**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**По лабораторной работе № 3**

**по дисциплине «Web-Технологии»**

Тема: **МОДУЛЬ АДМИНИСТРИРОВАНИЯ ПРИЛОЖЕНИЯ «СОЦИАЛЬНАЯ СЕТЬ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Болкунов В.О. |
| Преподаватель |  | Беляев С. А. |

Санкт-Петербург

2022

**Цель работы.**

Целью работы является изучение возможностей применения компилятора Babel, библиотеки jQuery, препроцессора LESS, препроцессора SASS/SCSS, инструмента выполнения повторяющихся задач GULP, освоение инструмента сборки Webpack, регистрация разработанных модулей, формирования навыков построения структурированных web-приложений, освоение особенностей стандартных библиотек. Для достижения поставленной цели требуется решить следующие задачи:

– разработка интерфейса web-приложения с использованием Figma (<https://www.figma.com/>);

– создание web-сервера на основе express, настройка маршрутов, подготовка и обработка REST-запросов (серверная часть);

– разработка стилей web-приложения с использованием LESS или SASS/SCSS;

**Общая формулировка задачи.**

Необходимо создать web-приложение, обеспечивающее администрирование социальной сети: можно управлять участниками, их ролями, сообщениями. Основные требования следующие:

1. Перечень участников, их друзей, сообщений и т.п. хранится в JSONфайлах на сервере.

2. В качестве сервера используется Node.JS с модулем express. 3. Разработка ведется с использованием стандарта не ниже ECMAScript2015, используются ES6 модули.

4. Стили описываются с использованием LESS или SASS, при этом используются ключевые методы LESS/SASS (переменные, вложенные блоки, миксины, операторы и т. п.).

5. Клиентская часть разрабатывается с использованием jQuery (работа с DOM, AJAX-запросы), используются компоненты jQuery UI или Bootstrap.

6. Предусмотрена HTML-страница для списка пользователей (ФИО, дата рождения, email, фотография, роль, статус). Предусмотрена возможность редактировать данные пользователя, изменять роль (администратор, пользователь), изменять статус (не подтверждённый пользователь, активный, заблокированный).

7. Предусмотрены: – HTML-страница для списка друзей пользователя; – HTML-страница для списка новостей друзей пользователей.

8. Взаимодействие браузера с сервером осуществляется по протоколу HTTPS, все изменения сохраняются в соответствующие json-файлы на сервере.

9. Сборка клиентской части (преобразования less или sass, pug или ejs, babel, минификация) осуществляется с использованием двух инструментов: GULP и Webpack. Это должны быть две отдельные сборки в разные папки.

10. Регистрация и удаление разработанных модулей в npm.

11. Для всех страниц web-приложения разработан макет интерфейса с использованием Figma (https://www.figma.com/). Преимуществом будет, если будет предусмотрена работа с – фотографиями пользователя: список фотографий и возможность блокирования и активации фотографий администратором; – новостями: возможность блокирования и активации новостей.

**Выполнение работы.**

**Промежуточное api**

В модуле *api* описаны общие для клиентского и серверного приложений структуры данных и объекты.

* Интерфейс объекта пользователя библиотеки

export interface Holder   
 holder: string | null;  
 returnDate: string | null;

* Информация о книге

export interface BookData   
 id?: number;  
 title: string;  
 author: string;  
 year: number;

* Данные о книге передающиеся клиенту

export interface BookSchema extends Holder, BookData

* Книга с обложкой (хранится на сервере)

export interface Book extends BookSchema   
 cover?: string;

* Тип фильтра в запросе

export enum RequestType   
 *ALL*,  
 *AVAILABLE*,  
 *EXPIRED*,

* Конфигурация клиент-серверного взаимодействия

export const config   
 serverHost: "localhost",  
 server: "http://localhost:3000/api/",  
 client: "http://localhost:8080",  
 reqTypeName: "type",  
 endpoints: {  
 login: "login/",  
 bookList: "books/",  
 covers: "covers/",  
 book: "book/",   
};

* Интерфейс объекта пользователя приложением

export interface User   
 login: string;  
 pwd: string;

**Серверная часть (REST)**

Управление книгами производит класс Library

class Library   
 private id = 1;  
 private books: Map<number, Book> = new Map();  
 constructor(init?: Book[])  
 getAll(req: RequestType): number[]  
 add(book: Book): number   
 get(id: number): Book | null   
 delete(id: number): boolean

REST-api библиотеки реализовано с помощью роутера express в модуле *libRouter.ts*

libRouter.get("/books", (req, res) => {…});libRouter.get("/covers/:id", (req, res) => {…});libRouter.post("/book", uploader.single("cover"), (req, res) => {…});libRouter  
 .route("/book/:id")  
 .all((req, res, next) => {… }).get((req, res) => {…}).put(uploader.single("cover"), (req, res) => {…}).patch((req, res) => {…});

Аутентификация пользователя осуществляется с помощью библиотеки passport.js с использованием стратегии Web-токенов. При отправке post-запроса с корректными данными пользователю отправляется cookie с подписанным токеном. Модуль *auth.ts* предоставляет middlware для аутентификации. Логин и пароль для входа хранятся в объекте admin

const admin: User = {  
 login: "admin",  
 pwd: "admin",  
};

passport.use(  
 new JwtStrategy(…)  
);authRouter.post("/login", (req, res) => {…});  
authRouter.get(  
 "/login",  
 passport.authenticate("jwt", { session: false }),  
 (req, res) => {…}  
);

**Клиентская часть**

Загрузку списка, запросы на добавление и удаление книг осуществляет промежуточный класс ListLoader

class ListLoader {  
 readonly books: number[] = [];  
 listUrl(): URL  
 get addUrl(): string   
 async fetch(req: RequestType)  
 async add(book: FormData)

remove(id?: number)

Загрузку и работу с отдельными книгами осуществляет класс BookLoader (поле book публичное, т.к. предоставляется компонентам как реактивное свойство)

class BookLoader   
 readonly id: number;  
 book: Book | null = null;  
 constructor(id: number)   
 get url()  
 get coverUrl()  
 async fetch()  
 async deleteBook()async patchBook(data: Holder) async putBook(bd: BookData, cover: File | null)

Карточка книги в списке книг представляется компонентом BookCard, который самостоятельно загружает книгу по переданному параметру id.

class BookCard extends Vue   
 @Prop({ type: *Number*, required: true }) readonly id!: number;  
 private loader: BookLoader = new BookLoader(this.id);  
 private BookAction = BookAction;  
 private expired = *expired*;  
 $refs!: {  
 cover: HTMLDivElement;  
 };  
 private async mounted()@Emit("action")  
 private action(act: BookAction): [BookAction, BookLoader]

Ввод информации о книге (при добавлении и редактировании) происходит через промежуточный компонент формы книги.

class BookForm extends Vue   
 readonly book: BookData = { title: "", author: "", year: 0 };  
 $refs!: {  
 cover: HTMLInputElement;  
 form: HTMLFormElement;  
 };reset()requestSubmit()private onSubmit()@Emit("submit-form")  
 private submitForm(): FormData @Emit("submit-data")  
 private submitData(): [BookData, File | null]

Компоненты BookFormModal, DeleteModal, GiveModal предоставляют модальные окна для ввода информации о добавленной книги, для подтверждения удаления и для выдачи книги соответственно

class BookFormModal extends Vue   
 private modal!: Modal;  
 $refs!: {  
 modal: HTMLDivElement;  
 form: BookForm;  
 };  
 private mounted()  
 show()  
 @Emit("submit")  
 private submit(event: FormData): FormData

class DeleteModal extends Vue   
 private modal!: Modal;  
 $refs!: {  
 modal: HTMLDivElement;  
 };  
 private mounted()  
 show()  
 @Emit("delete")  
 private deleteEvent()

class GiveModal extends Vue   
 private modal!: Modal;  
 private name = "";  
 private days = 0;  
  
 $refs!: {  
 modal: HTMLDivElement;  
 };  
 private mounted()  
 show()  
 private accept()  
 @Emit("give")  
 private giveEvent(): Holder

**Компоненты представления роутера**

LoginView – стартовая страница, на которую роутер переадресовывает неавторизированного пользователя

class LoginView extends Vue {  
 private login = "";  
 private pwd = "";  
 private readonly url = new *URL*(  
 config.endpoints.login,  
 config.server  
 ).toString();  
 private valid = true;  
 $refs!: {  
 invalidToast: HTMLDivElement;  
 };  
 private async submit()

private reset()

BookListView – отображает список карточек книг по выбранному

фильтру, здесь осуществляется добавление, выдача, возврат и удаление книг.

class BooksListView extends Vue {  
 private RequestType = RequestType;  
 private loader = new ListLoader();private selected: BookLoader | null = null;  
 $refs!: {  
 deleteModal: DeleteModal;  
 giveModal: GiveModal;  
 addModal: BookFormModal;  
 };  
 private async mounted()

private onDelete()

private onGive(holder: Holder) private bookAction(act: BookAction, loader: BookLoader)

BookView – страница редактирования отдельной книги. Здесь также возможны выдача, возврат и удаление (с переходом на страницу списка) книги.

class BookView extends Vue {  
 private id: number = *parseInt*(this.$route.params.book\_id);  
 private loader: BookLoader | null = null;  
 private expired = *expired*;  
 private cacheKey = 0;  
 $refs!: {  
 form: BookForm;  
 cover: HTMLInputElement;  
 saveToast: HTMLDivElement;  
 };  
 private async mounted()

private deleteBook()

private patchBook(data: Holder)   
 private async putBook(form: [BookData, File | null])

**Примеры экранных форм**.

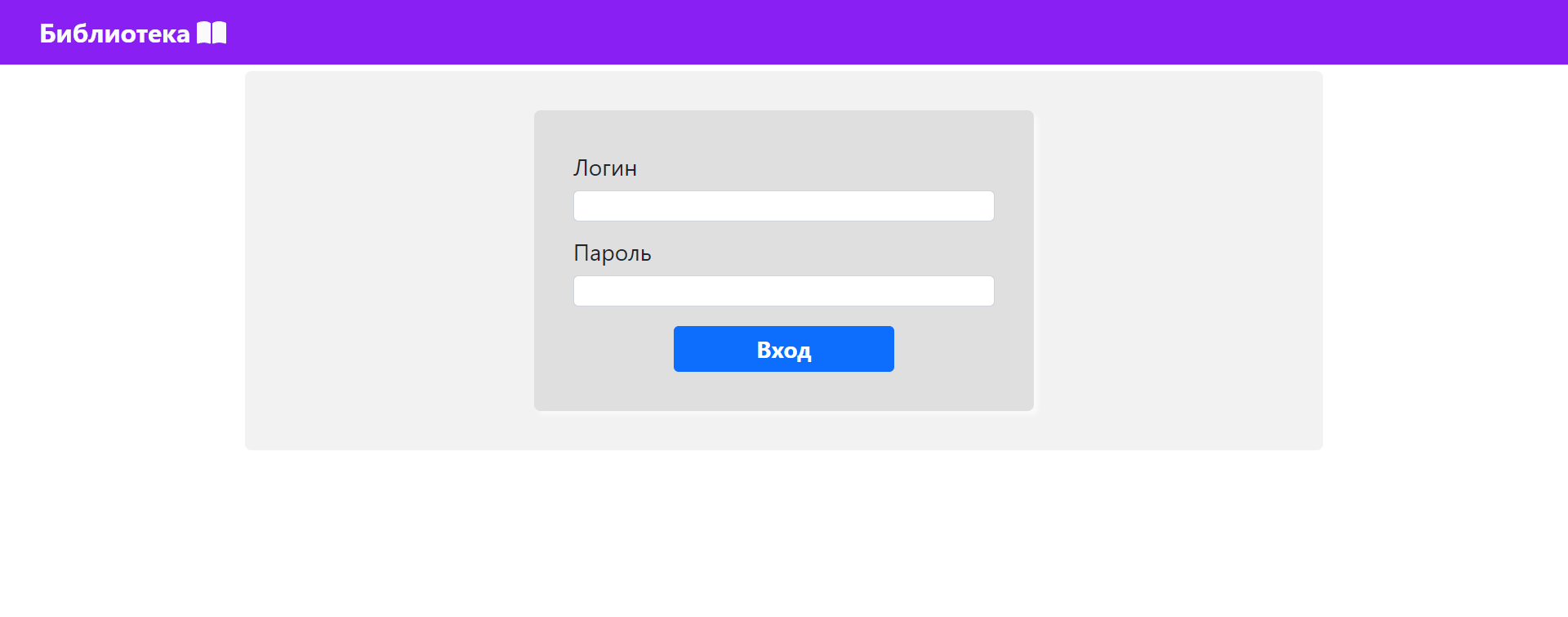


Рисунок : Стартовая страница

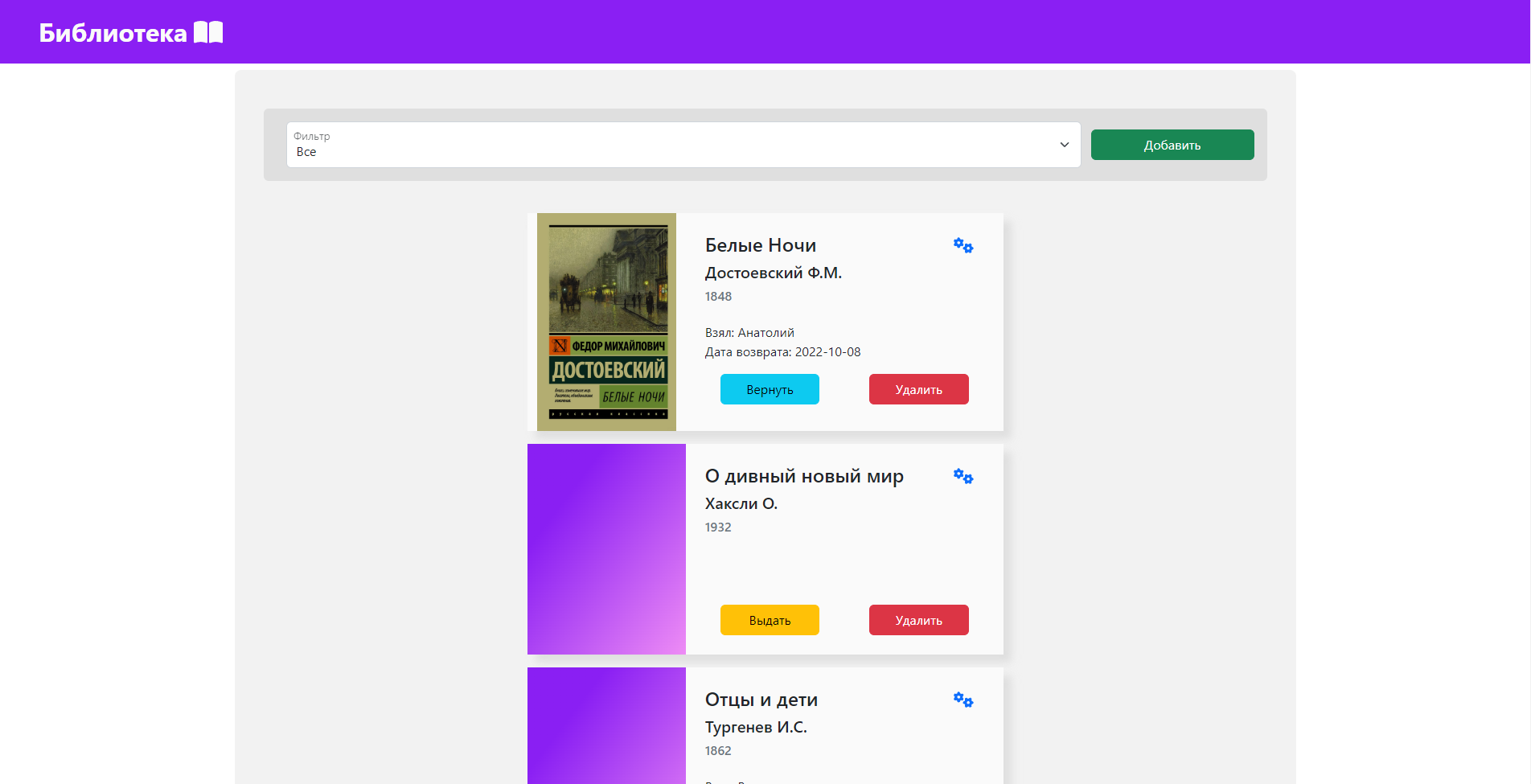


Рисунок : Список книг

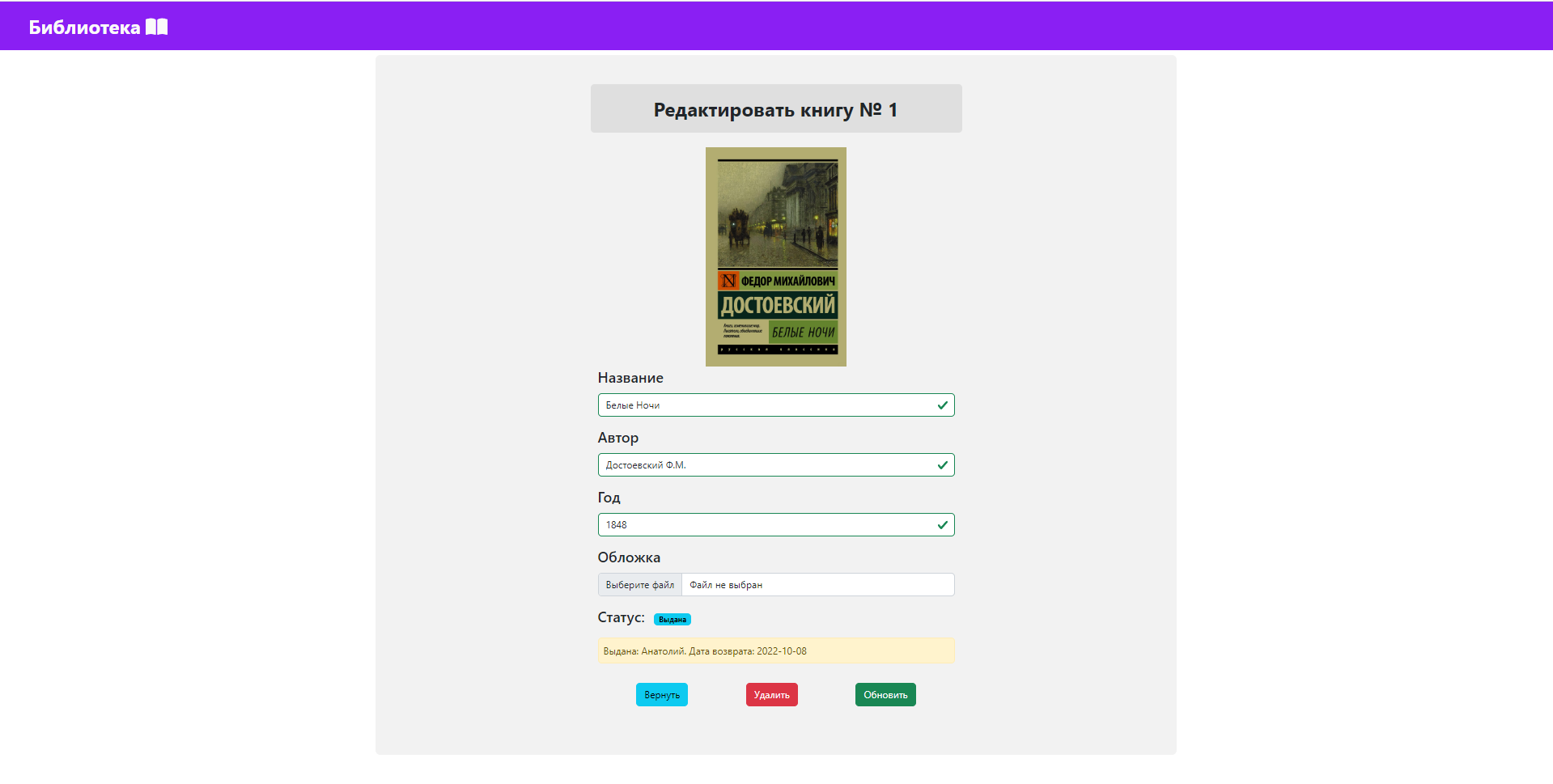


Рисунок : Страница редактирования книги

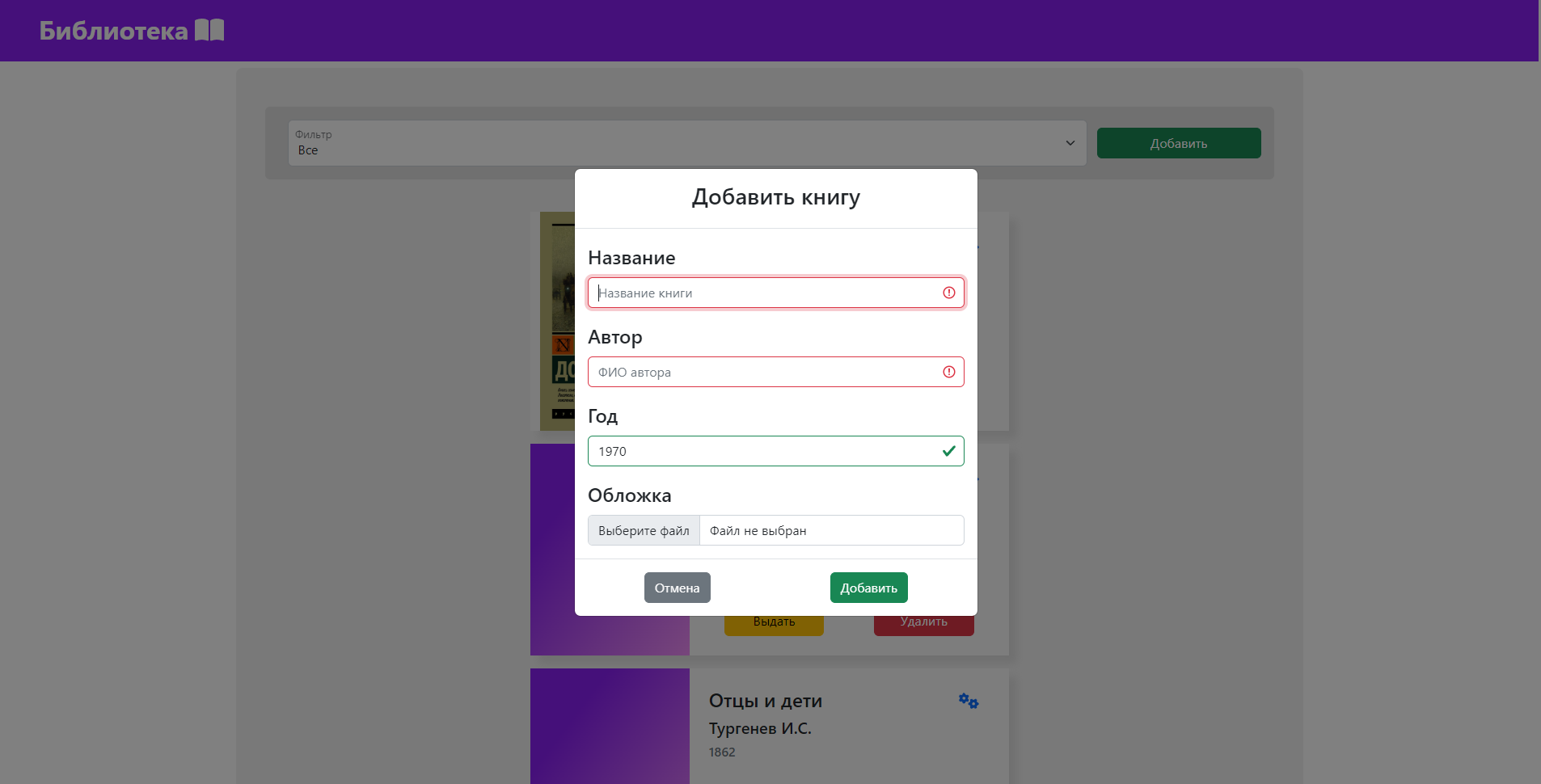
****

Рисунок : Добавление книги

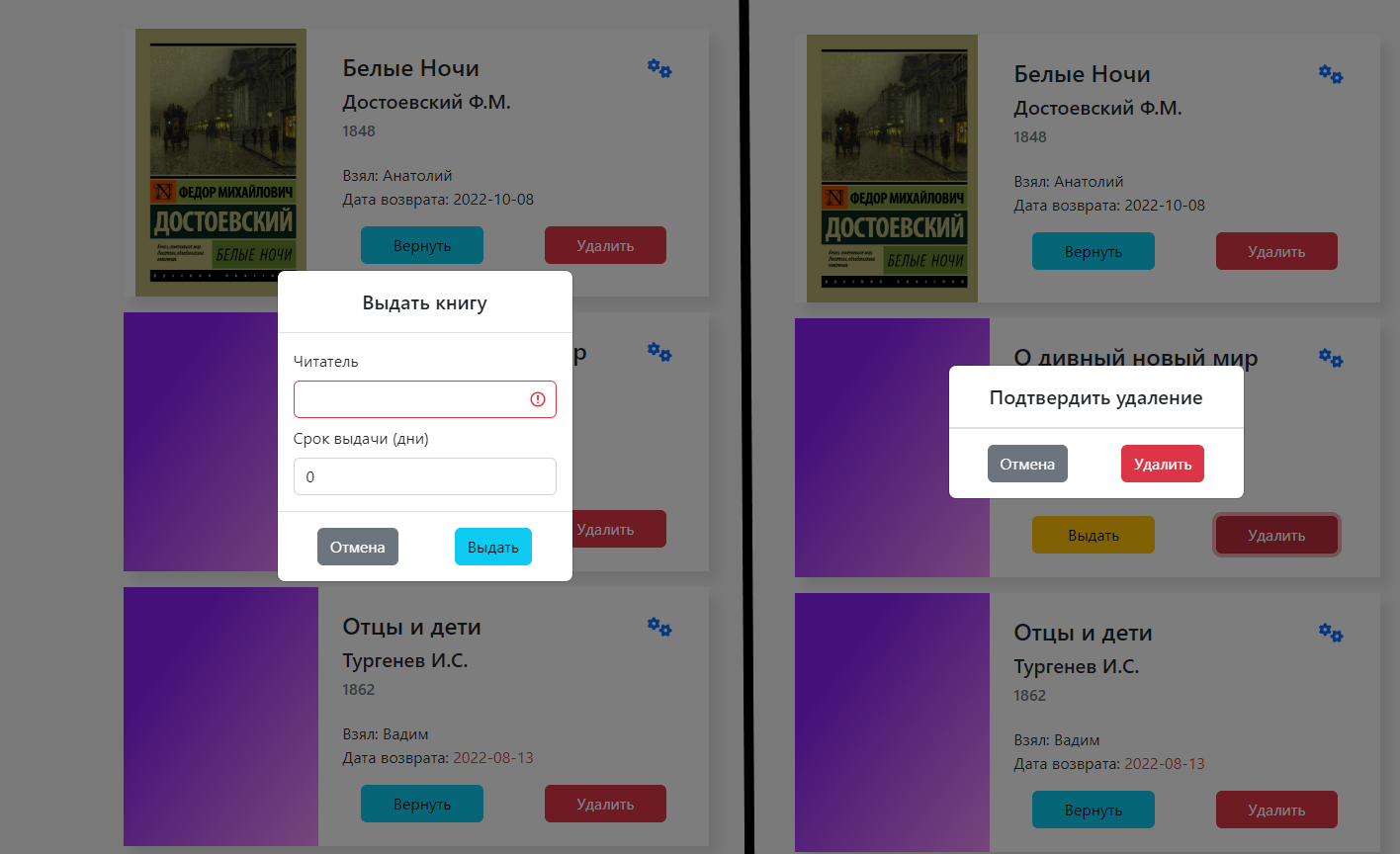
****

Рисунок : выдача и удаление

**Выводы.**

В ходе работы были изучены принципы взаимодействия клиентского и серверного приложения, изучена и реализована архитектура REST приложения управления библиотекой. Для соответствующего серверного приложения на основе созданного макета было реализовано клиентское приложение, взаимодействующее с сервером посредством ajax-запросов.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

***Файл api/index.ts***

*// Информация о пользователе библиотеки (кто берёт книги)*export interface Holder {  
 holder: string | null;  
 returnDate: string | null;  
}  
  
*// Функция проверки сроки сдачи книги*export function *expired*(holder: Holder): boolean {  
 return holder.returnDate != null && new Date() > new Date(holder.returnDate);  
}  
  
*// Основная информация о книге*export interface BookData {  
 id?: number;  
 title: string;  
 author: string;  
 year: number;  
}  
  
*// Вся информация о книге (отправляется клиенту)*export interface BookSchema extends Holder, BookData {}  
  
*// Информация о книге дополненная название файла обложки (хранится на сервере)*export interface Book extends BookSchema {  
 cover?: string;  
}  
  
*// Тип выбора списка книг*export enum RequestType {  
 *ALL*,  
 *AVAILABLE*,  
 *EXPIRED*,  
}  
  
*// Настройки клиент-серверного взаимодействия*export const config = {  
 serverHost: "localhost",  
 server: "http://localhost:3000/api/",  
 client: "http://localhost:8080",  
 reqTypeName: "type",  
 endpoints: {  
 login: "login/",  
 bookList: "books/",  
 covers: "covers/",  
 book: "book/",  
 },  
};  
  
*// Данные для входа*export interface User {  
 login: string;  
 pwd: string;  
}

***Файл server/Library.ts***

import { RequestType, Book, *expired* } from "../api";  
  
*// Класс управления книгами*export class Library {  
 private id = 1;  
 private books: Map<number, Book> = new Map();  
  
 constructor(init?: Book[]) {  
 if (init) for (const book of init) this.add(book);  
 }  
  
 getAll(req: RequestType): number[] {  
 const books = Array.from(this.books.keys());  
 switch (req) {  
 case RequestType.*ALL*:  
 return books;  
 case RequestType.*AVAILABLE*:  
 return books.filter((id) => !this.get(id)!.holder);  
 case RequestType.*EXPIRED*:  
 return books.filter((id) => *expired*(this.get!(id)!));  
 }  
 return [];  
 }  
  
 add(book: Book): number {  
 book.id = this.id++;  
 this.books.set(book.id, book);  
 return book.id;  
 }  
  
 get(id: number): Book | null {  
 return this.books.get(id) ?? null;  
 }  
  
 delete(id: number): boolean {  
 return this.books.delete(id);  
 }  
}

***Файл server/libRouter.ts***

import { *Router* } from "express";  
import { Book, BookData, BookSchema, Holder, RequestType } from "../api";  
import path from "path";  
import fs from "fs";  
import { Library } from "./Library";  
import multer from "multer";  
  
export const storagePath = "./storage";  
export const uploader = *multer*({ dest: storagePath });  
  
*// Начальная инициализация библиотеки*const library = new Library(  
 JSON.parse(  
 fs.*readFileSync*(path.resolve(storagePath, "library.json"), "utf-8")  
 ) as BookSchema[]  
);  
  
export const libRouter = *Router*();  
  
*// Получение списка id книг*libRouter.get("/books", (req, res) => {  
 res.json(  
 library.getAll(*parseInt*((req.query.type ?? "0") as string) as RequestType)  
 );  
});  
  
*// Обложки книг*libRouter.get("/covers/:id", (req, res) => {  
 const book = library.get(*parseInt*(req.params.id));  
 if (book?.cover) res.download(path.resolve(storagePath, book.cover));  
 else res.sendStatus(204);  
});  
  
*// Добавление книги*libRouter.post("/book", uploader.single("cover"), (req, res) => {  
 const book = req.body as BookData;  
 res.json(  
 library.add({  
 ...book,  
 cover: req.file?.filename,  
 holder: null,  
 returnDate: null,  
 } as Book)  
 );  
});  
  
*// Действия с книгой*libRouter  
 .route("/book/:id")  
 .all((req, res, next) => {  
 res.locals.book = library.get(*parseInt*(req.params.id));  
 if (!res.locals.book) res.sendStatus(204);  
 else next();  
 })  
 *// Получение информации о книге* .get((req, res) => {  
 res.json({ ...res.locals.book, cover: undefined } as BookSchema);  
 })  
 *// Обновление книги* .put(uploader.single("cover"), (req, res) => {  
 const cover = req.file?.filename,  
 book = req.body as BookData;  
  
 if (cover) {  
 *// Удаление предыдущей обложки* if (res.locals.book.cover)  
 fs.*unlink*(path.resolve(storagePath, res.locals.book.cover), (err) => {  
 if (err) console.log("Cover deleting error", err);  
 });  
 res.locals.book.cover = cover;  
 }  
 [res.locals.book.title, res.locals.book.author, res.locals.book.year] = [  
 book.title,  
 book.author,  
 book.year,  
 ];  
 res.sendStatus(200);  
 })  
 *// Выдача/возврат книги* .patch((req, res) => {  
 const data = req.body as Holder;  
  
 [res.locals.book.holder, res.locals.book.returnDate] = [  
 data.holder,  
 data.returnDate,  
 ];  
 res.sendStatus(200);  
 })  
 *// Удаление книги* .delete((req, res) => {  
 if (res.locals.book.cover)  
 fs.*unlink*(path.resolve(storagePath, res.locals.book.cover), (err) => {  
 console.log(`Cover ${res.locals.book.id} deleting error`, err);  
 });  
 library.delete(res.locals.book.id);  
 res.sendStatus(200);  
 });

***Файл server/auth.ts***

import passport from "passport";  
import { JwtFromRequestFunction, Strategy as JwtStrategy } from "passport-jwt";  
import { *Router* } from "express";  
import jwt from "jsonwebtoken";  
import { User } from "../api";  
  
export const authRouter = *Router*();  
  
*// Данные для входа*const admin: User = {  
 login: "admin",  
 pwd: "admin",  
};  
  
*// Закрытый ключ*export const secretKey = "#S3cR3t\_K3y#";  
  
*// Извлечение токена из куки*const cookieExtractor: JwtFromRequestFunction = function (req): string {  
 return req.signedCookies?.token ?? null;  
};  
  
*// Проверка токена*passport.use(  
 new JwtStrategy(  
 {  
 jwtFromRequest: cookieExtractor,  
 secretOrKey: secretKey,  
 },  
 (payload, done) => {  
 done(null, { name: admin.login });  
 }  
 )  
);  
  
*// Вход по логину и паролю, устанавливает токен в куки*authRouter.post("/login", (req, res) => {  
 const data = req.body as User;  
 console.log(data);  
 if (data.login == admin.login && data.pwd == admin.pwd) {  
 const token = jwt.*sign*({ name: admin.login }, secretKey, {  
 expiresIn: "1h",  
 });  
 res.cookie("token", token, {  
 maxAge: 60 \* 60 \* 1000,  
 httpOnly: true,  
 signed: true,  
 });  
  
 res.sendStatus(200);  
 } else res.sendStatus(401);  
});  
  
*// Точка проверки действительности токена*authRouter.get(  
 "/login",  
 passport.authenticate("jwt", { session: false }),  
 (req, res) => {  
 res.sendStatus(200);  
 }  
);  
  
export default passport;

***Файл server/index.ts***

import express from "express";  
import *cors* from "cors";  
import { libRouter } from "./libRouter";  
import { config } from "../api";  
import passport, { authRouter, secretKey } from "./auth";  
import cookieParser from "cookie-parser";  
  
const app = *express*();  
const port = 3000;  
  
app.use(  
 *cors*({  
 origin: config.client,  
 credentials: true,  
 })  
);  
app.use(*cookieParser*(secretKey));  
app.use(*express*.json());  
app.use(*express*.urlencoded({ extended: true }));  
app.use(passport.initialize());  
app.use("/api", authRouter);  
app.use("/api", passport.authenticate("jwt", { session: false }), libRouter);  
  
app.listen(port, config.serverHost, function () {  
 console.log(`Server started at http://${config.serverHost}:${port}`);  
});

***Файл src/util/BookLoader.ts***

import axios from "axios";  
import { Book, BookData, BookSchema, config, Holder } from "@/../api";  
  
*/\* Класс-контроллер для книг, осуществляет загрузку и выгрузку данных о книге  
 (при использовании в компонентах поле book становится реактивным)\*/*export class BookLoader {  
 readonly id: number;  
 book: Book | null = null;  
  
 constructor(id: number) {  
 this.id = id;  
 }  
  
 get url() {  
 return new URL(config.endpoints.book + this.id, config.server).toString();  
 }  
  
 get coverUrl() {  
 return new URL(config.endpoints.covers + this.id, config.server).toString();  
 }  
  
 async fetch() {  
 this.book = {  
 ...(await axios.get<BookSchema>(this.url)).data,  
 cover:  
 (await axios.head(this.coverUrl)).status == 200  
 ? this.coverUrl  
 : undefined,  
 } as Book;  
 }  
  
 async deleteBook() {  
 if (this.book) {  
 await axios.delete(this.url.toString());  
 this.book = null;  
 }  
 }  
  
 *// Обновление информации о клиенте (кто взял)* async patchBook(data: Holder) {  
 if (this.book) {  
 await axios.patch(this.url.toString(), data);  
 [this.book.holder, this.book.returnDate] = [data.holder, data.returnDate];  
 }  
 }  
  
 *// Обновление информации о самой книге* async putBook(bd: BookData, cover: File | null) {  
 if (this.book) {  
 const form = new FormData();  
 if (this.book) {  
 form.append("title", bd.title);  
 form.append("author", bd.author);  
 form.append("year", bd.year.toString());  
 if (cover) {  
 form.append("cover", cover);  
 }  
 }  
 await axios.put(this.url.toString(), form);  
 [this.book.title, this.book.author, this.book.year, this.book.cover] = [  
 bd.title,  
 bd.author,  
 bd.year,  
 (await axios.head(this.coverUrl)).status == 200  
 ? this.coverUrl  
 : undefined,  
 ];  
 }  
 }  
}

***Файл src/util/ListLoader.ts***

import { config, RequestType } from "@/../api";  
import axios from "axios";  
  
export class ListLoader {  
 readonly books: number[] = [];  
  
 listUrl(): URL {  
 return new *URL*(config.endpoints.bookList, config.server);  
 }  
  
 get addUrl(): string {  
 return new *URL*(config.endpoints.book, config.server).toString();  
 }  
  
 async fetch(req: RequestType) {  
 const url = this.listUrl();  
 url.searchParams.append(config.reqTypeName, req.toString());  
 const res = await axios.get<number[]>(url.toString());  
 this.books.splice(0, this.books.length, ...res.data);  
 }  
  
 async add(book: FormData) {  
 const res = await axios.post<number>(this.addUrl, book);  
 this.books.unshift(res.data);  
 }  
  
 remove(id?: number) {  
 const index = id ? this.books.indexOf(id) : -1;  
 if (index !== -1) this.books.splice(index, 1);  
 }  
}

***Файл src/styles/main.scss***

$primary: #8a1ff3;  
$secondary: #ee8cf5;  
$text-light: #fafafa;  
$bg-light: #fafafa;  
$bg-grey1: #f2f2f2;  
$bg-grey2: #dfdfdf;

***Файл src/components/BookCard.vue***

<template>  
 <div class="book row m-3 d-flex justify-content-center" v-if="loader.book">  
 <div class="cover col-6 col-md-4 d-flex align-items-center" ref="cover">  
 <img class="" width="100%" :src="loader.book.cover" />  
 </div>  
 <div class="col-12 col-md-8 p-4">  
 <div class="row">  
 <h4 class="col-10">  
 {{ loader.book.title }}  
 </h4>  
 <router-link  
 class="col-2 fs-5"  
 :to="{  
 name: 'book',  
 params: { book\_id: this.loader.id.toString() },  
 }"  
 >  
 <font-awesome-icon icon="fa-gears" />  
 </router-link>  
 </div>  
 <h5>{{ loader.book.author }}</h5>  
 <h6 class="text-muted">{{ loader.book.year }}</h6>  
  
 <div :class="{ 'opacity-0': !loader.book.holder }">  
 <p class="mt-4 mb-0">Взял: {{ loader.book.holder }}</p>  
 <p>  
 Дата возврата:  
 <span  
 :class="{  
 'text-danger': expired(loader.book),  
 }"  
 >  
 {{ loader.book.returnDate }}  
 </span>  
 </p>  
 </div>  
  
 <div class="row justify-content-around">  
 <button  
 v-if="!loader.book.holder"  
 class="col-5 col-md-4 btn btn-warning"  
 @click="action(BookAction.*GIVE*)"  
 >  
 Выдать  
 </button>  
 <button  
 v-else  
 class="col-5 col-md-4 btn btn-info"  
 @click="action(BookAction.*RETURN*)"  
 >  
 Вернуть  
 </button>  
 <button  
 class="col-5 col-md-4 btn btn-danger"  
 @click="action(BookAction.*DELETE*)"  
 >  
 Удалить  
 </button>  
 </div>  
 </div>  
 </div>  
</template>  
  
<script lang="ts">  
import { *Component*, *Emit*, *Prop*, Vue } from "vue-property-decorator";  
import { BookLoader } from "@/util/BookLoader";  
import { *expired* } from "@/../api";  
  
*// Действия передаваемые из карточки книги*export enum BookAction {  
 *DELETE*,  
 *GIVE*,  
 *RETURN*,  
}  
  
*// Карточка книги в списке книг*@Component({})  
export default class BookCard extends Vue {  
 @Prop({ type: *Number*, required: true }) readonly id!: number;  
 private loader: BookLoader = new BookLoader(this.id);  
 private BookAction = BookAction;  
 private expired = *expired*;  
  
 $refs!: {  
 cover: HTMLDivElement;  
 };  
  
 private async mounted() {  
 this.loader = new BookLoader(this.id);  
 await this.loader.fetch();  
 console.log(this.loader.book);  
 if (this.loader.book?.cover) this.$refs.cover.style.background = "inherit";  
 }  
  
 *// Событие карточки (удаление, выдача, возврат)* @Emit("action")  
 private action(act: BookAction): [BookAction, BookLoader] {  
 return [act, this.loader];  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "../styles/main";  
  
.book {  
 background-color: $bg-light;  
 box-shadow: 10px 10px 10px $bg-grey2;  
  
 .cover {  
 background: linear-gradient(to right bottom, $primary 25%, $secondary);  
 background-size: contain;  
 background-repeat: no-repeat;  
 background-position: center center;  
 transition: all 0.3s ease-in-out;  
  
 &:hover {  
 transform: scale(0.98);  
 }  
 }  
}  
</style>

***Файл src/components/BookForm.vue***

<template>  
 <form ref="form" @submit.prevent="onSubmit">  
 <div class="mb-3">  
 <label for="title" class="form-label fs-4 fw-semibold">Название</label>  
 <input  
 v-model="book.title"  
 :class="book.title === '' ? 'is-invalid' : 'is-valid'"  
 type="text"  
 class="form-control"  
 id="title"  
 required  
 placeholder="Название книги"  
 />  
 </div>  
  
 <div class="mb-3 mt-3">  
 <label for="author" class="form-label fs-4 fw-semibold">Автор</label>  
 <input  
 v-model="book.author"  
 :class="book.author === '' ? 'is-invalid' : 'is-valid'"  
 type="text"  
 class="form-control"  
 id="author"  
 required  
 placeholder="ФИО автора"  
 />  
 </div>  
  
 <div class="mb-3 mt-3">  
 <label for="year" class="form-label fs-4 fw-semibold">Год</label>  
 <input  
 v-model="book.year"  
 :class="book.year == '' ? 'is-invalid' : 'is-valid'"  
 type="number"  
 class="form-control"  
 id="year"  
 required  
 />  
 </div>  
  
 <div class="mt-3">  
 <label for="cover" class="form-label fs-4 fw-semibold">Обложка</label>  
 <input  
 type="file"  
 class="form-control"  
 id="cover"  
 ref="cover"  
 placeholder="Обложка книги"  
 accept="image/\*"  
 />  
 </div>  
 </form>  
</template>  
  
<script lang="ts">  
import { *Component*, *Emit*, Vue } from "vue-property-decorator";  
import { BookData } from "@/../api";  
  
*// Компонент формы редактирования книги, используется в модальном окне добавления и на странице самой книги*@Component({})  
export default class BookForm extends Vue {  
 readonly book: BookData = { title: "", author: "", year: 0 };  
  
 $refs!: {  
 cover: HTMLInputElement;  
 form: HTMLFormElement;  
 };  
  
 *// Сброс формы* reset() {  
 this.$refs.form.reset();  
 [this.book.title, this.book.author, this.book.year] = ["", "", 1970];  
 }  
  
 *// Попытка "отправить" форму, если валидация пройдена вызываются события submit-data и submit-form* requestSubmit() {  
 this.$refs.form.requestSubmit();  
 }  
  
 *// Проброс события submit формы* private onSubmit() {  
 this.submitData();  
 this.submitForm();  
 }  
  
 *// Отправка данных в виде FormDat-ы* @Emit("submit-form")  
 private submitForm(): FormData {  
 let form = new *FormData*();  
 form.append("title", this.book.title);  
 form.append("author", this.book.author);  
 form.append("year", this.book.year.toString());  
 if (this.$refs.cover.files?.length) {  
 form.append("cover", this.$refs.cover.files[0]);  
 }  
 return form;  
 }  
  
 *// Отправка данных в виде объекта книги и файла* @Emit("submit-data")  
 private submitData(): [BookData, File | null] {  
 return [this.book, this.$refs.cover.files?.item(0) ?? null];  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/BookFormModal.vue***

<template>  
 <section class="modal fade" data-bs-backdrop="static" ref="modal">  
 <div class="modal-dialog modal-dialog-centered modal-md">  
 <div class="modal-content">  
 <div class="modal-header justify-content-around">  
 <h3 class="text-center">Добавить книгу</h3>  
 </div>  
 <BookForm  
 class="modal-body"  
 ref="form"  
 @submit-form="submit"  
 ></BookForm>  
 <div class="modal-footer d-flex justify-content-around">  
 <button class="btn btn-secondary" @click="modal.hide()">  
 Отмена  
 </button>  
 <button class="btn btn-success" @click="$refs.form.requestSubmit()">  
 Добавить  
 </button>  
 </div>  
 </div>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { *Component*, *Emit*, Vue } from "vue-property-decorator";  
import BookForm from "@/components/BookForm.vue";  
import { Modal } from "bootstrap";  
  
*// Всплывающее окно формы книги для её добавления в библиотеку*@Component({  
 components: { BookForm },  
})  
export default class BookFormModal extends Vue {  
 private modal!: Modal;  
  
 $refs!: {  
 modal: HTMLDivElement;  
 form: BookForm;  
 };  
  
 private mounted() {  
 this.modal = new Modal(this.$refs.modal);  
 }  
  
 show() {  
 this.modal.show();  
 this.$refs.form.reset();  
 }  
  
 @Emit("submit")  
 private submit(event: FormData): FormData {  
 this.modal.hide();  
 return event;  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/DeleteModal.vue***

<template>  
 <section class="modal fade" data-bs-backdrop="static" ref="modal">  
 <div class="modal-dialog modal-dialog-centered modal-sm">  
 <div class="modal-content text-center">  
 <div class="modal-header justify-content-center">  
 <h5 class="modal-title">Подтвердить удаление</h5>  
 </div>  
 <div class="modal-footer justify-content-around">  
 <button class="btn btn-secondary" @click="modal.hide()">  
 Отмена  
 </button>  
 <button class="btn btn-danger" @click="deleteEvent">Удалить</button>  
 </div>  
 </div>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { *Component*, *Emit*, Vue } from "vue-property-decorator";  
import { Modal } from "bootstrap";  
  
*// Всплывающее окно подтверждения удаления книги*@Component  
export default class DeleteModal extends Vue {  
 private modal!: Modal;  
  
 $refs!: {  
 modal: HTMLDivElement;  
 };  
  
 private mounted() {  
 this.modal = new Modal(this.$refs.modal);  
 }  
  
 show() {  
 this.modal.show();  
 }  
  
 @Emit("delete")  
 private deleteEvent() {  
 this.modal.hide();  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/GiveModal.vue***

<template>  
 <section class="modal fade" data-bs-backdrop="static" ref="modal">  
 <form  
 class="modal-dialog modal-dialog-centered modal-sm"  
 @submit.prevent="accept"  
 @reset.prevent="modal.hide()"  
 >  
 <div class="modal-content">  
 <div class="modal-header justify-content-center">  
 <h5 class="modal-title">Выдать книгу</h5>  
 </div>  
  
 <div class="modal-body">  
 <label for="name" class="form-label">Читатель</label>  
 <input  
 type="text"  
 v-model="name"  
 class="form-control"  
 id="name"  
 :class="name === '' ? 'is-invalid' : 'is-valid'"  
 required  
 />  
  
 <label for="name" class="mt-2 form-label">Срок выдачи (дни)</label>  
 <input  
 type="number"  
 v-model="days"  
 class="form-control"  
 id="name"  
 min="0"  
 />  
 </div>  
  
 <div class="modal-footer justify-content-around">  
 <button type="reset" class="btn btn-secondary">Отмена</button>  
 <button type="submit" class="btn btn-info">Выдать</button>  
 </div>  
 </div>  
 </form>  
 </section>  
</template>  
  
<script lang="ts">  
import { *Component*, *Emit*, Vue } from "vue-property-decorator";  
import { Modal } from "bootstrap";  
import { Holder } from "@/../api";  
  
*// Всплывающее окно выдачи книги*@Component  
export default class GiveModal extends Vue {  
 private modal!: Modal;  
 private name = "";  
 private days = 0;  
  
 $refs!: {  
 modal: HTMLDivElement;  
 };  
  
 private mounted() {  
 this.modal = new Modal(this.$refs.modal);  
 }  
  
 show() {  
 this.modal.show();  
 this.name = "";  
 this.days = 0;  
 }  
  
 private accept() {  
 this.modal.hide();  
 this.giveEvent();  
 }  
  
 @Emit("give")  
 private giveEvent(): Holder {  
 let date = new Date();  
 let days = +date.getDate() + +this.days;  
 date.setDate(days);  
 return { holder: this.name, returnDate: date.toISOString().split("T")[0] };  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/router/index.ts***

import Vue from "vue";  
import VueRouter, { RouteConfig } from "vue-router";  
import axios from "axios";  
import { config } from "@/../api";  
  
Vue.use(VueRouter);  
  
const routes: Array<RouteConfig> = [  
 {  
 path: "/",  
 name: "login",  
 component: () => import("../views/LoginView.vue"),  
 },  
 {  
 path: "/home",  
 name: "home",  
 component: () => import("../views/BooksListView.vue"),  
 },  
 {  
 path: "/book/:book\_id",  
 name: "book",  
 component: () => import("../views/BookView.vue"),  
 },  
];  
  
const router = new VueRouter({  
 routes,  
});  
  
async function isAuthenticated(): Promise<boolean> {  
 try {  
 await axios.get(new URL(config.endpoints.login, config.server).toString(), {  
 withCredentials: true,  
 });  
 return true;  
 } catch (err) {  
 return false;  
 }  
}  
  
router.beforeEach(async (to, from, next) => {  
 const auth = await isAuthenticated();  
 if (auth) {  
 if (to.name === "login") next({ name: "home" });  
 else next();  
 } else {  
 if (to.name !== "login") next({ name: "login" });  
 else next();  
 }  
});  
  
export default router;

***Файл src/views/LoginView.vue***

<template>  
 <form  
 class="login container rounded-3 col-12 col-md-10 col-lg-6 p-5"  
 @submit.prevent="submit"  
 >  
 <div class="fs-3">  
 <label for="login" class="form-label">Логин</label>  
 <input  
 v-model="login"  
 type="text"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="login"  
 required  
 />  
  
 <label for="pwd" class="mt-3 form-label">Пароль</label>  
 <input  
 v-model="pwd"  
 type="password"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="pwd"  
 required  
 />  
 </div>  
 <div class="mt-4 row d-flex justify-content-center">  
 <button type="submit" class="btn btn-primary col-6 fs-3 fw-bold">  
 Вход  
 </button>  
 </div>  
  
 <div class="toast-container position-fixed bottom-0 end-0 p-3">  
 <div ref="invalidToast" class="toast" role="alert">  
 <div class="alert alert-danger m-0 text-center p-2">  
 <span>Неверные данные для входа</span>  
 </div>  
 </div>  
 </div>  
 </form>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue } from "vue-property-decorator";  
import axios from "axios";  
import { config, User } from "@/../api";  
import { Toast } from "bootstrap";  
  
@Component({})  
export default class LoginView extends Vue {  
 private login = "";  
 private pwd = "";  
 private readonly url = new *URL*(  
 config.endpoints.login,  
 config.server  
 ).toString();  
 private valid = true;  
  
 $refs!: {  
 invalidToast: HTMLDivElement;  
 };  
  
 private async submit() {  
 try {  
 await axios.post(  
 this.url,  
 {  
 login: this.login,  
 pwd: this.pwd,  
 } as User,  
 { withCredentials: true }  
 );  
 this.$router.push({ name: "home" });  
 } catch (err) {  
 new Toast(this.$refs.invalidToast, {  
 delay: 5000,  
 }).show();  
 this.valid = false;  
 }  
 }  
  
 private reset() {  
 this.valid = true;  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "../styles/main";  
.login {  
 background-color: $bg-grey2;  
 box-shadow: 5px 5px 5px $bg-light;  
}  
</style>

***Файл src/views/BooksLitsView.vue***

<template>  
 <section class="row justify-content-center">  
 *<!-- Верхняя панель с фильтром и кнопкой добавить -->* <div  
 class="filter row align-items-center p-3 rounded justify-content-between"  
 >  
 <div class="col-8 col-md-10 form-floating">  
 <select  
 id="select"  
 class="form-select"  
 @change="loader.fetch($event.target.value)"  
 >  
 <option :value="RequestType.*ALL*" selected>Все</option>  
 <option :value="RequestType.*AVAILABLE*">В наличии</option>  
 <option :value="RequestType.*EXPIRED*">Возврат просрочен</option>  
 </select>  
 <label for="select" class="ms-2">Фильтр</label>  
 </div>  
 <button  
 class="col-4 col-md-2 btn btn-success"  
 @click="$refs.addModal.show()"  
 >  
 Добавить  
 </button>  
 </div>  
  
 *<!-- Список книг -->* <article class="col-xs-12 col-md-10 col-lg-6 mt-4 p-0">  
 <transition-group v-if="loader.books" name="books">  
 <BookCard  
 v-for="id of loader.books"  
 :id="id"  
 :key="id"  
 ref="bookCards"  
 @action="bookAction(...$event)"  
 ></BookCard>  
 </transition-group>  
 </article>  
  
 *<!-- Модальные окна -->* <DeleteModal ref="deleteModal" @delete="onDelete"> </DeleteModal>  
 <GiveModal ref="giveModal" @give="onGive"></GiveModal>  
 <BookFormModal ref="addModal" @submit="loader.add($event)"></BookFormModal>  
 </section>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue } from "vue-property-decorator";  
import BookCard, { BookAction } from "@/components/BookCard.vue";  
import { Holder, RequestType } from "@/../api";  
import DeleteModal from "@/components/DeleteModal.vue";  
import GiveModal from "@/components/GiveModal.vue";  
import { BookLoader } from "@/util/BookLoader";  
import BookFormModal from "@/components/BookFormModal.vue";  
import { ListLoader } from "@/util/ListLoader";  
  
*// Список книг с фильтрами*@Component({  
 components: { BookFormModal, GiveModal, DeleteModal, BookCard },  
})  
export default class BooksListView extends Vue {  
 private RequestType = RequestType;  
 private loader = new ListLoader();  
 *// Контроллер книги выбранной в карточке (передаётся из её действия)* private selected: BookLoader | null = null;  
  
 $refs!: {  
 deleteModal: DeleteModal;  
 giveModal: GiveModal;  
 addModal: BookFormModal;  
 };  
  
 private mounted() {  
 this.loader.fetch(RequestType.*ALL*);  
 }  
  
 private onDelete() {  
 this.selected?.deleteBook();  
 this.loader.remove(this.selected?.id);  
 }  
  
 private onGive(holder: Holder) {  
 this.selected?.patchBook(holder);  
 }  
  
 *// Обработка действий из карточек* private bookAction(act: BookAction, loader: BookLoader) {  
 this.selected = loader;  
 switch (act) {  
 case BookAction.*DELETE*:  
 this.$refs.deleteModal.show();  
 break;  
 case BookAction.*GIVE*:  
 this.$refs.giveModal.show();  
 break;  
 case BookAction.*RETURN*:  
 loader.patchBook({ holder: null, returnDate: null });  
 break;  
 }  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "../styles/main";  
  
.filter {  
 background-color: $bg-grey2;  
}  
  
.books-enter-active,  
.books-leave-active {  
 transition: all 0.5s ease-in-out;  
}  
  
.books-enter-to,  
.books-leave {  
 max-height: 40vh;  
}  
  
.books-enter,  
.books-leave-to {  
 opacity: 0;  
 max-height: 0;  
}  
</style>

***Файл src/views/BookView.vue***

<template>  
 <article  
 class="row justify-content-center col-12 col-md-8 col-lg-6 m-auto"  
 v-if="loader?.book"  
 >  
 <div class="redact-title text-center fs-2 p-3 fw-bold rounded">  
 Редактировать книгу № {{ id }}  
 </div>  
  
 <img  
 class="mt-4 col-7 col-md-6 col-lg-5"  
 v-if="loader.book.cover"  
 :src="loader.book.cover + `?cache=${cacheKey}`"  
 />  
  
 <BookForm ref="form" @submit-data="putBook"></BookForm>  
  
 <div class="mb-3 mt-3">  
 <p class="fs-4 fw-semibold">  
 <span class="me-3">Статус:</span>  
 <span class="fs-6">  
 <span v-if="!loader.book.holder" class="badge text-bg-success"  
 >В наличии</span  
 >  
 <span v-else>  
 <span v-if="!expired(loader.book)" class="badge text-bg-info">  
 Выдана  
 </span>  
 <span v-else class="badge text-bg-warning">Возврат просрочен</span>  
 </span>  
 </span>  
 </p>  
  
 <div v-if="loader.book.holder">  
 <p  
 class="mt-3 p-2 alert"  
 :class="expired(loader.book) ? 'alert-danger' : 'alert-warning'"  
 >  
 Выдана: {{ loader.book.holder }}. Дата возврата:  
 {{ loader.book.returnDate }}  
 </p>  
 </div>  
  
 <div class="mt-3 p-3 d-flex justify-content-around">  
 <button  
 type="button"  
 v-if="!loader.book.holder"  
 class="btn btn-warning"  
 @click="$refs.giveModal.show()"  
 >  
 Выдать  
 </button>  
 <button  
 v-else  
 type="button"  
 class="btn btn-info"  
 @click="patchBook({ holder: null, returnDate: null })"  
 >  
 Вернуть  
 </button>  
 <button  
 type="button"  
 class="btn btn-danger"  
 @click="$refs.deleteModal.show()"  
 >  
 Удалить  
 </button>  
 <button  
 type="button"  
 class="btn btn-success"  
 @click="$refs.form.requestSubmit()"  
 >  
 Обновить  
 </button>  
 </div>  
 </div>  
  
 *<!-- Модальные окна -->* <DeleteModal ref="deleteModal" @delete="deleteBook"> </DeleteModal>  
 <GiveModal ref="giveModal" @give="patchBook"></GiveModal>  
  
 <div class="toast-container position-fixed bottom-0 end-0 p-3">  
 <div ref="saveToast" class="toast" role="alert">  
 <div class="alert alert-success m-0 text-center p-2">  
 <span>Обновлено</span>  
 </div>  
 </div>  
 </div>  
 </article>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue } from "vue-property-decorator";  
import { BookData, *expired*, Holder } from "@/../api";  
import DeleteModal from "@/components/DeleteModal.vue";  
import GiveModal from "@/components/GiveModal.vue";  
import { Toast } from "bootstrap";  
import BookForm from "@/components/BookForm.vue";  
import { BookLoader } from "@/util/BookLoader";  
  
*// Страница книги*@Component({  
 components: { BookForm, GiveModal, DeleteModal },  
})  
export default class BookView extends Vue {  
 private id: number = *parseInt*(this.$route.params.book\_id);  
 private loader: BookLoader | null = null;  
 private expired = *expired*;  
 private cacheKey = 0;  
  
 $refs!: {  
 form: BookForm;  
 cover: HTMLInputElement;  
 saveToast: HTMLDivElement;  
 };  
  
 private async mounted() {  
 this.loader = new BookLoader(this.id);  
 await this.loader.fetch();  
 if (this.$refs.form.book && this.loader.book)  
 [  
 this.$refs.form.book.title,  
 this.$refs.form.book.author,  
 this.$refs.form.book.year,  
 ] = [  
 this.loader.book.title,  
 this.loader.book.author,  
 this.loader.book.year,  
 ];  
 }  
  
 private deleteBook() {  
 this.loader?.deleteBook();  
 this.$router.push({ name: "home" });  
 }  
  
 private patchBook(data: Holder) {  
 this.loader?.patchBook(data);  
 }  
  
 private async putBook(form: [BookData, File | null]) {  
 await this.loader?.putBook(...form);  
 new Toast(this.$refs.saveToast).show();  
 this.cacheKey = *Date*.now();  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "../styles/main";  
  
.redact-title {  
 background-color: $bg-grey2;  
}  
</style>

***Файл src/views/LoginView.vue***

<template>  
 <form  
 class="login container rounded-3 col-12 col-md-10 col-lg-6 p-5"  
 @submit.prevent="submit"  
 >  
 <div class="fs-3">  
 <label for="login" class="form-label">Логин</label>  
 <input  
 v-model="login"  
 type="text"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="login"  
 required  
 />  
  
 <label for="pwd" class="mt-3 form-label">Пароль</label>  
 <input  
 v-model="pwd"  
 type="password"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="pwd"  
 required  
 />  
 </div>  
 <div class="mt-4 row d-flex justify-content-center">  
 <button type="submit" class="btn btn-primary col-6 fs-3 fw-bold">  
 Вход  
 </button>  
 </div>  
  
 <div class="toast-container position-fixed bottom-0 end-0 p-3">  
 <div ref="invalidToast" class="toast" role="alert">  
 <div class="alert alert-danger m-0 text-center p-2">  
 <span>Неверные данные для входа</span>  
 </div>  
 </div>  
 </div>  
 </form>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue } from "vue-property-decorator";  
import axios from "axios";  
import { config, User } from "@/../api";  
import { Toast } from "bootstrap";  
  
@Component({})  
export default class LoginView extends Vue {  
 private login = "";  
 private pwd = "";  
 private readonly url = new *URL*(  
 config.endpoints.login,  
 config.server  
 ).toString();  
 private valid = true;  
  
 $refs!: {  
 invalidToast: HTMLDivElement;  
 };  
  
 private async submit() {  
 try {  
 await axios.post(  
 this.url,  
 {  
 login: this.login,  
 pwd: this.pwd,  
 } as User,  
 { withCredentials: true }  
 );  
 this.$router.push({ name: "home" });  
 } catch (err) {  
 new Toast(this.$refs.invalidToast, {  
 delay: 5000,  
 }).show();  
 this.valid = false;  
 }  
 }  
  
 private reset() {  
 this.valid = true;  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "../styles/main";  
.login {  
 background-color: $bg-grey2;  
 box-shadow: 5px 5px 5px $bg-light;  
}  
</style>

***Файл src/App.vue***

<template>  
 <div id="app">  
 <header class="header container-fluid pt-3 pb-3 ps-5 pe-5 fs-2">  
 <span class="me-2">Библиотека</span>  
 <font-awesome-icon icon="fa-book-open" />  
 </header>  
 <main class="view container rounded-3 mt-2 p-5">  
 <router-view />  
 </main>  
 </div>  
</template>  
  
<script lang="ts">  
import { Component, Vue } from "vue-property-decorator";  
  
@Component  
export default class App extends Vue {}  
</script>  
  
<style lang="scss">  
@import "bootstrap/scss/bootstrap";  
@import "styles/main";  
  
#app {  
}  
  
.header {  
 background-color: $primary;  
 color: $text-light;  
 font-weight: bold;  
}  
  
.view {  
 background-color: $bg-grey1;  
}  
</style>

***Файл src/main.ts***

import Vue from "vue";  
import App from "./App.vue";  
import router from "./router";  
import { library } from "@fortawesome/fontawesome-svg-core";  
import { faBookOpen, faGears } from "@fortawesome/free-solid-svg-icons";  
import { FontAwesomeIcon } from "@fortawesome/vue-fontawesome";  
import axios from "axios";  
  
library.add(faBookOpen, faGears);  
Vue.component("font-awesome-icon", FontAwesomeIcon);  
  
Vue.config.productionTip = false;  
new Vue({  
 router,  
 render: (h) => h(App),  
}).$mount("#app");  
  
*// Добавить куки ко всем запросам*axios.interceptors.request.use(function (config) {  
 config.withCredentials = true;  
 return config;  
});

***Файл public/index.html***

<!DOCTYPE html>  
<html lang="">  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width,initial-scale=1.0">  
 <link rel="icon" href="<%= BASE\_URL %>favicon.ico">  
 <title><%= htmlWebpackPlugin.options.title %></title>  
</head>  
<body>  
<noscript>  
 <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled.  
 Please enable it to continue.</strong>  
</noscript>  
<div id="app"></div>  
</body>  
</html>

***Файл storage/library.json***

[  
 {  
 "title": "Белые Ночи",  
 "author": "Достоевский Ф.М.",  
 "year": 1848,  
 "holder": "Анатолий",  
 "returnDate": "2022-10-08",  
 "cover": "cover1.jpg"  
 },  
 {  
 "title": "О дивный новый мир",  
 "author": "Хаксли О.",  
 "year": 1932,  
 "holder": null,  
 "returnDate": null  
 },  
 {  
 "title": "Отцы и дети",  
 "author": "Тургенев И.С.",  
 "year": 1862,  
 "holder": "Вадим",  
 "returnDate": "2022-08-13"  
 }  
]