Документација

за проектот Book Recomender

предмет: Напреден Веб Дизајн

Учесник: Дарко Диванисов 201023

Професор: Јоксимоски Бобан **Асистент: Додевска Мила**

Вовед

Проектот **Book Recommender** е веб-базирана апликација дизајнирана да го подобри искуството на откривање нови книги за корисниците. Оваа веб апликација им овозможува на корисниците да разгледаат разновидна колекција на книги, да ги зачуваат омилените книги и да ги изрази своите преференции преку означување на омилени жанрови. Апликацијата исто така вклучува основни функционалности за менаџирање со корисници, како што се регистрација и најава, обезбедувајќи персонализирано искуство за секој корисник.

Една од клучните карактеристики на овој проект е системот за препораки, кој ја користи LLaMA API. По избирање на омилени книги и жанрови, системот за препораки предлага нови книги прилагодени на интересите на корисникот.

Користени Технологии

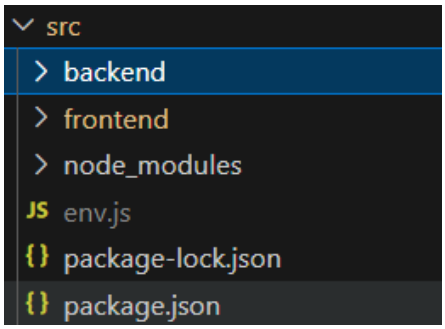
Проектот **Book Recommender** го користи технолошки стек MEBH . Еве преглед на технологиите и клучни пакети што се користат:

- **Frontend**: Фронт-ендот е развиен со **Vue.js.** Фронтендот е креиран користејќи го **vue-cli**. За подобрување на функционалноста, се користат неколку важни пакети:
    - **Axios**: Користен за правење HTTP барања до бекенд API-то.
    - **Bootstrap**: Вклучен за стилизирање на компоенттите.
    - **Vuex**: Користен за управување со состојбата.
    - **Vuex-persistedstate:** За чување на состојбата и по рефреш.
- **Backend**: Бекендот е базиран на **Express** во **Node.js**. Клучните пакети што се користат вклучуваат:
    - **Express**: Основен фрејмворк кој се користи за градење RESTful API и управување со рутирање.
    - **CORS**: Конфигуриран за ракување со Cross-Origin Resource Sharing, овозможувајќи ефикасна комуникација помеѓу фронт-ендот и бекендот.
    - **Joi**: Користен за валидација на податоците, обезбедувајќи дека податоците испратени до серверот ги исполнуваат одредени критериуми пред да бидат обработени.
    - **node-cache**: Имплементиран за кеширање на често пристапувани податоци, подобрувајќи ја перформансата и намалувајќи го оптоварувањето на базата на податоци.
    - **jsonwebtoken**: Користен за ракување со автентикација и авторизација преку JSON Web Tokens, користен е за создавање токен за најавените корисници.
    - **groq-sdk**: Овој SDK е користен за интеграција со LLaMA API-то, овозможувајќи ефективна обработка и управување со препораките базирани на интересите на корисниците.
- **База на податоци**: Податоците на апликацијата се складираат во **MongoDB** користејќи ја услугата **MongoDB Atlas**.
- **Конкурентно стартување**: За истовремено стартување на фронт-ендот и бекендот, се користи пакетот **concurrently**. Овој пакет овозможува паралелно стартување на повеќе процеси.
- **Околина на работа**: За околина на работа е искористен VScodium, а за складирање на проектот GitHub.

Структура на Проектот

Проектот е структуpiran во три главни делови:

- backend: содржи серверски код и логика.
- frontend: содржи клиентски код и кориснички интерфејс.
- env.js: содржи конекцски стринг до базата, Llama Api...
- Package.json: конфигурација за **concurently** која стартува и клиентскиот и серверскиот дел истовремено.



сл. 1 Надворешна структура

Backend структура

1. package.json: Ги содржи сите пакети потребни на backend-от

2. server/

a. controlers/

- **book.js**: Содржи функции за управување со книги.
  - **showBooks(req, res)**: Прифаќа податоци за книги со пагинација(бара по 15 книги) од базата на податоци. Користи кеширање за подобрување на перформансите.
  - **showBook(req, res)**: Презема детали за една книга според нејзиниот ID.
  - **updateBook(req, res)**: Ажурира бројот на допаѓања на книга и ја невалидира кеш-спремената информација за таа книга.
  - **addComment(req, res)**: Додава коментар на книга.

        - **removeComment(req, res)**: Брише коментар од книга.
    - **bookmarks.js**: Управува со омилените книги на корисниците.
        - **showBookmarks(req, res)**: Презема список на омилени книги на корисникот со пагинација(по 15 книги при еднонбарање) и нивните детали.
        - **showBookmark(req, res)**: Проверува дали конкретна книга е омилена од страна на корисникот.
        - **createBookmark(req, res)**: Додава нова омилена книга за корисникот.
        - **deleteBokmark(req, res)**: Брише омилена книга за корисникот.
    - **genres.js**: Управува со жанровските преференции на корисниците.
        - **getGenres(req, res)**: Презема омилените жанрови на корисникот.
        - **addLikedGenres(req, res)**: Ажурира омилените жанрови на корисникот.
    - **login.js**: Управува со логирање на корисниците.
        - **login(req, res)**: Автентицира корисник врз основа на е-пошта и лозинка. Генерира JWT токен ако е успешен.
    - **register.js**: Управува со регистрација на корисници.
        - **register(req, res)**: Регистрира нов корисник и издава JWT токен.
    - **recomendation.js**: Обезбедува препораки за книги користејќи Groq API.
        - **recomend(req, res)**: Испраќа барање до Groq API за препораки за книги врз основа на омилените книги и жанрови на корисникот. Анализира и форматира одговорот.

b. **cache.js**

- Обезбедува функции за управување со кешираните податоци за книги.
    - **getCacheData(cache, page, limit, genre)**: Презема кеширани податоци за одредена страница, лимит и жанр.
    - **setCacheData(cache, page, limit, data, genre)**: Складира податоци во кешот за одредена страница, лимит и жанр.
    - **deleteCacheData(cache, page, limit, genre)**: Брише кеширани податоци за одредена страница, лимит и жанр.

c. **mongodbconnection.js**

- Управува со поврзување со MongoDB.
    - **connect(collection)**: Поврзува со одредена MongoDB колекција.

d. **token.js**

- Управува со создавање на JWT токени.
    - **jwtTokenCreate(user)**: Генерира JWT токен за даден корисник со рок на траење од една недела.

e. **validation.js**

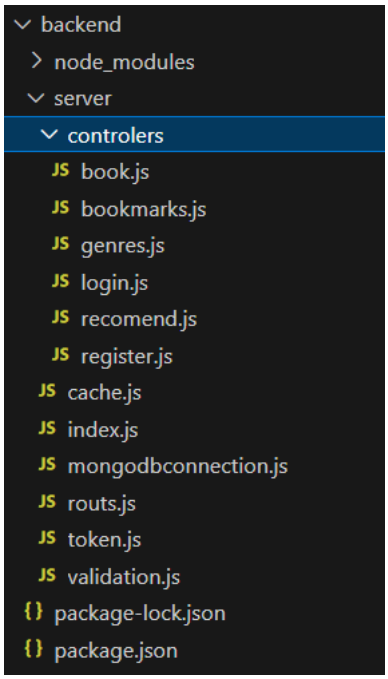
- Валидација на внесот на корисникот за регистрација и логирање.
    - **register(req, res, next)**: Валидација на податоци за регистрација (корисничко име, е-пошта, лозинка) со помош на **joi**.
    - **login(req, res, next)**: Валидација на податоци за логирање (е-пошта, лозинка) со помош на **joi**.

f. **routes.js**

- Дефинира патеките на апликацијата и ги асоцира со контролерските функции.

g. **index.js**

- Дефинира основа на апликацијата.



сл. 2 Структура на backend

Frontend структура

1. package.json: Ги содржи сите пакети потребни на frontend-от

2. public/ - Колекција од слики, иконки потребни за frontend-от

3. services/

- **services.js**: Колекција од функции, секоја прави различен повик до backend-от.

4. src/

a. components/

- **BooksShow.vue**: Компонента за приказ на низа од книги.
- **ComentsShow.vue**: Компонента за приказ на коментарите на една книга.
- **FilterTab.vue**: Компонента за приказ на таб кои е филтер за жанри.
- **PrevPageNext.vue**: Компонента за приказ на контроли за поместување на следна или претходна страна и броеви на страници

b. **router/index.js** : Содржи рутирање на сите views на frontend-от

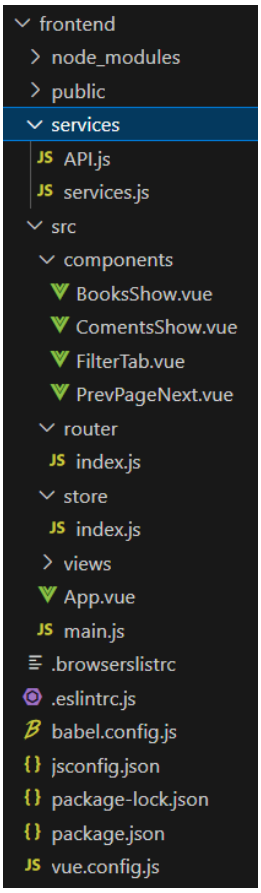
c. **store/index.js :** Ја содржи состојбата на апликацијата.

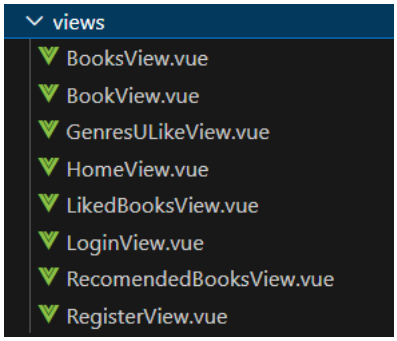
d. **views/**

- **BooksView.vue**: Приказ на сите книги заедно со филтер компонентата и компонентата за поместување на страници
- **BookView.vue**: Приказ на една книга заедно со компонентата за коментар на таа книга.
- **GenresULikeView.vue**: Приказ на жанри кои може да се селектираат.
- **HomeView.vue**: Домашната страница каде стојат линковите за избирање на омилен жанр и линкот за рекомендирање на книги.
- **LikedBooksView.vue**: Приказ на сите боокмаркнати книги.
- **LoginView.vue**: Приказ на форма за логирање на корисник.
- **RegisterView.vue**: Приказ на форма за регистрирање на корисник.
- **RecomendedBooksView.vue**: Приказ на книги кои се препорачани со Llama API.

e. **App.vue**: Овде има side bar во кој стојат линковите до другите погледи и исто така во main делот се вградуваат останатите погледи во зависност од кој линк е претиснат.

f. **main.js:** Формирање и дефинирање на апликацијата.





сл. 3.1 Views делот од frontend

сл. 3 Структура на frontend

Функционалности

1. Корисничка Автентикација и Управување:

Најава и Одјава:

- **Најава:**
    - **Функционалност:** Корисниците треба да се најават за да добијат пристап до функции кои се ексклузивни за автентицирани корисници. Дали најавата ќе биде успешна зависи од тоа дали ќе го помине валидацискиот процес кои се реализира со помош на **joi**.

    - **Ажурирање на Vuex складиштето:** Складиштето се ажурира со детали за корисникот и автентикациски токен. Овие информации се користат за извршување на автентицирани за промена на рута низ веб страницата и одржување на статусот на најавата низ целата апликација.
    - **Интеграција со Бекендот:** Бекендот ги валидира корисничките податоци, генерира токен и враќа информации за корисникот.
- **Одјава:**
    - **Функционалност:** Одјавувањето го чисти податоците на сесијата на корисникот, вклучувајќи го токенот и информациите за корисникот складирани во Vuex складиштето. Оваа акција ја ресетира состојбата на апликацијата на нејзините почетни вредности, осигурувајќи дека не остануваат чувствителни податоци достапни.
    - **Ресетирање на Vuex складиштето:** По одјавувањето, складиштето се ресетира на автентикациската состојба, вклучувајќи го податоците за корисникот и поставките за моментната страница/жанр. Апликацијата потоа ги пренасочува корисниците на домашната страница или страницата за најава.

Регистрација:

- **Регистрација:**
    - **Функционалност:** Новите корисници можат да создадат сметка со обезбедување на потребни информации, како што се корисничко име, е-пошта и лозинка. Овие информации се испраќаат до бекендот за обработка, и по успешна регистрација, корисниците автоматски е најавен и може да пристапи до системот.
    - **Интеграција со Бекендот:** Бекендот ги обработува деталите за регистрација, создава нов кориснички запис и може да испрати потврда или порака за успех. Овој процес обично вклучува валидација (joi) на внесените податоци и безбедно чување на лозинките на корисниците.

Автентикација Потребна:

- **Чувари на рутите:**
    - **Функционалност:** Некои рути, како што се оние за приказ на книги, омилени жанрови и препораки, бараат корисниците да бидат најавени. Чуварите на рутите се имплементираат за да проверуваат статусот на автентикацијата пред да дозволат пристап до овие патеки.

сл.4 чувар на рута

сл.5 логиќа за автентикација

2. Управување со Книги:

Прегледување на Книги:

- **Список на Книги:**
    - **Функционалност:** Корисниците можат да прегледуваат страница со книги која вклучува основни детали како наслов, автор и слика од корицата. Овој список се прикажува на кориснички пријателски начин, овозможувајќи на корисниците лесно да прелистуваат достапни книги. Содржи филтер на жанри и исто така контролер на страници (бидејќи на едно барање од базата добива само 15 книги) со кој може да ги прикаже наредните 15.
    - **Интеграција со Бекендот:** Бекендот обезбедува страница со книги базирана на побараната страница и жанр. Ова вклучува пребарување на базата на податоци и испраќање одговор со релевантни податоци за книги.
- **Детали за Книгата:**
    - **Функционалност:** Корисниците можат да прегледуваат детални информации за специфична книга преку навигација до нејзината уникатна ID патека. Овој детален приказ вклучува сеопфатен опис, авторот, детали за таговите, опција за боокмарк и опција за оставање на коментар.

- **Интеграција со Бекендот:** Бекендот го повлекува детализираните информации за специфична книга од базата на податоци и ги враќа на фронтендот. Овие податоци се користат за пополнување на приказот на деталите за книгата, исто така се прават и други соодветни повици до бекендот во зависност од тоа дали се боокмаркнува книга или пак се остава коментар за таа книга.

Боокмарк:

- **Преглед на Боокмарк:**
  - **Функционалност:** Автентицираните корисници можат да ја прегледуваат листата на книги што ги имаат зачувано(боокмаркнато). Оваа функција им овозможува на корисниците брзо да пристапат до книгите што ги имаат зачувано и исто така се искористува при препорака на книги.
  - **Интеграција со Бекендот:** Бекендот обезбедува крајна точка за повлекување на листата на зачувани книги за специфичен корисник, базирано на неговите зачувани преференции.

Коментари:

- **Преглед и Додавање Коментари:**
  - **Функционалност:** Најаваените корисници можат да прегледуваат и додаваат коментари на деталите за книгата. Оваа функција им овозможува на корисниците да ги споделат своите мислења и повратни информации за книгите, кои се видливи за други корисници.
  - **Интеграција со Бекендот:** Бекендот обезбедува крајни точки за повлекување и додавање коментари поврзани со специфични книги. Ова вклучува чување на нови коментари во базата на податоци и повлекување на постоечките коментари за приказ.

3. Кориснички Лајкови на Жанрови:

Жанрови што Корисникот Ги Лајкнал:

- **Преглед и Управување со Лајкувани Жанрови:**
  - **Функционалност:** Корисниците можат да прегледуваат и ажурираат свои лајкувани жанрови, што влијае на препораките што ги добиваат. Оваа функција им овозможува на корисниците да ја прилагодат својата искуствена основа според своите преференции и лајкови на жанрови.
  - **Интеграција со Бекендот:** Бекендот обезбедува крајна точка за повлекување и ажурирање на списокот на лајкувани жанрови за специфичен корисник. Ова вклучува додавање или отстранување на

жанрови во/од омилени на корисникот и ажурирање на нивните преференции во базата на податоци.

4. Преференции на Корисници:

Препораки за Книги:

- **Персонализирани Препораки:**
  - **Функционалност:** Системот генерира препораки за книги за автентицирани корисници базирани на нивните претходно зачувани книги и омилени жанрови. Оваа функција има за цел да им понуди на корисниците релевантни предлози за книги прилагодени на нивните читачки преференции.
  - **Интеграција со LLaMA API:** LLaMA API се користи за генерирање на персонализирани препораки преку анализа на преференциите на корисникот и историјата на читање. Ова вклучува испраќање на податоци за корисниците до API-то и примање на список на препорачани книги. Односно подетално се испраќа листа од боокмаркс на корисникор и зачувани жанри, на LlaMA моделот и prompt кој бара слични книги на боокмаркнатите и да бидат во областа на испратените жанри.
  - **Интеграција со Бекендот:** Бекендот податоците на корисниците (жанрите и боокмаркнатите книги) ги испраќа до LLaMA API, потоа добива листа од книги со име на книгата автор и крата дескрипција. Тие информации се враќаат на фронтендот.
  - **Назад на фронтендот**: На фронтендот се прави повик до GoogleBooks API за побарување на слика за секоја од книгите, доколку неможе да се добие слика се става дефултна слика. Доколку има некаков проблем при побарувањето на книгите од бекендот се прикажува порака на екран.
  - ЗАБЕЛЕШКА: Не сум сигурен што е точно проблемот но понекогаш одговорот од LlaMa Api е невалиден па затоа не добивам ниедна книга, ова ми се има случено 3 од околу 15 тестирања**.** Претпоставка ми е дека вероватно одговорот го надминува бројот на токени што може да има еден респонс.