Technická dokumentácia pre webovú aplikáciu Library Management System

Obsah

1. Úvod a popis aplikácie

Webová aplikácia Library Management System je určená na automatizáciu procesov v knižnici. Zabezpečuje evidenciu titulov, členov a zápožičiek.
Používatelia, ktorím je táto aplikácia určená, sú zamestnanci knižnice.

2. Použité technológie a ich výhody

- **HTML, CSS, JavaScript:** Základné jazyky pre vytvorenie štruktúry, dizajnu a dynamických prvkov na strane klienta.
- **TypeScript:** Nadstavba nad JavaScriptom pre lepšiu typovú kontrolu a prehľadnosť kódu.
- **Tailwind CSS:** Framework na efektívne vytváranie responzívneho dizajnu.
- **API:** Napojenie aplikácie na backend cez adresu https://student-fed1.metis.academy s využitím endpointov podľa API dokumentácie.

3. Návrh projektu

3.1 Technický návrh projektu

- **Štruktúra kódu:** Organizovaná do zložiek (pages, img, script, style), kde sú súbory rovnakého druhu.
  V koreňovom adresári sú umiestnenené konfiguračné súbory: package.json, package-lock.json, tailwind.config.js, tsconfig.json.
  node_modules - obsahuje všetky závislosti projektu nainštalované pomocou nástrojov ako npm.
  index.html - hlavná stránka aplikácie.

- **Responzivita**: Implementovaná pre zariadenia typu iPhone XR pomocou media queries a responzívnych prvkov.

3.2 Dôležité návrhové rozhodnutia

3.2.1 Rozhodnutie o použitých technológiách

Návrh: Použitie kombinácie HTML, CSS, JavaScript, TypeScript a Tailwind CSS.

Odôvodnenie: Tieto technológie ponúkajú robustné možnosti pre tvorbu interaktívnych a responzívnych webových aplikácií.
TypeScript pridáva statickú typovú kontrolu nad JavaScriptom, čo pomáha v prevencii chýb a zvyšuje prehľadnosť kódu. Pridaním typov je kód čitateľnejší a zrozumiteľnejší, čo uľahčuje údržbu a spoluprácu v tíme.
Tailwind ponúka veľké množstvo preddefinovaných tried, čo umožňuje rýchle a jednoduché vytváranie responzívneho dizajnu. Tailwind generuje štýly priamo na základe použitých tried, čo znamená, že konečné súbory majú malú veľkosť v porovnaní s inými CSS frameworkami a umožňuje jednoduché prispôsobenie dizajnu pomocou konfigurácie alebo vlastných štýlov.

3.2.2 Rozhodnutie o štruktúre kódu

Návrh: Organizácia kódu do zložiek (pages, img, script, style).

Odôvodnenie: Členenie kódu do zložiek zjednodušuje správu projektu, umožňuje lepšiu škálovateľnosť a zlepšuje prehľadnosť.

3.3 Možný spôsob zabezpečenia systému

3.3.1 Autentifikácia a autorizácia:

- **Prihlasovanie:** Zabezpečiť, aby každý zamestnanec mal unikátne meno a heslo. Heslá by mali byť silné a ukladané v bezpečnej podobe.
- **Dvojfaktorová autentifikácia (2FA):** Poskytnúť možnosť povinného alebo voliteľného použitia dvojfaktorovej autentifikácie pre dodatočnú vrstvu bezpečnosti.

3.3.2 Prístupové práva:

- **Role a oprávnenia:** Priradiť zamestnancom rôzne role (napríklad zamestnanec, manažér, administrátor) a priradiť oprávnenia na základe ich úloh. Nie každý zamestnanec by mal mať prístup ku všetkým častiam systému.
- **Oprávnenie na úrovni údajov:** Určiť, ktorí zamestnanci majú prístup k akým údajom (napr. citlivým údajom o používateľoch).

3.3.3 Monitorovanie aktivít:

- **Záznamy aktivít:** Zaznamenávať aktivity používateľov, aby sa mohlo sledovať neštandardné správanie alebo potenciálne hrozby.
- **Upozornenia:** V prípade podozrivých aktivít alebo neplatných prihlásení posielať upozornenia alebo blokovať účty.

3.3.4 Zabezpečenie komunikácie:

- **Šifrovanie:** Použiť HTTPS na šifrovanie komunikácie medzi klientom a serverom, aby sa zabránilo odpočúvaniu.
- **Bezpečné API:** Zabezpečiť API pomocou overovania tokenu, aby sa predišlo neoprávnenému prístupu.

3.3.5 Správa hesiel:

- **Politika hesiel:** Určiť politiku týkajúcu sa platnosti hesiel a vyžadovať od zamestnancov ich pravidelnú zmenu.
- **Zabudnuté heslo:** Poskytnúť bezpečný spôsob obnovenia zabudnutých hesiel, napríklad prostredníctvom overeného e-mailu alebo otázky na obnovenie.

3.3.6 Fyzická bezpečnosť:

- **Fyzický prístup k serverom:** Zabezpečiť fyzický prístup k serverom a databázam, aby sa predišlo neoprávnenému prístupu.

3.3.7 Aktualizácie a zraniteľnosti:

- **Pravidelné aktualizácie:** Udržiavať systém a všetky používané knižnice aktualizované, aby sa predišlo zraniteľnostiam.
- **Testovanie bezpečnosti:** Pravidelne vykonávať testy penetrácie na identifikáciu možných zraniteľností.

3.3.8 Zálohovanie:

- **Pravidelné zálohy:** Pravidelne zálohovať údaje a uistiť sa, že sú dostupné v prípade katastrofy alebo útoku ransomware.

4. Implementácia

4.1 Klúčové problémy a ich riešenia

4.1.1 Problém s funkcionalitiou vyhľadávania (Filter)

Problém: Implementácia filtrovania vyžadovala viac času a správnu logiku.
Riešenie: Vytvorenie funkcie filter.ts, ktorá umožňuje vyhľadávať zadané údaje iba v konkrétnom stĺpci tabuľky údajov.

4.1.2 Možné riešenia a alternatívy

4.1.2.1 Riešenie asynchrónnych operácií v JavaScripte

Alternatívy:

- Použitie Promise s Callbacks.
- Využitie async/await.
- Implementácia knižnice ako Axios pre lepšiu manipuláciu s asynchrónnymi volaniami.

Odôvodnenie voľby:

- Vybrali sme async/await, pretože poskytuje čistú syntax a jednoduchú manipuláciu s asynchrónnym kódom.

4.2 Členenie projektu

LMS aplikácia/
|-- img/
|-- node_modules/
|-- pages/
|-- script/
|-- style/
|-- index.html
|-- package.json
|-- package-lock.json
|-- tailwind.config.js
|-- tsconfig.json

- img/: Zložka pre obrázky.
- node_modules/: Zložka s inštalovanými závislosťami projektu.
- pages/: Priečinok pre html stránky aplikácie.
- script/: Priečinok pre JavaScript alebo TypeScript súbory.
- style/: Priečinok pre štýlovacie súbory (napr. CSS).
- index.html: Hlavná stránka aplikácie.
- package.json: Súbor s metadátami projektu a zoznamom závislostí.
- package-lock.json: Súbor, ktorý zabezpečuje konkrétnu verziu každej nainštalovanej závislosti.
- tailwind.config.js: Konfiguračný súbor pre Tailwind CSS.
- tsconfig.json: Konfiguračný súbor pre TypeScript.

Zložka Pages:

addBook.html - zobrazuje formulár na pridanie knihy do zápožicky

addDVD.html - zobrazuje formulár na pridanie DVD do zápožicky

addMember.html - zobrazuje formulár na pridanie člena do zoznamu členov

allMembers.html - zobrazuje zoznam všetkých členov

allRentals.html - zobrazuje zoznam všetkých zápožičiek

books.html - zobrazuje zoznam všetkých kníh

detailBook.html - zobrazuje stránku detailu knihy

detailDvd.html - zobrazuje stránku detailu DVD

detailMember.html - zobrazuje stránku detailu člena

dvd.html - zobrazuje stránku zoznamu DVD nosičov

guide.html - zobrazuje stránku prehľadnej navigácie

notifications.html - zobrazuje stránku všetkých upozornení, ktoré sa v aplikácii generujú po úkonoch akými su pridanie člena, pridanie zápožičky, vrátenie zápožičky, penalizácia po nevrátení zápožičky na čas

queueOfRequests.html - zobrazuje stránku titulov o ktoré majú členovia záujem a aktuálne nie sú žiadne voľne dostupné kópie titulov na požičanie

rentalsPastDue.html - zobrazuje stránku s titulmi po prekročení výpožičnej doby

rentTitle.html - zobrazuje stránku pre zapožičanie titulu

Zložka Style:

common.css - štýl pre jednotné zobrazenie stránok v prehliadači

forms.css - štýly pre formuláre

guide.css - štýly pre navigáciu stránky Guide.html

logo.css - štýl pre logo

menu.css - štýl pre navigačné menu

style.css - nastavenia pre použitie frameworku tailwindcss

style-compiled.css - prekompilovaný štýl frameworku tailwindcss

tables.css - štýl pre všetky tabuľky na stránkach

Zložka Script obsahuje súbory Javascriptu a Typescriptu

addBook.js - súbor s funkcionalitou na pridanie knihy

addDvd.js - súbor s funkcionalitou na pridanie Dvd

addMember.js - súbor s funkcionalitou na pridanie člena

books.js - súbor s funkcionalitou na zobrazenie kníh

detailBook.js - súbor s funkcionalitou na zobrazenie detailu knihy

detailDvd.js - súbor s funkcionalitou na zobrazenie detailu Dvd

detailMember.js - súbor s funkcionalitou na zobrazenie detailu člena

dvd.js - súbor s funkcionalitou na zobrazenie dvd titulov

filter.js - prekompilovaný filter.ts na javascript súbor

filter.ts - súbor s funkcionalitou na vyhľadávanie v aplikácii

members.js - súbor s funkcionalitou na zobrazenie zoznamu členov

notifications.js - súbor s funkcionalitou na zobrazenie zoznamu upozornení

queueOfRequests.js - súbor s funkcionalitou na zobrazenie už zapožičaných zapožičiek, o ktoré majú ďalší členovia záujem

rentals.js - prekompilovaný súbor rentals.ts v typescripte

rentals.ts - súbor s funkcionalitou na zobrazenie zapožičaných dvd a kníh

rentalsPastDue.js - súbor s funkcionalitou na zobrazenie titulov po prekročení zápožičnej doby

rentTitle.js - súbor s funkcionalitou na urobenie zápožičky titulov

Zložka img:

book.png - obrázok knihy

dvd.png - obrázok DVD

libraryBackground.jpg - obrázok pozadia

member.png - obrázok ikony člena

4.3. Kľúčové funkcie

4.3.1 Funkcia na asynchrónne odoslanie dát na server

```
post(url, data)
```

Táto funkcia slúži na odoslanie dát na server pomocou HTTP POST requestu. Využíva Fetch API pre asynchrónne spracovanie requestov.

Parametre

- `url` (string): URL adresa servera, kam sa majú dáta odoslať.
- `data` (Object): Dáta, ktoré sa majú odoslať na server vo formáte JSON.

Výnimky

- V prípade neúspešného Fetch requestu alebo spracovania odpovede, funkcia vyhodí chybový objekt.

Návratová hodnota

- Promise reprezentujúci stav odosielania dát. V prípade úspechu môže vrátiť údaje z odpovede servera.

4.3.2 Funkcia na asynchrónne načítanie dát z danej URL adresy

```
fetchData(url)
```

Táto funkcia slúži na asynchrónne načítanie dát z daného URL pomocou Fetch API. Využíva asynchrónne programovanie na spracovanie Fetch requestu a parsovanie odpovede vo formáte JSON.

Parametre

- `url` (string): URL adresa, odkiaľ sa majú dáta získať.

Výnimky

- V prípade zlyhania Fetch requestu alebo spracovania odpovede, funkcia vyhodí chybový objekt. Chybové hlásenie sa zobrazí pomocou vyskakovacieho okna s upozornením.

Návratová hodnota

- Promise, ktorý sa vyrieši na dáta z odpovede vo formáte JSON. V prípade chyby vráti prázdne pole.

4.3.3 Funkcia na asynchrónne odosielanie aktualizovaných dát

`updateData()`

Táto funkcia slúži na aktualizáciu existujúcich dát prostredníctvom HTTP PUT requestu.
Parametre

- Žiadne parametre nie sú priamo vyžadované pre túto funkciu.

Lokálne Premenné

- `updateDataconst`**:** Objekt obsahujúci aktualizované dáta pre knihu, ktoré sa posielajú na server v tele požiadavky.

Výnimky

- Ak HTTP request vráti neúspešnú odpoveď, funkcia vyvolá chybu a vypíše informácie o neúspechu.

4.3.4 Funkcia na asynchrónne vymazanie dát zo servera

`deleteData(id)`

Táto asynchrónna funkcia slúži na vymazanie dát zo servera pomocou HTTP DELETE requestu. Používateľ je pred vymazaním informovaný o potrebe potvrdenia.
Parametre

- `id` (string): Identifikátor (ID) titulu, ktorý sa má vymazať zo servera.

Výnimky

- V prípade neúspešného Fetch requestu alebo spracovania odpovede, funkcia vyhodí chybový objekt. Chybové hlásenie sa zobrazí pomocou vyskakovacieho okna s upozornením.

Návratová hodnota

- Funkcia nevracia žiadnu hodnotu, pretože jej úlohou je vykonať operáciu vymazania.

4.3.5 Vytvorenie tabuľky s titulmi

`createTable(books)`

Táto asynchrónna funkcia slúži na dynamické vytvorenie tabuľky s informáciami o tituloch. Ak neexistujú žiadne dáta na zobrazenie, zobrazí sa správa o ich absencii.

Parametre

- `books` (array): Pole objektov obsahujúcich informácie o tituloch, ktoré majú byť zobrazené v tabuľke.

Výnimky

- Žiadne explicitné výnimky nie sú definované v tejto funkcii.

Návratová hodnota

- Funkcia nevracia žiadnu hodnotu.

4.3.6 Vytvorenie bunky v tabuľke s textovým obsahom

`createTd(value)`

Táto funkcia slúži na vytvorenie bunky v rámci tabuľky s textovým obsahom.

Parametre

- `value` (string): Textový obsah bunky.

Výnimky

- Žiadne explicitné výnimky nie sú definované v tejto funkcii.

Návratová hodnota

- Funkcia vracia vytvorený element bunky (`<td>`).

4.3.7 Vytvorenie ikony s odkazom v bunke tabuľky

`createIcon(id)`

Táto funkcia slúži na vytvorenie ikony s odkazom v bunke tabuľky.
Parametre

- `id` (string): Jedinečný identifikátor titulu, ktorý bude vložený do URL odkazu.

Výnimky

- Žiadne explicitné výnimky nie sú definované v tejto funkcii.

Návratová hodnota

- Funkcia vracia vytvorený element bunky (`<td>`) obsahujúci ikonu a odkaz.

4.3.8 Vytvorenie tlačidla pre mazanie titulu

`createButton(id)`

Táto funkcia slúži na vytvorenie tlačidla pre mazanie titulu v rámci tabuľky.
Parametre

- `id` (string): Jedinečný identifikátor titulu, ktorý bude prepojený s funkcionalitou mazania.

Výnimky

- Žiadne explicitné výnimky nie sú definované v tejto funkcii.

Návratová hodnota

- Funkcia vracia vytvorený element tlačidla (`<td>` obsahujúci `<button>`).

4.3.9 Vymazanie obsahu tabuľky

`clearTable()`

Táto funkcia slúži na vymazanie obsahu existujúcej tabuľky.

Parametre

- Žiadne parametre nie sú vyžadované pre túto funkciu.

Výnimky

- Žiadne explicitné výnimky nie sú definované v tejto funkcii.

Návratová hodnota

- Funkcia nemá explicitnú návratovú hodnotu.

5. Overenie riešenia

Aplikácia prechádza testovacím procesom pri ktorom sa vytvárajú testovacie prípady a scenáre pre manuálne testovanie.

5.1 Testovacie scenáre

Správa Titulov:

- Scenár: Používateľ pridá nový titul do systému.
- Očakávaný výsledok: Nový titul je pridaný a zobrazuje sa v systéme.
- Scenár: Používateľ aktualizuje informácie o existujúcom titule.
- Očakávaný výsledok: Údaje o titule sa úspešne aktualizujú v systéme.
- Scenár: Používateľ vymaže titul.
- Očakávaný výsledok: Titul je úspešne odstránený zo systému.

Správa Členov:

- Scenár: Používateľ pridá nového člena knižnice.
- Očakávaný výsledok: Nový člen je pridaný a zobrazuje sa v systéme.
- Scenár: Používateľ aktualizuje informácie o existujúcom člene.
- Očakávaný výsledok: Informácie o člene sa úspešne aktualizujú v systéme.
- Scenár: Používateľ vymaže člena.
- Očakávaný výsledok: Člen je úspešne odstránený zo systému.

Správa Pôžičiek:

- Scenár: Používateľ vytvorí novú zápôžičku.
- Očakávaný výsledok: Nová zápožička je vytvorená a evidovaná v systéme.
- Scenár: Používateľ predĺži zápožičku.
- Očakávaný výsledok: Zápožička je úspešne predĺžená, aktualizuje sa čas na vrátenie zápožičky a počet predĺžení.
- Scenár: Používateľ vráti knihu po skončení pôžičky.
- Očakávaný výsledok: Kniha je úspešne vrátená a aktualizuje sa dostupnosť v systéme.

5.2 Pokrytie testami

Test Case ID: 1
Test Case: Pridanie Knihy
Description: Overenie pridania knihy do aplikácie.
Precondition: Otvorená stránka addBook.html aplikácie LMS

Step 1: Vyplňte Title: "Testovacia Kniha"
Step 2: Vyplňte Author: "Autor Knihy"
Step 3: Vyplňte ISBN: “123456789”
Step 4: Vyplňte number of pages: “300”
Step 5: Vyplňte Available Copies: “5”
Step 6: Vyplňte Total Available Copies: “10”
Step 7: Kliknite na tlačidlo "Pridať Knihu".
Expected result: Vyskočí alert s hláškou "The title with these data has been entered into the database."
Actual result: Vyskočil alert s hláškou "The title with these data has been entered into the database."
Status: Pass

Test Case ID: 2
Test Case: Odstránenie Knihy
Description: Overenie vymazania knihy z aplikácie.
Precondition: Otvorená stránka books.html

Step 1: Kliknite v tabuľke kníh pri knihe s názvom “Testovacia Kniha” na tlačidlo Delete.
Step 2: Vyskočí potvrdzujúce okno na potvrdenie vymazania knihy. Povrďte vymazanie knihy stlačneím OK.
Expected result: Vyskočenie okna Deleted successfully. Po stlačení OK sa stránka presmeruje na zoznam kníh.
Actual result: Vyskočenie okna Deleted successfully. Po stlačení OK sa stránka presmerovala na zoznam kníh.
Status: Pass