myší a příkaz pro získání objektu dle atributu `data-cy`, který se v kódu používá pouze pro účely testování.

V budoucnu bych doporučil doplnit do těchto modulů další testovací scénáře tak, aby pokrývaly co nejvíce případu použití, protože kvalita aplikace je závislá na pokrytí kódu testy a také na kvalitě manuálního testování. Z již vytvořených scénářů se dá vycházet.

Závěr

Tato práce se zabývala analýzou hry Agilně, která si klade za cíl v rámci jednoho cvičení seznámit studenty s agilním vývojem především ve frameworku SCRUM. Cílem této práce je zefektivněním procesu vytvoření grafického návrhu, který je klíčovou součástí této hry. Na základě analýzy hry byly zformulovány hlavní funkční a nefunkční požadavky, use casy a byl popsán a vytvořen návrh nového grafického editoru sloužícího pro zefektivnění tohoto procesu.

Nově implementovaná webová aplikace je zaměřená pouze na tvorbu grafického návrhu pro hru Agilně, čímž urychluje a usnadňuje průběh této hry. Má moderní grafické rozhraní rozdělené do několika částí dle účelu a využívá modální okna pro usnadnění uživatelských interakcí s aplikací. Aplikace umožňuje navhrnout prototyp webové aplikace, včetně rozdělení na jednotlivé stránky. Tento prototyp je možné následně spustit v interaktivní prohlídce. Pro snadný návrh grafiky jsou v aplikaci připraveny nejpoužívanější objekty, včetně komplexních objektů pro okamžité použití na kreslícím plátně, které není nutné editovat.

Aplikace byla implementována za použití nejnovějších technologií a přístupů. Pro implementaci byla využita knihovna React a pro udržování globálního stavu byla použita knihovna Redux. Pro tvorbu kreslícího plátna byla využita knihovna Konva. Tyto technologie a přístupy byly popsány v rámci kapitoly rešerše. Implementace je modulární, což zlepšuje její udržitelnost a usnadňuje její budoucí rozvoj. Hlavní důraz byl kladen na část kreslícího plátna, které má velké množství funkcí, a proto byla tato část rozdělena na více modulů zaměřených pouze na jednu funkci.

Po prvním otevření se aplikace uloží do prohlížeče a v podporovaných prohlížečích je možné ji navíc nainstalovat do systému. To zrychluje její používání, a lze díky tomu aplikaci používat i bez aktivního připojení k internetu. Této funkce bylo dosaženo pomocí technologie Service Worker a pomocí správného nastavení aplikačního manifestu.

Nakonec byla aplikace řádně otestována automatickými testy s využitím nástrojů Cypress a Percy, které usnadňují psaní testů a vizuální kontrolu kres-

lícího plátna. Dále bylo provedeno uživatelské testování použitelnosti, které má za cíl odhalit problematické části ve funkčnosti aplikace. Testování proběhlo úspěšně a uživatelé hodnotili aplikaci kladně. Jako hlavní pozitivum nejčastěji zmiňovali čistý design a intuitivní uživatelské rozhraní. Během testování bylo nalezeno několik problémů, z nichž byly ty nejčastější problémy rovnou opraveny, a k těm méně častým byla doporučena budoucí vylepšení.

Literatura

[1] Desktop App vs Web App: Comparative Analysis. *Digital Skynet*, červen 2020, [cit. 2021-09-05]. Dostupné z: https://digitalskynet.com/blog/Desktop-App-vs-Web-App-Comparative-Analysis

[2] Desktop Browser Market Share Worldwide. *StatCounter*, [cit. 2021-11-04]. Dostupné z: https://gs.statcounter.com/browser-market-share/desktop/worldwide

[3] Google Trends: React, Angular, Vue. *Google Trends*, 2021, [cit. 2021-10-02]. Dostupné z: https://trends.google.com/trends/explore?date=2021-01-01%202021-12-31&q=react,Angular,Vue

[4] React: A JavaScript library for building user interfaces. *Facebook*, 2021, [cit. 2021-10-11]. Dostupné z: https://reactjs.org/

[5] React: Should I use Hooks, classes, or a mix of both? *Facebook*, 2021, [cit. 2021-10-12]. Dostupné z: https://reactjs.org/docs/hooks-faq.html#should-i-use-hooks-classes-or-a-mix-of-both

[6] Prasanjith, D.: 6 Reasons to Use React Hooks Instead of Classes. *Medium*, září 2020, [cit. 2021-10-12]. Dostupné z: https://blog.bitsrc.io/6-reasons-to-use-react-hooks-instead-of-classes-7e3ee745fe04

[7] Jöch, D.: Functional vs Class-Components in React. *Medium*, červenec 2018, [cit. 2021-10-12]. Dostupné z: https://djoech.medium.com/functional-vs-class-components-in-react-231e3fbd7108

[8] Redux: A Predictable State Container for JS Apps. *Redux*, 2021, [cit. 2021-10-15]. Dostupné z: https://redux.js.org/

[9] Konva.js - HTML5 2d canvas js library for desktop and mobile applications. *KonvaJS*, 2021, [cit. 2021-10-20]. Dostupné z: https://konvajs.org/

[10] Functional vs Non Functional Requirements. *GeeksForGeeks*, duben 2021, [cit. 2021-10-05]. Dostupné z: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[11] Use Cases. *usability.gov*, říjen 2021, [cit. 2021-10-09]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/use-cases.html

[12] Peham, T.: The Ultimate Guide To Designing In The Browser. *usersnap*, [cit. 2021-11-02]. Dostupné z: https://usersnap.com/blog/guide-designing-in-the-browser/

[13] Assentorp, P.: Stop designing in Sketch. Design in the Browser. *prototypr.io*, březen 2017, [cit. 2021-11-02]. Dostupné z: https://blog.prototypr.io/stop-designing-in-sketch-design-in-the-browser-c102bcdcdbb

[14] Mojeed, I.: 5 React performance optimization techniques. *LogRocket*, září 2021, [cit. 2021-10-22]. Dostupné z: https://blog.logrocket.com/5-react-performance-optimization-techniques/

[15] How to make PWAs installable. *MDN Web Docs*, [cit. 2021-9-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs

[16] Making a Progressive Web App. *Create React App*, [cit. 2021-9-02]. Dostupné z: https://create-react-app.dev/docs/making-a-progressive-web-app/

[17] Workbox Strategies. *Workbox*, květen 2021, [cit. 2021-10-20]. Dostupné z: https://developers.google.com/web/tools/workbox/modules/workbox-strategies

[18] Functional Testing Vs Non-Functional Testing. *Software Testing Help*, listopad 2021, [cit. 2021-11-11]. Dostupné z: https://www.softwaretestinghelp.com/functional-testing-vs-non-functional-testing/

[19] Maintenance Testing. *Better QA*, [cit. 2021-11-11]. Dostupné z: https://betterqa.co/software-testing-services/maintenance-testing/

[20] Usability Testing 101. *Nielsen Norman Group*, [cit. 2021-11-13]. Dostupné z: https://www.nngroup.com/articles/usability-testing-101/

[21] Unadkat, J.: Cypress vs Selenium: Key Differences. *BrowserStack*, červen 2020, [cit. 2021-11-20]. Dostupné z: https://www.browserstack.com/guide/cypress-vs-selenium

[22] Testing has been broken for too long. *Cypress*, [cit. 2021-11-25]. Dostupné z: https://www.cypress.io/how-it-works/

Příloha **A**

Seznam použitých zkratek

PWA Progessive web application

HTML HyperText Markup Language

XML Extensible Markup Language

DOM Document Object Model

UI User Interface

JSON JavaScript Object Notation

SVG Scalable Vector Graphics

ČVUT České vysoké učení technické v Praze

UK Univerzita Karlova - Praha

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `user-testing` ..... testovací protokoly z testování použitelnosti
- `build` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF