

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему СТВОРЕННЯ ПРОГРАМНОЇ СИСТЕМИ ОПРАЦЮВАННЯ
ДАНИХ ДЛЯ КЕРУВАННЯ БІБЛІОТЕЧНИМ ФОНДОМ
CREATING OF A DATA PROCESSING SYSTEM FOR
LIBRARY COLLECTION MANAGEMENT

Виконав(ла): студент(ка) 4 курсу, групи КНТ-212

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ГЛАЗУНОВ Н.М.

(ПРИЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2026 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГЛАЗУНОВА Нікіти Максимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Створення програмної системи опрацювання даних для керування бібліотечним фондом. Creation of a Data Processing System for Library Collection Management

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “07” квітня 2026 року № 139

2. Строк подання студентом проєкту (роботи) 10 червня 2026 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис системи опрацювання даних. 4. Експлуатація, тестування та експериментальне дослідження системи опрацювання даних.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури системи опрацювання даних.	2 тиждень	Розділ 3
5	Розробка системи опрацювання даних.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження системи опрацювання даних.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)


(підпис)

Нікіта ГЛАЗУНОВ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
93 с., 4 табл., 30 рис., 3 дод., 21 джерело.

PYTHON, PYCHARM, SQLITE, БАЗА ДАНИХ, БІБЛІОТЕКА, МОВА ПРОГРАМУВАННЯ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ.

Об'єкт дослідження – процеси обчислень, пов'язані з організацією та опрацюванням даних для керування бібліотечним фондом.

Предмет дослідження – системи опрацювання даних для керування бібліотечним фондом.

Мета роботи – розробка системи опрацювання даних для керування бібліотечним фондом для підвищення ефективності діяльності бібліотеки.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm, система управління базами даних SQLite.

Результати. Розроблено проєктні рішення для створення системи опрацювання даних для керування бібліотечним фондом. Створено систему опрацювання даних для керування бібліотечним фондом. Проведено дослідження розробленої системи опрацювання даних для керування бібліотечним фондом.

Висновки. Мету роботи досягнуто. Розроблено систему опрацювання даних для керування бібліотечним фондом за допомогою мови програмування Python, середовища розробки PyCharm та системи управління базами даних SQLite.

Галузь використання – бібліотечні організації.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 93 pages, 4 tables, 30 figures, 3 appendixes, 21 sources.

PYTHON, PYCHARM, SQLITE, DATABASE, LIBRARY, PROGRAMMING LANGUAGE, DATABASE MANAGEMENT SYSTEM.

The object of research is computing processes associated with the organization and processing of data for managing the library collection.

The subject of the research is data processing systems for managing the library collection.

The purpose of this work is to develop the data processing system for managing the library collection to increase the efficiency of the library.

Materials, methods and technical tools: Python programming language, PyCharm development environment, SQLite database management system.

Results. Project solutions for the creation of data processing system for managing the library collection has been designed. The data processing system for managing the library collection has been developed. The study of the developed data processing system for managing the library collection has been conducted.

Conclusions. The goal of the work has been achieved. The data processing system for managing the library collection using the Python programming language, the PyCharm development environment, and the SQLite database management system.

Scope of use – library organizations.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Системи опрацювання даних для керування бібліотечним фондом	11
1.2 Аналіз систем опрацювання даних для керування бібліотечним фондом.....	19
1.3 Висновки за розділом 1	29
2 Матеріали і методи	31
2.1 Вибір мови програмування	31
2.2 Вибір середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом	33
2.3 Вибір засобів зберігання та опрацювання даних.....	36
2.4 Висновки за розділом 2	38
3 Опис системи опрацювання даних	40
3.1 Структура системи опрацювання даних для керування бібліотечним фондом	40
3.2 Функціонування системи опрацювання даних для керування бібліотечним фондом	44
3.3 Розробка бази даних системи опрацювання даних для керування бібліотечним фондом	46
3.4 Проєктування інтерфейсу системи опрацювання даних для керування бібліотечним фондом	48
3.5 Висновки за розділом 3	50
4 Експлуатація, тестування та експериментальне дослідження системи опрацювання даних	53
4.1 Призначення й умови застосування системи опрацювання даних.....	53
4.2 Характеристики системи опрацювання даних для керування	

бібліотечним фондом	55
4.3 Інструкція по експлуатації програми.....	56
4.3.1 Звернення до програми	56
4.3.2 Вхідні й вихідні дані	56
4.3.3 Повідомлення.....	57
4.4 Виконання системи опрацювання даних для керування бібліотечним фондом	57
4.5 Тестування системи опрацювання даних для керування бібліотечним фондом	60
4.6 Висновки за розділом 4	61
Висновки	62
Перелік джерел посилання	65
Додаток А Технічне завдання	67
Додаток Б Фрагмент тексту програми	72
Додаток В Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

SLiMS – Senayan Library Management System;

БД – база даних;

БФ – бібліотечний фонд;

ІС – інформаційна система;

СОД – система опрацювання даних;

СУБД – система управління базами даних.

ВСТУП

Актуальність створення системи опрацювання даних для керування бібліотечним фондом полягає у зростаючих вимогах до ефективного управління інформаційними ресурсами в умовах цифрової трансформації. Традиційні, переважно паперові або частково автоматизовані методи обліку вже не забезпечують необхідної швидкості доступу до даних, точності пошуку та можливості оперативного оновлення інформації про фонди [1]-[4].

Сучасні бібліотеки стикаються з постійним збільшенням обсягів інформації, різноманітністю носіїв та форм зберігання, а також зростанням очікувань користувачів щодо швидкого знаходження потрібних матеріалів. Окрім цього, важливою стає інтеграція електронних ресурсів, підтримка віддаленого доступу та можливість аналітичного контролю за станом фонду [1].

Впровадження програмної системи для автоматизації управління бібліотечним фондом забезпечує підвищення рівня організації бібліотечних процесів, мінімізує ризик помилок під час ведення обліку, оптимізує роботу персоналу й економить час користувачів. Така система створює основу для комплексної модернізації бібліотек, спрощує процес інвентаризації, сприяє більш раціональному використанню фонду та дає змогу здійснювати оперативне планування його розвитку [2]-[3].

Таким чином, розробка системи опрацювання даних для керування бібліотечним фондом є актуальною через необхідність підвищення ефективності управління інформаційними ресурсами, забезпечення доступності знань та підтримку високої якості обслуговування користувачів у сучасному інформаційному середовищі [3]-[4].

Проте деякі системи опрацювання даних для керування бібліотечним фондом для створення та впровадження потребують значних матеріальних і часових ресурсів. Ліцензування програмного забезпечення, придбання обладнання, навчання персоналу та перенесення даних у цифрову форму можуть стати серйозним фінансовим навантаженням для бібліотечні установи.

Крім того, системи опрацювання даних для керування бібліотечним фондом вимагають постійного оновлення та технічної підтримки. Без регулярного супроводу вони поступово втрачають ефективність, виникають проблеми з безпекою та сумісністю з новими стандартами. Також можливими є труднощі під час адаптації персоналу, особливо якщо працівники не мають достатнього досвіду роботи з цифровими технологіями [1]-[5].

Тому актуальною є розробка системи опрацювання даних для керування бібліотечним фондом. У дипломній кваліфікаційній роботі бакалавра розв'язується актуальне завдання розробки системи опрацювання даних для керування бібліотечним фондом для підвищення ефективності діяльності бібліотеки.

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та систем опрацювання даних для керування бібліотечним фондом;
- здійснити проектування системи опрацювання даних для керування бібліотечним фондом;
- створити систему опрацювання даних для керування бібліотечним фондом;
- дослідити розроблену систему опрацювання даних для керування бібліотечним фондом.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи опрацювання даних для керування бібліотечним фондом

Системи опрацювання даних для керування бібліотечним фондом являють собою інтегровані програмні рішення, призначені для впорядкування роботи з інформаційними ресурсами, що зберігаються у бібліотеках та споріднених установах. Їхнім основним призначенням є забезпечення безперервного контролю за рухом фонду, підтримання точності бібліографічних описів, підвищення швидкості пошуку матеріалів, а також оптимізація взаємодії між працівниками та користувачами бібліотеки. Завдяки таким системам оновлення записів, формування електронних каталогів, облік примірників, реєстрація читачів і супровід видачі матеріалів здійснюються централізовано і значно ефективніше, ніж за традиційних методів [1].

До характерних особливостей таких систем належить здатність об'єднувати дані різних типів у єдиному середовищі, підтримувати стандартизовані схеми опису, використовувати механізми пошуку за багатьма параметрами, а також забезпечувати доступ через браузер або спеціалізовані інтерфейси. Значна увага приділяється підтримці взаємодії з зовнішніми сервісами, зокрема репозитаріями, системами аутентифікації та інструментами резервного копіювання. У різних системах акцент може робитися на обслуговуванні великих електронних архівів, на роботі зі шкільними чи корпоративними фондами або на підтримці високого рівня автоматизації процесів у публічних бібліотеках [1].

Типологія таких рішень охоплює прості локальні інструменти для невеликих установ, масштабовані веборієнтовані платформи для середовищ із широким колом користувачів, а також комплексні архітектури, здатні підтримувати репозитарні функції, керувати цифровими документами та забезпечувати організацію довготривалого зберігання. Частина систем орієнтована на максимальну автономність, тоді як інші будуються навколо

модульності й розширюваності, дозволяючи поєднувати різні компоненти залежно від конкретних потреб [1].

Переваги таких рішень проявляються у підвищенні точності обліку, зменшенні навантаження на персонал, прискоренні обслуговування читачів і забезпеченні постійного доступу до каталогів. Електронне опрацювання інформації зменшує кількість помилок, полегшує оновлення записів та відкриває можливість пошуку за складними характеристиками. Додатковою перевагою є можливість інтеграції з інструментами аналітики, що дає змогу відстежувати динаміку використання фонду та планувати його розвиток [1].

Недоліки систем опрацювання даних для керування бібліотечним фондом пов'язані із залежністю від технічної інфраструктури, потребою у стабільному доступі до мережі, складністю міграції даних між різними системами, а також необхідністю постійного оновлення програмного забезпечення. Іноколи виникають обмеження у роботі з певними форматами, або ж інтерфейси потребують додаткового доопрацювання, щоб відповідати потребам конкретних установ. У випадках, коли система є надто потужною для невеликої бібліотеки, її розгортання може виявитися надмірно ресурсомістким [1]-[2].

Проблеми використання систем опрацювання даних для керування бібліотечним фондом найчастіше пов'язані з недостатньою підготовкою персоналу, складністю адаптації працівників до нових інструментів, недостатнім рівнем фінансування або браком кваліфікованої технічної підтримки. Можуть виникати труднощі з оновленням структури даних, підтриманням сумісності з іншими сервісами чи забезпеченням належного рівня захисту інформації [1]-[2].

Шляхи розвитку систем опрацювання даних для керування бібліотечним фондом пов'язуються з поступовим переходом до хмарних рішень, розширенням можливостей автоматичного аналізу даних, удосконаленням інтерфейсів та збільшенням рівня інтеграції з різними цифровими платформами. Очікується поширення інструментів на основі штучного

інтелекту, які зможуть автоматично доповнювати метадані, упорядковувати великі масиви цифрових документів та підвищувати якість пошуку. Розвиток зосереджуватиметься на створенні гнучких екосистем, здатних адаптуватися до різних типів бібліотек і забезпечувати зручний доступ до інформаційних ресурсів незалежно від формату зберігання чи масштабу установи [1]-[2].

Поступовий розвиток бібліотек від традиційних книгозбірень до багатофункціональних інформаційних центрів зумовив потребу у програмному забезпеченні, здатному підтримувати широкий спектр операцій. За сучасних умов бібліотека виконує роль комунікаційного та освітнього середовища, а тому потребує інструментів, що дозволяють опрацьовувати різномірні ресурси, забезпечувати швидке оновлення записів та підтримувати доступ до цифрових колекцій. Еволюція таких систем від простих локальних рішень до складних вебзастосунків та хмарних платформ відображає зміну завдань, що стоять перед бібліотечною сферою [1]-[2].

Необхідність спеціалізованих систем опрацювання даних для керування бібліотечним фондом пояснюється тим, що ручні методи роботи вже неспроможні забезпечити оперативне керування великими інформаційними фондами. Автоматизоване опрацювання даних дає змогу уникати помилок, оптимізувати обіг матеріалів, контролювати стан фонду та підтримувати зручний доступ до колекції через пошукові інструменти. У таких системах передбачені механізми, які дозволяють поновлювати записи, управляти читачами, реєструвати нові примірники, формувати електронні каталоги і надавати читачам можливість переглядати доступні матеріали через окремий інтерфейс [1]-[2].

Зміст системи опрацювання даних для керування бібліотечним фондом зазвичай охоплює блоки, що відповідають за створення структурованих описів документів, організацію видачі, облік надходжень, опрацювання періодики, керування користувачами та формування звітів. У межах таких рішень формуються дані про фонд, забезпечується відстеження руху ресурсів, фіксуються операції з матеріалами та надається можливість аналізувати

активність читачів чи якість укомплектування колекції. Для користувачів створюється простір, у якому можна знайти потрібну інформацію, переглянути електронні описи або залишити запит на отримання документів [1]-[2].

Вибір системи опрацювання даних для керування бібліотечним фондом залежить від потреб установи, її масштабів, особливостей фонду й доступних ресурсів. Кожне рішення має власний підхід до інтерфейсу, архітектури та функціональних можливостей, тому оцінюється здатність програмного забезпечення адаптуватися до конкретних умов, підтримувати необхідні формати опису, взаємодіяти з іншими цифровими сервісами і бути зручним для персоналу та відвідувачів. Важливо, щоб система могла працювати стабільно, ефективно та без надмірних витрат на обслуговування [2]-[3].

Сучасні тенденції розвитку платформ опрацювання даних для керування бібліотечним фондом охоплюють розширення хмарних технологій, удосконалення мобільного доступу, інтеграцію з навчальними, архівними й інформаційними системами, а також використання інтелектуальних інструментів для оброблення даних. Поширюється застосування технологій, здатних аналізувати поведінку користувачів, автоматично формувати метадані та покращувати пошукові механізми. Зростає потреба у більш гнучких інтерфейсах, які враховують різний рівень підготовки користувачів і дозволяють працювати з бібліотечним фондом без зайвих перешкод [2]-[3].

Бібліотечні установи постійно змінюються та пристосовуються до нових запитів суспільства. Поступове впровадження цифрових технологій суттєво вплинуло на організацію роботи з фондами, а системи опрацювання даних для керування бібліотечними ресурсами стали важливим елементом цієї трансформації. Поширення рішень, що базуються на відкритому доступі до програмного коду, надало бібліотекам можливість самостійно контролювати та вдосконалювати інструменти, за допомогою яких здійснюється управління інформаційними матеріалами. Завдяки таким підходам змінився характер автоматизації, оскільки стали доступними гнучкі й прозорі механізми оброблення бібліотечного фонду [2]-[3].

Гнучкість у налаштуванні є ще одною причиною поширення відкритих систем опрацювання даних для керування бібліотечним фондом. Їхня структура дозволяє модифікувати робочі процеси, адаптувати функціональні можливості до внутрішньої організації закладу та формувати цифрову інфраструктуру, що відповідає специфічним вимогам. Завдяки цьому програмні модулі опрацювання даних для керування бібліотечним фондом можуть працювати у точній відповідності до локальних правил і процедур, що підвищує ефективність опрацювання даних [3]-[4].

Суттєве значення має і співпраця між фахівцями, які беруть участь у створенні й підтримці рішень для опрацювання даних для керування бібліотечним фондом. Спільноти, що об'єднують бібліотекарів та розробників, здійснюють постійне оновлення інструментів, пропонують нові функціональні можливості, усувають недоліки й поширюють досвід використання. Такий підхід пришвидшує розвиток бібліотечних систем та сприяє стабільності їх роботи [3]-[4].

Прозорість роботи інструментів опрацювання даних для керування бібліотечним фондом на основі відкритого коду створює підґрунтя для довіри, оскільки кожен компонент можна перевірити та оцінити з погляду безпеки та відповідності потребам установи. За необхідності можлива незалежна оцінка вразливостей, а процес їх усунення відбувається значно швидше завдяки активній взаємодії учасників спільноти [3]-[4].

Особливу увагу привертає здатність таких систем взаємодіяти з іншими цифровими платформами. Підтримка відкритих стандартів дає змогу поєднувати інструменти керування фондами з зовнішніми інформаційними ресурсами, засобами автентифікації чи електронними базами, забезпечуючи безперервний доступ до різноманітного контенту. Завдяки цьому користувачі отримують цілісне цифрове середовище, у якому пошук, перегляд і використання матеріалів здійснюються без технічних бар'єрів [3]-[4].

У результаті системи опрацювання даних для керування бібліотечним фондом на основі відкритих рішень відіграють ключову роль у модернізації

бібліотечної діяльності. Вони поєднують економічну доцільність, широкі можливості змінювання, прозорість і ефективну взаємодію з іншими інформаційними платформами, формуючи сучасний підхід до організації роботи з бібліотечними ресурсами [3]-[4].

Функціонування систем опрацювання даних для керування бібліотечним фондом спрямовується на забезпечення швидкого доступу до фондів, оптимізацію пошуку матеріалів та підтримку безперебійної взаємодії між користувачами й ресурсами. Значна увага приділяється можливості розширення їхніх можливостей у міру збільшення обсягу інформації, що дає змогу адаптуватися до складніших завдань без потреби у радикальному оновленні інфраструктури. Завдяки належній архітектурі такі системи здатні ефективно працювати навіть за умов стрімкого зростання фондів [3]-[4].

Особливий акцент робиться на гнучкості систем опрацювання даних для керування бібліотечним фондом, що дозволяє бібліотекам підтримувати власні робочі процеси, не обмежуючись фіксованими механізмами. Такий підхід сприяє формуванню більш керованого та передбачуваного середовища роботи з даними. У межах практичного використання системи можуть оптимізуватися для швидкого виконання запитів, що забезпечує зручність для читачів і персоналу, а також дозволяє скорочувати час на опрацювання інформації [4]-[5].

Розвиток систем опрацювання даних для керування бібліотечним фондом стимулює перехід до моделі, де бібліотека отримує більше самостійності в питаннях технічного супроводу та контролю над власними фондами. Використання рішень, які передбачають відкритість архітектури, сприяє посиленню уваги до захисту даних і надійності, що є важливими аспектами для установ, які працюють з інформацією різного характеру. Забезпечується можливість повного контролю над тим, як зберігаються, опрацьовуються й захищаються записи, пов'язані з фондами та користувачами [4]-[5].

За умов постійних змін у сфері інформаційного середовища системи опрацювання даних для керування бібліотечним фондом відкривають можливість до впровадження нових рішень без суттєвих обмежень. Розвиток технологічної бази відбувається поступово, що дозволяє покращувати якість обслуговування та підтримувати актуальність бібліотечних послуг. Робота з такими системами також сприяє підвищенню кваліфікації персоналу, оскільки формуються навички, пов'язані з адмініструванням, налаштуванням і підтримкою програмних інструментів [4]-[5].

Сучасні рішення у цій сфері приділяють особливу увагу інтелектуальним можливостям, захищеному зберіганню інформації та використанню технологій, що підвищують точність і швидкість опрацювання фондів. Використання інструментів пошуку з інтелектуальними механізмами дозволяє системі швидко встановлювати відповідність між запитом та відповідними записами, що робить взаємодію з інформаційним масивом більш інтуїтивною. Механізми доступу адаптовані таким чином, щоб користувачеві було зручно працювати як у межах бібліотеки, так і поза нею [5]-[6].

У межах таких систем відбувається також упорядкування даних про користувачів і їх взаємодію з фондом. Це забезпечує точність роботи з читачькими записами та створює можливість персоналізації сервісів. Завдяки цьому вдається підтримувати зручні умови для взаємодії між читачем та інформаційними ресурсами, а також формувати рекомендації відповідно до інтересів користувачів [6]-[7].

Важливим напрямом розвитку таких систем є розширення доступу до ресурсів через мобільні пристрої та перехід до моделі, де робота з даними здійснюється без прив'язки до конкретного робочого місця. Залучення технологій дистанційного зберігання створює умови для безперервної роботи навіть за відсутності локальної інфраструктури. Такий підхід також зменшує навантаження на технічні служби, оскільки значна частина технічного обслуговування переноситься на рівень платформи [6]-[7].

Особлива увага приділяється також надійності та конфіденційності. Системи створюються з урахуванням потреби у збереженні приватної інформації, а також контролюють доступ до окремих функцій, щоб забезпечити захищеність даних. Завдяки цим механізмам відомості про користувачів і фонди залишаються недоступними для сторонніх [6]-[7].

Подальший розвиток таких рішень орієнтується на вдосконалення зручності взаємодії, розширення аналітичних можливостей і адаптацію під змінні умови роботи. У результаті системи опрацювання даних для керування бібліотечними ресурсами перетворюються на багатофункціональні інструменти, які підтримують динамічне середовище та забезпечують ефективне використання інформаційних ресурсів у сучасному бібліотечному просторі [6]-[7].

У цілому системи опрацювання даних для керування бібліотечним фондом формують основу для створення бібліотечної інфраструктури, здатної ефективно реагувати на зміни, підтримувати стабільність роботи та забезпечувати користувачам та читачам якісний доступ до інформаційних ресурсів [6]-[7].

Визначено, що системи опрацювання даних для керування бібліотечним фондом являють собою інтегровані програмні рішення, призначені для впорядкування роботи з інформаційними ресурсами, що зберігаються у бібліотеках та споріднених установах. Їхнім основним призначенням є забезпечення безперервного контролю за рухом фонду, підтримання точності бібліографічних описів, підвищення швидкості пошуку матеріалів, а також оптимізація взаємодії між працівниками та користувачами бібліотеки. Завдяки таким системам оновлення записів, формування електронних каталогів, облік примірників, реєстрація читачів і супровід видачі матеріалів здійснюються централізовано і значно ефективніше, ніж за традиційних методів.

1.2 Аналіз систем опрацювання даних для керування бібліотечним фондом

Проаналізуємо системи опрацювання даних для керування бібліотечним фондом [8]-[15].

Програмна система SLiMS (Senayan Library Management System) (рис. 1.1) є програмою управління навчанням, що являє собою вебзастосунок. Програмне забезпечення Canvas LMS подається як універсальне середовище для організації бібліотечного процесу, у якому поєднуються засоби для опису фонду, ведення каталогів та доступу до електронних матеріалів. У її основі лежить підхід, що забезпечує повний цикл опрацювання документів – від внесення описів до надання читачам інструментів для пошуку й перегляду інформації. Усі операції здійснюються через інтерфейс, який автоматично підлаштовується під різні пристрої та підтримує роботу з графічними елементами [8], [13].

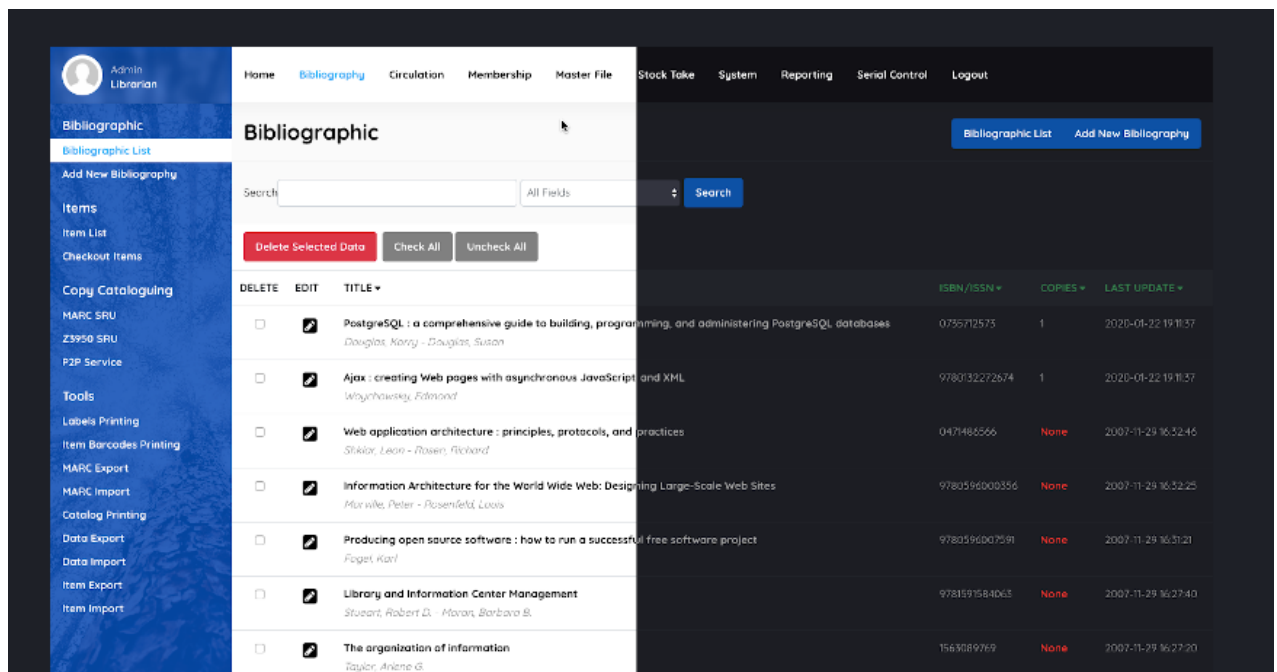


Рисунок 1.1 – Програмна система SLiMS [13]

У середовищі SLiMS реалізовано механізм публічного доступу, що дає можливість переглядати бібліотечні матеріали через вебінтерфейс, який уміє працювати з мініатюрами обкладинок та різними режимами пошуку. Для електронних ресурсів пропонується можливість прикріплення файлів у найрізноманітніших форматах, а структуру записів можна представляти у поширених моделях метаданих, придатних для обміну з іншими платформами. Підтримується формування потоків даних та надання метаданих у стандартизованих форматах, що спрощує інтеграцію з архівними та бібліотечними системами [8], [13].

Організація каталогів у SLiMS забезпечується засобами, які дозволяють описувати, групувати та структурувати бібліографічні записи, зокрема з можливістю формування збірних каталогів. Окремі модулі відповідають за роботу із серійними виданнями, а також за опрацювання окремих примірників, де використовується штрих-кодування. Доступна система керування довідковими таблицями, що містять інформацію про типи документів, класифікаційні позначення, видавництва, місця зберігання та інші службові дані [8], [13].

Функціональні можливості SLiMS охоплюють організацію видачі, контроль повернень, резервування матеріалів та застосування гнучких правил обслуговування читачів. Передбачено модуль для ведення обліку користувачів та оновлення їхніх даних. Окремий інструментарій присвячений інвентаризації фонду, що забезпечує контроль наявності всіх документів. У системі формуються статистичні показники та аналітичні звіти, які допомагають оцінювати стан фонду та роботу бібліотеки [8], [13].

У підтримуваному середовищі доступні засоби для налаштування функціональних модулів, визначення параметрів роботи системи, керування правами користувачів і груп, а також утиліти для формування штрих-кодів та створення копій бази даних. Пошук у каталогах може підсилюватися зовнішніми механізмами індексації, що підвищує швидкість та точність обробки запитів. Усе це робить SLiMS гнучким інструментом, придатним для

використання на різних операційних системах і в різних умовах роботи бібліотечних установ [8], [13].

Основними характеристиками програмної системи SLiMS є такі [8], [13]:

- онлайн доступ до каталогу: система SLiMS забезпечує можливість публічного перегляду бібліотечних матеріалів через вебінтерфейс із підтримкою мініатюр обкладинок та різних режимів пошуку [8], [13];

- підтримка цифрових файлів: у кожному бібліографічному записі SLiMS можуть зберігатися вкладення у різних форматах, що дає змогу поєднувати традиційні та електронні ресурси [8], [13];

- розширені метадані: структура записів SLiMS подається у поширених стандартах опису, що полегшує обмін інформацією з іншими системами та сервісами [8], [13];

- інструменти для збору метаданих: у SLiMS підтримуються механізми формування та передавання структурованих даних, які застосовуються у бібліотечних та архівних середовищах [8], [13];

- управління каталогом: у SLiMS передбачено повноцінний набір функцій для формування, редагування та впорядкування бібліографічних записів із використанням графічних елементів [8], [13];

- облік примірників: опрацювання копій документів здійснюється з використанням ідентифікації за штрих-кодами та реєстрацією їхнього місцезнаходження [8], [13];

- управління довідковими даними: у системі SLiMS підтримуються службові таблиці, що містять інформацію про типи документів, видавництва, авторів, класифікаційні ознаки та інші атрибути [8], [13];

- модуль видачі: у SLiMS реалізовано механізми позичання, повернення, резервування документів та застосування гнучких правил обслуговування читачів [8], [13];

- керування користувачами: у SLiMS зберігаються дані про читачів та здійснюється контроль їхніх прав доступу до функцій системи [8], [13];

- інвентаризаційні засоби: у SLiMS передбачено інструменти для перевірки наявності фонду, що полегшує проведення інвентаризаційних процедур [8], [13];
- статистика та звіти: система SLiMS формує аналітичні дані та підсумкові звіти щодо роботи бібліотеки й стану фонду [8], [13];
- системні налаштування: у SLiMS реалізовано керування параметрами роботи програмного середовища та конфігурацією окремих модулів [8], [13];
- утиліти обслуговування: у програмній системі SLiMS доступні засоби для генерації штрих-кодів, створення резервних копій та виконання системних операцій [8], [13];
- адаптивний інтерфейс: інтерфейс програмної системи SLiMS автоматично підлаштовується під різні пристрої, забезпечуючи зручний доступ до функцій системи [8], [13];
- зовнішня індексація: у програмній системі SLiMS підтримується підключення сторонніх систем індексації для прискорення та покращення пошуку бібліографічних записів [8], [13].

Серед недоліків програмної системи SLiMS можна відзначити певні обмеження у гнучкості налаштувань, через які адаптація під специфічні потреби окремих бібліотек може вимагати додаткових доопрацювань. Іноді спостерігається залежність від сторонніх компонентів, що може ускладнювати розгортання або підтримку на різних серверних середовищах. Деякі інструменти адміністрування виглядають перевантаженими, через що опанування системи працівниками без досвіду роботи з автоматизованими бібліотечними платформами потребує більше часу. У певних випадках можливе зниження продуктивності при роботі з великими масивами даних, що проявляється у сповільненні пошуку та обробки запитів. У питаннях масштабування інколи простежуються межі, пов'язані з архітектурою системи, через що розширення функціональних можливостей або інтеграція з зовнішніми сервісами потребує глибшого технічного втручання [8], [13].

Програма Invenio (рис. 1.2) являє собою програмну платформу з відкритим кодом, створену для організації та підтримки великих електронних зібрань, у яких зберігаються різноманітні інформаційні ресурси. Програмна система Invenio орієнтована на комплексне керування бібліотечними процесами, що охоплюють роботу з електронними фондами, реєстрацію та опис документів, а також процедури взаємодії між бібліотеками. У її основі закладено підхід, що дозволяє поєднувати стандартні бібліотечні операції з можливістю розширення функцій для побудови повноцінних цифрових інфраструктур [9], [14].



Рисунок 1.2 – Програмна система Invenio [14]

Архітектура Invenio включає засоби для створення власних програмних рішень, пов'язаних із пошуком інформації, адмініструванням цифрових сховищ та організацією доступу до даних. Платформа має механізми перевірки користувачів та підтримує різні моделі зберігання, що дозволяє гнучко формувати структуру репозиторію та забезпечувати контроль над доступом до матеріалів [9], [14].

Система Invenio вирізняється тим, що поєднує бібліотечну функціональність із можливостями інструментарію для створення великих інформаційних сервісів. Засоби бек-офісу уможливають зручне виконання завдань, пов'язаних із каталогізацією, обігом ресурсів та придбанням нових матеріалів, а інтерфейс та API дають змогу інтегрувати рішення у складні цифрові середовища [9], [14].

Перевагою Invenio є її відкритість і здатність адаптуватися до широкого кола потреб, а також підтримка різнорівневого керування файлами та метаданими. Разом із тим у деяких випадках інтерфейс може потребувати подальшого вдосконалення для досягнення більшої зручності роботи, особливо коли йдеться про користувачів без спеціальної підготовки [9], [14].

Програмна система Invenio має такі особливості [9], [14]:

- гнучка архітектура: система Invenio побудована так, щоб її можна було змінювати та розширювати відповідно до потреб окремої установи, забезпечуючи адаптивність цифрових репозиторіїв [9], [14];
- підтримка великих сховищ: платформа Invenio придатна для опрацювання масштабних інформаційних фондів і стабільно функціонує при роботі з великими обсягами даних [9], [14];
- розвинені механізми пошуку: в Invenio забезпечується можливість створення ефективних пошукових рішень, включаючи корпоративні системи пошуку та індексацію інформаційних ресурсів [9], [14];
- робота з метаданими: у Invenio реалізовано широкий спектр інструментів для опрацювання, зберігання та стандартизованого подання метаданих у цифрових репозиторіях [9], [14];
- потужна система API: у Invenio наявні інструменти для інтеграції з іншими програмними рішеннями, що дозволяє розширювати функціональність та будувати нові сервіси на базі платформи [9], [14];
- сучасний бек-офіс: адміністративна частина системи Invenio підтримує повний цикл роботи з бібліотечними матеріалами, включаючи каталогізацію, обіг та організацію придбання [9], [14];

- модуль автентифікації: у Invenio реалізовано гнучкі засоби контролю доступу, що дають змогу налаштовувати правила взаємодії користувачів із ресурсами [9], [14];
- підтримка зовнішніх систем зберігання: інструменти Invenio дозволяють застосовувати різні моделі зберігання даних, що полегшує організацію цифрових архівів та репозиторіїв [9], [14];
- можливість створення прикладних рішень: платформа Invenio надає засоби для проєктування сервісів, пов'язаних із керуванням інформаційними потоками, пошуком та зберіганням ресурсів [9], [14];
- відкритий код: доступність програмного коду Invenio спрощує доопрацювання системи та стимулює її розвиток спільнотою [9], [14].

Серед недоліків програмної системи Invenio можна відзначити певну складність початкового налаштування, оскільки її модульна структура вимагає ретельної конфігурації та розуміння внутрішніх механізмів. Інтерфейс керування також не завжди сприймається інтуїтивно через велику кількість інструментів і налаштувань, що може ускладнити ознайомлення нових користувачів із системою. Додаткові труднощі виникають під час розгортання на серверах зі специфічними вимогами до сумісності, адже деякі компоненти потребують поглиблених технічних знань. Також система іноді демонструє підвищені вимоги до ресурсів у випадках опрацювання значних масивів даних, що може сповільнювати роботу за відсутності належної оптимізації [9], [14].

Програмна система Colibris (рис. 1.3) являє собою хмарну платформу, створену для централізованого керування бібліотечними ресурсами та полегшення доступу до різноманітних інформаційних зібрань. Її функціонування спрямоване на те, щоб користувач міг безперешкодно здійснювати пошук у фондах, що містять книжкові видання, періодику, аудіовізуальні матеріали та інші носії. Система дає змогу швидко отримувати відомості про величезну кількість позицій, що можливо завдяки застосуванню штрихового кодування для ідентифікації бібліографічних записів [10], [15].

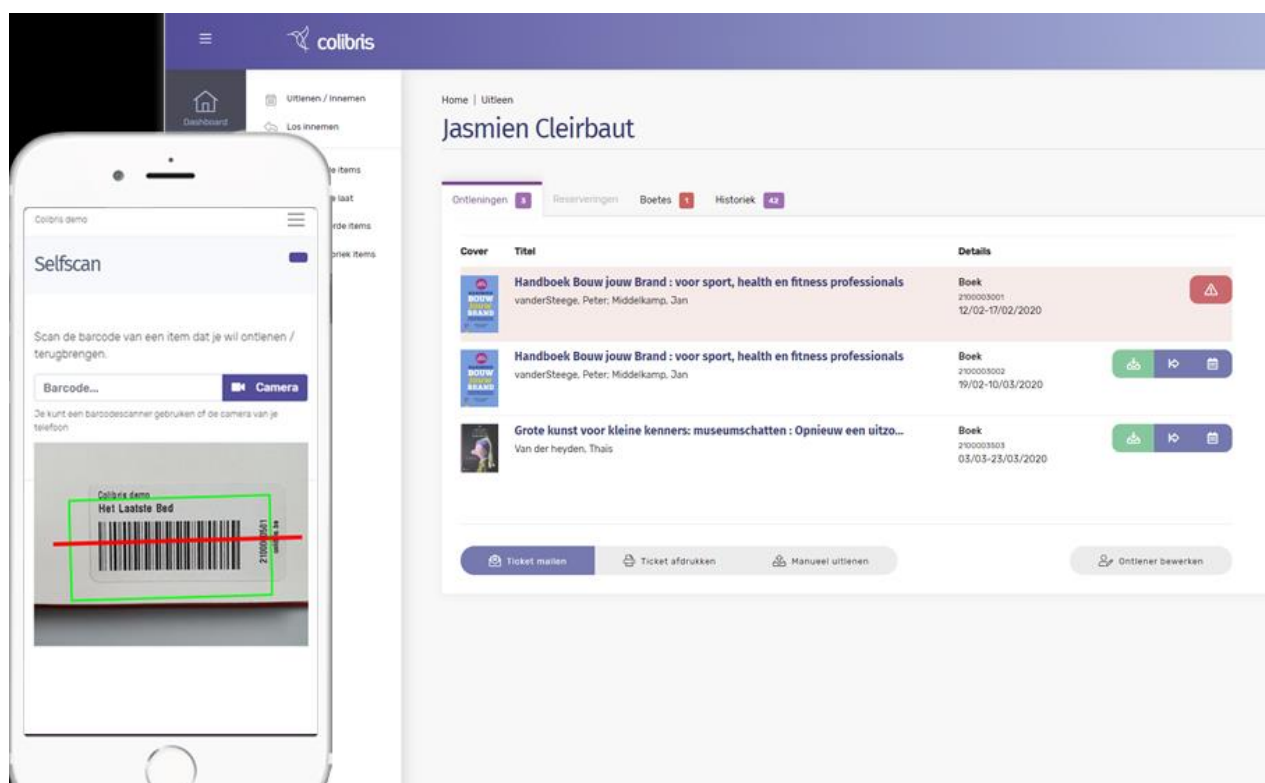


Рисунок 1.3 – Програмна система Colibris [15]

Процес взаємодії з бібліотечним фондом у програмній системі Colibris побудований так, щоб видача та повернення примірників не потребували додаткових складних дій: усе ґрунтується на унікальних кодах, що зчитуються під час реєстрації руху матеріалів. Програмна система Colibris підтримує технологію самостійного сканування, що робить обслуговування ще оперативнішим. Пакети рішень для установ включають не лише програмну частину, але й обладнання для зчитування кодів, постійні оновлення, інструктаж користувачів, консультаційну допомогу фахівців та надійні механізми резервного копіювання. Особливістю платформи є те, що кількість осіб, які працюють із системою, не має обмежень, незалежно від їхнього рівня доступу [10], [15].

Програмна система Colibris використовується широким колом організацій, серед яких навчальні заклади, урядові структури, медичні установи та музеї, що свідчить про універсальність технологічного рішення. Його розвиток орієнтовано на забезпечення максимально доступної вартості для

цільових користувачів, а також на безперервне вдосконалення функціональних можливостей та сервісної підтримки [10], [15].

Програмна система Colibris має такі особливості [10], [15]:

- хмарна архітектура: програмна система Colibris забезпечує доступ до бібліотечних ресурсів через мережу без потреби у локальному встановленні програми [10], [15];
- універсальний пошук: програмна система Colibris дає змогу здійснювати запити по різних видах колекцій і швидко отримувати бібліографічні дані [10], [15];
- підтримка штрих-кодів: програмна система Colibris реалізує ідентифікацію примірників за унікальними кодами та прискорює процес видачі й повернення [10], [15];
- функція самостійного сканування: програмна система Colibris дозволяє користувачам опрацьовувати вибрані матеріали без участі бібліотекаря [10], [15];
- інтегрований пакет послуг: програмна система Colibris містить оновлення системи, технічну підтримку, резервне копіювання та навчання персоналу в межах одного рішення [10], [15];
- відсутність обмежень доступу: програмна система Colibris не встановлює ліміти на кількість користувачів, адміністраторів чи одиниць фонду [10], [15];
- орієнтація на економічність: програмна система Colibris пропонує комплексний функціонал з мінімальними витратами для організацій різного масштабу [10], [15];
- широке застосування: програмна система Colibris використовується різними типами установ, що підтверджує її універсальність та адаптивність до потреб різних галузей [10], [15].

Серед недоліків програмної системи Colibris можна відзначити певну залежність від стабільності Інтернет з'єднання, оскільки робота платформи повністю ґрунтується на хмарній інфраструктурі, а також обмежені можливості

для глибокого налаштування інтерфейсу та окремих модулів, що може ускладнювати адаптацію рішення до специфічних вимог окремих установ. Додатковою складністю є потреба в спеціалізованому обладнанні для зчитування штрих-кодів, без якого система не розкриває свій повний функціонал, а також залежність від сторонніх сервісів, які забезпечують технічну підтримку та оновлення, що зменшує рівень автономності користувачів [10], [15].

Порівняльну характеристику систем опрацювання даних для керування бібліотечним фондом наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння систем опрацювання даних для керування бібліотечним фондом

Критерій порівняння	SLiMS [13]	Invenio [14]	Colibris [15]
Простота встановлення	+	—	+
Підтримка цифрових репозиторіїв	+—	+	—
Можливість розширення через модулі	+—	+	+—
Активність спільноти	+	+	+—
Інтеграція з іншими сервісами	+—	+	+—
Локалізація	+	+—	+—

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато систем опрацювання даних для керування бібліотечним фондом. Проте деякі системи опрацювання даних для керування бібліотечним фондом для створення та впровадження потребують значних матеріальних і часових ресурсів. Ліцензування програмного забезпечення, придбання обладнання, навчання персоналу та перенесення даних у цифрову форму можуть стати серйозним фінансовим навантаженням для бібліотечні установи.

Крім того, системи опрацювання даних для керування бібліотечним фондом вимагають постійного оновлення та технічної підтримки. Без регулярного супроводу вони поступово втрачають ефективність, виникають проблеми з безпекою та сумісністю з новими стандартами. Також можливими є труднощі під час адаптації персоналу, особливо якщо працівники не мають достатнього досвіду роботи з цифровими технологіями. Тому актуальною є розробка системи опрацювання даних для керування бібліотечним фондом.

При розробці системи опрацювання даних для керування бібліотечним фондом необхідно забезпечити такі функціональні вимоги:

- підтримка відображення головної сторінки застосунку для можливості доступу до основних функцій системи;
- підтримка можливості роботи з користувачами бібліотечного фонду;
- можливість отримання інформації про елементи бібліотечного фонду, зокрема, інформації про те, чи є відповідний елемент у наявності;
- можливість оформлення видачі книжок;
- підтримка можливості оформлення повернення книжок;
- підтримка можливості отримання аналітичної інформації про користувачів бібліотечного фонду, які не повернули взяті книги.

1.3 Висновки за розділом 1

Визначено, що системи опрацювання даних для керування бібліотечним фондом являють собою інтегровані програмні рішення, призначені для впорядкування роботи з інформаційними ресурсами, що зберігаються у бібліотеках та споріднених установах. Їхнім основним призначенням є забезпечення безперервного контролю за рухом фонду, підтримання точності бібліографічних описів, підвищення швидкості пошуку матеріалів, а також оптимізація взаємодії між працівниками та користувачами бібліотеки.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для керування бібліотечним

фондом. Проте деякі системи опрацювання даних для керування бібліотечним фондом для створення та впровадження потребують значних матеріальних і часових ресурсів. Ліцензування програмного забезпечення, придбання обладнання, навчання персоналу та перенесення даних у цифрову форму можуть стати серйозним фінансовим навантаженням для бібліотечні установи. Крім того, системи опрацювання даних для керування бібліотечним фондом вимагають постійного оновлення та технічної підтримки. Без регулярного супроводу вони поступово втрачають ефективність, виникають проблеми з безпекою та сумісністю з новими стандартами. Також можливими є труднощі під час адаптації персоналу, особливо якщо працівники не мають достатнього досвіду роботи з цифровими технологіями. Тому актуальною є розробка системи опрацювання даних для керування бібліотечним фондом.

Сформульовано функціональні вимоги до системи опрацювання даних для керування бібліотечним фондом.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці системи опрацювання даних для керування бібліотечним фондом було використано мову програмування Python [16]-[17].

Python є не лише інструментом для розробників-практиків, а й універсальним засобом для навчання та досліджень. Python може працювати на різних програмних платформах і легко поєднуватися з модулями, написаними іншими мовами (наприклад, C або Java). Завдяки цьому можна створювати гібридні рішення, у яких найбільш ресурсоємні компоненти виконуються максимально ефективно, а високорівневу логіку залишено Python [16]-[17].

Мова є відкритою і розвивається у динамічному співтоваристві, що постійно вдосконалює її можливості та впроваджує нові стандарти якості. Регулярні оновлення підтримують сучасні вимоги до продуктивності, структури коду та безпеки [16]- [17].

Крім того, Python має чітку та добре структуровану документацію, що значно полегшує розробку і підтримку програмного забезпечення. Коли йдеться про системи, які розвиватимуться багато років, це стає суттєвою перевагою – нові розробники можуть легко долучитися до проєкту і швидко розібратися в існуючому коді [16]-[17].

Таким чином, Python поєднує потужність, зручність і відкритість, що робить його перспективним вибором для широкого спектру застосувань – від навчальних проєктів до комплексних інформаційних систем [16]-[17].

Вибір Python як основної мови програмування для створення системи опрацювання даних у бібліотечній сфері є обґрунтованим через поєднання технологічної гнучкості та зручності реалізації функціональних можливостей. Мова програмування Python відзначається простим і зрозумілим синтаксисом, що дозволяє прискорити розробку та зменшити кількість помилок у коді. Це особливо важливо при проєктуванні складних інформаційних систем, де

потрібно швидко адаптувати рішення до вимог користувачів або змін у структурі даних [16]-[17].

Мова має добре розвинену екосистему бібліотек, орієнтованих на роботу з базами даних, аналітику та вебінтерфейси. Інструменти на кшталт Django чи Flask дають змогу створювати зручні вебзастосунки для взаємодії з бібліотечним фондом, тоді як SQLAlchemy або інші ORM-рішення значно спрощують роботу зі сховищами інформації. Це забезпечує можливість реалізації як локальної системи, так і повноцінного Інтернет доступу для користувачів [16]-[17].

Обґрунтування вибору мови програмування для розробки системи опрацювання даних для керування бібліотечним фондом наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки системи опрацювання даних для керування бібліотечним фондом

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	C++
Швидкість реалізації функціоналу	+	+-	+-
Робота з базами даних	+	+	+-
Вартість та доступність інструментів	+	+-	+
Простота розробки	+	+-	-
Спільнота та підтримка	+	+-	+-

Окремою перевагою Python є активна спільнота та велика кількість відкритих рішень, які можуть бути використані як основа для нових розробок, що зменшує час і витрати на створення системи. Крім того, Python легко інтегрується з сучасними інструментами машинного навчання, що відкриває

перспективи впровадження розумного пошуку, автоматичного аналізу читацького попиту або рекомендаційних сервісів у майбутньому [16]-[17].

Таким чином, Python забезпечує баланс між швидкістю розробки, розширюваністю функціоналу та технологічною перспективністю, що робить його доцільним вибором для створення ефективної системи керування бібліотечним фондом у сучасному інформаційному середовищі.

Отже, для реалізації системи опрацювання даних для керування бібліотечним фондом обрано мову програмування Python через поєднання технологічної гнучкості та зручності реалізації функціональних можливостей. Python відзначається простим і зрозумілим синтаксисом, що дозволяє прискорити розробку та зменшити кількість помилок у коді. Крім того, мова Python має добре розвинену екосистему бібліотек, орієнтованих на роботу з базами даних, аналітику та вебінтерфейси.

2.2 Вибір середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом

В якості середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом було обрано середовище PyCharm [18]-[19].

Середовище розробки PyCharm дозволяє створювати комфортне робоче середовище, орієнтоване на продуктивність і довготривалу підтримку програмних проєктів. Це середовище пропонує не лише широкий спектр інструментів для написання коду, а й глибоку автоматизацію повсякденних задач розробника [18]-[19].

Однією з суттєвих характеристик PyCharm є розвинена система рефакторингу, яка допомагає швидко змінювати структуру коду без ризику порушення його логіки. Це особливо корисно для масштабних систем, таких як ті, що забезпечують керування бібліотечними фондами, де код постійно оновлюється та доповнюється новим функціоналом [18]-[19].

PyCharm активно стежить за дотриманням стандартів оформлення коду, що підвищує зрозумілість програми й полегшує роботу всіх учасників проєкту. До того ж, вбудовані засоби моніторингу продуктивності допомагають виявляти місця, де система може працювати повільніше, й оперативно оптимізувати їх [18]-[19].

Не менш важливою перевагою є можливість налаштовувати середовище під власні потреби: від вибору теми інтерфейсу до встановлення додаткових модулів через велику бібліотеку плагінів. Це робить PyCharm гнучким інструментом, який може адаптуватися під конкретний стиль і вимоги розробника [18]-[19].

Отже, PyCharm не лише прискорює процес створення програмного забезпечення, а й забезпечує довготривалу підтримку проєкту, сприяє підвищенню якості коду та комфорту розробників, що є важливим фактором для реалізації стабільних і масштабованих інформаційних систем [18]-[19].

Використання PyCharm як середовища розробки для створення системи опрацювання даних у бібліотечній сфері є виправданим завдяки його професійній орієнтації на Python та комплексному набору інструментів, необхідних для розробки повноцінних застосунків. PyCharm забезпечує зручну навігацію в проєкті, автоматичне доповнення коду та інтелектуальне виявлення помилок, що допомагає підвищити якість програмного продукту й скоротити час його створення [18]-[19].

Однією з ключових переваг цього середовища є вбудована підтримка вебфреймворків, роботи з базами даних, створення та тестування REST-інтерфейсів – тобто саме тих функцій, які є критично важливими для систем керування бібліотечним фондом. PyCharm дозволяє легко під'єднуватися до різних СУБД, переглядати структуру таблиць і виконувати запити безпосередньо в межах IDE, що робить роботу з даними значно ефективнішою [18]-[19].

Також середовище має потужні можливості для віртуалізації та управління залежностями проєкту. Це забезпечує стабільність розробки та

гарантує, що система буде коректно працювати на цільовому обладнанні бібліотеки чи сервері [18]-[19].

Важливо й те, що PyCharm підтримує командну роботу: інтеграція з Git та іншими інструментами контролю версій дозволяє синхронізувати внесені зміни між різними розробниками, що особливо корисно під час еволюції системи та впровадження нових функцій [18]-[19].

Сукупність цих можливостей робить PyCharm оптимальним вибором для проєкту, де потрібна надійність, масштабованість та повний цикл роботи з даними, від програмування та тестування до розгортання й подальшої підтримки системи [18]-[19].

Обґрунтування вибору середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	IDLE
Автоматичне доповнення коду	+	+	—
Розвинені інструменти рефакторингу	+	+—	—
Інтеграція з базами даних	+	+—	+—
Підтримка вебфреймворків	+	+—	+—
Наявність професійних засобів налагодження	+	+	—

Таким чином, для створення системи опрацювання даних для керування бібліотечним фондом обрано середовище розробки PyCharm, що поєднує інструменти для роботи з базами даних, веброзробки, налагодження та командної взаємодії.

2.3 Вибір засобів зберігання та опрацювання даних

Для зберігання та опрацювання даних було обрано систему управління базами даних SQLite [20]-[21].

Використання SQLite для розробки системи опрацювання даних у бібліотечному середовищі є раціональним рішенням через поєднання простоти та надійності, які ця СУБД пропонує. Система управління базами даних SQLite здатна працювати у середовищах з обмеженими ресурсами та мінімальними вимогами до обслуговування. Установам, які не мають спеціального ІТ-персоналу, важливо мати рішення, що практично не потребує конфігурації й може стабільно функціонувати навіть на звичайному офісному обладнанні [20]-[21].

Ще однією перевагою SQLite є її автономність: усі дані й механізми управління базою знаходяться в одному файлі, тому відпадає потреба у складній системі доступів чи налаштуванні серверних служб. Це дозволяє уникнути залежності від мережевої інфраструктури – система залишатиметься працездатною навіть за відсутності підключення до Інтернету [20]-[21].

Важливим є й те, що SQLite має високу сумісність між різними операційними системами. У разі подальшого розширення бібліотечної системи або перенесення її на інший пристрій питання міграції даних буде вирішене максимально просто: достатньо скопіювати один файл бази [20]-[21].

Попри свою легку архітектуру, СУБД підтримує стандарт SQL, що спрощує розробку, а також дає можливість у майбутньому перейти на більш масштабне серверне рішення (наприклад, PostgreSQL або MySQL), якщо бібліотечна система значно зросте. Таким чином, SQLite виступає не лише як

доступний інструмент для старту, але й як гнучкий фундамент для розвитку системи [20]-[21].

Отже, легкість інтеграції, невибагливість до ресурсів, відсутність потреби у технічному супроводі та можливість подальшого масштабування роблять SQLite практичним і стратегічно правильним вибором для створення системи керування бібліотечним фондом [20]-[21].

Системі управління базами даних SQLite не потрібне окреме серверне програмне забезпечення чи складна конфігурація, тому система може бути розгорнута практично миттєво та без додаткових витрат на інфраструктуру. Це дозволяє зосередитися на функціональних аспектах застосунку, а не на адмініструванні середовища [20]-[21].

SQLite зберігає дані у вигляді одного файлу, що полегшує резервування, перенесення або оновлення бази без участі спеціалістів з адміністрування баз даних. Така особливість особливо вигідна для невеликих бібліотек або для локальної системи, де важливо мінімізувати технічні вимоги та забезпечити оперативне обслуговування [20]-[21].

СУБД SQLite забезпечує достатню продуктивність для обробки повсякденних операцій: обліку книжкового фонду, реєстрації читачів, виконання пошуку та статистичного аналізу руху видань. При цьому SQLite гарантує цілісність і захист даних за допомогою транзакційного механізму та перевірених засобів збереження інформації [20]-[21].

Ще одна перевага полягає у хорошій сумісності з Python: мова має вбудовані засоби для роботи з SQLite без додаткових драйверів, що спрощує розробку і прискорює взаємодію застосунку з базою [20]-[21].

Таким чином, SQLite є доцільним вибором, коли необхідно отримати компактну, недорогу та водночас функціональну систему зберігання даних, здатну забезпечити стабільну роботу бібліотечного програмного забезпечення без зайвого технічного ускладнення [20]-[21].

Обґрунтування вибору засобів зберігання та опрацювання даних наведено у таблиці 2.3.

Таблиця 2.3 – Обґрунтування вибору засобів зберігання та опрацювання даних

Критерій порівняння систем управління базами даних	Системи управління базами даних		
	SQLite	PostgreSQL	Oracle DB
Легка інтеграція з Python	+	+	+–
Можливість автономної роботи	+	+–	–
Низька вартість використання	+	+	–
Продуктивність для невеликих та середніх баз	+	+–	+–
Мобільність та переносимість даних	+	+–	–

Таким чином, для зберігання та опрацювання даних обрано систему управління базами даних SQLite через простоту використання, автономність, легку інтеграцію з Python і мінімальні вимоги до технічного супроводу. Інші системи управління базами даних підходять для масштабних або корпоративних систем, але для локальної або невеликих бібліотек вони надто ресурсозатратні та складні в адмініструванні.

2.4 Висновки за розділом 2

Для реалізації системи опрацювання даних для керування бібліотечним фондом обрано мову програмування Python через поєднання технологічної гнучкості та зручності реалізації функціональних можливостей. Python відзначається простим і зрозумілим синтаксисом, що дозволяє прискорити розробку та зменшити кількість помилок у коді. Крім того, мова Python має

добре розвинену екосистему бібліотек, орієнтованих на роботу з базами даних, аналітику та вебінтерфейси.

Для створення системи опрацювання даних для керування бібліотечним фондом обрано середовище розробки PyCharm, що поєднує інструменти для роботи з базами даних, веброзробки, налагодження та командної взаємодії.

Для зберігання та опрацювання даних обрано систему управління базами даних SQLite через простоту використання, автономність, легку інтеграцію з Python і мінімальні вимоги до технічного супроводу. Інші системи управління базами даних підходять для масштабних або корпоративних систем, але для локальної або невеликих бібліотек вони надто ресурсозатратні та складні в адмініструванні.

3 ОПИС СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

3.1 Структура системи опрацювання даних для керування бібліотечним фондом

Структуру системи опрацювання даних для керування бібліотечним фондом наведено на рис. 3.1.

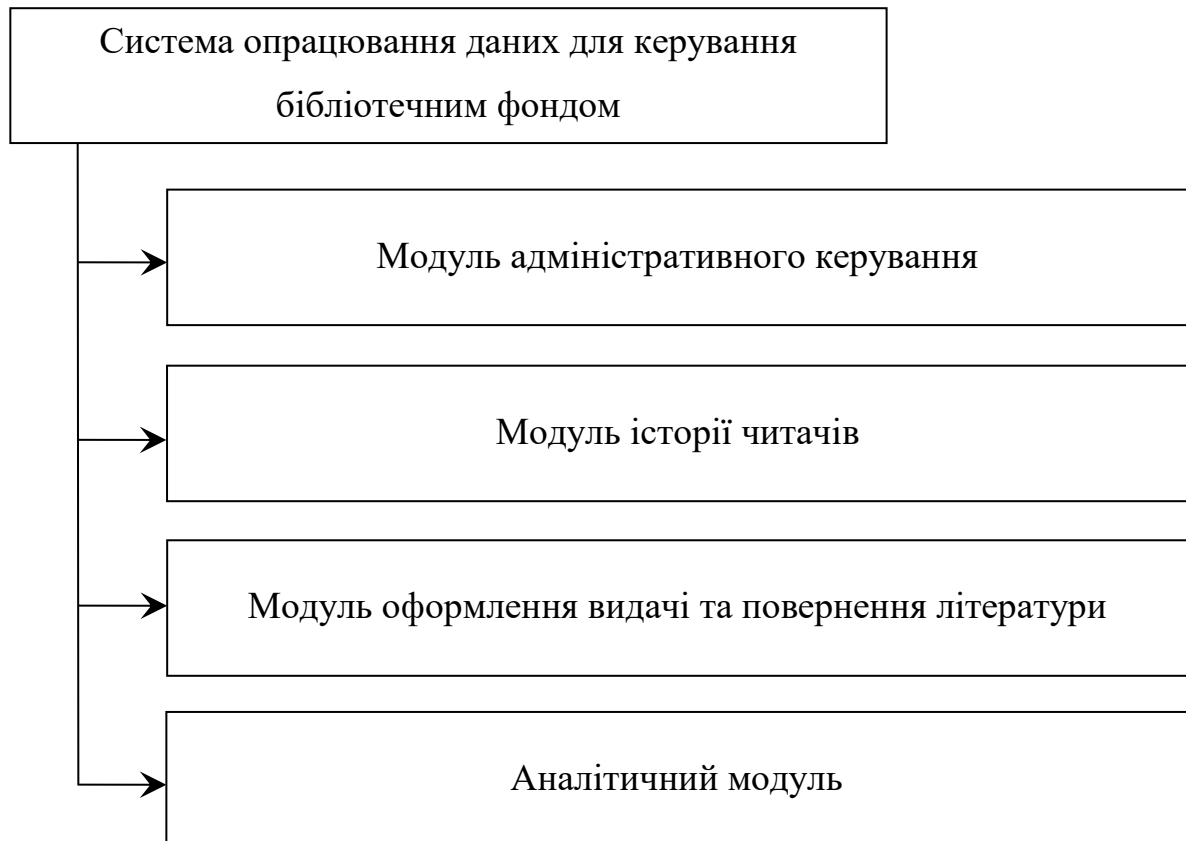


Рисунок 3.1 – Структура системи опрацювання даних для керування бібліотечним фондом

Структура системи опрацювання даних, призначеної для керування бібліотечним фондом, побудована таким чином, щоб забезпечити узгоджене функціонування всіх процесів, пов'язаних із роботою з книжковими ресурсами та обслуговуванням читачів. У її основу покладено чотири функціональні модулі, кожен із яких виконує власну роль, але водночас тісно взаємодіє з іншими складовими, утворюючи єдину інформаційну платформу.

Модуль адміністративного керування відповідає за формування й підтримку актуальної інформаційної бази. Через цей модуль здійснюється створення нових записів про літературу та користувачів, оновлення відомостей у разі потреби, а також видалення застарілих чи некоректних даних. Саме адміністративний модуль забезпечує контроль над структурою фонду та підтримує коректність інформації, яка використовується всіма іншими модулями.

Модуль історії читачів дозволяє відстежувати індивідуальну активність кожного користувача. У ній відображається перелік отриманих примірників зі зазначенням часу їх видачі та повернення. Завдяки цьому модулю легко оцінюється динаміка користування фондами, формується хронологія взаємодії зі всіма книгами та забезпечується швидкий доступ до інформації, необхідної для контролю та внутрішнього аналізу.

Модуль оформлення видачі та повернення літератури. У межах цього модуля вибирається потрібна книга, а також визначається користувач, який отримує або повертає видання. Механізм обробки транзакцій організований так, щоб фіксація кожної операції відбувалася оперативно та безпомилково, гарантуючи точність обліку бібліотечних одиниць і стабільність роботи всієї системи.

Аналітичний модуль формує перелік користувачів, які не здали книги протягом тривалого часу, зокрема понад один місяць. Модуль аналізу боржників допомагає оперативно виявляти проблемні випадки та полегшує роботу персоналу з контролем за поверненням фонду. Наявність цього модуля забезпечує більшу дисциплінованість користувачів та ефективніше управління ресурсами бібліотеки.

Завдяки такій структурній організації система працює як цілісний механізм, де кожен модуль виконує важливу функцію й забезпечує умови для безперервного, точного та впорядкованого опрацювання всіх даних, пов'язаних із керуванням бібліотечним фондом.

Архітектуру системи опрацювання даних для керування бібліотечним фондом (рис. 3.2) побудовано з урахуванням запропонованої вищі структури функціональних модулів та моделі Model-View-Controller (MVC).

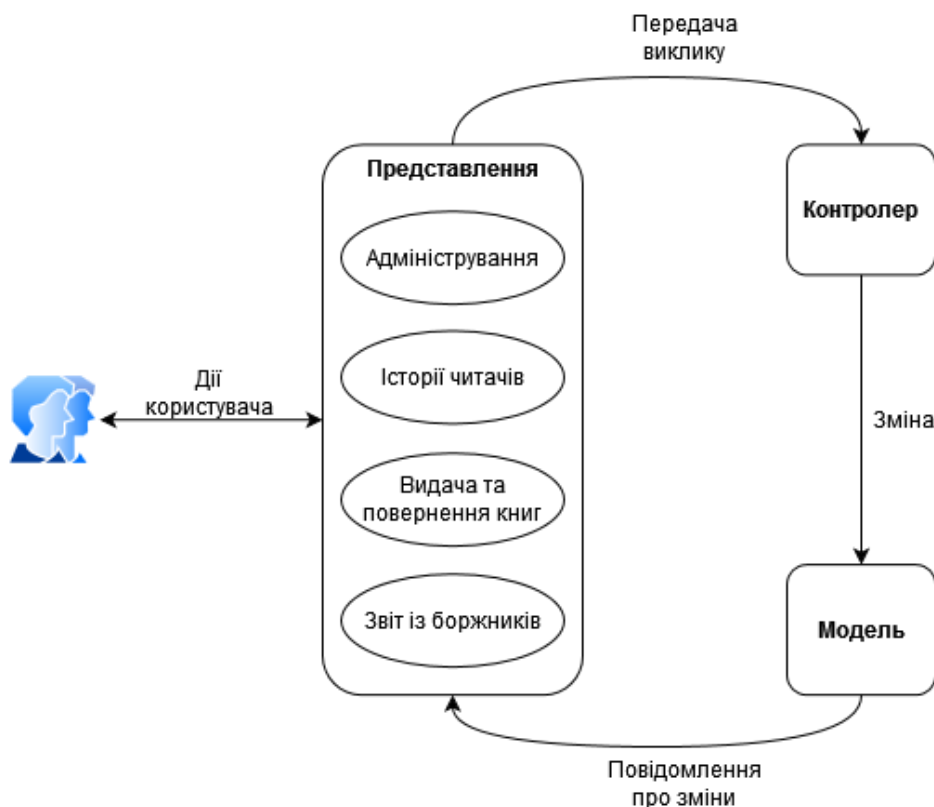


Рисунок 3.2 – Архітектура системи опрацювання даних для керування бібліотечним фондом на основі моделі MVC

Архітектура системи опрацювання даних для керування бібліотечним фондом організована відповідно до підходу MVC, де кожен елемент виконує чітко визначену роль у взаємодії між інтерфейсом, логікою обробки та сховищем даних. У центрі функціонування цієї структури знаходиться Представлення, через яке користувачі ініціюють робочі дії. Саме сюди спрямовані всі їхні запити, що відображені у вигляді стрілок «Дії користувача» на схемі. Представлення охоплює чотири функціональні блоки: підсистему адміністрування, інструменти відображення історії читачів, механізми реєстрації процесів видачі й повернення літератури та модуль, відповідальний за автоматичне створення списку боржників.

Після надходження будь-якої ініціативи від користувача інтерфейс системи опрацювання даних для керування бібліотечним фондом передає її до Контролера, який виступає центральним координатором системи. Контролер системи опрацювання даних для керування бібліотечним фондом інтерпретує отримані запити, перевіряє їхню коректність і визначає подальший шлях обробки. За потреби він звертається до Моделі системи опрацювання даних для керування бібліотечним фондом, що відповідає за логіку доступу до інформаційних ресурсів та реалізацію операцій із даними. Взаємодія Контролера з Моделлю відбувається у двох напрямках: спочатку ініціюється виконання відповідної операції, а після завершення обробки Модель повертає Контролеру результати, які пізніше відображаються у Представленні.

Модель у цій архітектурі виконує роль ядра бізнес-логіки. Через неї реалізується збереження, модифікація та отримання відомостей про книги, читачів, взяті екземпляри та борги. Саме Модель забезпечує узгодженість і цілісність бібліотечного фонду, обробляє операції додавання або редагування даних, формує інформацію про хронологію користування літературою, контролює статуси видачі та повернення, а також генерує набір відомостей, необхідних для створення звіту про прострочені книги.

Після обробки даних Контролер системи опрацювання даних для керування бібліотечним фондом повертає сформовану інформацію назад у Представлення, де вона перетворюється у зрозумілий для користувача вигляд. Таким чином, кожен із чотирьох модулів інтерфейсу відображає лише те, що передає йому Контролер системи опрацювання даних для керування бібліотечним фондом, не взаємодіючи безпосередньо з джерелом даних.

Уся система опрацювання даних для керування бібліотечним фондом функціонує як узгоджена структура, у якій Представлення відповідає за взаємодію з користувачами, Контролер керує потоками запитів, а Модель забезпечує роботу з даними. Така організація системи опрацювання даних для керування бібліотечним фондом дозволяє легко підтримувати та розширювати систему, розмежовує відповідальність між компонентами та забезпечує

ефективне керування бібліотечним фондом незалежно від обсягу інформації чи кількості користувачів.

3.2 Функціонування системи опрацювання даних для керування бібліотечним фондом

Функціонування системи опрацювання даних для керування бібліотечним фондом подамо за допомогою схеми, зображеної на рис. 3.3.

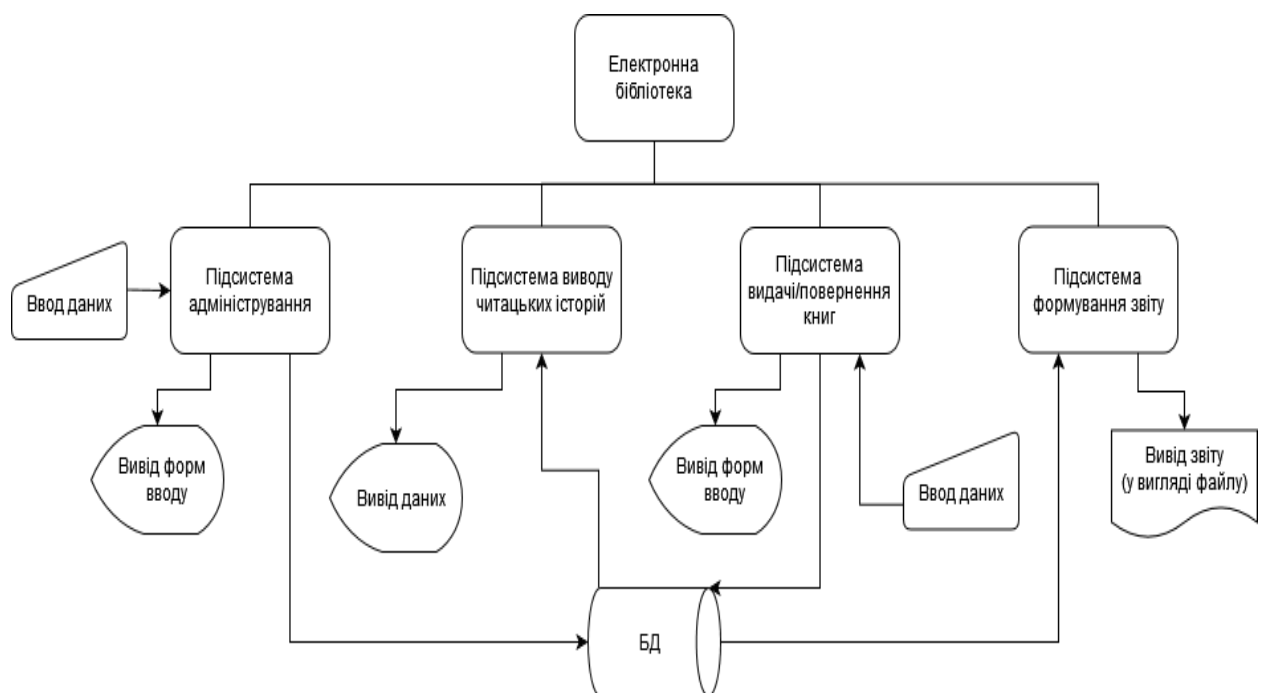


Рисунок 3.3 – Схема функціонування системи опрацювання даних для керування бібліотечним фондом

Функціонування системи опрацювання даних для керування бібліотечним фондом ґрунтується на послідовній взаємодії окремих механізмів, кожен з яких відповідає за певний сегмент роботи з інформацією. У процесі побудови системи було сформовано низку логічних схем, що охоплюють обробку довідкових даних про книги та читачів, фіксацію історій користування літературою, відстеження руху видань між фондом і читачами, а також генерацію довідкових матеріалів для службової звітності.

Процес адміністрування реалізує операції, пов'язані з підтримкою актуального стану даних. Система передбачає можливість створення нових записів про книги та користувачів, внесення змін у вже існуючі відомості та вилучення інформації, що втратила актуальність. Усі ці дії виконуються через інтерфейс, який спрямовує команди на відповідні модулі обробки, де відбувається внесення правок у сховище даних.

Режим перегляду історій читачів організований таким чином, щоб користувач міг оперативно знаходити інформацію про використання бібліотечного фонду конкретною особою. Після вибору потрібного читача відображається впорядкована добірка відомостей про всі книги, які ця особа отримувала в певні періоди, із зазначенням дат видачі та фактів повернення. Система автоматично формує ці дані на основі журналу руху примірників і показує їх у вигляді узгодженої таблиці.

Механізм оформлення видачі та реєстрації повернення книг керує переміщенням кожного окремого примірника. Коли необхідно видати книгу, система пропонує перелік доступних позицій з вказаними ідентифікаторами примірників та їхнім статусом у фонді. Після вибору потрібного екземпляра користувач визначає читача, який отримує книгу, і система фіксує відповідний запис. У випадку повернення процедура працює аналогічно: достатньо підтвердити, що конкретний примірник повернувся, після чого його статус змінюється на доступний.

Окремий логічний модуль відповідає за побудову контрольного звіту. Він автоматично аналізує дані про всі видані книги, виявляє примірники, повернення яких прострочене більше ніж на тридцять днів, та формує структурований файл для подальшого використання працівниками бібліотеки. До такого документа включаються відомості про читача, його контакти, інформація про книгу та конкретний екземпляр, що залишився неповернутим, а також дата видачі, яка дозволяє визначити термін прострочення.

У результаті взаємодії цих алгоритмів система забезпечує повний цикл роботи з бібліотечним фондом: від підтримки структурованої бази даних до

фіксації кожного кроку руху примірників і автоматичного формування службової документації.

3.3 Розробка бази даних системи опрацювання даних для керування бібліотечним фондом

База даних системи опрацювання даних для керування бібліотечним фондом розроблена таким чином, щоб відображати всі об'єкти, з якими працює бібліотека, а також їхні взаємозв'язки.

Схему бази даних системи опрацювання даних для керування бібліотечним фондом наведено на рис. 3.4.

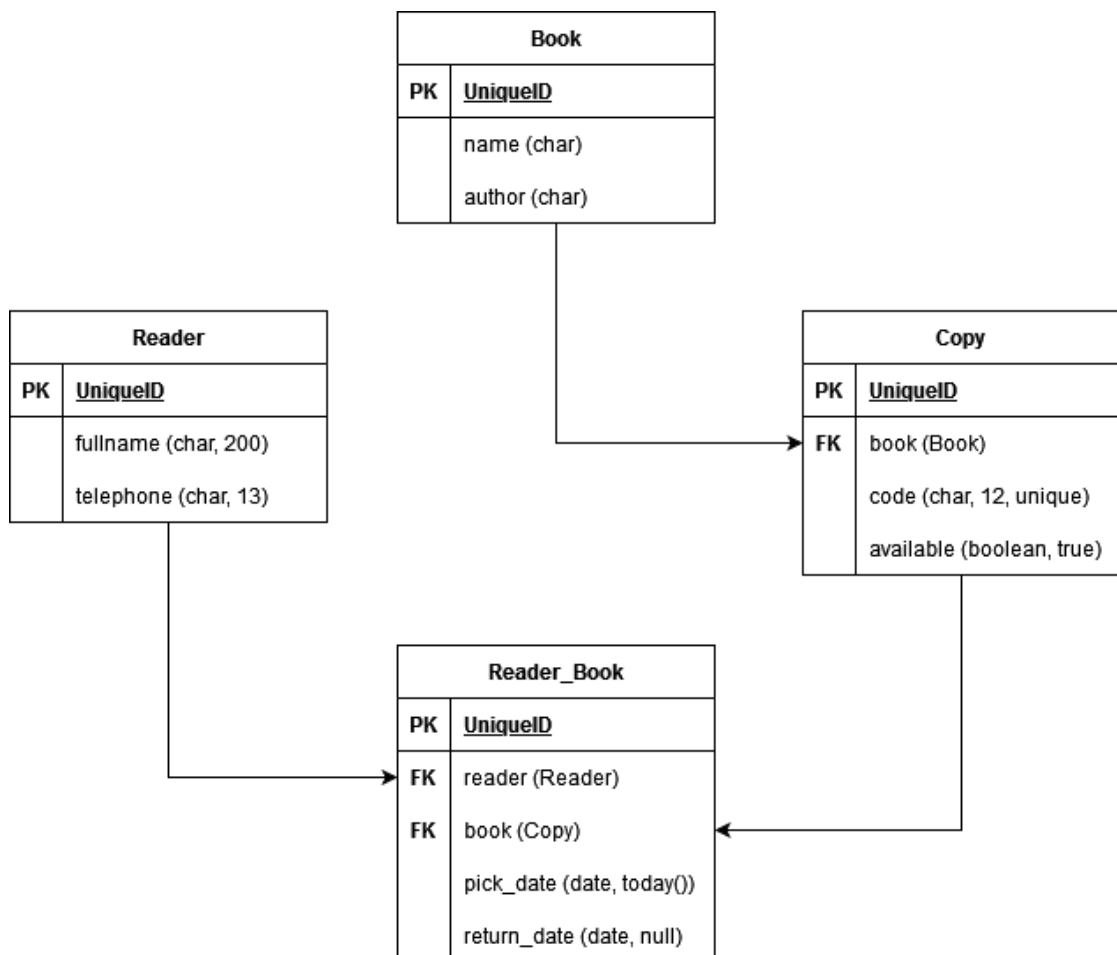


Рисунок 3.4 – Схема бази даних системи опрацювання даних для керування бібліотечним фондом

Концептуальна модель бази даних системи опрацювання даних для керування бібліотечним фондом охоплює кілька логічних сутностей, кожна з яких відповідає за певний тип інформації, а разом вони утворюють цілісну структуру, здатну відтворювати всі процеси, пов'язані з обліком, видачею та поверненням літератури.

Центральне місце займає група даних про користувачів бібліотеки. У спеціальній сутності зберігаються унікальні ідентифікатори читачів, їхні повні імена та контактні номери. Поля цієї сутності мають фіксовані обмеження щодо типів та розмірів даних, що забезпечує однорідність усіх записів і унеможливорює появу некоректної інформації. Аналогічно організовано структуру, яка відповідає за дані про книжковий фонд: кожне видання має власний ідентифікаційний номер, назву та ім'я автора, що дозволяє однозначно ідентифікувати будь-яку книгу у базі.

Для відображення фізичних примірників використовується окрема сутність, оскільки одна і та сама книга може бути представлена кількома копіями. Кожен примірник пов'язаний із записом про книгу через зовнішній ключ, має власний інвентарний код і позначку, що вказує на його поточний статус у фонді. Завдяки цьому база даних може відстежувати, чи доступний конкретний екземпляр, чи він тимчасово перебуває у читача.

Окремий блок моделі зосереджений на описі операцій, пов'язаних із рухом примірників між фондом та користувачами. У спеціальній сутності фіксується, хто саме отримав конкретну копію книги, коли вона була видана й чи повернуто її назад. Для цього використано зв'язки з таблицями читачів і примірників, а також поля з датами, що дозволяють точно визначити часові межі використання книги. Наявність таких записів дає можливість системі автоматично визначати прострочення та забезпечувати формування необхідних звітів.

Уся структура бази даних системи опрацювання даних для керування бібліотечним фондом побудована таким чином, щоб дані передавали логіку роботи бібліотечного процесу: від загальної інформації про книги й

користувачів до детального відстеження шляху кожного окремого примірника. Це забезпечує узгодженість інформації, зручність пошуку та обробки, а також можливість автоматизації багатьох рутинних операцій у системі керування бібліотечним фондом.

3.4 Проєктування інтерфейсу системи опрацювання даних для керування бібліотечним фондом

Проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для керування бібліотечним фондом виконано з урахуванням вимог технічного завдання. При розробленні системи опрацювання даних для керування бібліотечним фондом враховані функціональні вимоги до програми:

- підтримка відображення головної сторінки застосунку для можливості доступу до основних функцій системи;
- підтримка можливості роботи з користувачами бібліотечного фонду;
- можливість отримання інформації про елементи бібліотечного фонду, зокрема, інформації про те, чи є відповідний елемент у наявності;
- можливість оформлення видачі книжок;
- підтримка можливості оформлення повернення книжок;
- підтримка можливості отримання аналітичної інформації про користувачів бібліотечного фонду, які не повернули взяті книги.

Інтерфейс головної форми системи опрацювання даних для керування бібліотечним фондом наведено на рис. 3.5.

Інтерфейс сторінки з інформацією про користувачів бібліотеки наведено на рис. 3.6.

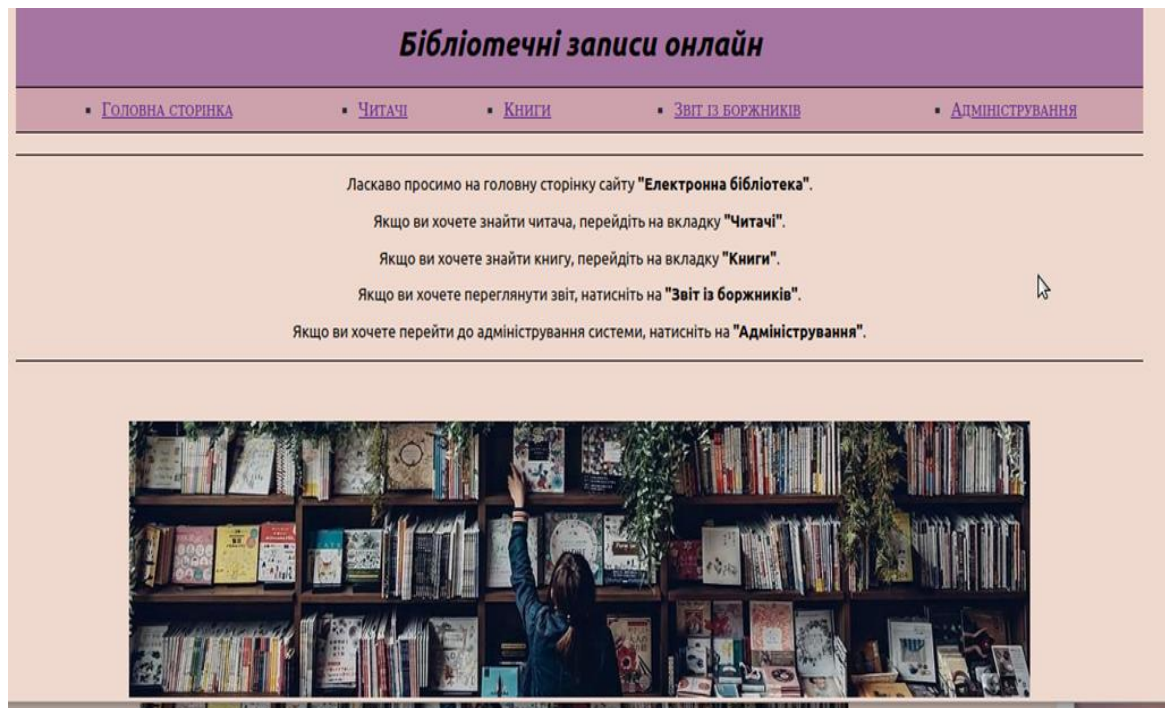


Рисунок 3.5 – Інтерфейс головної форми системи опрацювання даних для керування бібліотечним фондом

Бібліотечні записи онлайн	
Головна сторінка Читачі Книги Звіт із боржників Адміністрування	
Читач	Телефон
▪ Іваненко Володимир Іванович	+380509997788
▪ Казаченко Ігор Андрійович	+380931112233
▪ Бедан Роман Петрович	+380931112234
▪ Василенко Жанна Володимирівна	+380938882235
▪ Лазар Петро Васильович	+380934442235
▪ Олексієнко Ігор Романович	+380933332235

Рисунок 3.6 – Сторінка з інформацією про користувачів бібліотеки

Форму видачі книжки читачу наведено на рис. 3.7.

Бібліотечні записи онлайн

[Головна сторінка](#)
[Читачі](#)
[Книги](#)
[Звіт із боржників](#)
[Адміністрування](#)

Гаррі Поттер і філософський камінь - Дж. К. Роулінг

A5421

Читач:

Казаченко Ігор Андрійович
 Бедан Роман Петрович
 Василенко Жанна Володимирівна
Лазар Петро Васильович
 Олексієнко Ігор Романович

Віддати

Рисунок 3.7 – Форма видачі книжки читачу

Форму повернення книги наведено на рис. 3.8.

Бібліотечні записи онлайн

[Головна сторінка](#)
[Читачі](#)
[Книги](#)
[Звіт із боржників](#)
[Адміністрування](#)

Гаррі Поттер і філософський камінь - Дж. К. Роулінг

A5422

Читач	Узято
Олексієнко Ігор Романович	26 травня 2026 р.

Повернути

Рисунок 3.8 – Форма повернення книги

3.5 Висновки за розділом 3

Розроблено систему опрацювання даних для керування бібліотечним фондом.

Запропоновано структуру системи опрацювання даних для керування бібліотечним фондом. Визначено, що в основу структури системи опрацювання

даних для керування бібліотечним фондом покладено чотири функціональні модулі (модуль адміністративного керування, модуль історії читачів, модуль оформлення видачі та повернення літератури та аналітичний модуль), кожен із яких виконує власну роль, але водночас тісно взаємодіє з іншими складовими, утворюючи єдину інформаційну платформу.

Архітектура системи опрацювання даних для керування бібліотечним фондом організована відповідно до підходу MVC, де кожен елемент виконує чітко визначену роль у взаємодії між інтерфейсом, логікою обробки та сховищем даних, внаслідок чого уся система функціонує як узгоджена структура, у якій Представлення відповідає за взаємодію з користувачами, Контролер керує потоками запитів, а Модель забезпечує роботу з даними. Така організація системи опрацювання даних для керування бібліотечним фондом дозволяє легко підтримувати та розширювати систему, розмежовує відповідальність між компонентами та забезпечує ефективне керування бібліотечним фондом незалежно від обсягу інформації чи кількості користувачів.

Описано функціонування системи опрацювання даних для керування бібліотечним фондом. Функціонування системи опрацювання даних для керування бібліотечним фондом ґрунтується на послідовній взаємодії окремих механізмів, кожен з яких відповідає за певний сегмент роботи з інформацією. У процесі побудови системи було сформовано низку логічних схем, що охоплюють обробку довідкових даних про книги та читачів, фіксацію історій користування літературою, відстеження руху видань між фондом і читачами, а також генерацію довідкових матеріалів для службової звітності.

Розроблено базу даних системи опрацювання даних для керування бібліотечним фондом, яка побудована таким чином, щоб дані передавали логіку роботи бібліотечного процесу: від загальної інформації про книги й користувачів до детального відстеження шляху кожного окремого примірника. Це забезпечує узгодженість інформації, зручність пошуку та обробки, а також

можливість автоматизації багатьох рутинних операцій у системі керування бібліотечним фондом.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для керування бібліотечним фондом.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ СИСТЕМИ ОПРАЦЮВАННЯ ДАНИХ

4.1 Призначення й умови застосування системи опрацювання даних

Система опрацювання даних призначена для підтримки процесу керування бібліотечним фондом. Система опрацювання даних написана на мові Python через поєднання технологічної гнучкості та зручності реалізації функціональних можливостей, крім того Python відзначається простим і зрозумілим синтаксисом, що дозволяє прискорити розробку та зменшити кількість помилок у коді. В якості середовища розробки для створення системи опрацювання даних для керування бібліотечним фондом обрано PyCharm, що поєднує інструменти для роботи з базами даних, веброзробки, налагодження та командної взаємодії. Для зберігання та опрацювання даних обрано систему управління базами даних SQLite через простоту використання, автономність, легку інтеграцію з Python і мінімальні вимоги до технічного супроводу.

Система опрацювання даних для керування бібліотечним фондом забезпечує виконання таких функцій:

- підтримка відображення головної сторінки застосунку для можливості доступу до основних функцій системи;
- підтримка можливості роботи з користувачами бібліотечного фонду;
- можливість отримання інформації про елементи бібліотечного фонду, зокрема, інформації про те, чи є відповідний елемент у наявності;
- можливість оформлення видачі книжок;
- підтримка можливості оформлення повернення книжок;
- підтримка можливості отримання аналітичної інформації про користувачів бібліотечного фонду, які не повернули взяті книги.

Для роботи системи опрацювання даних для керування бібліотечним фондом потрібні такі технічні та програмні ресурси: апаратна частина має забезпечувати стабільне виконання всіх операцій, пов'язаних з обробкою

даних, формуванням звітів та одночасною взаємодією декількох користувачів. Оптимально, щоб комп'ютерна система була оснащена процесором із частотою, яка не опускається нижче рівня, достатнього для швидкого виконання запитів до бази даних і робочої логіки програми. Тактової швидкості сучасного багатоядерного процесора початкового рівня цілком вистачає для комфортної роботи у звичайному бібліотечному середовищі, де навантаження залишається помірним.

Оперативна пам'ять повинна мати обсяг, що гарантує плавність роботи інтерфейсу та відсутність затримок під час виконання основних операцій, зокрема під час звернень до сховища даних, сортування інформації, побудови звітних документів або обробки таблиць із великим обсягом записів. Мінімальний рекомендований обсяг пам'яті відповідає типовим вимогам настільних застосунків, проте бажано мати запас, який дозволить системі працювати без уповільнень навіть у разі збільшення бази.

Обсяг постійного сховища має бути достатнім для розміщення самої програми, бібліотек, залежностей та файлів бази даних. Оскільки використовується СУБД не вимагає складних інсталяцій або значного простору, вимоги до накопичувача залишаються невисокими, але бажано використовувати твердотільний диск, що скорочує час доступу до файлів і підвищує загальну продуктивність роботи.

Щодо програмної частини, операційна система повинна підтримувати сучасні версії мови програмування Python, а також мати можливість запускати додаткові бібліотеки, необхідні для роботи графічного інтерфейсу, взаємодії з базою даних та створення звітних документів. Середовище виконання має бути сумісним з інструментами розробки та компонентами, що використовуються системою.

Функціональні характеристики розробленої системи опрацювання даних для керування бібліотечним фондом наведено у технічному завданні (додаток А). Фрагмент тексту програмного коду системи опрацювання даних для керування бібліотечним фондом наведено у додатку Б.

4.2 Характеристики системи опрацювання даних для керування бібліотечним фондом

Система опрацювання даних для керування бібліотечним фондом характеризується здатністю забезпечувати повний цикл роботи з інформацією про книги, читачів та рух примірників. Її функціонування побудоване так, щоб усі етапи взаємодії з бібліотечним фондом здійснювалися послідовно, узгоджено та без зайвих затримок. У програмі реалізовано механізми збереження, редагування та обробки структурованих даних, завдяки чому підтримується актуальний стан картотеки та інформації про користувачів. До основних характеристик належить стабільність виконання операцій, безпечність доступу до даних та чітке розмежування ролей між окремими логічними складовими.

У системі передбачено можливість швидкого пошуку та перегляду даних, що дозволяє оперативно отримувати інформацію про будь-яку книгу або читача. Важливою властивістю є автоматизоване оновлення стану примірників під час їх видачі або повернення, що дає змогу підтримувати коректність записів без додаткових ручних дій. Усі операції обробляються таким чином, щоб виключити дублювання даних і зменшити ймовірність помилок під час внесення змін.

Система здатна формувати службові документи на основі накопиченої інформації, генеруючи звітні матеріали у стандартизованому вигляді. Водночас інтерфейс забезпечує інтуїтивну взаємодію користувача з усіма модулями, що робить роботу передбачуваною й зручною навіть для осіб, які не мають глибокого технічного досвіду.

Архітектура системи опрацювання даних для керування бібліотечним фондом дає змогу легко масштабувати його під розширений книжковий фонд або збільшення кількості користувачів. Усі компоненти узгоджено працюють із базою даних, що гарантує цілісність інформації та безперервність доступу до неї. Завдяки такому підходу система поєднує простоту використання,

функціональну повноту та надійність, необхідні для ефективної організації бібліотечних процесів.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску системи опрацювання даних для керування бібліотечним фондом необхідно запустити відповідний ру-файл.

4.3.2 Вхідні й вихідні дані

Вхідними даними до системи опрацювання даних для керування бібліотечним фондом є інформація, що надходить від користувача під час роботи з інтерфейсом. До неї належать відомості про читачів, зміни у вже внесених записах, дані про книги та їхні примірники, а також параметри, пов'язані з видачею чи поверненням примірників. Усі ці відомості надходять у структурованому вигляді та одразу опрацьовуються системою для подальшого збереження або оновлення існуючих записів.

Вихідними даними системи опрацювання даних для керування бібліотечним фондом є результати виконання дій користувача, що відображають оновлений стан бази та надають потрібні відомості для подальшої роботи. До таких результатів належать сформовані таблиці про читачів і книги, інформація про доступність примірників, відображення історії видач, уточнені дані про повернення, а також автоматично згенеровані звітні матеріали, що містять детальну характеристику випадків прострочення або інших важливих подій. Усі вихідні дані подаються у зрозумілому та впорядкованому вигляді, що дозволяє користувачеві швидко орієнтуватися в інформації.

4.3.3 Повідомлення

Користувач системи опрацювання даних для керування бібліотечним фондом може отримати такі повідомлення: повідомлення про успішне внесення нових записів, підтвердження збереження змін, сигнали про помилки під час введення даних, попередження про відсутність доступних примірників, інформацію про те, що вибрана книга перебуває в користуванні, а також сповіщення про необхідність повернення примірників.

4.4 Виконання системи опрацювання даних для керування бібліотечним фондом

Після запуску системи опрацювання даних для керування бібліотечним фондом користувачу відображається головна сторінка (рис. 4.1).

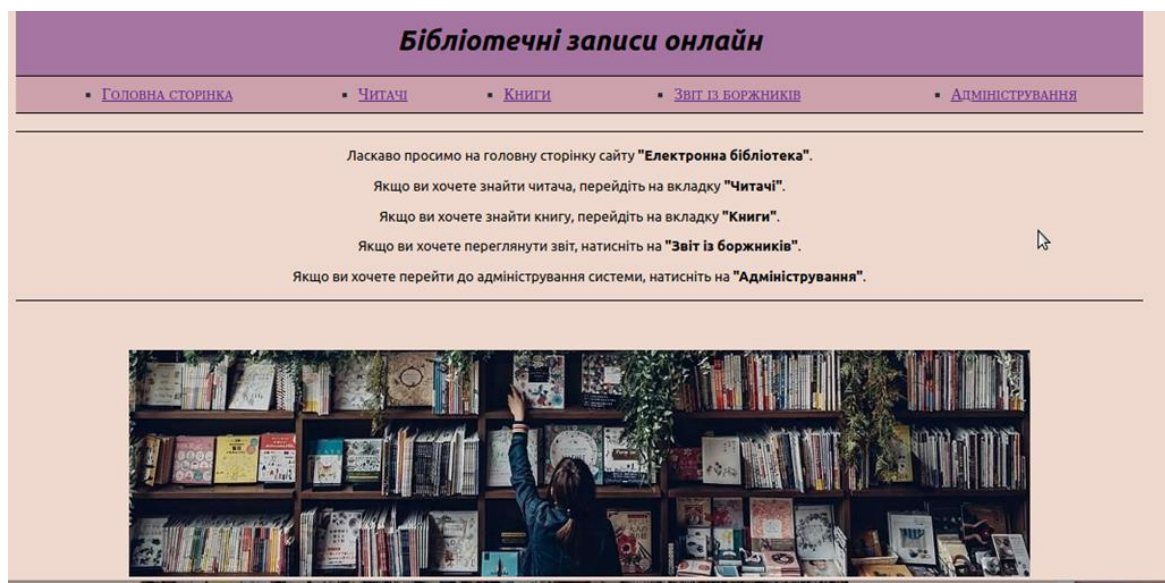


Рисунок 4.1 – Інтерфейс головної форми системи опрацювання даних для керування бібліотечним фондом

При переході за посиланням Читачі відображається сторінка з переліком користувачів бібліотечного фонду (рис. 4.2).

Бібліотечні записи онлайн	
▪ Головна сторінка ▪ Читачі ▪ Книги ▪ Звіт із боржників ▪ Адміністрування	
Читач	Телефон
▪ Іваненко Володимир Іванович	+380509997788
▪ Казаченко Ігор Андрійович	+380931112233
▪ Бедан Роман Петрович	+380931112234
▪ Василенко Жанна Володимирівна	+380938882235
▪ Лазар Петро Васильович	+380934442235
▪ Олексієнко Ігор Романович	+380933332235

Рисунок 4.2 – Сторінка з користувачами бібліотеки

Для отримання інформації про книжки необхідно перейти за посиланням Книги та обрати потрібну книжку (рис. 4.3), внаслідок чого відобразиться інформація про елементи бібліотечного фонду (зокрема, коди книжок), які відповідають обраній книзі.

Бібліотечні записи онлайн	
▪ Головна сторінка ▪ Читачі ▪ Книги ▪ Звіт із боржників ▪ Адміністрування	
Гаррі Поттер і філософський камінь Дж. К. Роулінг	
Код книги	У наявності
▪ A5420	-
▪ A5421	-
▪ A5422	-

Рисунок 4.3 – Інформація про елементи бібліотечного фонду

При виборі елементу бібліотечного фонду, який є вільним, відкривається список читачів, які можуть отримати відповідну книжку (рис. 4.4). Після вибору конкретного читача та натискання кнопки Віддати у базі даних програми буде зроблено відповідну відмітку про те, що факт видачі книжки відбувся.

Бібліотечні записи онлайн

Головна сторінка

Читачі

Книги

Звіт із боржників

Адміністрування

Гаррі Поттер і філософський камінь - Дж. К. Роулінг

A5421

Читач:

Казаченко Ігор Андрійович

Бедан Роман Петрович

Василенко Жанна Володимирівна

Лазар Петро Васильович

Олексієнко Ігор Романович

Віддати

Рисунок 4.4 – Форма видачі книжки читачу

Для оформлення повернення книжки необхідно перейти за відповідним посиланням та натиснути кнопку Повернути (рис. 4.5).

Бібліотечні записи онлайн

Головна сторінка

Читачі

Книги

Звіт із боржників

Адміністрування

Гаррі Поттер і філософський камінь - Дж. К. Роулінг

A5422

Читач

Узято

Олексієнко Ігор Романович

26 травня 2026 р.

Повернути

Рисунок 4.5 – Форма повернення книги

При виборі пункту меню Звіт із боржників відкривається сторінка зі списком осіб, які не повернули книги (рис. 4.6). У відповідному списку наведено інформацію про читача, зокрема, його прізвище та телефон, а також інформацію про книгу та дату її видачі.

Відомості електронної бібліотеки

Боржники

Читач	Телефон	Книга	Узято
Олексієнко Ігор Романович	+380933332235	Гаррі Потер і філософський камінь (A5422)	03.03.2026
Бедан Роман Петрович	+380931112234	Історія державності (I7325)	05.03.2026

Сторінка 1 з 1

Рисунок 4.6 – Сторінка звіту про осіб, які не повернули взяті книги

4.5 Тестування системи опрацювання даних для керування бібліотечним фондом

Проведено тестування системи опрацювання даних для керування бібліотечним фондом.

Підтверджено, що система опрацювання даних для керування бібліотечним фондом працює коректно і стабільно, виконуючи всі необхідні функціональні вимоги та успішно справляючись із основною задачею підтримки процесу функціонування бібліотеки.

Розроблена система забезпечує організацію та опрацювання даних, пов’язаних із підтримкою процесів керування бібліотечним фондом.

4.6 Висновки за розділом 4

Описано систему опрацювання даних для керування бібліотечним фондом.

Проведено тестування створеної системи опрацювання даних для керування бібліотечним фондом. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов'язаних із підтримкою процесів керування бібліотечним фондом.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси обчислень, пов'язані з організацією та опрацюванням даних для керування бібліотечним фондом.

Визначено, що системи опрацювання даних для керування бібліотечним фондом являють собою інтегровані програмні рішення, призначені для впорядкування роботи з інформаційними ресурсами, що зберігаються у бібліотеках та споріднених установах. Їхнім основним призначенням є забезпечення безперервного контролю за рухом фонду, підтримання точності бібліографічних описів, підвищення швидкості пошуку матеріалів, а також оптимізація взаємодії між працівниками та користувачами бібліотеки.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато систем опрацювання даних для керування бібліотечним фондом. Проте деякі системи опрацювання даних для керування бібліотечним фондом для створення та впровадження потребують значних матеріальних і часових ресурсів. Ліцензування програмного забезпечення, придбання обладнання, навчання персоналу та перенесення даних у цифрову форму можуть стати серйозним фінансовим навантаженням для бібліотечні установи. Крім того, системи опрацювання даних для керування бібліотечним фондом вимагають постійного оновлення та технічної підтримки. Без регулярного супроводу вони поступово втрачають ефективність, виникають проблеми з безпекою та сумісністю з новими стандартами. Також можливими є труднощі під час адаптації персоналу, особливо якщо працівники не мають достатнього досвіду роботи з цифровими технологіями. Тому актуальною є розробка системи опрацювання даних для керування бібліотечним фондом.

Сформульовано функціональні вимоги до системи опрацювання даних для керування бібліотечним фондом.

Для реалізації системи опрацювання даних для керування бібліотечним фондом обрано мову програмування Python через поєднання технологічної

гнучкості та зручності реалізації функціональних можливостей. Python відзначається простим і зрозумілим синтаксисом, що дозволяє прискорити розробку та зменшити кількість помилок у коді. Крім того, мова Python має добре розвинену екосистему бібліотек, орієнтованих на роботу з базами даних, аналітику та вебінтерфейси.

Для створення системи опрацювання даних для керування бібліотечним фондом обрано середовище розробки PyCharm, що поєднує інструменти для роботи з базами даних, веброзробки, налагодження та командної взаємодії.

Для зберігання та опрацювання даних обрано систему управління базами даних SQLite через простоту використання, автономність, легку інтеграцію з Python і мінімальні вимоги до технічного супроводу. Інші системи управління базами даних підходять для масштабних або корпоративних систем, але для локальної або невеликих бібліотек вони надто ресурсозатратні та складні в адмініструванні.

Розроблено систему опрацювання даних для керування бібліотечним фондом.

Запропоновано структуру системи опрацювання даних для керування бібліотечним фондом. Визначено, що в основу структури системи опрацювання даних для керування бібліотечним фондом покладено чотири функціональні модулі (модуль адміністративного керування, модуль історії читачів, модуль оформлення видачі та повернення літератури та аналітичний модуль), кожен із яких виконує власну роль, але водночас тісно взаємодіє з іншими складовими, утворюючи єдину інформаційну платформу.

Архітектура системи опрацювання даних для керування бібліотечним фондом організована відповідно до підходу MVC, де кожен елемент виконує чітко визначену роль у взаємодії між інтерфейсом, логікою обробки та сховищем даних, внаслідок чого уся система функціонує як узгоджена структура, у якій Представлення відповідає за взаємодію з користувачами, Контролер керує потоками запитів, а Модель забезпечує роботу з даними. Така організація системи опрацювання даних для керування бібліотечним фондом

дозволяє легко підтримувати та розширювати систему, розмежовує відповідальність між компонентами та забезпечує ефективне керування бібліотечним фондом незалежно від обсягу інформації чи кількості користувачів.

Описано функціонування системи опрацювання даних для керування бібліотечним фондом. Функціонування системи опрацювання даних для керування бібліотечним фондом ґрунтується на послідовній взаємодії окремих механізмів, кожен з яких відповідає за певний сегмент роботи з інформацією. У процесі побудови системи було сформовано низку логічних схем, що охоплюють обробку довідкових даних про книги та читачів, фіксацію історій користування літературою, відстеження руху видань між фондом і читачами, а також генерацію довідкових матеріалів для службової звітності.

Розроблено базу даних системи опрацювання даних для керування бібліотечним фондом, яка побудована таким чином, щоб дані передавали логіку роботи бібліотечного процесу: від загальної інформації про книги й користувачів до детального відстеження шляху кожного окремого примірника. Це забезпечує узгодженість інформації, зручність пошуку та обробки, а також можливість автоматизації багатьох рутинних операцій у системі керування бібліотечним фондом.

Виконано проєктування інтерфейсу взаємодії користувача з системою опрацювання даних для керування бібліотечним фондом.

Проведено тестування створеної системи опрацювання даних для керування бібліотечним фондом. Результати тестування показали, що система забезпечує організацію та опрацювання даних, пов'язаних із підтримкою процесів керування бібліотечним фондом.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What Is A Library Software. URL: <https://www.aeanet.org/what-is-a-library-software/> (date of access: 19.04.2026).
2. The Power of Software Libraries: Definition, Importance, and Examples. URL: <https://mph-intl.com/blog/the-power-of-software-libraries-definition-importance-and-examples/> (date of access: 19.04.2026).
3. The Transformative Role of Open-Source Software in Library Automation. URL: <https://limbd.org/the-transformative-role-of-open-source-software-in-library-automation/> (date of access: 19.04.2026).
4. Open Source Software for Digital Library. URL: <https://open-innovation-projects.org/blog/open-source-software-for-digital-library-a-comprehensive-guide-to-building-and-managing-your-librarys-digital-collections> (date of access: 19.04.2026).
5. Guide to Unlocking the Benefits of Library Automation Software. URL: <https://www.grtech.com/blog/benefits-of-library-automation> (date of access: 19.04.2026).
6. Integrated Library System: Benefits, Features, and Best Solutions. URL: <https://five.co/blog/integrated-library-system/> (date of access: 19.04.2026).
7. Library Management Systems And How Are They Evolving. URL: <https://ischoolmaster.com/blog/library-management-systems-and-its-evolution> (date of access: 19.04.2026).
8. Free Open-Source Library Management Systems for Various Library Types. URL: <https://medevel.com/library-management-systems-lims-1716/> (date of access: 19.04.2026).
9. Best Open Source & Free Library Management Systems. URL: <https://www.techjockey.com/blog/best-open-source-free-library-management-software> (date of access: 19.04.2026).

10. List of the Top 25 Library Management Systems. URL: <https://topbusinesssoftware.com/categories/library-management/> (date of access: 19.04.2026).
11. Top 10 Library Management Software. URL: <https://www.bookrunch.com/top/library/> (date of access: 19.04.2026).
12. Top 10 Library Management Software Tools: Features, Pros, Cons & Comparison. URL: <https://www.devopsschool.com/blog/top-10-library-management-software-tools-in-2025-features-pros-cons-comparison/> (date of access: 19.04.2026).
13. Senayan Library Management System. URL: <https://slims.web.id/web/> (date of access: 19.04.2026).
14. Invenio. URL: <https://inveniosoftware.org/> (date of access: 19.04.2026).
15. Colibris. URL: <https://www.colibris.net/> (date of access: 19.04.2026).
16. Python Packaging User Guide. URL: <https://packaging.python.org/en/latest/> (date of access: 19.04.2026).
17. Python Syntax. URL: <https://www.pythontutorial.net/python-basics/python-syntax/> (date of access: 19.04.2026).
18. PyCharm Goes AI. URL: <https://www.jetbrains.com/pycharm/features/ai/> (date of access: 19.04.2026).
19. PyCharm Community Edition. URL: <https://pycharm-community-edition.en.softonic.com/> (date of access: 19.04.2026).
20. SQLite Documentation. URL: <https://sqlite.org/docs.html> (date of access: 19.04.2026).
21. SQLite Tutorial. URL: <https://www.sqlitetutorial.net/> (date of access: 19.04.2026).

ДОДАТОК А
Технічне завдання

Вступ

Програмна система опрацювання даних може використовуватися для підтримки процесу керування бібліотечним фондом.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Створення програмної системи опрацювання даних для керування бібліотечним фондом», затверджене наказом Національного університету «Запорізька політехніка» № 139 від 07 квітня 2026 р.

А.2 Призначення розробки

Система опрацювання даних призначена для забезпечення керування бібліотечним фондом.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Система опрацювання даних для керування бібліотечним фондом забезпечує виконання таких функцій:

- підтримка відображення головної сторінки застосунку для можливості доступу до основних функцій системи;
- підтримка можливості роботи з користувачами бібліотечного фонду;
- можливість отримання інформації про елементи бібліотечного фонду, зокрема, інформації про те, чи є відповідний елемент у наявності;
- можливість оформлення видачі книжок;
- підтримка можливості оформлення повернення книжок;

– підтримка можливості отримання аналітичної інформації про користувачів бібліотечного фонду, які не повернули взяті книги.

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс системи опрацювання даних для керування бібліотечним фондом повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Система опрацювання даних для керування бібліотечним фондом повинна забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації системи опрацювання даних для керування бібліотечним фондом необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічний засобів

Для роботи системи опрацювання даних для керування бібліотечним фондом потрібні такі технічні та програмні ресурси: апаратна частина має забезпечувати стабільне виконання всіх операцій, пов'язаних з обробкою даних, формуванням звітів та одночасною взаємодією декількох користувачів. Оптимально, щоб комп'ютерна система була оснащена процесором із частотою, яка не опускається нижче рівня, достатнього для швидкого виконання запитів до бази даних і робочої логіки програми. Тактової швидкості сучасного

багатоядерного процесора початкового рівня цілком вистачає для комфортної роботи у звичайному бібліотечному середовищі, де навантаження залишається помірним.

Оперативна пам'ять повинна мати обсяг, що гарантує плавність роботи інтерфейсу та відсутність затримок під час виконання основних операцій, зокрема під час звернень до сховища даних, сортування інформації, побудови звітних документів або обробки таблиць із великим обсягом записів. Мінімальний рекомендований обсяг пам'яті відповідає типовим вимогам настільних застосунків, проте бажано мати запас, який дозволить системі працювати без уповільнень навіть у разі збільшення бази.

Обсяг постійного сховища має бути достатнім для розміщення самої програми, бібліотек, залежностей та файлів бази даних. Оскільки використовується СУБД не вимагає складних інсталяцій або значного простору, вимоги до накопичувача залишаються невисокими, але бажано використовувати твердотільний диск, що скорочує час доступу до файлів і підвищує загальну продуктивність роботи.

Щодо програмної частини, операційна система повинна підтримувати сучасні версії мови програмування Python, а також мати можливість запускати додаткові бібліотеки, необхідні для роботи графічного інтерфейсу, взаємодії з базою даних та створення звітних документів. Середовище виконання має бути сумісним з інструментами розробки та компонентами, що використовуються системою.

A.3.6 Вимоги до маркування і пакування

Система опрацювання даних для керування бібліотечним фондом може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва – «Система опрацювання даних для керування бібліотечним фондом».

A.4 Вхідні дані до роботи

Вхідними даними до системи опрацювання даних для керування бібліотечним фондом є інформація, що надходить від користувача під час роботи з інтерфейсом. До неї належать відомості про читачів, зміни у вже внесених записах, дані про книги та їхні примірники, а також параметри, пов'язані з видачею чи поверненням примірників. Усі ці відомості надходять у структурованому вигляді та одразу опрацьовуються системою для подальшого збереження або оновлення існуючих записів.

Вихідними даними системи опрацювання даних для керування бібліотечним фондом є результати виконання дій користувача, що відображають оновлений стан бази та надають потрібні відомості для подальшої роботи. До таких результатів належать сформовані таблиці про читачів і книги, інформація про доступність примірників, відображення історії видач, уточнені дані про повернення, а також автоматично згенеровані звітні матеріали, що містять детальну характеристику випадків прострочення або інших важливих подій. Усі вихідні дані подаються у зрозумілому та впорядкованому вигляді, що дозволяє користувачеві швидко орієнтуватися в інформації.

ДОДАТОК Б
Фрагмент тексту програми

```

def cnst_db():
    dbpth = os_mod.path.join(os_mod.getcwd(), 'libsdb.db')
    return dbpth

def ntr_db():
    dbpth = cnst_db()
    return dbpth

def stt_db():
    sqldb.create_all()

class RDR(sqldb.Model):
    __tablename__ = 'readers'
    pk = Clm('id', Int, primary_key=True, autoincrement=True)
    flnm = Clm('fullname', Str(200), nullable=False)
    tlphn = Clm('telephone', Str(64), nullable=True)
    rdrbk = rlshp('RDRBK', backref='rdr')

    def t2dct(self):
        rtn = {}
        rtn['pk'] = self.pk
        rtn['flnm'] = self.flnm
        rtn['tlphn'] = self.tlphn
        return rtn

class BK(sqldb.Model):
    __tablename__ = 'books'
    pk = Clm('id', Int, primary_key=True, autoincrement=True)
    nm = Clm('name', Str(200), nullable=False)
    thr = Clm('author', Str(200), nullable=True)
    cps = rlshp('CP', backref='bk')

    def t2dct(self):
        rtn = {}
        rtn['pk'] = self.pk
        rtn['nm'] = self.nm
        rtn['thr'] = self.thr
        return rtn

class CP(sqldb.Model):
    __tablename__ = 'copies'
    pk = Clm('id', Int, primary_key=True, autoincrement=True)
    bk_pk = Clm('book', Int, FK('books.id'))
    cd = Clm('code', Str(64), nullable=False)
    st = Clm('status', Str(32), nullable=False)
    bk_r = rlshp('BK', foreign_keys=[bk_pk])
    rdrbk = rlshp('RDRBK', backref='cp')

```

```

def t2dct(self):
    rtn = {}
    rtn['pk'] = self.pk
    rtn['bk_pk'] = self.bk_pk
    rtn['cd'] = self.cd
    rtn['st'] = self.st
    return rtn

class RDRBK(sqldb.Model):
    __tablename__ = 'reader_book'
    pk = Clm('id', Int, primary_key=True, autoincrement=True)
    rdr_pk = Clm('reader', Int, FK('readers.id'))
    cp_pk = Clm('copy', Int, FK('copies.id'))
    pkd = Clm('pick_date', Str(64), nullable=False)
    rtd = Clm('return_date', Str(64), nullable=True)
    rdr_r = rlshp('RDR', foreign_keys=[rdr_pk])
    cp_r = rlshp('CP', foreign_keys=[cp_pk])

    def t2dct(self):
        rtn = {}
        rtn['pk'] = self.pk
        rtn['rdr_pk'] = self.rdr_pk
        rtn['cp_pk'] = self.cp_pk
        rtn['pkd'] = self.pkd
        rtn['rtd'] = self.rtd
        return rtn

def nw_dt():
    return DT.now().strftime('%Y-%m-%d')

def wks_df():
    return (DT.now() - TD(days=40)).strftime('%Y-%m-%d')

def q_rdr_all():
    rs = RDR.query.all()
    rlst = []
    for r in rs:
        rlst.append(r.t2dct())
    return rlst

def q_bk_all():
    rs = BK.query.all()
    blst = []
    for b in rs:
        blst.append(b.t2dct())
    return blst

```

```

def q_cp_by_bk(bkpk):
    rs = CP.query.filter_by(bk_pk=bkpk).all()
    lst = []
    for c in rs:
        lst.append(c.t2dct())
    return lst

def q_rdrbk_by_rdr(rdrpk):
    rs = RDRBK.query.filter_by(rdr_pk=rdrpk).all()
    lst = []
    for r in rs:
        lst.append(r.t2dct())
    return lst

def crt_rdr(flnm, tlphn):
    nw = RDR(flnm=flnm, tlphn=tlphn)
    sqldb.session.add(nw)
    sqldb.session.commit()
    return nw.t2dct()

def crt_bk(nm, thr):
    nw = BK(nm=nm, thr=thr)
    sqldb.session.add(nw)
    sqldb.session.commit()
    return nw.t2dct()

def crt_cp(bkpk, cd):
    nw = CP(bk_pk=bkpk, cd=cd, st='AVL')
    sqldb.session.add(nw)
    sqldb.session.commit()
    return nw.t2dct()

def crt_rdrbk(rdrpk, cppk):
    dt = nw_dt()
    nw = RDRBK(rdr_pk=rdrpk, cp_pk=cppk, pkd=dt, rtd=None)
    sqldb.session.add(nw)
    cp = CP.query.get(cppk)
    if cp is not None:
        cp.st = 'OUT'
    sqldb.session.commit()
    return nw.t2dct()

def rtrn_rdrbk(rdrbkpk):
    rb = RDRBK.query.get(rdrbkpk)
    if rb is None:
        return None
    rb.rtd = nw_dt()
    cp = CP.query.get(rb.cp_pk)

```

```

    if cp is not None:
        cp.st = 'AVL'
    sqldb.session.commit()
    return rb.t2dct()

def dlt_rdr(rdrpk):
    rdr = RDR.query.get(rdrpk)
    if rdr is None:
        return False
    sqldb.session.delete(rdr)
    sqldb.session.commit()
    return True

def dlt_bk(bkpk):
    bk = BK.query.get(bkpk)
    if bk is None:
        return False
    cps = CP.query.filter_by(bk_pk=bkpk).all()
    for c in cps:
        sqldb.session.delete(c)
    sqldb.session.delete(bk)
    sqldb.session.commit()
    return True

def mdf_rdr(rdrpk, flnm, tlphn):
    rdr = RDR.query.get(rdrpk)
    if rdr is None:
        return None
    rdr.flnm = flnm
    rdr.tlphn = tlphn
    sqldb.session.commit()
    return rdr.t2dct()

def mdf_bk(bkpk, nm, thr):
    bk = BK.query.get(bkpk)
    if bk is None:
        return None
    bk.nm = nm
    bk.thr = thr
    sqldb.session.commit()
    return bk.t2dct()

def mdf_cp(cppk, cd, st):
    cp = CP.query.get(cppk)
    if cp is None:
        return None
    cp.cd = cd
    cp.st = st

```

```

        sqlldb.session.commit()
        return cp.t2dct()

def q_overdue():
    th = wks_df()
    rs = RDRBK.query.filter(RDRBK.rtd == None).filter(RDRBK.pkd
<= th).all()
    lst = []
    for r in rs:
        d = r.t2dct()
        rdr = RDR.query.get(r.rdr_pk)
        cp = CP.query.get(r.cp_pk)
        bk = None
        if cp is not None:
            bk = BK.query.get(cp.bk_pk)
        if rdr is not None:
            d['rdr_flnm'] = rdr.flnm
            d['rdr_tlphn'] = rdr.tlphn
        if cp is not None:
            d['cp_cd'] = cp.cd
        if bk is not None:
            d['bk_nm'] = bk.nm
        lst.append(d)
    return lst

rq = rq_orig
rd = rd_orig
rl = urlfr_orig
rts = rts_orig
sndf = sndf_orig
mkrsp = mkrsp_orig

def htm_hdr(ttl):
    hdr = """
    <!doctype html>
    <html>
    <head>
    <meta charset="utf-8">
    <title>{ttl}</title>
    <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    table { border-collapse: collapse; width: 100%; }
    th, td { border: 1px solid #ccc; padding: 8px; text-align:
left; }
    form { margin: 10px 0; }
    input, select { padding: 6px; margin: 4px; }
    .lnk { margin-right: 10px; }
    </style>

```

```

    </head>
    <body>
    <h2>{ttl}</h2>
    <div><a class="lnk" href="/">Головна</a><a class="lnk"
href="/rdr">Читачі</a><a class="lnk" href="/bk">Книги</a><a
class="lnk" href="/iss">Видача/Повернення</a><a class="lnk"
href="/rpt">Звіт боржників</a></div>
    <hr>
    """.format(ttl=ttl)
    return hdr

def htm_ftr():
    ftr = ""
    <hr>
    <div>Система керування бібліотечним фондом</div>
    </body>
    </html>
    ""
    return ftr

@app.route('/')
def hme():
    rds = q_rdr_all()
    bks = q_bk_all()
    cnt_r = len(rds)
    cnt_b = len(bks)
    bd = htm_hdr('Головна') + f'<div>Кількість читачів:
{cnt_r}</div><div>Кількість книг: {cnt_b}</div>' + htm_ftr()
    return bd

@app.route('/rdr', methods=['GET', 'POST'])
def rdr_lst():
    if rq.method == 'POST':
        nm = rq.form.get('flnm', '').strip()
        tl = rq.form.get('tlphn', '').strip()
        if nm != '':
            crt_rdr(nm, tl)
            return rd(rl('rdr_lst'))
    rds = q_rdr_all()
    tb =
'<table><tr><th>Ім'я</th><th>Телефон</th><th>Дії</th></tr>'
    for r in rds:
        tb +=
f'<tr><td>{r["flnm"]}</td><td>{r["tlphn"]}</td><td><a
href="/rdr/{r["pk"]}">Деталі</a></td></tr>'
        tb += '</table>'
    frm = '<form method="post"><input name="flnm"
placeholder="ПІБ"><input name="tlphn"

```

```

placeholder="Телефон"><input type="submit"
value="Додати"></form>'
    bd = htm_hdr('Читачі') + frm + tb + htm_ftr()
    return bd

@app.route('/rdr/<int:rdrpk>', methods=['GET', 'POST'])
def rdr_dt(rdrpk):
    rdr = RDR.query.get(rdrpk)
    if rdr is None:
        return htm_hdr('Помилка') + '<div>Читача не
знайдено</div>' + htm_ftr()
    if rq.method == 'POST':
        if 'mdf' in rq.form:
            nm = rq.form.get('flnm', '').strip()
            tl = rq.form.get('tlphn', '').strip()
            mdf_rdr(rdrpk, nm, tl)
        if 'dlt' in rq.form:
            dlt_rdr(rdrpk)
            return rd(rl('rdr_lst'))
        if 'rdrbk' in rq.form:
            cp_pk = int(rq.form.get('cp_pk'))
            crt_rdrbk(rdrpk, cp_pk)
            return rd(rl('rdr_dt', rdrpk=rdrpk))
    his = q_rdrbk_by_rdr(rdrpk)
    tbl = '<table><tr><th>Код екз.</th><th>Назва</th><th>Дата
видачі</th><th>Дата повернення</th><th>Дії</th></tr>'
    for h in his:
        cp = CP.query.get(h['cp_pk'])
        bk = None
        if cp is not None:
            bk = BK.query.get(cp.bk_pk)
        cd = cp.cd if cp is not None else ''
        nm = bk.nm if bk is not None else ''
        pkd = h['pkd']
        rtd = h['rtd'] if h['rtd'] is not None else ''
        act = ''
        if h['rtd'] is None:
            act = f'<a href="/rtrn/{h["pk"]}">Підтвердити
повернення</a>'
        tbl +=
f'<tr><td>{cd}</td><td>{nm}</td><td>{pkd}</td><td>{rtd}</td><td>{
act}</td></tr>'
        tbl += '</table>'
    allbks = q_bk_all()
    opt = '<select name="cp_pk">'
    for b in allbks:
        cps = q_cp_by_bk(b['pk'])
        for c in cps:

```

```

        if c['st'] == 'AVL':
            opt += f'<option value="{c["pk"]}">{b["nm"]} /
{c["cd"]}</option>'
            opt += '</select>'
            frm = f'<form method="post"><input name="flnm"
value="{rdr.flnm}"><input name="tlphn"
value="{rdr.tlphn}"><button name="mdf">Змінити</button><button
name="dlt">Видалити</button></form><form
method="post"><div>Видати книгу: {opt}<button
name="rdrbk">Видати</button></div></form>'
            bd = htm_hdr('Карточка читача') +
f'<div><b>{rdr.flnm}</b><div>{rdr.tlphn}</div></div>' + frm + tbl
+ htm_ftr()
            return bd

@app.route('/bk', methods=['GET', 'POST'])
def bk_lst():
    if rq.method == 'POST':
        nm = rq.form.get('nm', '').strip()
        thr = rq.form.get('thr', '').strip()
        if nm != '':
            crt_bk(nm, thr)
            return rd(rl('bk_lst'))
    bks = q_bk_all()
    tb =
'<table><tr><th>Назва</th><th>Автор</th><th>Копії</th><th>Дії</th>
</tr>'
    for b in bks:
        cps = q_cp_by_bk(b['pk'])
        cnt = len(cps)
        tb +=
f'<tr><td>{b["nm"]}</td><td>{b["thr"]}</td><td>{cnt}</td><td><a
href="/bk/{b["pk"]}">Деталі</a></td></tr>'
        tb += '</table>'
        frm = '<form method="post"><input name="nm"
placeholder="Назва"><input name="thr" placeholder="Автор"><input
type="submit" value="Додати"></form>'
        bd = htm_hdr('Книги') + frm + tb + htm_ftr()
        return bd

@app.route('/bk/<int:bkpk>', methods=['GET', 'POST'])
def bk_dt(bkpk):
    bk = BK.query.get(bkpk)
    if bk is None:
        return htm_hdr('Помилка') + '<div>Книгу не
знайдено</div>' + htm_ftr()
    if rq.method == 'POST':
        if 'mdf' in rq.form:

```

```

        nm = rq.form.get('nm', '').strip()
        thr = rq.form.get('thr', '').strip()
        mdf_bk(bkpk, nm, thr)
    if 'dlt' in rq.form:
        dlt_bk(bkpk)
        return rd(rl('bk_lst'))
    if 'crt_cp' in rq.form:
        cd = rq.form.get('cd', '').strip()
        if cd != '':
            crt_cp(bkpk, cd)
    if 'mdf_cp' in rq.form:
        cppk = int(rq.form.get('cppk'))
        cd = rq.form.get('cd_val', '')
        st = rq.form.get('st_val', '')
        mdf_cp(cppk, cd, st)
    return rd(rl('bk_dt', bkpk=bkpk))
cps = q_cp_by_bk(bkpk)
tbl =
'<table><tr><th>Код</th><th>Статус</th><th>Дії</th></tr>'
    for c in cps:
        tbl +=
f'<tr><td>{c["cd"]}</td><td>{c["st"]}</td><td><form method="post"
style="display:inline;"><input type="hidden" name="cppk"
value="{c["pk"]}"><input name="cd_val" value="{c["cd"]}"><select
name="st_val"><option value="AVL">AVL</option><option
value="OUT">OUT</option></select><button
name="mdf_cp">Змінити</button></form></td></tr>'
        tbl += '</table>'
        frm = f'<form method="post"><input name="nm"
value="{bk.nm}"><input name="thr" value="{bk.thr}"><button
name="mdf">Змінити</button><button
name="dlt">Видалити</button></form><form method="post"><input
name="cd" placeholder="Код екземпляру"><button
name="crt_cp">Додати екземпляр</button></form>'
        bd = htm_hdr('Картка книги') +
f'<div><b>{bk.nm}</b></div><div>{bk.thr}</div></div>' + frm + tbl +
htm_ftr()
    return bd

@app.route('/iss', methods=['GET', 'POST'])
def iss_lst():
    if rq.method == 'POST':
        if 'iss' in rq.form:
            rdrpk = int(rq.form.get('rdrpk'))
            cppk = int(rq.form.get('cppk'))
            crt_rdrbk(rdrpk, cppk)
        return rd(rl('iss_lst'))
    bks = q_bk_all()

```

```

rds = q_rdr_all()
sel_bk = '<select id="bkssel" name="bkpk"
onchange="this.form.submit()">'
sel_bk += '<option value="">Оберіть книгу</option>'
for b in bks:
    sel_bk += f'<option value="{b["pk"]}">{b["nm"]}</option>'
sel_bk += '</select>'
form = f'<form method="post">Оберіть книгу: {sel_bk}</form>'
bd = htm_hdr('Видача/Повернення') + form
bd += '<hr><div>Оберіть читача і екземпляр для
видачі</div><form method="post"><select name="rdrpk">'
for r in rds:
    bd += f'<option value="{r["pk"]}">{r["flnm"]}</option>'
bd += '</select><select name="cppk">'
ava_cps = CP.query.filter_by(st='AVL').all()
for c in ava_cps:
    bk = BK.query.get(c.bk_pk)
    nm = bk.nm if bk is not None else ''
    bd += f'<option value="{c.pk}">{nm} / {c.cd}</option>'
bd += '</select><button name="iss">Видати</button></form>' +
htm_ftr()
return bd

```

```

@app.route('/rtrn/<int:rdrbkpk>', methods=['GET'])
def rtrn(rdrbkpk):
    rtrn_rdrbk(rdrbkpk)
    return rd(rl('hme'))

```

```

@app.route('/rpt', methods=['GET'])
def rpt_lst():
    lst = q_overdue()
    buf = io_mod.BytesIO()
    c = rc.Canvas(buf, pagesize=A4)
    x = 50
    y = 800
    c.setFont("Helvetica-Bold", 12)
    c.drawString(x, y, "Звіт боржників")
    y -= 30
    c.setFont("Helvetica", 10)
    for it in lst:
        lines = []
        nm = it.get('rdr_flnm', '')
        tl = it.get('rdr_tlphn', '')
        bk = it.get('bk_nm', '')
        cd = it.get('cp_cd', '')
        pkd = it.get('pkd', '')
        ln = f'Читач: {nm} Телефон: {tl}'
        c.drawString(x, y, ln)

```

```

        y -= 15
        ln2 = f'Книга: {bk} Код: {cd} Дата видачі: {pkd}'
        c.drawString(x+20, y, ln2)
        y -= 25
        if y < 120:
            c.showPage()
            y = 800
        c.save()
        buf.seek(0)
        rsp = mkrsp(buf.read())
        rsp.headers['Content-Type'] = 'application/pdf'
        rsp.headers['Content-Disposition'] = 'attachment;
filename=bzht.pdf'
        return rsp

def ld_cps(cn, bk_id):
    cr = cn.cursor()
    r = cr.execute("SELECT * FROM cp WHERE bk_id=?",
(bk_id,)).fetchall()
    return r

def ld_rdr_hstr(cn, rdr_id):
    cr = cn.cursor()
    r = cr.execute(
        "SELECT cp.cd, rd_bk.pk_dt, rd_bk.rt_dt FROM rd_bk JOIN
cp ON cp.id = rd_bk.cp_id WHERE rd_bk.rdr_id=?",
        (rdr_id,))
    ).fetchall()
    return r

def mk_pk(cn, rdr_id, cp_id, pk_dt):
    cr = cn.cursor()
    cr.execute("INSERT INTO rd_bk (rdr_id, cp_id, pk_dt, rt_dt)
VALUES (?, ?, ?, NULL)", (rdr_id, cp_id, pk_dt))
    cr.execute("UPDATE cp SET avl=0 WHERE id=?", (cp_id,))
    cn.commit()

def mk_rt(cn, cp_id, rt_dt):
    cr = cn.cursor()
    cr.execute("UPDATE rd_bk SET rt_dt=? WHERE cp_id=? AND rt_dt
IS NULL", (rt_dt, cp_id))
    cr.execute("UPDATE cp SET avl=1 WHERE id=?", (cp_id,))
    cn.commit()

def ld_brwr(cn):
    cr = cn.cursor()
    r = cr.execute(

```

```

        "SELECT rdr.fn, rdr.tl, bk.nm, cp.cd, rd_bk.pk_dt FROM
rd_bk "
        "JOIN rdr ON rd_bk.rdr_id=rdr.id "
        "JOIN cp ON rd_bk.cp_id=cp.id "
        "JOIN bk ON cp.bk_id=bk.id "
        "WHERE rd_bk.rt_dt IS NULL"
    ).fetchall()
    return r

def hndl_nw_rdr(app, cn):
    @app.route('/nw_rdr', methods=['POST'])
    def _nw_rdr():
        fn = request.form.get("fn")
        tl = request.form.get("tl")
        nst_rdr(cn, fn, tl)
        return jsonify({"st": "rdr_nstd"})

def hndl_nw_bk(app, cn):
    @app.route('/nw_bk', methods=['POST'])
    def _nw_bk():
        nm = request.form.get("nm")
        athr = request.form.get("athr")
        nst_bk(cn, nm, athr)
        return jsonify({"st": "bk_nstd"})

def hndl_nw_cp(app, cn):
    @app.route('/nw_cp', methods=['POST'])
    def _nw_cp():
        bk_id = request.form.get("bk_id")
        cd = request.form.get("cd")
        avl = request.form.get("avl")
        nst_cp(cn, bk_id, cd, avl)
        return jsonify({"st": "cp_nstd"})

def hndl_pk(app, cn):
    @app.route('/mk_pk', methods=['POST'])
    def _mk_pk():
        rdr_id = request.form.get("rdr_id")
        cp_id = request.form.get("cp_id")
        pk_dt = request.form.get("pk_dt")
        mk_pk(cn, rdr_id, cp_id, pk_dt)
        return jsonify({"st": "pk_mkd"})

def hndl_rt(app, cn):
    @app.route('/mk_rt', methods=['POST'])
    def _mk_rt():
        cp_id = request.form.get("cp_id")
        rt_dt = request.form.get("rt_dt")

```

```

        mk_rt(cn, cp_id, rt_dt)
        return jsonify({"st": "rt_mkd"})

def hndl_rdrs(app, cn):
    @app.route('/rdrs')
    def _rdrs():
        r = ld_rdrs(cn)
        return jsonify(r)

def hndl_bks(app, cn):
    @app.route('/bks')
    def _bks():
        r = ld_bks(cn)
        return jsonify(r)

def hndl_cps(app, cn):
    @app.route('/cps')
    def _cps():
        bk_id = request.args.get("bk_id")
        r = ld_cps(cn, bk_id)
        return jsonify(r)

def hndl_rdr_hstr(app, cn):
    @app.route('/rdr_hstr')
    def _rdr_hstr():
        rdr_id = request.args.get("rdr_id")
        r = ld_rdr_hstr(cn, rdr_id)
        return jsonify(r)

def hndl_brwr(app, cn):
    @app.route('/brwr')
    def _brwr():
        r = ld_brwr(cn)
        return jsonify(r)

```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Створення програмної системи опрацювання даних для керування бібліотечним фондом

Нікіта ГЛАЗУНОВ,

ст. групи КНТ-212

керівник Тетяна КОЛПАКОВА,

к.т.н., доцент

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

- **Об'єкт дослідження** – процеси обчислень, пов'язані з організацією та опрацюванням даних для керування бібліотечним фондом.
- **Предмет дослідження** – програмні системи опрацювання даних для керування бібліотечним фондом.
- **Метою роботи є** розробка програмної системи опрацювання даних для керування бібліотечним фондом для підвищення ефективності діяльності бібліотеки.

2

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі бакалавра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та систем опрацювання даних для керування бібліотечним фондом;
- здійснити проєктування системи опрацювання даних для керування бібліотечним фондом;
- створити систему опрацювання даних для керування бібліотечним фондом;
- дослідити розроблену систему опрацювання даних для керування бібліотечним фондом.

3

Рисунок В.3 – Слайд 3

Порівняння існуючих аналогів

Критерій порівняння	SLiMS	Invenio	Colibris
Простота встановлення	+	–	+
Підтримка цифрових репозиторіїв	+–	+	–
Можливість розширення через модулі	+–	+	+–
Активність спільноти	+	+	+–
Інтеграція з іншими сервісами	+–	+	+–
Локалізація	+	+–	+–

4

Рисунок В.4 – Слайд 4

Порівняння мов програмування

Критерій порівняння мов програмування	Мова програмування		
	Python	Java	C++
Швидкість реалізації функціоналу	+	+-	+-
Робота з базами даних	+	+	+-
Вартість та доступність інструментів	+	+-	+
Простота розробки	+	+-	-
Спільнота та підтримка	+	+-	+-

5

Рисунок В.5 – Слайд 5

Обґрунтування вибору середовища розробки

Критерій порівняння середовищ розробки	Середовища розробки		
	PyCharm	Visual Studio Code	IDLE
Автоматичне доповнення коду	+	+	-
Розвинені інструменти рефакторингу	+	+-	-
Інтеграція з базами даних	+	+-	+-
Підтримка вебфреймворків	+	+-	+-
Наявність професійних засобів налагодження	+	+	-

6

Рисунок В.6 – Слайд 6

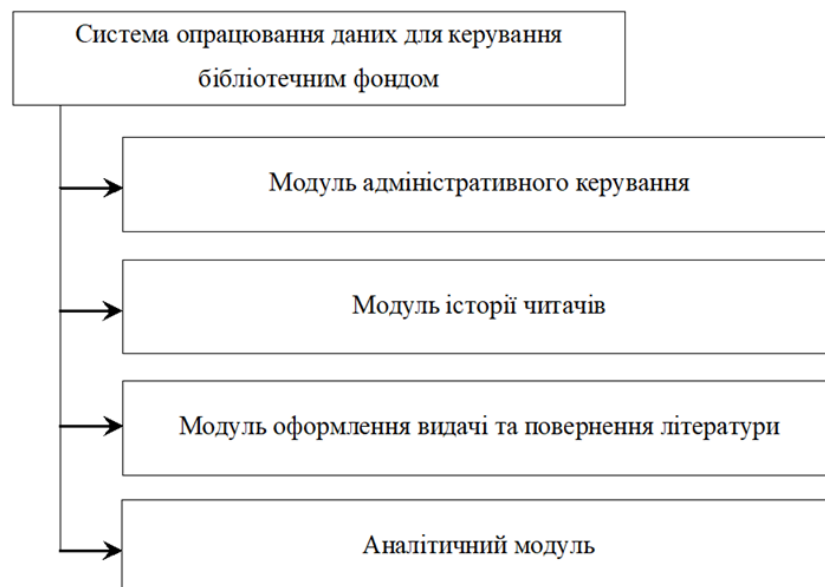
Обґрунтування вибору засобів зберігання та опрацювання даних

Критерій порівняння систем управління базами даних	Системи управління базами даних		
	SQLite	PostgreSQL	Oracle DB
Легка інтеграція з Python	+	+	+-
Можливість автономної роботи	+	+-	-
Низька вартість використання	+	+	-
Продуктивність для невеликих та середніх баз	+	+-	+-
Мобільність та переносимість даних	+	+-	-

7

Рисунок В.7 – Слайд 7

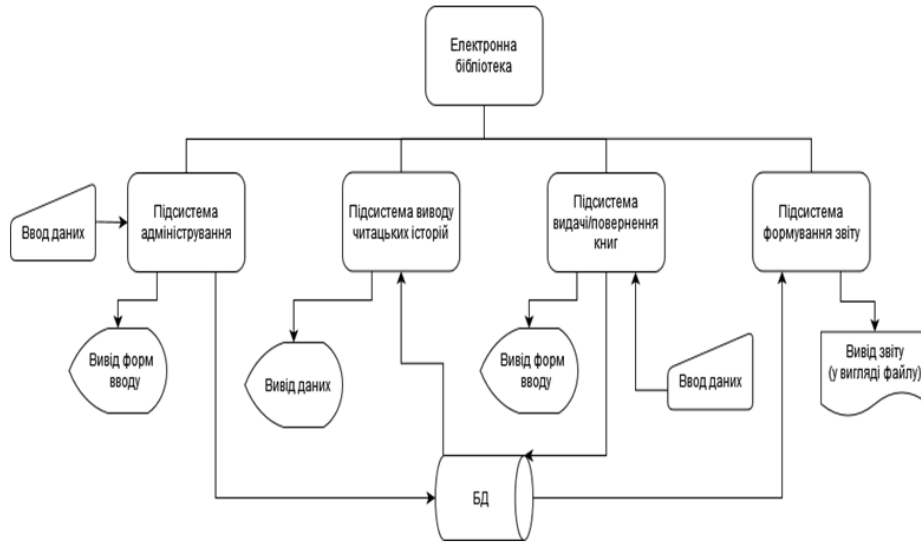
Структура системи опрацювання даних для керування бібліотечним фондом



8

Рисунок В.8 – Слайд 8

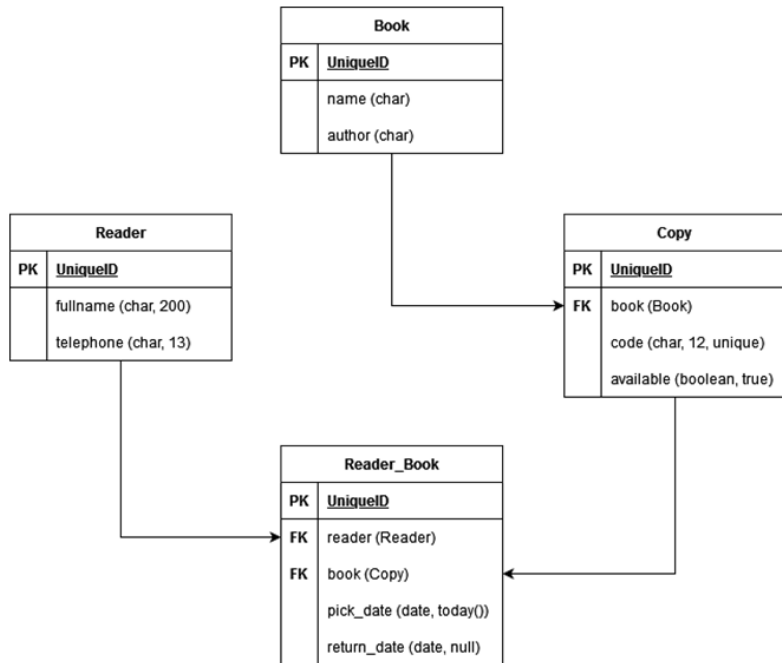
Схема функціонування СОД для керування бібліотечним фондом



9

Рисунок В.9 – Слайд 9

Схема бази даних



10

Рисунок В.10 – Слайд 10

Інтерфейс програми. Головна сторінка. Сторінка користувачів бібліотеки



Бібліотечні записи онлайн	
Головна сторінка • Читачи • Книги • Звіт із боржників • Адміністрування	
Читач	Телефон
Іваненко Володимир Іванович	+380509997788
Казаченко Ігор Андрійович	+380931112233
Бедан Роман Петрович	+380931112234
Василенко Жанна Володимирівна	+380938882235
Лазар Петро Васильович	+380934442235
Олексієнко Ігор Романович	+380933332235

11

Рисунок В.11 – Слайд 11

Інтерфейс застосунку. Робота з видачею та повернення книжок

Бібліотечні записи онлайн

Головна сторінка • Читачи • Книги • Звіт із боржників • Адміністрування

Гаррі Поттер і філософський камінь
Дж. К. Роулінг

Код книги	У наявності
A5420	-
A5421	-
A5422	-

Бібліотечні записи онлайн

Головна сторінка • Читачи • Книги • Звіт із боржників • Адміністрування

Гаррі Поттер і філософський камінь - Дж. К. Роулінг
A5421

Читач:
Казаченко Ігор Андрійович
Бедан Роман Петрович
Василенко Жанна Володимирівна
Лазар Петро Васильович
Олексієнко Ігор Романович

Відати

Бібліотечні записи онлайн

Головна сторінка • Читачи • Книги • Звіт із боржників • Адміністрування

Гаррі Поттер і філософський камінь - Дж. К. Роулінг
A5422

Читач	Узято
Олексієнко Ігор Романович	26 травня 2026 р.

Повернути

Відомості електронної бібліотеки

Боржники

Читач	Телефон	Книга	Узято
Олексієнко Ігор Романович	+380933332235	Гаррі Поттер і філософський камінь (A5422)	03.03.2026
Бедан Роман Петрович	+380931112234	Історія державності (I7325)	05.03.2026

Сторінка 1 з 1

12

Рисунок В.12 – Слайд 12

Висновки

В ході виконання дипломної кваліфікаційної роботи бакалавра було проаналізовано та досліджено процеси обчислень, пов'язані з організацією та опрацюванням даних для керування бібліотечним фондом.

Сформульовано функціональні вимоги до системи опрацювання даних для керування бібліотечним фондом. Для реалізації системи опрацювання даних для керування бібліотечним фондом обрано мову програмування Python, середовище розробки PyCharm та систему управління базами даних SQLite.

Розроблено систему опрацювання даних для керування бібліотечним фондом. Архітектура програми організована відповідно до підходу MVC. Функціонування системи опрацювання даних для керування бібліотечним фондом ґрунтується на послідовній взаємодії окремих модулів, кожен з яких відповідає за певний сегмент роботи з інформацією.

Проведено тестування створеної системи опрацювання даних для керування бібліотечним фондом. Усі завдання випускної дипломної роботи бакалавра повністю виконано.

13

Рисунок В.13 – Слайд 13