УДК 004.032.24+004.422.635+004.451.83

Р. Р. Іваськів, Львів
Т. В. Нерода, Львів

МОДЕЛЬ ВИКОНАННЯ СЦЕНАРІЇВ ПРИ ПРОЕКТУВАННІ СЕРВЕРНОЇ ЧАСТИНИ КОМП'ЮТЕРИЗОВАНОЇ БІБЛІОТЕЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Abstract. The architectural solution is substantiated and the optimal scripts configuration for deployment of unified information platform of computerized system for distributed library services is well-grounded. Constructed database connection interfaces are built, also the server program part to provide flexible system expansion when compiling end-user terminals software is designed.

Вступ. Всебічна інформатизація практично усіх галузей людської діяльності вимагає постійної оптимізації прискорення та спрощення доступу до затребуваних ресурсів. Водночас з розширенням інтерактивних можливостей користувачів зростають вимоги до підвищення продуктивності програмних середовищ, що може бути забезпечене шляхом їх інтеграції у єдину інформаційну платформу. Такий стан речей потребує значних зусиль розробників на етапі проектування, а також матеріальних і технічних затрат в експлуатації проектів, різко обмежуючи гнучкість їх розгортання й переорієнтування на інший рід діяльності.

Тому, бізнес-логіка переважної більшості корпоративних інформаційних платформ в основному ґрунтується на централізованому розташуванні декількох жорстко пов'язаних функціональних комплексів з пропрієтарним програмним забезпеченням різного цільового призначення. Зокрема, для реєстрації та відстеження матеріальних потоків у бібліотечних закладах використовуються інформаційні бібліотечні системи.

Аналіз публікацій та постановка задачі. На сьогодні основна увага дослідників зосереджена на пошуку шляхів подальшої комп'ютеризації бібліотечних процесів оцифрування сховищ, розширення застосовуваних інформаційних технологій у придбанні фондів та впровадження телекомунікаційних засобів для обслуговування користувачів тощо [1-6]. Як зазначалось, наведені види діяльності підтримуються розокремленими компонентами, які дублюють деякі стадії, а деякі не виконують взагалі. Також, наявні джерела не наводять аналіз модельного представлення компонентів, що не дозволяє виявити надлишковий функціонал та не забезпечує створення окремих рішень для інтерфейсів кінцевого користувача. Таким чином, постає задача інтегрування обумовлених середовищ в уніфікований інформаційний простір науково-технічної бібліотеки закладу вищої освіти, що дозволить охопити широкий спектр інформаційних потреб кінцевих користувачів як суб'єктів єдиного академічного середовища.

Вирішення задачі. Представлений проект клієнт-серверної комп'ютеризованої інформаційної бібліотечної системи реалізовано за принципами односторінкового застосунку (SPA), з фронтендом на кінцевому терміналі, з яким працює користувач, бекендом, розміщеним з боку сервера. При розгортанні проекту було прийняте рішення не пов'язувати ці компоненти напряму, а організувати обмін даними між ними за викликом віддалених процедур на основі REST API: з бекенду надсилаються лише запитувані дані (data), натомість фронтенд для визначення прав доступу поточного користувача в пакеті Header HTTP запиту надсилає ключ доступу (токен), попередньо згенерований на сервері при логуванні користувача в системі з використанням файлу *LoginController.js* (рис. 1). Застосований підхід до проектування архітектури дозволить гнучко масштабувати систему під інші способи входу [7], а також надасть змогу розширювати фронтенд точки входу для користувача [8].

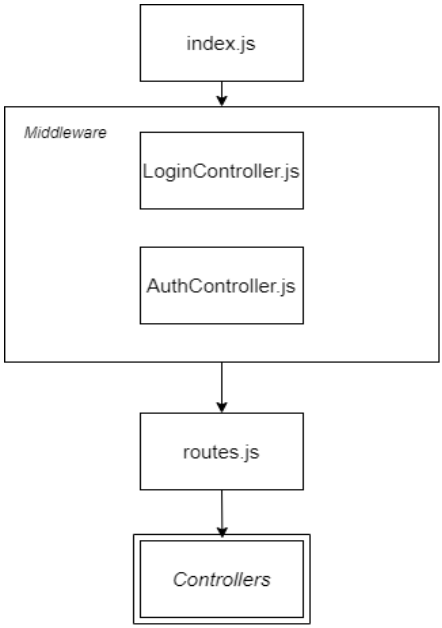

Рис. 1. Пакетування відомостей про автентифікацію користувача

У файлі *index.js* як у вхідній точці системи здійснюється підключення основних модулів і програмних бібліотек, необхідних для функціонування інформаційної системи: модулі підімкнення до баз даних, підімкнення маршрутизації (роутінг) для REST API та інших. Через це, коли користувач науково-технічної бібліотеки здійснює запит на необхідний роут (маршрут), спочатку відбувається перевірка доступів (авторизація) користувача в файлі *AuthController.js*. Якщо користувач не має доступу до системи (якщо немає

токену, термін дії токену закінчився, такого користувача не існує в системі, неправильний ключ-підпис токену, недійсний токен), на кінцевий термінал бібліотечних фондів користувачу надсилається відповідна помилка та пропонується здійснити вхід в систему повторно (*LoginController.js*). Коли перевірка відбулась успішно, то до об'єкту запиту (request) дописуються дані користувача (ID в базі даних, емейл, ім'я та ін.), і вже після цього продовжується робота користувача в системі та здійснюється перехід по потрібному маршруті. Таким чином, основною задачею маршрутизації (файл *routes.js*) є розподілення потоків запитів на відповідні контролери, а також попередня перевірка доступів і можливостей, тобто авторизація та автентифікація користувачів. Для цього використовується механізм *middleware* (проміжних) підфункцій, які у випадку позитивного результату авторизації модифікують тіло http-запиту після чого передають на відповідні контролери та функції системи для подальшої обробки.

У проектованій системі для різних потреб використовуються реляційна та нереляційна база даних. Для роботи з реляційною базою даних застосовано механізм об'єктно-реляційного відображення (ORM), що дозволило, не здійснюючи прямі SQL-запити, створювати процедурні методи з залученням об'єктно-орієнтованих підходів до роботи з таблицями, опрацьовуючи їх як об'єкти. Опісля механізмом ORM ці об'єкти та операції з ними транслюються в SQL-запит. На етапах опрацювання інформаційних запитів, де відсутня необхідність опису структур у табличній формі, вирішено застосовувати нереляційну MongoDB. Специфіка цієї СКБД дозволяє звертатися до елементів бази як до об'єктів в мові програмування (рис. 2), безпосередньо використовуючи методи розробника, що забезпечить швидкість обчислення і компактність програмного коду.

Підімкнення до баз даних MongoDB та Postgres забезпечують файли-конектори *db.mongo.js* та *db.postgre.js*, які зберігають містять реалізацію об'єкта драйвера, параметрами в яких виступають дані доступів до віддалених баз даних. Після чого здійснюється експорт реалізованого об'єкта: описане забезпечення принципу *Don't repeat yourself* (DRY) значно полегшує роботу з масивами даних, а також перемикання на інші джерела.

При проектуванні програмного забезпечення для роботи з сутностями баз даних використано об'єкти з описом реалізації таблиць у формі класу, де елементом опису виступає одиниця типу збереженого контенту, що забезпечується архітектурним шаблоном Model-View-Controller (MVC). У випадку з реляційними базами даних такими об'єктами виступають поля таблиць. Відтак, для відлагодження та фільтрування отриманих даних на стороні бекенду використовувано стандартизовані механізми валідації, що дозволяє перевіряти відповідність типу поля, а також відловлювати помилки та заборонені події до того, як відбудеться відправлення запиту з даними, що пришвидшує загальну роботу системи. Для полегшення реалізації зв'язних запитів (join) використовується механізм оголошення методів класу.

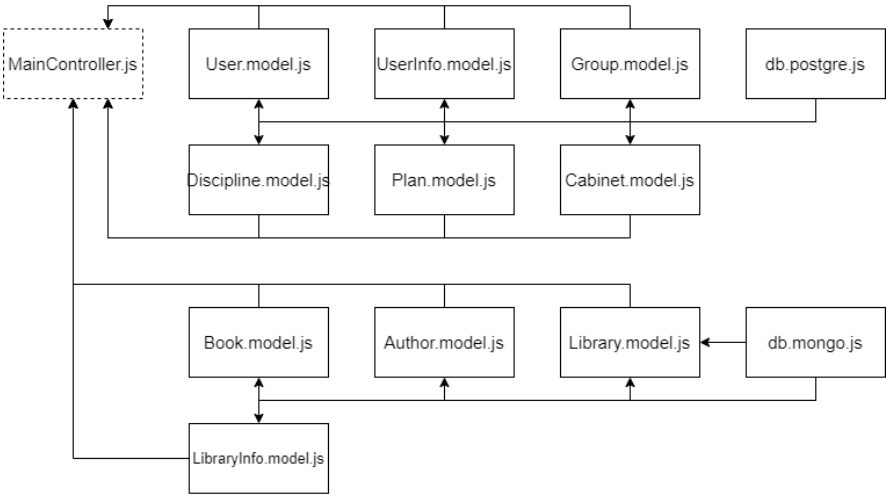

Рис. 2. Модельне представлення об'єктів збережуваних відомостей в базах даних

Основні операції з даними, підготовка даних на вивід, отримання даних з бази даних здійснюється контролерах (рис. 3) на фінальному етапі пакетування відомостей про автентифікацію користувача (рис. 1).

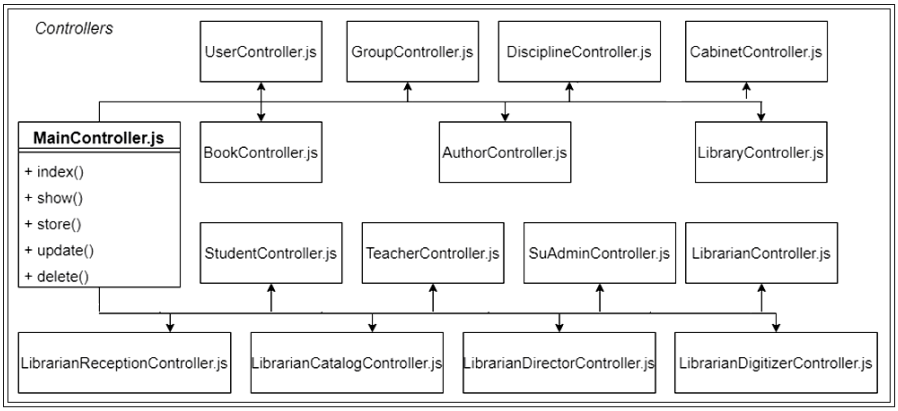

Рис. 3. Архітектура бізнес-логіки серверної частини

В більшості контролерів є потреба реалізацій задач *Create, Read, Update, Delete* (*CRUD*), тому для уникнення дублювання програмного коду (*DRY*), вирішено винести ці операції в окремий контролер (*MainController.js*) з відповідними методами *store*, *show*, *update*, *delete*.

При збереженні (store) необхідно вказати тіло об'єкту для збереження і засобами *ORM* надіслати ці дані запитом в базу даних. При читанні (*show*) та видаленні (*delete*) в загальному випадку достатньо вказати ідентифікатор

запису, після чого здійснюється відповідний запит в базу даних.

Для оновлення (*update*) даних необхідно надіслати запит, який міститиме і ідентифікатор запису, і значення об'єкту для оновлення. Також для того, щоб отримати всі записи моделі використовується метод *index*, який не потребує додаткових параметрів (в загальному випадку). Також в *MainController* здійснюється підключення моделей до контролера (рис. 2). При реалізації інших контролерів здійснюється наслідування *MainController* , після чого стає можливим використання описаних в ньому методів. Також в новому контролері є можливість за потреби перевизначити методи *MainController* .

Описаний підхід дозволяє полегшити можливість адаптування системи під різні платформи (рис. 4), зменшуючи дублювання коду при компілюванні клієнта для мобільних систем (Android, iOS), стаціонарних платформ (Windows, Linux, Mac), а також інших типів супровідних сервісів (зчитувач QR та штрих-коду, планшет Брайля та синтезатор мовлення тощо).

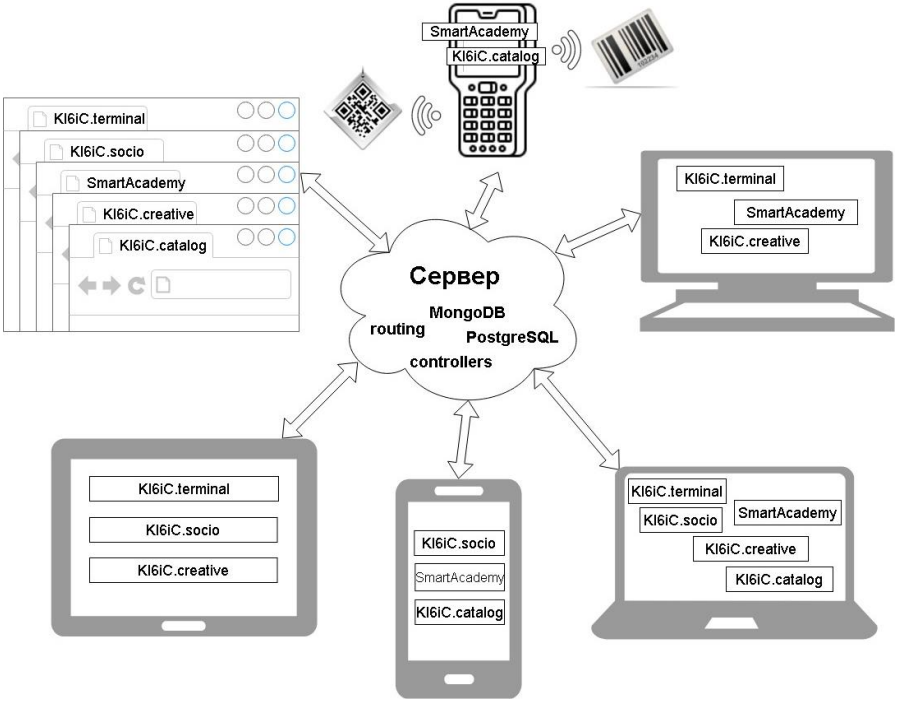

Рис. 4. Доставка цільового контенту на кінцевий термінал авторизованого користувача

Також впровадження обумовленої архітектури бізнес-логіки серверної частини та визначеної конфігурації оптимальних сценаріїв забезпечує доступ до ключових модулів комп'ютеризованої інформаційної бібліотечної системи

з окремих незалежних інтерфейсних терміналів кінцевих користувачів [9] із застосуванням різних типів обчислювальних пристроїв для переглядача оцифрованих зібрань науково-технічної бібліотеки, редактора фондів книгозбірні, медіа-платформи наукових досліджень, соціальної освітньої мережі та модуля моніторингу академічних досягнень [10].

Висновки

Наведене архітектурне рішення розгортання інтерактивного односторінкового застосунку та обґрунтована конфігурація сценаріїв реалізують адаптивну інфраструктуру для впровадження уніфікованої інформаційної платформи комп'ютеризованої системи розподіленого надання бібліотечних послуг для закладу вищої освіти. Побудовані конструктивні інтерфейси підімкнення баз даних і спроектована програмна частина сервера в подальшому забезпечать гнучке розширення кінцевого терміналу на довільні апаратні платформи користувачів з підтримкою типового мобільного чи десктопного системного забезпечення, здійснюючи горизонтальне розширення платформи з мінімалізованими затратами для приведення кодової бази до робочого стану.

1. *Бруй О.* Система стратегічного управління процесно-орієнтованою бібліотекою. Вісн. Книжк. палати. 2015. № 1 С. 14–17.
2. *Пилко И. С.* Информационные и библиотечные технологии. СПб.: Профессия, 2008. 342 с.
3. *Автономова Н.* Інформаційні продукти та послуги як результат виробничої діяльності бібліотек. Наукові праці НБУ. 2009. №. 25. С. 253–260.
4. *Ахметова Н. Р.* Інформаційний простір бібліотек ВНЗ. Сучасна бібліотека у науково-освітньому просторі. 2014. С. 131–139.
5. *Дригайло С. В.* Бібліотечно-інформаційні продукти і послуги для користувачів наукових бібліотек. Бібліотекознавство. Документознавство. Інформологія. 2010. № 4. С. 79–86.
6. *Колесникова Т.* Еволюція комунікаційних моделей діяльності бібліотек ВНЗ в умовах інформатизації. Бібліотечний вісник, 2013. № 2. С. 17–24.
7. *Іваськів Р. Р.* Обумовлення вимог до проектування автоматизованої бібліотечної інформаційної системи. Матеріали XXIII Міжнародної науково-практичної конференції з проблем ВПГ, 24 листопада 2016 р., м. Київ. С. 51-53.
8. *Іваськів Р. Р.* Визначення режимів автоматизованого інтерфейсу комп'ютеризованої інформаційної бібліотечної системи. Друкарство молоде. 2019. С. 37-39.
9. Рішення про реєстрацію договору № 42/1, який стосується права автора на твір №4027. Комп'ютерна програма «Web-термінал переглядача бібліотечних фондів» («KI6iC.terminal») / *Р. Іваськів, Т. Нерода*. Міністерство економічного розвитку і торгівлі України; дата реєстр. 27.09.2018.
10. *Ivaskiv R., Neroda T.* Modular technology of adaptation of training-methodological materials in scientific libraries of institutions of higher education. Videonauka: International Online Journal of Science. 2018. Vol. 2(10). – P. 27-33.

http://doi.org/10.5281/zenodo.3859683

Поступила 10.10.2019р.