Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва інституту)

Кафедра комп‘ютерних та інформаційних технологій і систем

(повна назва кафедри)

Пояснювальна записка до дипломного проекту (роботи)

бакалавра

(ступеню вищої освіти)

на тему

Автоматизована інформаційна система

обліку діяльності шкільної бібліотеки

Виконав: студент 4 курсу, групи 401-ТН

спеціальності

122 Комп’ютерні науки \_

(шифр і назва напряму)

Міщенко О. О.

(прізвище та ініціали)

Керівник Гафіяк А. М.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Полтава – 2021 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

**спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Автоматизована інформаційна система обліку діяльності шкільної бібліотеки»**

**Студента групи 401-ТН Міщенка Олександра Олексійовича**

Керівник роботи

доктор педагогічних наук,

доцент Гафіяк А.М.

Завідувач кафедри

кандидат технічних наук,

доцент Головко Г.В.

Полтава – 2021

**РЕФЕРАТ**

Пояснювальна записка містить: 70с., 24 рис., 8 джерел, 2 додатки.

**Об’єкт дослідження** – інформаційна система шкільної бібліотеки.

**Предмет дослідження –** інструментальні засоби розробки програмного забезпечення.

**Мета кваліфікаційної роботи –** розроблення програми для автоматизації обліку діяльності шкільної бібліотеки.

**Ключові слова:** автоматизована бібліотечна інформаційна система, шкільна бібліотека, прикладна програма, база даних, каталог, пошук.

**ANNOTATION**

Explanatory note contains: 70p., 24 pic., 8 sources, 2 appendices.

**The object of research** – information system of the school library.

**The subject of research** – software development tools.

**The purpose of the qualification work** – develop a program for automating the accounting of school library activities.

**Keywords:** automated library information system, school library, application program, database, catalog, search.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 6](#_Toc74066009)

[ВСТУП 7](#_Toc74066010)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ 8](#_Toc74066011)

[1.1 Опис предметної області 8](#_Toc74066012)

[1.2 Аналіз існуючих рішень 8](#_Toc74066013)

[1.3 Постановка задачі 9](#_Toc74066014)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ДІЯЛЬНОСТІ ШКІЛЬНОЇ БІБЛІОТЕКИ 10](#_Toc74066015)

[2.1 Моделювання функціоналу та структури за допомогою   
UML-діаграм 10](#_Toc74066016)

[2.2 Проектування бази даних 21](#_Toc74066017)

[РОЗДІЛ 3 РОЗРОБКА АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ДІЯЛЬНОСТІ ШКІЛЬНОЇ БІБЛІОТЕКИ 22](#_Toc74066018)

[3.1 Вибір та обґрунтування технологій для розробки програмного забезпечення 22](#_Toc74066019)

[3.1.1 Вибір та обґрунтування використання мови C# для розроблення програмного забезпечення. 22](#_Toc74066020)

[3.1.2 Вибір та обґрунтування використання WPF для розроблення програмного інтерфейсу. 22](#_Toc74066021)

[3.1.3 Вибір та обґрунтування використання MySQL у якості системи керування базами даних. 23](#_Toc74066022)

[3.2 Розробка бази даних 23](#_Toc74066023)

[3.3 Програмна реалізація АБІС 27](#_Toc74066024)

[РОЗДІЛ 4 ТЕСТУВАННЯ 33](#_Toc74066025)

[4.1 Вибір виду тестування 33](#_Toc74066026)

[4.2 План тестування 33](#_Toc74066027)

[ВИСНОВКИ 39](#_Toc74066028)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 40](#_Toc74066029)

[ДОДАТОК А СКРИПТ ГЕНЕРАЦІЇ ТАБЛИЦЬ БАЗИ ДАНИХ 41](#_Toc74066030)

[ДОДАТОК Б КОД ПРОГРАМИ 44](#_Toc74066031)

# **ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ**

**БД** – база даних.

**ОС** – операційна система.

**ПК** – персональний комп’ютер.

**ІС** – інформаційна система.

**АБІС** – автоматизована бібліотечна інформаційна система.

**UML –** Unified Modeling Language.

**CLR –** Common Language Runtime.

**WPF** – Windows Presentation Foundation.

**ЛКМ** – ліва кнопка миші.

**Комбобокс** – елемент інтерфейсу користувача, представляє собою випадаючий список.

# **ВСТУП**

В наш час інформаційні технології являються невід’ємною складовою повсякденного життя. Інформатизація відіграє важливу роль   
в найрізноманітніших сферах діяльності людини, у тому числі навчальному процесі. Існує багато систем які вирішують індивідуальні задачі в сфері   
освіти, а сукупність цих систем та програмних засобів, автоматизують процес навчання. На те, що раніше людина витрачала декілька днів комп’ютер може виконати за лічені хвилини.

Безпосередньо в навчальному процесі автоматизуються майже всі процеси, починаючи з контролю відвідувань, закінчуючи відстеженням успішності в навчанні. Одним із таких процесів є облік діяльності бібліотеки.

Аналіз існуючого програмного забезпечення показав недостачу готових рішень в сфері обліку діяльності бібліотек. Саме тому метою роботи є створення програми, яка дозволить автоматизувати процес обліку діяльності шкільної бібліотеки.

Головними завданнями кваліфікаційної роботи є розробка структури автоматизованої інформаційної системи, програмна реалізація цієї структури та тестування з подальшим введенням в експлуатацію.

# **РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ**

* 1. **Опис предметної області**

Предметною областю кваліфікаційної роботи є розробка інформаційної системи для обліку діяльності шкільної бібліотеки.

Як правило, такі системи забезпечують наступний функціонал:

* адміністрування,
* каталогізація,
* читацький пошук,
* обслуговування читачів.
  1. **Аналіз існуючих рішень**

Розглянемо декілька вже існуючі рішень для ведення обліку діяльності бібліотеки.

1С: Бібліотека – програма розроблена на платформі 1С. Дана система має можливість штрих-кодування книжкового фонду, документообіг бібліотеки в тому числі формуляри читачів ведуться в електронному вигляді, є можливість віддаленого доступу до ІС. Дана програма більшою мірою призначена для використання у великих бібліотеках і погано підходить для шкільних інформаційно-бібліотечних центрів.

ІРБІС64 – являє собою типове інтегроване рішення в області автоматизації бібліотечних технологій і призначена для використання в бібліотеках будь-якого типу і профілю в якості одного з основних компонентів. АБІС має модульну структуру, що дозволяє при необхідності розширити основний функціонал. Головною перевагою цієї системи є низькі системні вимоги.

МегаПро – комплексне рішення автоматизації діяльності бібліотеки, побудови бібліотечних мереж та організації доступу до них на основі веб-технологій. Система, як і в «ІРБІС64», має модульну структуру, робота з модулями реалізована через веб-інтерфейс, що дозволяє виконувати її як в локальній мережі, так і віддалено через мережу Інтернет, тому всі основні інструменти управління каталогами та базами даних доступні в браузері.

МАРК-SQL ­– функціонально мало чим відрізняється від попередньо представлених систем, а саме: створення електронного каталогу, формування та друк вихідних форм, обслуговування читачів, ведення статистичної та управлінської звітності, можливість доступу через веб-інтерфейс. Відмінною особливістю цієї ІС є її незалежність від використовуваної в організації бази даних, що не прив'язує процеси в бібліотеці до якогось одного продукту і спрощує процес перенесення даних з інших бібліотечних систем. Недоліком системи можна назвати високі системні вимоги порівняно з іншими АБІС [1].

* 1. **Постановка задачі**

Завданнями кваліфікаційної роботи є програмна реалізація автоматизованої інформаційної системи шкільної бібліотеки, а також тестування цієї системи з подальшим введенням в експлуатацію.

Програма повинна забезпечувати наступний функціонал:

1. Реєстрація в системі.
2. Розмежування прав доступу.
3. Перегляд каталогу книг.
4. Журнал обліку видачі та повернення літератури.
5. Внесення книг в базу даних.
6. Видалення списаних книг з бази даних.
7. Пошук літератури в базі даних.

# **РОЗДІЛ 2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ДІЯЛЬНОСТІ ШКІЛЬНОЇ БІБЛІОТЕКИ**

* 1. **Моделювання функціоналу та структури за допомогою   
     UML-діаграм**

UML – це уніфікований графічний мова моделювання призначена для опису, візуалізації, проектування та документування об’єктно орієнтованих систем. UML покликаний підтримувати процес моделювання прикладних систем на основі об’єктно орієнтованого підходу, організовувати взаємозв'язок концептуальних і програмних понять та відображати проблеми масштабування складних систем. Моделі створені за допомогою UML використовуються на всіх етапах життєвого циклу програмного засобу, починаючи з бізнес-аналізу і закінчуючи підтримкою системи [2].

Проаналізуємо функції, які можуть виконуватися користувачами системи, та зобразимо їх у вигляді діаграми варіантів використання (рис. 2.1).

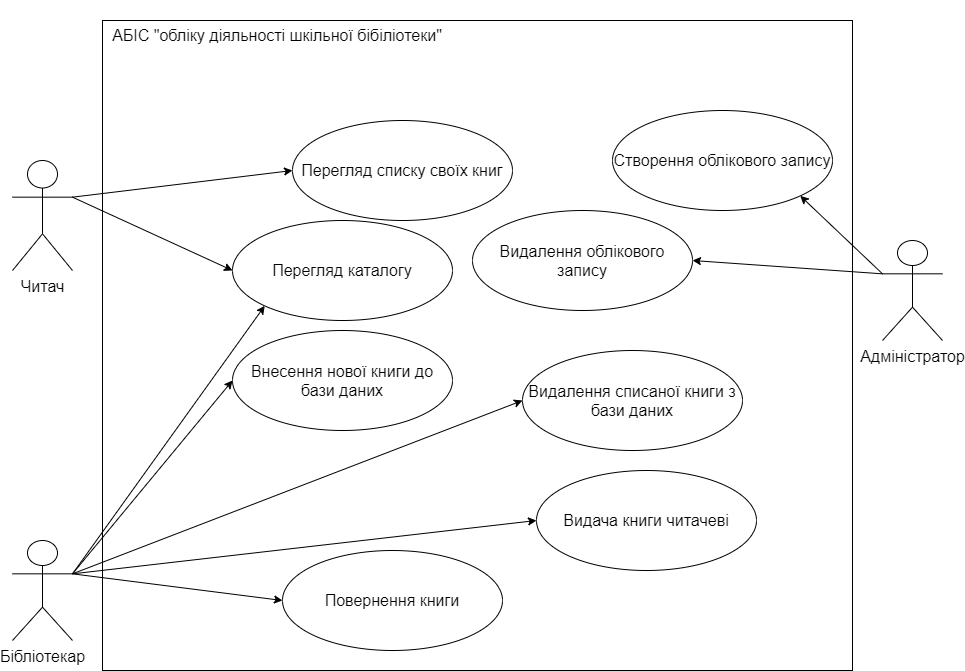


Рисунок 2.1 – Варіантів використання для АБІС «обліку діяльності шкільної бібліотеки»

В системі передбачені три типи користувачів:

1. Читач. Має доступ до перегляду каталогу та списку книг які знаходяться в його користуванні.
2. Бібліотекар. Може переглядати каталог, заносити нові книги в базу даних та видаляти списані книги з неї, а також видавати літературу читачу і приймати повернені книги.
3. Адміністратор. Має можливість створення та видалення облікових записів користувачів.

Щоб спроектувати інтерфейсу користувача були розроблені діаграми діяльності для: адміністратора, бібліотекара та читача.

Задачею адміністратора є реєстрація та видалення облікових записів користувачів системи.

На рисунку 2.2 зображена діаграма діяльності для процесу «Реєстрація користувача» .

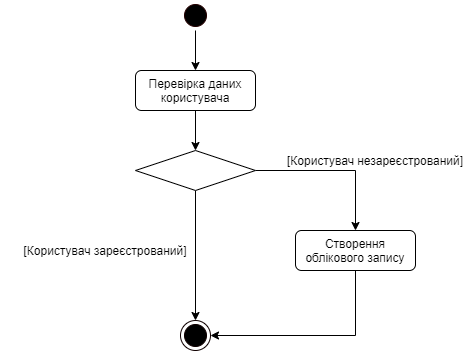


Рисунок 2.2 – Діаграма діяльності для процесу «Реєстрація користувача»

Як видно на діаграмі, перед початком реєстрації адміністратор повинен перевірити дані користувача на наявність в базі даних у випадку, якщо облікового запису не знайдено, необхідно створити новий обліковий запис.

На рисунку 2.3 зображена діаграма діяльності для процесу «Видалення облікового запису користувача»

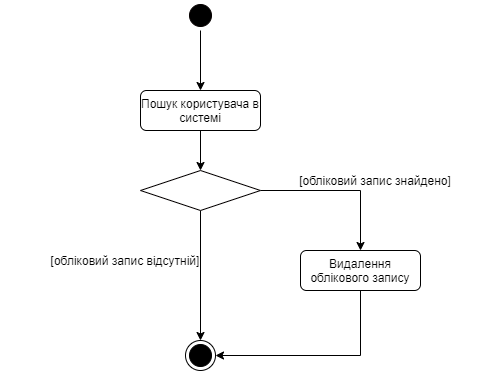


Рисунок 2.3 – Діаграма діяльності для процесу «Видалення облікового запису користувача»

Спочатку адміністратор повинен перевірити чи існує такий обліковий запис, після чого у випадку знаходження збігів виконує видалення облікового запису користувача.

Задачами бібліотекара є внесення книги в базу даних, видалення списаних книги з неї, а також прийом і видача літератури читачеві.

На рисунку 2.4 зображена діаграма діяльності для процесу «Внесення книги в базу даних»

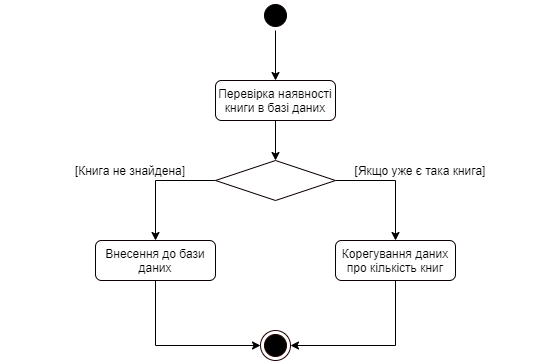


Рисунок 2.4 – Діаграма діяльності для процесу «Внесення книги до бази даних»

Щоб уникнути дублювання записів, перед внесенням до БД, бібліотекар повинен перевірити наявність цієї книги в БД у випадку наявності такої книги необхідно скорегувати загальну кількість таких книг, якщо запис про таку книгу відсутній данні про неї потрібно занести в базу даних.

На рисунку 2.5 зображена діаграма діяльності для процесу «Видалення списаної книги з бази даних»

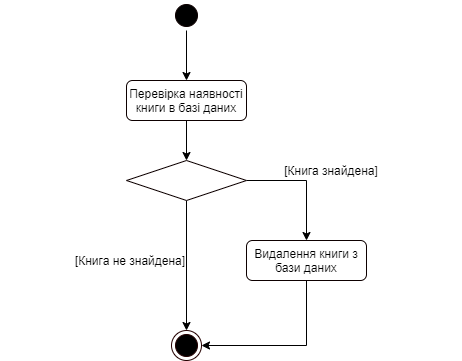


Рисунок 2.5 – Діаграма діяльності для процесу «Видалення списаної книги з бази даних»

Процес видалення починається з перевірки наявності такої книги в базі даних у разі знаходження збігів бібліотекар виконує операцію по видаленню цієї книги, у випадку якщо така книга незнайдена процес завершуєтеся.

На рисунку 2.6 зображена діаграма діяльності для процесу «Видача книги»

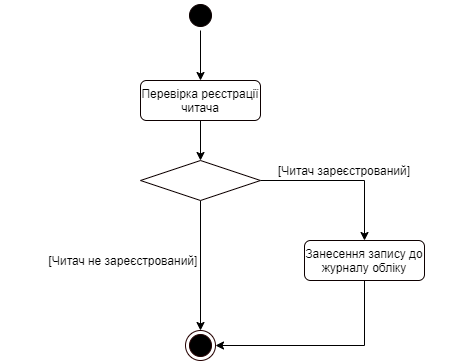


Рисунок 2.6 – Діаграма діяльності для процесу «Видача книги читачу»

Перед тим як здійснити видачу книги потрібно перевірити дані читача у разі, якщо читач має обліковий запис бібліотекар вносить запис про видачу до журналу обліку, в іншому випадку процес завершуються.

На рисунку 2.7 зображена діаграма діяльності для процесу «Повернення книги»

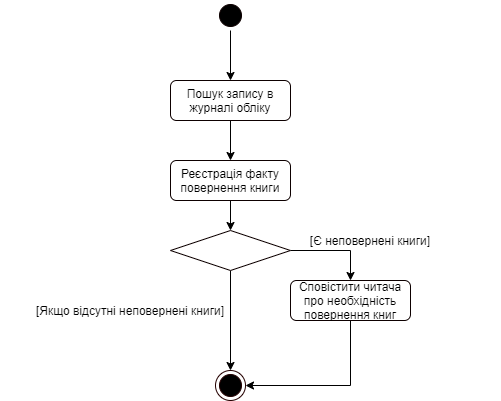


Рисунок 2.7 – Діаграма діяльності для процесу «Повернення книги»

Спочатку бібліотекар проводить пошук запису про видачу в обліковому журналі, після чого реєструє факт повернення книги, якщо у читача є книги які необхідно повернути співробітник сповіщає читача про цю необхідність.

Функціями які може користуватися читач є: перегляд каталогу та списку книг які знаходяться в його користуванні.

На рисунку 2.8 зображена діаграма діяльності для процесу «Перегляд списку книг які знаходяться в користуванні»

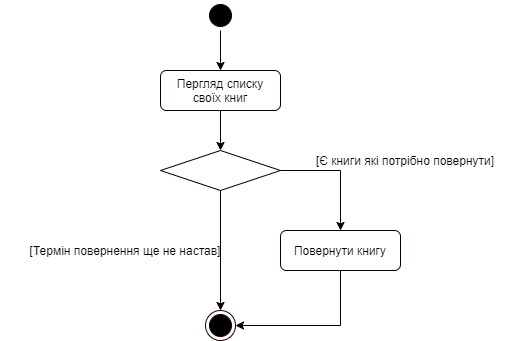


Рисунок 2.8 – Діаграма діяльності для процесу «Перегляд списку книг які знаходяться в користуванні»

Щоб не пропустити строк здачі підручника читач може переглядати список книг які знаходяться в його користуванні, у випадку, якщо строк здачі вже настав йому необхідно звернутися до бібліотекара для проведення процедури повернення.

Опис логіки використання системи зображено на діаграмах послідовності для сценаріїв: реєстрація користувача, видача книги, повернення книги, перегляд списку книг які знаходяться в користуванні читача.

На рисунку 2.9 зображена діаграма послідовності для сценарію «Реєстрація користувача». Опис повідомлень діаграми послідовності занесено до таблиці 2.1.

Таблиця 2.1 – Опис повідомлень діаграми послідовності «Реєстрація користувача»

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер повідомлення** | **Об'єкт - відправник повідомлення** | **Об'єкт - одержувач повідомлення** | **Назва** |
| 1 | Користувач | Адміністратор | Запит на реєстрацію |
| 2 | Адміністратор | Інтерфейс програми | Вхід в систему |
| 3 | Інтерфейс програми | База даних | Логін і пароль |
| 4 | База даних | База даних | Перевірка даних |
| 5 | База даних | Інтерфейс програми | Авторизація в програмі |
| 6 | Адміністратор | Інтерфейс програми | Обрати «Створити користувача» |
| 7 | Адміністратор | Інтерфейс програми | Обрати тип облікового запису |
| 8 | Адміністратор | Інтерфейс програми | Ввести дані в форму |
| 9 | Адміністратор | Інтерфейс програми | Виконати запит на створення |
| 10 | Адміністратор | Інтерфейс програми | Збереження введених даних |
| 11 | Адміністратор | Користувач | Дані для входу в систему |

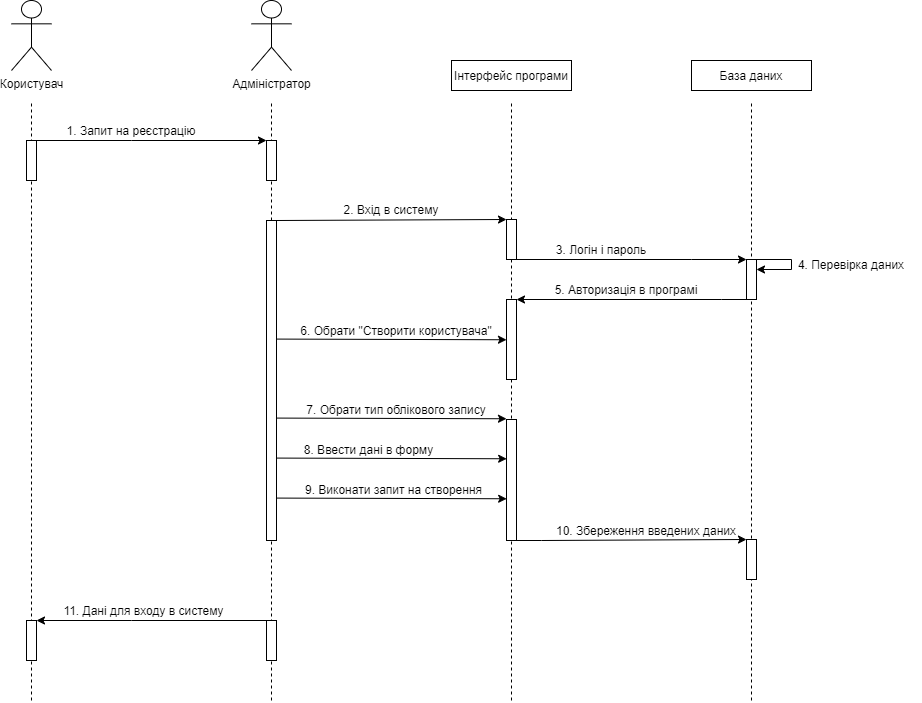


Рисунок 2.9 – Діаграма послідовності для сценарію «Реєстрація користувача»

На рисунку 2.10 зображена діаграма послідовності для сценарію «Видача книги». Опис повідомлень діаграми послідовності занесено до таблиці 2.2.

Таблиця 2.2 – Опис повідомлень діаграми послідовності «Видача книги»

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер повідомлення** | **Об'єкт - відправник повідомлення** | **Об'єкт - одержувач повідомлення** | **Назва** |
| 1 | 2 | 3 | 4 |
| 1 | Бібліотекар | Інтерфейс програми | Вхід в систему |
| 2 | Інтерфейс програми | База даних | Логін і пароль |
| 3 | База даних | База даних | Перевірка даних |
| 4 | База даних | Інтерфейс програми | Авторизація в програмі |
| 5 | Бібліотекар | Інтерфейс програми | Обрати «Видача книги» |
| 6 | Бібліотекар | Інтерфейс програми | Внесення даних в форму |

Продовження таблиці 2.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 7 | Бібліотекар | Інтерфейс програми | Запит на видачу |
| 8 | Інтерфейс програми | База даних | Збереження введених даних |

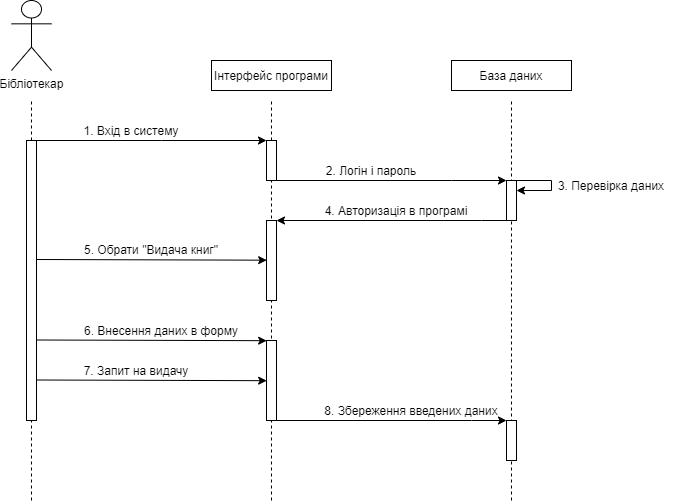


Рисунок 2.10 – Діаграма послідовності для сценарію «Видача книги»

На рисунку 2.11 зображена діаграма послідовності для сценарію «Повернення книги». Опис повідомлень діаграми послідовності занесено до таблиці 2.3.

Таблиця 2.3 – Опис повідомлень діаграми послідовності «Повернення книги»

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер повідомлення** | **Об'єкт - відправник повідомлення** | **Об'єкт - одержувач повідомлення** | **Назва** |
| 1 | 2 | 3 | 4 |
| 1 | Бібліотекар | Інтерфейс програми | Вхід в систему |
| 2 | Інтерфейс програми | База даних | Логін і пароль |

Продовження таблиці 2.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 3 | База даних | База даних | Перевірка даних |
| 4 | База даних | Інтерфейс програми | Авторизація в програмі |
| 5 | Бібліотекар | Інтерфейс програми | Обрати «Змінити статус» |
| 6 | Бібліотекар | Інтерфейс програми | Обрати користувача |
| 7 | Бібліотекар | Інтерфейс програми | Обрати книгу |
| 8 | Бібліотекар | Інтерфейс програми | Обрати новий статус |
| 9 | Інтерфейс програми | База даних | Збереження змін |

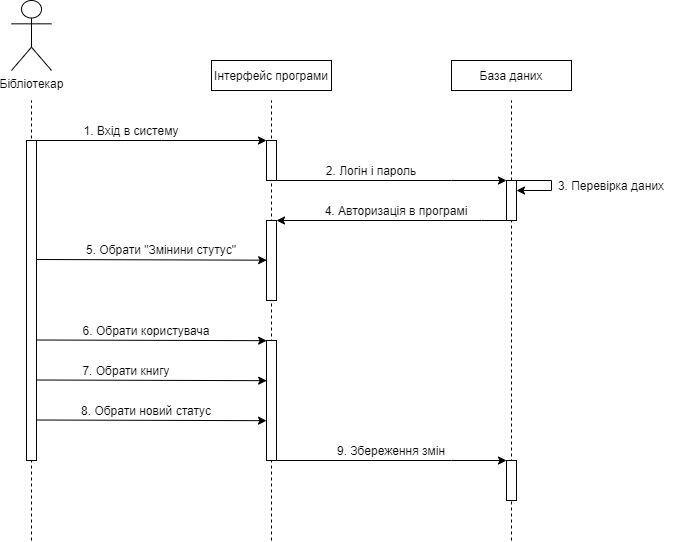


Рисунок 2.11 – Діаграма послідовності для сценарію «Повернення книги»

На рисунку 2.12 зображена діаграма послідовності для сценарію «Перегляд списку книг які знаходяться в користуванні читача». Опис повідомлень діаграми послідовності занесено до таблиці 2.4.

Таблиця 2.4 – Опис повідомлень діаграми послідовності «Перегляд списку книг які знаходяться в користуванні читача»

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер повідомлення** | **Об'єкт - відправник повідомлення** | **Об'єкт - одержувач повідомлення** | **Назва** |
| 1 | Читач | Інтерфейс програми | Вхід в систему |
| 2 | Інтерфейс програми | База даних | Логін і пароль |
| 3 | База даних | База даних | Перевірка даних |
| 4 | База даних | Інтерфейс програми | Авторизація в програмі |
| 5 | Читач | Інтерфейс програми | Обрати «Мої книги» |
| 6 | Інтерфейс програми | База даних | Запит на вибірку книг читача |
| 7 | База даних | База даних | Пошук збігів |
| 8 | База даних | Інтерфейс програми | Результат запиту |
| 9 | Інтерфейс програми | Читач | Список книг читача |

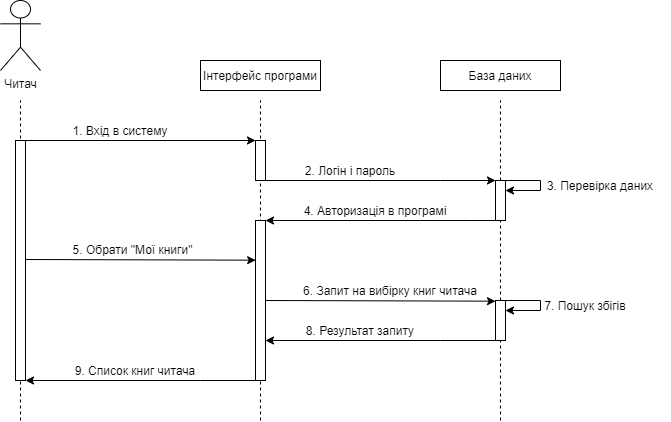


Рисунок 2.12 – Діаграма послідовності для сценарію «Перегляд списку книг які знаходяться в користуванні читача»

* 1. **Проектування бази даних**

Перейдемо до проектування бази даних. Визначимо склад таблиць та зв’язки між ними, для цього скористаємось веб-додатком «dbdesigner.net» (рис. 2.13)

Визначимо основні сутності:

* читачі;
* персонал;
* адміністратори;
* журнал обліку;
* книги;
* сховище.

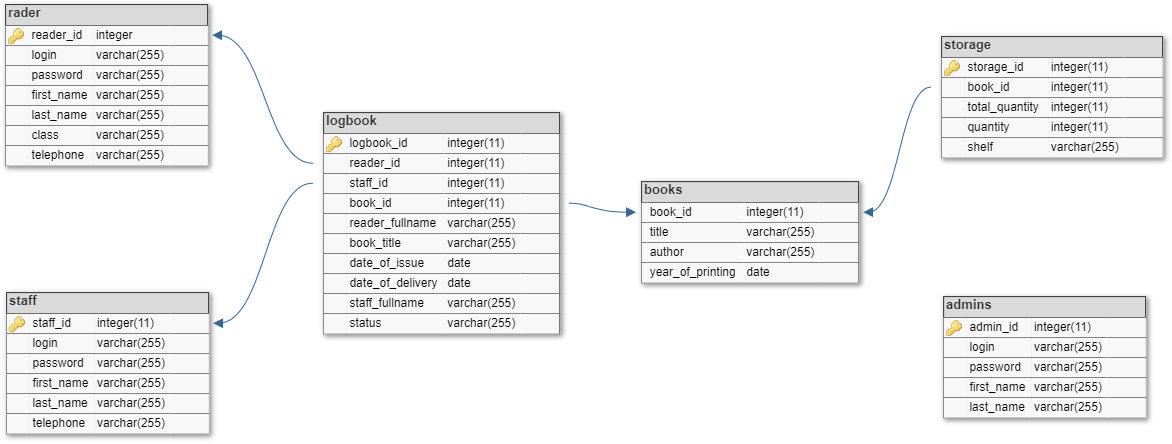


Рисунок 2.13 – Структура БД

# **РОЗДІЛ 3 РОЗРОБКА АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ДІЯЛЬНОСТІ ШКІЛЬНОЇ БІБЛІОТЕКИ**

* 1. **Вибір та обґрунтування технологій для розробки програмного забезпечення**

**3.1.1 Вибір та обґрунтування використання мови C# для розроблення програмного забезпечення.** C# – це одна з багатьох мов програмування яка базується а платформі .NET. Він об’єктно-орієнтований, що дозволяє створювати багаторазові компоненти для найрізноманітніших типів додатків. Багато функцій C# запозичив з інших мов програмування, таких як Delphi, С++ та Java, що багатократно розширило його можливості. Також важливим моментом є те, що C# керована мова, а це означає, що для її виконання потрібна середовище виконання .NET Common Language (CRL). По суті, програму написану на мові C#, виконує CLR він керує пам’яттю, виконує збір сміття та ін. Компілятор C# видає проміжну мову, а не машинний код, а CLR за допомогою функції Just-In-Time компілює його, метод за методом, до скомпільованого машинного коду в пам'яті після чого виконує та керує кодом.

Оскільки C# вимагає CLR, ви повинні встановити його у своїй системі. Усі нові ОС сімейства Windows постачаються з версією CLR. CLR є частиною .NET, тому, якщо ви бачите оновлення для середовища виконання .NET Framework, він містить бібліотеку класів CLR та .NET Framework. Звідси випливає, що якщо ви скопіюєте свою програму C # на іншу машину, то на цій машині також повинен бути встановлений CLR [3].

**3.1.2 Вибір та обґрунтування використання WPF для розроблення програмного інтерфейсу.** WPF – це векторна графічна підсистема, яка використовується для візуалізації інтерфейсу користувача у додатках на базі ОС Windows. Використання векторів дозволяє ефективно масштабувати компоненти інтерфейсу користувача до будь-якого розміру, не спотворюючи та не пікселюючи їх. [4].

Перевагами WPF є:

* можливість відокремлювати інтерфейс від логіки програми;
* зручна прив’язка даних;
* наявність функції віртуалізації інтерфейсу користувача дозволяє обробляти великі набори даних;
* ефективні функції управління та функції повторного використання надають гнучкість при побудові інтерфейсів.

**3.1.3 Вибір та обґрунтування використання MySQL у якості системи керування базами даних.** MySQL – система управління базами даних для реляційних баз даних. Це програмне забезпечення з відкритим кодом, що робить його використання безкоштовним і надає можливість змінити його вихідний код відповідно до потреб. Незважаючи на те, що MySQL є програмним забезпеченням з відкритим кодом, є можливість придбати комерційну ліцензійну версію, щоб отримати послуги преміум-підтримки.

Також одною з переваг є те, що MySQL досить легкий у вивченні порівняно з іншим програмним забезпеченням для управління базами даних, таким як Oracle Database або Microsoft SQL Server.

MySQL може працювати на різних платформах UNIX, Linux, Windows тощо. Ви можете встановити його на сервері або навіть на робочому столі. Крім того, MySQL надійний, масштабований і швидкий [6].

* 1. **Розробка бази даних**

Визначимо основні сутності ІС:

Сутність «Читачі» містить інформацію про читачів та має наступні атрибути:

* reader\_id – ідентифікатор читача, має тип int(11), є первинним ключем, не може бути NULL;
* login – має тип varchar(255), не може бути NULL;
* password – має тип varchar(255), не може бути NULL;
* first\_name – має тип varchar(255), не може бути NULL;
* last\_name – має тип varchar(255), не може бути NULL;
* class – має тип varchar(255);
* telephone – має тип varchar(255), не може бути NULL;

Сутність «Персонал» містить інформацію про бібліотекарів та має наступні атрибути:

* staff\_id – ідентифікатор бібліотекара, має тип int(11), є первинним ключем, не може бути NULL;
* login – має тип varchar(255), не може бути NULL;
* password – має тип varchar(255), не може бути NULL;
* first\_name – має тип varchar(255), не може бути NULL;
* last\_name – має тип varchar(255), не може бути NULL;
* telephone – має тип varchar(255), не може бути NULL;

Сутність «Адміністратори» містить інформацію про бібліотекарів та має наступні атрибути:

* admin\_id – ідентифікатор адміністратора, має тип int(11), є первинним ключем, не може бути NULL;
* login – має тип varchar(255), не може бути NULL;
* password – має тип varchar(255), не може бути NULL;
* first\_name – має тип varchar(255), не може бути NULL;
* last\_name – має тип varchar(255), не може бути NULL;

Сутність «Журнал обліку» містить записи про видачу та повернення книг і має наступні атрибути:

* logbook\_id – ідентифікатор запису, має тип int(11), є первинним ключем, не може бути NULL;
* reader\_id – ідентифікатор читача, має тип int(11), є вторинним ключем, не може бути NULL;
* staff\_id – ідентифікатор бібліотекара, має тип int(11), є вторинним ключем, не може бути NULL;
* book\_id – ідентифікатор книги, має тип int(11), є вторинним ключем, не може бути NULL;
* reader\_full\_­name – має тип varchar(255);
* book\_title – має тип varchar(255), не може бути NULL;
* date\_of\_issue – має тип date, не може бути NULL;
* date\_of\_delivery – має тип date, не може бути NULL;
* staff\_full\_name – має тип varchar(255);
* status – має тип varchar(255), не може бути NULL;

Сутність «Книги» містить інформацію про книги та має наступні атрибути:

* book\_id – ідентифікатор книги, має тип int(11), є первинним ключем, не може бути NULL;
* title – має тип varchar(255), не може бути NULL;
* author – має тип varchar(255), не може бути NULL;
* year\_of\_printing – має тип year, не може бути NULL;

Сутність «Сховище» містить інформацію про місце зберігання так кількість книг і має наступні атрибути:

* book\_id – ідентифікатор книги, має тип int(11), є первинним ключем, не може бути NULL;
* title – має тип varchar(255), не може бути NULL;
* author – має тип varchar(255), не може бути NULL;
* year\_of\_printing – має тип year, не може бути NULL;

Для створення бази даних було використано програму dbForge Studio for MySQL. Після встановлення з’єднання з БД за допомогою дизайнера створено таблиці та встановлені зв’язки між ними. Результат роботи представлено у вигляді автоматично згенерованої діаграми структури БД (рис. 3.1).

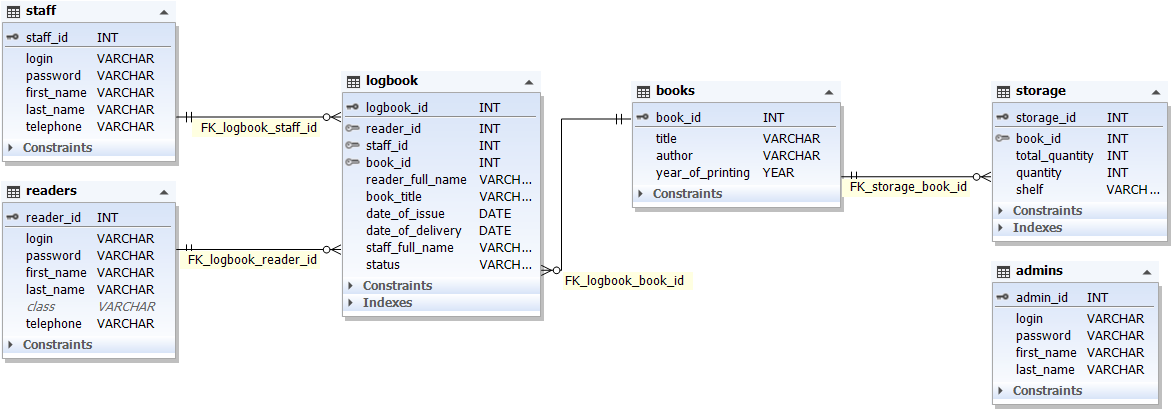


Рисунок 3.1 – Діаграма структури бази даних

Переконавшись, що дії виконано вірно було згенеровано скрипт для створення таблиць, за допомогою якого системний адміністратор буде проводити розгортання на локальному комп’ютері або хостингу для подальшого використання (див. додаток А).

* 1. **Програмна реалізація АБІС**

Розробка виконана згідно представленими раніше діаграмам діяльності та послідовності. Результат розробки розглянемо на прикладі меню бібліотекара. Запустивши програму, користувач відразу побачить вікно авторизації (рис. 3.2).

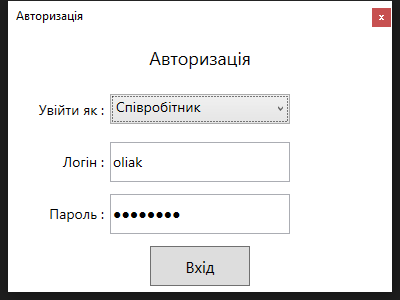


Рисунок 3.2 – Вигляд вікна авторизації

Після вибору ролі та введення логіна, паролю відкриється головне вікно програми відповідно до прав аккаунту (рис. 3.3).

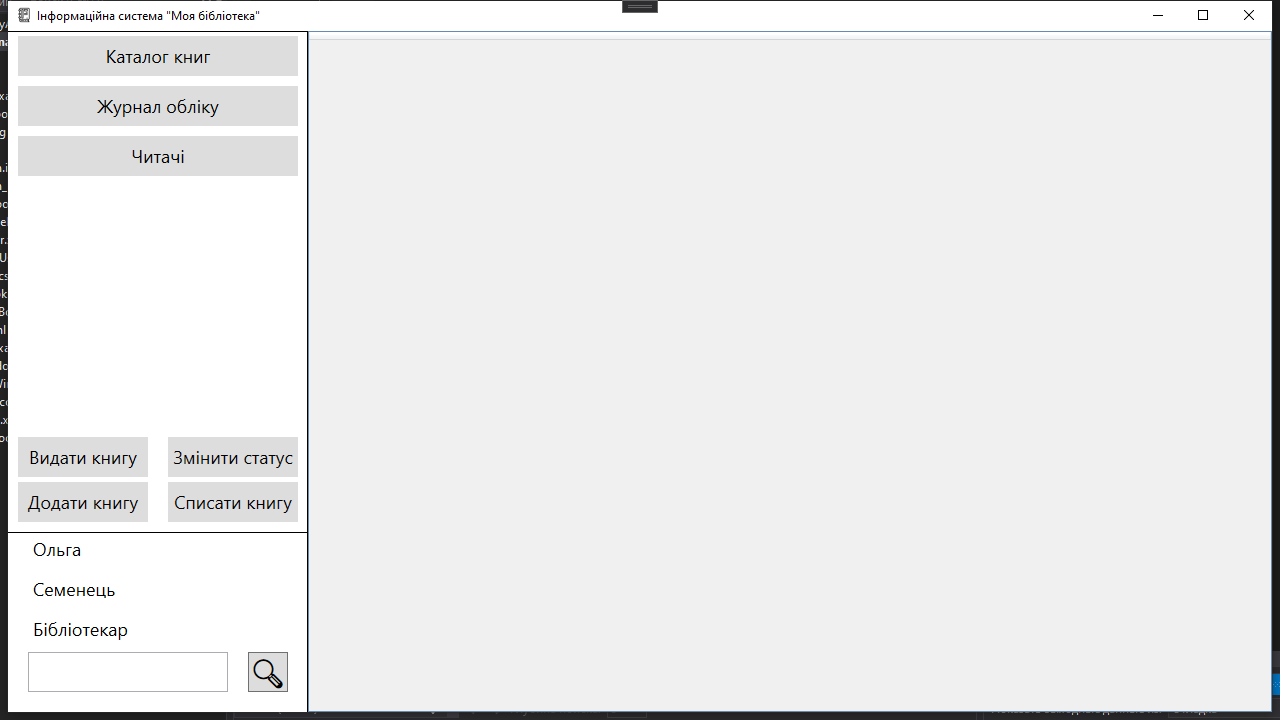


Рисунок 3.2 – Вигляд головного вікна

В верхній частині бокового меню знаходяться елементи керування програмою, а в нижній частині цього меню розміщена інформація про поточного користувача та рядок пошуку. Елементи «Каталог книг», «Журнал обліку», «Читачі» відповідають за відображення відповідної інформації в поточному вікні (рис. 3.3-3.6).

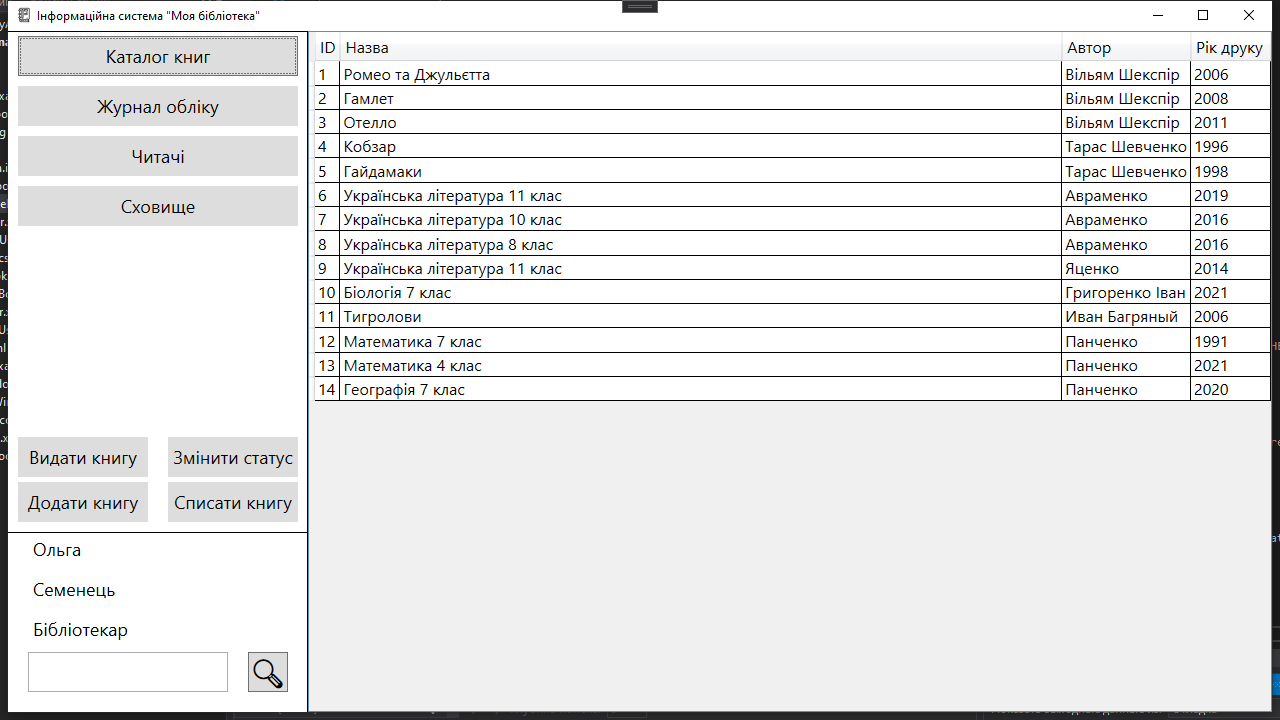


Рисунок 3.3 – Відображення інформації елемента «Каталог книг»

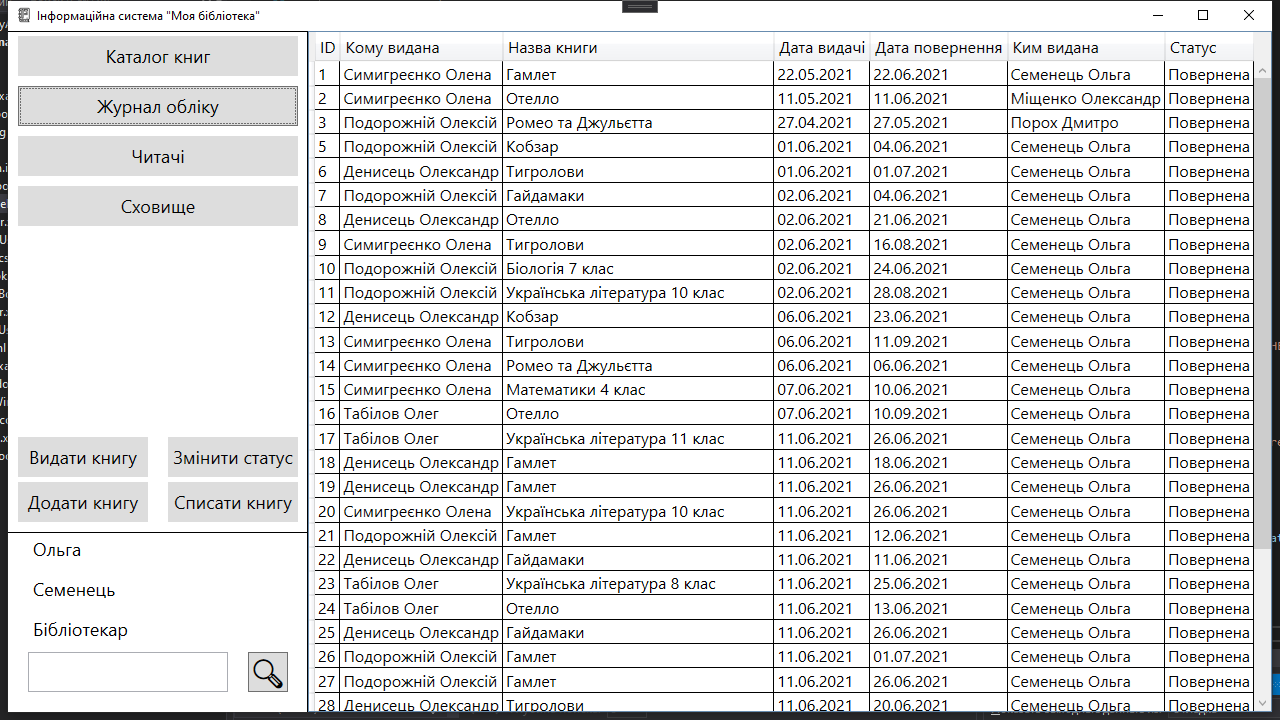


Рисунок 3.4 – Відображення інформації елемента «Журнал обліку»

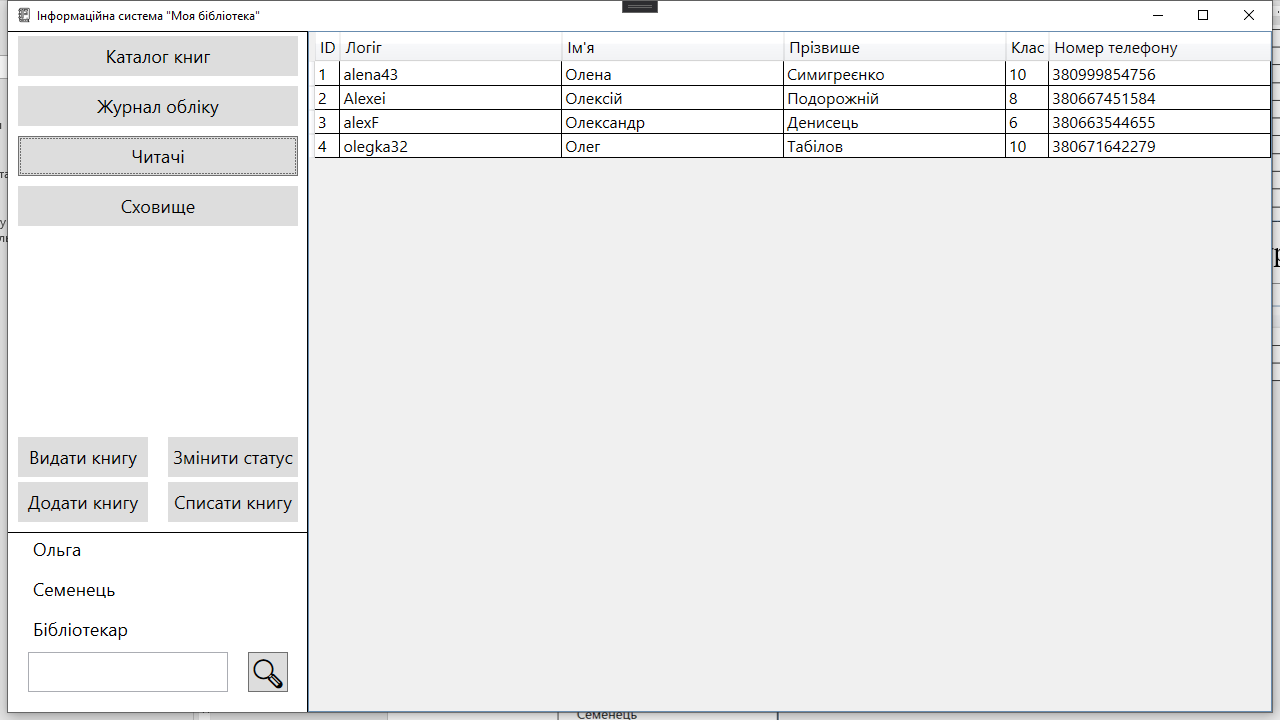


Рисунок 3.5 – Відображення інформації елемента «Читачі»

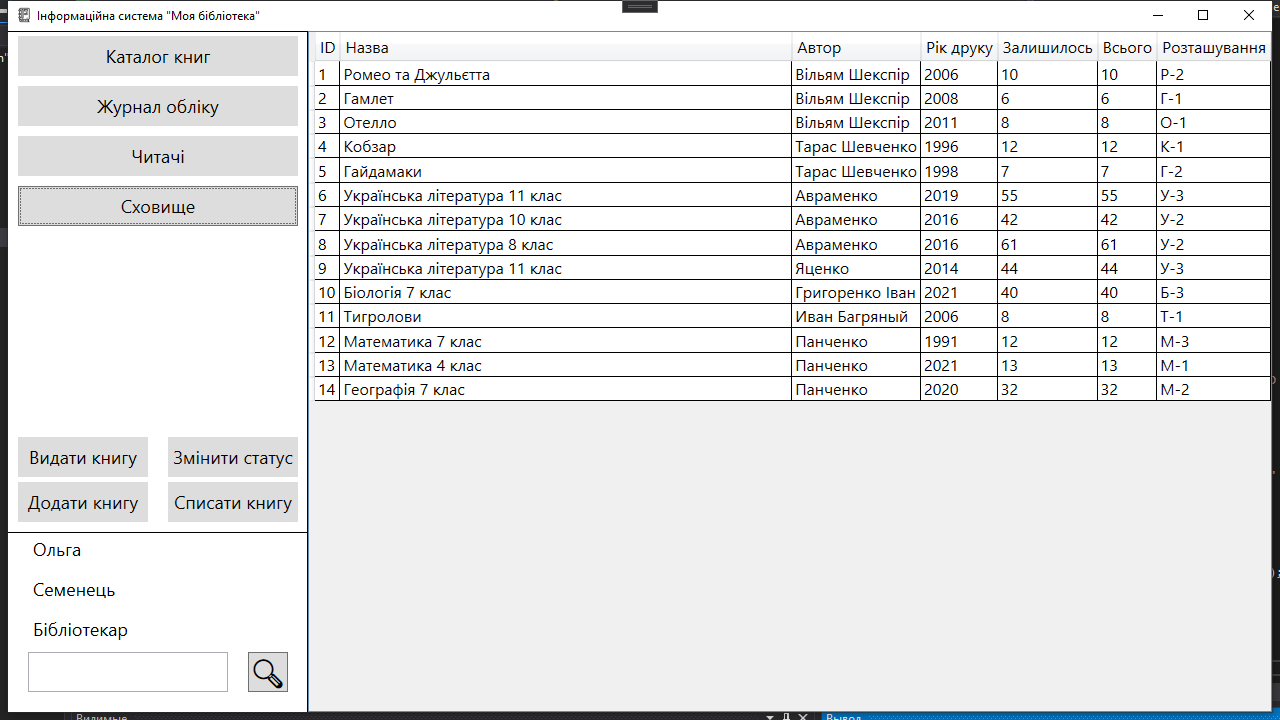


Рисунок 3.6 – Відображення інформації елемента «Сховище»

Елемент меню «Видати книгу» відповідає за створення запису про факт видачі книги читачеві, після натиснення з’являєтеся вікно в якому бібліотекар обирає користувача, назву книги, автора, рік друку та дату до якої потрібно повернути книгу (рис. 3.7).

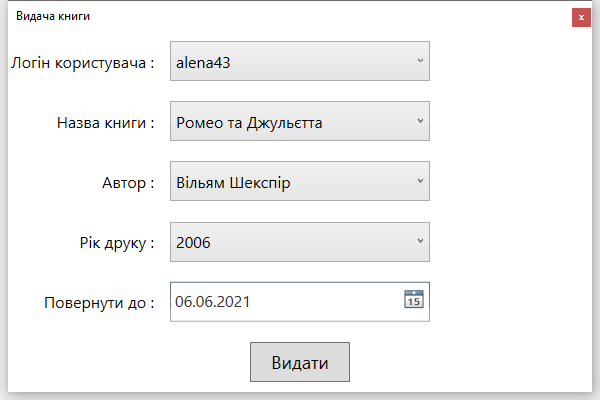


Рисунок 3.7 – Вигляд вікна видачі книги

Елемент меню «Змінити статус» призначений для зміни статусу в журналі обліку. Коли читач повертає книгу статус змінюють на «Повернена», якщо читач не може повернути книгу їй провоюєтеся статус   
«Загублена» (рис. 3.8).

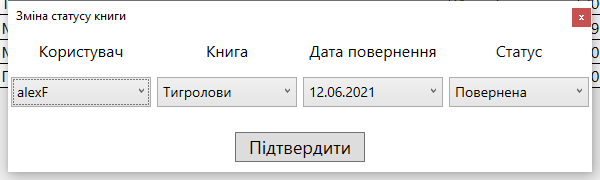


Рисунок 3.8 – Вигляд вікна зміни статусу

Елемент меню «Додати книгу» призначений для внесення книги в базу даних, передбачено два випадка: коли це нова книга і записів в базі даних про неї відсутні, та коли запис про таку книгу вже є. В першому випадку необхідно повністю ввести дані про книгу, а в іншому обрати її із списку та ввести число яке необхідно додати до загальної кількості (рис. 3.9-3.10).

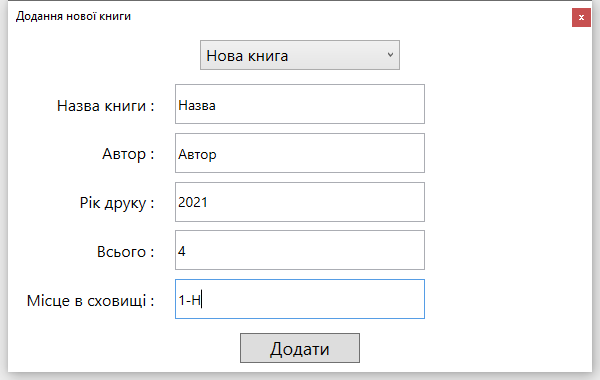


Рисунок 3.9 – Вигляд вікна «Додати книгу» для додання нової книги

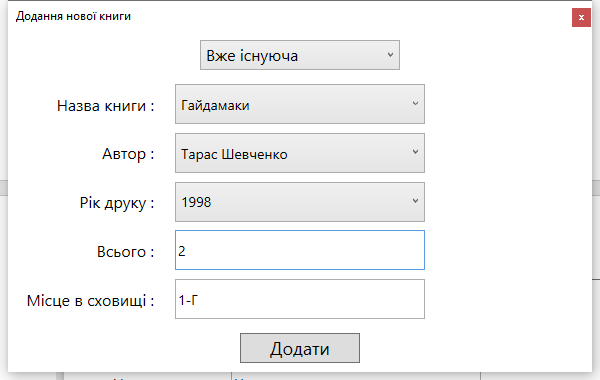


Рисунок 3.10 – Вигляд вікна «Додати книгу» для додання вже існуючої книги

Елемент меню «Списати книгу» призначений для видалення книг з бази даних, щоб провести видалення необхідно обрати дані про книгу та ввести потрібну кількість (рис. 3.11).

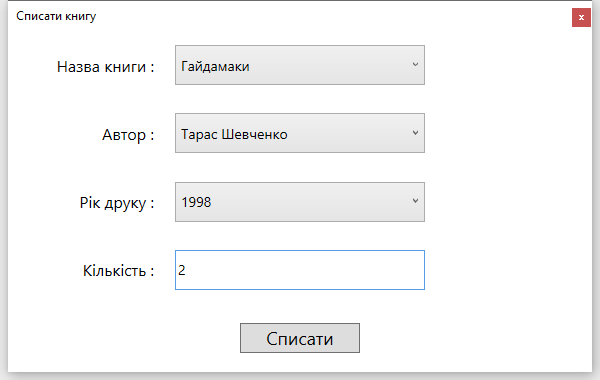


Рисунок 3.11 – Вигляд вікна для видалення книги

Отже, було створено прикладну програму для обліку діяльності шкільної бібліотеки. На цьому етап програмної реалізації завершуєтеся.

# **РОЗДІЛ 4 ТЕСТУВАННЯ**

* 1. **Вибір виду тестування**

Тестування – це процес оцінки системи з метою випробування складових програми та виявлення помилок. Кінцевою метою тестування є надання користувачу якісного продукту, тому цей етап дуже вадливий.

За характером та сферою застосування існують різні типи тестування програмного забезпечення. Це пов’язано з тим, що не всі види тестування підходять для тих чи інших продуктів. Кожен тип має свої плюси і мінуси. Виділяють два основних типи тестування:

* Функціональне – перевіряє функції програмного забезпечення.
* Нефункціональне – перевіряє надійність, зручність використання та продуктивність [8].

Для тестування програми було обрано метод ручного функціонального тестування із застосовуючи негативного та позитивного підходу.

* 1. **План тестування**

Для проведення тестування визначимо функції які доступні всім користувачам, а також функції які доступні в залежності від типу користувача.

Функції, що повинні бути протестовані для всіх користувачів:

* авторизація;
* пошук в поточному розділі.

Функції, що повинні бути протестовані для адміністратора:

* створення нового користувача;
* видалення існуючого користувача.

Функції, що повинні бути протестовані для бібліотекара:

* додання нової книги;
* додання вже існуючої книги;
* видалення списаної книги;
* видача книги;
* повернення книги;
* перегляд записів в журналі обліку.

Функції, що повинні бути протестовані для читача:

* перегляд списку книг, що знаходяться в його користуванні;
* перегляд каталогу книг.

Розробимо тест-кейси, які необхідно перевірити. Спочатку перевіримо загальнодоступні функції.

Таблиця 4.1 – Тест кейси для функцій доступних всім користувачам

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості входу в систему | |
| Передумова | | Програма запущена | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на порожній комбобокс | | З’являється випадаючий список з типами користувачів |
| 2 | Обрати тип користувача | | В комбобоксі відображається обраний тип користувача |
| 3 | Ввести коректні дані | | Введено коректні дані |
| 4 | Натиснути ЛКМ на кнопку «Вхід» | | З’являється головне вікно програми |
| Опис | | Перевірка функції пошуку | |
| Передумова | | Користувач авторизований | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на розділ в якому буде проводитись пошук | | В поточному вікні відобразилась інформація розділу |
| 2 | Натиснути ЛКМ на поле пошуку | | Поле пошуку стало активним |
| 3 | Ввести дані для пошуку | | Введено коректні дані для пошуку |
| 4 | Натиснути ЛКМ на кнопку пошуку | | В поточному вікні відображається результат пошуку |

Розробимо тест-кейси для перевірки функцій доступних тільки адміністратору.

Таблиця 4.2 – Тест кейси для функцій адміністратора

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка функції створення нового облікового запису | |
| Передумова | | Користувач авторизований як адміністратор | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Створити користувача» | | З’являється вікно створення нового облікового запису |
| 2 | В комбобоксі обрати тип користувача | | В комбобоксі відображається обраний тип користувача |
| 3 | Заповнити всі поля | | Всі поля заповнені коректними даними |
| 4 | Натиснути ЛКМ на кнопку «Створити» | | З’являєтеся повідомлення «Обліковий запис створено» та в БД створюється новий запис |
| Опис | | Перевірка функції видалення облікового запису | |
| Передумова | | Користувач авторизований як адміністратор | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Видалити користувача» | | З’являється вікно видалення облікового запису |
| 2 | В комбобоксі обрати тип користувача | | В комбобоксі відображається обраний тип користувача |
| 3 | Ввести логін облікового запису | | Введено логін існуючого облікового запису |
| 4 | Натиснути ЛКМ на кнопку «Перевірити» | | В поточному вікні з’являєтеся інформація про обліковий запис |
| 5 | Натиснути ЛКМ на кнопку «Видалити» | | Обраний обліковий запис видаляєтеся з БД та з’являєтеся повідомлення «Обліковий запис видалено» |
| Опис | | Перевірка введення пробілів при створенні нового облікового запису | |
| Передумова | | Користувач авторизований як адміністратор | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Створити користувача» | | З’являється вікно створення нового облікового запису |
| 2 | В комбобоксі обрати тип користувача | | В комбобоксі відображається обраний тип користувача |
| 3 | Заповнити всі поля пробілами | | Всі поля заповнені пробілами |
| 4 | Натиснути ЛКМ на кнопку «Створити» | | З’являєтеся повідомлення «Заповніть всі поля» |

Продовження таблиці 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка можливості видалення неіснуючого облікового запису | |
| Передумова | | Користувач авторизований як адміністратор | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Видалити користувача» | | З’являється вікно видалення облікового запису |
| 2 | В комбобоксі обрати тип користувача | | В комбобоксі відображається обраний тип користувача |
| 3 | Ввести логін неіснуючого облікового запису | | Введено логін неіснуючого облікового запису |
| 4 | Натиснути ЛКМ на кнопку «Перевірити» | | З’являється повідомлення «Користувач незнайдений» |
| 5 | Натиснути ЛКМ на кнопку «Видалити» | | З’являєтеся повідомлення «Оберіть обліковий запис для видалення» |

Розробимо тест-кейси для перевірки функцій доступних тільки бібліотекару.

Таблиця 4.3 – Тест кейси для функцій бібліотекара

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка функції додання нової книги | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Додати книгу» | | З’являється вікно додання нової книги |
| 2 | В комбобоксі обрати пункт «Нова книга» | | В комбобоксі відображається надпис «Нова книга» |
| 3 | Заповнити всі поля | | Всі поля заповнені коректними даними |
| 4 | Натиснути ЛКМ на кнопку «Додати» | | З’являєтеся повідомлення «Книга успішно додана» та в БД створюється новий запис |
| Опис | | Перевірка функції додання уже існуючої книги | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Додати книгу» | | З’являється вікно додання нової книги |
| 2 | В комбобоксі обрати пункт «Вже існуюча» | | В комбобоксі відображається надпис «Вже існуюча» |
| 3 | Заповнити всі поля | | Всі поля заповнені коректними даними |
| 4 | Натиснути ЛКМ на кнопку «Додати» | | З’являєтеся повідомлення «Книга успішно додана» та в БД створюється новий запис |

Продовження таблиці 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка функції видалення списаної книги | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Списати книгу» | | З’являється вікно видалення книги |
| 2 | В комбобоксі «Назва книги» обрати книгу | | В комбобоксі «Назва книги» відображається назва книги |
| 3 | В комбобоксі «Автор» обрати автора | | В комбобоксі «Автор» відображається автор книги |
| 4 | В комбобоксі «Рік друку» обрати рік друку | | В комбобоксі «Рік друку» відображається рік друку |
| 5 | В текстовому полі вказати кількість книг | | В текстовому полі вказана необхідна кількість книг |
| 6 | Натиснути ЛКМ на кнопку «Списати» | | Вказана кількість книг видалена з БД, з’являєтеся повідомлення про успішне видалення |
| Опис | | Перевірка функції видачі книги | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Видати книгу» | | З’являється вікно видачі книг |
| 2 | В комбобоксі «Логін користувача» обрати логін користувача | | В комбобоксі «Логін користувача» відображається логін обраного користувача |
| 3 | В комбобоксі «Назва книги» обрати книгу | | В комбобоксі «Назва книги» відображається назва книги |
| 4 | В комбобоксі «Автор» обрати автора книги | | В комбобоксі «Автор» відображається автор книги |
| 5 | В комбобоксі «Рік друку» обрати рік друку книги | | В комбобоксі «Рік друку» відображається рік друку книги |
| 6 | Обрати дату в полі  «Повернути до» | | Обрана коректна дата |
| 7 | Натиснути ЛКМ на кнопку «Видати» | | В БД створюється запис про факт видачі книги, з’являється повідомлення «Виконано» |

Продовження таблиці 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка функції повернення книги | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Змінити книгу» | | З’являється вікно повернення книги |
| 2 | В комбобоксі «Користувач» обрати логін користувача | | В комбобоксі «Користувач» відображається логін обраного користувача |
| 3 | В комбобоксі «Книга» обрати книгу | | В комбобоксі «Книга» відображається назва книги |
| 3 | В комбобоксі «Дата повернення» обрати дату повернення книги | | В комбобоксі «Дата повернення» відображається дата повернення книги |
| 4 | В комбобоксі «Статус» обрати книгу | | В комбобоксі «Статус» відображається назва книги |
| 5 | Натиснути ЛКМ на кнопку «Видати» | | В БД створюється запис про факт повернення книги, з’являється повідомлення «Статус запису успішно змінений» |
| Опис | | Перевірка функції перегляду записів в журналі обліку | |
| Передумова | | Користувач авторизований як бібліотекар | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Журнал обліку» | | В поточному вікні відображається інформація про видані книги |

Розробимо тест-кейси для перевірки функцій які доступних читачу.

Таблиця 4.3 – Тест кейси для функцій доступних читачу

|  |  |  |  |
| --- | --- | --- | --- |
| Опис | | Перевірка функції перегляду списку книг, що знаходяться в його користуванні | |
| Передумова | | Користувач авторизований як читач | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Мої книги» | | В поточному вікні відображається інформація про книги, що знаходяться в користуванні поточного користувача |
| Опис | | Перевірка функції перегляду каталогу книг | |
| Передумова | | Користувач авторизований як читач | |
| № | Дія | | Очікуваний результат |
| 1 | Натиснути ЛКМ на кнопку «Каталог» | | В поточному вікні відображається інформація про книги всі книги які занесені до БД |

# **ВИСНОВКИ**

Мета кваліфікаційної роботи бакалавра полягала в розробці програми для автоматизації обліку діяльності шкільної бібліотеки. Для цього був проведений аналіз існуючих рішень та поставлена задача реалізувати головні функції, а саме:

1. Реєстрація та видалення облікових записів.
2. Розмежування прав доступу.
3. Перегляд фонду бібліотеки.
4. Облік видачі та повернення літератури читачем.
5. Внесення та видалення книг з БД.
6. Пошук в поточному розділі.

Для втілення функціоналу, представленого вище, в другому розділі розроблено UML-діаграми та спроектовано структуру бази даних, після чого в третьому розділі проведено вибір та обґрунтування мови програмування C# і графічної підсистеми WPF в якості інструментів для розробки програми.   
В якості систему управління базами даних обрано MySQL, використання якої надає можливість розміщення БД як в локальній мережі навчального закладу так і на віддаленому сервері в глобальній мережі. Підчас розробки бази даних було визначено сутності, створено таблиці та за допомогою зовнішніх ключів встановлено зв’язки між ними. В результаті програмної реалізації створено прикладну програму яка забезпечить автоматизацію обліку діяльності шкільної бібліотеки. Щоб перевірити працездатність програми в четвертому розділі проведено вибір виду тестування, розроблено тест-кейси та проведене тестування в результаті якого недоліків не було виявлено.

Отже розроблена програма відповідає всім поставленим умовам.   
В якості подальшого вдосконалення можлива розробка розширених функцій, таких як: імпорт та експорт даних з таблиць MS Excel, фільтрація інформації в колонках таблиці.

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Огляд автоматизованих бібліотечних систем [Електронний ресурс]: – Режим доступу: https://lala.lanbook.com/obzor-avtomatizirovannyh-bibliotechnyh-sistem-4-ochen-raznyh-produkta
2. Загальна характеристика мови UML [Електронний ресурс]: – Режим доступу: http://www.informicus.ru/default.aspx?SECTION=6&id=73& subdivisionid=2
3. Tutorial, links and Articles About C# [Електронний ресурс]: – Режим доступу: https://csharp-station.com
4. Эндрю Троелсен, Филипп Джепикс. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е издание, 2018 – 941с.
5. Бэрон Шварц, Петр Зайцев, Вадим Ткаченко. MySQL по максимуму, 2018 – 150с.
6. What is MySQL? & Why it is the world’s most popular open source database [Електронний ресурс]: – Режим доступу: https://www.mysqltutorial.org/what-is-mysql/
7. Роман Савин. Тестирование dot com, 2007 – 37с.
8. What Is Software Testing? All the Basics You Need to Know [Електронний ресурс]: – Режим доступу: https://www.testim.io/blog/software-testing-basics/
9. Алан Бьюли. Изучаем SQL, 2016 – 91с.
10. Дейв Томас, Энди Хант. Программист-прагматик, 2016 – 216с.
11. Кори Сандлер, Гленфорд Майерс, Том Баджетт. Искуство тестирование программ, 2012 – 188с.
12. Вон Вернон. Реализация методов предметно-ориентированного проектирования, 2016 – 158с.

# **ДОДАТОК А СКРИПТ ГЕНЕРАЦІЇ ТАБЛИЦЬ БАЗИ ДАНИХ**

--

-- Set character set the client will use to send SQL statements to the server

--

SET NAMES 'utf8';

--

-- Set default database

--

USE library;

--

-- Create table `staff`

--

CREATE TABLE staff (

staff\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

login varchar(255) NOT NULL,

password varchar(255) NOT NULL,

first\_name varchar(255) NOT NULL,

last\_name varchar(255) NOT NULL,

telephone varchar(255) NOT NULL,

PRIMARY KEY (staff\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 6,

AVG\_ROW\_LENGTH = 4096,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

--

-- Create table `readers`

--

CREATE TABLE readers (

reader\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

login varchar(255) NOT NULL,

password varchar(255) NOT NULL,

first\_name varchar(255) NOT NULL,

last\_name varchar(255) NOT NULL,

class varchar(255) DEFAULT NULL,

telephone varchar(255) NOT NULL,

PRIMARY KEY (reader\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 5,

AVG\_ROW\_LENGTH = 5461,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

--

-- Create table `books`

--

CREATE TABLE books (

book\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

title varchar(255) NOT NULL,

author varchar(255) NOT NULL,

year\_of\_printing year NOT NULL,

PRIMARY KEY (book\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 15,

AVG\_ROW\_LENGTH = 1489,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

--

-- Create table `storage`

--

CREATE TABLE storage (

storage\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

book\_id int UNSIGNED NOT NULL,

total\_quantity int NOT NULL,

quantity int NOT NULL,

shelf varchar(255) NOT NULL,

PRIMARY KEY (storage\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 15,

AVG\_ROW\_LENGTH = 1489,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

--

-- Create foreign key

--

ALTER TABLE storage

ADD CONSTRAINT FK\_storage\_book\_id FOREIGN KEY (book\_id)

REFERENCES books (book\_id);

--

-- Create table `logbook`

--

CREATE TABLE logbook (

logbook\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

reader\_id int UNSIGNED NOT NULL,

staff\_id int UNSIGNED NOT NULL,

book\_id int UNSIGNED NOT NULL,

reader\_full\_name varchar(255) DEFAULT NULL,

book\_title varchar(255) NOT NULL,

date\_of\_issue date NOT NULL,

date\_of\_delivery date NOT NULL,

staff\_full\_name varchar(255) DEFAULT NULL,

status varchar(255) NOT NULL,

PRIMARY KEY (logbook\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 36,

AVG\_ROW\_LENGTH = 1260,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

--

-- Create foreign key

--

ALTER TABLE logbook

ADD CONSTRAINT FK\_logbook\_book\_id FOREIGN KEY (book\_id)

REFERENCES books (book\_id);

--

-- Create foreign key

--

ALTER TABLE logbook

ADD CONSTRAINT FK\_logbook\_reader\_id FOREIGN KEY (reader\_id)

REFERENCES readers (reader\_id);

--

-- Create foreign key

--

ALTER TABLE logbook

ADD CONSTRAINT FK\_logbook\_staff\_id FOREIGN KEY (staff\_id)

REFERENCES staff (staff\_id);

--

-- Create table `admins`

--

CREATE TABLE admins (

admin\_id int UNSIGNED NOT NULL AUTO\_INCREMENT,

login varchar(255) NOT NULL,

password varchar(255) NOT NULL,

first\_name varchar(255) NOT NULL,

last\_name varchar(255) NOT NULL,

PRIMARY KEY (admin\_id)

)

ENGINE = INNODB,

AUTO\_INCREMENT = 9,

AVG\_ROW\_LENGTH = 5461,

CHARACTER SET utf8mb4,

COLLATE utf8mb4\_0900\_ai\_ci;

# **ДОДАТОК Б КОД ПРОГРАМИ**

* + - 1. Клас DataBase.

class DataBase

{

MySqlConnection connection = new MySqlConnection(setConnectionString());

public void openConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

connection.Open();

}

public void closeConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

connection.Close();

}

public MySqlConnection getConnection()

{

return connection;

}

private static string setConnectionString()

{

if (File.Exists("connection.conf"))

{

StreamReader stream = new StreamReader("connection.conf");

string connection\_string = stream.ReadToEnd().Trim();

stream.Close();

if (connection\_string == "")

MessageBox.Show("Сконфігуруйте підключення до бази даних в файлі 'connection.conf'");

else

return connection\_string;

}

else

File.Create("connection.conf");

return null;

}

}

1. Клас MainWindow.

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

int user\_id;

public void setUserId(string id)

{

user\_id = Convert.ToInt32(id);

}

public void addLibrarianButtons()

{

addButton("catalog", "Каталог книг");

addButton("logbook", "Журнал обліку");

addButton("readers", "Читачі");

addButton("storage", "Сховище");

addBookButton.Visibility = Visibility.Visible;

removeBookButton.Visibility = Visibility.Visible;

sendBookButton.Visibility = Visibility.Visible;

changeBookStatusButton.Visibility = Visibility.Visible;

DataBase dataBase = new DataBase();

dataBase.openConnection();

MySqlCommand select\_first\_name = new MySqlCommand("SELECT first\_name FROM staff WHERE staff\_id = " + user\_id, dataBase.getConnection());

first\_name.Text = select\_first\_name.ExecuteScalar().ToString();

MySqlCommand select\_last\_name = new MySqlCommand("SELECT last\_name FROM staff WHERE staff\_id = " + user\_id, dataBase.getConnection());

last\_name.Text = select\_last\_name.ExecuteScalar().ToString();

position.Text = "Бібліотекар";

dataBase.closeConnection();

}

public void addReaderButtons()

{

addButton("myBook", "Мої книги");

addButton("catalog", "Каталог книг");

DataBase dataBase = new DataBase();

dataBase.openConnection();

MySqlCommand select\_first\_name = new MySqlCommand("SELECT first\_name FROM readers WHERE reader\_id = " + user\_id, dataBase.getConnection());

first\_name.Text = select\_first\_name.ExecuteScalar().ToString();

MySqlCommand select\_last\_name = new MySqlCommand("SELECT last\_name FROM readers WHERE reader\_id = " + user\_id, dataBase.getConnection());

last\_name.Text = select\_last\_name.ExecuteScalar().ToString();

MySqlCommand select\_class = new MySqlCommand("SELECT class FROM readers WHERE reader\_id = " + user\_id, dataBase.getConnection());

position.Text = select\_class.ExecuteScalar().ToString() + " клас";

dataBase.closeConnection();

}

public void addAdminButtons()

{

addButton("createUser", "Створити користувача");

addButton("deleteUser", "Видалити користувача");

addButton("staff", "Персонал");

addButton("readers", "Читачі");

DataBase dataBase = new DataBase();

dataBase.openConnection();

MySqlCommand select\_first\_name = new MySqlCommand("SELECT first\_name FROM admins WHERE admin\_id = " + user\_id, dataBase.getConnection());

first\_name.Text = select\_first\_name.ExecuteScalar().ToString();

MySqlCommand select\_last\_name = new MySqlCommand("SELECT last\_name FROM admins WHERE admin\_id = " + user\_id, dataBase.getConnection());

last\_name.Text = select\_last\_name.ExecuteScalar().ToString();

position.Text = "Адміністратор";

dataBase.closeConnection();

}

private void addButton(string name, string content)

{

var button = new Button

{

Name = name,

Content = content,

Width = 280,

Height = 40,

Margin = new Thickness(5),

BorderThickness = new Thickness(0),

FontSize = 18

};

button.Click += someButtonClick;

SideMenu.Children.Add(button);

}

private void someButtonClick(object sender, RoutedEventArgs e)

{

var button = (Button)sender;

resetSelection(button);

switch (button.Name)

{

case "catalog":

removeColums();

catalogColums();

break;

case "logbook":

removeColums();

logbookColums();

break;

case "readers":

removeColums();

readerColums();

break;

case "staff":

removeColums();

staffColums();

break;

case "createUser":

CreateUser createUser = new CreateUser();

createUser.Show();

break;

case "deleteUser":

DeleteUser deleteUser = new DeleteUser();

deleteUser.Show();

break;

case "myBook":

removeColums();

myBooksColums();

break;

case "storage":

removeColums();

storageColums();

break;

}

}

private void resetSelection(Button button)

{

foreach (UIElement el in SideMenu.Children)

if (el is Button)

((Button)el).BorderThickness = new Thickness(0);

button.BorderThickness = new Thickness(1);

}

private void catalogColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("book\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("title"), Header = "Назва", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("author"), Header = "Автор" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("year\_of\_printing"), Header = "Рік друку", Width = 80 });

catalogDataLoad();

}

private void logbookColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("logbook\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("reader\_full\_name"), Header = "Кому видана" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("book\_title"), Header = "Назва книги", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("date\_of\_issue"), Header = "Дата видачі", });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("date\_of\_delivery"), Header = "Дата повернення" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("staff\_full\_name"), Header = "Ким видана" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("status"), Header = "Статус" });

(content.Columns[3] as DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy";

(content.Columns[4] as DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy";

logbookDataLoad();

}

private void readerColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("reader\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("login"), Header = "Логіг", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("first\_name"), Header = "Ім'я", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("last\_name"), Header = "Прізвише", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("class"), Header = "Клас" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("telephone"), Header = "Номер телефону", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

readersDataLoad();

}

private void staffColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("staff\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("login"), Header = "Логін", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("first\_name"), Header = "Ім'я", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("last\_name"), Header = "Прізвище", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("telephone"), Header = "Номер телефону", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

staffDataLoad();

}

private void myBooksColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("logbook\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("book\_title"), Header = "Назва книги", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("date\_of\_issue"), Header = "Дата видачі" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("date\_of\_delivery"), Header = "Дата повернення" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("staff\_full\_name"), Header = "Ким видана" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("status"), Header = "Статус" });

(content.Columns[2] as DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy";

(content.Columns[3] as DataGridTextColumn).Binding.StringFormat = "dd.MM.yyyy";

myBooksDataLoad();

}

private void storageColums()

{

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("book\_id"), Header = "ID" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("title"), Header = "Назва", Width = new DataGridLength(1, DataGridLengthUnitType.Star) });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("author"), Header = "Автор" });

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("year\_of\_printing"), Header = "Рік друку"});

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("quantity"), Header = "Залишилось"});

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("total\_quantity"), Header = "Всього"});

content.Columns.Add(new DataGridTextColumn { Binding = new Binding("shelf"), Header = "Розташування"});

storageDataLoad();

}

private void removeColums()

{

while (content.Columns.Count != 0)

content.Columns.RemoveAt(0);

}

private void staffDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT staff\_id, login, first\_name, last\_name, telephone FROM staff";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void readersDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT reader\_id, login, first\_name, last\_name, class, telephone FROM readers";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void catalogDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT book\_id, title, author, year\_of\_printing FROM books";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void storageDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT b.book\_id, b.title, b.author, b.year\_of\_printing, s.quantity, s.total\_quantity, s.shelf FROM books b RIGHT OUTER JOIN storage s ON b.book\_id = s.book\_id";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void logbookDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT \* FROM logbook";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void myBooksDataLoad()

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT \* FROM logbook WHERE reader\_id =" + user\_id;

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

content.ItemsSource = dataTable.DefaultView;

dataBase.closeConnection();

}

private void addBookButton\_Click(object sender, RoutedEventArgs e)

{

AddBook addBook = new AddBook();

addBook.Show();

}

private void removeBookButton\_Click(object sender, RoutedEventArgs e)

{

DeleteBook deleteBook = new DeleteBook();

deleteBook.Show();

}

private void sendBookButton\_Click(object sender, RoutedEventArgs e)

{

SendBook sendBook = new SendBook();

sendBook.setStaffId(user\_id);

sendBook.Show();

}

private void changeBookStatusButton\_Click(object sender, RoutedEventArgs e)

{

ChangeBookStatus changeBookStatus = new ChangeBookStatus();

changeBookStatus.Show();

}

private void searchButton\_Click(object sender, RoutedEventArgs e)

{

string selectedButtonName = "";

foreach (UIElement el in SideMenu.Children)

if (el is Button)

if (((Button)el).BorderThickness.ToString() != "0,0,0,0")

selectedButtonName = ((Button)el).Name.ToString();

if (searchTextBox.Text.Trim() != "")

if (selectedButtonName == "catalog")

{

makeSearchRequest("SELECT \* FROM books WHERE book\_id = @searchValue OR title = @searchValue OR author = @searchValue OR year\_of\_printing = @searchValue");

}

else if (selectedButtonName == "readers")

{

makeSearchRequest("SELECT \* FROM readers WHERE reader\_id = @searchValue OR login = @searchValue OR first\_name = @searchValue OR last\_name = @searchValue OR class = @searchValue OR telephone = @searchValue");

}

else if (selectedButtonName == "logbook")

{

makeSearchRequest("SELECT \* FROM logbook WHERE logbook\_id = @searchValue OR reader\_full\_name = @searchValue OR book\_title = @searchValue OR staff\_full\_name = @searchValue OR status = @searchValue");

}

else if (selectedButtonName == "staff")

{

makeSearchRequest("SELECT \* FROM staff WHERE staff\_id = @searchValue OR login = @searchValue OR first\_name = @searchValue OR last\_name = @searchValue OR telephone = @searchValue");

}

else if (selectedButtonName == "myBook")

{

makeSearchRequest("SELECT \* FROM logbook WHERE (logbook\_id = @searchValue OR reader\_full\_name = @searchValue OR book\_title = @searchValue OR staff\_full\_name = @searchValue OR status = @searchValue) AND reader\_id =" + user\_id);

}

else

{

MessageBox.Show("Оберіть каталог для пошуку");

}

else

{

MessageBox.Show("Строка пошуку пуста");

}

}

private void makeSearchRequest(string request)

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand(request, dataBase.getConnection());

cmd.Parameters.Add("@searchValue", MySqlDbType.VarChar).Value = (string)searchTextBox.Text.Trim();

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

content.ItemsSource = dataTable.DefaultView;

}

else

{

MessageBox.Show("Запит не дав результатів");

}

dataBase.closeConnection();

}

}

1. Клас Login.

public partial class Login : Window

{

public Login()

{

InitializeComponent();

}

private void loginButton\_Click(object sender, RoutedEventArgs e)

{

try

{

string login = loginBox.Text.Trim();

string password = passwordBox.Password.Trim();

DataBase dataBase = new DataBase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

if (loginUs.Text == "")

{

MessageBox.Show("Оберіть тип користувача");

}

else if (loginUs.Text == "Адміністратор")

{

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM admins WHERE login = @userLogin AND password = @userPassword", dataBase.getConnection());

cmd.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = login;

cmd.Parameters.Add("@userPassword", MySqlDbType.VarChar).Value = password;

adapter.SelectCommand = cmd;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MainWindow mainWindow = new MainWindow();

MySqlCommand set\_user\_id = new MySqlCommand("SELECT admin\_id FROM admins WHERE login = @login AND password = @password", dataBase.getConnection());

set\_user\_id.Parameters.Add("@login", MySqlDbType.VarChar).Value = login;

set\_user\_id.Parameters.Add("@password", MySqlDbType.VarChar).Value = password;

dataBase.openConnection();

string result = set\_user\_id.ExecuteScalar().ToString();

if (result != null)

{

mainWindow.setUserId((string)result);

}

mainWindow.addAdminButtons();

mainWindow.Show();

dataBase.closeConnection();

this.Close();

}

else

{

MessageBox.Show("Невірний логін або пароль");

}

}

else if (loginUs.Text == "Співробітник")

{

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM staff WHERE login = @userLogin AND `password` = @userPassword", dataBase.getConnection());

cmd.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = login;

cmd.Parameters.Add("@userPassword", MySqlDbType.VarChar).Value = password;

adapter.SelectCommand = cmd;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MainWindow mainWindow = new MainWindow();

MySqlCommand set\_user\_id = new MySqlCommand("SELECT staff\_id FROM staff WHERE login = @login AND password = @password", dataBase.getConnection());

set\_user\_id.Parameters.Add("@login", MySqlDbType.VarChar).Value = login;

set\_user\_id.Parameters.Add("@password", MySqlDbType.VarChar).Value = password;

dataBase.openConnection();

string result = set\_user\_id.ExecuteScalar().ToString();

if (result != null)

{

mainWindow.setUserId((string)result);

}

mainWindow.addLibrarianButtons();

mainWindow.Show();

dataBase.closeConnection();

this.Close();

}

else

{

MessageBox.Show("Невірний логін або пароль");

}

}

else if (loginUs.Text == "Читач")

{

MySqlCommand cmd = new MySqlCommand("SELECT \* FROM readers WHERE login = @userLogin AND password = @userPassword", dataBase.getConnection());

cmd.Parameters.Add("@userLogin", MySqlDbType.VarChar).Value = login;

cmd.Parameters.Add("@userPassword", MySqlDbType.VarChar).Value = password;

adapter.SelectCommand = cmd;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MainWindow mainWindow = new MainWindow();

MySqlCommand set\_user\_id = new MySqlCommand("SELECT reader\_id FROM readers WHERE login = @login AND password = @password", dataBase.getConnection());

set\_user\_id.Parameters.Add("@login", MySqlDbType.VarChar).Value = login;

set\_user\_id.Parameters.Add("@password", MySqlDbType.VarChar).Value = password;

dataBase.openConnection();

string result = set\_user\_id.ExecuteScalar().ToString();

if (result != null)

{

mainWindow.setUserId((string)result);

}

mainWindow.Show();

mainWindow.addReaderButtons();

dataBase.closeConnection();

this.Close();

}

else

{

MessageBox.Show("Невірний логін або пароль");

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

1. Клас CreateUser.

public partial class CreateUser : Window

{

public CreateUser()

{

InitializeComponent();

}

private void userType\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedItem = (ComboBoxItem)userType.SelectedValue;

var selectedRole = (string)selectedItem.Content;

cleatAllFiled();

if (selectedRole == "Адміністратор")

{

classBox.IsEnabled = false;

telephoneBox.IsEnabled = false;

}

else if (selectedRole == "Співробітник")

{

allowAllFileds();

classBox.IsEnabled = false;

}

else if (selectedRole == "Читач")

{

allowAllFileds();

}

}

private void createNewUserClick(object sender, RoutedEventArgs e)

{

string login = loginBox.Text.Trim();

string password = passwordBox.Password.Trim();

string first\_name = firstNameBox.Text.Trim();

string last\_name = lastNameBox.Text.Trim();

string clas = classBox.Text.Trim();

string telephone = telephoneBox.Text.Trim();

if (userType.Text == "")

{

MessageBox.Show("Оберіть тип користувача");

}

else if (checkLogin())

{

if (userType.Text == "Адміністратор")

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string sql = "INSERT INTO admins (login, password, first\_name, last\_name)" + "VALUES (@login, @password, @first\_name, @last\_name);";

MySqlCommand cmd = new MySqlCommand(sql, dataBase.getConnection());

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

cmd.Parameters.AddWithValue("@first\_name", first\_name);

cmd.Parameters.AddWithValue("@last\_name", last\_name);

cmd.ExecuteNonQuery();

dataBase.closeConnection();

this.Close();

}

else if (userType.Text == "Співробітник")

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string sql = "INSERT INTO staff (login, password, first\_name, last\_name, telephone)" + "VALUES (@login, @password, @first\_name, @last\_name, @telephone);";

MySqlCommand cmd = new MySqlCommand(sql, dataBase.getConnection());

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

cmd.Parameters.AddWithValue("@first\_name", first\_name);

cmd.Parameters.AddWithValue("@last\_name", last\_name);

cmd.Parameters.AddWithValue("@telephone", telephone);

cmd.ExecuteNonQuery();

dataBase.closeConnection();

this.Close();

}

else if (userType.Text == "Читач")

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

string sql = "INSERT INTO readers (login, password, first\_name, last\_name, class, telephone)" + "VALUES (@login, @password, @first\_name, @last\_name, @class, @telephone);";

MySqlCommand cmd = new MySqlCommand(sql, dataBase.getConnection());

cmd.Parameters.AddWithValue("@login", login);

cmd.Parameters.AddWithValue("@password", password);

cmd.Parameters.AddWithValue("@first\_name", first\_name);

cmd.Parameters.AddWithValue("@last\_name", last\_name);

cmd.Parameters.AddWithValue("@class", clas);

cmd.Parameters.AddWithValue("@telephone", telephone);

cmd.ExecuteNonQuery();

dataBase.closeConnection();

this.Close();

}

}

else

{

MessageBox.Show("Користувач з таким логіном вже існує");

}

}

// return false if login exist

private bool checkLogin()

{

string cmd\_string = "";

DataBase dataBase = new DataBase();

dataBase.openConnection();

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter();

if (userType.Text == "Читач")

{

cmd\_string = "SELECT login FROM readers";

}

else if (userType.Text == "Співробітник")

{

cmd\_string = "SELECT login FROM staff";

}

else if (userType.Text == "Адміністратор")

{

cmd\_string = "SELECT login FROM admins";

}

MySqlCommand cmd = new MySqlCommand(cmd\_string, dataBase.getConnection());

dataAdapter.SelectCommand = cmd;

dataAdapter.Fill(dataTable);

dataBase.closeConnection();

if (dataTable.Rows.Count != 0)

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (dataTable.Rows[i][0].ToString().ToLower() == loginBox.Text.Trim().ToLower())

{

return false;

}

}

return true;

}

private void cleatAllFiled()

{

foreach (UIElement el in content.Children)

{

if (el is TextBox)

((TextBox)el).Clear();

if (el is PasswordBox)

((PasswordBox)el).Clear();

}

}

private void allowAllFileds()

{

foreach (UIElement el in content.Children)

{

if (el is TextBox)

((TextBox)el).IsEnabled = true;

}

}

}

1. Клас ChangeBookStatus.

public partial class ChangeBookStatus : Window

{

public ChangeBookStatus()

{

InitializeComponent();

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT login FROM readers", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (readerLoginComboBox.Items.Count != 0)

readerLoginComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (readerLoginComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

readerLoginComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void readerLoginComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT book\_title FROM logbook WHERE reader\_id = (SELECT reader\_id FROM readers WHERE login = '" + (string)readerLoginComboBox.SelectedItem + "') AND status != 'Повернена'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (bookTitleComboBox.Items.Count != 0)

bookTitleComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (bookTitleComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

bookTitleComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

else

{

bookTitleComboBox.Items.Clear();

}

dataBase.closeConnection();

}

private void bookTitleComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT date\_of\_delivery FROM logbook WHERE reader\_id = (SELECT reader\_id FROM readers WHERE login = '" + (string)readerLoginComboBox.SelectedItem + "') AND status != 'Повернена'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (dateOfDelivery.Items.Count != 0)

dateOfDelivery.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (dateOfDelivery.Items.Contains(((DateTime)dataTable.Rows[i][0]).Date.ToString(@"dd.MM.yyyy")) != true)

{

dateOfDelivery.Items.Add(((DateTime)dataTable.Rows[i][0]).Date.ToString(@"dd.MM.yyyy"));

}

}

}

else

{

bookTitleComboBox.Items.Clear();

}

dataBase.closeConnection();

}

private void changeBookStatusButton\_Click(object sender, RoutedEventArgs e)

{

var selectedItem = (ComboBoxItem)statusComboBox.SelectedValue;

var status = (string)selectedItem.Content;

DataBase dataBase = new DataBase();

dataBase.openConnection();

// get book\_id

MySqlCommand get\_book\_id = new MySqlCommand("SELECT book\_id FROM logbook WHERE reader\_id = (SELECT reader\_id FROM readers WHERE login = @login) AND book\_title = @book\_title AND date\_of\_delivery = @date\_of\_delivery AND status != 'Повернена'", dataBase.getConnection());

get\_book\_id.Parameters.Add("@login", MySqlDbType.VarChar).Value = (string)readerLoginComboBox.SelectedItem;

get\_book\_id.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = (string)bookTitleComboBox.SelectedItem;

get\_book\_id.Parameters.Add("@date\_of\_delivery", MySqlDbType.Date).Value = Convert.ToDateTime(dateOfDelivery.Text);

string book\_id = get\_book\_id.ExecuteScalar().ToString();

// logbook status change

MySqlCommand cmd = new MySqlCommand("UPDATE logbook SET status = '" + status + "' WHERE reader\_id = (SELECT reader\_id FROM readers WHERE login = '" + (string)readerLoginComboBox.SelectedItem + "') AND book\_title = '" + (string)bookTitleComboBox.SelectedItem + "' AND date\_of\_delivery = @date\_of\_delivery AND status != 'Повернена'", dataBase.getConnection());

cmd.Parameters.Add("@date\_of\_delivery", MySqlDbType.Date).Value = Convert.ToDateTime(dateOfDelivery.Text);

cmd.ExecuteNonQuery();

// storage quantity change

if (status == "Повернена")

{

MySqlCommand quantityChange = new MySqlCommand("UPDATE storage SET quantity = quantity+1 WHERE book\_id = " + book\_id, dataBase.getConnection());

quantityChange.ExecuteNonQuery();

}

dataBase.closeConnection();

MessageBox.Show("Статус запису успішно змінено");

this.Close();

}

}

1. Клас AddBook.

public partial class AddBook : Window

{

public AddBook()

{

InitializeComponent();

}

private void addBook\_Click(object sender, RoutedEventArgs e)

{

if (isExist.Text == "")

{

MessageBox.Show("Оберіть один із варіантів");

}

else if (isExist.Text == "Нова книга")

{

string book\_title = book.Text.Trim();

string book\_author = author.Text.Trim();

int yeat\_of\_printing = Convert.ToInt32(yearOfPrinting.Text.Trim());

int total\_quantitiy = Convert.ToInt32(totalQuantity.Text.Trim());

string on\_shelf = shelf.Text.Trim();

DataBase dataBase = new DataBase();

dataBase.openConnection();

MySqlCommand cmd = new MySqlCommand("INSERT INTO books (title, author, year\_of\_printing) VALUES (@book\_name, @book\_author, @year\_of\_printing)", dataBase.getConnection());

cmd.Parameters.Add("@book\_name", MySqlDbType.VarChar).Value = book\_title;

cmd.Parameters.Add("@book\_author", MySqlDbType.VarChar).Value = book\_author;

cmd.Parameters.Add("@year\_of\_printing", MySqlDbType.VarChar).Value = yeat\_of\_printing;

cmd.ExecuteNonQuery();

MySqlCommand cmd2 = new MySqlCommand("INSERT INTO storage (book\_id, total\_quantity, quantity, shelf) VALUES ((SELECT MAX(book\_id) FROM books), @total\_quantity, @quantity, @shelf)", dataBase.getConnection());

cmd2.Parameters.Add("@total\_quantity", MySqlDbType.Int32).Value = total\_quantitiy;

cmd2.Parameters.Add("@quantity", MySqlDbType.Int32).Value = total\_quantitiy;

cmd2.Parameters.Add("@shelf", MySqlDbType.VarChar).Value = on\_shelf;

cmd2.ExecuteNonQuery();

dataBase.closeConnection();

MessageBox.Show("Книга успішно додана");

this.Close();

}

else if (isExist.Text == "Вже існуюча")

{

string book\_title = (string)bookComboBox.SelectedItem;

string book\_author = (string)authorComboBox.SelectedItem;

int yeat\_of\_printing = Convert.ToInt32((string)yearOfPrintingComboBox.SelectedItem);

int total\_quantitiy = Convert.ToInt32(totalQuantity.Text.Trim());

DataBase dataBase = new DataBase();

dataBase.openConnection();

MySqlCommand cmd = new MySqlCommand("UPDATE storage s SET total\_quantity = s.total\_quantity+@total\_quantity, s.quantity = s.quantity+@quantity WHERE s.book\_id = (SELECT b.book\_id FROM books b WHERE b.title = @book\_title AND b.author = @book\_author AND b.year\_of\_printing = @yeat\_of\_printing)", dataBase.getConnection());

cmd.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = book\_title;

cmd.Parameters.Add("@book\_author", MySqlDbType.VarChar).Value = book\_author;

cmd.Parameters.Add("@yeat\_of\_printing", MySqlDbType.VarChar).Value = yeat\_of\_printing;

cmd.Parameters.Add("@total\_quantity", MySqlDbType.Int32).Value = total\_quantitiy;

cmd.Parameters.Add("@quantity", MySqlDbType.Int32).Value = total\_quantitiy;

cmd.ExecuteNonQuery();

dataBase.closeConnection();

MessageBox.Show("Книга успішно додана");

this.Close();

}

}

private void isExist\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

ComboBoxItem selectedItme = (ComboBoxItem)comboBox.SelectedItem;

if ((string)selectedItme.Content == "Нова книга")

{

book.IsEnabled = true;

author.IsEnabled = true;

yearOfPrinting.IsEnabled = true;

totalQuantity.IsEnabled = true;

shelf.IsEnabled = true;

book.Visibility = Visibility.Visible;

author.Visibility = Visibility.Visible;

yearOfPrinting.Visibility = Visibility.Visible;

bookComboBox.Visibility = Visibility.Hidden;

authorComboBox.Visibility = Visibility.Hidden;

yearOfPrintingComboBox.Visibility = Visibility.Hidden;

}

else if ((string)selectedItme.Content == "Вже існуюча")

{

totalQuantity.IsEnabled = true;

shelf.IsEnabled = true;

book.Visibility = Visibility.Hidden;

author.Visibility = Visibility.Hidden;

yearOfPrinting.Visibility = Visibility.Hidden;

bookComboBox.Visibility = Visibility.Visible;

authorComboBox.Visibility = Visibility.Visible;

yearOfPrintingComboBox.Visibility = Visibility.Visible;

DataBase dataBase = new DataBase();

dataBase.openConnection();

string cmd = "SELECT book\_id, title, author FROM books";

DataTable dataTable = new DataTable();

MySqlDataAdapter dataAdapter = new MySqlDataAdapter(cmd, dataBase.getConnection());

dataAdapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (bookComboBox.Items.Contains(dataTable.Rows[i][1].ToString()) != true)

{

bookComboBox.Items.Add(dataTable.Rows[i][1].ToString());

}

}

}

dataBase.closeConnection();

}

}

private void bookComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selectedTitle = bookComboBox.SelectedItem.ToString();

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT author FROM books WHERE title = @book\_title", dataBase.getConnection());

cmd.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = selectedTitle;

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (authorComboBox.Items.Count != 0)

authorComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (authorComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

authorComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void authorComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT year\_of\_printing FROM books WHERE title = @book\_title AND author = @book\_author", dataBase.getConnection());

cmd.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = (string)bookComboBox.SelectedItem;

cmd.Parameters.Add("@book\_author", MySqlDbType.VarChar).Value = (string)authorComboBox.SelectedItem;

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (yearOfPrintingComboBox.Items.Count != 0)

yearOfPrintingComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (yearOfPrintingComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

yearOfPrintingComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

1. Клас DeleteBook.

public partial class DeleteBook : Window

{

public DeleteBook()

{

InitializeComponent();

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT title FROM books", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (bookComboBox.Items.Count != 0)

bookComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (bookComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

bookComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void bookComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT author FROM books WHERE title = '" + (string)bookComboBox.SelectedItem + "'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (authorComboBox.Items.Count != 0)

authorComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (authorComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

authorComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void authorComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT year\_of\_printing FROM books WHERE title = '" + (string)bookComboBox.SelectedItem + "' AND author = '" + (string)authorComboBox.SelectedItem +"'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (yearOfPrintingComboBox.Items.Count != 0)

yearOfPrintingComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (yearOfPrintingComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

yearOfPrintingComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void deleteBook\_Click(object sender, RoutedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("UPDATE storage SET total\_quantity = total\_quantity - @quantity, quantity = quantity - @quantity WHERE book\_id = (SELECT book\_id FROM books WHERE title = '" + (string)bookComboBox.SelectedItem + "' AND author = '" + (string)authorComboBox.SelectedItem + "' AND year\_of\_printing = '" + (string)yearOfPrintingComboBox.SelectedItem + "')", dataBase.getConnection());

cmd.Parameters.Add("@quantity", MySqlDbType.Int32).Value = Convert.ToInt32(totalQuantity.Text.Trim());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (yearOfPrintingComboBox.Items.Count != 0)

yearOfPrintingComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (yearOfPrintingComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

yearOfPrintingComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

MessageBox.Show("Книга '" + (string)bookComboBox.SelectedItem +"' автора '" +(string)authorComboBox.SelectedItem +"' рік друку '"+ (string)yearOfPrintingComboBox.SelectedItem+"' в кількості '"+(string)totalQuantity.Text +"' була успішно видалена");

this.Close();

}

}

1. Клас SendBook.

public partial class SendBook : Window

{

int staff\_id;

public SendBook()

{

InitializeComponent();

dateOfDeliveryDatePicker.Text = DateTime.Now.Date.ToString();

load\_readers\_login();

}

public void setStaffId(int id)

{

staff\_id = id;

}

private void readerIdComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT title FROM books", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (bookTitleComboBox.Items.Count != 0)

bookTitleComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (bookTitleComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

bookTitleComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void bookTitleComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

clear\_book\_data();

string selectedTitle = (string)bookTitleComboBox.SelectedItem;

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT author FROM books WHERE title = @book\_title", dataBase.getConnection());

cmd.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = selectedTitle;

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (bookAuthorComboBox.Items.Count != 0)

bookAuthorComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (bookAuthorComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

bookAuthorComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void bookAuthorComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selectedTitle = (string)bookTitleComboBox.SelectedItem;

string selectedAuthor = (string)bookAuthorComboBox.SelectedItem;

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT year\_of\_printing FROM books WHERE title = @book\_title AND author = @bool\_author", dataBase.getConnection());

cmd.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = selectedTitle;

cmd.Parameters.Add("@bool\_author", MySqlDbType.VarChar).Value = selectedAuthor;

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (yearOfPrintingComboBox.Items.Count != 0)

yearOfPrintingComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (yearOfPrintingComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

yearOfPrintingComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void sendBookButton\_Click(object sender, RoutedEventArgs e)

{

string reader\_login = (string)readerIdComboBox.SelectedItem;

string book\_title = (string)bookTitleComboBox.SelectedItem;

string book\_author = (string)bookAuthorComboBox.SelectedItem;

int yeat\_of\_printing = Convert.ToInt32(yearOfPrintingComboBox.SelectedItem);

string date\_of\_delivery = dateOfDeliveryDatePicker.Text.Trim();

if (dateOfDeliveryDatePicker.SelectedDate >= DateTime.Now.Date)

{

if ((reader\_login != "") && (book\_title != "") && (book\_author != "") && (yeat\_of\_printing != 0))

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

// get reader\_id

MySqlCommand get\_reader\_id = new MySqlCommand("SELECT reader\_id FROM readers WHERE login = @login", dataBase.getConnection());

get\_reader\_id.Parameters.Add("@login", MySqlDbType.VarChar).Value = reader\_login.Trim();

string reader\_id = get\_reader\_id.ExecuteScalar().ToString();

// get book\_id

MySqlCommand get\_book\_id = new MySqlCommand("SELECT book\_id FROM books WHERE title = @book\_title AND author = @book\_author AND year\_of\_printing = @year\_of\_printing", dataBase.getConnection());

get\_book\_id.Parameters.Add("@book\_title", MySqlDbType.VarChar).Value = book\_title.Trim();

get\_book\_id.Parameters.Add("@book\_author", MySqlDbType.VarChar).Value = book\_author.Trim();

get\_book\_id.Parameters.Add("@year\_of\_printing", MySqlDbType.Int32).Value = yeat\_of\_printing;

string book\_id = get\_book\_id.ExecuteScalar().ToString();

// make note in the logbook

MySqlCommand cmd = new MySqlCommand("INSERT INTO logbook (reader\_id, staff\_id, book\_id, book\_title, date\_of\_issue, date\_of\_delivery, status)" +

"VALUES(@reader\_id," + staff\_id + ", @book\_id, (SELECT title FROM books WHERE book\_id = @book\_title), @date\_of\_issue, @date\_of\_delivery, 'Видана')",

dataBase.getConnection());

cmd.Parameters.Add("@reader\_id", MySqlDbType.Int32).Value = Convert.ToInt32(reader\_id);

cmd.Parameters.Add("@book\_id", MySqlDbType.Int32).Value = Convert.ToInt32(book\_id);

cmd.Parameters.Add("@book\_title", MySqlDbType.Int32).Value = Convert.ToInt32(book\_id);

cmd.Parameters.Add("@date\_of\_issue", MySqlDbType.Date).Value = DateTime.Now.ToString(@"yyyy-MM-dd");

cmd.Parameters.Add("@date\_of\_delivery", MySqlDbType.Date).Value = Convert.ToDateTime(dateOfDeliveryDatePicker.Text);

cmd.ExecuteNonQuery();

// set reader/staff full name

MySqlCommand set\_reader\_full\_name = new MySqlCommand("UPDATE logbook l JOIN readers r ON l.reader\_id = r.reader\_id SET l.reader\_full\_name = CONCAT(r.last\_name, ' ', r.first\_name) WHERE r.reader\_id = l.reader\_id", dataBase.getConnection());

set\_reader\_full\_name.ExecuteNonQuery();

MySqlCommand set\_staff\_full\_name = new MySqlCommand("UPDATE logbook l JOIN staff s ON l.staff\_id = s.staff\_id SET l.staff\_full\_name = CONCAT(s.last\_name, ' ', s.first\_name) WHERE s.staff\_id = l.staff\_id", dataBase.getConnection());

set\_staff\_full\_name.ExecuteNonQuery();

// change quantity in storage

MySqlCommand quantityChange = new MySqlCommand("UPDATE storage SET quantity = quantity-1 WHERE book\_id = @book\_id", dataBase.getConnection());

quantityChange.Parameters.Add("@book\_id", MySqlDbType.Int32).Value = Convert.ToInt32(book\_id);

quantityChange.ExecuteNonQuery();

dataBase.closeConnection();

MessageBox.Show("Виконано");

this.Close();

}

else

{

MessageBox.Show("Заповніть всі поля");

}

}

else

{

MessageBox.Show("Дата повернення повинна бути більшою або рівною поточної дати");

}

}

private void load\_readers\_login()

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand cmd = new MySqlCommand("SELECT login FROM readers", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (readerIdComboBox.Items.Count != 0)

readerIdComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (readerIdComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

readerIdComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

private void clear\_book\_data()

{

bookAuthorComboBox.Items.Clear();

yearOfPrintingComboBox.Items.Clear();

}

}

1. Клас DeleteUser

public partial class DeleteUser : Window

{

public DeleteUser()

{

InitializeComponent();

}

private void userRoleComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedItem = (ComboBoxItem)userRoleComboBox.SelectedValue;

var selectedRole = (string)selectedItem.Content;

string cmdString = "";

clearUserInfo();

if (selectedRole != "")

{

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

if (selectedRole == "Читач")

{

cmdString = "SELECT login FROM readers";

}

else if (selectedRole == "Співробітник")

{

cmdString = "SELECT login FROM staff";

}

else if (selectedRole == "Адміністратор")

{

cmdString ="SELECT login FROM admins";

}

MySqlCommand cmd = new MySqlCommand(cmdString, dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

if (userLoginComboBox.Items.Count != 0)

userLoginComboBox.Items.Clear();

for (int i = 0; i < dataTable.Rows.Count; i++)

{

if (userLoginComboBox.Items.Contains(dataTable.Rows[i][0].ToString()) != true)

{

userLoginComboBox.Items.Add(dataTable.Rows[i][0].ToString());

}

}

}

dataBase.closeConnection();

}

}

private void userLoginComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var selectedItem = (ComboBoxItem)userRoleComboBox.SelectedValue;

var selectedRole = (string)selectedItem.Content;

clearUserInfo();

DataBase dataBase = new DataBase();

DataTable dataTable = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

if (selectedRole == "Читач")

{

MySqlCommand cmd = new MySqlCommand("SELECT first\_name, last\_name, class, telephone FROM readers WHERE login = '" + (string)userLoginComboBox.SelectedItem +"'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

firstNameTextBlock.Text = dataTable.Rows[0][0].ToString();

lastNameTextBlock.Text = dataTable.Rows[0][1].ToString();

classTextBlock.Text = dataTable.Rows[0][2].ToString();

telephoneTextBlock.Text = dataTable.Rows[0][3].ToString();

}

}

else if (selectedRole == "Співробітник")

{

MySqlCommand cmd = new MySqlCommand("SELECT first\_name, last\_name, telephone FROM staff WHERE login = '" + (string)userLoginComboBox.SelectedItem + "'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

firstNameTextBlock.Text = dataTable.Rows[0][0].ToString();

lastNameTextBlock.Text = dataTable.Rows[0][1].ToString();

telephoneTextBlock.Text = dataTable.Rows[0][2].ToString();

}

}

else if (selectedRole == "Адміністратор")

{

MySqlCommand cmd = new MySqlCommand("SELECT first\_name, last\_name FROM admins WHERE login = '" + (string)userLoginComboBox.SelectedItem + "'", dataBase.getConnection());

adapter.SelectCommand = cmd;

adapter.Fill(dataTable);

if (dataTable.Rows.Count > 0)

{

firstNameTextBlock.Text = dataTable.Rows[0][0].ToString();

lastNameTextBlock.Text = dataTable.Rows[0][1].ToString();

}

}

dataBase.closeConnection();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

var selectedItem = (ComboBoxItem)userRoleComboBox.SelectedValue;

var selectedRole = (string)selectedItem.Content;

string cmdString = "";

if ((string)userLoginComboBox.SelectedItem != null)

{

DataBase dataBase = new DataBase();

dataBase.openConnection();

if (selectedRole == "Читач")

{

cmdString = "DELETE FROM readers WHERE login = '" + (string)userLoginComboBox.SelectedItem + "'";

}

else if (selectedRole == "Співробітник")

{

cmdString = "DELETE FROM staff WHERE login = '" + (string)userLoginComboBox.SelectedItem + "'";

}

else if (selectedRole == "Адміністратор")

{

cmdString = "DELETE FROM admins WHERE login = '" + (string)userLoginComboBox.SelectedItem + "'";

}

userLoginComboBox.Items.Remove(userLoginComboBox.SelectedItem);

MySqlCommand cmd = new MySqlCommand(cmdString, dataBase.getConnection());

cmd.ExecuteNonQuery();

dataBase.closeConnection();

MessageBox.Show("Користувач успішно видалений");

this.Close();

}

else

{

MessageBox.Show("Оберіть користувача для видалення");

}

}

private void clearUserInfo()

{

firstNameTextBlock.Text = "";

lastNameTextBlock.Text = "";

classTextBlock.Text = "";

telephoneTextBlock.Text = "";

}

}