|  |  |  |
| --- | --- | --- |
| Logo__SSPU_2016_Barva | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Nástěnka úkolů – frontend v React.js** | | |
| Martin Dzida | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2020/2021 | |

#### Poděkování

*Chtěl bych poděkovat Matěji Nešutovi za skvělou spolupráci na projektu, za jeho spolehlivost, vstřícnost a svědomitou práci. Také bych rád poděkoval panu učiteli Mgr. Markovi Lučnému za doporučení tématu projektu.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 3. 1. 2021

*podpis autora práce*

**ANOTACE**

Výsledkem projektu je frontend pro webovou aplikaci nástěnky úkolů s využitím JavaScriptové knihovny React, která zajišťuje interaktivitu UI. Na responzivitě a designu aplikace se podílí framework Bootstrap. Uživatelské rozhraní je složeno z několika částí, tzv. komponentů, které mohou být samy o sobě stránkami nebo pouze menšími funkčními prvky, z nichž se nakonec utvoří kompletní stránka. Komponenty jsou uspořádány v hierarchii – existuje jeden root komponent a na něj navazují další child komponenty (např. přihlašovací stránka). Základní architektura komponentů této aplikace se přímo odvíjí od databázového modelu. Komunikace s backend REST API probíhá v podobě XMLHttp requestů pomocí knihovny axios v0.21.1, přenášená data jsou ve formátu JSON. Ta jsou ukládána v  jednotlivých komponentech, protože veškerá logika je řešena samostatně. Podle URL adres, které spravuje React Router, jsou následně vykreslovány příslušné stránky.

**Klíčová slova:** frontend, uživatelské rozhraní, React, JavaScript, Redux

OBSAH

1. [OBSAH 4](#_Toc61212673)
2. [Úvod 5](#_Toc61212674)
3. [1 Využité technologie 6](#_Toc61212675)

[1.1 React 6](#_Toc61212676)

[1.2 Redux 7](#_Toc61212677)

[1.2.1 React Redux 8](#_Toc61212678)

[1.3 React Router 8](#_Toc61212679)

[1.4 Bootstrap 8](#_Toc61212680)

1. [2 součásti Frontendu 9](#_Toc61212681)

[2.1 Modální okna 9](#_Toc61212682)

[2.2 Navbar 11](#_Toc61212683)

[2.3 Hlavní stránka 11](#_Toc61212684)

[2.3.1 Tabule 11](#_Toc61212685)

[2.3.2 Sidebar 14](#_Toc61212686)

[2.3.3 Logy 15](#_Toc61212687)

[2.4 Skupiny 15](#_Toc61212688)

[2.5 Formuláře 17](#_Toc61212689)

[2.5.1 Nahrávání souborů 18](#_Toc61212690)

[2.6 Routing 19](#_Toc61212691)

[2.7 Implementace Redux 20](#_Toc61212692)

[2.7.1 Store 20](#_Toc61212693)

[2.7.2 Actions 21](#_Toc61212694)

[2.7.3 Reducers 22](#_Toc61212695)

[2.8 Komunikace s API 23](#_Toc61212696)

[2.8.1 Axios 23](#_Toc61212697)

1. [ZÁVĚR 24](#_Toc61212698)
2. [Seznam použitýCH INFORMAČNÍCH ZDROJů 25](#_Toc61212699)

Úvod

Nápad na vytvoření frontendu s využitím knihovny React mi představil pan učitel Mgr. Lučný. O této technologii jsem v tu chvíli vůbec netušil, avšak už při prvotním krátkém seznámení se mi zalíbila a získal jsem tu počáteční motivaci, která je pro učení nových věcí nezbytná.

Dalším důvodem, proč jsem si zvolil tento projekt, byla možnost spolupráce s Matějem Nešutou, s nímž se dobře znám. Nikdy jsem neměl příležitost se podílet na týmovém projektu, tudíž tohle byla pro mě zcela nová a myslím si, že velice cenná, zkušenost, kterou budu moct v budoucnu uplatnit.

Mé cíle byly ze začátku nejasné – teprve jsem objevoval základy Reactu, neměl jsem žádnou celistvou vizi finálního výsledku a k tomu bylo třeba, aby moje část projektu byla v symbióze s Matějovou částí. Poté mi Matěj uvedl jeho plán na nástěnku úkolů a od tohoto okamžiku jsem konečně dostal nějaký směr, kterého jsem se mohl držet. Chtěl jsem co nejvíce pracovat s Reactem, abych si ho osvojil, což se mi nakonec splnilo.

První část dokumentace představuje hlavní technologie, které jsem v projektu hojně používal, a jejich funkční principy, a také vysvětluje jejich využití. Ve druhé části jsou detailně představeny konkrétní části uživatelského rozhraní. Třetí kapitola se zaměřuje na komunikaci mezi frontend a backend aplikace, popisuje způsob propojení těchto částí.

# Využité technologie

**Vytvoření projektu**

K vytvoření projektu jsem použil balík create-react-app, který vytvoří předpřipravenou základní aplikaci pomocí pár příkazů.



Nevýhodou je překvapující velikost projektu, kterou zapříčiňují node-modules.[[1]](#footnote-1)

## React

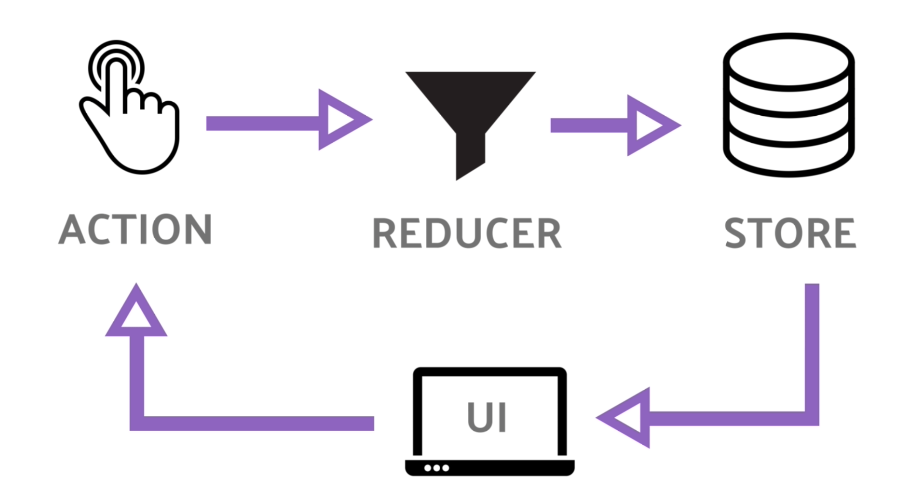
React je JavaScriptová knihovna pro tvorbu uživatelských rozhraní, vyvíjena společností Facebook a komunitou samostatných vývojářů a společností.

Základem každé React aplikace jsou samostatné komponenty, z nichž se následně sestaví komplexní uživatelské rozhraní. Komponent je většinou JavaScriptová třída (nebo funkce) a její součástí je state (= objekt, ve kterém jsou uloženy hodnoty pro daný komponent) a render metoda. Výstup render metody se jako React Element (= jednoduchý JavaScript objekt) nejdříve uloží v tzv. virtual DOM (VDOM) – synchronizovaná „kopie“ skutečného DOM[[2]](#footnote-2), uložená v paměti. Poté React porovná změny mezi virtual DOM a skutečným DOM a aktualizuje v DOM pouze ty části, jež byly pozměněny.

React umožňuje používat syntaxi JSX neboli JavaScript XML, která má velmi podobný vzhled jako HTML. JSX poskytuje způsob, jak strukturovat vykreslování komponentů pomocí syntaxe známé mnoha vývojářům.

## Redux

Redux je malá knihovna, která umožňuje JavaScript aplikacím uchovávat state (= objekt, ve kterém jsou uloženy hodnoty) na jednom místě. Toto místo se nazývá store. Dalšími hlavními pojmy jsou action a reducer. Action představuje nějakou definovanou událost – například kliknutí tlačítka. Při provedení této události se action pošle do reduceru. V reduceru se vyhodnotí, jaká action byla provedena, a podle toho se aktualizuje store.



*Obrázek*

Redux jsem se prvně rozhodl použít pouze pro ukládání autentifikačního tokenu, ale záhy jsem si řekl, že by měla mít tato knihovna v mém projektu větší uplatnění. Napadlo mě ukládat kromě tokenu také údaje o přihlášeném uživateli – například jméno, profilový obrázek a jiné. Díky tomu není třeba na téměř každé stránce posílat request, aby se získaly tyto údaje, protože se načtou do store hned po přihlášení.

### React Redux

React Redux slouží k propojení React komponentů s Redux store. Komponent Provider zpřístupňuje Redux store celé aplikaci a s pomocí funkce connect lze propojit určitý komponent se store.

## React Router

Jedná se o knihovnu pro deklarativní routing v React aplikacích. Routing probíhá v době, kdy se aplikace renderuje (dynamický routing). Kvůli tomu se téměř celá knihovna skládá z navigačních komponentů jako je třeba Link nebo Redirect.

Tuto knihovnu jsem použil, protože React je navržen pro tvorbu single-page aplikací, což ovšem naše aplikace není a k tomu bylo nutné spravovat URL adresy.

## Bootstrap

Bootstrap je open-source CSS framework, který je využíván k vytváření responzivních uživatelských rozhraní webových aplikací. Zmíněnou responzivitu (to znamená, že stránky jsou přizpůsobivé všem zařízením – od mobilních zařízení, až po stolní PC) opatřuje tzv.  grid systém. Jedná se o soustavu řádků a sloupců, která určuje rozložení obsahu podle zařízení, na kterém se stránky zobrazují.

Bootstrap obsahuje designové HTML a CSS šablony pro typografii, tlačítka, formuláře, tabulky a další frontend prvky. Součástí tohoto frameworku jsou i volitelné JavaScript pluginy. Je kompatibilní s nejnovějšími verzemi všech moderních prohlížečů (Chrome, Firefox, Opera, aj.).

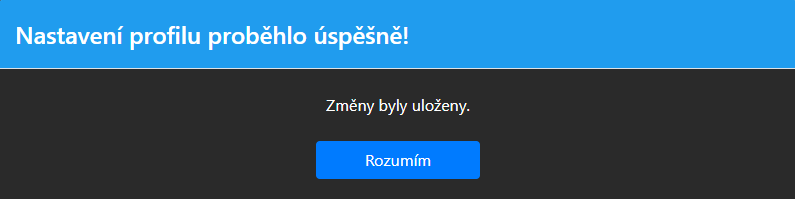
Už na začátku projektu jsem věděl, že využiji tohoto frameworku, protože s ním mám poměrně dost zkušeností a vím, jak s ním pracovat. V projektu používám Bootstrap verze 4.5.3.

# součásti Frontendu

## Modální okna

Jedná se o vyskakovací okno, které blokuje hlavní okno, dokud nebude obslouženo. Ve svém projektu používám modální okna za účelem předání informace, potvrzení uživatelské akce nebo získání uživatelského vstupu. Při stylizaci jsem uplatnil knihovnu React Bootstrap.



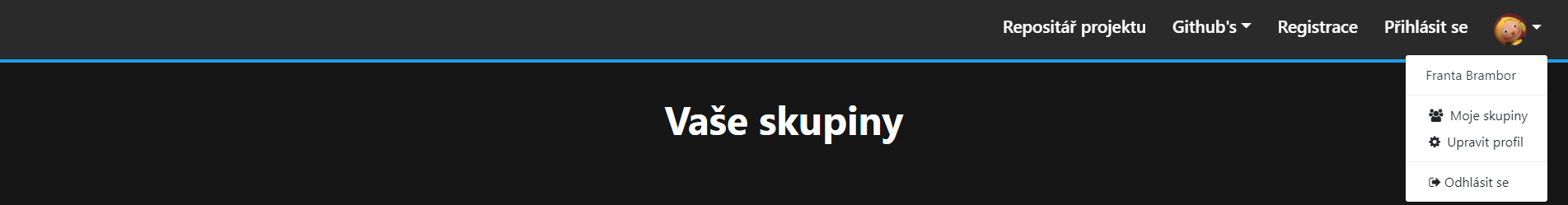


Některá modální okna jsou vyvolávána v child komponentech stránky, na které se zobrazují, tudíž jsou mnohem složitěji provedena, protože v Reactu se data posílají z parent do child komponentu. Například u modálního okna pro odebrání uživatele ze skupiny získávám data pomocí callback funkce, kterou musím posílat přes dva child komponenty, abych mohl poté zjistit, zda uživatel bude, nebo nebude odebrán ze skupiny.

****

## Navbar

Navbar je navigační lišta na vrcholu stránky, která pomáhá uživateli se orientovat v aplikaci. K vytvoření navbaru jsem použil Bootstrap šablonu, kterou jsem si postupně upravoval podle potřeby. Navbar obsahuje odkaz na repositář mého projektu, odkazy na můj a Matějův Github a navigační odkazy na přihlašovací a registrační stránku. Pokud je uživatel přihlášen, uvidí v navbaru jeho profilový obrázek, pod kterým se skrývá uživatelské menu.



## Hlavní stránka

Hlavní stránkou je myšlena stránka dané skupiny. Skládá se ze tří hlavních částí:

* tabule,
* sidebar,
* logy.

### Tabule

V horní části tabule je kalendářová lišta s nastavením roku a měsíce, ve spodní části je tabulka se všemi úkoly v daném měsíci a roce. Úkoly jsou rozřazeny do sloupců podle jejich fáze:

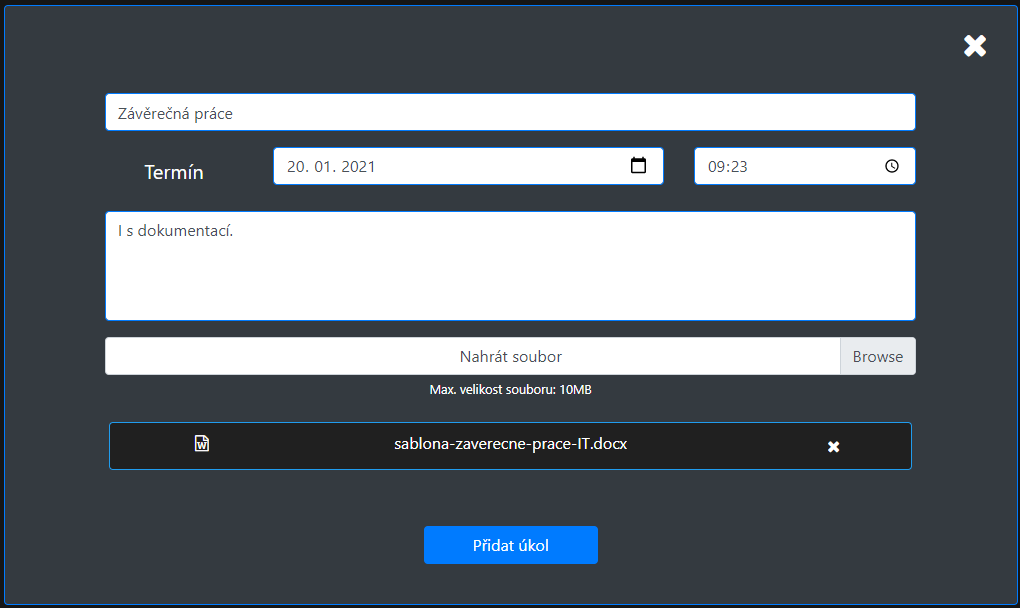
* neodstartováno,
* v průběhu,
* pozastaveno,
* dokončuje se,
* hotovo.

Každá buňka je samostatným komponentem a obsahuje název úkolu, autora a termín dokončení.





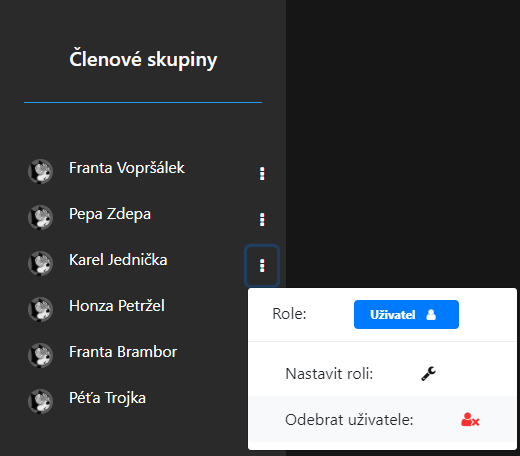
Pod tabulkou se nachází kromě tlačítka pro přidání uživatele do skupiny i tlačítko pro přidání úkolu. Po kliknutí se objeví formulář, ve kterém uživatel vyplní název, termín odevzdání a může i přidat popisek úkolu nebo nahrát vlastní soubory.



### Sidebar

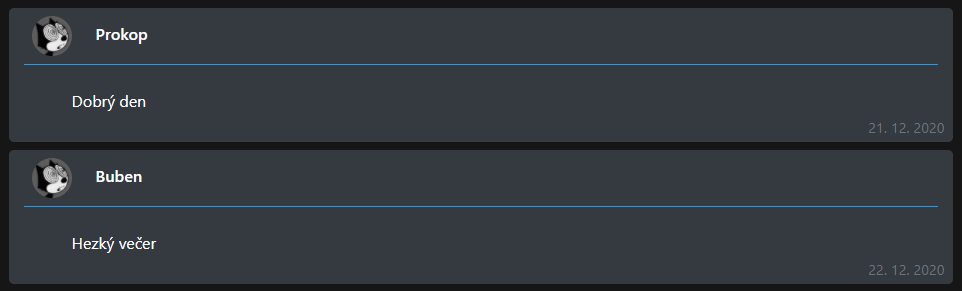
Jedná se o boční panel, v němž se vykreslují všichni členové skupiny s profilovými obrázky a rozbalovacím menu. Běžnému uživateli skupiny menu poslouží pouze k zjištění skupinové role jednotlivých členů, ale moderátor může nastavit roli běžným uživatelům nebo je přímo ze skupiny odebrat. Vlastník skupiny má oprávnění na obě operace u všech členů skupiny.

Obě zmíněné akce vyvolávají modální okno - pro potvrzení odebrání uživatele nebo pro výběr role.



### Logy

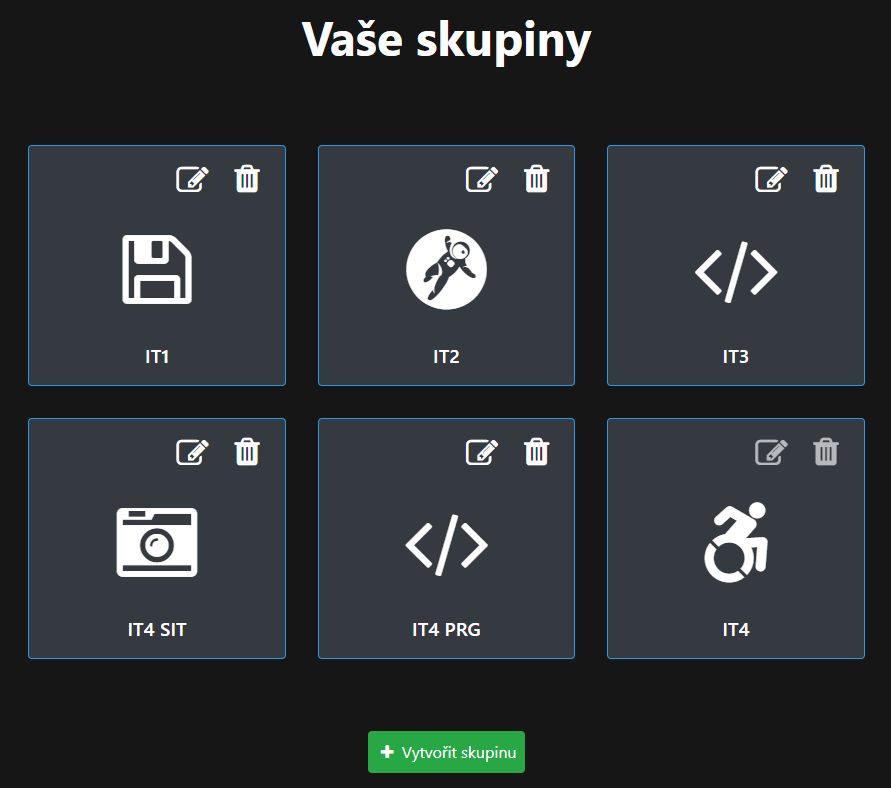
Logy jsou malé segmenty, jejichž účelem je oznámit přidání nového úkolu. Jsou umístěny na konci hlavní stránky. Skládají se ze záhlaví, ve kterém je profilový obrázek a jméno autora, a z těla, kde je zobrazena zpráva o přidání úkolu. V pravém dolním rohu se nachází datum přidání.



## Skupiny

Tato část pojednává o stránce, na níž jsou zobrazeny všechny skupiny, kterých je uživatel členem. Skupina je zastoupena kartou s názvem a ikonou skupiny[[3]](#footnote-3). Chtěl jsem namísto ikon použít obrázky, ale to nebylo v tomto případě možné.

V pravém rohu karty jsou tlačítka pro upravení názvu skupiny a odstranění skupiny. Obě tlačítka jsou funkční jedině v případě, že je přihlášený uživatel vlastníkem dané skupiny. Po kliknutí je vyvoláno modální okno – buď pro potvrzení odstranění skupiny, nebo pro zadání nového názvu skupiny.



Pod seznamem karet se nachází tlačítko pro vytvoření nové skupiny. Při kliknutí se objeví formulář, kde uživatel vyplní název, vybere si ikonu ze seznamu. Může přidat popisek skupiny. Poslední částí formuláře je přidání uživatelů do nové skupiny. Tato část se skládá z pole vyhledávání. Uživatel napíše do pole jméno uživatele, kterého chce přidat do skupiny, klikne na tlačítko s ikonou lupy a následně se mu zobrazí tabulka s výsledkem vyhledávání. Poté už jen stačí kliknout na tlačítko u hledaného profilu.

## Formuláře

Formuláře jsou významnou součástí tohoto uživatelského rozhraní. Vyskytují se na mnoha stránkách a plní několik funkcí – například přihlášení, registraci nebo nastavení profilu. Dokonce se objevují i v některých modálních oknech. Pro elegantnější vzhled jsem při tvorbě formulářů využíval Bootstrap šablon.

Formulář je složen ze vstupů. U každého vstupu je nutné, aby byla ošetřena změna hodnoty – onChange event. K tomu je potřeba vytvořit metodu, která bude tento event obstarávat. Tato metoda se nazývá event handler.





### Nahrávání souborů

Po dlouhém řešení tohoto problému jsem našel dva způsoby nahrávání souborů:

* jako lokální odkaz,
* jako base64 řetězec.

Ačkoliv se mi nejdříve jevilo první řešení jako to lepší, téměř hned jsem přišel na to, že  jediné vhodné řešení je to druhé.

Proces nahrání souboru začíná tím, že uživatel klikne na vstup typu „file“ a vybere soubor. Tento vstup je ošetřen samostatnou metodou. Soubor se získává z event.target.files, kam se po vybrání automaticky uloží jako objekt. Objekt obsahuje užitečné informace – například název souboru, typ souboru, pomocí kterého lze zkontrolovat, jestli uživatel nenahrál soubor nepovoleného formátu, nebo také velikost souboru, kterou můžeme chtít omezit.

Získání soubor z event.target.files nám sice poskytne informace o souboru, avšak samotný soubor takto uložit nelze. K tomu slouží FileReader[[4]](#footnote-4). Po úspěšném načtení je zavolána metoda onload, při které se do state uloží obsah a název souboru.



## Routing

Routing ve svém projektu řeším s pomocí knihovny React Router v5.2.0. Aby mohl routing v aplikaci fungovat, musí být celý obsah App komponentu „zabalen“ v Routeru komponentu. Jednotlivé Route komponenty zastupují URL adresy, které jsou definovány parametrem path. Paramentr component určuje, jaká stránka se bude na dané URL adrese vykreslovat.



Pohyb mezi stránkami zajišťují Link komponenty, které jsou v podstatě klasické <a> odkazy.



## Implementace Redux

Implementace v mém projektu je poměrně jednoduchá, ale jelikož to byl pro mě zcela nový koncept, tak jsem měl se zprovozněním velké potíže.

### Store

Store slouží jako globální state aplikace. To znamená, že na data ze store mohou používat všechny komponenty na rozdíl od běžného state, který se váže jen na svůj komponent. V aplikaci by měl být pouze jeden store. Aby mohly mít všechny komponenty přísup ke store, musí se root komponent aplikace „zabalit“ do Provider komponentu, jehož parametrem je samotný store. Komponenty se ke store připojují metodou connect.



Je vytvořen pomocí metody createStore, do které je poslán reducer. Pokud je použito více reducerů, musí se spojit do jednoho rootReduceru metodou combineReducers.





### Actions

Action popisuje nějakou událost, která se v aplikaci stala. Ve skutečnosti je to ale jenom jednoduchý JavaScriptový objekt s atributem type, který určuje, o jakou událost se jedná. Atribut payload je využíván pro přidání dodatečných informací, v tomto případě objektu s informacemi o uživateli.



### Reducers

Reducer je funkce, která se na základě action rozhodne (pokud je to třeba), jak zaktualizuje store. Jako parametry přijímá action a současný state, po vyhodnocení vrátí nový aktualizovaný state.

Reducer by se dal přirovnat k event handleru, který je podle typu eventu (action) danou událost ošetří.



## Komunikace s API

Z frontednu jsou na endpointy[[5]](#footnote-5) posílány XMLhttp requesty a následně jsou z nich zpět odeslány odpovědi. Pokud byl request úspěšný, odpověď bude obsahovat data ve formátu JSON, jinak bude poslána chybová hláška. Kromě toho je v téměř každé odpovědi posílán status requestu. Toto je v podstatě princip komunikace frontendu s backendem.

Je možné posílat requesty na zhruba 20 endpointů. Kromě autentifikačních endpointů je nutné posílat v hlavičce requestu token. Ten je získán z odpovědi na POST request, který je odeslán při přihlášení uživatele, a následně uložen v Redux store.



### Axios

Jedná se o malou JavaScriptovou HTTP klient knihovnu – slouží k posílání HTTP requestů. Automaticky přeměňuje data do formátu JSON, čehož jsem i v mé aplikaci využil. Axios také umožňuje requesty zrušit nebo i zachytit.

# **ZÁVĚR**

Cílem projektu bylo vytvořit interaktivní a responzivní uživatelské rozhraní s využitím technologie React a seznámit se nejen s touto technologií, ale také celkově s problematikou tvorby frontendu webových aplikací.

Většina cílů byla splněna, ale i přesto je na projektu hodně prostoru pro doladění. Vzhledu uživatelského rozhraní by určitě prospěly komplexnější grafické prvky. Dále bych rád zapracoval na mobilním zobrazení. Ačkoliv jsou stránky responzivní, některé komponenty se na mobilních zařízeních mohou vykreslovat nesprávně, například kvůli dlouhým názvům.

V budoucnu bych také mohl optimalizovat kód, až budu mít s tvorbou React aplikací větší zkušenosti, protože by se některé problémy daly zajisté řešit efektivněji a elegantněji.

Seznam použitýCH INFORMAČNÍCH ZDROJů

(Citace budou přidány)

* musí zahrnovat všechny prameny, knihy, internetové odkazy a další studijní podklady, z nichž jsme čerpali;
* kapitola se nečísluje a zde končí číslování stránek práce;
* jednotlivé publikace se uvádějí v abecedním pořadí podle příjmení autorů a iniciál jeho jména, který se píše za čárkou;
* příjmení autora se píše velkými písmeny;
* název publikace se zvýrazňuje kurzívou;
* jestliže jsou uvedeni více než tři autoři, je možné vypsat hlavního autora s poznámkou „a kol.“(a kolektiv

1. Je nutné mít nainstalovaný node.js v10 nebo novější. [↑](#footnote-ref-1)
2. Document Object Model [↑](#footnote-ref-2)
3. Ikony Font Awesome v4.7 [↑](#footnote-ref-3)
4. Objket, který možňuje webovým aplikacím asynchronně číst obsah nahranných souborů [↑](#footnote-ref-4)
5. Zjednodušeně backend URL adresy [↑](#footnote-ref-5)