

Детектор Плагиата v. 2867 - Отчёт оригинальности: 17.06.2025 14:06:41

Проанализированный документ: бакалаврська ЗАЛЕСЬКИЙ О.В (1).doc Лицензия:
ВОЛОДИМИР МАТІЄВСЬКИЙ

? Тип поиска: Поиск переписанного ? Язык: Uk

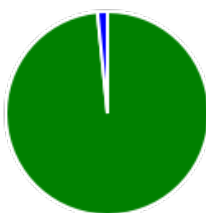
? Тип проверки: Интернет

ТЕЕ и кодировка: ifilter n/a

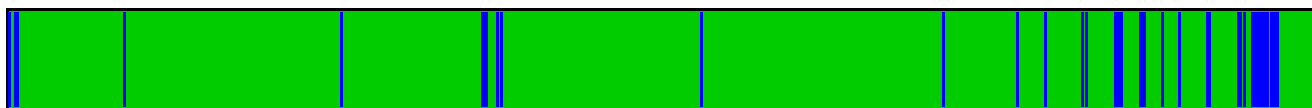
Детальный анализ тела документа:

? Диаграмма соотношения частей:

Плагат 0% Оригинал 98.43%
Кавычки 1.57% ИИ 0%



? Граф распределения зон:



? Источники плагиата: 10

- | | | | |
|------|-------|----|--|
| 0.1% | A B C | 10 | 1. https://web-crawler.plagiarism-detector.com/get-doc-pt?did=8EiiP4y7-zZBng |
| 0.1% | A B C | 10 | 2. https://dizz.in.ua/uk/proczes-rozrobki-interfejsu-koristuvacha-krok-za-krokom/ |
| 0.1% | A B C | 10 | 3. https://flexi-project.com/uk/маніфест-agile-ключові-цінності-та-принци/ |

? Детали обработанных ресурсов: 214 - ОК / 7 - Ошибка

? Важные замечания:

Википедия:	Google Книги:	Сервисы платных работ:	Античит:
[не обнаружено]	[не обнаружено]	[не обнаружено]	Обнаружено сокращение!

? Античит-отчет UACE:

- Статус: Анализатор **Включен** Нормализатор **Включен** сходство символов установлено на **100%**
- Обнаруженный процент загрязнения UniCode: **8,2%** с лимитом: 4%
- Процент нераспознанных символов после нормализации: **4,7%**
- Все подозрительные символы будут отмечены фиолетовым цветом: **Abcd...**
- Найдены невидимые символы: 0

Рекомендации по оценке:

Особое внимание следует уделить анализу этого отчета! Предполагается, что этот документ содержит значительное количество символов, чуждых языку документа. Это прямое указание на то, что автор документа использовал специальное программное обеспечение\онлайн-веб-сервис, чтобы эффективно

скрыть текст в попытке избежать обнаружения потенциального плагиата. Настоятельно рекомендуется передать это дело на более высокий уровень! В случае сомнений обращайтесь: в службу поддержки Детектора плагиата!

Алфавитная статистика и анализ символов:

🔗 Активные ссылки (URL-адреса, извлеченные из документа):

URL не найдены

🔗 Исключённые ресурсы:

URL не найдены

🔗 Включённые ресурсы:

URL не найдены

-МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЗ

Цитирования: **0,06%**

id: 1

"ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА"

Навчально-науковий інститут математики та інформаційних технологій
(назва факультету, інституту)
Кафедра інформаційних технологій та систем
(назва кафедри)

-
Пояснювальна записка до кваліфікаційної роботи
за першим (бакалаврським) рівнем освіти

на тему:-

-Проектування та розробка пошукового додатку

Цитирования: **0,01%**

id: 2

"Бібліотека"

з системою рекомендацій книг-

Виконав: здобувач вищої освіти 4 курсу спеціальності-

F2

Цитирования: **0,03%**

id: 3

"Інженерія програмного забезпечення"

(шифр і назва напрямку підготовки, спеціальності)

_____ Олександр ЗАЛЕСЬКИЙ

(прізвище та ініціали)

Керівник

_____ Микола СЕМЕНОВ

(прізвище та ініціали)

Рецензент

_____ Юрій КОЗУБ

(прізвище та ініціали)

Полтава - 2025-

-
ЗМІСТ-
ВСТУП2

1. РОЗДІЛ 1. ХАРАКТЕРИСТИКА СИСТЕМИ, ТА АНАЛІЗ ВИМОГ4

1.1 Призначення , область застосування4

1.2 Визначення цілей5

1.3 Аналіз Вимог8

1.4 Вимоги до інтерфейсу користувача9

1.5 Вимоги до апаратних, програмних та комунікаційних інтерфейсів11

1-.6 Вимоги до адаптації на місці11

1.7 Функції продукту	12
1.8 Обмеження	14
2. РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ ПРОГРАМНИХ СИСТЕМ	15
2.1. Огляд існуючих методів та основних принципів проєктування, архітектури програмного забезпечення, інженерних підходів і принципів керування ПЗ	15
2.2 Обґрунтування вибору методів та підходів до проєктування ПЗ	26
3. РОЗДІЛ 3. ОПИС ТА ОБГРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ ПО ПРОЄКТУВАННЮ ПРОГРАМНОЇ СИСТЕМИ	32
3.1 Структурні схеми системи (модулі системи та компоненти)	32
3.2 -Опис Архітектурної системи	36
3.3 Структура бази даних , моделювання бази даних (інфологічна модель)	37
4. РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ (РОЗРОБКА БАЗИ ДАНИХ БІБЛІОТЕКИ)	41
4.1 Розробка бази даних програми	42
4.1.1 Концептуальне проєктування бази даних	42-
4.1.2 Логічне проєктування баз даних	43
4.1.3 Фізичне проєктування бази даних	44
4.2 Проєктування структури програми та паттерни	45
4.3 Опис класів та методів застосування	48
4.4 Опис SQL-запитів до бази даних системи	49
5. РОЗДІЛ 5. ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ	54
5.1 Апаратні та програмні засоби створення та експлуатації програми	54
5.2 Інструкція користувача	54
5.3 Опис контрольних прикладів	55
ВИСНОВОК	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	59
ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ (Окремий файл)	60
ДОДАТОК Б ЕКРАННІ ФОРМИ	61
ДОДАТОК В Фрагменти Лістингу	65

ВСТУП

- У сучасному цифровому епохальному світі, де доступ до інформації є ключовим аспектом розвитку, онлайн бібліотеки стали важливою складовою навчального та професійного простору. Проєктування та створення програмного забезпечення для онлайн бібліотек відкриває безмежні можливості для надання широкого спектру літературних творів, наукових публікацій та ресурсів у доступній та зручній формі.

Мета даної бакалаврської роботи полягає у вивченні та аналізі процесу проєктування програмного забезпечення для онлайн бібліотеки. Розглянувши теоретичні аспекти проєктування програм, а також проведення аналізу вимог, архітектури системи, процесу розробки, тестування та управління якістю, мета роботи - надати інсайти та рекомендації для створення ефективної та функціональної онлайн бібліотеки.

Завдання включають аналіз вимог, визначення технічного завдання, вибір оптимальних методів проєктування, огляд архітектури та реалізацію функціоналу.

Аналіз потреб користувачів: Вивчення вимог та очікувань цільової аудиторії від онлайн бібліотеки, врахування їхніх потреб для розробки користувацького інтерфейсу.

-Проєктування функціональності: Визначення основних функцій та можливостей системи, створення сценаріїв взаємодії користувача з додатком.

Вибір технологій: Дослідження та вибір оптимальних технологій для реалізації функціональності додатку, зокрема баз даних, мов програмування та інтерфейсних рішень.

Розробка та втілення концепції: Створення прототипу онлайн бібліотеки, розробка програмного забезпечення, тестування та впровадження системи.

Оцінка ефективності: Проведення оцінки функціональності, користувацької зручності та загальної продуктивності онлайн бібліотеки.

Цілі- які ставляться на бакалаврський проєкт :

-Розробка функціонального та ефективного онлайн інструменту для доступу до літературних ресурсів.

Дослідження та застосування сучасних методів проектування та програмування для створення продуктивної системи.

Забезпечення зручного та інтуїтивно зрозумілого -інтерфейсу для користувачів.

Підвищення доступності до навчальних матеріалів та літературних творів за допомогою технологій.

-У контексті навчання дослідництва та практичного застосування технологій, створення онлайн бібліотек є важливим етапом у забезпеченні загального доступу до знань та інформації для широкого кола користувачів. Відповідно, ця робота буде зосереджена на висвітленні ключових аспектів проектування програмного забезпечення для створення ефективної та легко доступної онлайн бібліотеки.

Зрозуміння процесу створення онлайн бібліотеки та факторів, які впливають на їхню функціональність та успішну роботу, відіграє ключ-ову роль у розвитку сучасної освітньої та інформаційної інфраструктури. Результати даного дослідження можуть стати важливим внеском у сферу програмування та розробки онлайн інструментів для доступу до знань.

Для досягнення поставленої мети та вирішення зав-дань будуть використані методи дослідження, що включають аналіз вимог користувачів, дослідження існуючих систем онлайн бібліотек, вибір оптимальних технологій для реалізації програмного продукту та оцінка його функціональності та продуктивності.

Описана ба-калаврська робота включає в себе ретельний аналіз та систематизацію знань про процес проектування програмного забезпечення для створення онлайн бібліотек, що може стати корисним для фахівців у галузі ІТ, студентів технічних спеціальностей та всіх зацікавле-них у цій темі.

1. РОЗДІЛ 1. ХАРАКТЕРИСТИКА СИСТЕМИ, ТА АНАЛІЗ ВИМОГ

1.1 Призначення , область застосування

-

Бібліотеки залишаються одними з найбільш поширених місць, де можна отримати різноманітну літературу - книги, довідники та інші видання. Незважаючи на те, що деякі заклади переходять на онлайн-предоставлення книг через хмарні сховища, це поки що не є загальнопоширеним. Принципи отримання, зберігання та видачі книг за всі ці роки практично не змінилися.

Сучасна інтерпретація принципів роботи бібліотек переросла в автоматизовану бібліотечну інформаційну систему (АБІС). Однак, для точного визначення АБІС потрібно уточнити круг професійних завдань. Зокрема, основне завдання полягає в автоматизації традиційних бібліотечних технологій. До складових АБІС відносять реляційну базу даних та програму для взаємодії з цією базою даних [1].

На сьогодні на ринку вже існує -безліч рішень щодо автоматизації роботи бібліотек:

 Цитування: **0,01%** id: 4

"OPAC - Global",

 Цитування: **0,02%** id: 5

"АБІС Лібра"

та інші продукти.

Аналізуючи існуюче програмне забезпечення, можна сказати, що ці програми занадто складні для звичайних користувачів через складний інтерфейс та потребують складної налаштування. Ці рішення не підходять для невеликих бібліотек через зайву функціональність. Крім того, всі програми є платними, а при наявності невеликої кількості даних такі витрати не рентабельні [2].

-Тому необхідно розглянути можливість створення власної системи автоматизації роботи - онлайн- бібліотеки.

Система, над якою проводиться аналіз вимог, є онлайн бібліотекою (програмним забезпеченням), призначеною для надання користувачам доступу до електронних книг та ресурсів через інтернет. Основні мети та області застосування цієї системи включають:-

Надання доступу до літературних ресурсів: Онлайн бібліотека робить доступними книги у електронному форматі для читання та вивчення для широкого кола користувачів.

Зручний пошук та вибір книг: Система надає можливість швидкого пошуку книг за різними критеріями (назва, автор, жанр тощо) і забезпечує зручний інтерфейс для вибору та

відображення деталей книги.

Читання книг онлайн та офлайн: Користувачі можуть читати книги безпосередньо на платформі онлайн або завантажувати їх для читання в офлайн-режимі.

Управління контентом: Система надає можливість користувачам додавати книги до списку вибраного, оцінювати та залишати відгуки про книги.

Персоналізовані рекомендації: На основі історії читання система рекомендує користувачам книги, які можуть їм сподобатися.

О-бласти застосування включають освіту, науку, загальну читацьку аудиторію, студентів, дослідників та будь-яку особу, яка зацікавлена у доступі до різноманітного літературного контенту в електронній формі. Така система сприяє зручності, доступності та широко-му розповсюдженню знань і інформації.

1.2 Визначення цілей

-Надати доступ до бібліотечних ресурсів: Розробити платформу, яка забезпечить користувачам (читачам) можливість переглядати, шукати та отримувати доступ до різноманітних книг, журналів, наукових робіт тощо.

Організувати зручний пошук та навігацію: Створити -ефективні та інтуїтивно зрозумілі інструменти пошуку, які дозволять користувачам знаходити потрібні книги за назвою, автором, жанром або іншими параметрами.

Реалізувати особисті кабінети користувачів: Забезпечити можливість реєстрації та авторизації, а так-ож створення особистих профілів, де користувачі можуть переглядати історію взаємодії з бібліотекою, додавати обрані книги, залишати відгуки тощо.

Забезпечити можливість видачі та повернення книг: Реалізувати механізми онлайн замовлення та отримання книг, а- також процес повернення та оновлення статусу книг у бібліотеці.

Забезпечити безпеку даних та конфіденційність користувачів: Застосувати найкращі практики щодо зберігання та захисту особистих даних користувачів, включаючи паролі, інформацію про платежі тощ-о.

Розширити функціональні можливості: Постійно вдосконалювати та розширювати функціонал платформи, додавати нові функції на основі вимог користувачів і новітніх технологій.

Оптимізувати взаємодію та продуктивність: Забезпечити швидку та ефективну роботу платформи, щоб забезпечити максимальний комфорт для користувачів при роботі з бібліотекою.

Дана програма призначена для використання бібліотекарями- та користувачами- з метою скорочення часових затрат на заповнення даних про книги та читачів, бібліотекарів, а також обліку виданих книг. Вихідними даними для програми є:

інформація про читачів;

інформація про книги;

інформація про видання книг.

-Метою проєкту є - розробка додатку, що дозволяє вести базу даних бібліотеки, вести облік книг, видач книг, читачів, довідкової інформації про працівників з можливістю перегляду, створення, зміни та видалення записів у БД, а також пошук по таблицям.

Також н-еобхідно реалізувати -зменшення часових витрат бібліотекарів на розгортання ПЗ;

-Надати можливість -простого в освоєнні графічного інтерфейсу користувача;

-Реалізувати -значне зменшення затрат часу працівників бібліотеки на заповнення відповідних обліків та таблиць;

надання інформації у зручному форматі.

Для досягнення поставленої мети, будуть використані технології та інструменти, що дозволяють ефективно розробляти та пі-дтримувати програмне забезпечення.

Створення зручного та доступного середовища: Забезпечення легкого доступу до книжного контенту для користувачів будь-де і будь-коли.

Покращення користувацького досвіду: Забезпечення інтуїтивного та зручного інтерфейсу, що- робить використання бібліотеки приємним та ефективним для користувачів.

Ефективний пошук та навігація: Розробка механізмів швидкого та точного пошуку книг, які дозволять користувачам легко знаходити необхідний контент.

Забезпечення безпеки та конфіденційн-ості: Захист особистої інформації користувачів та даних системи від несанкціонованого доступу.

Розширення функціональності: Поступове розширення можливостей системи за допомогою нових функцій, які забезпечать додаткові можливості для користувачів.

Забезпеч-ення стабільності та надійності: Розробка програмного продукту, який працює

стабільно та надійно під час використання користувачами.

Задоволення потреб користувачів: Створення середовища, що задовольняє потреби та очікування користувачів, забезпечуючи поз-итивний досвід використання.

-1.3 Аналіз Вимог-

-Процес проектування вимог програмного забезпечення для онлайн бібліотеки є ключовим етапом, що включає ряд кроків та методів для створення функціональної та ефективної системи.[5]

Оцінка та документування вимог до системи від користувачів, що включає функ-ціональні та нетехнічні вимоги, необхідність системи безпеки та масштабованості. [5]

Проектування архітектури системи:

Розробка структури системи, включаючи визначення компонентів, модулів, їх взаємодію та інтерфейси, що забезпечують функціональність систе-ми.

Проектування інтерфейсу користувача (UI/UX):

Розробка зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, який відповідає їх очікуванням та забезпечує комфортне взаємодію з системою.

Проектування бази даних: Створення структури бази даних, -вибір типів даних, таблиць та зв'язків між ними, що дозволяє зберігати та ефективно обробляти інформацію.

Розробка деталей архітектури: Уточнення технічних деталей архітектури, вибір технологій та інструментів для реалізації функціональності системи.

Проек-тування модулів та компонентів:Розробка окремих модулів, їх функціональності та взаємодії з іншими частинами системи.

Тестування проектування: Перевірка архітектури на відповідність вимогам, виявлення та виправлення можливих недоліків та недоречностей у про-екті.

Документування проекту: Підготовка документації, що описує проектні рішення, структуру системи, інтерфейси та інші технічні аспекти.

Процес проектування ПЗ для онлайн бібліотеки передбачає систематичний та послідовний підхід до розробки, що дозволяє- створити функціональну та ефективну систему, відповідну вимогам користувачів та бізнес-потребам

1.4- Вимоги до інтерфейсу користувача

Користувацький інтерфейс повинен- містити:-

Зручний та інтуїтивний інтерфейс: Інтерфейс повинен бути простим у використанні та інтуїтивно зрозумілим для всіх категорій користувачів.- [-1-]-

Ергономічний дизайн: Важливо, щоб елементи управління були розміщені з урахуванням логічного порядку дій користувача та не заважали йому в користуванні.

Адаптивний дизайн: Інтерфейс повинен бути адаптований під різні типи пристроїв (комп'ютер, планшет, мо-більний телефон) для зручного використання на будь-якому пристрої.- [-1-]-

Чітка навігація: Забезпечити легку та швидку навігацію користувача по різних розділах та функціям платформи.- [-1-]-

Мінімалістичний дизайн: Уникати перевантаження інформацією, застосовувати прості та чисті дизайнерські рішення.

Функціональність пошуку та фільтрації: Забезпечити можливість швидкого пошуку та фільтрації книг за різними критеріями.

Підтримка особистих каб-інетів користувачів: Надати можливість реєстрації, авторизації, перегляду особистих даних та історії взаємодії з бібліотекою.- [-1-]-

Пояснення для користувачів: Забезпечити пояснення та інструкції для користувачів щодо користування різними функціями та можливостями платформи.- [-1-]-

Безпека та конфіденційність: Захист особистих даних користувачів, зокрема паролів, платіжної інформації та іншої конфіденційної інформації.

Контрастність та доступність: Забезпечити достатню контрастність, щоб інтерфейс був доступний для користувачів з рі-зними видами обмежень.

Авторизація та Реєстрація:

Проста форма для входу та створення облікового запису з обов'язковими полями (ім'я, електронна пошта, пароль).

Головна Сторінка:

Чіткий та привабливий дизайн, який відображає важливі функції та актуальну і-нформацію про нові книги або акції.

Меню або панель навігації для швидкого доступу до основних розділів бібліотеки.- [-1-]-

Пошук та Фільтрація:

Простий та зручний пошук книг за назвою, автором, жанром тощо.
Фільтри для точного та швидкого пошуку, наприклад, за рейтингом, роком видання, мовою тощо. - [-1-]-

Сторінка Книги:

Інформація про книгу (назва, автор, опис, обкладинка, рейтинг, відгуки).
Можливість читати книгу онлайн або завантажувати.
Кнопки для додавання книги до обраного, оцінювання та написання відгуку.

Профіль Користувача:

Огляд особистих даних -користувача та можливість зміни профілю.
Історія прочитаних книг, обране, активні позички тощо.

Адаптивність та Відповідність:

Адаптований дизайн для різних пристроїв (планшети, смартфони, ПК).
Відповідність вимогам щодо доступності та можливість використа-ння основних функцій навіть для користувачів з обмеженими можливостями.
Вимоги до інтерфейсу користувача мають за мету забезпечити зручність використання, чіткість структури та інтуїтивно зрозумілі дії для користувачів будь-якого рівня.

-1.5 -Вимоги до апаратних, програмних та комунікаційних інтерфейсів

Для роботи програми необхідно мати встановлену систему управління базами даних [MySQL](#) (локально), програмну платформу [.NET](#) 6, а також обчислювальну систему наступної мінімальної апаратної конфігу-рації:

Попри наявність новіших версій платформи, [.NET](#) 6 було обрано через його стабільність, сумісність з [MySQL](#) та підтримку довгострокової підтримки ([LTS](#)), що забезпечує безпечну експлуатацію.

процесор з тактовою частотою 1,0 ГГц;
ОЗУ об'ємом не менше 1 Г-б;
вільне місце на жорсткому диску 300 Мб;
відеокарта з об'ємом пам'яті не менше 514 Мб;
операційна система [Windows](#) 10;

наявність засобів введення-виведення: миша, клавіатура, монітор; Необхідно забезпечити програмне взаємодію системи з системою управління- базами даних [MySQL](#).

-1.6 -Вимоги до адаптації на місці

Перед використанням потрібно експортувати базу даних [Library.sql](#), що додається до програми, та переконатися в успішному завершенні цієї операції. Перевірити доступ програми до бази даних. Для встановлення програми достатньо зав-антажити теку з файлами програми. Запуск програми здійснюється при запуску файлу [LibraryDB.exe](#).

Вимоги до адаптації на місці можуть стосуватися здатності системи адаптуватися до специфічних вимог чи умов конкретного місця або регіону. Ось деякі можливі вим-оги:

Міжнародна підтримка:
Мульти-мовна підтримка для користувачів з різних країн.
Можливість зміни мови інтерфейсу системи.

Культурні аспекти:
Урахування особливостей культури та релігійних переконань у представленні матеріалів або вмісту бібліотеки.
Забезпечення відповідності законодавству та моральним нормам різних країн.

Локальні потреби:
Адаптація пошукових алгоритмів до специфічних особливостей місцевих потреб користувачів.
Врахування регіональних попередніх вподобань у відображенні рекомендацій та пр-опозицій.

Технічні аспекти:
Підтримка різних типів пристроїв, які є популярними в конкретній локації (наприклад, підтримка великої кількості смартфонів, планшетів чи мобільних пристроїв).
Оптимізація для швидкої роботи на мобільних мережах, які можуть бути- менш стабільними або повільними у деяких регіонах.

Системні вимоги:
Мінімальні вимоги до апаратних можливостей для забезпечення швидкої та ефективної роботи системи на різних пристроях.
Ці вимоги до адаптації на місці важливі для забезпечення успішної експлуатації продукту в різних країнах чи місцевостях з різними культурними, технічними та соціальними характеристиками.

-1.7- Функції продукту

Приклад додатку має бути здатний працювати з таблицями бази даних для обробки інформації про книги, читачів та їх видачі. У програмі повинна бути реалізована можливість пошуку інформації в таблиці за різними запитам. Також має бути функція генерації деяких звітів, таких як неповнолітні читачі, боржники та книги, що закінчилися.

Функції продукту

Цитування: 0,02%

id: 6

"Онлайн бібліотека"

можуть включати різноманітні можливості, спрямовані на задоволення потреб користувачів у доступі до літературних ресурсів та з-ручного їх використання. Ось деякі базові функції:

Реєстрація та Авторизація:

Можливість створення облікового запису та авторизації для доступу до функцій бібліотеки.

Пошук та Фільтрація:

Пошук книг за різними критеріями: назва, автор, жанр, рік видання то-що.

Фільтрація результатів пошуку для точнішого вибору.

Читання та Відгуки:

Можливість читання книг онлайн без необхідності завантаження.

Додавання відгуків та оцінок книг.

Управління Контентом:

Додавання нових книг та їх видалення.

Можливість редагування -інформації про книгу: опис, обкладинка, автор тощо.

Особистий Кабінет Користувача:

Моніторинг прочитаних книг.

Можливість створення власного списку обраних книг.

Управління особистими даними та паролем.

Сповіщення та Рекомендації:

Система сповіщень про нов-і книги, оновлення або акції.

Автоматичні рекомендації книг на основі історії читання та вибору користувача.

Система Користувацьких Прав:

Налаштування приватності, управління доступом до книг та профілю.

Підтримка Різних Форматів:

Підтримка різних форматів- книг: електронні книги у [PDF](#), [EPUB](#) тощо.

Ці функції дозволяють користувачам зручно та ефективно взаємодіяти з онлайн бібліотекою, забезпечуючи доступ до різноманітного контенту та можливості управління ним.

-1.8- Обмеження

Заборонено перейменовувати, редагувати та видаляти файли всередині файлової системи програми;

Програма не працює з іншими СУБД, і також потребує локального підключення до СУБД [MySQL](#);

Продукт успішно працює лише в ОС [Windows](#) 10-,11- з встановленою платформою [.NET](#) 6.0;

Продукт не передбачає автоматичного переходу на платформи, які не перераховані в даному документі.

-2. РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ПРОЄКТУВАННЯ ПРОГРАМНИХ СИСТЕМ

2.1. Огляд існуючих методів та основних принципів проєктування, архітектури програмного забезпечення, інженерних підходів і принципів керування ПЗ

Процес розробки програмного забезпечення для онла-йн бібліотеки базується на низці принципів, які гарантують його якість, ефективність та стабільність. -Ось кілька ключових принципів, які варто врахувати:

Принцип модульності: Розбиття системи на окремі модулі (або компоненти) дозволяє легко управляти та розвивати код, забезпечуючи його вищу структурованість та повторне використання[5].

-Принцип однорідності (-[Uniformity Principle](#)-): Усі частини системи повинні дотримуватися спільних стандартів та правил програмування для спрощення зрозумілості коду для розробників- [5].-

Принцип розділення відповідальності (-[Separation of Concerns](#)-): Різні аспекти

функціональності (такі як інтерфейс, бізнес-логіка, доступ до даних) повинні бути чітко відокремлені, що сприяє покращенню супроводжуваності та розвитку системи [5].

Принцип єдиної відповідальності (-[Single Responsibility Principle](#)-): Кожен модуль або клас повинен мати лише одну конкретну відповідальність, що спрощує тестування та зменшує взаємозалежність компонентів [5].

Принцип відкритості та закритості (-[Open](#)-/[Closed Principle](#)-): Система повинна бути відкритою для розширення новим функціоналом, але закритою для змін, тобто допускати розширення без необхідності зміни вже наявного коду [5].

-Принцип інверсії залежностей ([Dependency Inversion Principle](#)): Класи модулів мають залежати від абстракцій, а не від конкретних реалізацій, що сприяє зменшенню зв'язаності та підвищує гнучкість системи.

Ці принципи становлять основу для створення ефективно-ї та стабільної архітектури програмного забезпечення для онлайн бібліотеки, допомагаючи забезпечити гнучкість, розширюваність та якість системи [5].

-Огляд існуючих методів:

-[Waterfall](#)- (Каскадний): Традиційний метод, що передбачає послідовні етапи розробки від визначення вимог до випуску продукту.

Модель каскадного ([Waterfall](#)) процесу розробки програмного забезпечення - це лінійний підхід, де кожен етап розробки проходить послідовно один за одним. Ця модель передбачає, що кожен етап повинен бути повністю завершений, перш ніж почнеться наступний.

Основні етапи моделі каскадного процесу розробки ([Waterfall](#)) включають такі кроки:-[4]

-Визначення вимог: Збирання та аналіз вимог від клієнта та створення специфікацій програми.- [4].-

Проектування: Розробка архітектури програми на основі вимог та створення детальних планів реалізації.

Реалізація (розробка): Написання коду та створення програмного продукту на основі визначених вимог і планів.

Тестування: Перевірка програми на відповідність вимогам, виявлення та виправлення помилок.- [4].-

Впровадження (випуск): Введення програми в експлуатацію та постачання продукту клієнту або користувачам.

Підтримка: Надання підтримки продукту, виправлення помилок, оновлення та вдосконалення програмного забезпечення- [4].-

Основна перевага моделі каскадного процесу полягає в простоті та чіткості послідовності етапів, що полегшує управління проектом та робить його більш передбачуваним. Однак ця модель може бути менш гнучкою у порівнянні з іншими методологіями, оскільки будь-які зміни вимог після початку процесу розробки можуть призвести до складнощів та витратних коригувань на пізніших етапах.

--

-[Agile](#)- (Гнучкий): Група методів розробки ПЗ, таких як -[Scrum](#)-, -[Kanban](#)-, -[XP](#)- тощо, які дозволяють гнучко відповідати змінам у вимогах та швидко адаптуватися [3]

-Методологія [Agile](#) (гнучкий) - це набір принципів розробки програмного забезпечення, які ставлять акцент на гнучкість, співпрацю замість контрактів, реагування на зміни, а також визнання цінності робочого продукту. [Agile](#) спрямований на постійну ітеративну розробку, сприяє постійному вдосконаленню і враховує зміни вимог у процесі розробки [2].

Основні принципи методології [Agile](#) включають:

Гнучкість: Здатність швидко реагувати на зміни та адаптуватися до нових умов, навіть у середині розробки. [4]

Ітераційний -підхід: Розробка програмного забезпечення відбувається через короткі, ітеративні цикли розробки. [4]

Співпраця замість контракту: Більша увага приділяється співпраці та комунікації замість формальних контрактів. [4]

Самоорганізація і співпраця команди: Ком-анди самі приймають рішення щодо планування та виконання завдань. [4]

Постійна відмовка і покращення (постійне вдосконалення): Команда постійно аналізує свою роботу та шукає способи поліпшення. [3]

Продуктивна комунікація: Підтримка ефективно-ї та частої ко-мунікації серед членів команди та замовників. [5]

Методологія [Agile](#) дозволяє підтримувати гнучкість та швидкість реагування на зміни у

вимогах клієнта або ринкових умовах, проте вимагає активної участі замовника та досить стабільних вимог для успішної реалізації проекту.

Incremental (Інкрементальний): Розробка в кілька ітерацій, додавання нових функцій та можливостей на кожній ітерації. [4]

Методологія Інкрементального розроблення є однією зі стратегій розробки програмного забезпечення, де процес розробки розділяється на послідовні фази, а кожна нова фаза розширює функціональність попередньої. У цій методології розробки програмне забезпечення будується через поетапне додавання нового функціоналу, кожен з яких є інкрементом або приростом.

Основні аспекти інкрементального підходу включають:

Поступовий розвиток продукту: Програмне забезпечення розробляється етапами, де кожен новий інкремент (приріст) розширює функціональність продукту. [4]

Можливість внесення змін: Дозволяє замовнику вносити зміни вимог під час розробки.

Постійні ітерації і оцінка результатів: Поетапний процес дозволяє проводити регулярні перевірки та тестування, що полегшує виявлення помилок та швидке їх виправлення.

Поступове уточнення вимог: Кожен інкремент додається на основі попереднього, дозволяючи уточнювати вимоги та функціонал по мірі розвитку проекту. [-3-]

Цей метод підходить для проектів, де вимоги часто змінюються або можуть бути уточнені під час розробки. Він дає можливість постійно вдосконалювати та розширювати функціонал програмного продукту через додавання нових інкрементів з покращеними можливостями.

Spiral (Спіральний): Це комбінація каскадної моделі з елементами ітераційного розробки, що включає постійну оцінку ризиків та протилежності. [4]

Методологія

Цитування: 0,01%

id: 7

"Спіральний"

(**Spiral**) є гнучкою моделлю розробки програмного забезпечення, яка поєднує в собі елементи інших методологій розробки та має ітеративний характер. Модель

Цитування: 0,01%

id: 8

"Спіральний"

передбачає постійну адаптацію до нових вимог та управління ризиками.

Основні етапи методології

Цитування: 0,01%

id: 9

"Спіральний"

включають:

Оцінка ризиків (**Identification of Risks**): Аналіз потенційних ризиків, які можуть виникнути на різних етапах розробки програми.

Розробка стратегії (**Developing Strategies**): Розробка стратегії для управління ризиками та визначення шляхів їх уникнення або зменшення. [4]

Виконання (**Execution**): Реалізація програми -та проведення тестування, використання аналізу результатів для планування наступного ітераційного циклу.

Оцінка (**Evaluation**): Аналіз результатів інкрементальних ітерацій та прийняття рішення про продовження або коригування розробки з урахуванням отриманих -результатів.

Кожен ітераційний цикл у методології

Цитування: 0,01%

id: 10

"Спіральний"

може включати в себе віддалені етапи розробки, такі як збір вимог, проектування, реалізація, тестування та випуск продукту. Головний аспект цієї моделі полягає в тому, що ризики активно враховуються і зменшуються на кожному етапі розробки.

Метод

Цитування: 0,01%

id: 11

"Спіральний"

відмінно підходить для проектів, які потребують великої уваги до управління ризиками, а також тих, де вимоги можуть бути змінені під час розробки, дозволяючи фокусуватися на

гнучкості та ада-птації.

RAD (Радикальна розробка): Метод, орієнтований на швидкий процес розробки через використання прототипів та інтенсивну залученість замовника. [4]

Методологія **RAD** (**Rapid Application Development**) - це стратегія розробки програмного забезпечення, яка с-прямована на швидке розгортання системи за допомогою швидких ітерацій та великого акценту на роботі з клієнтом. [4]

Основні особливості методології **RAD**:

Швидкість розробки: Ця методологія спрямована на швидке створення продукту. Короткі ітерації дозволяють- швидко реагувати на зміни та додавати новий функціонал.

Процес взаємодії з клієнтом: Основний акцент робиться на активній співпраці та залученні клієнта на кожному етапі розробки.

Прототипування: Використання прототипів для демонстрації можливостей та отр-имання відгуку клієнта на ранніх стадіях розробки.

Робота в команді: Команди розробки працюють над великою кількістю завдань одночасно, прискорюючи процес розробки.

Рекурсивність ітерацій: Після отримання відгуку від клієнта, продукт може проходити через к-ілька ітерацій для уточнення та вдосконалення.

Методологія **RAD** найбільш ефективно використовується для проектів, де вимоги користувачів не зовсім чіткі та можуть змінюватися під час розробки. Ця методологія дозволяє швидко створювати та адаптувати програмн-е забезпечення з великою увагою до вимог клієнтів. Однак, вона може бути менш ефективною для проектів з жорсткими обмеженнями щодо бюджету та ресурсів.

DevOps: Підхід, що поєднує розробку та операції з метою автоматизації процесів виробництва ПЗ. [4]

DevO-ps підхід розглядається як сучасна практика забезпечення безперервної інтеграції, тестування та розгортання систем, що може бути корисним при подальшому масштабуванні або підтримці онлайн бібліотеки.

DevOps - це підхід до розробки програмного забезпечення,- що поєднує розробку (**Development**) та експлуатацію (**Operations**). Основною метою **DevOps** є створення швидких, стабільних та автоматизованих процесів розробки та розгортання програмного забезпечення.

Ключові аспекти методології **DevOps**:

Співпраця і комунікація:- Злагоджена робота між розробниками, тестувальниками та адміністраторами систем з метою вдосконалення процесів розробки та впровадження. [4]

Автоматизація: Використання інструментів для автоматизації процесів розгортання, тестування та моніторингу програм-ного забезпечення.

Неперервна інтеграція та неперервна доставка (**CI/CD**): Постійна інтеграція нового коду та автоматична доставка змін до продукції для забезпечення швидкої відповіді на зміни.

Моніторинг та забезпечення якості: Систематичний моніторинг робо-ти програми для виявлення помилок та забезпечення високої якості програмного забезпечення. [4]

Еластичність та масштабованість: Можливість швидко змінювати розмір та потужність системи в залежності від потреб.

DevOps сприяє зменшенню часу розгортання прогр-амного забезпечення, поліпшенню комунікації між розробниками та адміністраторами, покращує якість та надійність програмного забезпечення. Цей підхід є особливо корисним у проектах, де потрібна швидка реакція на зміни вимог та швидкий цикл розгортання нових- функцій чи виправлень.

Lean Software Development: Орієнтований на зменшення витрат та мінімізацію марнотратства, фокусуючись на потребах клієнта.

Lean Software Development - це методологія, яка базується на принципах **Lean Manufacturing** та фокусується на е-фективності виробництва і використанні ресурсів для створення програмного забезпечення. [4]

Основні принципи- **Lean Software Development**:

-Визначення цінності (**Value**): Цінність визначається перспективою клієнта - те, що клієнт готовий оплатити.

Ідентифікація потоку цінності (**Value Stream**): Визначення кроків, які необхідно зробити для перетворення вхідних вимог у цінні продукти. [4]

Створення -потягучості (**Pull**): Акцент на виробництві лише тих функцій чи функціоналів, які реально потрібні клієнтам.

Вдосконалення процесу ([Perfection](#)): Постійна праця над вдосконаленням процесів розробки для досягнення максимальної ефективності.

Управління якістю ([Quality](#)): Акцент на постійному контролі якості продукту та процесів його розробки.

Вдалий відбір ([Eliminate Waste](#)): Мінімізація витрат шляхом усунення непродуктивних процесів. [4]

Змінність ([Amplify Learning](#)): Створення сприятливого середовища для навчання- та вдосконалення.

Методологія [Lean Software Development](#) дозволяє оптимізувати розробку шляхом уникнення витрат часу і ресурсів на непотрібні процеси, надає можливість швидко реагувати на зміни та постійно покращувати якість програмного забезпечення. Цей підхід особливо корисний у проектах, де важлива оптимізація витрат та швидкість реакції на зміни.

[FDD \(Feature-Driven Development\)](#): Зосереджений на поступовому визначенні, плануванні, дизайні та реалізації функцій.

Огляд існуючих архітектур програмного забезпечення передбачає розгляд різних архітектурних стилів та шаблонів, які використовуються для розробки програмних продуктів. Ось деякі з них:

[MVC \(Model-View-Controller\)](#): Розділення програми на три основні компоненти: модель (дані), представлення (інтерфейс користувача) та контролер (логіка).

[Microservices](#) (Мікросервіси): Архітектурний підхід, що передбачає розбиття програми на невеликі автономні сервіси, що працюють разом.

[SOA \(Service-Oriented Architecture\)](#): Архітектурний підхід, де програма складається з різних сервісів, які комунікують між собою через мережу.

[Event-Driven Architecture \(EDA\)](#): Архітектурний стиль, де взаємодія між компонентами програми здійснюється за допомогою подій.

[Layered Architecture](#) (Шарова архітектура): Розділення програми на логічні шари (наприклад, презентаційний, бізнес-логіка, доступ до даних).

[Hexagonal Architecture \(Ports and Adapters\)](#): Фокусується на відокремленні бізнес-логіки від зовнішніх елементів через використання портів та адаптерів.

[Component-Based Architecture](#) (Архітектура на основі компонентів): Підхід, де програма розбита на окремі компоненти, які можуть бути розроблені та підтримувані окремо.

[CQRS \(Command Query Responsibility Segregation\)](#): Розділення запитів на читання та запис на окремі моделі, що дозволяє оптимізувати читання та запис.

Патерни проектування в програмуванні - це загальні рекомендації та шаблони, які виникли на основі накопиченого досвіду розробки програмного забезпечення. Ось кілька типових паттернів, які можна використовувати у розробці онлайн бібліотеки:

Модель---Вид---Контролер- ([Model-View-Controller, MVC](#)):

-Опис: Розділяє програму на три основні складові: модель (дані), вид (представлення) та контролер (логіка).

Використання в онлайн бібліотеці: Модель може включати дані про книги та користувачів, Вид - відображення інформації користувачам, а Контролер - обробку запитів користувачів.

Сервісний шар ([Service Layer](#)):

Опис: Визначає сервіси, які надають бізнес-логіку та функціональність системи.

Використання в онлайн бібліотеці: Сервіси можуть забезпечувати функції каталогування книг, авторизації користувачів та інші операції.

Репозиторій ([Repository](#)):

Опис: Відокремлює доступ до даних від бізнес-логіки та забезпечує уніфікований інтерфейс для роботи з базою даних.

Використання в онлайн бібліотеці: Репозиторії можуть використовуватися для взаємодії з базою даних книг, користувачів тощо.

Фабричний метод ([Factory Method](#)):

Опис: Надає спосіб створення об'єктів без прив'язки до конкретних класів.

Використання в онлайн бібліотеці: Може використовуватися для створення об'єктів книг, користувачів, які можуть бути різних типів.

Одинак ([Singleton](#)):

Опис: Гарантує, що певний клас має лише один екземпляр та надає глобальний доступ до цього екземпляра.

Використання в онлайн бібліотеці: Може застосовуватися для доступу до певних об'єктів, наприклад, системного журналу або менеджера сесій.

Фасад ([Facade](#)):

Опис: Надає простий інтерфейс до складних підсистем, щоб полегшити їхнє використання. Використання в онлайн бібліотеці: Може використовуватися для створення простого інтерфейсу для клієнтів бібліотеки для реєстрації, пошуку книг тощо

-2.2 Обґрунтування вибору методів та підходів до проєктування ПЗ

Обґрунтування методів та підходів до проєктування програмного забезпечення для онлайн бібліотеки є важливим етапом, що визначає правильний напрямок розробки продукту.

-Результативність будь-якого методу залежить від впровадження його принципів та відповідного підходу до роботи з ним. Тому вибір методу повинен базуватися на потребах проекту та можливостях команди розробників.

- Нижче наведено ті самі методи та підходи, які будуть використовуватися для розробки та проєктування для додатку онлайн бібліотеки:

-Метод:

[Agile \(Scrum\)](#): Гнучкий метод розробки, що дозволяє швидко адаптуватися до змін у вимогах та забезпечити більшу прозорість процесу розробки.

Архітектура:

Модульна архітектура є підхід, що полягає у розбитті програми на окремі, незалежні від інших час-тин, модулі. Кожен модуль відповідає за виконання певного завдання чи функції, і має свій власний інтерфейс для взаємодії з іншими модулями.

Основні переваги модульної архітектури включають:

Модульність і відокремленість: Модулі можуть бути розроблені, т-естовані та вдосконалені незалежно один від одного, спрощуючи процес розробки.

Легше супроводження і розширення: Модульна структура сприяє легкому виявленню помилок та їх локалізації, а також спрощує додавання нової функціональності.

Використання інтерфейсів: Використання чітко визначених інтерфейсів між модулями дозволяє розділити відповідальності та знизити залежність між ними.

Підтримка тестування: Модулі можуть бути протестовані окремо, що полегшує виявлення помилок та забезпечення їх правильного функціонування.

Підтримка багаторазового використання коду: Модулі, які відповідають за конкретні функції, можуть бути використані у різних частинах програми або навіть в різних проєктах. У модульній архітектурі кожен модуль може мати свої внутрішні складові, та-кі як класи, функції, методи, які допомагають виконувати визначені завдання. Важливо планувати структуру модулів заздалегідь, визначаючи їхні функціональні області та взаємодію між ними, щоб досягти більшої ефективності та гнучкості в розробці програмного - забезпечення.

Інженерний підхід:

[Model-View-Controller \(MVC\)](#): Розділення програми на модель (дані), представлення (інтерфейс користувача) та контролер (логіка).

[Test-Driven Development \(TDD\)](#): Розробка через написання тестів перед розробкою коду, що сприяє -високій якості програмного продукту.

-Методологія [Agile](#): Обґрунтування використання методології [Agile](#) базується на необхідності гнучкості та можливості швидко вносити зміни в систему відповідно до змінних вимог користувачів. Цей підхід дозволить швидше адаптуватися до змін та покращувати продукт на кожному етапі розробки.

Розробка через тестування ([Test-Driven Development, TDD](#)): Обґрунтування використання [TDD](#) полягає в підході, коли спочатку розробляються тести для функціоналу, а потім сам функціонал. Це дозволяє створювати стабільний та відповідний функціонал та зменшує кількість помилок у програмному коді.

Модульне тестування ([Unit Testing](#)): Обґрунтування використання модульних тестів полягає в тому, щоб перевірити окремі модулі (функції, класи тощо) на коректність роботи. Цей підхід дозволяє -виявляти помилки та забезпечує більшу надійність програмного забезпечення.

Принципи [SOLID](#) та [Clean Code](#): Обґрунтування застосування принципів [SOLID \(Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion\)](#) та практик

"чистого коду"

([Clean Code](#)) визначається бажанням створити систему, яка буде легко змінюватися та розширюватися з мінімальними зусиллями.

Проектування за допомогою [UML](#)-діаграм: Обґрунтування використання [UML](#)-діаграм ([Unified Modeling Language](#)) полягає в можливості візуалізації структури системи, її компонентів та взаємодій між ними, що сприяє кращому розумінню системи та зручності комунікації в команді розробників.

Обґрунтування вибору цих підходів полягає у прагненні до ефективного та гнучкого розробництва, яке дозволить створити високоякісне програмне забезпечення для онлайн бібліотеки, що відповідає сучасним стандартам розробки та вимогам користувачів.

-Вибір конкретного методу розробки програмного забезпечення для онлайн бібліотеки може бути залежний від різних факторів, таких як розмір команди розробників, складність проекту, обсяг та специфіка вимог клієнта, час та бюджет, доступні ресурси, технічні знання та навички команди тощо. Проте, -я вважаю що вибраний метод а саме м-етодологія [Agile](#), зокрема [Scrum](#) або [Kanban](#), може бути вибраним методом для розробки програмного забезпечення онлайн бібліотеки. [Agile](#) надає можливість гнучко реагувати на зміни вимог, швидше адаптувати продукт до потреб користувачів, ефективно керувати процесом розробки та сприяє постійному вдосконаленню продукту.-

-[Scrum](#) чи [Kanban](#) - це конкретні реалізації методології [Agile](#). [Scrum](#) використовує ітераційний підхід з короткими циклами розробки (спринтами), плануванням, регулярними відгуками та постійним вдосконаленням. [Kanban](#) спрямований на створення потокової моделі розробки з акцентом на візуалізацію робочого процесу та його оптимізацію.

Обираючи між [Scrum](#) та [Kanban](#), важливо врахувати специфіку проекту та потреби команди. [Scrum](#) може бути корисним для проектів, які вимагають стриманого планування та чіткого управління с-принтами. У той час як [Kanban](#) може бути ефективним у проектах, де важлива постійна відкритість до змін та потоковий процес розробки.

Обґрунтування використання методу [Scrum](#) для проектування програмного забезпечення онлайн бібліотеки може бути наступним:

Гнучкість та адаптивність до змін:

Підхід [Scrum](#) дозволяє гнучко реагувати на зміни вимог та пріоритетів, що є ключовим у розробці програмного продукту, особливо для онлайн бібліотеки, де можуть з'являтися нові технології або змінюватися потреби користувачів.-

Ітераційний підхід:

Робота у вигляді ітераційних спринтів у [Scrum](#) дозволяє розробляти та випускати функціонал частіше, забезпечуючи постійний прогрес у розвитку продукту.

Зосередження на важливому функціоналі:

[Scrum](#) спрямований на розробку функцій з високим пріоритетом, що дозволяє концентрувати зусилля на важливих для користувача аспектах онлайн бібліотеки, таких як зручний пошук, легкість використання та доступ до книг.

Регулярні відгуки та вдосконалення:

Регулярність відгуків під час планування, спринт-рев'ю та ретроспектив дозволяє виявляти недоліки швидше, вносити зміни та покращувати якість продукту.

Ефективне управління ризиками:

[Scrum](#) сприяє ранньому виявленню проблем, що дозволяє управляти ризиками та вирішувати їх у ранній стадії проекту.

Комунікація та співпраця в команді:

[Scrum](#) підтримує активну комунікацію та співпрацю між учасниками команди, що є ключовим для успішного розвитку продукту та уникнення можливих непорозумінь.

Використання методу [Scrum](#) у процесі розробки онлайн бібліотеки допоможе ефективно управляти проектом, забезпечуючи якісний та користувацько-орієнтований результат, відповідно до вимог користувачів та бізнес-потреб.

Спринт 1:

Мета: Базовий функціонал системи.

Реєстрація та авторизація користувачів.

Основна структура користувацького інтерфейсу.

Можливість перегляду списку книг.

Спринт 2:

Мета: Пошук та фільтрація книг.

Реалізація пошуку за назвою, автором та жанром.

Можливість фільтрувати книги за різними категоріями.

Спринт 3:

Мета: Функціонал читання та відгуки.

Додавання можливості читати книги онлайн.

Відгуки та оцінки книг.

Спринт 4:

Мета: Додатковий функціонал та покращення.

Додаткові функції для користувачів, такі як додавання книг до обраного, рекомендації тощо.

Виправлення помилок та вдосконалення інтерфейсу.

Кожен спринт - має свою мету та конкретний функціонал, який повинен бути реалізований протягом цього періоду. Перед кожним спринтом варто відпрацювати деталі, збір вимог, розробку та тестування, щоб забезпечити успішне виконання завдань та досягнення поставлених цілей.

-

-

3. РОЗДІЛ 3. ОПИС ТА ОБГРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ ПО ПРОЄКТУВАННЮ ПРОГРАМНОЇ СИСТЕМИ

-3.1 -Структурні схеми системи (модулі системи та компоненти)

Створення онлайн бібліотеки передбачає вибір оптимальної архітектури системи, що визначатиме основні принципи організації, взаємодії та компонентів програмного забезпечення. Рішення щодо архітектурно-го стилю базується на ряді факторів та потреб проєкту. Для онлайн бібліотеки розглядаються наступні архітектурні стилі:

МОДУЛЬНА СТРУКТУРА Цей стиль передбачає розділення системи на кілька частин.

Клієнтська частина відповідає за інтерфейс користувача, взаємодію з ним, тоді як серверна частина керує базою даних, обробкою запитів та забезпеченням даних.

Model-View-Controller (MVC): Цей архітектурний стиль використовується для розділення програми на три основні компоненти: Модель, Що представляє дані та логіку, Перегляд, Що відображає інтерфейс користувача, та Контролер, Що керує взаємодією між моделлю та переглядом.

Event-Driven Architecture (EDA): Цей стиль базується на обміні подіями між різними компонентами системи. Компоненти реагують на події та відповідно реагують на них. При виборі архітектурного стилю для онлайн бібліотеки важливо врахувати рівень складності проєкту, потреби користувачів, масштабованість, продуктивність та інші фактори, що впливають на успішність розробки та функціональність системи.

--Зважаючи на проектування додатку онлайн бібліотеки за допомогою **Service-Oriented Architecture**, основні етапи можуть виглядати наступним чином:

Аналіз бізнес-потреб та вимог:

Оцінка потреб користувачів: можливість пошуку книг, реєстрація користувачів, оформлення запитів на книги тощо.

Визначення функціональності: система каталогізації, процес видачі книг, профілі користувачів, аутентифікація та авторизація.

Розділення функціональності на сервіси:

Каталогізація книг.

Управління користувачами.

Авторизація та а-утентифікація.

Видача та повернення книг.

Сервіс пошуку книг.

Проектування **API** сервісів:

Визначення та документування інтерфейсів для кожного сервісу, включаючи доступні функції та методи.

Реалізація сервісів:

Розробка окремих сервісів згідно з визначеними- **API**, включаючи реалізацію логіки та доступ до баз даних для кожного сервісу.

Управління залежностями та взаємодією сервісів:

Забезпечення взаємодії між сервісами через визначені **API**, контроль над залежностями між

ними.

Тестування та валідація сервісів:

Пр-оведення тестів на кожному з сервісів для переконання, що вони працюють правильно та взаємодіють належним чином.

Розгортання та моніторинг:

Розгортання сервісів у продакшн та постійний моніторинг їхньої роботи для виявлення можливих проблем та оптимізації роботи системи.

Підтримка та розвиток:

Підтримка роботи сервісів у виробничому середовищі та їхній подальший розвиток, виправлення помилок, впровадження нових можливостей.

-Структурні схеми системи в контексті онлайн бібліотеки можуть бути представлені через кілька основних компонентів та їх взаємодію:

Фронтенд інтерфейсу користувача (**Frontend**):

Головна сторінка: Пошук книг, категорії, рекомендації, останні оновлення.

Сторінк-и книг: Відображення інформації про книгу, кнопки читання/завантаження, відгуки користувачів тощо.

Авторизація/Реєстрація: Форми для входу або створення облікового запису.

Бекенд (**Backend**):

Система управління базою даних: Зберігання інформації про книги, к-ористувачів, їх вибори та взаємодію з системою.

Додаткові сервіси: Система авторизації, система рекомендацій, система обробки платежів (якщо передбачено).

Інтеграція зовнішніх сервісів:

Системи платіжних шлюзів: Якщо передбачається платний доступ до контен-ту.

Сервіси аналітики: Для відстеження активності користувачів, оцінювання популярності книг тощо.

API для інтеграції з іншими платформами або соціальними мережами.

Хмарні технології та зберігання даних:

Зберігання книг у хмарних сховищах: Для забезпечення- доступу до контенту в режимі онлайн або офлайн.

Захист та резервне копіювання даних: Для забезпечення безпеки та надійності.

Ці структурні схеми відображають основні компоненти системи онлайн бібліотеки та їх взаємодію для забезпечення функціональності, б-езпеки та зручності користувачів. Вони дозволяють систематизувати роботу різних частин системи та забезпечити їх взаємодію для досягнення загальних цілей продукту.

Розробка онлайн бібліотеки передбачає створення ряду модулів та компонентів, які взаємодіють- між собою для забезпечення функціональності системи. Ось загальний огляд основних модулів та компонентів, що планується розробляти:

Модуль авторизації та аутентифікації:

Відповідає за процеси реєстрації нових користувачів та їх авторизацію в системі.

Моду-ль управління користувачами:

Надає можливість переглядати та змінювати інформацію про користувачів, включаючи їх особисті дані та історію.

Модуль управління книгами:

Відповідає за додавання нових книг, редагування інформації про книги та їх видалення.

Мод-уль пошуку та фільтрації:

Забезпечує можливість пошуку книг за різними критеріями (назва, автор, жанр тощо) та їх фільтрацію.

Модуль читання книг:

Надає засоби для зручного читання книг онлайн та можливість завантаження для офлайн-читання.

Модуль рейтингув-ання та відгуків:

Дозволяє користувачам залишати відгуки та оцінювати прочитані книги.

Модуль рекомендацій:

Надає персоналізовані рекомендації користувачам на основі їхньої історії читання.

Модуль керування вибраним контентом:

Забезпечує можливість користу-вачам додавати книги до списку вибраного для зручного доступу.

Кожен з цих модулів буде мати свою функціональність, взаємодіяти з базою даних та

іншими компонентами системи. Проєктування модулів та компонентів виконується з урахуванням їхньої незалежності, - легкості розширення та можливості масштабування, щоб забезпечити ефективну та стабільну роботу всієї системи онлайн бібліотеки.

-

-Малюнок- 3.1 - -Зв'язок файлів додатку

3.2 Опис Архітектурної системи

Архітектура онлайн бібліотеки включає різні компоненти, які спільно працюють для забезпечення функціональності та ефективності системи.

Клієнтська частина (**Frontend**): Цей компонент відповідає за інтерф-ейс користувача.

Включає у себе веб-інтерфейс або мобільні додатки, які дозволяють користувачам взаємодіяти з бібліотекою, шукати книги, переглядати інформацію про них, оформляти замовлення та виконувати інші дії.

Серверна частина (**Backend**): Цей компонент -забезпечує логіку бізнес-процесів. Включає в себе сервер, базу даних, додаткові сервіси для обробки запитів користувачів, управління даними та бізнес-логіку, наприклад, авторизацію, аутентифікацію, обробку оплати, пошук книг тощо.

База даних: Це централізо-ване сховище, де зберігається вся інформація про книги, користувачів, замовлення, рейтинги, коментарі та інша необхідна інформація для роботи системи.

Система авторизації та безпеки: Цей компонент відповідає за захист конфіденційності даних користувачів, а-втентифікацію, авторизацію та забезпечення безпеки системи в цілому.

Модуль управління контентом: Включає в себе функціонал для додавання, видалення та редагування інформації про книги, оновлення даних та управління контентом.

Сервіси та **API**: Додаткові сер-віси або **API**, які можуть використовуватися для підключення додаткових функцій, співпраці з іншими системами, обміну даними тощо.

Архітектура системи онлайн бібліотеки ретельно спроектована для забезпечення ефективної взаємодії між різними компонентами, заб-езпечення безпеки, швидкодії та функціональності для користувачів.

3.3 Структура бази даних , моделювання бази даних (інфологічна модель)

-Структура бази даних для онлайн бібліотеки може бути представлена через низку сутностей та їх взаємозв'язки. Ось можлива інфологічна модель бази даних для системи онлайн бібліотеки:

Користувачі (**Users**):

user_id (ідентифікатор користувача)

username (ім'я ко-ристувача)

email (електронна адреса користувача)

password_hash (хеш пароля)

інші властивості (наприклад, роль користувача)

Книги (**Books**):

book_id (ідентифікатор книги)

title (назва книги)

author (автор книги)

genre (жанр книги)

description (опис книги)

file_path (шлях до файлу з книгою)

інші властивості (наприклад, рік видання, кількість сторінок тощо)

Оцінки та відгуки (**Ratings/Reviews**):

rating_id (ідентифікатор оцінки)

user_id (зовнішній ключ до таблиці користувачів)

book_id (зовнішній ключ до таблиці кни-г)

rating (оцінка користувача)

review_text (текстовий відгук користувача)

Автори (**Authors**):

author_id (ідентифікатор автора)

author_name (ім'я автора)

інші властивості (наприклад, рік народження, країна тощо)

Категорії/Жанри (**Categories/Genres**):

category_id (ідентифікатор категорії)

[category_name](#) (назва категорії)

Ця інфологічна модель бази даних визначає основні сутності та їх атрибути, а також взаємозв'язки між ними. Наприклад, кожен користувач може оцінювати та залишати відгуки про книги, а книги можуть -бути призначені певним категоріям чи жанрам. Реалізація фізичної моделі бази даних (таблиці, ключі, індекси тощо) буде залежати від конкретної системи управління базами даних (наприклад, [MySQL](#), [PostgreSQL](#), [MongoDB](#) та ін.).

Ця модель може бути розширена або- змінена залежно від конкретних вимог до системи та додаткових функцій, які можуть бути включені до онлайн бібліотеки.

Логічне проектування - створення схеми бази даних на основі конкретної моделі даних. У проекті використовується реляційна база даних. Для- кожної виділеної сутності необхідно продумати поля, включаючи первинні та зовнішні ключі. Також були продумані зв'язки таблиць. У даному випадку всі зв'язки доцільно створити як

Цитування: 0,03%

id: 13

"один до багатьох".

У такого роду зв'язках рядок в першій таблиці може мати багато представлень в другій таблиці, але рядок у другій таблиці може мати тільки один рядок в першій таблиці. З урахуванням даних бізнес-процесів та предмета дослідження була створена логічна модель. Логічне представлення бази даних наведено на -малюнку- 4.2.

РК - ПЕРВИНИЙ КЛЮЧ

-[FK](#)- - -ЦЕ КЛЮЧ-, -ВИКОРИСТОВУЄТЬСЯ ДЛЯ З'ЄДНАННЯ ДВОХ ТАБЛИЦЬ РАЗОМ.

-

-Малюнок 3.-2- - -Інфологічна- модель баз [ы](#) данн [ых](#)

-Малюнок 3.2 - -[UML Diagram](#)-, -а-рхітектурна схема взаємодії компонентів онлайн бібліотеки.-

--

Малюнок 3.3 -[UML DIAGRAM](#)- Прецедентів, Інфологічна модель бази даних онлайн бібліотеки

4. РОЗДІЛ 4. ПРАКТИЧНА РЕАЛІЗАЦІЯ (-РОЗРОБКА БАЗИ ДАНИХ БІБЛІОТЕКИ-)

В якості системи управління базами даних для створення та зберігання реляційної БД бібліотеки був обраний -[MySQL](#)-, а також утиліта -[Open Server Panel](#)-. -[MySQL](#) - вільна реляційна система управління базами даних, розроблена та підтримувана американською компанією [Oracle](#). Ця СУБД є хорошим рішенням для малого та середнього програмного забезпечення. [MySQL](#) можна використовувати як сервер, до якого звертаються клієнти (локально або віддалено), проте СУБД підтримує функціонал, який дозволяє включати [MySQL](#) всередину програми [3]. Крім цих переваг, система управління базами даних є однією з найбільш поширених та популярних на ринку [4]. [Open Server Panel](#) - утиліта, - призначена для розробки, налагодження та тестування [WEB](#)-проектів, а також для надання веб-сервісів у локальних мережах. Цей програмний комплекс включає в себе набір серверного ПО та утиліт для адміністрування та налаштування всіх доступних компонентів. Дан-а утиліта підтримує і [MySQL](#), забезпечуючи зручне та швидке локальне підключення. Для зручної взаємодії з необхідною СУБД з метою розробки, [Open Server Panel](#) містить спеціальний [WEB](#)-інтерфейс для адміністрування - [phpMyAdmin](#), що полегшує роботу з створення -БД [5]. Для створення додатку з графічним інтерфейсом користувача була обрана платформа [Windows Forms](#), що працює на мові [C#](#). В якості середовища розробки була обрана [Microsoft Visual Studio 2022 Community](#). Для взаємодії з [MySQL](#) був обраний фреймворк [MySQL -Connector](#), оскільки він зручний у використанні.

[Windows Forms](#) - це платформа користувацького інтерфейсу для створення класичних додатків [Windows](#) за допомогою візуального конструктора в [Visual Studio](#). Платформа має можливість розміщення елементів керування - шляхом перетягування, спрощуючи створення класичних додатків [6]. [C#](#) - сучасна об'єктно-орієнтована мова програмування, розроблена компанією [Microsoft](#). Дана мова характеризується легкістю в освоєнні, універсальністю та безпекою [7]. [Visual Studio](#) є найбільш функціонально насиченою [IDE](#)-середовищем, в якому можна розробляти додатки на [C#](#). [Visual Studio](#) - це стартова платформа для написання, налагодження та збирання коду, а також подальшої публікації додатків. Інтегроване середовище розробки ([IDE](#)) є багатофункціональною програмою, яку

можна використовувати для різних аспектів розробки програмного забезпечення.

4.1 Розробка бази даних програми

4.1.1 Концептуальне проектування бази даних

Концептуальне проектування - це створення семантичної моделі предметної обл-асті (інформаційної моделі самого високого рівня абстракції). Концептуальна модель бази даних включає в себе сутності та, в деяких випадках, теоретичні зв'язки даних сутностей. Під час вивчення предметної області було виділено наступні сутності, які предст-авляють інформаційну цінність для задачі:

Видача книг - облік видання книг;

Читачі - всі зареєстровані читачі;

Книги - всі книги бібліотеки; Для виділення однакової інформації та оптимізації зберігання даних були виділені додаткові сутності:

Бібліотекарі -- облік бібліотекарів, що беруть участь у виданні;

Жанри - всі види жанрів, пов'язаних з книгами;

Видання - всі види типів видань книг;

Віковий рейтинг - всі види вікових рейтингів;

Зберігання - всі розташування книг; Концептуальне представлення бази даних -наведено на -малюнку- 4.1.

-Малюнок- 4.1 - Концептуальна модель баз-и- дан-их

-

4.1.2 Логічне проектування баз даних

Логічне проектування - створення схеми бази даних на основі конкретної моделі даних. У проєкті використовується реляційна база даних. Для кожної виділеної сутності необхідно продумати поля, включаючи первинні та зовніш-ні ключі. Також були продумані зв'язки таблиць. У даному випадку всі зв'язки доцільно створити як

” Цитування: 0,03%

id: 14

"один до багатьох".

У такого роду зв'язках рядок в першій таблиці може мати багато представлень в другій таблиці, але рядок у другій таблиці може мати тільки -один рядок в першій таблиці. З урахуванням даних бізнес-процесів та предмета дослідження була створена логічна модель. Логічне представлення бази даних наведено на рисунку 4.2.

РК - ПЕРВИННИЙ КЛЮЧ

-FK- - -ЦЕ КЛЮЧ-, -ВИКОРИСТОВУЄТЬСЯ ДЛЯ З'ЄДНАННЯ ДВОХ ТАБЛИЦЬ РАЗОМ.

-

-Малюнок- 4.2 - -Інфологічна- модель баз_ы данн_{ых}-

-

4.1.3 Фізичне проектування бази даних

Фізичне проектування - створення схеми бази даних для конкретної СУБД, яка включає всі необхідні таблиці, стовпці, зв'язки, властивості бази даних для фізичної реалізації. У цьому проєкті використовується [MySQL](#). Згідн-о з типами даних, зв'язками, логічною моделлю та іншими уточненнями з бізнес-процесів була реалізована фізична модель. Фізичне представлення таблиць бази даних наведено на -малюнку.-

-Малюнок- 4.3 - Датоло-гічна- модель баз_ы данн-их

-4.2 Проектування структури програми та паттерни

Проект [LibraryDB](#) складається з файлів вихідного коду:

[Form1.cs](#) - головна форма, на якій відображаються та редагуються дані таблиць, підготовлюються основні елементи управління;

[Form2.cs](#) !- довідкова форма для- відображення відомостей таблиць БД;

[DBExtraReqRequests.cs](#) - створення додаткових довідкових запитів;

[ViewDBObject.cs](#) - зв'язок функціоналу [Form1](#) з функціоналом нащадків [DBIntegration](#);

[DBIntegration.cs](#) та його нащадки - використання об'єктів-нащадків [IDBO-bject](#) у необхідних запитах;

[IDBObject.cs](#) та його нащадки - представлення об'єктів БД;

[IHuman.cs](#) - класифікація сутності

” Цитування: 0,01%	id: 15
"Людина"	
; Зв'язок файлів показано на малюнку 4.-4-.	
Для розробки онлайн бібліотеки можна використовувати різні патерни проектування, що допоможуть впоратися з різними завданнями. Ось деякі патерни, які можуть бути корисними у вашому додатку:	
Фабричний метод (Factory Method): Дозволяє створювати об'єкти різних типів книг чи користувачів залежно від контексту чи вхідних даних.	
Ось приклад коду на C# , що ілюструє реалізацію фабричного методу для створення об'єктів різних типів книг:	
Одинак (Singleton): Може бути використаний для створення єдиного об'єкту бібліотеки, який буде доступний у всій програмі.	
Метод GetInstance повертає той самий екземпляр класу Singleton кожного разу, коли він викликається. Перевірка <code>obj1 == obj2</code> демонструє, що ці об'єкти вказують на один і той же екземпляр класу Singleton .	
Цей шаблон може бути корисним, коли потрібно мати лише один екземпляр певного класу для всієї програми, наприклад, для збереження глобальних налаштувань або кешування об'єктів.	
Команда (Command): Дозволяє упаковувати запити на додавання, видалення чи редагування книг у окремі об'єкти, які можна передавати, відкладати чи відмінити.	
Спостерігач (Observer): Дозволяє створювати систему сповіщень про зміни у стані книг або користувачів.	
Ітератор (Iterator): Використовується для послідовного перегляду книг у бібліотеці.	
Кешування (Caching): Дозволяє зберігати книги, які часто запитуються, у кеші для підвищення швидкодії доступу.	
Модель-вид-контролер (Model-View-Controller , MVC): Допомагає розділити логіку програми, що обробляє бізнес-логіку (Модель), від візуального представлення даних (Вид), та управління взаємодією між ними (Контролер).	

4.3 Опис класів та методів застосування

Клас [Form1](#) містить поля для зберігання [ID](#) вибраного елемента в таблиці, таблицю, об'єкти класу [ViewDBObject](#) та [DBExtraReqRequests](#). Клас містить методи, які обробляють всі взаємодії з інтерфейсом та передають дані для запитів. Клас [DBExtraReqRequests](#) містить методи для виконання відповідних додаткових запитів при натисканні кнопок та виводить їх на внутрішню таблицю на вкладці

” Цитування: 0,01%	id: 16
--------------------	--------

"Запити".

Клас [Form2](#) містить методи обробки кнопки

” Цитування: 0,01%	id: 17
--------------------	--------

"Довідка"

на вкладці

” Цитування: 0,01%	id: 18
--------------------	--------

"Запити",

які виводять дані по всіх таблицях та шукають дані всередині таблиць. Клас [ViewDBObject](#) містить поле для обробки вибраної таблиці та множину масивів рядків для заповнення заголовків текстових полів на вкладці

” Цитування: 0,01%	id: 19
--------------------	--------

"Редагування".

Клас містить методи, які виконують різні перетворення для передачі даних до запитів. Абстрактний клас [DBIntegration](#) містить поле для підключення до БД.

Класи [DBIntegration](#) та його нащадки містять методи для передачі відповідних нащадків [DBObject](#) для виконання запитів до БД. Інтерфейс [DBObject](#) містить обов'язкове

визначення [ID](#) та методи конвертації масиву рядків в об'єкт та назад. Наслідники класу [IDBObject](#) перевизначають поле [ID](#) та доповнюються полями, які співпадають з колонками відповідних таблиць.

4.4 Опис [SQL](#)-запитів- до бази даних системи

У розробленій системі онлайн бібліотеки активно використовуються [SQL](#)-запити для реалізації взаємодії з базою даних. Нижче наведено основні типи запитів та їх функціональне призначення:

Отримання списку всіх книг:

Цей запит використо-вується для відображення всіх доступних книг на головній сторінці додатку.

Пошук книг за автором або назвою:

Цей запит дозволяє користувачам здійснювати пошук літератури за назвою або ім'ям автора.

Додавання нової книги:

Застосовується адміністраторами б-ібліотеки для поповнення бази даних новими записами. Формування звіту про боржників:

Цей запит визначає користувачів, які прострочили термін повернення книг.

Оновлення інформації про книгу:

Видалення книги з бази:

Всі [SQL](#)-запити реалізовані з урахування-м перевірки введених даних та захисту від [SQL](#)-ін'єкцій. Для цього використовується механізм параметризованих запитів, що підтримується у середовищі розробки через [ORM](#) або [ADO.NET](#).

-[SQL](#) -запити- клас-у- [DBInteractionBookLending](#) (-повний лістинг файлу- в -додатку- Б1):

-Форматування вибірки всій таблиці -[Book](#)-_-[lending](#):-

” Цитування: **0,02%** id: 20
"CALL SelectBookLending();"

Процедура [SelectBookLending](#):

-[BEGIN](#)

[SELECT](#) [lending_id](#) AS "ID -выдачи-", [CONCAT_WS](#)("

” Цитування: **0,08%** id: 21
", [Librarian.surname](#), [Librarian.name](#), [Librarian.patronymic](#)) AS "-

Бібліотекар-

” Цитування: **0,01%** id: 22
", [CONCAT_WS](#)("

” Цитування: **0,08%** id: 23
", [Reader.surname](#), [Reader.name](#), [Reader.patronymic](#)) AS "-

Читач-

” Цитування: **0,04%** id: 24
", [Books.book_name](#) AS "-

Книга-

” Цитування: **0,03%** id: 25
", [lend_amount](#) AS "-

Кількість-

” Цитування: **0,07%** id: 26
", [DATE_FORMAT](#)([date_of_issue](#), '%d.%m.%Y') AS "-


```
Дата -видачі-", DATE_FORMAT(return_date, '%d.%m.%Y') AS "-Дата -повернення-" FROM
Book_lending
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
INNER JOIN Books ON Books.book_id = Book_lending.book_id
-ORDER BY return_date;
END
```

-Зформовання вибірка запису таблиці -Book-_-lending- по -ID-:

```
"CALL SelectBookLendingByID({ID});":
```

Процедура- SelectBookLendingByID-:

```
-BEGIN
SELECT lending_id, CONCAT_WS(" ", Librarian.surname, Librarian.name, Librarian.patronymic),
CONCAT_WS(" ", Reader.surname, Reader.name, Reader.patronymic), Books.book_name,
lend_amount, DATE_FORMAT(date_of_issue, '%d.%m.%Y'), DATE_FORMAT(return_date, '-
%d.%m.%Y') FROM Book_lending
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
INNER JOIN Books ON Books.book_id = Book_lending.book_id
-WHERE lending_id = ID;
END
```

Форматованная -вибірка запису- таблиці -Book-_-lending- по -запиту-: "-CALL BookLendingSearch-('{-query-}')

Процедура- BookLendingSearch:

```
SELECT lending_id AS "ID -Видачі-", CONCAT_WS(" "
```

” Цитування: **0,08%** id: **27**
", Librarian.surname, Librarian.name, Librarian.patronymic) AS "-

Бібліотекар-

” Цитування: **0,01%** id: **28**
", CONCAT_WS(" "

” Цитування: **0,08%** id: **29**
", Reader.surname, Reader.name, Reader.patronymic) AS "-

Читач-

” Цитування: **0,04%** id: **30**
", Books.book_name AS "-

```
Книга-", lend_amount AS "-Кількість-", DATE_FORMAT(date_of_issue, '%d.%m.%Y') AS "-Дата
видачі-", DATE_FORMAT(return_date, '%d.%m.%Y') AS "-Дата відання-" FROM Book_lending
INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id
INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id
INNER JOIN Books ON Books.book_id = Book_lending.book_id
WHERE LOCATE(searchQuery, CONCAT_WS(" ", l-ending_id, Librarian.surname, Librarian.name,
Librarian.patronymic, Reader.surname, Reader.name, Reader.patronymic, Books.book_name,
lend_amount, date_of_issue, DATE_FORMAT(date_of_issue, '%d.%m.%Y'),
DATE_FORMAT(return_date, '%d.%m.%Y'))) = 1 ORDER BY re-tur_n_date;
```

-Додавання- запис-у- в таблицю -Book-_-lending-:

```
"CALL BookLendingInsert('{item.librarian}', '{item.reader}', '{item.book}', '{item.lendAmount}',
'{item.returnDate}')

```

-Процедура -BookLendingInsert-:

```
-BookLendingInsert:BEGIN
DECLARE librarianID INT;
DECLARE readerID INT;
DECLARE bookID INT;
SELECT lib_id INTO librarianID FROM Librarian WHERE LOCATE(librarianInfo, CONCAT_WS(" "
```

” Цитування: **0,18%** id: **31**
", lib_id, surname, name, patronymic)) = 1;

```
SELECT reader_id INTO re-aderID FROM Reader WHERE LOCATE(readerInfo, CONCAT_WS("
", reader_id, surname, name, patronymic)) = 1;
SELECT book_id INTO bookID FROM Books WHERE LOCATE(bookInfo, CONCAT_WS(" ",
book_name, ISBN)) = 1;
INSERT INTO Book_lending (lending_id, librarian_id, reader_id, book_id, lend_amount,
date_of_issue, return_date) VALUES (NULL, librarianID, readerID, bookID, lendAmount, NOW(),
returnDate);
```

-Змінення запису- в -таблиці -Book- -lending-:

```
-"CALL BookLendingChange('{currentID}', '{item.ID}', '{item.librarian}', '{item.reader}',
'{item.book}', '{item.lendAmount}', '{item.dateOfIssue}', '{item.returnDate}')"
-Процедура- BookLendingChange:
```

```
BEGIN
```

```
DECLARE librarianID INT;
```

```
DECLARE readerID INT;
```

```
DECLARE bookID INT;
```

```
SELECT lib_id INTO librarianID FROM Librarian WHERE LOCATE(librarianInfo, CONCAT_WS("

```

” Цитування: **0,18%**

id: 32

```
", lib_id, surname, name, patronymic)) = 1;
```

```
SELECT rea-der_id INTO readerID FROM Reader WHERE LOCATE(readerInfo, CONCAT_WS("

```

```
", reader_id, surname, name, patronymic)) = 1;
```

```
SELECT book_id INTO bookID FROM Books WHERE LOCATE(bookInfo, CONCAT_WS(" ",
book_name, ISBN)) = 1;
```

```
UPDATE Book_lending SET lendi-ng_id = lendingID, librarian_id = librarianID, reader_id =
readerID, book_id = bookID, lend_amount = lendingAmount, date_of_issue = issueDate,
return_date = returnDate WHERE lending_id = currentLendingID;
```

-Видалення- запис-у- в таблиці -Book- -lending-:

```
-"DELETE FROM Book_lending WHERE Book_lending.lending_id = {ID}"
```

-Триггер-и:

Триггер -вчитання кількості виданих- книг при -додаванні запису видачі-:

```
-CREATE TRIGGER `lending_after_insert` AFTER INSERT ON `Book_lending`
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount - new.lend_amount) WHERE
Books.book_id = NEW.book_id
```

-Триггер-и змінення кількості виданих книг- до -та після змінення запису видачі
соответсвенно-:

```
-CREATE TRIGGER `lending_before_update` BEFORE UPDATE ON `Book_lending`
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount + OLD.lend_amount)
WHERE Books.book_id = OLD.book_id
```

```
CREATE TRIGGER `lending_after_update` AFTER UPDATE ON `Book_lending`
```

```
-FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount - NEW.lend_amount)
WHERE Books.book_id = NEW.book_id
```

-Триггер -додавання видачі книг при видаленні запису- о в-дач-і-:

```
-CREATE TRIGGER `lending_before_delete` BEFORE DELETE ON `Book_lending`
FOR EACH ROW UPDATE Books SET Books.amount = (Books.amount + OLD.lend_amount)
WHERE Books.book_id = OLD.book_id
```

-Виборка читачів-, -які- не -повернули- книги -своєчасно-:

Процедура SelectDebtors:

```
-SELECT CONCAT_WS("

```

” Цитування: **0,08%**

id: 33

```
", Reader.surname, Reader.name, Reader.patronymic) AS "-
```

Должник-

”

"", [Цитування: 0,07%](#) `DATE_FORMAT(return_date, '%d.%m.%Y') AS "` id: 34

Дата в^идачі-

"", [Цитування: 0,07%](#) `DATE_FORMAT(return_date, '%d.%m.%Y') AS "` id: 35

"", `DATE_FORMAT(return_date, '%d.%m.%Y') AS "`

Дата возврата-", `TIMESTAMPDIFF(DAY, return_date, NOW()) AS "-Задолженность- (-дн-.)" FROM Book_lending`

`INNER JOIN Librarian ON Librarian.lib_id = Book_lending.librarian_id`

`INNER JOIN Reader ON Reader.reader_id = Book_lending.reader_id`

`INNER JOIN Books ON Books.book_id = Book_lending.book_id`

`WHERE TIMESTAMPDIFF(DAY, return_date, NOW(-)) 0`

`ORDER BY TIMESTAMPDIFF(DAY, return_date, NOW()) DESC;`

-

5-. РОЗДІЛ 5.- ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

5.1 Апаратні та програмні засоби створення та експлуатації програми

Для роботи додатку потрібне наступне апаратне забезпечення:

процесор з тактовою частотою 1.0 ГГц;

ОЗУ об'ємом не менше 1 Гб;

вільне місце на жорсткому дис-ку 300 Мб;

відеокарта з об'ємом пам'яті не менше 514 Мб;

операційна система [Windows 10](#);

наявність засобів введення-виведення: миша, клавіатура, монітор;

Програмні вимоги до додатку. На комп'ютері повинно бути встановлене наступне програмне забезпечення:

оп-ераційна система - [Windows 10](#);

будь-який браузер;

платформа [Microsoft .NET 6.0](#);

утиліта - [Open Server Panel](#) з включеним модулем [MySQL](#);

5.2 Інструкція користувача

Для роботи додатку необхідно: Встановити на комп'ютері програмну середу [.NET FRAMEWORK 6.0](#) з о-фіційного сайту Майкрософт. Встановити утиліту [Open Svrer Panel](#) з офіційного сайту [Open Server](#). Налаштувати утиліту на роботу з СУБД [MySQL](#): налаштування - модулі - [MySQL-8.0-Win10](#). Після цього перезапустити програму. Извлечь из папки файл [-Library--.sql-](#) в любую удобную директорию.

-X -из- [Open Server Panel PHPMyAdmin](#).

-Войти в СУБД (логин: [-root-](#), пароль: пустая строка).

Натисніть

"", [Цитування: 0,01%](#) `"Импорт"` id: 36

"Импорт"

та вибрати [Library.sql](#). Налаштувати користувача [root](#): облікові записи - [root](#) - змінити пароль - встановити пароль

"", [Цитування: 0,01%](#) `"root"` id: 37

"root".

Можна закрити всі непотрібні вікна. Перейти до папки, в якій знаходиться файл [LibraryDB.e-xe](#), і зробити на ньому подвійний клік мишкою (примітка: [Open Server Panel](#) повинен бути в стані

"", [Цитування: 0,01%](#) `"Запустити"` id: 38

"Запустити"

). На малюнку 5.1 виділено файл додатку та імпортована база даних.

Рисунок 5.1 - -Файлова- система -додатку

-

5.3 Опис контрольних прикладів

Для запуску додатку необхідно натиснути двічі лівою кнопкою миші на файл [LibraryDB.exe](#), після чого відкриється головна сторінка додатку (рис. А1). Для початку роботи з таблицями

їх потрібно завантажити. Для цього необхідно натиснути на випадаючий список таблиць та вибрати потрібну таблицю (наприклад,

” Цитирования: 0,02% id: 39

"Видача книг"

). Після цього вікно оновиться та завантажить дані (рис. А2). Користувач може скористатися пошуком у вибраній таблиці. Для цього необхідно натиснути на текстове поле-

” Цитирования: 0,01% id: 40

"Пошук:"

та ввести

натиснути клавішу **Enter** після введення запиту. У вкладці

” Цитирования: 0,01% id: 41

"Таблиця"

відображатимуться результати пошуку (рис. А8). При натисканні на кнопки

” Цитирования: 0,01% id: 42

"Додати",

” Цитирования: 0,01% id: 43

"Змінити",

” Цитирования: 0,01% id: 44

"Видалити"

на вкладці будь-який результат операції оновить таблицю на вкладці

” Цитирования: 0,01% id: 45

"Таблиця",

а вкладка

” Цитирования: 0,01% id: 46

"Редагування"

очиститься. Для додавання запису необхідно перейти на вкладку

” Цитирования: 0,01% id: 47

"Редагування".

Потім можна ввести необхідні дані та натиснути кнопку

” Цитирования: 0,01% id: 48

"Додати"

(рис. А3). Після цієї операції таблиця на вкладці

” Цитирования: 0,01% id: 49

"Таблиця"

оновиться, а вкладка

” Цитирования: 0,01% id: 50

"Редагування"

очиститься. Для зміни або видалення запису необхідно двічі клацнути на відповідну запис та дані будуть додані на вкладку

” Цитирования: 0,01% id: 51

"Редагування"

(рис. А5).

У користувача на вкладці

” Цитирования: 0,01% id: 52

"Редагування"

є можливість викликати довідкове вікно, що надасть необхідну інформацію для введення з

можливістю пошуку (рис. A4). Для копіювання необхідної запису потрібно вибрати її клацанням миші та натиснути комбінацію клавіш **Ctrl** + **C**. Для вставки даних - **Ctrl** + **V**. Для зміни необхідно оновити відповідні текстові поля та -натиснути кнопку

Цитування: 0,01% id: **53**
"Змінити"

(рис. A6). Для видалення необхідно натиснути кнопку

Цитування: 0,01% id: **54**
"Видалити"

(рис. A6). Для виконання додаткових запитів потрібно натиснути на вкладку

Цитування: 0,01% id: **55**
"Запити"

та вибрати необхідний запит, після чого таблиця на вкладці

Цитування: 0,01% id: **56**
"Запити"

покаже результа-ти (рис. A7). Для виходу з додатку потрібно закрити вікно.

ВИСНОВОК

Проектування програмного забезпечення для онлайн бібліотеки - це складний та багатогранний процес, що вимагає уважного аналізу вимог, ефективного архітектурного планування та гнучкості п-ід час реалізації.

Під час розробки дипломної я виконав основні етапи цієї роботи які включають в себе детальний огляд актуальних аспектів, визначення цілей проєкту, встановлення вимог користувачів та технічного завдання, обґрунтування методів проектування та архітектурних рішень.

Створення програмного продукту, спрямованого на розвиток онлайн бібліотеки, вимагає уваги до деталей та активної участі розробників у процесі роботи над проєктом.

Ключовими аспектами успішного проектування є не лише технічні знання, а й уміння ефективно спілкуватися в команді, гнучко адаптуватися до змін та активно враховувати вимоги користувачів.

Також була проведена розробка технічного завдання яке було реалізовано паралельно, визначення архітектурних вирішень, вивчення та застосування методів проектування та управління - усе це необхідно для успішної розробки програмного продукту, який відповідатиме вимогам ринку та задовольнятиме потреби користувачів. Адаптивність, відкритість до змін та постійне удосконалення - ключові аспекти-, що дозволять створити ефективну та користувацьки орієнтовану онлайн бібліотеку.

-Також у- результаті вирішення -технічної- задачі було спроектовано базу даних та систему, яка автоматизує процес роботи бібліотеки. Проект дозволяє зберігати та шукати необхідні дані, а також взаємодіяти з ними через зручний додаток. Додаток дозволяє обробляти дані будь-якого рядка в БД: додавати,- змінювати, видаляти та шукати рядки в усіх наданих таблицях. Додаток також може автоматично (з перевіркою) віднімати кількість тих книг, які видавали читачам, та додавати відповідну кількість до повернутих книг, а також знаходити читачів, які молодші за 1-8 років, читачів, які не повернули книги вчасно та книги, які закінчились на складі. Додаток **LibraryDB** може використовуватись приватними та муніципальними бібліотеками малого та середнього рівня для автоматизації ведення обліку діяльності своєї бібліотеки.- У майбутньому планується розширення функціоналу додатку та оптимізація його роботи з великими обсягами даних. З урахуванням -проведеної- роботи можна стверджувати, що поставлені цілі та задачі в -проектувальному та -технічному завданні були виконані.

