МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки і підприємництва

Кафедра інформаційних технологій

Допущено до захисту
"_____" _____ 2025 р.
Зав. кафедри, к.т.н., доцент
_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА ДОДАТКУ ДЛЯ УПРАВЛІННЯ РОБОТОЮ БІБЛІОТЕКИ

Виконав – студент 2 курсу, групи 22 к-кн
освітньої програми "Комп’ютерні науки"
спеціальності 122 Комп’ютерні науки
Шкурко Максим Валерійович

Науковий керівник – к.е.н., доцент
Концеба Сергій Михайлович

Умань – 2025

ЗМІСТ

ВСТУП

РОЗДІЛ 1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1. Аналіз способів автоматизації управління бібліотекою

1.2. Порівняльний аналіз аналогів систем управління роботи бібліотек

1.3. Аналіз напрямків удосконалення програмного забезпечення для автоматизації управління бібліотеками

1.4. Постановка задач для розробки автоматизованої системи

РОЗДІЛ 2 ПРОЕКТУВАННЯ МОДЕЛІ, МЕТОДІВ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1. Аналіз вхідних та вихідних даних автоматизованої системи бібліотеки

2.2. Проектування архітектури та моделі автоматизованої системи

2.3. Проектування загального алгоритму роботи системи

2.4. Проєктування макетів графічного інтерфейсу користувача

2.5. Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Розробка інтерфейсу користувача

3.2. Розробка сервісу обробки даних та сервісу сповіщень

3.3. Розробка сервісу управління книгами

3.4. Тестування основного функціоналу

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

ВСТУП

Сучасне інформаційне суспільство характеризується стрімким розвитком цифрових технологій, які дедалі глибше проникають в усі сфери життя, зокрема — в освіту, культуру та інформаційне обслуговування. Бібліотеки, які тривалий час були осередками накопичення та поширення знань, нині стикаються з необхідністю трансформації своєї діяльності у відповідь на нові вимоги часу. Користувачі очікують швидкого доступу до інформації, можливості онлайн-взаємодії з бібліотекою, дистанційного бронювання, перегляду електронного каталогу, продовження термінів користування літературою, отримання повідомлень про нові надходження та інше. Усе це можливо реалізувати лише за умови впровадження сучасних автоматизованих систем управління.

Багато бібліотек в Україні та світі ще користуються застарілими обліковими формами, паперовими каталогами або фрагментарними електронними системами, які не забезпечують інтеграції всіх процесів у єдиний інформаційний простір. Це призводить до затримок у обслуговуванні, помилок при видачі або поверненні літератури, втрати актуальної інформації про фонди тощо. Наявність ефективного додатку для управління роботою бібліотеки дозволяє централізовано керувати всіма ключовими процесами — від формування бази книг і реєстрації читачів до аналітики користування та генерування статистичних звітів.

Особливо актуальним є питання створення гнучких та масштабованих рішень, які можна адаптувати до бібліотек різного типу — від шкільних і студентських до публічних чи спеціалізованих установ. В умовах обмеженого фінансування вітчизняних бібліотек важливою є також можливість створення доступного та відкритого програмного забезпечення з використанням безкоштовних інструментів та технологій з відкритим кодом.

Крім того, розвиток мобільних платформ і месенджерів відкриває нові перспективи для інтерактивної взаємодії користувачів з бібліотекою. Інтеграція додатку з Telegram-ботами, електронною поштою або SMS-сповіщеннями дозволяє суттєво покращити якість комунікації з читачами та зробити бібліотечне обслуговування більш зручним і сучасним.

Таким чином, розробка додатку для управління роботою бібліотеки є не лише актуальним, але й соціально значущим напрямом у сфері автоматизації інформаційних процесів. Це рішення сприятиме підвищенню ефективності роботи бібліотек, зручності для користувачів та наближенню бібліотечної інфраструктури до світових цифрових стандартів.

Метою дослідження є розробка програмного додатку, який забезпечує автоматизоване управління основними процесами роботи бібліотеки, включаючи облік, пошук, замовлення та бронювання літератури, а також формування повідомлень та рекомендацій для користувачів.

Об'єктом дослідження виступає інформаційна система бібліотеки як комплекс технічних та програмних засобів для організації та підтримки її діяльності.

Предметом дослідження є методи та програмні засоби автоматизації процесів керування бібліотечними ресурсами, а також їхня реалізація у вигляді прикладного програмного додатку.

Завдання дослідження включають:

- аналіз існуючих систем автоматизації бібліотек;
- постановка задачі розробки додатку;
- побудова архітектури системи та моделювання її компонентів;
- розробка серверної та клієнтської частин додатку;
- реалізація механізмів авторизації, обліку літератури, бронювання, сповіщень;
- тестування та аналіз ефективності роботи створеної системи.

Методи дослідження, що використовувалися у роботі, включають методи системного аналізу, об’єктно-орієнтованого проєктування, моделювання бізнес-процесів, а також програмну реалізацію із застосуванням сучасних мов програмування, інструментів інтеграції API та бібліотек для реалізації мікросервісної архітектури.

Таким чином, дана кваліфікаційна робота спрямована на розробку прикладного рішення, яке відповідає сучасним вимогам до автоматизованих систем управління бібліотеками та забезпечує розширену функціональність для читачів і адміністраторів.

РОЗДІЛ 1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз способів автоматизації управління бібліотекою

У сучасних умовах цифрової трансформації ключовим завданням бібліотек є не лише збереження та надання доступу до інформаційних ресурсів, а й забезпечення високого рівня обслуговування користувачів за допомогою інноваційних технологій. Автоматизація управління бібліотекою дозволяє досягти ефективного функціонування бібліотечних процесів, скоротити витрати часу на рутинні операції та розширити інформаційні послуги для користувачів.

Одним із найбільш поширених способів автоматизації управління бібліотекою є впровадження автоматизованих бібліотечно-інформаційних систем. АБІС забезпечують інтеграцію основних бібліотечних процесів: комплектування фонду, каталогізацію, обслуговування користувачів, ведення електронного каталогу, видачу та повернення документів, ведення звітності.

Серед найпоширеніших систем можна виділити:

- ІРБІС — вітчизняна система, що підтримує MARC-формат, адаптована до українських бібліотек.
- Koha — міжнародна відкрита система з веб-інтерфейсом, яка активно впроваджується в академічних і публічних бібліотеках.
- Aleph — комерційна система від компанії Ex Libris, використовується у великих наукових бібліотеках.
- Libra, АБУ — локальні рішення для шкільних і муніципальних бібліотек України.

Функціональність таких систем передбачає:

- Створення та ведення електронного каталогу;
- Пошук літератури за різними критеріями;
- Ведення бази даних читачів;
- Реєстрацію видачі та повернення документів;
- Формування статистичних та управлінських звітів.

RFID-технології (радіочастотна ідентифікація) значно спрощують процеси обліку фонду та обслуговування користувачів. Встановлення RFID-міток на бібліотечні ресурси дозволяє автоматизувати:

- Швидку видачу/повернення книг без участі бібліотекаря;
- Проведення інвентаризації фонду;
- Систему проти крадіжок;
- Пункти самообслуговування користувачів.

Застосування RFID-технологій підвищує точність і швидкість операцій, зменшує навантаження на персонал, а також забезпечує додаткову безпеку фонду.

Однією з ключових складових автоматизації є створення електронного каталогу, що забезпечує зручний пошук і доступ до бібліотечних ресурсів. Більшість сучасних АБІС дозволяють інтеграцію з веб-сайтами бібліотек, забезпечуючи дистанційний доступ до каталогу, можливість попереднього замовлення літератури, продовження терміну користування та перегляд інформації про стан читацького формуляра.

Окремі бібліотеки впроваджують мобільні додатки або чат-боти для спрощення комунікації з користувачами. Це значно розширює аудиторію та підвищує якість обслуговування.

Оцифрування бібліотечних документів є важливим напрямом автоматизації, який забезпечує:

- Збереження рідкісних та цінних документів;
- Надання віддаленого доступу до електронних копій;
- Використання ресурсів у навчальному та науковому процесі без фізичного зносу оригіналів.

Цифрові колекції формуються в окремих репозитаріях або інтегруються в електронні бібліотеки загальнодоступного користування. Така діяльність відповідає принципам відкритої науки (Open Science) та відкритого доступу (Open Access).

Сучасні бібліотеки дедалі частіше інтегруються з:

- Електронними системами навчання (Moodle, Google Classroom);
- Репозитаріями наукових публікацій;
- Електронними базами даних (Scopus, Web of Science, eLIBRARY);
- Каталогами інших бібліотек (наприклад, через протокол Z39.50).

Інтеграція дозволяє студентам, науковцям і дослідникам отримувати доступ до розширених інформаційних ресурсів без потреби фізично відвідувати бібліотеку.

Найновіші тренди автоматизації бібліотек пов’язані із застосуванням технологій штучного інтелекту (ШІ). До можливих застосувань належать:

- Рекомендаційні системи на основі аналізу вподобань користувачів;
- Інтелектуальні пошукові системи;
- Автоматична класифікація та аналіз змісту документів;
- Чат-боти для консультування користувачів.

ШІ дозволяє зробити бібліотеку персоналізованою та максимально адаптованою до потреб кожного користувача.

Автоматизація управління бібліотекою є багаторівневим процесом, що охоплює технічні, організаційні та інформаційні аспекти. Систематичне впровадження сучасних технологій сприяє підвищенню ефективності бібліотечної діяльності, покращенню якості обслуговування користувачів та збереженню культурної спадщини. Кожна бібліотека обирає свій шлях автоматизації, залежно від власних ресурсів, потреб користувачів та стратегічних цілей.

1.2 Порівняльний аналіз аналогів систем управління роботи бібліотек

На сьогоднішній день існує ряд автоматизованих систем управління бібліотекою, які пропонують різноманітні можливості для оптимізації роботи бібліотек та покращення обслуговування користувачів. Проведення порівняльного аналізу цих систем допоможе визначити їх переваги та недоліки,

а також визначити потенційні напрямки для вдосконалення власної автоматизованої системи управління бібліотекою.

Koha (рис. 1.1) є однією з найпоширеніших у світі відкритих автоматизованих бібліотечно-інформаційних систем (АБІС), яка активно впроваджується у публічних, академічних та шкільних бібліотеках. Система розроблена у Новій Зеландії в 1999 році і з того часу набула глобального поширення завдяки відкритому програмному коду, активній спільноті розробників і широким функціональним можливостям.



Рис. 1.1. Зображення програмного продукту «Koha»

Koha базується на веб-технологіях і підтримує роботу з MARC21, Z39.50, RFID, що забезпечує її сумісність з міжнародними бібліотечними стандартами. Система розповсюджується на умовах ліцензії GNU General Public License (GPL), що дозволяє вільно використовувати, модифікувати та поширювати її програмний код.

Основні функціональні можливості Koha:

- Комплектування бібліотечного фонду з веденням обліку замовлень, постачальників і надходжень;
- Каталогізація документів за MARC-форматом;

- Облік читачів з веденням електронних формулярів;
- Видача/повернення літератури із реєстрацією порушень термінів;
- Пошук у каталозі за різними параметрами (ключове слово, автор, тема, індекс тощо);
- Створення онлайн-каталогу (OPAC) з вільним доступом для користувачів;
- Формування статистичних звітів та аналітичних даних для управління.

Koha має два інтерфейси:

- Адміністративний інтерфейс для бібліотекарів, де реалізовано всі функції керування фондом і обслуговуванням;
- OPAC (Online Public Access Catalog) — веб-інтерфейс для користувачів, який дозволяє здійснювати пошук документів, продовжувати терміни користування, замовляти книги та залишати відгуки.

Переваги Koha:

- Відкритий код — можливість адаптації системи до локальних умов та інтеграції з іншими сервісами;
- Веб-архітектура — не потребує складного встановлення на клієнтських комп’ютерах, доступ через браузер;
- Міжнародна підтримка — регулярні оновлення, документація та активна спільнота користувачів;
- Підтримка мультимовності, включно з українською мовою;
- Гнучке адміністрування прав доступу до модулів та операцій.

Koha використовується в ряді українських бібліотек, зокрема в університетських бібліотеках (наприклад, бібліотека Українського католицького університету) та в окремих публічних бібліотеках, які прагнуть перейти на сучасні технології без додаткових витрат на ліцензії.

Завдяки своїй відкритості та масштабованості, Koha є перспективним рішенням для бібліотек, що прагнуть впровадити автоматизацію з мінімальними

витратами. Вона дозволяє створити повнофункціональну інформаційну екосистему, доступну як бібліотекарям, так і читачам у зручному онлайн-режимі.

FOLIO (рис. 1.2) (від англ. The Future Of Libraries Is Open) — це інноваційна модульна платформа з відкритим кодом, призначена для повної автоматизації бібліотечних процесів. Вона поєднує в собі функціональність класичної автоматизованої бібліотечно-інформаційної системи (АБІС) із можливостями сучасної екосистеми додатків, побудованої на мікросервісній архітектурі.

FOLIO є результатом міжнародної співпраці бібліотекарів, розробників та організацій, таких як EBSCO, Open Library Environment (OLE) та Index Data. Система активно розвивається з 2016 року й уже запроваджена в провідних університетських бібліотеках світу [4].

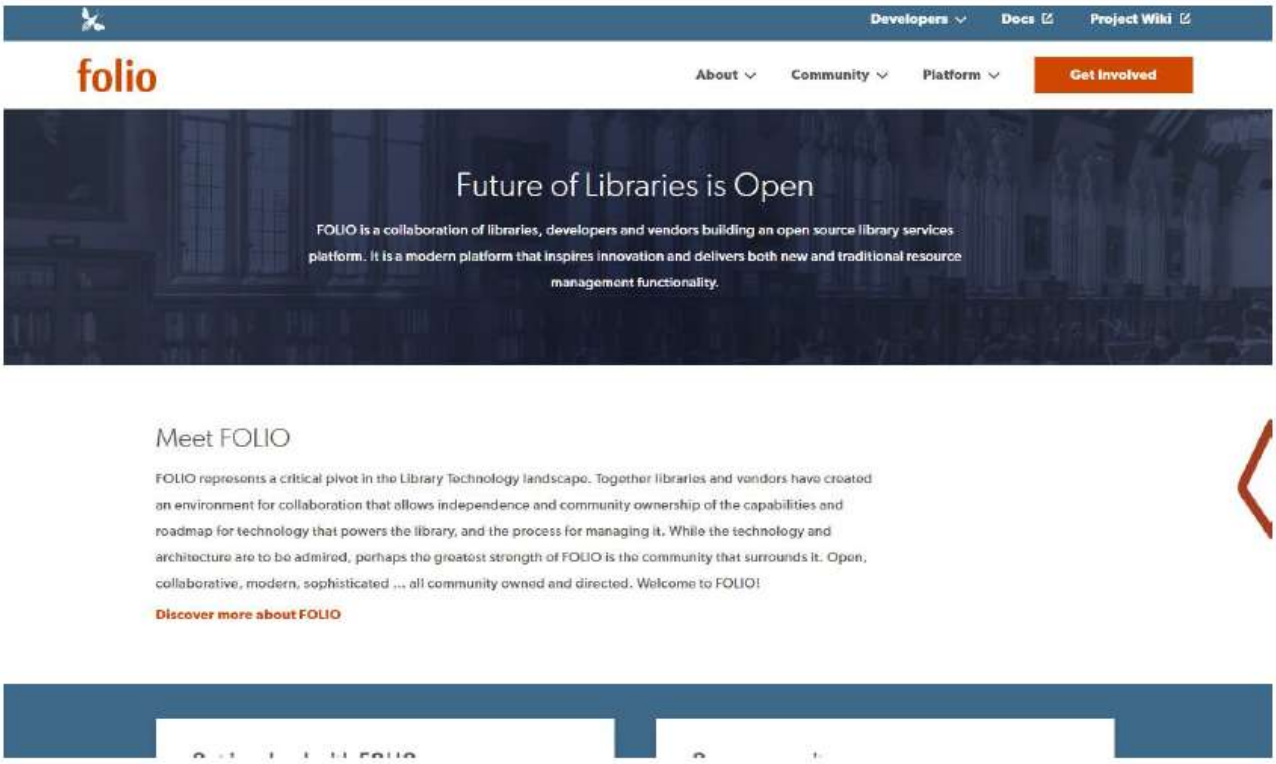


Рис. 1.2. Зображення програмного продукту «FOLIO»

Основні характеристики FOLIO:

- Модульна архітектура — кожен компонент системи (наприклад, каталогізація, облік користувачів, аналітика) реалізований у вигляді окремого модуля, який можна вмикати, вимикати чи замінювати;

- Інтерфейс як для бібліотекарів, так і для розробників, з можливістю гнучкого налаштування;
- Підтримка сучасних технологій: REST API, JSON, Kubernetes, Docker;
- Веб-орієнтована: вся робота з платформою здійснюється через браузер.

Основні функції FOLIO:

- Керування фондами (acquisitions, inventory);
- Каталогізація з підтримкою MARC21 та перспективою переходу на BIBFRAME;
- Облік користувачів та історії їх активності;
- Видача та повернення документів, управління штрафами та продовженнями;
- Статистика та звітність, інтеграція з аналітичними інструментами;
- Пошук та доступ до документів через інтегрований або зовнішній OPAC (наприклад, VuFind або EBSCO Discovery Service);
- Інтеграція з іншими системами ВНЗ (реєстрами студентів, електронними курсами, науковими репозитаріями).

Переваги FOLIO:

- Гнучкість і масштабованість: підходить для великих академічних бібліотек із складною структурою;
- Повністю відкритий код (Open Source) — можливість доопрацювання та адаптації під потреби конкретної бібліотеки;
- Інтеграція з міжнародними ресурсами (EBSCO, OCLC, ORCID тощо);
- Сучасний інтерфейс користувача та підтримка мультимовності;
- Можливість хостингу в хмарі або локального розгортання.

Платформа FOLIO уже впроваджена або тестується в університетських бібліотеках США, Німеччини, Фінляндії, Данії, а також окремих пілотних проектах у Східній Європі. Наприклад, Колумбійський університет (США) перейшов на FOLIO для підтримки багатомільйонного фонду та інтеграції з дослідницькою інфраструктурою. Попри складність у встановленні та потребу в технічній підтримці, FOLIO розглядається як перспективна платформа для

українських наукових бібліотек, які прагнуть вийти на міжнародний рівень обслуговування. Його відкритість, масштабованість і активна міжнародна спільнота роблять цю систему привабливою для університетів, що мають IT-підтримку та стратегічну мету цифрової трансформації.

Alma (рис. 1.3) — це сучасна, інтегрована хмарна платформа для автоматизації бібліотечної діяльності, розроблена компанією Ex Libris, що є частиною корпорації Clarivate. Alma орієнтована насамперед на великі академічні, дослідницькі та національні бібліотеки, які потребують високого рівня інтеграції, автоматизації та масштабованості.

Платформа забезпечує повний цикл бібліотечного управління, включаючи придбання, каталогізацію, обробку електронних і друкованих ресурсів, облік користувачів, аналітику та звітність. Alma функціонує як Software-as-a-Service (SaaS) — усі компоненти системи доступні через хмару, що дозволяє мінімізувати витрати на інфраструктуру та технічну підтримку з боку бібліотеки [5].



Рис. 1.3. Зображення програмного продукту «Alma»

Основні характеристики Alma:

- Єдина централізована система для управління всіма типами ресурсів: друкованими, електронними, цифровими;
- Повна інтеграція з іншими продуктами Ex Libris (наприклад, Primo для пошуку, Leganto для навчальних курсів, Esploro для наукових даних);
- Автоматизація рутинних процесів — облік надходжень, метаданих, обробка запитів користувачів тощо;
- Підтримка відкритих стандартів: MARC21, Dublin Core, BIBFRAME, Z39.50, OpenURL, OAI-PMH;
- Гнучка система звітності з візуалізацією даних через аналітичний інструмент Alma Analytics (на базі Oracle BI).

Функціональні можливості Alma:

- Придбання ресурсів (acquisitions): керування бюджетами, постачальниками, замовленнями;
- Каталогізація та керування метаданими;
- Керування електронними ресурсами (ERM): інтеграція з електронними журналами, базами даних, доступом за ліцензіями;
- Облік користувачів і взаємодія з ними через інтегрований CRM;
- Взаємодія з відкритими репозитаріями та підтримка відкритої науки;
- Міжбібліотечний абонемент (ILL) та кооперація в рамках бібліотечних консорціумів.

Переваги Alma:

- Хмарна модель з регулярними оновленнями та високою надійністю;
- Глобальне охоплення — понад 2000 бібліотек у світі використовують Alma;
- Інтеграція з навчальними та науковими платформами університетів (LMS, системи управління наукою, репозитарії);
- Підтримка багатьох мов, включаючи українську;
- Інтуїтивний інтерфейс, орієнтований на покращення ефективності роботи персоналу.

Обмеження:

- Alma є комерційним продуктом, що передбачає щорічну ліцензійну оплату, яка може бути значною для невеликих бібліотек;
- Платформа працює лише в хмарному середовищі, отже повністю залежить від наявності інтернет-доступу;
- Обмежена можливість модифікації функціоналу — налаштування здійснюються в межах дозволених параметрів системи.

Alma активно використовується в провідних університетах та бібліотечних консорціумах, зокрема в Гарвардському університеті (США), Університеті Оксфорда (Велика Британія), Національній бібліотеці Ізраїлю, Консорціумах бібліотек Німеччини, Канади, Австралії та інших країн. Alma поки що не має широкого розповсюдження в Україні через вартість ліцензії та вимоги до IT-інфраструктури. Проте її застосування можливе в рамках великих академічних або національних проєктів, особливо якщо передбачається інтеграція з іншими продуктами Ex Libris, міжнародними наукометричними системами або платформами відкритого доступу.

Порівняльний аналіз особливостей розглянутих аналогів із власною розробкою подано в таблиці 1.1.

1.1. Порівняльний аналіз програмних продуктів аналогів

Характеристика	Koha	FOLIO	Alma
Тип платформи	Відкрите ПЗ (Open Source)	Відкрите ПЗ (Open Source)	Комерційне рішення (Proprietary SaaS)
Модель розгортання	Локально або хмарно	Локально або хмарно	Тільки хмарна (SaaS)
Модульність	Стандартна модульна архітектура	Висока, мікросервіси	Висока, але закрита
Мова інтерфейсу	Багатомовна (є українська)	Багатомовна (є українська)	Багатомовна (є українська)
Підтримка MARC/BIBFRAME	MARC21	MARC21, планується підтримка BIBFRAME	MARC21, BIBFRAME
Каталог користувачів (OPAC)	Вбудований OPAC	Можна інтегрувати VuFind, EDS тощо	Працює з Primo

Характеристика	Koha	FOLIO	Alma
Підтримка електронних ресурсів	Обмежена, через плагіни	Повноцінна підтримка	Потужна інтеграція з е-ресурсами
Аналітика і звітність	Базові звіти, SQL-запити	Інтегровані аналітичні модулі	Потужна BI-аналітика (Alma Analytics)
Інтеграція з іншими системами	API, Z39.50, SIP2, LDAP	REST API, підтримка багатьох стандартів	API, інтеграція з продуктами Ex Libris
Спільнота користувачів	Активна, глобальна	Зростаюча, академічна	Професійна підтримка від Ex Libris
Вартість	Безкоштовне ПЗ, є витрати на підтримку	Безкоштовне ПЗ, але потрібні ресурси	Платна підписка, висока вартість
Підходить для	Малих і середніх бібліотек	Середніх і великих академічних бібліотек	Великих академічних та національних бібліотек
Можливість кастомізації	Висока, через зміну коду	Висока, модульна архітектура	Обмежена, залежить від Ex Libris
Вимоги до IT-персоналу	Низькі або середні	Високі	Мінімальні (все керується через Ex Libris)

Порівняння трьох сучасних бібліотечних систем — Koha, FOLIO та Alma — свідчить про суттєві відмінності у підходах до автоматизації бібліотечних процесів, рівні технічної складності, гнучкості, вартості та цільової аудиторії.

- Koha є оптимальним вибором для невеликих і середніх бібліотек, зокрема публічних та освітніх, які потребують базової автоматизації з мінімальними витратами. Її відкритий код і активна спільнота дозволяють адаптувати систему до локальних потреб, однак для повноцінної підтримки електронних ресурсів можуть знадобитися додаткові модулі або доопрацювання.
- FOLIO позиціонується як перспективна система для академічних бібліотек із потужним IT-супроводом. Її модульна мікросервісна архітектура забезпечує високу гнучкість і масштабованість, але водночас вимагає значних ресурсів на встановлення, налаштування та технічне обслуговування. FOLIO є прикладом платформи, яка поєднує бібліотечну функціональність з принципами сучасного програмного середовища.

- Alma — це повнофункціональна комерційна платформа, що ідеально підходить для великих академічних, наукових або національних бібліотек, які готові інвестувати в якісний сервіс та стабільну хмарну інфраструктуру. Високий рівень автоматизації, інтеграції та підтримки робить Alma привабливою, проте її висока вартість і закритість коду обмежують гнучкість для індивідуального налаштування.

Таким чином, вибір системи автоматизації повинен базуватися на стратегічних потребах конкретної бібліотеки, її розмірах, бюджетних можливостях, IT-потенціалі та вимогах до обслуговування користувачів. Універсального рішення не існує, але кожна з розглянутих систем має свої сильні сторони, що дозволяє знайти оптимальний варіант у межах конкретного проєкту або установи.

1.3 Аналіз напрямків удосконалення програмного забезпечення для автоматизації управління бібліотеками

У контексті цифрової трансформації та активного впровадження інформаційно-комунікаційних технологій у різні сфери діяльності, автоматизація бібліотек набуває нового стратегічного значення. Програмне забезпечення, яке використовується для управління бібліотечними процесами, потребує постійного оновлення та адаптації до сучасних умов, потреб користувачів і розвитку інформаційних технологій.

Аналіз тенденцій розвитку бібліотечних інформаційних систем дозволяє виокремити низку ключових напрямів удосконалення таких систем:

1. Інтеграція з електронними та цифровими ресурсами

Сучасне бібліотечне ПЗ повинно підтримувати повноцінну роботу з електронними документами, цифровими колекціями, базами даних, репозитаріями, електронними журналами. Зокрема, бібліотеки мають забезпечувати доступ до відкритих наукових ресурсів (Open Access), керування електронними передплатами (ERM) та повноцінну роботу з системами ідентифікації (DOI, ORCID). Університетська бібліотека Кембриджу впровадила

FOLIO для підтримки керування електронними ресурсами, забезпечуючи користувачам єдиний доступ до друкованих і цифрових фондів.

2. Хмарні технології та SaaS-моделі

Перехід до хмарної архітектури дозволяє забезпечити постійний доступ до бібліотечних сервісів, зменшити витрати на технічне обслуговування, пришвидшити оновлення ПЗ і підвищити безпеку. Системи на основі моделі Software-as-a-Service (SaaS) стали стандартом для великих академічних бібліотек. Система Alma від Ex Libris активно використовується у бібліотеках понад 40 країн, зокрема у Гарварді, Сорбонні та Національній бібліотеці Ізраїлю.

3. Аналітика та інструменти бізнес-аналізу

Бібліотеки дедалі частіше використовують аналітичні дані для управлінських рішень. Це включає звітність про використання ресурсів, поведінку користувачів, ефективність фондів тощо. Сучасне ПЗ повинно містити гнучкі механізми збору, обробки та візуалізації даних. В Alma реалізовано модуль Alma Analytics на основі Oracle BI, який дозволяє створювати адаптовані звіти для адміністраторів.

4. Мобільність та багатоплатформеність

Значна частина користувачів звертається до бібліотечних ресурсів з мобільних пристроїв, тому інтерфейси систем повинні бути адаптивними або мати окремі мобільні додатки. Деякі інсталяції Koha використовують інтеграцію з мобільними додатками на базі Progressive Web Apps (PWA) для доступу до електронного каталогу.

5. Персоналізація обслуговування

Сучасні користувачі очікують персоналізованих сервісів: рекомендації за інтересами, індивідуальні сповіщення, віртуальні книжкові полиці тощо. Платформа Primo VE, яка працює в парі з Alma, забезпечує персоналізований інтерфейс користувача з рекомендаційною системою.

6. Штучний інтелект та машинне навчання

Хоча ці технології ще на етапі впровадження, деякі бібліотеки вже застосовують AI для автоматичної класифікації та тегування ресурсів, побудови

чат-ботів для користувачів, прогнозування попиту на книги. Національна бібліотека Фінляндії використовує AI для автоматичного створення описів стародруків у цифровому архіві.

7. Відкриті стандарти та інтероперабельність

Для підвищення ефективності обміну даними з іншими системами (електронними журналами, репозитаріями, бібліотечними об'єднаннями) ПЗ повинно підтримувати міжнародні стандарти: MARC21, BIBFRAME, Dublin Core, Z39.50, OAI-PMH, OpenURL тощо. FOLIO з самого початку розробляється як модульна відкрита платформа з API, яка підтримує інтеграцію з будь-якими зовнішніми сервісами.

8. Захист даних та інформаційна безпека

Системи мають відповідати міжнародним стандартам безпеки даних, включаючи шифрування, автентифікацію, контроль доступу, аудит змін, відповідність до GDPR. Alma сертифікована за ISO/IEC 27001 та дотримується вимог GDPR.

9. Автоматизація внутрішніх бібліотечних процесів

Окрім користувацьких сервісів, важливо автоматизувати такі процеси, як: облік надходжень, інвентаризація, списання, облік роботи персоналу, бюджетування та документообіг. Koha надає можливість створення кастомних звітів SQL, що дозволяє ефективно вести внутрішню аналітику.

10. Модульність і кастомізація Системи мають бути побудовані на принципах відкритої архітектури, що дозволяє додавати або прибирати функціональні модулі відповідно до потреб установи. FOLIO підтримує мікросервісну архітектуру, що дозволяє організаціям використовувати лише ті модулі, які їм необхідні (циркуляція, каталогізація, ERM тощо).

Таким чином, розвиток програмного забезпечення для автоматизації бібліотек спрямований на підвищення гнучкості, функціональності, безпеки та зручності у користуванні. Світові практики демонструють, що успішне впровадження таких систем залежить від поєднання інноваційних технологій і глибокого розуміння потреб як бібліотечних працівників, так і кінцевих

користувачів. Для українських бібліотек особливо актуальними є рішення з відкритим кодом (наприклад, Koha чи FOLIO), які дозволяють розвивати інформаційні системи з урахуванням національного контексту та обмеженого фінансування.

1.4. Постановка задач для розробки автоматизованої системи

У сучасних умовах цифровізації інформаційного середовища зростає потреба у створенні ефективного інструменту для автоматизації бібліотечної діяльності. Особливо це актуально для невеликих навчальних, муніципальних або спеціалізованих бібліотек, де необхідне просте у впровадженні, доступне за ресурсами та гнучке рішення для організації внутрішніх процесів.

Створити програмний додаток, який забезпечить автоматизацію основних бібліотечних процесів: облік фонду, реєстрацію користувачів, видачу та повернення книг, формування звітів та статистики, а також зручний доступ до каталогу документів.

Основні функції, які має реалізовувати додаток:

1. Облік бібліотечного фонду
   - Додавання нових видань (запис бібліографічної інформації, інвентаризаційний номер, кількість примірників).
   - Редагування та вилучення записів.
   - Пошук і фільтрація за автором, назвою, роком видання, тематикою тощо.
2. Реєстрація користувачів
   - Створення облікового запису читача.
   - Зберігання персональних даних, історії відвідувань і користування фондами.
   - Ведення читацьких формулярів.
3. Облік видачі та повернення документів
   - Фіксація дати видачі та терміну повернення.

- Формування нагадувань або повідомлень про прострочення.
- Автоматичне оновлення статусу книги.

4. Формування звітів та статистики
   - Звіти про рух фонду (надходження, вибуття).
   - Статистика активності користувачів.
   - Найпопулярніші видання за період.
5. Пошукова система в каталозі для користувачів
   - Доступ до електронного каталогу.
   - Можливість переглядати наявність примірників.
   - Функція попереднього бронювання книги.
6. Адміністративна панель
   - Керування довідниками (жанри, видавництва, місця зберігання).
   - Користувацькі права доступу (читач, бібліотекар, адміністратор).
   - Резервне копіювання та відновлення даних.

Технічні вимоги до реалізації:

- Платформа: Веб-додаток або мультиплатформовий десктоп-додаток (за потребою мобільна адаптація).
- База даних: SQLite / MySQL / PostgreSQL (залежно від масштабу).
- Мова програмування: Python (Django, Flask), PHP (Laravel), JavaScript (Node.js + React/Vue), або інша сучасна технологія.
- Інтерфейс: інтуїтивно зрозумілий, адаптивний, з підтримкою української мови.
- Безпека: Аутентифікація, контроль доступу, захист персональних даних.
- Можливість масштабування у майбутньому (додавання нових модулів, інтеграція з іншими сервісами).

Обмеження та припущення:

- Додаток призначений для використання у бібліотеці з обмеженим штатом та середнім обсягом фонду (до 10 000 одиниць зберігання).
- Система не передбачає інтеграцію з міжнародними бібліотечними об’єднаннями або системами RFID на першому етапі.

- У розробці враховується можливість подальшого розширення функціоналу.

Очікувані результати:

- Зручна система управління бібліотекою, яка значно спрощує роботу бібліотекаря.
- Прозора структура обліку і контролю за фондом.
- Підвищення якості обслуговування користувачів бібліотеки.

РОЗДІЛ 2 ПРОЕКТУВАННЯ МОДЕЛІ, МЕТОДІВ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз вхідних та вихідних даних автоматизованої системи бібліотеки

Ефективне функціонування автоматизованої бібліотечної системи неможливе без чіткого розуміння структури та особливостей інформаційних потоків. Усі дані, що обробляються в системі, поділяються на вхідні, внутрішні (проміжні) та вихідні. Кожен з цих типів відіграє важливу роль у забезпеченні повного циклу роботи бібліотеки — від реєстрації фонду до формування статистичних звітів.

Вхідними даними є всі відомості, які надходять до системи від користувачів або персоналу бібліотеки. Основу вхідної інформації становлять бібліографічні відомості про видання: назва книги, автор, рік видання, видавництво, шифр УДК, кількість примірників тощо. Ці дані вводяться або вручну бібліотекарем, або імпортуються з готових каталогів чи зовнішніх джерел.

Крім цього, система приймає персональні дані користувачів, які реєструються для користування бібліотечними ресурсами. До таких відомостей належать ім’я, прізвище, контактна інформація, дата народження, навчальний заклад або установа тощо.

Також до вхідних даних належать записи про видачу книг користувачам, дати повернення, інформація про продовження терміну користування виданням або про втрату книги. Всі ці дії фіксуються в системі як події, що впливають на поточний статус бібліотечного фонду.

Окремим типом вхідних даних є адміністративна інформація, що використовується для налаштування системи: це дані про жанри, типи видань, місця зберігання, а також параметри доступу, права користувачів тощо.

У процесі роботи система генерує низку проміжних даних, які не вводяться безпосередньо користувачами, але створюються автоматично на основі взаємодії між сутностями. Сюди належать, наприклад, історія використання книг певним

читачем, поточний статус кожного примірника (вільний, виданий, втрачений), аналітичні зведення по фонду тощо.

Також система зберігає службові дані: лог-файли дій користувачів, записи про оновлення даних, помилки та сповіщення. Ці дані важливі для технічного обслуговування системи, її безпеки та відновлення у разі збоїв.

Вихідна інформація — це результати обробки, які система надає користувачам, бібліотекарям або адміністрації. Найбільш поширеним типом вихідних даних є результати пошуку в електронному каталозі. Користувач має змогу переглядати назви книг, доступність, короткий опис, місце зберігання, а також бачити, чи можна зарезервувати книгу.

Для бібліотекаря система генерує статистику видач, звіти про запаси фонду, кількість активних користувачів, найпопулярніші книги тощо. Також передбачено формування звітів для адміністрації, наприклад: щомісячна або квартальна звітність про рух бібліотечного фонду, динаміку відвідувань, виявлення книг, які тривалий час не повернуті.

Система також забезпечує вихідні повідомлення — наприклад, сповіщення користувачів про наближення терміну повернення книги або про прострочення, які можуть надсилатися електронною поштою або відображатися в особистому кабінеті.

Всі дані в системі повинні бути уніфіковані, повні, актуальні та несуперечливі. Необхідно забезпечити єдиний формат введення дати, правильне оформлення імен та назв, уникнення дублювання записів, особливо при масовому імпорті даних. Для забезпечення конфіденційності інформації слід передбачити розмежування прав доступу до даних відповідно до ролі користувача.

Таким чином, ефективна обробка вхідних і вихідних даних є запорукою стабільної та зручної роботи автоматизованої бібліотечної системи. Ретельне проектування структури даних дозволяє уникнути помилок, забезпечити цілісність інформації та підвищити рівень обслуговування користувачів. Усі етапи взаємодії з даними — від введення до аналізу — повинні бути чітко

описані, автоматизовані та оптимізовані відповідно до потреб сучасної бібліотеки.

2.2 Проектування архітектури та моделі автоматизованої системи

Для візуального представлення архітектури системи та зв'язків її компонентів (сервісів) було створено структурну діаграму, що зображена на рисунку 2.1.

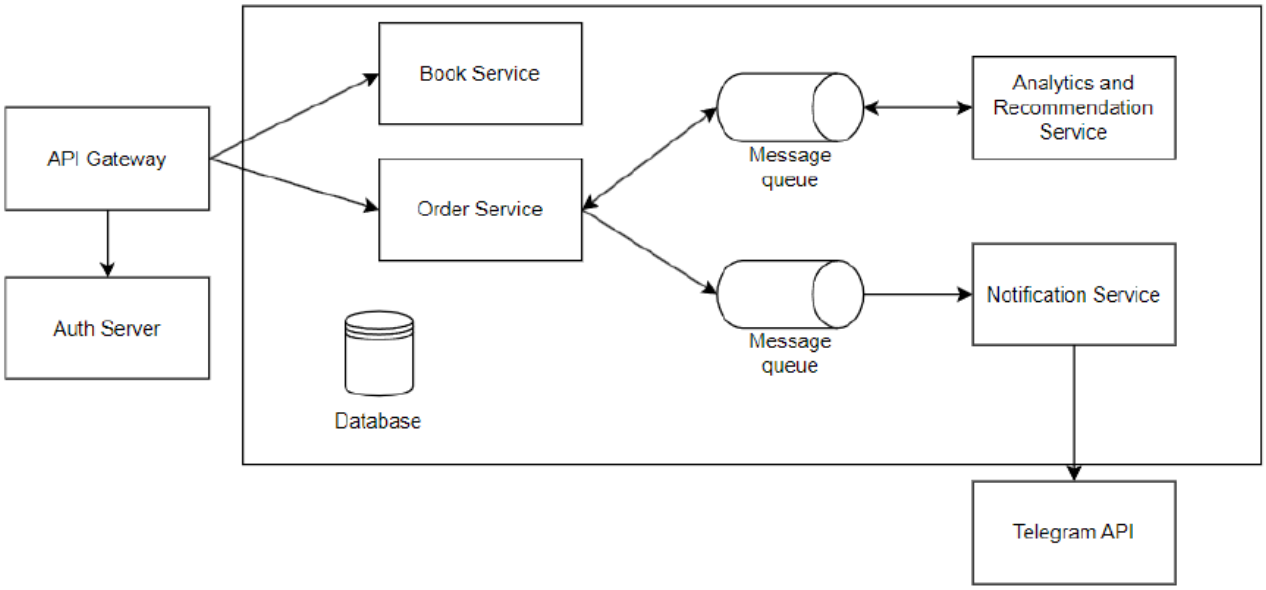


Рис. 2.1. Структурна діаграма автоматизованої системи

Автоматизована система управління бібліотекою побудована на сучасній мікросервісній архітектурі, де кожен компонент виконує чітко визначені функції. Така структура забезпечує масштабованість, надійність та зручність у розробці й супроводі. Нижче описано основні компоненти системи:

1. API Gateway. API-шлюз є вхідною точкою до всієї системи. Він приймає зовнішні запити від користувачів або інших сервісів, виконує первинну обробку запиту (наприклад, перевірку авторизації) та перенаправляє його до відповідного мікросервісу. API Gateway забезпечує централізований контроль доступу, балансування навантаження, логування та безпеку трафіку між зовнішнім світом і внутрішніми сервісами.

2. Auth Server. Сервер автентифікації відповідає за перевірку особистості користувача. Він реалізує механізми входу в систему, реєстрації, перевірки токенів доступу та розмежування прав доступу відповідно до ролей (читач, бібліотекар, адміністратор). Без успішної автентифікації користувач не може взаємодіяти з іншими компонентами системи.

3. Book Service. Цей компонент відповідає за облік бібліотечного фонду. До його функцій входять:

- додавання та редагування записів про книги;
- зберігання інформації про авторів, жанри, роки видання тощо;
- обробка запитів на пошук книг;
- перевірка доступності конкретного примірника.

Book Service працює з базою даних і також передає частину подій (наприклад, зміну стану книги) у чергу повідомлень для подальшої аналітики.

4. Order Service. Цей сервіс відповідає за роботу з замовленнями користувачів:

- реєстрація факту видачі книги;
- бронювання та продовження строку користування;
- фіксація повернення книги;
- відмітка про втрату або пошкодження.

Order Service також генерує події, що надсилаються до черги повідомлень для подальшої обробки в аналітичних і сповіщальних сервісах.

5. Message Queue (Черга повідомлень). Це інфраструктурний компонент, який дозволяє організувати асинхронну комунікацію між мікросервісами. Дані, які надходять у чергу, зберігаються до моменту обробки іншими службами, що гарантує стійкість системи до навантажень і збоїв. Черга використовується для надсилання подій до сервісів аналітики, рекомендацій та сповіщень.

6. Analytics and Recommendation Service. Цей сервіс відповідає за:

- аналіз користувацької поведінки (які книги читають, скільки часу триває користування тощо);
- статистику обігу фонду;

- створення персоналізованих рекомендацій для читачів на основі історії взаємодії з бібліотекою;
- формування управлінських звітів для адміністрації.

Отримує дані з Book Service та Order Service через чергу повідомлень.

7. Notification Service. Сервіс сповіщень інформує користувачів про події в системі, зокрема:

- нагадування про наближення терміну повернення книги;
- повідомлення про нові надходження;
- сповіщення про прострочення або зміни в бронюванні.

Цей компонент отримує дані з Order Service та надсилає повідомлення через зовнішні канали (наприклад, Telegram API).

8. Telegram API. Telegram API використовується для надсилання повідомлень користувачам у месенджері Telegram. Це дозволяє швидко, безпечно та зручно комунікувати з читачами, використовуючи вже знайомі їм засоби зв’язку. Інтеграція з Telegram підвищує зручність сервісу та стимулює користувачів активніше користуватись бібліотекою.

9. Database. Центральна база даних містить усю необхідну інформацію для функціонування системи:

- дані про книги та примірники;
- інформацію про користувачів;
- історію замовлень;
- службову інформацію (журнали подій, налаштування тощо).

Всі сервіси, що потребують зберігання або читання даних, взаємодіють з цією базою через відповідні API.

Описані компоненти спільно формують гнучку, масштабовану й сучасну систему для автоматизації процесів у бібліотеці. Кожен мікросервіс виконує чітко окреслену функцію, що полегшує її розвиток, тестування, модернізацію та впровадження нових можливостей у майбутньому.

Модель автоматизованої системи управління робою бібліотек з складається із ряду компонентів, таких як: інтерфейс користувача, клієнтська частина системи, серверна частина системи та база даних. Ці компоненти дозволяють забезпечити ефективну та відмовостійку працездатність системи.

Модель системи може бути представлена за допомогою схеми, яку зображено на рисунку 2.2.

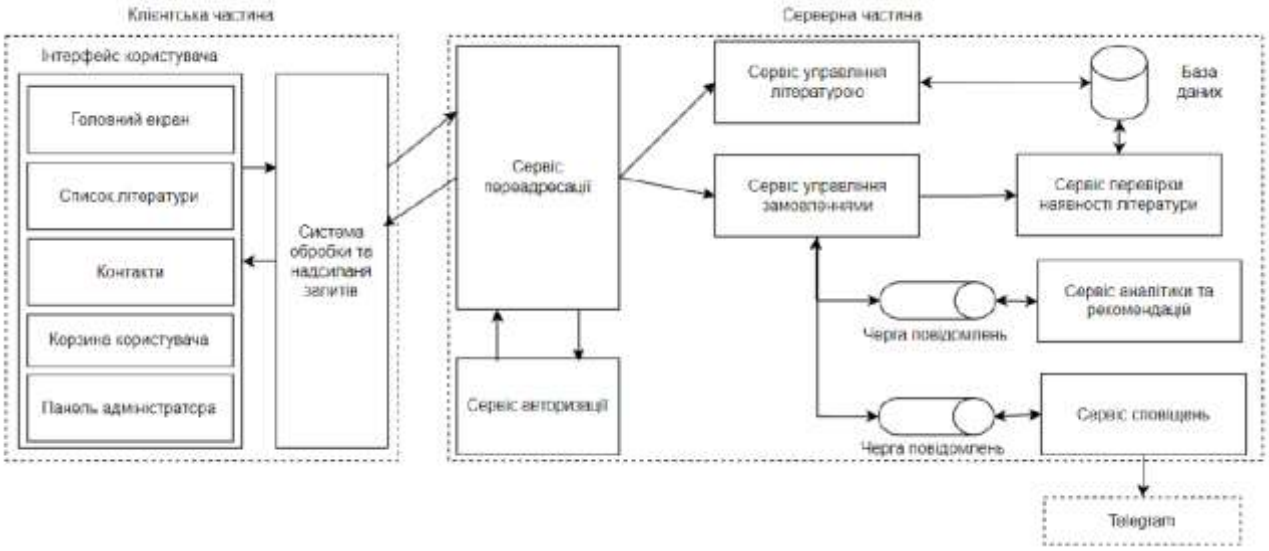


Рис. 2.2. Зображення системи

Архітектура розроблюваної системи побудована за принципом розподілення на клієнтську та серверну частини, що забезпечує ефективне функціонування системи, підвищену продуктивність, масштабованість і зручність в обслуговуванні.

Клієнтська частина — це інтерфейс, з яким безпосередньо взаємодіє кінцевий користувач: читач, бібліотекар чи адміністратор. Вона реалізує засоби для взаємодії з системою, надсилання запитів до серверної частини та отримання відповідей у зручному форматі.

Основні елементи клієнтської частини:

- Головний екран. Відображає загальну інформацію про бібліотеку, пошуковий рядок, популярну літературу та навігаційне меню.
- Список літератури. Дає змогу користувачеві переглядати каталог доступних книг, здійснювати пошук та фільтрацію за жанрами, авторами чи роком видання.

- Контакти. Забезпечує доступ до контактної інформації бібліотеки, включаючи адресу, години роботи, електронну пошту та інші канали зв’язку.
- Кошик користувача. Містить перелік книг, які користувач планує взяти у користування, забронювати або продовжити термін користування.
- Панель адміністратора. Призначена для бібліотекарів і адміністраторів. Містить інструменти для керування фондом, замовленнями, користувачами та налаштуваннями системи.
- Система обробки та надсилання запитів. Це проміжний компонент, який забезпечує взаємодію між інтерфейсом користувача та серверною частиною. Обробляє дії користувача, формує запити до серверу й відображає відповіді.

Серверна частина виконує всі основні логічні, аналітичні та управлінські функції системи. Вона приймає запити з клієнтської частини, обробляє їх і повертає відповідні результати.

Основні компоненти серверної частини:

- Сервіс переадресації. Центральний координаційний компонент, що отримує запити від клієнтської частини, розпізнає тип запиту та перенаправляє його до відповідного модуля обробки.
- Сервіс авторизації. Відповідає за автентифікацію користувачів (вхід, реєстрація, перевірка ролей). Працює спільно із сервісом переадресації для контролю доступу до функціоналу системи.
- Сервіс управління літературою. Обробляє дії, пов’язані з книжковим фондом: додавання, редагування, видалення, пошук, перегляд. Зберігає та отримує дані з бази даних.
- Сервіс управління замовленнями. Реалізує логіку взаємодії користувачів з літературою: видача, бронювання, повернення книг, продовження строків користування тощо.

- Сервіс перевірки наявності літератури. Перевіряє, чи доступна конкретна книга для видачі або бронювання. Працює у зв’язці з сервісами замовлень і літератури.
- Сервіс аналітики та рекомендацій. Збирає дані про дії користувачів, аналізує їх та формує рекомендації (наприклад, «Вам також може сподобатися...»). Працює з подіями, що надходять через чергу повідомлень.
- Сервіс сповіщень. Генерує повідомлення користувачам щодо важливих подій: наближення терміну повернення книги, нагадування, нові надходження тощо. Сповіщення надсилаються через зовнішні сервіси, зокрема Telegram.
- Черги повідомлень. Зв'язують основні бізнес-сервіси з сервісами аналітики та сповіщень. Дають змогу реалізувати асинхронну, безпечну та розвантажену передачу даних.
- База даних. Централізоване сховище всієї інформації системи: дані про книги, користувачів, замовлення, повідомлення, аналітику тощо.

Архітектура системи є логічно поділеною на фронтальну (клієнтську) та бекендну (серверну) частини, кожна з яких виконує специфічні функції. Такий підхід забезпечує гнучкість у розробці, підтримці та розширенні функціоналу системи, а також підвищує зручність користувача й ефективність роботи бібліотечного персоналу.

2.3. Проектування загального алгоритму роботи системи

Розробка загального алгоритму системи є фундаментальним кроком у створенні автоматизованої системи управління бібліотекою, оскільки він визначає логіку її роботи та послідовність взаємодії компонентів.

Для автоматизованої системи управління бібліотекою було розроблено загальний алгоритм роботи системи (рис. 2.3 та рис. 2.4).

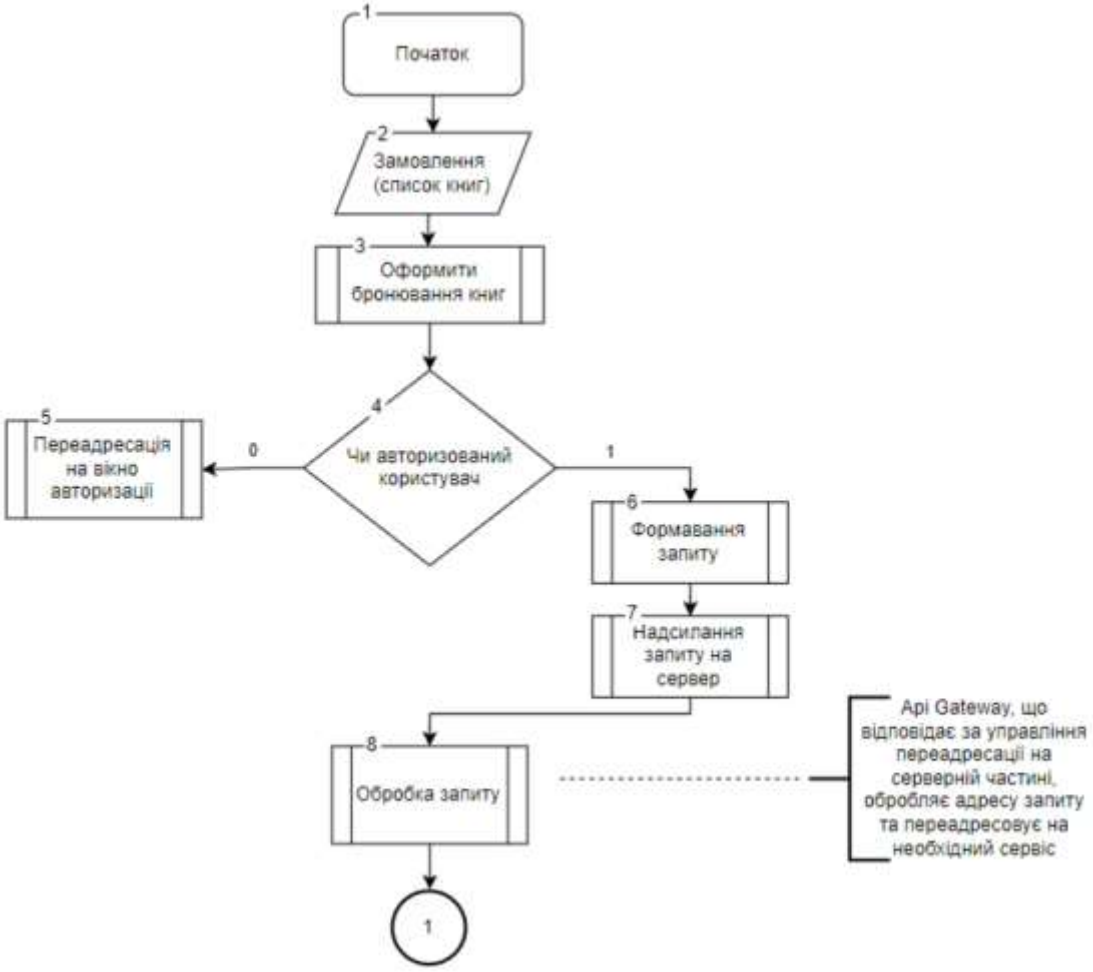


Рис. 2.3. Зображення загального алгоритму системи №1

На діаграмі зображено покроковий алгоритм взаємодії користувача з автоматизованою бібліотечною системою під час оформлення бронювання літератури. Процес складається з декількох етапів, починаючи від вибору книги та завершуючи обробкою запиту на сервері.

Опис етапів алгоритму:

1. Початок (1). Користувач запускає систему (вебзастосунок чи мобільний додаток) для взаємодії з бібліотекою.

2. Замовлення (список книг) (2). Відображається доступний каталог літератури. Користувач переглядає перелік книг, за потреби здійснює пошук, фільтрацію чи сортування.

3. Оформити бронювання книг (3). Користувач обирає одну або декілька книг для бронювання та ініціює відповідну дію.

4. Перевірка авторизації (4). Система перевіряє, чи є користувач авторизованим у системі.

- Якщо користувач не авторизований (0) – виконується переадресація на вікно авторизації (5), де користувач повинен ввести свої облікові дані.
- Якщо користувач авторизований (1) – система переходить до формування запиту.

5. Формування запиту (6). Система автоматично формує запит на бронювання обраної літератури з урахуванням ідентифікатора користувача та вибраних книг.

6. Надсилання запиту на сервер (7). Сформований запит передається на сервер через API Gateway – компонент, що керує перенаправленням запитів у серверній частині. Gateway обробляє адресу запиту та переспрямовує його до відповідного мікросервісу.

7. Обробка запиту (8). Відповідний серверний сервіс (наприклад, сервіс управління замовленнями) приймає запит, перевіряє умови доступності книг, оновлює статус літератури в базі даних і формує відповідь.

8. Завершення (1). Після успішної обробки запиту, користувач отримує повідомлення про підтвердження бронювання або повідомлення про неможливість виконання дії (у випадку, якщо книга вже недоступна).

У процесі взаємодії з сервером центральну роль відіграє API Gateway — це проміжна ланка між клієнтською частиною та серверними сервісами. Він виконує такі функції:

- перевіряє валідність запиту,
- обробляє маршрутизацію,
- переадресовує запит до потрібного мікросервісу (наприклад, сервісу управління замовленнями або перевірки наявності літератури).

Алгоритм забезпечує логічну, послідовну й безпечну взаємодію користувача із системою. Перевірка авторизації гарантує контроль доступу до функціоналу, а розподіл обов’язків між клієнтською частиною, API Gateway і

мікросервісами серверної частини забезпечує гнучкість і масштабованість архітектури бібліотечної системи

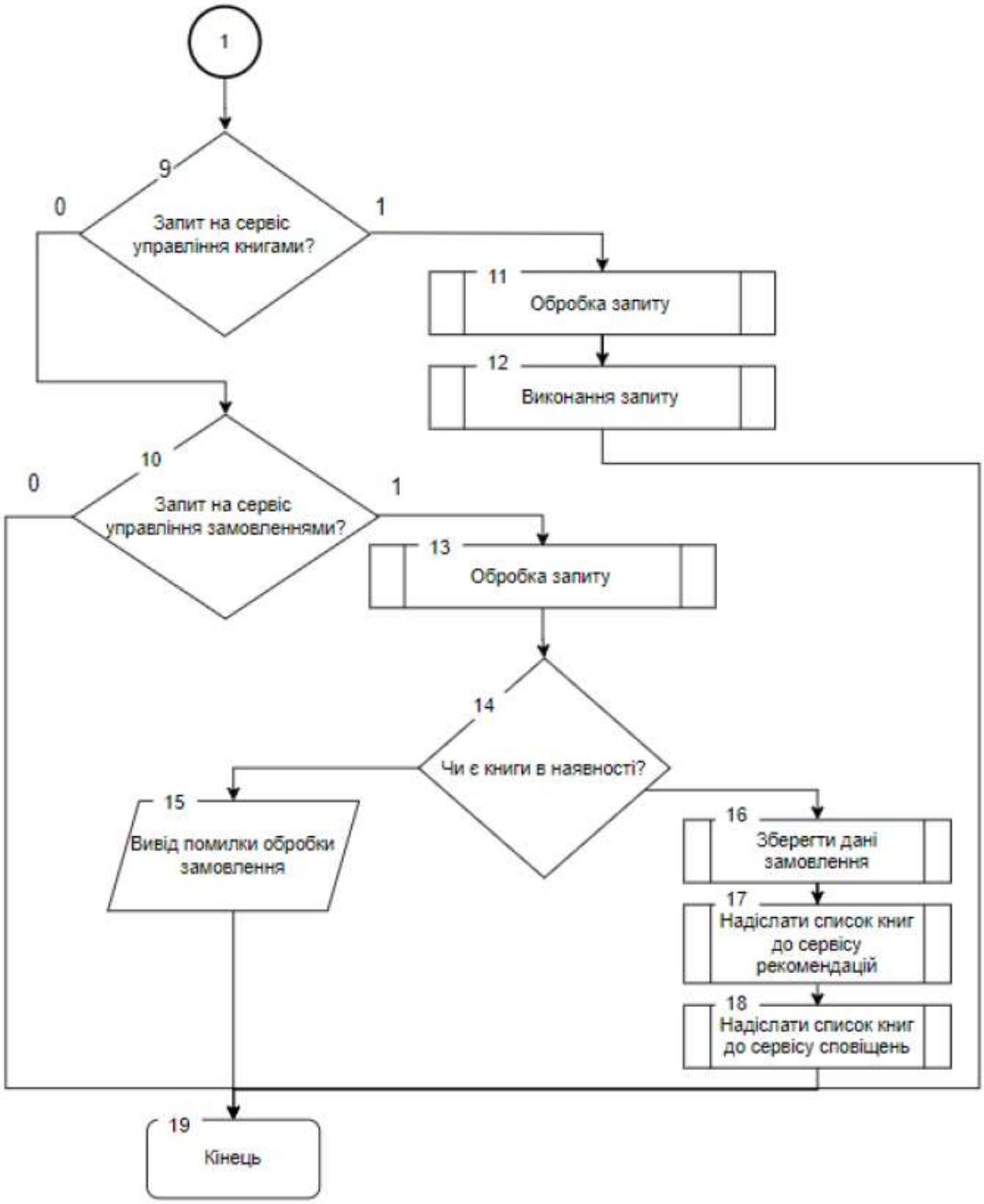


Рис. 2.4. Зображення загального алгоритму системи №2

Даний алгоритм описує логіку маршрутизації та обробки запитів у серверній частині автоматизованої системи управління бібліотекою. Запит, що надійшов від клієнта через API Gateway, надалі спрямовується до відповідного сервісу – залежно від його типу. Схема відображає, яким чином сервер розпізнає тип запиту, обробляє його, перевіряє наявність книг і передає дані до супутніх служб.

Етапи роботи алгоритму:

1. Початок (1). Сервер отримує запит, що був сформований клієнтською частиною через інтерфейс користувача та переданий через шлюз API.

2. Визначення типу запиту: сервіс управління книгами? (9). Система перевіряє, чи стосується запит управління каталогом книг (наприклад, додавання, редагування, видалення, перегляд книг).

- Якщо так (1) – виконується обробка запиту (11) та виконання відповідної дії (12) (наприклад, оновлення інформації у базі даних), після чого процес завершено.
- Якщо ні (0) – перевіряється інший варіант.

3. Чи це запит до сервісу управління замовленнями? (10). Якщо запит не пов’язаний із книгами, система перевіряє, чи стосується він обробки замовлень на літературу (бронювання книг, перегляд історії тощо).

- Якщо запит не відповідає і цьому критерію – завершується з помилкою.
- Якщо так (1) – система переходить до обробки замовлення (13).

4. Перевірка наявності книг (14). Після обробки запиту на бронювання система перевіряє, чи доступна запитана література.

- Якщо книги відсутні – система генерує повідомлення про помилку обробки замовлення (15) і завершує процес (19).
- Якщо книги наявні – виконується збереження замовлення.

5. Подальші дії при успішному бронюванні:

- Збереження даних замовлення (16) — інформація про користувача та бронювання записується до бази даних.
- Надсилання списку книг до сервісу аналітики та рекомендацій (17) — ці дані використовуються для формування рекомендацій (наприклад, схожі книги або популярні вибори).
- Надсилання списку книг до сервісу сповіщень (18) — запускається механізм інформування користувача (наприклад, через Telegram).

6. Кінець процесу (19). Після завершення всіх дій система повертає відповідь клієнтській частині й завершує виконання.

Описаний алгоритм реалізує чітке розгалуження логіки обробки запитів на сервері згідно з принципами мікросервісної архітектури. Кожен тип запиту (керування літературою або замовленнями) направляється до свого сервісу, а у випадку успішного замовлення система взаємодіє з модулями аналітики та сповіщень, забезпечуючи розширену функціональність. Такий підхід дозволяє забезпечити масштабованість, гнучкість і стійкість системи.

2.4. Проєктування макетів графічного інтерфейсу користувача

Інтерфейс користувача відповідає за взаємодію користувачів з системою. У проекті автоматизованої системи управління бібліотекою, інтерфейс користувача виконує такі ключові функції: візуалізація інформації, формування макету для створення бронювань та авторизації користувача.

Візуалізація інформації. Інтерфейс відображає всю необхідну інформацію про книги, включаючи назву, автора, жанр. Це дозволяє користувачам швидко та зручно знаходити потрібні книги. Для кращого представлення компонентів, що відповідають за візуалізацію даних, було розроблено схему, що зображена на рисунку 2.5.

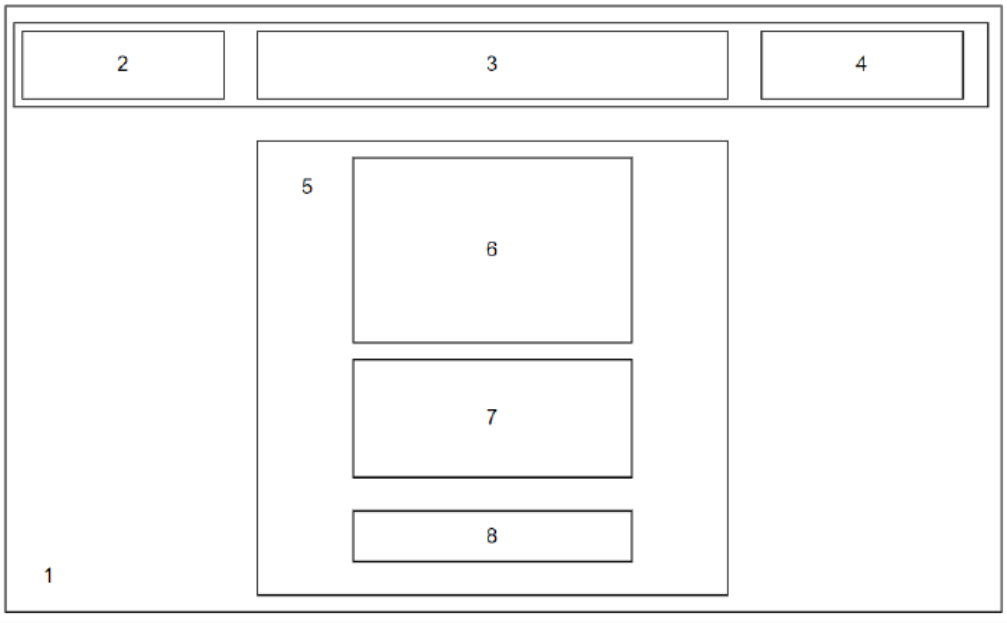


Рис. 2.5. Зображення схеми побудови екрану інтерфейсу користувача, що відповідає за відображення списку книг

Опис усіх елементів схеми:

1. Головний екран сторінки.

2. Логотип програмного застосунку (у реалізованому варіанті представлений, як «Книжка» + «Система управління бібліотекою»).

3. Навігаційна панель користувача, що включатиме наступні елементи: головна сторінка, сторінка списку літератури та сторінка, що відповідає за контакти розробника.

4. Додаткові функціональні елементи – це елементи, що відповідають за авторизацію користувача, можливість зміни теми веб сайту (можливі варіанти тем: день, ніч), функціональний елемент корзини замовлення та елемент переходу до сторінки адміністратора.

5. Стандартизований контейнер подання інформації про книгу, він використовується для налаштування подання інформації про книги структуровано списком чи у табличному вигляді.

6. Зображення книги.

7. Інформація про книгу, секція інформації про книгу міститиме таку інформацію як: назва книги, автор, короткий опис та жанр.

8. Функціональний елемент «Додати книгу» - це кнопка, що дозволяє додати книгу до списку замовлення.

Створення бронювань. Інтерфейс дозволяє користувачам створювати бронювання на книги, які наразі недоступні. Це забезпечує зручність для читачів, оскільки вони можуть бути впевнені, що отримають потрібну книгу, коли вона стане доступною.

Авторизація користувачів. Інтерфейс забезпечує можливість користувачам авторизуватися в системі, використовуючи свої облікові дані (логін та пароль). Це дозволяє відстежувати історію користування кожного читача, надавати персоналізовані рекомендації та контролювати доступ до певних функцій системи, таких як створення бронювань та використання панелі адміністратора. Для кращого представлення компонентів, що відповідають за авторизацію користувачів, було розроблено схему, що зображена на рисунках 2.6 та 2.7.

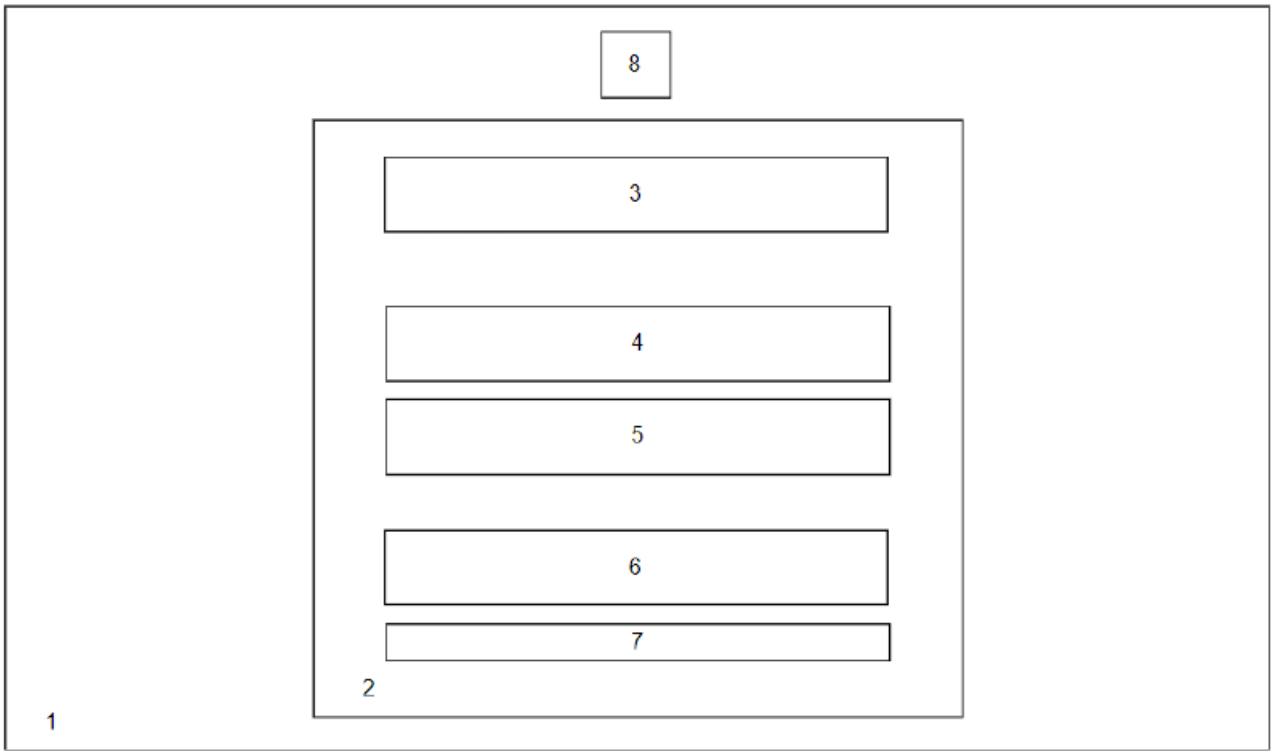


Рис. 2.6. Зображення схеми побудови вікна входу в існуючий аккаунт

Опис усіх елементів схеми:

1. Головний екран.
2. Контейнер, що відповідає за елементи вікна авторизації.
3. Елемент заголовку, що вказуватиме на призначення вікна авторизації.
4. Поле вводу даних електронної пошти користувача для авторизації.
5. Поле вводу даних паролю користувача для авторизації.
6. Секція для переадресації на реєстрацію користувача.
7. Функціональний елемент входу в обліковий запис.
8. Функціональний елементи закриття вікна авторизації

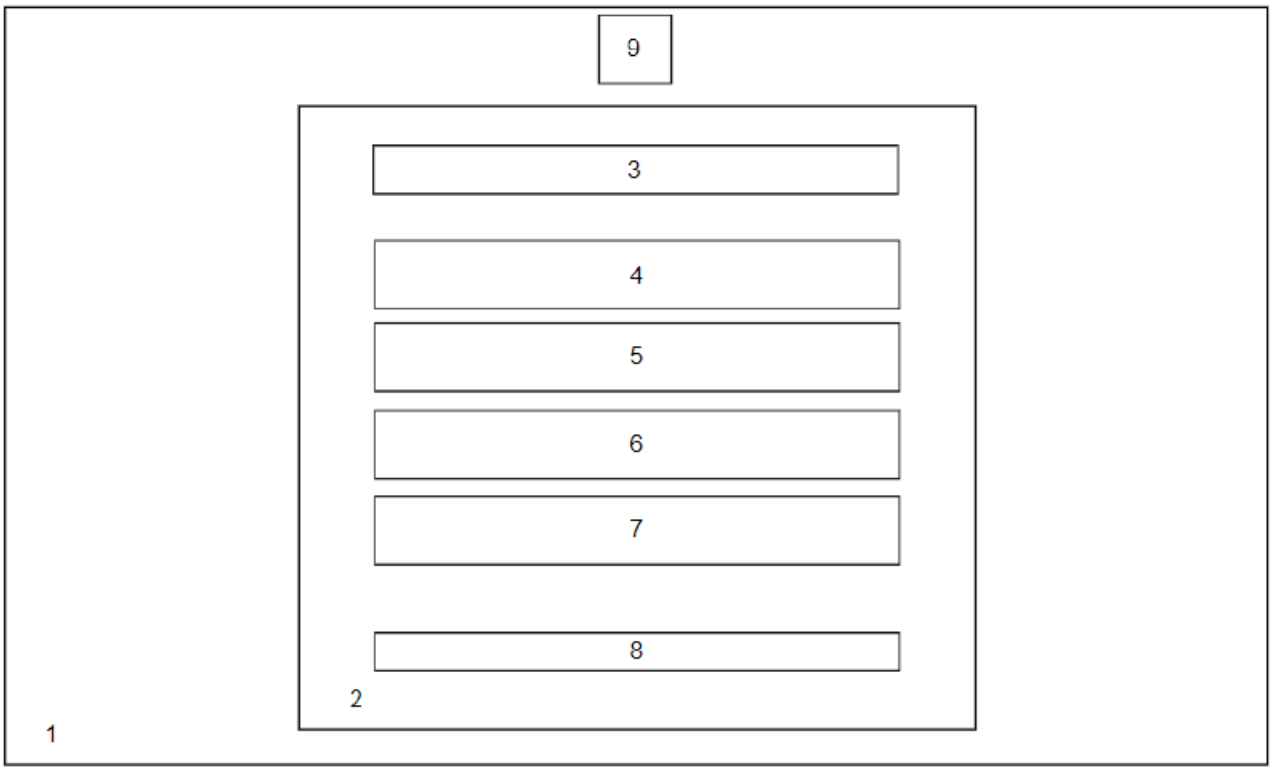


Рис. 2.7.Зображення схеми побудови вікна реєстрації нового користувача

Опис усіх елементів схеми:

1. Головний екран.
2. Контейнер, що відповідає за елементи вікна авторизації.
3. Елемент заголовку, що вказуватиме на призначення вікна реєстрації.
4. Поле вводу даних електронної пошти користувача для реєстрації.
5. Поле вводу даних номера телефону користувача для реєстрації.
6. Поле вводу даних імена та прізвища користувача для реєстрації.
7. Поле вводу даних пароля користувача для реєстрації.
8. Функціональний елемент входу в обліковий запис.
9. Функціональний елементи закриття вікна авторизації.

Отже, у підрозділі було описано особливості розробки графічного інтерфейсу користувача для системи та зображено приклади схематичного позиціювання графічних компонентів.

2.5. Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

При проєктуванні та реалізації автоматизованої системи управління бібліотекою надзвичайно важливим є вибір мови програмування, яка відповідатиме вимогам до надійності, масштабованості, зручності супроводу та підтримки сучасних технологій. Серед найбільш поширених мов, які застосовуються в розробці таких систем, варто розглянути Java, C# та Python. Кожна з цих мов має свої переваги, обмеження та сферу оптимального використання, тому порівняння між ними є доцільним для обґрунтованого вибору.

Java є однією з найбільш стабільних та поширених мов програмування, яка активно використовується для створення корпоративних систем, у тому числі систем автоматизації бібліотек. Однією з головних переваг Java є її платформонезалежність. Завдяки віртуальній машині Java (JVM), додатки, написані на цій мові, можуть працювати на різних операційних системах без необхідності модифікації коду [13].

Крім того, Java має велику кількість готових фреймворків, які значно спрощують розробку. Наприклад, Spring Framework дозволяє будувати потужні веб-додатки з чіткою архітектурою, зокрема за мікросервісним підходом, що є актуальним для сучасних бібліотечних систем. Завдяки підтримці багатопотоковості Java здатна ефективно обробляти велику кількість паралельних запитів, що важливо при роботі з одночасним доступом багатьох користувачів [13].

Однак Java має і певні недоліки. Одним із них є складніший у порівнянні з іншими мовами синтаксис, що може ускладнювати навчання та пришвидшення розробки. Також створення навіть простих функцій потребує більше рядків коду, ніж, наприклад, у Python [13].

У цілому, Java є надійним вибором для розробки серверної частини системи, особливо якщо проект має на меті розгортання на різних платформах та потребує високого рівня продуктивності.

Мова програмування C# є основним інструментом розробки в середовищі Microsoft. Вона була створена як альтернатива Java й має багато спільних рис з нею, хоча відрізняється глибокою інтеграцією з платформою .NET. C# вирізняється сучасним і лаконічним синтаксисом, що робить розробку більш швидкою та зручною. Завдяки розвинутому середовищу розробки Visual Studio програміст має у своєму розпорядженні потужні інструменти для налагодження, тестування та візуального проєктування інтерфейсів [14].

Однією з вагомих переваг C# є тісна інтеграція з Windows, що робить його оптимальним вибором у випадку, коли система розгортається в корпоративному середовищі на базі ОС Windows. У цьому разі доступ до баз даних, мережевих ресурсів та служби Windows реалізується швидко й ефективно. Також платформа .NET Core (і її наступник — .NET 6/7) розширила можливості розробки, дозволяючи створювати кросплатформенні веб- та десктоп-додатки [14].

До недоліків мови C# можна віднести її часткову залежність від екосистеми Microsoft, що іноді обмежує вибір хостингів або серверного оточення. Проте ці обмеження з кожним роком стають менш актуальними завдяки розвитку .NET як відкритої кросплатформеної технології [14].

Таким чином, C# є чудовим варіантом для розробки бібліотечних систем, особливо якщо передбачається інтеграція з Windows-середовищем або використання хмарних сервісів Azure [14].

Python є мовою програмування, яка здобула величезну популярність завдяки своїй простоті та універсальності. Її синтаксис є максимально наближеним до природної мови, що дозволяє швидко розробляти прототипи та мінімізує кількість помилок у коді. Python активно використовується в галузях аналітики даних, машинного навчання, веб-розробки, а також у створенні мікросервісних архітектур [15].

У контексті автоматизації бібліотек Python може бути особливо корисним при реалізації сервісів аналітики, рекомендацій або статистичної обробки користувацьких запитів. Фреймворки Django або Flask дозволяють реалізувати серверну частину веб-додатку за досить короткий термін. Крім того, Python має

потужні засоби інтеграції з іншими мовами, що дозволяє використовувати його в гібридних рішеннях.

До недоліків Python належать менша продуктивність у порівнянні з Java або C#, оскільки він є інтерпретованою мовою. У випадку високонавантажених систем це може бути критичним, тому Python краще використовувати у тих частинах системи, де швидкодія не є ключовою вимогою. Також Python менш підходить для побудови великих монолітних додатків, хоча у поєднанні з мікросервісною архітектурою ці обмеження можна обійти [15].

Загалом, Python є чудовим вибором для реалізації аналітичних модулів, служб рекомендацій або навіть для повноцінної реалізації невеликої бібліотечної системи з відкритим кодом.

У підсумку, кожна з розглянутих мов має свої сильні сторони та сценарії ефективного застосування. Java є найкращим варіантом для створення надійних, масштабованих серверних рішень із високим навантаженням. C# доцільно використовувати в інтеграції з корпоративними Windows-середовищами та для розробки хмарних сервісів. Python, своєю чергою, є незамінним інструментом у швидкій розробці, побудові прототипів, аналітичних модулів і додатків з невеликим навантаженням.

У багатьох сучасних проєктах застосовується комбінований підхід, коли різні частини системи реалізуються з використанням різних мов: наприклад, основна логіка на Java, аналітика на Python, а клієнтський додаток – на JavaScript. Такий підхід дозволяє максимально ефективно використати переваги кожної технології та побудувати гнучку, стійку й ефективну систему.

У процесі розробки автоматизованих систем управління бібліотеками (АСУБ) важливе місце займає вибір програмного середовища, в якому здійснюється проєктування, написання та налагодження програмного коду. В залежності від архітектури системи, технологічного стеку та вимог до продуктивності, розробник може використовувати різні середовища розробки. Найчастіше при створенні серверної логіки застосовуються мови програмування Java, Python, C#, а для клієнтської частини — JavaScript та його фреймворки. У

цьому контексті доцільно порівняти можливості трьох найпоширеніших інструментів: Eclipse, IntelliJ IDEA та Visual Studio Code (VSCode).

Порівняльний аналіз мов програмування наведено в таблиці 2.1.

2.1. Порівняння мов програмування

Критерій	Java	C#	Python
Паралелізм та багатопоточність	+	+	+
Платформонезалежність	+	-	+
Підтримка інструментів та бібліотек	+	+	+
Спільнота розробників	+	+	+
Надійність та безпека	+	+	+
Масштабованість	+	+	-
Підсумок	6	5	5

Eclipse — це класичне інтегроване середовище розробки, яке має глибоку інтеграцію з мовою Java, тому часто використовується для створення корпоративних додатків. Це середовище підтримує роботу з фреймворками на кшталт Spring, Hibernate, Maven, а також дозволяє працювати з сервлетами, що актуально для створення серверної частини бібліотечних систем. Eclipse підходить для командної розробки складних монолітних або мікросервісних систем, але вимагає детального налаштування та іноді може бути повільним у роботі, особливо на слабких машинах. Попри це, наявність великої кількості плагінів та відкритий вихідний код роблять його доволі гнучким і придатним для проєктів з обмеженим бюджетом.

На відміну від Eclipse, IntelliJ IDEA — це сучасне комерційне середовище, орієнтоване на продуктивність розробника. Його безкоштовна версія (Community Edition) вже містить базову підтримку Java та Kotlin, а платна версія (Ultimate Edition) відкриває доступ до повного спектра інструментів для веб-розробки, роботи з базами даних, REST API та інших компонентів, які часто використовуються у системах управління бібліотеками. IntelliJ автоматично розпізнає структуру проєкту, пропонує інтелектуальні підказки, оптимізує імпорти та надає зручний інтерфейс налагодження, що суттєво скорочує час розробки. Це середовище найкраще підходить для реалізації складної бізнес-

логіки, інтеграції з хмарними рішеннями, сервісами рекомендацій або Telegram-ботами, які можуть входити до складу АСУБ.

Visual Studio Code (VSCode) є текстовим редактором з великими можливостями розширення. Хоча спочатку він був орієнтований на фронтенд-розробку, завдяки численним плагінам VSCode можна використовувати як повноцінне середовище для розробки як серверної, так і клієнтської частини систем. Наприклад, за допомогою відповідних розширень можна організувати середовище для розробки на Java, Python, JavaScript або навіть C#. У випадку бібліотечних систем, VSCode буде особливо корисним для реалізації інтерфейсу користувача — адаптивного веб-застосунку, інтеграції з REST API, підключення до бази даних або взаємодії з телеграм-ботами. Його перевагою є невелика вага, швидкість запуску та простота налаштування. Однак, для складних систем із великою кількістю взаємопов’язаних модулів, VSCode може виявитися менш зручним, ніж повноцінні IDE, оскільки вимагає ручного налаштування проєкту, залежностей, компіляторів тощо.

Загалом, для розробки повноцінної багаторівневої автоматизованої системи управління бібліотекою, яка включає як серверну, так і клієнтську частини, найкращим вибором стане IntelliJ IDEA. Це середовище поєднує в собі гнучкість, високу продуктивність та зручність у розробці складної бізнес-логіки. Eclipse залишається надійним варіантом у проєктах, де переважає використання Java і потрібно суворе дотримання архітектурних шаблонів. VSCode, у свою чергу, найдоцільніше використовувати для швидкої розробки веб-інтерфейсів, REST-клієнтів, чат-ботів або мікросервісів на легких технологіях.

Таким чином, вибір середовища розробки має ґрунтуватися на архітектурі системи, використовуваних мовах програмування, потребі в інтеграції з хмарними сервісами або повідомленнями, а також на вимогах до продуктивності та гнучкості середовища.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка інтерфейсу користувача

Розробка інтерфейсу користувача (UI) є критично важливим етапом у створенні будь-якої програмної системи, включаючи автоматизовану систему управління бібліотекою. UI виступає мостом між користувачем та функціональністю системи, забезпечуючи ефективну та приємну взаємодію.

Для автоматизованої системи управління робою бібліотек було розроблено наступні компоненти інтерфейсу користувача: головний екран, екран списку літератури та екран контактів розробника. Інтерфейс користувача системи автоматизації управління бібліотеки було реалізовано за допомогою технологій HTML, CSS та Javascript. Оскільки HTML, CSS та JavaScript є ідеальним поєднанням для створення інтерфейсу користувача автоматизованої системи управління бібліотекою завдяки своїй універсальності, доступності та гнучкості. Ці стандартні веб-технології підтримуються всіма сучасними браузерами, що забезпечує кросплатформеність та доступність системи для користувачів з різних пристроїв. Користувачам не потрібно встановлювати додаткове програмне забезпечення, що робить систему більш зручною. HTML дозволяє створювати структуру сторінок, CSS відповідає за їх візуальне оформлення, а JavaScript забезпечує інтерактивність та динамічну поведінку. Разом вони дозволяють створювати адаптивні інтерфейси, які підлаштовуються під різні розміри екранів та пристроїв [20].

Головний екран користувача – це центральний елемент інтерфейсу, який надає доступ до основних функцій системи. Головний екран користувача системи управління бібліотекою зображено на рисунку 3.1 та 3.2.

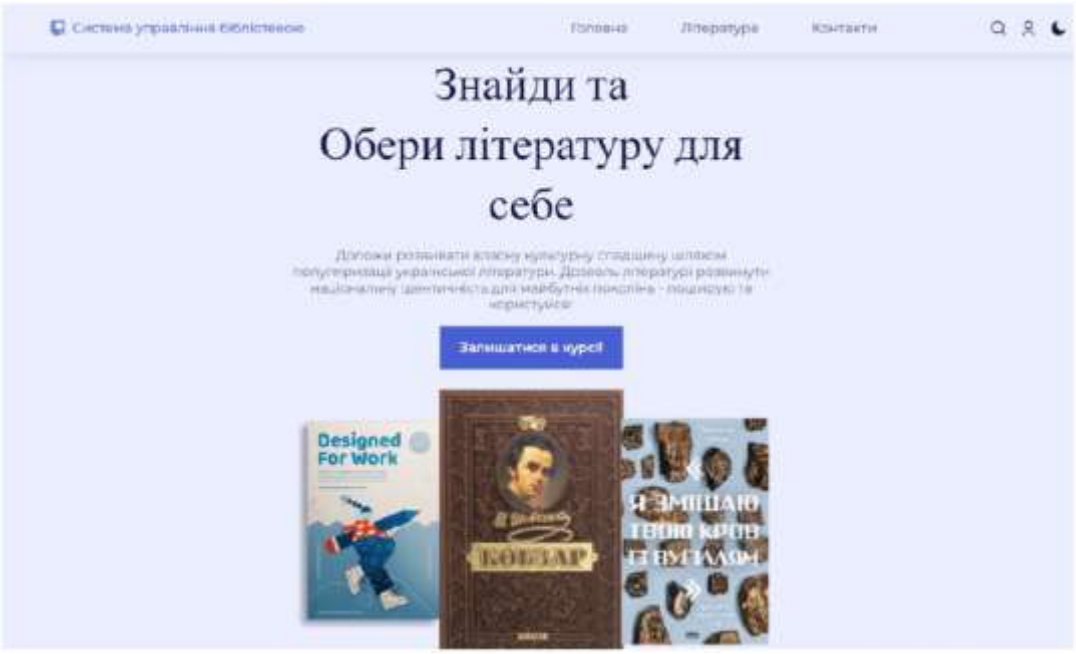


Рис. 3.1. Зображення головного екрану системи частина 1

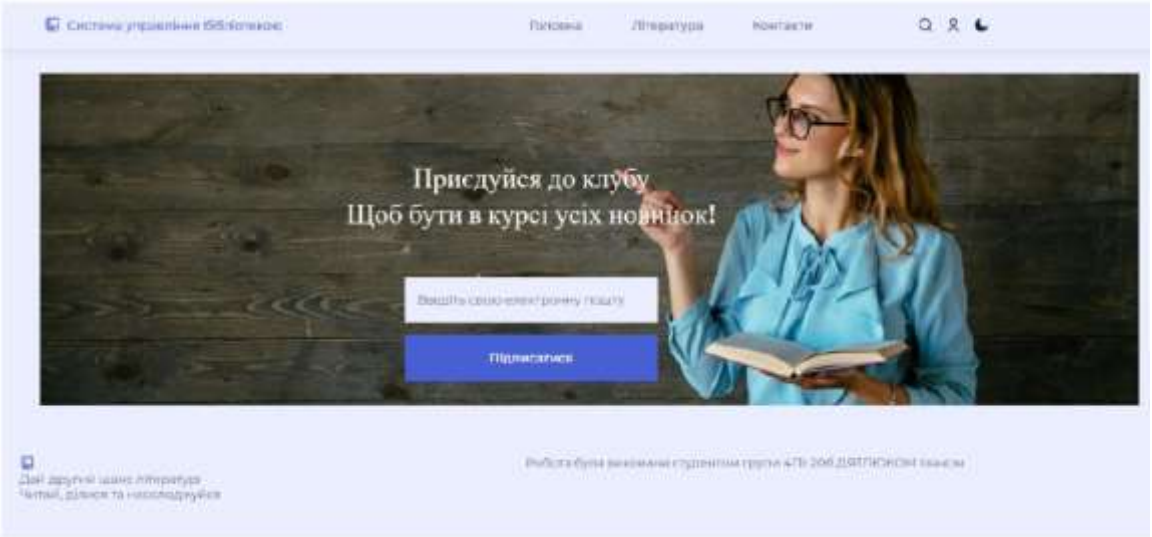


Рис. 3.2. Зображення головного екрану системи частина 2

Ця веб-сторінка є головною сторінкою системи управління бібліотекою. Вона містить основне меню з посиланнями на інші розділи сайту, такі як "Головна", "Література" та "Контакти". На головній сторінці банер з гаслом бібліотеки та кнопкою "Залишатися в курсі!". Нижче розташований слайдер з зображеннями книг, який демонструє новинки або популярні видання. Також є розділ для приєднання до клубу, де користувачі можуть залишити свою електронну пошту для отримання новин. Сторінка має входу та реєстрації користувачів, а також можливість зміни теми оформлення (світла/темна) (див.

рис. 3.3). Ці функції реалізовані за допомогою JavaScript, який забезпечує інтерактивність сторінки. В цілому, сторінка має простий та зрозумілий дизайн, фокусуючись на основних функціях системи управління бібліотекою.

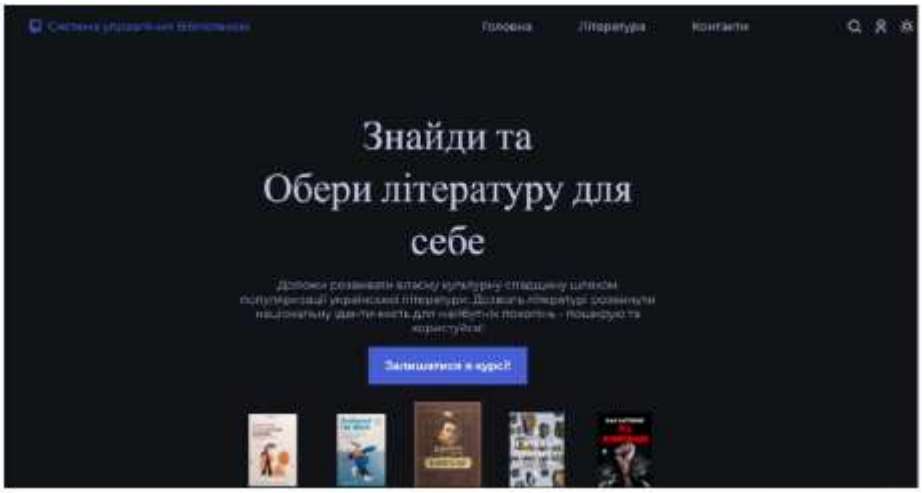


Рис. 3.3. Зображення головного екрану з використанням темної теми

Екран списку літератури – це екран, що надає користувачеві усю необхідну та актуальну інформацію про наявні книги, на цьому екрані зображено список усіх доступних книг із додатковою інформацією.

Ця веб-сторінка є каталогом книг в автоматизованій системі управління бібліотекою. У верхній частині сторінки розташовано логотип бібліотеки, меню для навігації по сайту (з посиланнями на головну сторінку, список літератури та контакти), а також іконки для пошуку, перегляду кошика, входу/реєстрації та зміни теми оформлення сайту.

Основна частина сторінки містить заголовок "Список літератури" та список книг у вигляді карток. Кожна картка містить зображення обкладинки книги, її назву, автора та короткий опис. Також є кнопка "Додати", яка дозволяє користувачам додавати книгу до свого кошика для подальшого замовлення (див. рис. 3.4).

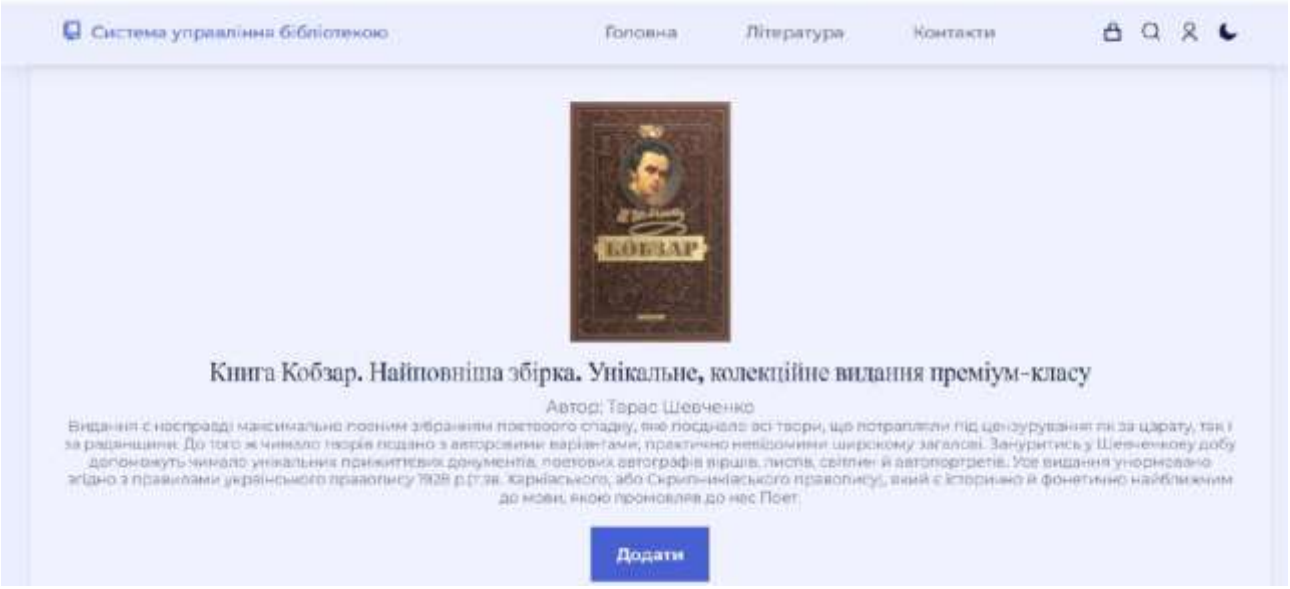


Рис. 3.4. Зображення сторінки «Література»

Екран «Контакти» сторінка, що надає інформацію про розробника системи управління бібліотекою, включаючи його ім'я, групу, короткий опис проекту та контактну інформацію. Сторінка також містить стандартні елементи, такі як шапка, пошук, форми авторизації та реєстрації, та підвал, що забезпечує зручну навігацію по сайту та доступ до основних функцій.

Для повноцінної роботи користувача з системою, йому необхідно провести авторизацію (за наявності облікового запису в систему) або реєстрації (у випадку якщо користувач користується системою вперше).

Зображення із вікнами авторизації та реєстрації продемонстровано на рисунках 3.5 та 3.6.

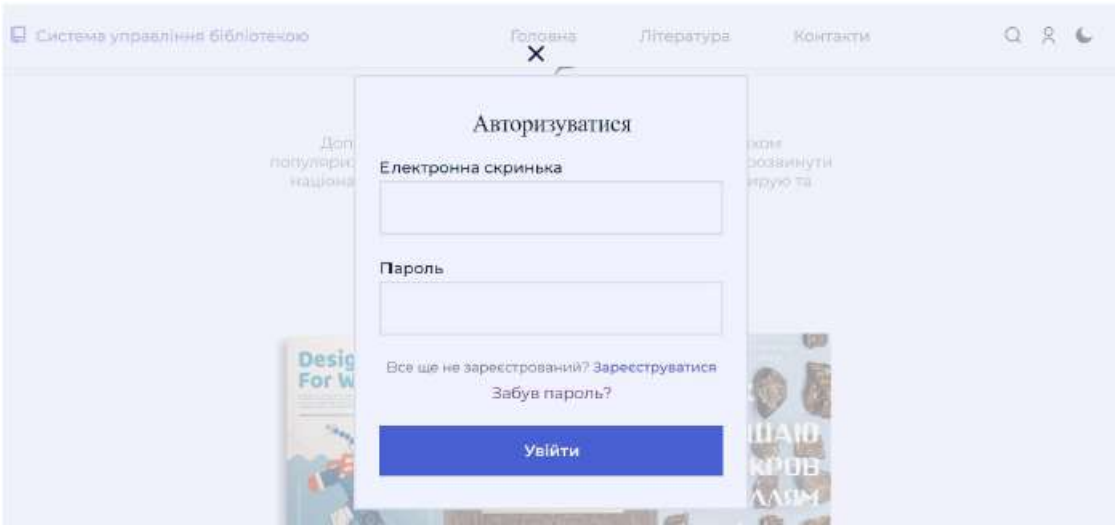


Рис. 3.5. Зображення вікна авторизації

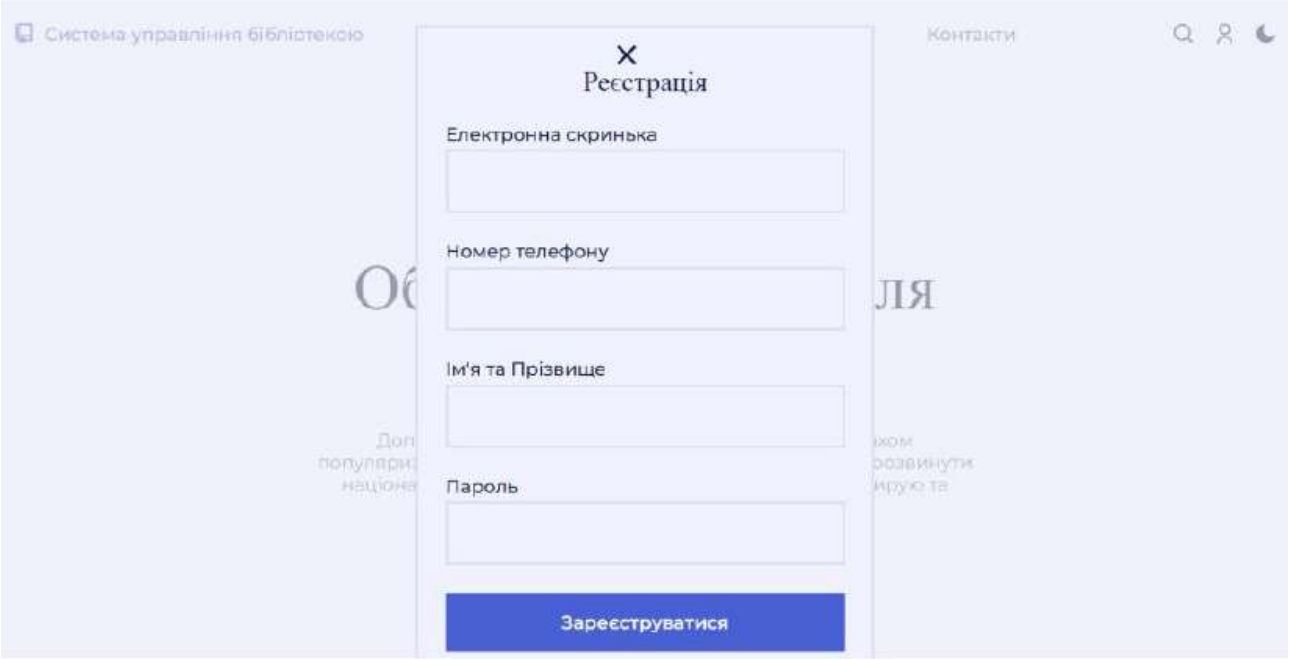


Рис. 3.6. Зображення вікна реєстрації

Для авторизації користувача в системі йому необхідно ввести дані уже існуючого аккаунту, а саме дані про електронну скриньку користувача та його пароль в системі, після чого користувач отримує доступ до можливості формування замовлень. У випадку, якщо користувач користується системою вперше йому необхідно натиснути на виділене синім кольором слово «Зареєструватися» (рис. 3.7).

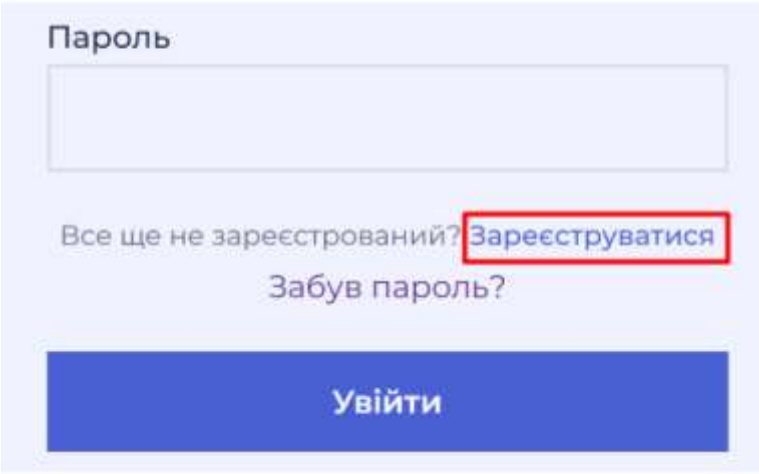


Рис. 3.7. Зображення вказівки для подальшої реєстрації користувача

Після успішної авторизації/реєстрації користувач матиме змогу для подальшого формування замовлення.

Для формування замовлення користувачеві необхідно перейти на сторінку списку літератури, та за допомого кнопки «Додати», вибрати бажану кількість книг (рис. 3.8).

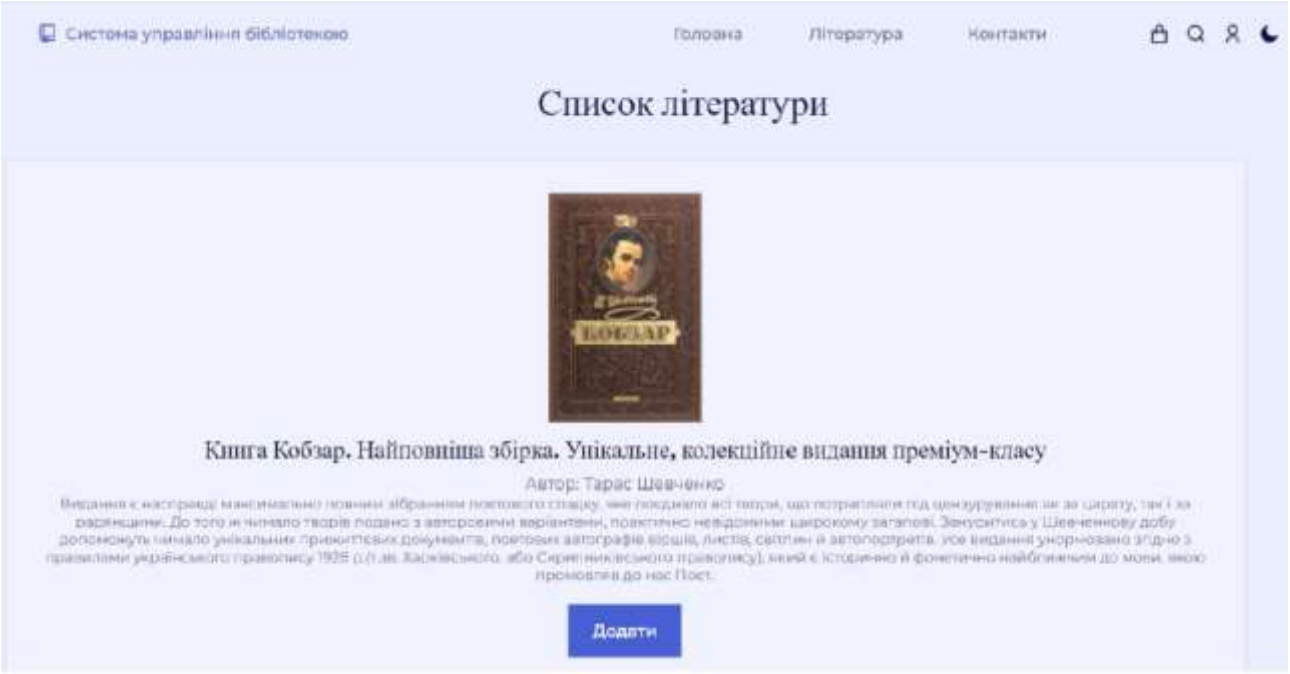


Рис. 3.8. Зображення прикладу сторінки «Література»

Після натиску на кнопку «Додати» у авторизованого користувача з’являється можливість підтвердити замовлення, натиснувши на іконку корзини (рис. 3.9).



Рис. 3.9. Зображення вказівки для підтвердження замовлення

У меню корзини користувач має змогу видалити елемент із неї або підтвердити свій вибір, натиснувши на кнопку «Підтвердити» (рис. 3.10).

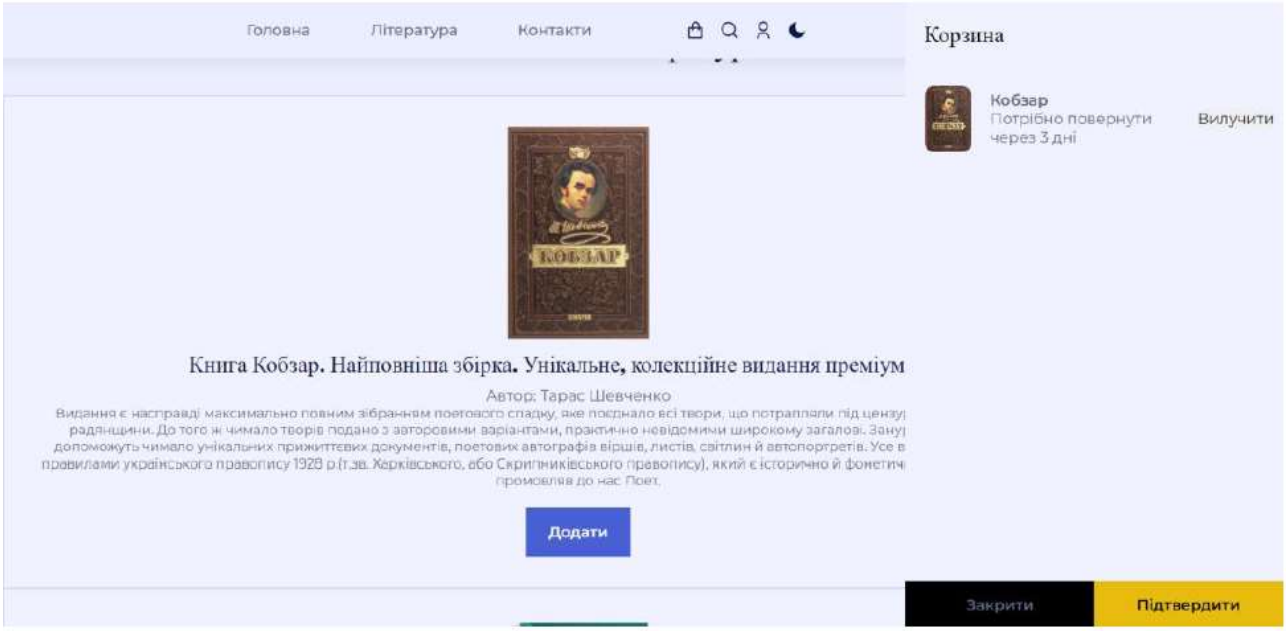


Рис. 3.10. Зображення графічного елементу корзини

Для подальшого отримання сповіщень, користувачу необхідно під’єднатися до Telgam боту: https://t.me/libraryNBot (рис. 3.11).

Також, можлива авторизація користувача із привілейованими правами доступу, які передбачаються під час розробки системи. Привілейовані права надають користувачу можливість отримувати додаткові аналітичні дані з системи (рис. 3.12).

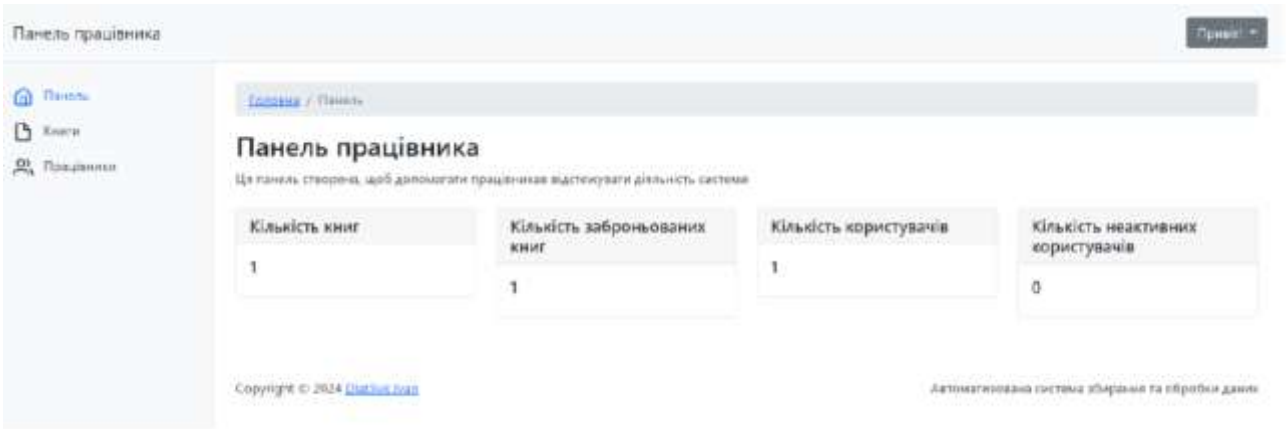


Рис. 3.12. Зображення панелі працівника

Уся необхідна інформація знаходиться на сторінці «Панель», тут наводяться такі дані:

- кількість книг – це загальна кількість книг, яких було заброньовано чи повернуто за весь час роботи системи;

- кількість заброньованих книг – це загальна кількість книг, які наразі заброньовані користувачами;

- кількість користувачів – це кількість активних користувачів, що бронюють книги впродовж заданого часу (1.5 роки);

- кількість неактивних користувачів – це кількість користувачів, які деактивували свої облікові записи або не проводили ніяких операцій у системі протягом заданого часу (1.5 роки).

Отже, у підрозділі роботи, було обґрунтовано розробку важливість розробки інтерфейсу користувача для системи управління робою бібліотеки та детально описано розроблюванні сторінки для системи.

3.2. Розробка сервісу обробки даних та сервісу сповіщень

Розробка сервісу обробки даних та формування рекомендацій – це одне із основних завдань проєкту, що полягає у налаштуванні окремого сервісу, що відповідатиме за зчитування даних для подальшого формування персоналізованих рекомендацій.

Сервіс обробки даних та формування рекомендацій був розроблений з використанням Apache Kafka, потужної платформи обміну повідомленнями. До проекту Spring Boot була додана залежність spring-kafka, що надала інструменти для роботи з Kafka.

Була створена конфігурація споживача, де визначили ключові параметри, такі як адреси Kafka брокерів, унікальний ідентифікатор групи споживачів, а також правила для розшифровки ключів та значень повідомлень.

Після цього був створений сам споживач, який був підписаний на тему в Kafka, що в свою чергу отримувала повідомлення від сервісу управління замовленнями. Цей споживач почав отримувати повідомлення з цієї теми та обробляти їх відповідно до логіки сервісу. Отримавши повідомлення, споживач аналізує дані, формує персоналізовані рекомендації для користувачів та зберігає

їх у базі даних. Ці рекомендації потім використовуються системою для надання користувачам релевантних пропозицій щодо книг. Щоб забезпечити високу продуктивність та масштабованість, споживач налаштований на роботу в групі споживачів. Це дозволяє розподілити навантаження між кількома екземплярами сервісу та забезпечити ефективну обробку великої кількості повідомлень.

Таким чином, завдяки інтеграції з Apache Kafka та ретельно продуманому алгоритму споживача, сервіс обробки даних та формування рекомендацій забезпечує користувачам бібліотеки персоналізований та зручний досвід взаємодії, пропонуючи їм книги, які відповідають їхнім інтересам та вподобанням. Завдяки інтеграції з Apache Kafka та ретельній розробці алгоритму споживача, сервіс обробки даних та формування рекомендацій став невід'ємною частиною автоматизованої системи управління бібліотекою, забезпечуючи її користувачам персоналізований та зручний досвід взаємодії.

Сервіс сповіщень є важливою частиною автоматизованої системи управління бібліотекою, забезпечуючи зв'язок між системою та користувачами через Telegram. Його основна мета — своєчасно інформувати читачів про важливі події, такі як затримка повернення книг та надсилання персоналізованих рекомендацій.

Розробка цього сервісу включала наступні кроки:

Інтеграція з Telegram API. Для взаємодії з Telegram було використано Telegram Bot API, що дозволило створити бота та налаштувати його функціональність.

Детальні кроки налаштування інтеграції з Telegram API:

1. Відкрити спеціалізованого Telegram боту для створення власних ботів (рис. 3.13).

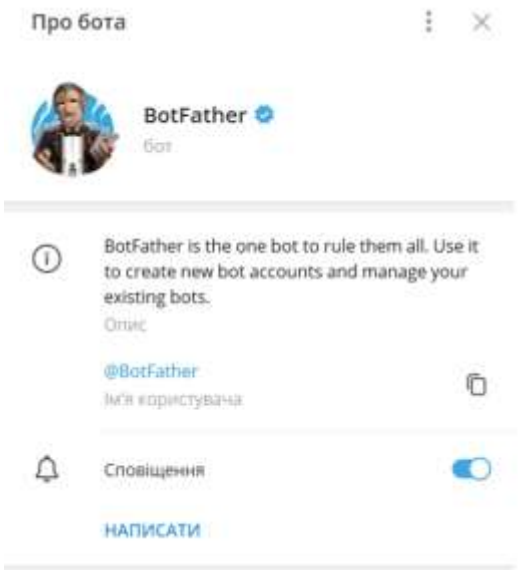


Рис. 3.13. Зображення інформації про бота

2. Натиснути на кнопку «start» для того, щоб розпочати роботу з чатом та отримати доступ до розширених можливостей (рис. 3.14).



Рис. 3.14. Зображення діалогового вікна

3. Наступним кроком є вибір функції /newbot, після чого потрібно було вказати локальну та глобальну назву бота. Як результат виконання попередніх дій, отримуємо усі необхідні дані для взаємодії бота із сервісом (рис. 3.15).

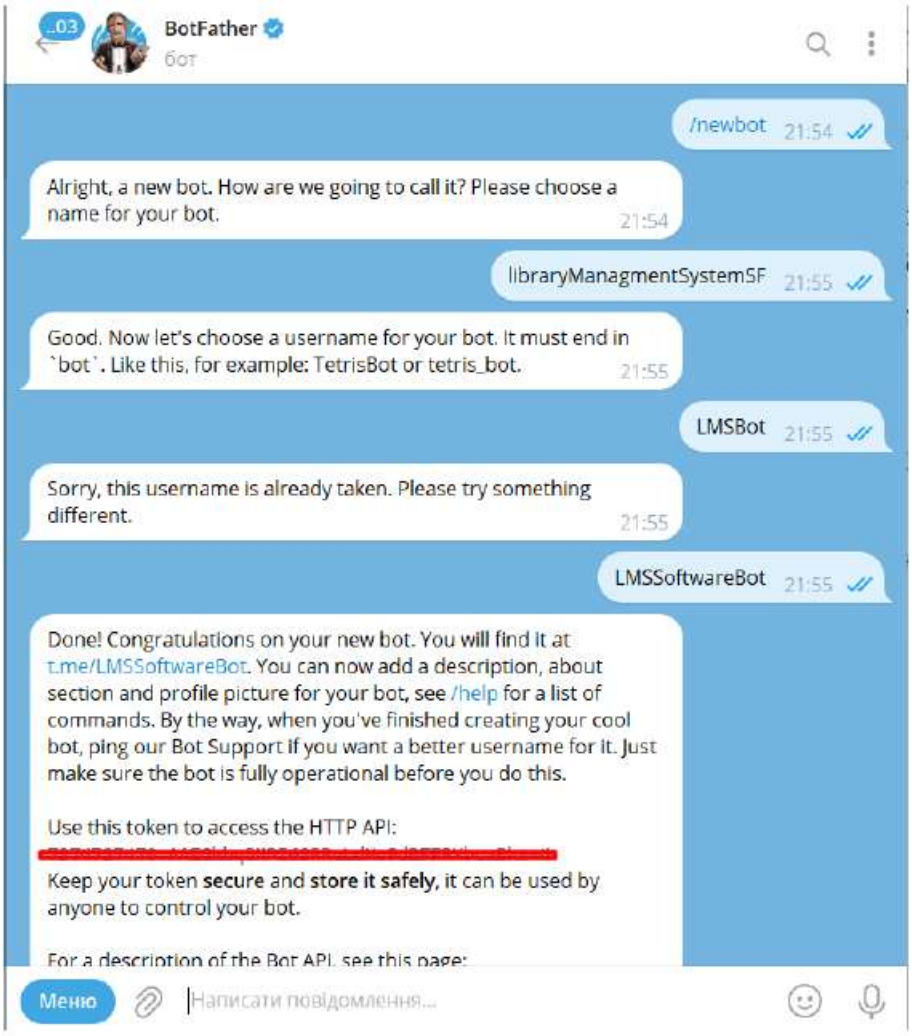


Рис. 3.15. Зображення повідомлення із генерацією коду доступу до створеного боту

Після налаштування боту для сповіщень на стороні Telegram, надається можливість реалізації власного бота і управління його логікою роботи, для цього необхідно створити відповідний клас, що реалізовуватиме базовий необхідний функціонал Telegram ботів.

Створення власного класу бота в системі управління бібліотекою дозволяє реалізувати необхідний функціонал та логіку роботи бота, адаптуючи його до специфічних потреб бібліотеки. Цей клас може містити такі методи:

Обробка команд:

•onStartCommand: обробляє команду /start, яка зазвичай використовується для ініціалізації бота та підписки користувача на сповіщення;

•onHelpCommand: обробляє команду /help, яка надає користувачеві інформацію про доступні команди та функції бота;

•onStopCommand: обробляє команду /stop, яка призупиняє роботу сповіщень для користувача.

Обробка повідомлень:

•onMessageReceived: обробляє текстові повідомлення від користувачів. Тут реалізований основний функціонал по формуванню відповіді користувачу;

•onCallbackQuery: обробляє натискання кнопок в інтерактивних повідомленнях.

Відправка повідомлень:

- sendMessage: відправляє текстове повідомлення користувачеві.

Наступними етапами розробки сервісу є налаштування підключення до Kafka серверу та налаштування отримання повідомлень.

Для отримання повідомлень від сервісу управління замовленнями було налаштовано підключення до Apache Kafka. Використано бібліотеку spring-kafka, яка надає зручні інструменти для роботи з Kafka в Spring Boot.

Для конфігурації споживача було створено клас конфігурації KafkaConsumerConfig, де визначено:

- адреси Kafka брокерів (bootstrap.servers);
- ідентифікатор групи споживачів (group.id);
- десеріалізатори для ключів та значень повідомлень (key.deserializer, value.deserializer).

Для створення споживача Kafka було створено клас споживача NotificationConsumer, який підписаний на тему Kafka, з якої надходять повідомлення від сервісу управління замовленнями. У методі listen споживача реалізовано логіку обробки повідомлень про затримки та рекомендації.

Таким чином, як результат налаштування взаємодії програми з сервісом управління замовленнями та інтеграцією його із Telegram для налаштувати боту сповіщень, було реалізовано сервіс, який відповідає поставленим вимогам по

роботі із користувачам задля сповіщення його стосовно замовлених книг та надсилання рекомендаційної літератури.

3.3. Розробка сервісу управління книгами

Сервіс управління книгами – це сервіс, що надає функціонал для управління літературою в системі, основне призначення його – управління життєвим циклом об’єкта інформації літератури у системі. Управління життєвим циклом полягає у: додаванні нової літератури, редагування та видалення існуючої та фільтрація.

Сервіс управління книгами в системі автоматизації бібліотеки реалізовано на основі Spring Boot, використовуючи MySQL для зберігання даних, MapStruct для мапінгу об'єктів, Spring Web для створення REST API, Lombok для спрощення коду, та Spring Data JPA для взаємодії з базою даних.

Основні моделі даних:

1. Book: Основна сутність, що представляє книгу в системі. Поля:
   - id (Long): унікальний ідентифікатор книги, генерується автоматично;
   - title (String): назва книги (обов'язкове поле);
   - author (String): автор книги (обов'язкове поле);
   - isbn (String): міжнародний стандартний номер книги (ISBN) (обов'язкове поле, унікальне);
   - category (String): категорія книги (обов'язкове поле);
   - description (String): опис книги (необов'язкове поле).
   - coverImage (String): посилання на обкладинку книги (необов'язкове поле).
2. BookRequest: DTO (Data Transfer Object) для передачі даних про книгу від клієнта до сервера при створенні або оновленні книги.
3. BookResponse: DTO для передачі даних про книгу від сервера до клієнта.

Для можливості взаємодії сервісу із зовнішніми елементами системи було розроблено BookController, що відповідає за обробку HTTP-запитів, пов'язаних з

книгами. Використовує @RestController та @RequestMapping("/books") для визначення базового шляху до ендпоінтів. Має такі методи:

- createBook: приймає BookRequest, створює нову книгу та зберігає її в базі даних;
- getAll: повертає список всіх книг у вигляді BookResponse;
- getById: повертає книгу за її ідентифікатором у вигляді BookResponse;
- getAllByCategory: повертає список книг певної категорії у вигляді BookResponse;
- deleteById: видаляє книгу за її ідентифікатором.
- createBook: створює нову книгу на основі BookRequest, перетворює її на Book та зберігає в базі даних;
- getAll: отримує всі книги з бази даних, перетворює їх на BookResponse та повертає список;
- getById: отримує книгу за ідентифікатором з бази даних, перетворює її на BookResponse та повертає;
- getAllByCategory: отримує всі книги певної категорії з бази даних, перетворює їх на BookResponse та повертає список;

Також було запроваджено клас BookService, що містить бізнес-логіку для роботи з книгами. Використовує BookRepository для взаємодії з базою даних. Використовує BookMapper для перетворення між сутностями Book та DTO BookRequest і BookResponse. Основні методи класу:

- createBook: створює нову книгу на основі BookRequest, перетворює її на Book та зберігає в базі даних;
- getAll: отримує всі книги з бази даних, перетворює їх на BookResponse та повертає список;
- getById: отримує книгу за ідентифікатором з бази даних, перетворює її на BookResponse та повертає;
- getAllByCategory: отримує всі книги певної категорії з бази даних, перетворює їх на BookResponse та повертає список;

- deleteById: видаляє книгу за ідентифікатором з бази даних.

Також було реалізовано ряд класів, що відповідають за збереження та форматування даних:

4. BookRepository – інтерфейс, що розширює JpaRepository та надає методи для роботи з книгами в базі даних. Spring Data JPA автоматично генерує реалізацію цього інтерфейсу на основі його методів.

5. BookMapper – інтерфейс, що визначає методи для перетворення між сутностями Book та DTO BookRequest і BookResponse.MapStruct автоматично генерує реалізацію цього інтерфейсу.

У результаті дослідження розробки сервісу, що відповідальний за управління книгами, було описано основні програмні компоненти програмного коду.

3.4. Тестування основного функціоналу

Для тестування системи було запроваджено ряд функціональних тестів, які дозволяють перевірити ефективність виконання програми.

Тест кейси, які відносяться до процесу валідації та управлінням даними книг на стороні контролера:

Тест-кейс №1. Створення книги з валідними даними.

Передумова: користувач системи авторизований із привілейованим доступом (адміністратор).

Кроки:

1. Відправити POST-запит на /books з валідними даними книги у форматі JSON (за допомогою форми додавання книг).
2. Перевірити, що отримано HTTP-статус 201 Created (відбувається на стороні програмної реалізації клієнтської частини).
3. Перетворити JSON-відповідь на об'єкт BookDto (відбувається на стороні програмної реалізації серверної частини).

4. Перевірити, чи повернутий об'єкт книги відповідає очікуваним даним (окрім автоматично згенерованого ID) (відбувається на стороні програмної реалізації серверної частини).

5. Видалити створену книгу з бази даних після завершення тесту (відбувається на стороні програмної реалізації серверної частини).

Очікуваний результат: створений об'єкт книги повертається у відповіді та зберігається в базі даних. HTTP-статус 201 Created.

Результат: отриманий результат збігається з очікуваним, створений об'єкт книги повертається у відповіді та об’єкт збережений в базі даних. Отриманий HTTP-статус 201 Created.

Тест-кейс №2. Отримання всіх книг.

Передумова: немає передумов.

Кроки:

1. Відправити GET-запит на /books (клієнтська частина автоматично відправляє запит при переході на сторінку списку літератур).

2. Перевірити, що отримано HTTP-статус 200 OK (відбувається на стороні програмної реалізації серверної частини).

3. Перетворити JSON-відповідь на масив об'єктів BookDto (відбувається на стороні програмної реалізації серверної частини).

4. Порівняти отриманий список книг з попередньо визначеним списком стандартних книг, перевіряючи їх кількість та вміст (відбувається на стороні програмної реалізації серверної частини).

Очікуваний результат: повний список книг повертається у відповіді. HTTP-статус 200 OK.

Результат: отриманий результат збігається з очікуваним, повний список книг повертається у відповіді. Отриманий HTTP-статус 200 OK.

Тест-кейс №3. Отримання книги за валідним ID.

Передумова: немає передумов.

Кроки:

1. Вибрати довільну книгу зі списку книг.

2. Відправити GET-запит на /books/{id}, де {id} - це ID вибраної книги (відбувається автоматично на стороні клієнтської частини).

3. Перевірити, що отримано HTTP-статус 200 OK (відбувається на стороні програмної реалізації серверної частини).

4. Перетворити JSON-відповідь на об'єкт BookDto (відбувається на стороні програмної реалізації серверної частини).

5. Перевірити, чи повернутий об'єкт книги відповідає очікуваним даним вибраної книги (відбувається на стороні програмної реалізації серверної частини).

Очікуваний результат: книга з заданим ID повертається у відповіді. HTTP-статус 200 OK.

Результат: отриманий результат збігається з очікуваним, книга з заданим ID повертається у відповіді. Отриманий HTTP-статус 200 OK.

Тест-кейс №4. Отримання книги за невалідним ID.

Передумова: немає передумов

Кроки:

1. Відправити GET-запит на /books/{id}, де {id} - це невалідний ID книги.

2. Перевірити, що отримано HTTP-статус 404 Not Found.

Очікуваний результат: Помилка 404 Not Found при запиті з невалідним ID.

Відправити DELETE-запит на /books/{id}, де {id} - це ID тестової книги (за допомогою панелі адміністратора).

Перевірити, що отримано HTTP-статус 204 No Content.

Відправити повторний GET-запит на /books, щоб отримати список всіх книг.

Перевірити, що тестова книга більше не присутня у списку.

Результат: отриманий результат збігається з очікуваним, отриманий HTTP-статус 404 Not Found.

Тест-кейс 5. Видалення книги за валідним ID.

Передумова: користувач системи авторизований із привілейованим доступом (адміністратор).

Кроки:

Очікуваний результат: Книга успішно видалена з системи. HTTP-статус 204 No Content.

Результат: отриманий результат збігається з очікуваним, книга успішно видалена. Отриманий HTTP-статус 204 No Content.

Також варто розглянути тест кейси, які полягали у тестуванні роботи системи з базою даних. Для тестування системи із базою даних було створено відповідний клас. Цей клас тестів перевіряє роботу BookRepository, який є ключовим компонентом для взаємодії з базою даних у системі управління бібліотекою. Він використовує фреймворк Spring Data JPA для спрощення операцій з базою даних.

Детальний опис тестів:

Тест-кейс №6. Пошук книг за категорією з валідним ID.

Передумова: Перед виконанням тесту, за допомогою анотації @Sql, в базу даних додаються тестові дані (книги та категорії).

Кроки:

1. Визначити ідентифікатор категорії, для якої потрібно знайти книги.
2. Викликати метод findAllByCategoriesId з цим ідентифікатором.

1. Перевірити, що кількість повернутих книг дорівнює кількості книг, пов'язаних з цією категорією в тестових даних.
2. Перевірити, чи перша книга в результаті має непустий набір категорій (це підтверджує, що зв'язок з категоріями встановлений коректно).

Після завершення тесту, тестові дані видаляються з бази даних. Зберегти тестову книгу в базі даних.

Викликати метод findById з ідентифікатором збереженої книги.

Перевірити, чи повернута книга не є null.

Перевірити, чи всі поля книги (крім id) збігаються з полями тестової книги.

Очікуваний результат: Метод findAllByCategoriesId повертає всі книги, що належать до вказаної категорії.

Результат: отриманий результат збігається з очікуваним, отримано всі книги, що належать вказаній категорії.

Тест-кейс №7. Пошук книги за валідним ID.

Передумова: Тестова книга зберігається в базі даних.

Кроки:

Очікуваний результат: Метод findById коректно повертає книгу за її ідентифікатором.

Результат: отриманий результат збігається з очікуваним, отримано книгу, що відповідає ідентифікаторові.

Тест-кейс №8. Пошук всіх книг.

Передумова: немає передумов.

Кроки:

1. Визначити очікувану кількість книг, що має бути повернена (в даному випадку, це кількість книг в тестових даних).

2. Створити об'єкт Pageable для пагінації результатів (встановлено розмір сторінки 10).

3. Викликати метод findAll з цим об'єктом Pageable.

Перевірити, чи кількість книг на першій сторінці результатів збігається з очікуваною кількістю. Налаштувати моки репозиторію та мапера, щоб вони повертали очікувані значення при виклику їх методів.

Викликати метод save сервісу з коректними вхідними даними.

Перевірити, що повернутий об'єкт BookDto збігається з очікуваним.

Очікуваний результат: Метод findAll коректно повертає всі книги, доступні в базі даних.

Результат: отриманий результат збігається з очікуваним, отримано всі книги, що наявні в базі даних під час тесту.

Ці тести демонструють найкращі практики тестування репозиторіїв Spring Data JPA, забезпечуючи перевірку ключових операцій з базою даних та гарантуючи коректну роботу системи управління бібліотекою.

Для забезпечення максимальної точності виконання програмного коду, було також запроваджені тести на рівні сервісів (частини коду, що відповідає за логіку роботи програмної системи).

Основним класом, що запроваджує тести на рівні сервісів є BookServiceTest. Цей клас використовує фреймворк Mockito для створення моків (імітацій) репозиторію (BookRepository) та мапера (BookMapper), що дозволяє ізолювати бізнес-логіку від деталей взаємодії з базою даних та перетворення об'єктів.

Тести зосереджені на перевірці методів сервісу BookService, таких як save, getAll, getById, update та deleteById.

Тест-кейс №9. Збереження книги з валідними даними.

Передумова: немає передумов.

Кроки:

Очікуваний результат: Метод save повертає об'єкт книги з коректними даними.

Результат: отриманий результат збігається з очікуваним, метод повернув книгу з очікуваними даними.

Тест-кейс №10. Отримання книг з пагінацією.

Передумова: немає передумов.

Кроки:

Налаштувати мок репозиторію, щоб він повертав сторінку з однією тестовою книгою.

Викликати метод getAll з об'єктом Pageable, що вказує на пагінацію.

Перевірити, що повернутий список книг містить лише один елемент - очікувану книгу.

Очікуваний результат: метод getAll повертає сторінку з книгами відповідно до заданої пагінації.

Результат: отриманий результат збігається з очікуваним, метод повернув сторінку з книгами відповідно до заданої пагінації.

Тест-кейс №11. Отримання книги за валідним ID.

Передумова: немає передумов.

Кроки:

Налаштувати мок репозиторію, щоб він повертав тестову книгу при запиті за її ідентифікатором.

Викликати метод getById з існуючим ідентифікатором.

Перевірити, що повернутий об'єкт книги збігається з очікуваним.

Очікуваний результат: метод getById повертає книгу за її ідентифікатором, якщо книга існує.

Очікуваний результат: метод getById повертає книгу за її ідентифікатором, якщо книга існує.

Результат: отриманий результат збігається з очікуваним, метод повернув книгу, що відповідає ідентифікаторові.

Тест-кейс №12. Отримання книги за невалідним ID.

Передумова: немає передумов.

Кроки:

Налаштувати мок репозиторію, щоб він повертав порожній Optional при запиті за неіснуючим ідентифікатором.

Викликати метод getById з неіснуючим ідентифікатором.

Перевірити, що було викинуто виняток EntityNotFoundException з очікуваним повідомленням про помилку.

Очікуваний результат: метод getById викидає виняток EntityNotFoundException, якщо книга з вказаним ідентифікатором не знайдена.

Результат: отриманий результат збігається з очікуваним, метод повернув очікувану помилку, через відсутність книги за ідентифікатором у базі даних.

Тест-кейс №13. Видалення книги за валідним ID.

Передумова: немає передумов.

Кроки:

Налаштувати мок репозиторію, щоб він успішно видаляв книгу при запиті за її ідентифікатором.

Викликати метод deleteById з існуючим ідентифікатором.

Перевірити, що метод завершився без винятків.

Очікуваний результат: метод deleteById успішно видаляє книгу з системи.

Результат: отриманий результат збігається з очікуваним, після виконання методу книга була видалена із бази даних.

Тест-кейс №14. Видалення книги за невалідним ID.

Передумова: немає передумов.

Кроки:

1. Налаштувати мок репозиторію, щоб він повертав порожній Optional при запиті на видалення неіснуючої книги.
2. Викликати метод deleteById з неіснуючим ідентифікатором.
3. Перевірити, що було викинуто виняток EntityNotFoundException з очікуваним повідомленням про помилку.

Очікуваний результат: метод deleteById викидає виняток EntityNotFoundException, якщо книга з вказаним ідентифікатором не знайдена.

Результат: отриманий результат збігається з очікуваним, метод повернув очікувану помилку, оскільки книга за ідентифікатором була відсутня у базі даних.

Важливі аспекти, які були застосовані під час тестування:

1. Ізоляція бізнес-логіки. Використання моків дозволяє зосередитися на тестуванні саме бізнес-логіки сервісу, не зачіпаючи роботу репозиторію та мапера;
2. Покриття різних сценаріїв. Тести охоплюють основні сценарії використання сервісу (створення, отримання, оновлення, видалення) та враховують можливі помилки (відсутність книги).

Отже, у ході роботи над підрозділом бакалаврської дипломної роботи було запроваджено та описано ряд тест-кейсів, що відповідають за тестування різних шарів програмної системи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно досягнуто мету дослідження — створено програмний додаток, призначений для автоматизації основних процесів управління роботою бібліотеки. Робота охопила весь цикл створення програмного забезпечення: від аналізу предметної області до розробки, моделювання структури, реалізації функціоналу, тестування та формування пропозицій щодо впровадження і розвитку системи.

Проведено всебічний аналіз існуючих засобів автоматизації бібліотечної справи, таких як Koha, FOLIO, Alma. Вивчення цих систем дозволило визначити їхні переваги, недоліки, а також найкращі практики, які були враховані при формуванні вимог до власного програмного продукту. Зокрема, акцент було зроблено на зручності інтерфейсу, гнучкості модульної структури, підтримці віддаленої взаємодії користувача з системою.

У ході роботи сформульовано функціональні та нефункціональні вимоги до системи, спроєктовано її архітектуру із поділом на серверну і клієнтську частини, а також розроблено структурну діаграму, яка відображає логіку взаємодії між основними компонентами. Серверна частина забезпечує обробку запитів, взаємодію з базою даних, а клієнтська — надає зручний користувацький інтерфейс для взаємодії з системою. Реалізовано повноцінний алгоритм роботи додатку — від входу користувача в систему до виконання таких дій, як пошук, замовлення, бронювання або повернення літератури.

Для розробки системи використано сучасні технології, зокрема мову програмування JavaScript, середовище Node.js для бекенду, фреймворки для фронтенду (наприклад, React або Vue), засоби REST API для обміну даними та базу даних для зберігання інформації про книги, читачів та дії. Також продемонстровано можливість інтеграції з Telegram-ботом для сповіщення користувачів про завершення терміну користування літературою або надходження новинок.

У результаті тестування системи підтверджено її стабільну роботу, коректну обробку типових запитів, доступність інтерфейсу для користувача з

різним рівнем цифрової грамотності. Отримані результати свідчать про ефективність реалізованого функціоналу: автоматизовано процес реєстрації користувачів, ведення електронного каталогу, бронювання, повернення книг, надсилання нагадувань.

Розроблений додаток є прикладом гнучкого рішення, яке може бути адаптовано до потреб як великих університетських бібліотек, так і невеликих шкільних чи спеціалізованих установ. Крім того, система легко масштабована, що дозволяє надалі розширювати її можливості — зокрема, додавати підтримку електронних книг, QR-кодування примірників, ведення статистики користування фондами, автоматичне формування звітності для адміністрації тощо.

Загалом результати дослідження підтверджують доцільність використання сучасних інформаційних технологій для підвищення ефективності бібліотечної роботи. Запропонований додаток не лише оптимізує діяльність персоналу, а й підвищує якість обслуговування користувачів, забезпечуючи зручний, швидкий та інтуїтивно зрозумілий доступ до бібліотечних ресурсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дятлюк І. С., Романюк О. В. Інтеграція сучасних технологій із програмним забезпеченням для ефективного управління ресурсами бібліотеки. Матеріали LIII науково-технічної конференції підрозділів ВНТУ, Вінниця, 20-22 березня 2024 р. Електрон. текст. дані. 2024. URI: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20691.
2. Дятлюк І. С., Романюк О.В. Впровадження гібридної системи рекомендацій із використанням штучного інтелекту. / Молодь в науці: дослідження, проблеми, перспективи (МН-2024), Вінниця 2024. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/21655/17880
3. Koha URL: https://koha-community.org/
4. Folio URL: https://folio.org/
5. Alma URL: https://almacam.com
6. The Principles of Designing Microservices URL: https://redis.io/blog/implementing-designing-microservices
7. Exploring Monolith Architecture: Pros, Cons, and Modern Alternatives URL: https://medium.com/@sonalisahi/exploring-monolith-architecture-pros-cons-and-modern-alternatives-1cefefdfce96
8. Microservice Architecture URL: https://microservices.io/
9. Building Applications with Serverless Architectures URL: https://aws.amazon.com/lambda/serverless-architectures-learn-more
10. Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor Recommender Systems Handbook – Нью-Йорк 2010.
11. Monolithic Approach vs. Microservices Approach: Which is Right for Your Application? URL: https://www.linkedin.com/pulse/monolithic-approach-vs-microservices-which-right-your-majid-sheikh.
12. Service Discovery URL: https://avinetworks.com/glossary/service-discovery/.

13. Шилдт Г., Ковард Д. Java: Повний довідник, Тринадцяте видання. Нью-Йорк: McGraw Hill Education, 2024.
14. C# URL: https://learn.microsoft.com/dotnet/
15. Python URL: https://www.python.org/about/
16. Eclipse URL: https://uk.wikipedia.org/wiki/Eclipse
17. What is Intellij IDEA - A Comprehensive Guide URL: https://www.simplilearn.com/what-is-intellij-idea-article
18. VS Code URL: https://uk.wikipedia.org/wiki/Visual_Studio_Code
19. The Importance of User Interface Design URL: https://www.plego.com/importance-user-interface-design/
20. Lemay L., Colburn R., Kyrnin J. HTML, CSS & JavaScript Web Publishing. Indianapolis: Sams Teach Yourself
21. The Importance of Testing And How to Do It Properly URL: https://www.linkedin.com/pulse/importance-testing-how-do-properly-peter-tylee-lahjc
22. Типи тестування: Types of software testing URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing
23. Островська Л.В. Автоматизація бібліотечно-інформаційних процесів. Львів: ЛНУ, 2020. 210 с.
24. Кузнєцова С.О. Управління бібліотекою в умовах цифровізації. Харків: ХНУ, 2021. 198 с.
25. Сіренко С.М. Основи програмування: створення прикладних додатків. К.: Либідь, 2020. 336 с.
26. Шевчук В.В. Розробка програмного забезпечення: технології та інструменти. К.: Академвидав, 2020. 416 с
27. Мартинюк С.І. Програмування інформаційних систем. Тернопіль: ТНТУ, 2018. 420 с.
28. Pressman R.S., Maxim B.R. Software Engineering: A Practitioner's Approach. 8th ed. McGraw-Hill, 2014.

29. Кравченко О.І. Автоматизація роботи бібліотеки засобами інформаційних технологій // Інформаційні технології і засоби навчання. 2021. №3(83). С. 23–30.
30. Бондаренко Ю.П. Розробка веб-додатків для автоматизації бібліотечного обслуговування // Науковий журнал «Інформаційні технології». 2020. №2. С. 48–54.
31. Чорнобай А.В. Управління бібліотечними фондами в інформаційній системі // Вісник ХНУ. Серія: Інформатика. 2019. №43. С. 67–72.
32. Бєляєва Т.М. Автоматизація бібліотечних процесів: стан та перспективи розвитку // Бібліотечний вісник. 2020. №4. С. 45–50.
33. Solovev V.D. Application of Information Technologies for Library Automation // International Journal of Advanced Studies. 2020. Vol. 10, No. 3. P. 101–110.
34. Ashoor S. Library Automation Systems: A Review of Development and Trends // Journal of Library Science. 2020. Vol. 6(2). P. 34–42.
35. Singh R. Library Management Systems: Evolution and Current Trends // International Journal of Library and Information Science. 2021. Vol. 13(1). P. 12–21.
36. Зайцева О.П. Веб-сервіси в бібліотечних інформаційних системах // Інформаційні технології і засоби навчання. 2022. №2(88). С. 29–35.
37. Welling L., Thomson L. PHP and MySQL Web Development. 5th ed. — Addison-Wesley, 2016.
38. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. Cengage Learning, 2016.
39. Martin R.C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.
40. Breeding M. Library Technology Reports: Library Systems Report 2021. American Library Association, 2021.
41. Kumbhar R. Library Automation: From Card Catalogue to Web-Based Catalogue. Chandos Publishing, 2011.