МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Web-технологии»
Тема: Rest-приложение управления библиотекой

Студент гр. 3343 _____ Пименов П.В.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2025

Задание

Необходимо создать web-приложение управления домашней библиотекой, которое предоставляет список книг, их можно отфильтровать по признакам «в наличии», «возврат просрочен», есть возможность выдать книгу для чтения и вернуть книгу. Основные требования следующие:

1. Начальное и текущее состояние библиотеки хранится в JSON-файле на сервере.

2. В качестве сервера используется Node.JS с модулем express.

3. В качестве модуля управления шаблонами HTML-страниц используется pug либо ejs, все web-страницы должны быть сделаны с использованием pug либо ejs.

4. Предусмотрена страница для списка книг, в списке предусмотрена фильтрация по дате возврата и признаку «в наличии», предусмотрена возможность добавления и удаления книг. Удаление книг – с подтверждением.

5. Предусмотрена страница для карточки книги, в которой ее можно отредактировать (минимум: автор, название, дата выпуска) и выдать читателю или вернуть в библиотеку. В карточке книги должно быть очевидно: находится ли книга в библиотеке, кто ее взял (имя) и когда должен вернуть (дата).

6. Информация о читателе вводится с использованием всплывающего модального диалогового окна (<dialog>).

7. Оформление страниц выполнено с использованием CSS (допустимо использование w3.css).

8. Взаимодействие между браузером и web-сервером осуществляется с использованием REST.

9. Фильтрация списка книг осуществляется с использованием AJAXзапросов.

10. Логика приложения реализована на языке JavaScript (либо TypeScript).

11. Для всех страниц web-приложения разработан макет интерфейса с использованием Figma (https://www.figma.com/).

12. При оформлении элементов управления используются иконки (например, Font Awesome, https://fontawesome.ru/all-icons/). Преимуществом будет создание и использование аутентификации на основе passport.js (http://www.passportjs.org/), в качестве примера можно использовать https://nodejsdev.ru/guides/webdraftt/authentication/. Преимуществом будет реализация загрузки и отображения обложек книг

Выполнение работы

Разработано простое web-приложение «Домашняя библиотека» на Node.js + Express с использованием EJS в качестве шаблонизатора. Состояние библиотеки (начальное и текущее) хранится в JSON-файле на сервере (data/books.json). Приложение предоставляет REST API для управления книгами, веб-интерфейс для просмотра списка и карточки книги, AJAX-фильтрацию списка и модальное окно <dialog> для ввода данных читателя. Присутствует минимальная локальная аутентификация через passport-local (требуется для операций добавления/редактирования/удаления/выдачи/возврата).

Структура проекта

- app.js — точка входа приложения, настройка Express, сессий, passport, рендеринга и маршрутов.
- package.json — зависимости и скрипты запуска.
- Dockerfile, docker-compose.yml, .dockerignore — контейнеризация приложения.
- /routes
  - index.js — роуты для веб-страниц (рендер EJS), обработка форм (создать/редактировать книгу), маршруты для логина/логаута.
  - api.js — REST API:
    - /api/books (GET/POST/PUT/DELETE),
    - /api/books/:id/checkout
    - /api/books/:id/return.
- utils.js — вспомогательные функции (например, ensureAuth).
  - /views — EJS-шаблоны:
- layout.ejs — базовый шаблон (шапка, подключение стилей/скриптов).
- books.ejs — страница списка книг (фильтры, контейнер для AJAX-загрузки списка).
- book.ejs — карточка книги / форма редактирования / выдача / возврат.
- login.ejs — страница входа.

- /public
  - /css/style.css — минималистичный стиль CSS.
  - /js/books.js — клиентская логика: AJAX-запросы для фильтрации и отрисовки списка.
- /covers — директория для загруженных обложек книг.
- /data
  - books.json — JSON-файл с начальным списком книг (и место хранения текущего состояния).
- /sessions — директория для хранения файлов сессий (используется session-file-store).

Ключевые компоненты и функции

app.js:

Настройка EJS и express-ejs-layouts. Настройка статической отдачи /public. Подключение session-file-store для хранения сессий (замена MemoryStore). Инициализация passport-local с тестовым пользователем. Подключение роутеров (/ и /api). Глобальная обработка ошибок (в режиме разработки выдает стек ошибки для отладки).

routes/api.js:

GET /api/books — возвращает список книг; поддерживает фильтры через query: available, dueBefore, dueAfter, search (по автору/названию).

GET /api/books/:id — получить одну книгу.

POST /api/books — создать книгу (защищён).

PUT /api/books/:id — обновить книгу (защищён).

DELETE /api/books/:id — удалить книгу (защищён).

POST /api/books/:id/checkout — выдать книгу: требуется holder и dueDate (защищён).

POST /api/books/:id/return — вернуть книгу (защищён).

routes/index.js:

Рендер страниц /books, /books/new, /books/:id, /login. Обработка multipart/form-data для загрузки обложек (multer). Работа с data/books.json: чтение/запись (сериализация в JSON).

public/js/books.js:

Выполняет AJAX-запросы к /api/books по параметрам фильтрации и отрисовывает список.

views/book.ejs:

Форма редактирования (title, author, year, загрузка cover). Кнопки выдачи/возврата с модальным <dialog> для ввода имени читателя и даты возврата. Кнопка удаления с подтверждением.

data/books.json:

Набор тестовых книг (включён расширенный список ~20 записей).

Запуск

В Docker:

docker-compose up --build

Открыть http://localhost:3000/books

Примонтированы директории data и public/covers (чтобы данные и обложки были персистентны между перезапусками).

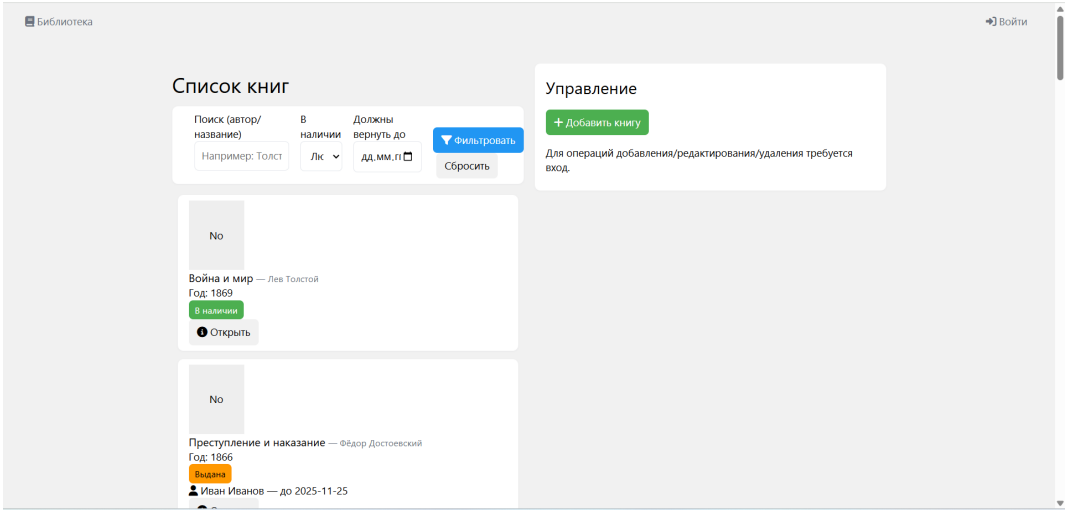

Рисунок 1 – Главная страница со списком книг

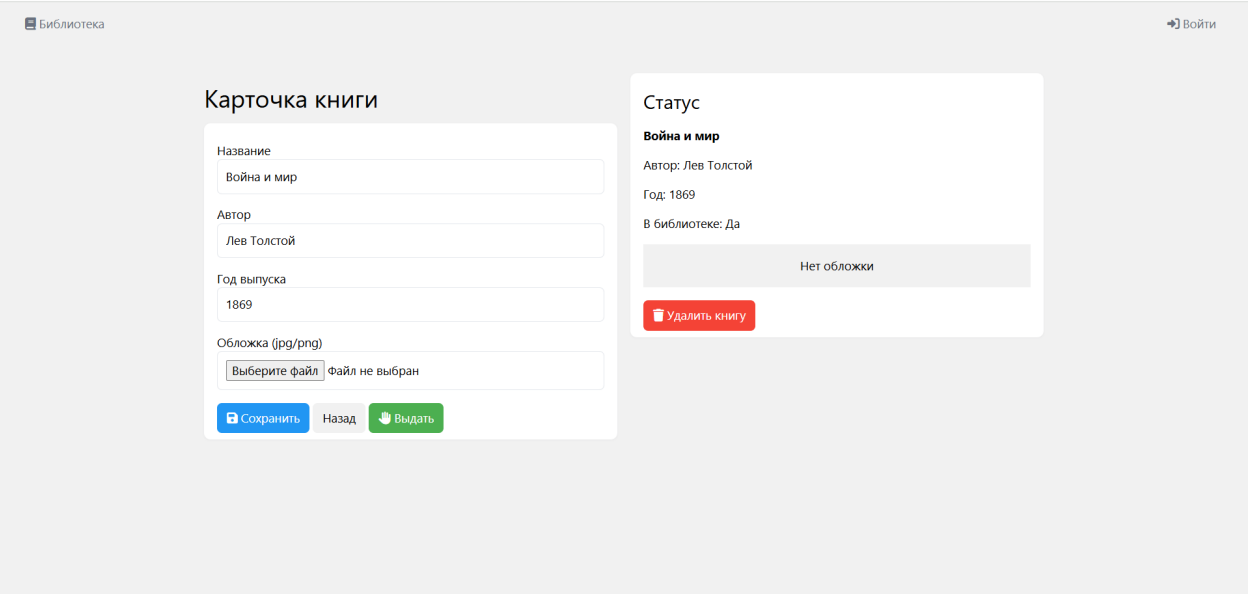

Рисунок 2 – Карточка книги

Вывод

Разработано веб-приложение управления домашней библиотекой, которое реализует основные требования лабораторной работы: хранение состояния в JSON, сервер на Node.js/Express, шаблоны EJS, REST API, AJAX-фильтрация, страницы списка и карточки книги с возможностью выдачи/возврата и модальным вводом данных читателя.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

public/css/style.css:

```
:root{
    --bg: #fafafa;
    --card: #ffffff;
    --muted: #6b7280;
    --accent: #0b74de;
    --success: #16a34a;
    --danger: #dc2626;
    --radius: 8px;
    --gap: 12px;
    --shadow: 0 1px 2px rgba(16,24,40,0.04);
    font-size: 15px;
}

* { box-sizing: border-box; }
body {
    margin: 0;
    background: var(--bg);
    color: #111827;
    font-family: Inter, "Segoe UI", Roboto, system-ui, -apple-system,
"Helvetica Neue", Arial;
    -webkit-font-smoothing:antialiased;
    -moz-osx-font-smoothing:grayscale;
    line-height:1.4;
    padding-bottom: 40px;
}

/* Header */
.w3-bar { background: transparent !important; box-shadow:none;
padding:18px 20px; }
.w3-bar .w3-button { color: var(--muted); text-decoration:none; margin-
right:8px; }
.w3-bar .w3-button:hover { color: var(--accent); }

/* Container */
.w3-container { max-width: 1100px; margin: 0 auto; padding: 18px; }

/* Card */
```

```
      .w3-card { background: var(--card); border-radius: var(--radius); box-
shadow: var(--shadow); }

      /* Filters */
      #filterForm .w3-input, #filterForm .w3-select {
          border: 1px solid #e6e9ee; padding:10px; border-radius:6px;
background: transparent;
      }

      /* Book list */
      #booksList .book-card { margin-bottom: var(--gap); padding: 12px;
display:flex; gap: 12px; align-items:center; }
      .book-cover-thumb {
          width: 72px; height: 96px; object-fit:cover; border-radius:6px;
background:#f3f4f6; flex-shrink:0;
          border:1px solid #eef2f6;
      }
      .book-item h4 { margin: 0 0 6px 0; font-size: 16px; }
      .book-item small { color: var(--muted); font-size: 13px; }

      /* Tags */
      .w3-tag { border-radius: 6px; padding: 4px 8px; font-size: 13px;
display:inline-block; }
      .w3-tag.w3-green { background: rgba(22,163,74,0.12); color: var(--
success); border: 1px solid rgba(22,163,74,0.12); }
      .w3-tag.w3-orange { background: rgba(249,115,22,0.08); color: #f97316;
border: 1px solid rgba(249,115,22,0.08); }

      /* Buttons */
      .w3-button { border: none; background: transparent; color: var(--
accent); padding: 8px 12px; cursor:pointer; border-radius:6px; }
      .w3-button.w3-blue, .w3-button.w3-green, .w3-button.w3-red, .w3-
button.w3-orange { color: #fff; }
      .w3-button.w3-blue { background: var(--accent); }
      .w3-button.w3-green { background: var(--success); }
      .w3-button.w3-red { background: var(--danger); }
      .w3-button.w3-light-grey { background: #f3f4f6; color: var(--muted); }

      /* Forms */
      input[type="text"], input[type="date"], input[type="file"], .w3-input {
          width:100%; padding:10px; border:1px solid #e6e9ee; border-
radius:6px; background:#fff;
      }
```

```
/* Page layout tweaks */
.w3-row { display:flex; gap: 16px; align-items:flex-start; }
.w3-col { flex:1; }

/* Small screens */
@media (max-width:800px){
    .w3-row { flex-direction:column; }
    .w3-container { padding: 12px; }
    .book-cover-thumb { width: 56px; height:80px; }
}
```

docker-compose.yml:

```
services:
  web:
    build: .
    container_name: lb2
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - ./data:/app/data
      - ./public/covers:/app/public/covers
      - ./public:/app/public
    environment:
      - PORT=3000
```

Dockerfile:

```
FROM node:22-alpine

WORKDIR /app

COPY package*.json ./

RUN npm clean-install

COPY . .

RUN mkdir -p public/covers data

EXPOSE 3000

ENV NODE_ENV=production
```

```
CMD ["node", "app.js"]
```

public/js/books.js:

```
// Client-side script: loads list via AJAX and handles filtering,
deletion via dialog
document.addEventListener('DOMContentLoaded', function(){
    const listEl = document.getElementById('booksList');
    const filterForm = document.getElementById('filterForm');
    const applyBtn = document.getElementById('applyFilter');
    const resetBtn = document.getElementById('resetFilter');

    async function loadBooks() {
        const params = new URLSearchParams();
        const q = document.getElementById('search').value.trim();
        if (q) params.append('search', q);
        const available = document.getElementById('available').value;
        if (available !== '') params.append('available', available);
        const dueBefore = document.getElementById('dueBefore').value;
        if (dueBefore) params.append('dueBefore', dueBefore);

        const res = await fetch('/api/books?' + params.toString());
        const books = await res.json();
        renderList(books);
    }

    function renderList(books) {
        if (!books || !books.length) {
            listEl.innerHTML = '<div class="w3-panel w3-white w3-
padding">Книги не найдены.</div>';
            return;
        }
        const rows = books.map(b => {
            const cover = b.cover ? `<img src="${b.cover}" class="book-
cover-thumb" />` : `<div
style="width:80px;height:100px;background:#EEE;display:flex;align-
items:center;justify-content:center">No</div>`;
            const holder = b.available ? '' : `<div><i class="fa fa-
user"></i> ${b.holder} — до ${b.dueDate}</div>`;
            const actions = `
        <div class="book-actions">
          <a class="w3-button w3-light-grey" href="/books/${b.id}"><i
class="fa fa-info-circle"></i> Открыть</a>
```

```
            </div>`;
                return `<div class="w3-card w3-padding book-card">
            <div class="book-item">
              ${cover}
              <div style="flex:1">
                <h4 style="margin:0">${b.title} <small>—
${b.author}</small></h4>
                <div>Год: ${b.year || '-'}</div>
                <div>${b.available ? '<span class="w3-tag w3-green">В
наличии</span>' : '<span class="w3-tag w3-orange">Выдана</span>'} ${holder}
</div>
              </div>
              ${actions}
            </div>
          </div>`;
            });
            listEl.innerHTML = rows.join('');
        }

        applyBtn.addEventListener('click', (e) => { e.preventDefault();
loadBooks(); });
        resetBtn.addEventListener('click', (e) => {
            e.preventDefault();
            filterForm.reset();
            loadBooks();
        });

        // initial load
        loadBooks();
    });
```

routes/api.js:

```
const express = require('express');
const router = express.Router();
const path = require('path');
const fs = require('fs');
const { v4: uuidv4 } = require('uuid');
const { ensureAuth } = require('./utils');

const DATA_DIR = path.join(__dirname, '..', 'data');
const BOOKS_FILE = path.join(DATA_DIR, 'books.json');

function readBooks() {
```

```
        const raw = fs.readFileSync(BOOKS_FILE, 'utf8');
        return JSON.parse(raw || '[]');
    }
    function writeBooks(data) {
        fs.writeFileSync(BOOKS_FILE, JSON.stringify(data, null, 2), 'utf8');
    }

    // GET /api/books - supports ?available=true/false & dueBefore=YYYY-MM-
DD & dueAfter=YYYY-MM-DD
    router.get('/books', (req, res) => {
        let books = readBooks();
        const { available, dueBefore, dueAfter, search } = req.query;
        if (available !== undefined) {
            const av = available === 'true';
            books = books.filter(b => !!b.available === av);
        }
        if (dueBefore) {
            const before = new Date(dueBefore);
            books = books.filter(b => b.dueDate && new Date(b.dueDate) <=
before);
        }
        if (dueAfter) {
            const after = new Date(dueAfter);
            books = books.filter(b => b.dueDate && new Date(b.dueDate) >=
after);
        }
        if (search) {
            const s = search.toLowerCase();
            books = books.filter(b =>
                (b.title && b.title.toLowerCase().includes(s)) ||
                (b.author && b.author.toLowerCase().includes(s))
            );
        }
        res.json(books);
    });

    // GET one
    router.get('/books/:id', (req, res) => {
        const books = readBooks();
        const book = books.find(b => b.id === req.params.id);
        if (!book) return res.status(404).json({ error: 'Not found' });
        res.json(book);
    });
```

```
    // Create (protected)
    router.post('/books', ensureAuth, (req, res) => {
        const books = readBooks();
        const { title, author, year } = req.body;
        const newBook = {
            id: uuidv4(),
            title: title || 'Untitled',
            author: author || 'Unknown',
            year: year || '',
            available: true,
            holder: null,
            dueDate: null,
            cover: null
        };
        books.push(newBook);
        writeBooks(books);
        res.status(201).json(newBook);
    });

    // Update (protected)
    router.put('/books/:id', ensureAuth, (req, res) => {
        const books = readBooks();
        const book = books.find(b => b.id === req.params.id);
        if (!book) return res.status(404).json({ error: 'Not found' });
        const { title, author, year } = req.body;
        if (title !== undefined) book.title = title;
        if (author !== undefined) book.author = author;
        if (year !== undefined) book.year = year;
        writeBooks(books);
        res.json(book);
    });

    // Delete (protected)
    router.delete('/books/:id', ensureAuth, (req, res) => {
        let books = readBooks();
        const beforeCount = books.length;
        books = books.filter(b => b.id !== req.params.id);
        if (books.length === beforeCount) return
res.status(404).json({ error: 'Not found' });
        writeBooks(books);
        res.json({ success: true });
    });
```

```
// Checkout (issue book to reader) (protected)
router.post('/books/:id/checkout', ensureAuth, (req, res) => {
    const books = readBooks();
    const book = books.find(b => b.id === req.params.id);
    if (!book) return res.status(404).json({ error: 'Not found' });
    const { holder, dueDate } = req.body;
    if (!holder || !dueDate) return res.status(400).json({ error:
'holder and dueDate required' });
    if (!book.available) return res.status(400).json({ error: 'Book not
available' });
    book.available = false;
    book.holder = holder;
    book.dueDate = dueDate;
    writeBooks(books);
    res.json(book);
});

// Return book (protected)
router.post('/books/:id/return', ensureAuth, (req, res) => {
    const books = readBooks();
    const book = books.find(b => b.id === req.params.id);
    if (!book) return res.status(404).json({ error: 'Not found' });
    book.available = true;
    book.holder = null;
    book.dueDate = null;
    writeBooks(books);
    res.json(book);
});

module.exports = router;
```

routes/index.js:

```
const express = require('express');
const router = express.Router();
const path = require('path');
const fs = require('fs');
const multer = require('multer');
const { v4: uuidv4 } = require('uuid');
const ensureAuth = require('./utils').ensureAuth;

const DATA_DIR = path.join(__dirname, '..', 'data');
const BOOKS_FILE = path.join(DATA_DIR, 'books.json');
```

```
const COVERS_DIR = path.join(__dirname, '..', 'public', 'covers');

if (!fs.existsSync(COVERS_DIR)) fs.mkdirSync(COVERS_DIR, { recursive:
true });

const storage = multer.diskStorage({
    destination: (req, file, cb) => cb(null, COVERS_DIR),
    filename: (req, file, cb) => {
        const id = uuidv4();
        const ext = path.extname(file.originalname);
        cb(null, id + ext);
    }
});
const upload = multer({ storage });

function readBooks() {
    const raw = fs.readFileSync(BOOKS_FILE, 'utf8');
    return JSON.parse(raw || '[]');
}
function writeBooks(data) {
    fs.writeFileSync(BOOKS_FILE, JSON.stringify(data, null, 2), 'utf8');
}

// Home -> redirect to /books
router.get('/', (req, res) => res.redirect('/books'));

// Books list page (front-end)
router.get('/books', (req, res) => {
    res.render('books');
});

// New book form (protected)
router.get('/books/new', ensureAuth, (req, res) => {
    res.render('book', { book: null, errors: [] });
});

// Book detail / edit page
router.get('/books/:id', (req, res) => {
    const books = readBooks();
    const book = books.find(b => b.id === req.params.id);
    if (!book) return res.status(404).send('Not found');
    res.render('book', { book, errors: [] });
});
```

```
    // Create book (multipart for cover) - protected
    router.post('/books', ensureAuth, upload.single('cover'), (req, res) =>
{
        const books = readBooks();
        const { title, author, year } = req.body;
        const id = uuidv4();
        const cover = req.file ? '/public/covers/' + req.file.filename :
null;
        const newBook = {
            id, title: title || 'Untitled', author: author || 'Unknown',
year: year || '',
            available: true, holder: null, dueDate: null, cover
        };
        books.push(newBook);
        writeBooks(books);
        res.redirect('/books/' + id);
    });

    // Update book (multipart) - protected
    router.post('/books/:id', ensureAuth, upload.single('cover'), (req, res)
=> {
        const books = readBooks();
        const book = books.find(b => b.id === req.params.id);
        if (!book) return res.status(404).send('Not found');
        const { title, author, year } = req.body;
        book.title = title || book.title;
        book.author = author || book.author;
        book.year = year || book.year;
        if (req.file) {
            book.cover = '/public/covers/' + req.file.filename;
        }
        writeBooks(books);
        res.redirect('/books/' + book.id);
    });

    // Simple login/logout routes
    const passport = require('passport');
    router.get('/login', (req, res) => {
        res.render('login');
    });
    router.post('/login', passport.authenticate('local', {
        successRedirect: '/books',
```

```
        failureRedirect: '/login',
        failureFlash: true
    }));
    router.get('/logout', (req, res, next) => {
        req.logout(function(err) {
            if (err) return next(err);
            res.redirect('/books');
        });
    });

    module.exports = router;
```

routes/utils.js:

```
    // Utilities for routers
    module.exports = {
        ensureAuth: function(req, res, next) {
            // For simplicity: allow read-only endpoints without auth,
protect mutation endpoints.
            if (req.isAuthenticated && req.isAuthenticated()) return next();
            // If AJAX request, return 401 JSON, else redirect to login
            const acceptsJson = req.headers['accept'] &&
req.headers['accept'].includes('application/json');
            if (acceptsJson || req.xhr) return res.status(401).json({ error:
'Unauthorized' });
            return res.redirect('/login');
        }
    };
```

views/book.ejs:

```
    <div class="w3-row">
        <div class="w3-col m8">
            <h2><%= book ? 'Карточка книги' : 'Добавить книгу' %></h2>

            <form action="<%= book ? ('/books/' + book.id) : '/books' %>"
method="post" enctype="multipart/form-data" class="w3-container w3-card w3-
padding">
                <p>
                    <label>Название</label>
                    <input class="w3-input" name="title" value="<%= book ?
book.title : '' %>">
                </p>
                <p>
                    <label>Автор</label>
```

```
                    <input class="w3-input" name="author" value="<%= book ?
book.author : '' %>">
                </p>
                <p>
                    <label>Год выпуска</label>
                    <input class="w3-input" name="year" value="<%= book ?
book.year : '' %>">
                </p>
                <p>
                    <label>Обложка (jpg/png)</label>
                    <input type="file" name="cover" accept="image/*">
                </p>

                <div class="w3-margin-top">
                    <button class="w3-button w3-blue" type="submit"><i
class="fa fa-save"></i> Сохранить</button>
                    <a href="/books" class="w3-button w3-light-
grey">Назад</a>

                    <% if (book) { %>
                        <!-- Checkout / Return buttons -->
                        <% if (book.available) { %>
                            <button id="openCheckout" type="button"
class="w3-button w3-green"><i class="fa fa-hand-paper"></i> Выдать</button>
                        <% } else { %>
                            <form id="returnForm" style="display:inline"
method="post" action="/api/books/<%= book.id %>/return?_method=POST"></form>
                            <button id="doReturn" type="button" class="w3-
button w3-orange"><i class="fa fa-undo"></i> Вернуть</button>
                        <% } %>
                    <% } %>
                </div>
            </form>
        </div>

        <div class="w3-col m4">
            <div class="w3-card w3-padding">
                <h3>Статус</h3>
                <% if (book) { %>
                    <p><strong><%= book.title %></strong></p>
                    <p>Автор: <%= book.author %></p>
                    <p>Год: <%= book.year %></p>
```

```
                    <p>В библиотеке: <%= book.available ? 'Да' : 'Нет'
%></p>
                    <% if (!book.available) { %>
                        <p>Взял: <strong><%= book.holder %></strong></p>
                        <p>Должен вернуть: <strong><%= book.dueDate
%></strong></p>
                    <% } %>
                    <% if (book.cover) { %>
                        <img src="<%= book.cover %>" alt="cover" style="max-
width:100%">
                    <% } else { %>
                        <div class="w3-center w3-padding-16 w3-light-
grey">Нет обложки</div>
                    <% } %>
                    <form id="deleteForm" method="post"
action="/api/books/<%= book.id %>?_method=DELETE"></form>
                    <button id="deleteBtn" class="w3-button w3-red w3-
margin-top"><i class="fa fa-trash"></i> Удалить книгу</button>
                <% } else { %>
                    <div class="w3-center w3-text-grey">Сохраните книгу,
затем можно будет выдать/удалить.</div>
                <% } %>
            </div>
        </div>
    </div>

    <!-- Checkout dialog -->
    <dialog id="checkoutDialog">
        <form method="dialog" class="w3-container">
            <h3>Выдать книгу</h3>
            <p>
                <label>Имя читателя</label>
                <input id="holderName" class="w3-input" required>
            </p>
            <p>
                <label>Дата возврата</label>
                <input id="dueDate" type="date" class="w3-input" required>
            </p>
            <div class="w3-right">
                <button id="checkoutCancel" class="w3-button w3-light-
grey">Отмена</button>
                <button id="checkoutOk" class="w3-button
w3-green">Выдать</button>
```

```
            </div>
        </form>
    </dialog>

    <script>
        (function(){
            const bookId = "<%= book ? book.id : '' %>";
            const deleteBtn = document.getElementById('deleteBtn');
            if (deleteBtn) {
                deleteBtn.addEventListener('click', async () => {
                    if (confirm('Удалить книгу?')) {
                        const resp = await fetch('/api/books/' + bookId,
{ method: 'DELETE' });
                        if (resp.ok) location.href = '/books';
                        else alert('Ошибка при удалении');
                    }
                });
            }

            const openCheckout = document.getElementById('openCheckout');
            if (openCheckout) {
                openCheckout.addEventListener('click', () => {
                    document.getElementById('checkoutDialog').showModal();
                });
            }
            const checkoutOk = document.getElementById('checkoutOk');
            if (checkoutOk) {
                checkoutOk.addEventListener('click', async () => {
                    const holder =
document.getElementById('holderName').value.trim();
                    const dueDate =
document.getElementById('dueDate').value;
                    if (!holder || !dueDate) { alert('Заполните данные');
return; }
                    const resp = await fetch('/api/books/' + bookId +
'/checkout', {
                        method: 'POST',
                        headers: { 'Content-Type': 'application/json' },
                        body: JSON.stringify({ holder, dueDate })
                    });
                    if (resp.ok) location.reload();
                    else {
                        const err = await resp.json();
```

```
                        alert('Ошибка: ' + (err.error||''));
                    }
                });
            }

            const doReturn = document.getElementById('doReturn');
            if (doReturn) {
                doReturn.addEventListener('click', async () => {
                    const resp = await fetch('/api/books/' + bookId +
'/return', { method: 'POST' });
                    if (resp.ok) location.reload();
                    else alert('Ошибка при возврате');
                });
            }
        })();
    </script>
```

views/books.ejs:

```
    <div class="w3-row">
        <div class="w3-col m8 l9">
            <h2>Список книг</h2>

            <div class="w3-card w3-padding w3-margin-bottom">
                <form id="filterForm" class="w3-row-padding">
                    <div class="w3-col s12 m4">
                        <label>Поиск (автор/название)</label>
                        <input id="search" class="w3-input" name="search"
placeholder="Например: Толстой">
                    </div>
                    <div class="w3-col s6 m2">
                        <label>В наличии</label>
                        <select id="available" class="w3-select"
name="available">
                            <option value="">Любой</option>
                            <option value="true">Да</option>
                            <option value="false">Нет</option>
                        </select>
                    </div>
                    <div class="w3-col s6 m3">
                        <label>Должны вернуть до</label>
                        <input id="dueBefore" type="date" class="w3-input"
name="dueBefore">
                    </div>
```

```
                    <div class="w3-col s12 m3 w3-center" style="padding-top:
22px;">
                        <button id="applyFilter" class="w3-button w3-
blue"><i class="fa fa-filter"></i> Фильтровать</button>
                        <button id="resetFilter" type="button" class="w3-
button w3-light-grey">Сбросить</button>
                    </div>
                </form>
            </div>

            <div id="booksList" class="w3-row-padding"></div>
        </div>

        <div class="w3-col m4 l3">
            <div class="w3-card w3-padding">
                <h3>Управление</h3>
                <p>
                    <a href="/books/new" class="w3-button w3-green"><i
class="fa fa-plus"></i> Добавить книгу</a>
                </p>
                <p>
                    Для операций добавления/редактирования/удаления
требуется вход.
                </p>
            </div>
        </div>
    </div>

    <!-- Dialogs for confirmation -->
    <dialog id="confirmDialog">
        <form method="dialog" class="w3-container">
            <p id="confirmText">Вы уверены?</p>
            <div class="w3-right">
                <button id="confirmCancel" class="w3-button w3-light-
grey">Отмена</button>
                <button id="confirmOk" class="w3-button
w3-red">Удалить</button>
            </div>
        </form>
    </dialog>
```

views/layout.ejs:

```
<!DOCTYPE html>
```

```
<html lang="ru">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,initial-
scale=1" />
    <title>Домашняя библиотека</title>
    <!-- W3.CSS -->
    <link rel="stylesheet"
href="https://www.w3schools.com/w3css/4/w3.css">
    <!-- Font Awesome -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/
all.min.css">
    <link rel="stylesheet" href="/public/css/style.css">
</head>
<body class="w3-light-grey">
<div class="w3-bar w3-dark-grey w3-padding">
    <a href="/books" class="w3-bar-item w3-button"><i class="fa fa-
book"></i> Библиотека</a>
    <div class="w3-right">
        <% if (currentUser) { %>
            <span class="w3-bar-item">Привет, <%=
currentUser.displayName %></span>
            <a href="/logout" class="w3-bar-item w3-button"><i class="fa
fa-sign-out-alt"></i> Выйти</a>
        <% } else { %>
            <a href="/login" class="w3-bar-item w3-button"><i class="fa
fa-sign-in-alt"></i> Войти</a>
        <% } %>
    </div>
</div>

<div class="w3-container w3-margin-top">
    <% if (messages && messages.error) { %>
        <div class="w3-panel w3-pale-red w3-border">
            <ul>
                <% messages.error.forEach(function(m){ %><li><%= m
%></li><% }); %>
            </ul>
        </div>
    <% } %>

    <%- body %>
```

```
</div>

<script src="/public/js/books.js" defer></script>
</body>
</html>
```

views/login.ejs:

```
<div class="w3-card w3-padding" style="max-width:400px;margin:auto">
    <h3>Вход</h3>
    <form action="/login" method="post" class="w3-container">
        <p>
            <label>Логин</label>
            <input class="w3-input" name="username">
        </p>
        <p>
            <label>Пароль</label>
            <input class="w3-input" name="password" type="password">
        </p>
        <p>
            <button class="w3-button w3-blue"
type="submit">Войти</button>
            <a href="/books" class="w3-button w3-light-grey">Отмена</a>
        </p>
    </form>
</div>
```

.docker-ignore:

```
node_modules
npm-debug.log
data/*.lock
```

app.js:

```
const express = require('express');
const path = require('path');
const bodyParser = require('body-parser');
const session = require('express-session');
const FileStoreFactory = require('session-file-store'); // <-- добавлено
const methodOverride = require('method-override');
const passport = require('passport');
const LocalStrategy = require('passport-local').Strategy;
const flash = require('connect-flash');
const fs = require('fs');
const expressLayouts = require('express-ejs-layouts');
```

```
const apiRouter = require('./routes/api');
const indexRouter = require('./routes/index');

const DATA_DIR = path.join(__dirname, 'data');
const BOOKS_FILE = path.join(DATA_DIR, 'books.json');

if (!fs.existsSync(DATA_DIR)) fs.mkdirSync(DATA_DIR);
if (!fs.existsSync(BOOKS_FILE)) fs.writeFileSync(BOOKS_FILE, '[]',
'utf8');

const app = express();
const PORT = process.env.PORT || 3000;

// View engine
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');

// express-ejs-layouts middleware
app.use(expressLayouts);
app.set('layout', 'layout');

// Static
app.use('/public', express.static(path.join(__dirname, 'public')));

// Body parser
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(methodOverride('_method'));

// Session + passport
// Используем file store вместо MemoryStore, чтобы убрать предупреждение
про MemoryStore
const FileStore = FileStoreFactory(session);
const sessionStoreOptions = {
    path: path.join(__dirname, 'sessions'),
    logFn: function() {} // чтобы не было лишних логов от FileStore
};

app.use(session({
    store: new FileStore(sessionStoreOptions),
    secret: process.env.SESSION_SECRET || 'secret-home-library',
    resave: false,
```

```
        saveUninitialized: false,
        cookie: { maxAge: 1000 * 60 * 60 * 24 } // 1 day
    }));
    app.use(flash());
    app.use(passport.initialize());
    app.use(passport.session());

    // Simple local strategy with single user (for demo). Change as needed.
    // Можно заменить чтением из env: process.env.ADMIN_USER / ADMIN_PASS
    const USERS = [
        { id: '1', username: process.env.ADMIN_USER || 'admin', password:
process.env.ADMIN_PASS || 'password', displayName: 'Admin' }
    ];
    passport.use(new LocalStrategy((username, password, done) => {
        const user = USERS.find(u => u.username === username);
        if (!user) return done(null, false, { message: 'Incorrect
username' });
        if (user.password !== password) return done(null, false, { message:
'Incorrect password' });
        return done(null, user);
    }));
    passport.serializeUser((user, done) => done(null, user.id));
    passport.deserializeUser((id, done) => {
        const user = USERS.find(u => u.id === id);
        done(null, user || null);
    });

    // Make user available in templates
    app.use((req, res, next) => {
        res.locals.currentUser = req.user;
        res.locals.messages = req.flash();
        next();
    });

    // Routers
    app.use('/', indexRouter);
    app.use('/api', apiRouter);

    // Debug error handler (show stack in dev)
    app.use((err, req, res, next) => {
        console.error('Unhandled error:', err && (err.stack || err));
        const stack = err && (err.stack || String(err));
```

```
        if (req.path.startsWith('/api') || (req.headers.accept &&
req.headers.accept.includes('application/json'))) {
            return res.status(500).json({ error: 'Internal Server Error',
detail: stack });
        }
        res.status(500).send(`<h1>Internal Server
Error</h1><pre>${stack}</pre>`);
    });

    // Start server
    app.listen(PORT, () => {
        console.log(`Home Library app listening at http://0.0.0.0:${PORT}
(NODE_ENV=${process.env.NODE_ENV || 'production'})`);
    });
```

package.json:

```
{
    "name": "lb2",
    "version": "1.0.0",
    "main": "app.js",
    "scripts": {
        "start": "node app.js",
        "dev": "nodemon app.js"
    },
    "dependencies": {
        "body-parser": "^1.20.3",
        "connect-flash": "^0.1.1",
        "ejs": "^3.1.10",
        "express": "^4.21.2",
        "express-session": "^1.18.2",
        "method-override": "^3.0.0",
        "multer": "2.0.2",
        "passport": "^0.7.0",
        "passport-local": "^1.0.0",
        "uuid": "^9.0.1",
        "express-ejs-layouts": "^2.5.1",
        "session-file-store": "^1.5.0"
    },
    "devDependencies": {
        "nodemon": "^3.1.11"
    }
}
```