МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення

для автоматизації роботи бібліотеки-філіалу

Виконала: студентка групи 4151

_____ Мустратова А.О.

(підпис, ПІБ)

Керівник роботи:

доцент каф. ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М

(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Мустратовій Анастасії Олексіївні
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для автоматизації роботи бібліотеки-філіалу

Керівник роботи Макарова Лідія Миколаївна

Затверджені наказом ректора №256-уч від «11» травня 2022 р.

2. Термін подання роботи: 10.06.2022 р.

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів

Титульний аркуш, Вступ, Основні цілі розробки програмного забезпечення, Діаграма варіантів використання програмного забезпечення для автоматизації роботи бібліотеки-філіалу, Концептуальна модель бази даних програмного забезпечення для автоматизації роботи бібліотеки-філіалу, Діаграми діяльності, Діаграма класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу, Діаграма послідовності, Інструментарій розробки, Результати розробки, Висновки, Заключний аркуш.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 11.05.2022 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	24.03.2022	ПП*
2.	Опис та аналіз предметної галузі	28.03.2022	ПП*
3.	Постановка задачі	31.03.2022	ПП*
4.	Ескізний проєкт програмного забезпечення	27.04.2022	
5.	Технічний проєкт програмного забезпечення	06.05.2022	
6.	Робочий проєкт програмного забезпечення	13.05.2022	
7.	Підготовка розділу Результати розробки програмного забезпечення	17.05.2022	
8.	Підготовка розділу з охорони праці	18.05.2022	ПП*
9.	Підготовка розділу Висновки	19.05.2022	
10.	Оформлення списку використаних джерел та додатків	20.05.2022	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	10.06.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	12.06.2022	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	14.06.2022	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	24.06.2022	

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

Студент _____ Мустратова А.О.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – розробці програмного забезпечення для автоматизації роботи бібліотеки-філіалу, котре дозволить працівникам бібліотеки-філіалу спростити та пришвидшити свою роботу та полегшити ведення звітності.

У кваліфікаційній роботі представлені аналіз та опис предметної галузі за темою кваліфікаційної роботи, аналіз існуючих аналогів та постанова задачі на розробку програмного забезпечення. Розроблено ескізний, технічний та робочий проекти програмного забезпечення, представлено результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена на 88 сторінках машинописного тексту та містить: 13 рисунків, 15 таблиць, 5 додатків, список використаних джерел з 17 найменувань.

ABSTRACT

The qualification work is dedicated to solving the practical task of software engineering, characterized by complexity and uncertainty of conditions – software development for work automation of the library-branch, which will allow simplifying and speeding up the work of the library-branch employees and making it easier to keep an accountability system.

In qualification work presents subject area analysis and description based on the qualification work thesis, analysis of existing analogues and formulation of the problem for software development. The sketch, technical and working projects of the software, results of development and the section on labor protection are developed.

The qualification work is presented on 88 pages of typewritten text and contains: 13 figures, 15 tables, 5 appendices, list of references from 17 names.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	база даних
ОС	операційна система
ПЗ	програмне забезпечення
ПІБ	прізвище ім’я по батькові
ПК	персональний комп’ютер
UML	unified modeling language

ВСТУП

Одним з ключових напрямків в області автоматизації бізнес-процесів з використанням інформаційних технологій є розробка баз даних, що дозволяє вирішувати проблеми зберігання, обробки і систематизації інформації згідно з індивідуальними вимогами користувача.

Збільшення обсягу та структурної складності збережених даних, розширення кола користувачів інформаційних систем призвели до широкого поширення найбільш зручних і порівняно простих для розуміння реляційних (табличних) СУБД.

Все це також можна віднести до управління бібліотекою. Існують готові рішення для автоматизації роботи бібліотеки. До них відносяться, наприклад:

- програмне забезпечення «Моя бібліотека» – це інтегрована система для управління бібліотеками [1];
- «Calibre 3.35.0» – забезпечує створення та ведення бібліотечних електронних каталогів [2];
- «MyLib 1.0» – універсальна програма, яка призначена для ведення обліку книг та читачів [3].

Ці та інші аналогічні програмні рішення добре працюють при використанні у великих бібліотеках, зокрема, центральних бібліотеках міст. Але коли мова йде про невелику та відокремлену бібліотеку-філіал, дані програмні системи мають наступні недоліки:

- для використання зазначеного програмного забезпечення потрібно мати необхідні специфічні навички для налаштування серверної частини або відповідного спеціаліста у штаті співробітників бібліотеки-філіалу;
- надлишкова функціональність, яка не потрібна для роботи бібліотеки-філіалу.

Кваліфікаційна робота передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та

невизначеністю умов, а саме розробку програмного забезпечення для автоматизації роботи бібліотеки-філіалу, котре дозволить працівникам бібліотеки-філіалу спростити та пришвидшити свою роботу та полегшити ведення звітності.

Для досягнення поставленої мети необхідно:

– проаналізувати інформаційну систему бібліотеки-філіалу та визначити специфіку її роботи, розглянути існуючі програмні рішення та здійснити постановку задачі до розробки програмного забезпечення;

– на основі постановки задачі розробити технічне завдання на розробку програмного забезпечення для автоматизації роботи бібліотеки-філіалу;

– розробити ескізний, технічний та робочий проекти програмного забезпечення для автоматизації роботи бібліотеки-філіалу;

– здійснити тестування та випробування розробленого програмного забезпечення;

– розробити документацію до програмного забезпечення.

В результаті повинні отримати програмний продукт, який забезпечить якісне подання інформації, вирішить проблему достовірності і зменшить кількість часу, затраченого на паперову роботу працівниками бібліотеки-філіалу.

1 АНАЛІЗ ТА ЗАГАЛЬНИЙ ОГЛЯД ІНФОРМАЦІЙНОЇ СИСТЕМИ БІБЛІОТЕКИ-ФІЛІАЛУ

1.1 Загальний огляд інформаційної системи бібліотеки-філіалу

Для розробки програмного забезпечення та проектування бази даних предметною галуззю є бібліотека. У бібліотеці зібрані книги різних авторів, видавництв, років видання – все це становить предметну галузь дослідження.

База даних створюється для економії часу при пошуку книг, а так само швидкого отримання відповідей на такі питання, як:

- наявність або відсутність даної книги в бібліотеці;
- де знаходиться шукана книга;
- хто автор шуканої книги;
- які книги даного автора зібрані в бібліотеці;
- скільки примірників цієї книги є в бібліотеці.

Для ведення бібліотечних каталогів, організації пошуку необхідних видань і бібліотечної статистики в базі повинні зберігатися відомості, велика частина яких розміщуються в анотованих каталожних картках. Аналіз запитів на літературу (як читачами, так і співробітниками бібліотек) показує, що для пошуку відповідних видань (за тематикою, автором, художником, видавництвом і т.п.) і відбору потрібного видання (наприклад, за інструкцією) слід виділити наступні атрибути каталожної картки:

- автор (прізвище та ім`я (ініціали) або псевдонім кожного автора видання);
- назва (заголовок) видання;
- номер тому (частини, книги, випуску);
- вид видання (збірник, довідник, монографія і т.п.);
- укладач (прізвище та ім`я (ініціали) кожного з упорядників видання);
- мова, з якої виконаний переклад видання;

– перекладач (прізвище та ініціали кожного перекладача);

– під чиєю редакцією (прізвище та ім`я (ініціали) кожного з титульних редакторів);

– художник (прізвище та ім`я (ініціали) кожного художника-ілюстратора) – для художніх видань, ілюстрованих оригінальними малюнками;

– повторність видання (друге, одинадцяте і т.п.);

– характер перевидання (виправлене, доповнене, перероблене, стереотипне і т.п.);

– місце видання (місто);

– видавництво (назва видавництва);

– рік випуску видання;

– авторський знак.

Бібліотечний шифр і авторський знак використовуються при складанні каталогів та організації розстановки видань на полицях: за змістом (відповідно до бібліотечного шифру) і алфавітом (відповідно до авторського знаку).

Існує 7 класів розподілу книг:

1. Природні науки.

2. Техніка. Технічні науки.

3. Сільське і лісове господарство.

4. Охорона здоров'я.

5/6. Суспільні та гуманітарні науки.

7. Бібліографічні посібники. Довідкові видання. Журнали.

Кожен з семи класів ділиться на підкласи і наступні ступені ділення, наприклад, клас 2. Техніка. Технічні науки:

22 Радіоелектроніка.

22.97 Обчислювальна техніка.

22.973 Електронні обчислювальні машини і пристрої.

22.973.2 Електронні обчислювальні машини та пристрої дискретної дії.

Авторський знак, що складається з першої літери прізвища (псевдоніма) автора або назви видання (для видань без автора) і числа, відповідного стилю,

найбільш наближається з написання до перших літер прізвища (назви), спрощує розстановку книжок на полицях в алфавітному порядку.

До об'єктів і атрибутів, що дозволяють охарактеризувати окремі екземпляри видань, місця їх зберігання і читачів, можна віднести [4]:

– дата придбання конкретної книги;
– ціна конкретної книги;
– номер читацького квитка (формуляра);
– прізвище читача, ім'я, по батькові;
– адреса читача;
– телефон читача;
– дата видачі читачеві конкретної книги;
– дата повернення книги.

1.2 Аналіз існуючих програмних рішень

У ході аналізу існуючих програмних рішень було розглянуто три програми, а саме: «Моя бібліотека» [1], «Calibre 3.35.0» [2], «MyLib 1.0» [3]. Розглянемо більш детально кожну із них.

«Моя бібліотека»

Автоматизована інформаційна бібліотечна система «Моя бібліотека» – це інтегрована система для управління бібліотеками. Вона дає такі можливості [1]:

– Каталогізація всіх типів документів (книги, періодика, електронні ресурси і т.д.).
– Аналітичний розпис і розпис статей періодичних видань.
– Формування довідників (нормативні файли).
– Ведення читачів (картотека читачів, видач).

– Пошук і бронювання книжок з урахуванням наявності примірників, видач читачам (передбачена постановка в чергу на бронювання), OPAC пошук і on-line бронювання для користувачів.

Працівник бібліотеки може:

– здійснювати каталогізацію книг / підручників і інших типів документів, що знаходяться у фонді бібліотеки;

– виконувати пошук за алфавітом;

– вводити дані про кожного читача;

– автоматизувати процес видачі / повернення книг / підручників;

– отримувати списки боржників, списки забронюваних книг, поточні видачі, список читачів з простроченим абонементом та ін.;

– формувати списки рекомендованих підручників і навчальних посібників;

– створювати віртуальні виставки з новими надходженнями.

Робота повністю через Web-інтерфейс, що дозволяє скоротити роботу по установці та супроводу системи тільки на сервері.

Велика перевага системи – можливість роботи в багатомовному оточенні, оскільки дані зберігаються і обробляються в кодуванні UNICODE. Таким чином, можна вводити документи на будь-яких мовах. Крім того, таке рішення дозволяє легко адаптувати систему для роботи на різних робочих мовах.

Для роботи в рамках корпорацій є модуль пошуку і імпорту записів з сервером Z39.50, що дозволяє здійснювати:

– корпоративну каталогізацію;

– ведення каталогів електронних ресурсів;

– відкритий доступ до електронних каталогів і повнотекстових баз даних.

Реалізована система управління бронюванням і рухом фонду.

Передбачена можливість використання технології штрих-кодування для книг і читацьких квитків.

Система орієнтована на можливість роботи як в локальній бібліотечній

мережі (Intranet), так і в корпоративній мережі бібліотек (Internet).

Наявність модуля OPAC забезпечує через Web-інтерфейс доступ користувачів до електронних каталогів і оформлення замовлення на бронювання книг.

Дана система безумовно може розглядатися як вдале рішення для впровадження процесу автоматизації в бібліотеках завдяки раціональному і дружньому інтерфейсу, простоті в установці і прийнятній ціни.

З мінусів можливо визначити такі:

– необхідність встановити та налаштувати сервер Apache, PHP, My SQL, звичайному користувачу це буде важко зробити;

– для запуску сервера потрібно мати необхідні навички або відповідних спеціалістів. Також, щоб розширити можливості програми, потрібно встановити додаткові модулі, що також створює незручності для користувача [1].

«Calibre 3.35.0»

Забезпечує створення та ведення бібліотечних електронних каталогів. Дані про комплектування вводяться у відповідні каталоги і можуть бути перенесені в інший електронний каталог з одночасною модифікацією інформації [2].

Забезпечує автоматичне створення повного комплекту каталожних карток і передбачує можливість попереднього налаштування і редагування картки.

Виконує пошук по будь-якому набору пошукових елементів і їх комбінацій. Крім простого і ясного пошукового інтерфейсу, реалізована можливість складання складних пошукових запитів.

Має готові словники ключових слів, що містять близько сорока тисяч термінів з природничих і гуманітарних наук. Програма допускає їх коригування, а також створення своїх власних словників для будь-якого елемента бібліографічного опису. Словники використовуються при введенні і пошуку інформації для вибору термінів з готових списків.

Має широкі можливості налаштування. Адміністратор може змінювати список бібліотечних каталогів, бібліографічних елементів, створювати і видаляти

робочі листи введення, додавати і видаляти пошукові елементи, розробляти вихідні форми і багато іншого.

До мінусів даного програмного забезпечення можливо віднести те, що програма більше орієнтована на роботу з електронними книгами [2].

«MyLib 1.0»

З допомогою цієї програми можливо досить легко створити базу даних по всій бібліотеці і стежити за видачею книг читачам. За своїми можливостями MyLib нагадує платну програму «Облік книг» [3].

Є список авторів книг, присутніх в бібліотеці, а також список і опис книг вибраного автора, інформація про видачу обраної книги. Є дві додаткові робочі вкладки (Пошук і Список читачів), а також можливість додавання нових книг.

Починати роботу слід зі створення бази даних по книгах. На жаль, в програмі немає деяких полів, типу ISBN, кількості сторінок і деяких інших, однак при особливому бажанні такі поля можна буде вручну додати в базу даних програми. Виділивши книгу в каталозі «Книги автора», можливо внести зміни в дані про неї, а також видати книгу на руки читачу.

Для зручності роботи з великою кількістю людей, потрібно створити ще й базу даних читачів. На жаль, засобами програми «MyLib» створити таку базу неможливо, але ніщо не заважає заповнити її безпосередньо, використовуючи MS Access.

Є два варіанти пошуку: по авторам або за назвами книг. Однак, якщо книги, відповідної заданим параметрам пошуку немає в базі, то список залишиться порожнім, додаткових повідомлень користувач при цьому не побачить [3].

Плюси:

- простий інтерфейс і управління програмою;
- необмежена кількість записів в базі даних;
- зручна система стеження за виданими книгами.

Мінуси:

- незручний механізм пошуку;

– немає вбудованого редактора списку читачів;
– немає функції імпорту, експорту і друку даних з програми;
– немає можливості додавання додаткових полів в опис книг;
– наявність прямого доступу до основної бази даних.

1.3 Постановка задачі до розробки програмного забезпечення для автоматизації роботи бібліотеки-філіалу

Виходячи з аналізу предметної області та дослідження існуючих програмних рішень, було вирішено розробити програмне забезпечення для автоматизації роботи бібліотеки-філіалу.

Користувачем даного програмного забезпечення є співробітники бібліотеки. За підтримку програмного забезпечення відповідає системний адміністратор програмного забезпечення, який слідкує за цілісністю бази даних та збереженням інформації.

В результаті в БД використовуються такі вхідні дані:

1) Читацький квиток:
- ПІБ;
- адреса;
- телефон.

2) Інформація по книзі:
- заголовок;
- тип видання;
- номер тома;
- бібліотечний шифр;
- назва видавництва;
- ПІБ автора;
- мова;
- ПІБ художника;

- ПІБ перекладача;
- тип палітурки;
- об’єм книги;
- дата отримання;
- ціна.

3) Інформація про зарезервовані книги:

- номер читацького квитка;
- названа книги;
- дата замовлення.

4) Інформація про видані книги:

- номер читацького квитка;
- назва книги;
- дата видачі;
- термін;
- дата повернення.

Вихідні данні:

1) Звіт «Видані книги»:

- номер читацького квитка;
- назва книги;
- дата видачі книги.

2) Звіт «Інвентаризація

- назва книги;
- тип видання;
- автор;
- дата отримання;
- ціна.

3) Звіт «Планове повернення»;

- номер читацького квитка;
- назва книги;
- дата повернення книги.

4) Звіт «Заборгованість».

- номер читацького квитка;
- назва книги;
- дата повернення книги.

Програмне забезпечення повинно надавати наступні можливості для користувача:

- авторизація;
- пошук необхідної інформації;
- оформлення або редагування читацького квитка;
- оформлення видачі книги;
- оформлення бронювання книги;
- перегляд інформації про книги та читацькому квитку;
- занесення та редагування даних про книгу;
- видалення інформації про книги;
- формування необхідних звітів.

У програмному забезпеченні доступ до даних має тільки співробітник бібліотеки. Для входу в систему необхідно пройти авторизацію.

Детальніша постановка задачі представлена у технічному завданні, яке наведено у додатку А.

Було виділено вимоги до складу та параметрів технічних засобів.

Система повинна задовольняти вище вказаним вимогам на комп’ютері наступній мінімальній комплекції:

- CPU: Intel(R) Celeron(R) – 2.16 МГц;
- RAM: 2 GB;
- HDD: 10 GB вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7. Необхідна наявність встановленого Java Runtime Environment (JRE 1.8).

2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ БІБЛІОТЕКИ-ФІЛІАЛУ

2.1 Ескізний проект програмного забезпечення

2.1.1 Вибір мови моделювання

При створенні складних інженерних систем прийнято використовувати прийоми моделювання. Складність більшості створюваних сьогодні програмних систем не поступається складності багатьом інженерним спорудженням, тому моделювання програмних систем є досить актуальним завданням. Більш того, у таких концепціях, як MDA (Model DrivenA rchitecture – архітектура на основі моделей) і MDD (Model Driven Development – розробка на базі моделей), моделям приділяється центральна роль у процесі створення програмного продукту. Основною ідеєю цих концепцій є представлення процесу створення програмного продукту у вигляді ланцюжка трансформацій його вихідної моделі в готову програмну систему.

Майже у всіх інструментальних засобах, що втілює ідеї MDD, як мова моделювання використовується мова UML (Unified Modeling Language – уніфікована мова моделювання), цілком або які-небудь її частини.

UML – це мова, призначена для візуалізації, специфікації, конструювання й документування програмних систем. Слово «уніфікована» у назві мови означає, що UML може використовуватися для моделювання широкого кола додатків від вбудованих систем і систем реального часу до розподілених web-додатків. Виразні засоби мови дозволяють описати систему зі всіх точок зору, що мають відношення до розробки й розгортання [5].

Мова UML призначена для рішення наступних завдань:

– надати в розпорядження користувачів готову до використання виразну потужну мову візуального моделювання, що дозволяє розробляти осмислені моделі й обмінюватися ними;

– передбачити внутрішні механізми розширюваності й спеціалізації базових концепцій мови;

– забезпечити максимальну незалежність проекту створення програмного забезпечення від конкретних мов програмування й процесів розробки;

– забезпечити формальну основу для однозначної інтерпретації мови;

– стимулювати розширення ринку об’єктно-орієнтованих інструментальних засобів створення програмного забезпечення;

– інтегрувати кращий практичний досвід використання мови й реалізації програмних засобів його підтримки [5].

2.1.2 Побудова діаграми варіантів використання

Діаграма варіантів використання (Use Case Diagram) – це UML діаграма за допомогою якої в графічному вигляді можна зобразити вимоги до розроблюваної системі. Діаграма варіантів використання – це вихідна концептуальна модель проектованої системи, вона не описує внутрішній устрій системи.

Діаграма варіантів використання складається з ряду елементів. Основними елементами є: варіанти використання або прецедент (use case), актор або дійова особа (actor) і відносини між акторами і варіантами використання (relationship).

Актором називається будь-який об'єкт, суб'єкт або система, що взаємодіє з модельованої бізнес-системою ззовні для досягнення своїх цілей або вирішення певних завдань. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка служить джерелом впливу на модельовану систему [6].

У мові UML є кілька стандартних видів відносин між акторами й варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

Діаграма варіантів використання програмного забезпечення для

автоматизації роботи бібліотеки-філіалу представлена на рисунку 2.1.

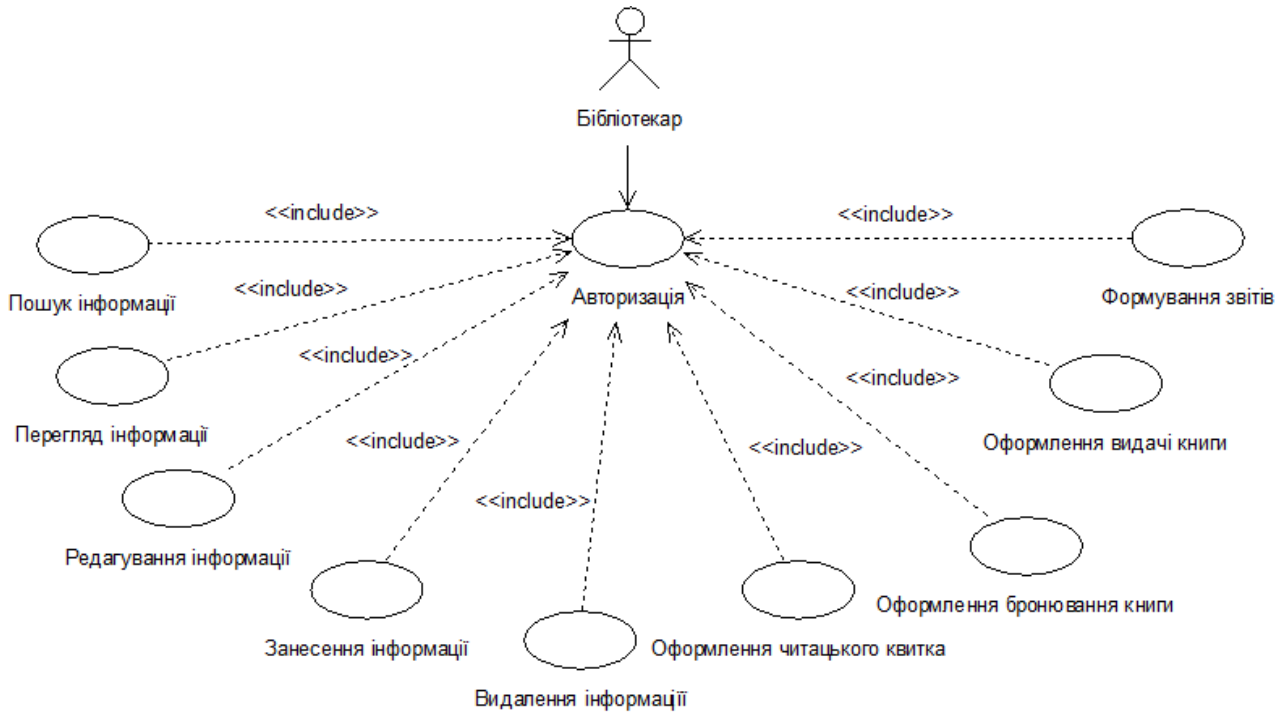

Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення для автоматизації роботи бібліотеки-філіалу

На моделі варіантів використання відображено 1 актор та 10 прецедентів. Бібліотекар має наступні функції: авторизація, пошук інформації, перегляд інформації, редагування інформації, занесення інформації, видалення інформації, оформлення читацького квитка, оформлення бронювання книги, оформлення видачі книги, формування звітів.

До прецеденту «Перегляд інформації» відносяться наступні таблиці: автор, видавництво, інформація по книзі, перекладач, укладач, художник, читацький квиток, бронювання книги, видача, мова, палітурка, тип видання.

До прецеденту «Редагування інформації» відносяться наступні таблиці: автор, видавництво, інформація по книзі, перекладач, укладач, художник, читацький квиток, бронювання книги, видача, мова, палітурка, тип видання.

До прецеденту «Занесення інформації» відносяться наступні таблиці: автор, видавництво, інформація по книзі, перекладач, укладач, художник, читацький квиток, бронювання книги, видача, мова, палітурка, тип видання.

До прецеденту «Видалення інформації» відносяться наступні таблиці: автор, видавництво, інформація по книзі, перекладач, укладач, художник, читацький квиток, бронювання книги, видача, мова, палітурка, тип видання.

2.1.3 Розробка специфікації варіантів використання

Розглянемо більш детально кожний варіант використання. Потік подій варіанта використання складається з:

- короткого опису;
- передумов;
- основного потоку подій;
- альтернативного потоку подій;
- постумов.

Ці складові частини для кожного варіанту використання розроблюваного програмного забезпечення приведені у таблицях 2.1 – 2.10.

Таблиця 2.1 – Опис варіанту використання "Авторизація"

Опис	Складова
1	2
Короткий опис	Дана функція дозволяє особі пройти перевірку для доступу до системи.
Передумови	Система запущена та виводить форму ідентифікацій. Наявність ідентифікаційного запису у базі.
Основний потік подій	1. Система виводить форму для ведення логіну та паролю; 2. Користувач водить логін та пароль та підтверджує введення; 3. Система перевіряє вірність ведення логіну та паролю; 4. Якщо дані достовірні то завантажується програма.

Продовження табл. 2.1

1	2
Альтернативний потік подій	1 Система виводить форму для ведення логіну та паролю; 2 Користувач водить логін та пароль та підтверджує введення; 3 Система перевіряє вірність ведення логіну та паролю; 4 Повідомлення про те, що логін або пароль не вірні; 5 Надання повторного ведення даних.
Постумови	Користувач авторизований в системі.

Таблиця 2.2 – Опис варіанту використання "Пошук інформації"

Опис	Складова
Короткий опис	Дана функція дозволяє користувачу здійснювати пошук необхідної інформації.
Передумови	Користувач авторизований у системі.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмною; 3. Користувач обирає таблицю по якій буде відбуватись пошук необхідних даних; 4. Користувач водить необхідні дані для пошуку; 5. Підтверджує пошук даних.
Альтернативний потік подій	Відсутній.
Постумови	Вивід знайденої інформації.

Таблиця 2.3 – Опис варіанту використання "Перегляд інформації"

Опис	Складова
1	2
Короткий опис	Призначений для перегляду довідкової інформації.
Передумови	Користувач авторизований у системі.
Редагування інформації	
Короткий опис	Призначений для редагування довідкової інформації.
Передумови	Користувач авторизований у системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач вибирає потрібний довідник; 6. Система відкриває список записів у довіднику; 4. Користувач вибирає запис та натискає «Змінити». 5. Користувач змінює всі потрібні реквізити елементу та натискає кнопку «Зберегти»; 6. Система обробляє інформацію, якщо нема помилок зберігає запис.

Продовження табл. 2.3

1	2
Альтернативний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 7. Система виводить список довідників; 8. Користувач вибирає потрібний довідник; 9. Система відкриває список записів у довіднику; 4. Користувач вибирає запис та натискає «Змінити». 5. Користувач змінює всі потрібні реквізити елементу та натискає кнопку «Відміна»; 6. Система обробляє інформацію, якщо нема помилок зберігає запис.
Постумови	Відредагований запис в довіднику.

Таблиця 2.4 – Опис варіанту використання "Редагування інформації"

Опис	Складова
Короткий опис	Призначений для редагування довідкової інформації.
Передумови	Користувач авторизований у системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач вибирає потрібний довідник; 6. Система відкриває список записів у довіднику; 7. Користувач вибирає запис та натискає «Змінити». 8. Користувач змінює всі потрібні реквізити елементу та натискає кнопку «Зберегти»; 9. Система обробляє інформацію, якщо нема помилок зберігає запис.
Альтернативний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач вибирає потрібний довідник; 6. Система відкриває список записів у довіднику; 7. Користувач вибирає запис та натискає «Змінити». 8. Користувач змінює всі потрібні реквізити елементу та натискає кнопку «Відміна»; 9. Система обробляє інформацію, якщо нема помилок зберігає запис.
Постумови	Відредагований запис в довіднику.

Таблиця 2.5 – Опис варіанту використання "Занесення інформації"

Опис	Складова
Короткий опис	Призначений для заповнення довідкової інформації.
Передумови	Користувач авторизований у системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач вибирає потрібний довідник; 6. Система відкриває список записів у довіднику; 7. Користувач вибирає «Додати». 8. Система виводить пустий запис; 9. Користувач заповнює всі потрібні реквізити нового елементу та натискає зберегти; 10. Система обробляє інформацію, якщо нема помилок зберігає запис.
Альтернативний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач вибирає потрібний довідник; 6. Система відкриває список записів у довіднику; 7. Користувач вибирає «Додати». 8. Система виводить пустий запис; 9. Користувач заповнює всі потрібні реквізити нового елементу та натискає кнопку «Відміна»; 10. Система обробляє інформацію, якщо нема помилок зберігає запис.
Постумови	Створено новий запис в довіднику.

Таблиця 2.6 – Опис варіанту використання "Видалення інформації"

Опис	Складова
Короткий опис	Призначений для видалення довідкової інформації.
Передумови	Користувач авторизований у системі. Користувач повинен вибрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи з довідниками; 4. Система виводить список записів у довіднику; 5. Користувач вибирає потрібний запис для видалення, після чого натискає «Видалити». 6. Система обробляє інформацію, якщо нема помилок видаляє запис.
Альтернативний потік подій	Відсутній.
Постумови	Видалення запису у довіднику.

Таблиця 2.7 – Опис варіанту використання "Оформлення читацького квитка"

Опис	Складова
Короткий опис	Призначення для створення нового читацького квитка.
Передумови	Користувач авторизований в системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація. 2. Система виводить головне вікно роботи з програмною; 3. Користувач обирає поле для роботи з довідниками; 4. Система виводить список довідників; 5. Користувач обирає довідник «Читацький квиток» та натискає кнопку «Додати»; 6. В новій формі заповнює всі необхідні реквізити нового елементу та натискає кнопку «Зберегти» ; 7. Система обробляє інформацію, якщо немає помилок, зберігає запис.
Альтернативний потік подій	Відсутній.
Постумови	Створено новий читацький квиток.

Таблиця 2.8 – Опис варіанту використання "Оформлення бронювання книги"

Опис	Складова
Короткий опис	Призначений для оформлення бронювання книги
Передумови	Користувач авторизований в системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмною; 3. Користувач обирає поле для роботи з документами. 4. Обирає документ «Бронювання книги» та натискає кнопку «Добавить»; 5. В новій формі заповнює всі необхідні елементи нового елементу та натискає кнопку «Зберегти»; 6. Система обробляє інформацію, якщо немає помилок, зберігає запис.
Альтернативний потік подій	Відсутній.
Постумови	Сформовано нове бронювання на книгу.

Таблиця 2.9 – Опис варіанту використання "Оформлення видачі книги"

Опис	Складова
Короткий опис	Призначений для оформлення видачі книги
Передумови	Користувач авторизований в системі. Користувач повинен обрати довідник для роботи з ним.
Основний потік подій	1. Авторизація; 2. Система виводить головне вікно роботи з програмною; 3. Користувач обирає поле для роботи з документами; Обирає документ «Видача» та натискає кнопку «Додати»; 4. В новій формі заповнює всі необхідні елементи нового елементу та натискає кнопку «Зберегти»; 5. Система обробляє інформацію, якщо немає помилок, зберігає запис.
Альтернативний потік подій	Відсутній.
Постумови	Сформовано видача книги.

Таблиця 2.10 – Опис варіанту використання "Формування звітів"

Опис	Складова
Короткий опис	Призначений для формування звітів.
Передумови	Користувач авторизований в системі. Користувач повинен обрати звіт для роботи з ним.
Основний потік подій	1. Ідентифікація; 2. Система виводить головне вікно роботи з програмою; 3. Користувач вибирає поле для роботи зі звітами; 4. Система ви водить список звітів 5. Користувач вибирає потрібний звіт; 6. Система виводить форму для роботи зі звітом; 7. Користувач заповнює всі потрібні для формування звіту реквізити та натискає «Сформувати»; 8. Система обробляє інформацію, якщо нема помилок виводить сформований звіт.
Альтернативний потік подій	Відсутній.
Постумови	Сформовано видача книги.

Таким чином, варіанти використання, які наведені у таблицях 2.1 – 2.10, описують всю необхідну функціональність розроблюваного програмного забезпечення.

2.1.4 Побудова концептуальної моделі бази даних

Моделі «сутність – зв'язок» зручні тим, що процес створення моделі є ітераційним. Розробивши перший наближений варіант моделі, можна уточнювати її, опитуючи експертів предметної області. При цьому документацією, де фіксуються результати бесід, є сама модель «сутність – зв'язок» [7].

Сутність – будь-який конкретний чи абстрактний об'єкт до розглянутої предметної області. Сутності – це базові типи інформації, які зберігаються в БД (в реляційної БД кожної сутності призначається таблиця).

Атрибут – це властивість сутності в предметній області. Його найменування повинне бути унікальним для конкретного типу сутності.

Зв'язок – взаємозв'язок між сутностями в предметній області. Зв'язки представляють собою з'єднання між частинами БД (в реляційної БД – це з'єднання між записами таблиць).

При моделюванні прийнято виражати (іменувати) сутність іменником або іменником з прикметником, що характеризує його, а зв'язок – дієсловом, що поєднує два чи більше іменників.

Сутності та зв'язки можуть мати свої атрибути. Наприклад, сутність громадянин має атрибут номер паспорту, а зв'язок між сутностями гравець та аккаунт володіє атрибутом останній вхід.

Кожна сутність (якщо це не слабка сутність) повинна мати мінімальний набір унікальних атрибутів, що зветься первинним ключем [7].

Концептуальна модель БД представлена на рисунку 2.2.

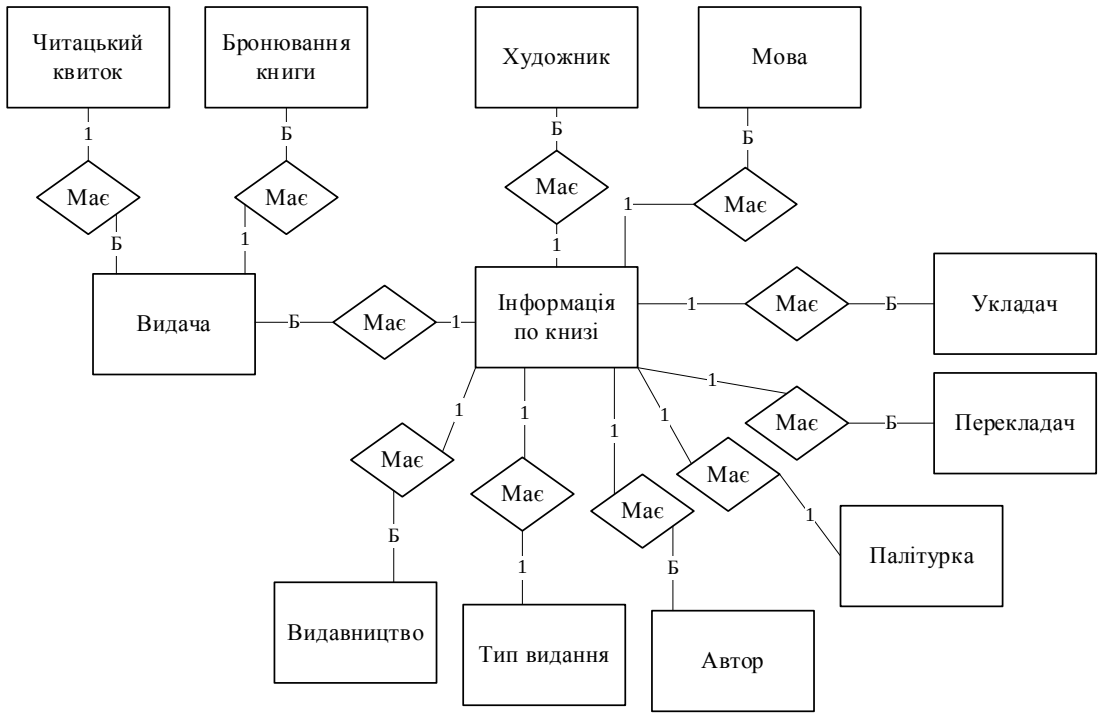

Рисунок 2.2 – Концептуальна модель БД програмного забезпечення для автоматизації роботи бібліотеки-філіалу

Атрибути кожної з сутностей представлені у таблиці 2.11.

Таблиця 2.11 – Атрибути сутностей

Сутність	Атрибут
Читацький квиток	Код квитка, ПІБ, адреса, телефон.
Інформація по книзі	Код книги, заголовок, тип видання, номер тома, бібліотечний шифр, код видавництва, код автора, код мови, код художника, код перекладача, код палітурки, об’єм книги, дата отримання, ціна.
Мова	Код мови, мова.
Автор	Код автора, ПІБ.
Укладач	Код укладача, ПІБ.
Художник	Код художника, ПІБ.
Перекладач	Код перекладача, ПІБ.
Палітурка	Код палітурки, тип, палітурки.
Видача	Код квитка, код книги, дата видачі, термін, дата повернення
Тип видання	Код типу, найменування.
Видавництво	Код видавництва, найменування, країна, місто.
Бронювання книг	Код бронювання, код книги, код квитка, код видачі, дата замовлення бронювання.

В результаті виділено ключові сутності з їх атрибутами, що присутні в концептуальній моделі бази даних і відносини, які можуть встановлюватися між цими сутностями.

2.1.5 Розробка сценаріїв варіантів використання

Кожен об'єкт системи, що володіє певною поведінкою, може знаходиться в певних станах, переходити зі стану в стан, здійснюючи певні дії в процесі реалізації сценарію поведінки об'єкта. Поведінка більшості об'єктів реальних систем можна представити з точки зору теорії кінцевих автоматів, тобто поведінка об'єкта відбивається в його станах, і даний тип діаграм дозволяє відобразити це графічно. Для цього використовується два види діаграм: Statechart diagram (діаграма станів) і Activity diagram (діаграма активності) [8].

Сценарії варіантів використання представлено на рисунках 2.3 – 2.5. Було представлено декілька варіантів використання «Авторизація», «Оформлення видачі книги», «Оформлення бронювання книги».

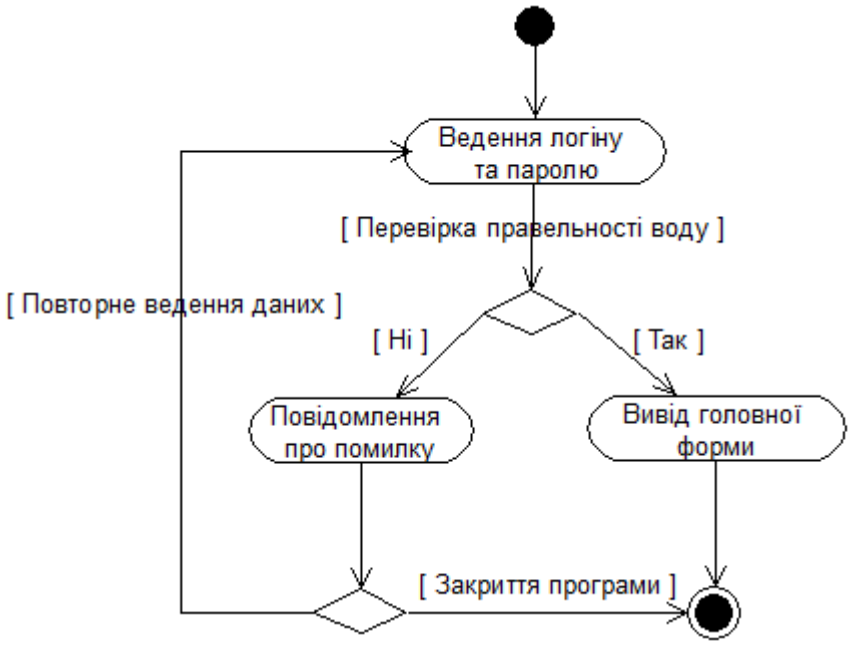

Рисунок 2.3 – Діаграма діяльності для прецеденту «Авторизація»

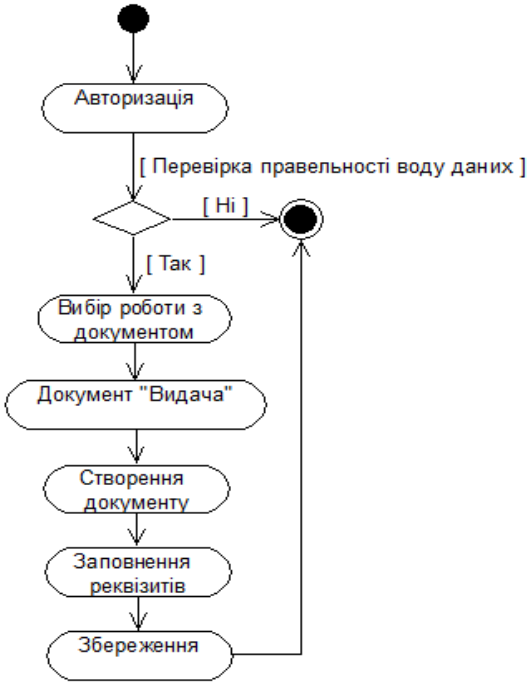

Рисунок 2.4 – Діаграма діяльності для прецеденту «Оформлення видачі книги»

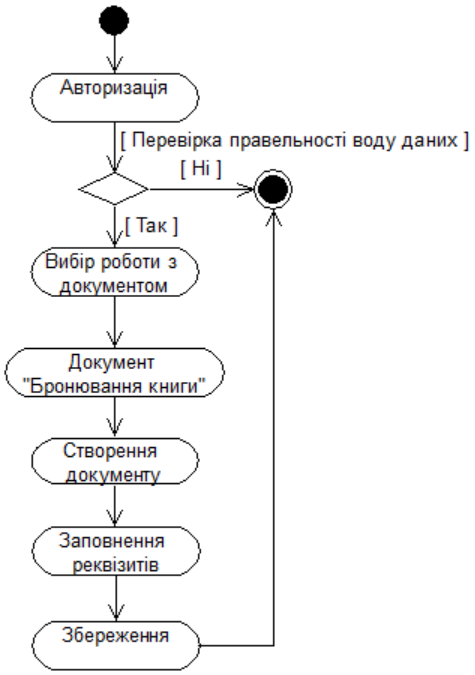

Рисунок 2.5 – Діаграма діяльності для прецеденту «Оформлення бронювання книги»

Дані діаграми діяльності для прецедентів добре відображають послідовність дій та події, що ініціюють дії або є кінцевим результатом.

2.1.6 Розробка прототипу користувацького інтерфейсу

Користувальницький інтерфейс являє собою сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії становлять діалоги. Під діалогом у цьому випадку розуміють регламентований обмін інформацією між людиною та комп'ютером, здійснюваний в реальному масштабі часу та спрямований на спільне рішення конкретної задачі: обмін інформацією та координацію дій. Кожний діалог складається з окремих процесів введення-виведення, які фізично забезпечують зв'язок користувача і комп'ютера [9].

На рисунках 2.6 – 2.8 представлено прототип користувацького інтерфейсу програмного забезпечення.

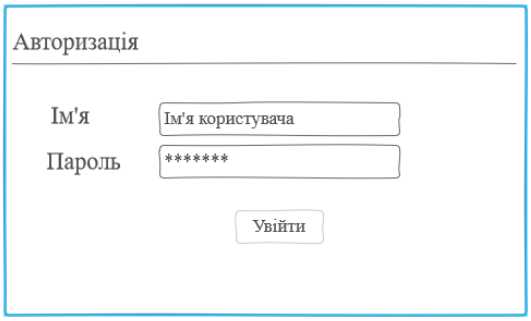

Рисунок 2.6 – Ескіз форми «Авторизація»

Рисунок 2.7 – Ескіз форми «Читацький квиток»

Рисунок 2.8 – Ескіз форми «Інформація по книзі»

Таким чином, розроблено графічний інтерфейс програмного забезпечення для автоматизації роботи бібліотеки-філіалу.

2.2 Технічний проект програмного забезпечення

2.2.1 Побудова логічної моделі бази даних

Перед тим як створювати таблиці БД, форми та інші об'єкти, потрібно задати структуру бази даних. Добра структура бази даних є основою для створення адекватної вимогам, ефективної бази даних. Сам процес проектування бази даних являє собою складний процес.

Логічна модель є основою БД, вона повинна відображати взаємозв’язки між реляційними таблицями.

Логічна модель БД представлена на рисунку 2.9.

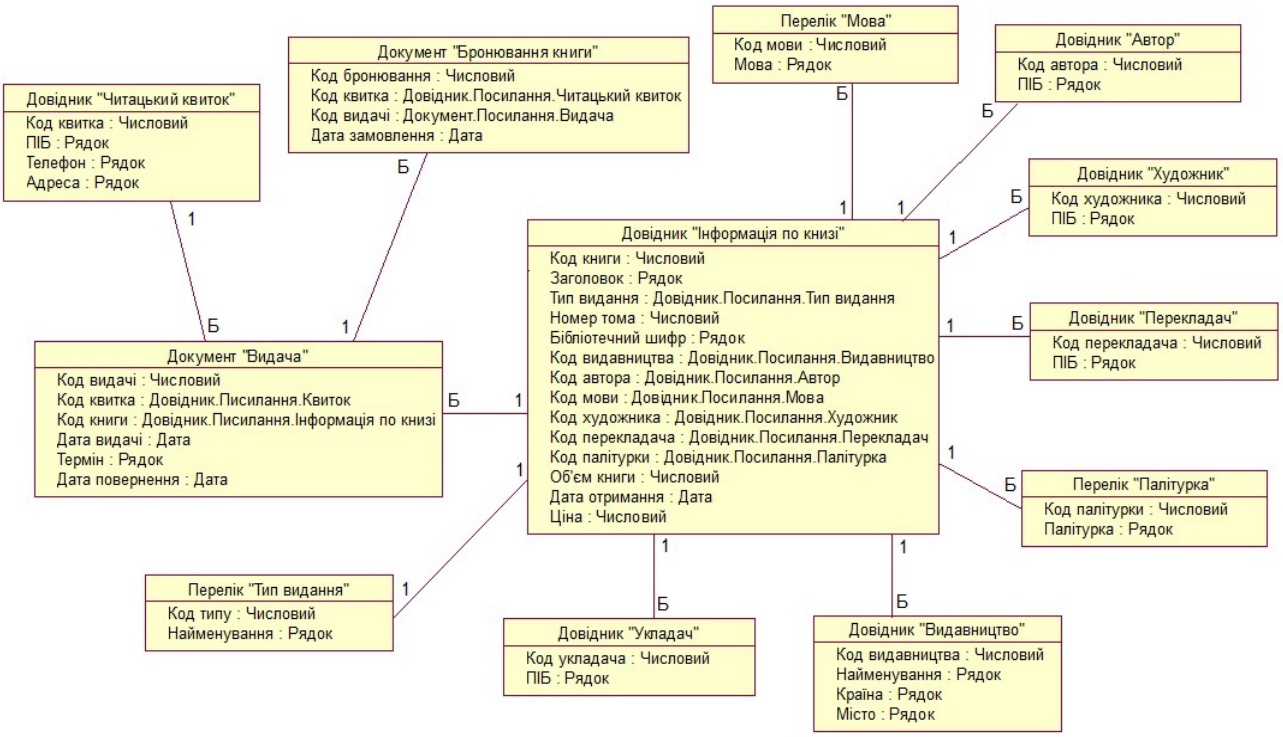

Рисунок 2.9 – Логічна модель БД програмного забезпечення для автоматизації роботи бібліотеки-філіалу

На логічній моделі відображено таблиці, поля таблиць, типи полів та зв’язки між таблицями.

2.2.2 Побудова статичної моделі програмного забезпечення

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити пакети та вкладені пакети [10]. Діаграма класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу представлена на рисунку 2.10.

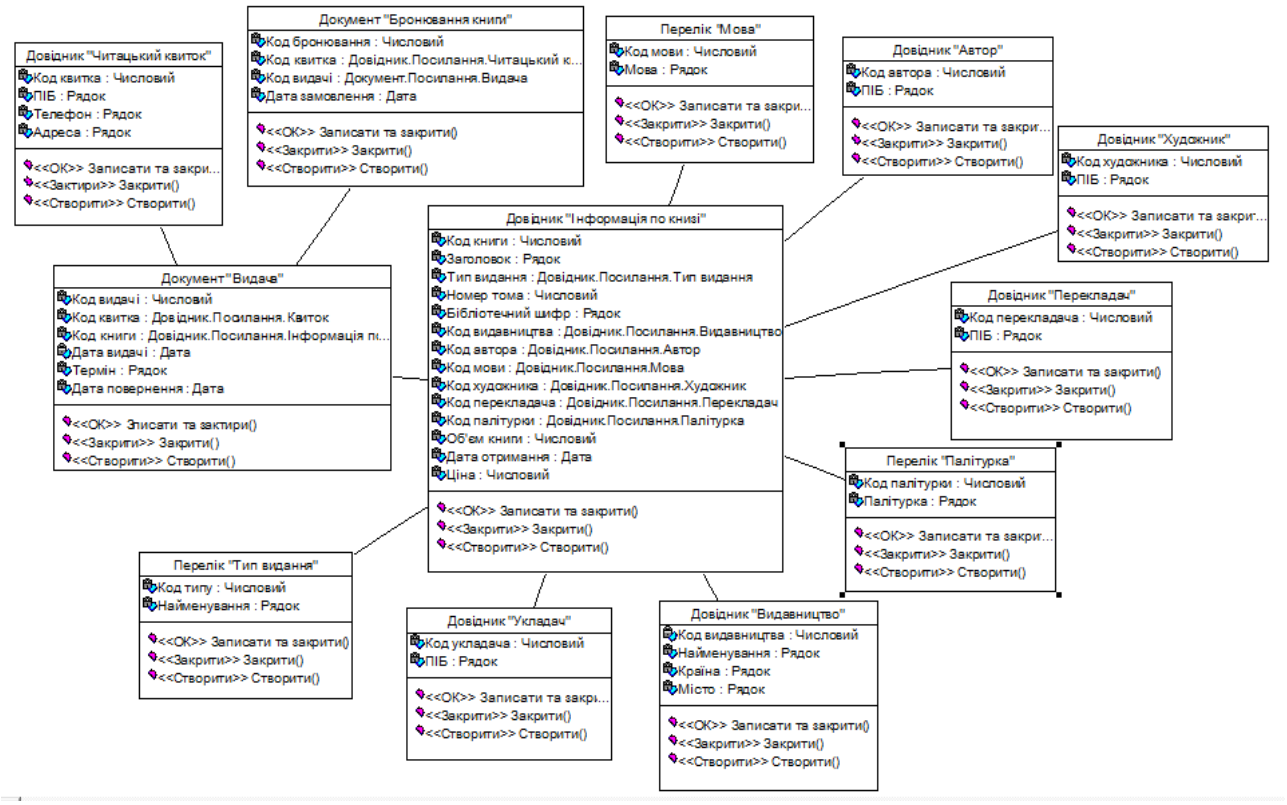

Рисунок 2.10 – Діаграма класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу

На діаграмі класів відображені наступні елементи: довідник «Читацький квиток», документ «Видача», довідник «Інформація по книзі», довідник «Перекладач», довідник «Укладач», довідник «Художник», документ «Бронювання книги», документ «Видача», перелік «Мова», перелік «Палітурка», перелік «Тип видання». Вказано поля таблиць та типи полів.

2.2.3 Розробка специфікації класів програмного забезпечення

Специфікація класу для об'єкта – це визначення його властивостей (які характеризують стан об'єкта) і методів (які є способом реалізації його поведінки).

Специфікація класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу представлена в таблиці 2.12.

Таблиця 2.12 – Специфікація класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу

Читацький квиток	
1	2
Відповідальність	Даний клас відповідає за інформацію по читачам.
Атрибут	Код квитка, ПІБ, телефон, адреса.
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Інформація по книзі	
Відповідальність	Даний клас відповідає за інформацію по книзі
Атрибут	Код книги, заголовок, тип видання, номер тома бібліотечний шифр, код видавництва, код випуску, код автора, код мови, код художника, код перекладача, код палітурки, об’єм книги, дата отримання, ціна.
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Мова	
Відповідальність	Даний клас відповідає за інформацію по мові, на якій мові написана книжка
Атрибут	Код мови, мова
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Автор	
Відповідальність	Даний клас відповідає за інформацію по авторам
Атрибут	Код автора ПІБ
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Укладач	
Відповідальність	Даний клас відповідає за інформацію по постачальникам укладачам
Атрибут	Код укладача ПІБ
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити

Продовження табл. 2.12

1	2
Художник	
Відповідальність	Даний клас відповідає за інформацію по художникам
Атрибут	Код художника ПІБ
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Перекладач	
Відповідальність	Даний клас відповідає за інформацію по перекладачам
Атрибут	Код перекладача ПІБ
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Палітурка	
Відповідальність	Даний клас відповідає за інформацію палітурки книжки
Атрибут	Код палітурки Код видання Ціна Дата покупки
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Видача	
Відповідальність	Даний клас відповідає за інформацію по видачі книжок читачеві
Атрибут	Код квитка, код книги, код палітурки, дата видачі, термін дата, повернення.
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Тип видання	
Відповідальність	Даний клас відповідає за інформацію вид видання книжки
Атрибут	Код виду Найменування
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити
Видавництво	
Відповідальність	Даний клас відповідає за інформацію по видавництву
Атрибут	Код видавництва, найменування, країна, місто
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити

Закінчення табл. 2.12

1	2
Бронювання книг	
Відповідальність	Даний клас відповідає за бронювання книги
Атрибут	Код бронювання, код книги, код квитка, код видачі, дата замовлення бронювання
Операції зі специфікаціями нетривіальних операцій	ОК(): записати та закрити Створити():створити Закрити():закрити

Таким чином, була розроблена специфікація класів програмного забезпечення для автоматизації роботи бібліотеки-філіалу.

2.2.4 Побудова динамічної моделі програмного забезпечення

Динамічна модель програми може бути представлена:

- діаграмами діяльності;
- діаграмами послідовності;
- діаграмами взаємодії;
- діаграмам станів.

Діаграма послідовності (sequence diagram) – діаграма, на якій показані взаємодії об'єктів, упорядковані за часом їхнього прояву [11].

Динамічна модель програми представлена у вигляді діаграми послідовності.

Діаграми послідовності по деяким варіантам використання представлені на рисунках 2.11 – 2.13.

У варіанті використання «Авторизація» користувач спочатку вибирає ім’я під яким він матиме змогу увійти до системи, вводить пароль. Діаграма послідовності для варіанту використання «Авторизація» представлена на рисунку 2.11.

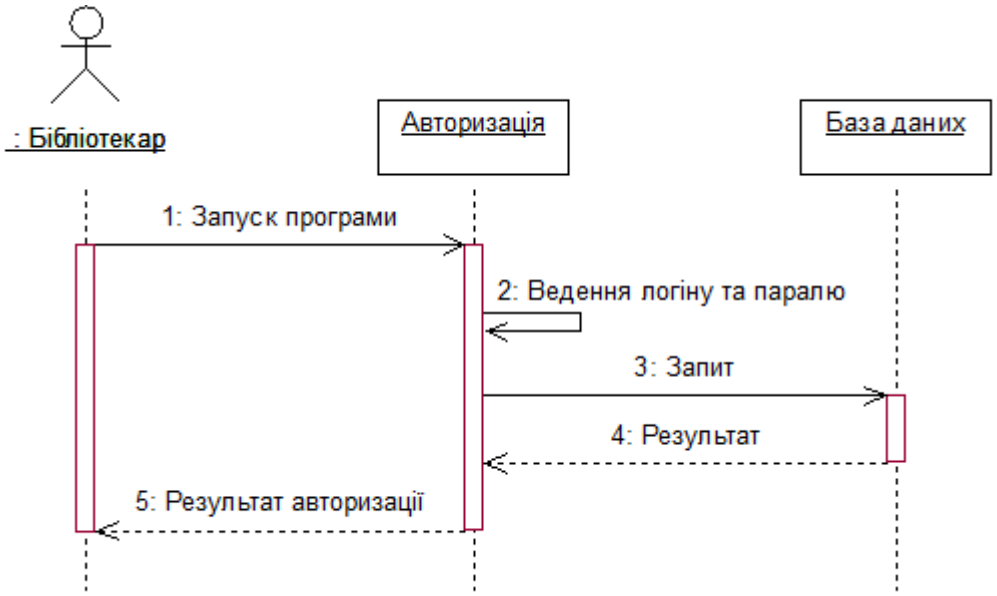

Рисунок 2.11 – Діаграма послідовності для варіанту використання «Авторизація»

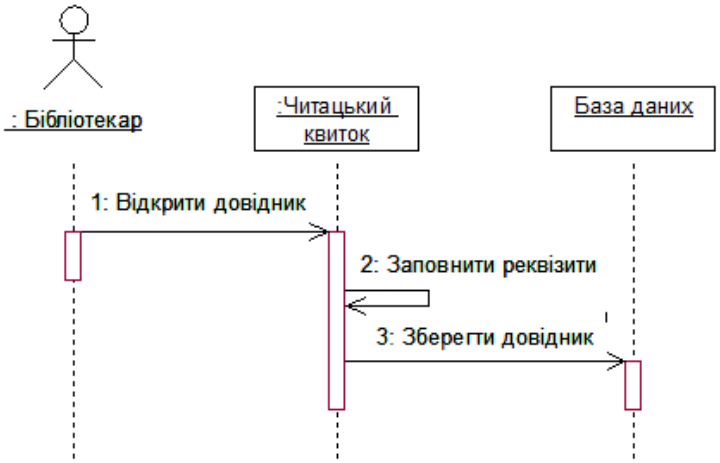

Рисунок 2.12 – Діаграма послідовності для варіанту використання заповнення довідника «Читацький квиток»

Робота з довідниками, а саме процес заповнення довідника представлений на прикладі довідника «Читацький квиток», що показано на рисунку 2.12. Заповнення починається з обрання довідника, зміни та заповнення реквізитів. Після заповнення перевіряється правильність введених даних і, у разі успішного введення, інформація зберігається у довіднику. Робота з іншими довідниками

виконується аналогічно.

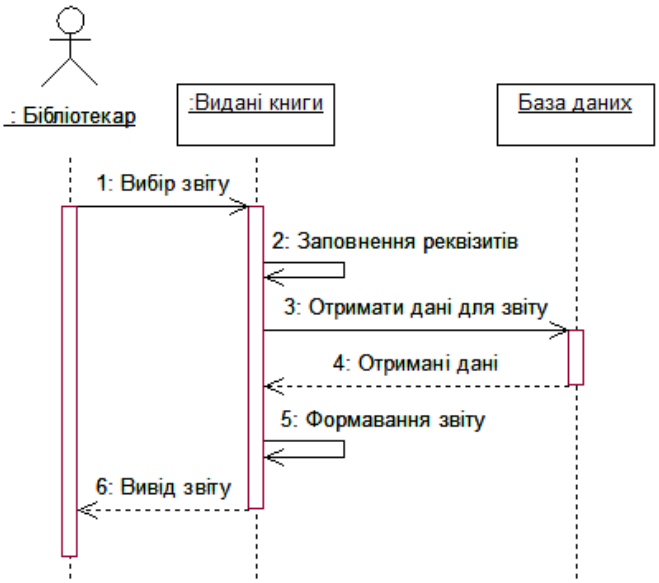

Рисунок 2.13 – Діаграма послідовності для варіанту використання «Видані книги»

Процес формування звіту представлений на прикладі звіту «Видані книги», що показано на рисунку 2.13. Формування починається з обрання звіту та заповнення реквізитів. Після заповнення перевіряється правильність введених даних і, у разі успішного введення, формується звіт.

2.3 Робочий проект програмного забезпечення

2.3.1 Обґрунтування вибору мови програмування та СКБД

Серед мов програмування для реалізації програмного забезпечення для автоматизації роботи бібліотеки-філіалу було обрано мову Java, тому що вона є зручною для побудови мережевих додатків та забезпечує незалежність від платформи.

Java це об'єктно-орієнтована мова програмування, випущена у 1995 році компанією «Sun Microsystems» як основний компонент платформи Java. З 2009

року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Мова Java також призначена для створення програм, які можуть працювати в розподіленому середовищі Internet на базі протоколів TCP/IP. Крім того, в мові Java наявний засіб передачі повідомлень в межах внутрішнього адресного простору. Це дозволяє забезпечити віддалене виконання процедур. Ці інтерфейси включені у пакет RMI (remote metod invocation). Цей засіб привносить високий рівень абстракції в програмування для середовища клієнт/сервер [12].

Для розробки програмного забезпечення для автоматизації роботи бібліотеки-філіалу оберемо бібліотеку серед двох бібліотек Java, що найбільш широко використовуються для розробки програм з графічним інтерфейсом: JavaFX та Swing. Оскільки JavaFX була створена на основібібліотеки Swing, вона значно розширила функціонал бібліотеки Swing та внесла нові способи створення елементів графічного інтерфейсу користувача. Отже, доцільніше обрати для розробки ПЗ бібліотеку JavaFX.

MySQL – це система управління реляційними базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш у швидкості й гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. SQL як частина системи MySQL можна охарактеризувати як мова структурованих запитів, що використовується для доступу до баз даних.

MySQL – це ПЗ з відкритим кодом. Застосовувати його і модифікувати може будь-хто. Таке ПЗ можна отримувати за допомогою Internet і використовувати безкоштовно. При цьому кожен користувач може вивчити вихідний код і змінити його у відповідності зі своїми потребами [13].

2.3.2 Реалізація та тестування основних класів програмного забезпечення

Для реалізації програмного забезпечення для автоматизації роботи бібліотеки-філіалу було обрано мову програмування Java з бібліотекою JavaFX та СУБД MySQL.

Для тестування програмного забезпечення є такі основні групи методів:

- методи тестування по формальним специфікаціям (методи чорного ящика);
- метод структурного тестування (методи білого ящика).

До групи методів чорного ящика відносяться наступні методи: розбиття на класи еквівалентності та граничні умови та інші. До методів білого ящика відносяться наступні методи: покриття операторів, покриття переходів та інші.

Оскільки методи структурного тестування вимагають більших затрат ресурсів, то в кваліфікаційній роботі виконувалось тестування по специфікаціям, а саме розбиття на класи еквівалентності.

Програму розглядали як чорний ящик. Тестування зводилися до послідовного вводу тестових наборів даних й аналізу отриманих результатів.

В таблиці 2.13 представлені класи еквівалентності та граничні значення вхідних даних класу «Читацький квиток»:

– прізвище;
– ім’я;
– по-батькові;
– телефон;
– адреса.

Таблиця 2.13 – Класи еквівалентності вхідних даних для класу формування зведеної відомості

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Прізвище	Рядок	Графічні зображення (в тому числі і смайлики)
Ім’я	Рядок	Графічні зображення (в тому числі і смайлики)
По-батькові	Рядок	Графічні зображення (в тому числі і смайлики)
Телефон	Рядок	Графічні зображення (в тому числі і смайлики)
Адреса	Рядок	Графічні зображення (в тому числі і смайлики)

Почнемо з тестування для правильних класів еквівалентності класу «Читацький квиток», наведених в табл. 2.14.

Таблиця 2.14 – Тести з правильних класів еквівалентності для класу «Читацький квиток»

Номер тесту	Вхідні умови	Правильні класи еквівалентності	Вихідні дані
1	Прізвище	В поле «Прізвище» введено прізвище читача – Іванов.	Вивід на екран прізвище читача. Результат вірний.
2	Ім’я	В поле «Ім’я» введено ім’я читача – Іван.	Вивід на екран ім’я читача. Результат вірний.
3	По-батькові	В поле «По-батькові» введено по-батькові читача – Іванович.	Вивід на екран по-батькові читача. Результат вірний.
4	Телефон	В поле «Телефон» введено номер телефону читача – 063 322 223 3.	Вивід на екран номеру телефону читача. Результат вірний.
5	Адреса	В поле «Адреса» введено адресу читача – Центральний 9, кв. 9.	Вивід на екран адреси читача. Результат вірний.

Отже, за результатами тестування правильних класів еквівалентності класу «Читацький квиток» програмне забезпечення для автоматизації роботи бібліотеки-філіалу працює правильно і без збоїв.

Наступним кроком буде перевірка неправильних класів еквівалентності класу «Читацький квиток», », наведених в табл. 2.15.

Таблиця 2.15 – Тести з неправильних класів еквівалентності класу «Читацький квиток»

Номер тесту	Вхідні умови	Неправильні класи еквівалентності	Вихідні дані
1	Прізвище	В поле «Прізвище» введемо будь-який символ ASCII	Програма виводить повідомлення про помилку. Результат підтверджено.
2	Ім’я	В поле «Ім’я» ведемо графічний елемент – смайлик (скопіюємо з інтернету)	Після ведення смайлика виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено
3	По-батькові	В поле «По-батькові» ведемо будь-які цифри.	Після ведення цифр виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено
4	Телефон	В поле «Телефон» графічний елемент – смайлик (скопіюємо з інтернету)	Після ведення смайлика виводиться повідомлення про помилку. Програма далі працює правильно. Результат підтверджено
5	Адреса	В поле «Адреса» ведемо будь-який символ ASCII.	Програма виводить повідомлення про помилку. Результат підтверджено

Отже, за результатами тестування неправильних класів еквівалентності класу «Читацький квиток» програмне забезпечення для автоматизації роботи бібліотеки-філіалу працює правильно, не допускаючи до введення будь-яких інших елементів, окрім стандартних символів вводу. Тестування інших класів виконується аналогічно.

2.3.3 Випробування програмного забезпечення

Під час випробувань було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу.

Випробування ПЗ для автоматизації роботи бібліотеки-філіалу проводилося на цільовому обладнанні в тій конфігурації, яка заявлена в Технічному завданні (див. Додаток А).

Всі виявлені недоліки ПЗ для автоматизації роботи бібліотеки-філіалу були зафіксовані в протоколах і усунуті розробником до моменту впровадження ПЗ в дію.

Випробування було проведено за стратегією «чорного ящика». За рахунок дуже великої кількості функцій, які потрібно було випробовувати, представлено тільки декілька результатів випробувань функцій ПЗ:

1) Функція «Оформлення та редагування читацького квитка»

На головному вікні обрали довідник «Читацький квиток» та натиснули кнопку «Додати». З’явилось відповідне вікно, у якому обираємо читача, вносимо необхідні реквізити та натискаємо кнопку «Закрити та зберегти». У результаті отримали сформований читацьких квиток.

2) Функція «Оформлення бронювання книги»

На головному вікні обрали таблицю «Бронювання», та натиснули кнопку «Додати». З’явилось відповідне вікно, у якому обираємо читача, книгу та натискаємо кнопку «Закрити та зберегти». У результаті отримали сформоване бронювання на книгу.

3) Функція «Оформлення видачі книги»

На головному вікні обрали таблицю «Видача», та натиснули кнопку «Додати». З’явилось відповідне вікно у якому обираємо читача, книгу та натискаємо кнопку «Закрити та зберегти». У результаті отримали сформовану видачу на книгу.

4) Функція «Формування звіту «Видані книги»»

В головному меню натиснули на кнопку «Звіти». З’явилося вікно з можливими звітами. Вибрали звіт «Видані книги» та виконали встановлені початкові дії (вибрали відповідні параметри звіту). В результаті сформувався звіт «Видані книги».

Повністю програма та методика випробувань програмного забезпечення для автоматизації роботи бібліотеки-філіалу наведено у Додатку Д.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ БІБЛІОТЕКИ-ФІЛІАЛУ

В представленій кваліфікаційній роботі була розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено програмне забезпечення для автоматизації роботи бібліотеки-філіалу.

Програмне забезпечення дозволяє автоматично вести облік книг, облік читачів, формувати та роздруковувати заявку на бронювання та видачу книги, формувати необхідні звіті.

Проведено огляд літературних джерел, нормативних документів, Internet-джерел для аналізу роботи та методів можливого автоматизування роботу бібліотекара.

Проведено пошук та аналіз уже існуючих програмних рішень для вирішення проблеми, обґрунтована необхідність та виконана постановка задачі до розробки програмного забезпечення для автоматизації роботи бібліотеки-філіалу.

В рамках ескізного проекту було виконано:

- побудовано діаграму варіантів використання;
- розроблено специфікації варіантів використання;
- побудовано концептуальну модель БД;
- розроблено сценарії варіантів використання;
- розроблено прототип користувацького інтерфейсу.

В рамках технічного проекту було виконано:

- побудовано логічну модель БД;
- побудовано статичну модель програмного забезпечення;
- розроблено специфікації класів програмного забезпечення;
- побудовано динамічну модель програмного забезпечення.

В рамках робочого проекту було виконано:

- обґрунтовано вибір мови програмування та СКБД;

- виконано реалізацію та тестування основних класів програмного забезпечення;
- виконано випробування програмного забезпечення.

Програмне забезпечення було розроблено з використанням мови програмування Java з бібліотекою JavaFX та СУБД MySQL. Програмне забезпечення використовує незначну кількість ресурсів на комп’ютері.

Результатом роботи являється програмне забезпечення для автоматизації роботи бібліотеки-філіалу та можливість ведення обліку книг та читачів.

Було виконано тестування та випробування ПЗ, яке показало повну працездатність та відповідність вимогам, які представлені у технічному завданні, яке наведено у додатку А.

Текст ПЗ представлений у додатку Б – «Текст програми».

Для якісної експлуатації програмного забезпечення також було розроблено опис програми, який представлено у Додатку В та керівництво користувача, яке представлено у додатку Г – «Інструкція користувача».

Програма та методика випробування програмного забезпечення представлена у додатку Д.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров’я і працездатності людини у процесі трудової діяльності [14].

Актуальність даної теми визначається тим, що на сьогодні в Україні складається дуже небезпечна ситуація в питаннях охорони праці, життя та здоров’я людини. Велика кількість законів залишається на папері і не працюють. Значна кількість працівників не мають офіційного оформлення, що призводить до втрати можливості законного захисту їх прав.

Охорона праці містить три основних складових частини: правові норми трудового законодавства, виробничу санітарію, гігієну та техніку безпеки, а також протипожежний захист і електробезпеку [15].

Мета охорони праці – забезпечення безпечних, нешкідливих і сприятливих умов праці через вирішення багатьох складних завдань, основними з яких є:

– проектування підприємств, технологічних процесів і конструювання обладнання з обов'язковим виконанням вимог охорони праці;

– знаходження оптимальних співвідношень між різними факторами виробничого середовища, що дозволяє забезпечити мінімум несприятливого впливу їх на здоров'я працівників;

– встановлення, законодавче оформлення визначених норм кожного з несприятливих або небезпечних факторів, систематичний король за їх застосуванням;

– розробка конкретних заходів щодо покращення умов праці та забезпечення її безпеки на основі застосування у виробництві новітніх досягнень науки і техніки;

– застосування раціональних засобів захисту працівників від впливу несприятливих факторів виробничого середовища, а також втілення організаційних заходів, які нейтралізують або послаблюють ступінь їх впливу на організм людини;

– розробка та застосування методів і засобів оцінки ефективності заходів з охорони праці, що плануються і здійснюються.

4.2 Аналіз шкідливих і небезпечних факторів при роботі на персональному комп'ютері

При роботі на персональному комп'ютері співробітники стикаються з дією таких небезпечних фізичних і психологічних чинників як - підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, відсутність або нестача природного світла, електричний струм, статична електрика, розумове перенапруження, перенапруження очей, монотонність праці, емоційні перевантаження. Розглянемо чинники докладніше.

При роботі монітора виникають два типи випромінювань: електростатичне і електромагнітне. Навколо електростатичного зарядженого монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними котушками, які знаходяться біля цокольній частині ЕПТ. Результати медичних досліджень показують, що пил, який електризується, може викликати опік шкіри, а електромагнітне випромінювання може призводити до зниження загальної продуктивності оператора.

Крім цього важливим моментом є показники частоти вертикальної і горизонтальної розгортки ЕПТ. Прийняті стандарти на монітори не дозволяють використовувати ЕПТ і відеоадаптери з частотою вертикальної розгортки менше 75 Гц.

Мікрокліматичні параметри впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки. Для комфортної

роботи приймається температура 230С і 180С в теплу і холодну пору року відповідно, при відносній вологості 55% [16].

Також існують інші небезпечні шкідливі фактори, такі як:

Небезпека ураження електричним струмом. Все обладнання ЕОМ, як будь-яка електроустановка, представляють для людини велику потенційну небезпеку, оскільки в процесі експлуатації або проведення профілактичних робіт людина може торкнутися струмовідні частини.

Експериментальні дослідження показали, що людина відчуває подразнюючу дію змінного струму промислової частоти силою 0,6-1,5 мА і постійного струму 5-7 мА. Ці струми не становлять серйозної небезпеки для діяльності організму людини, оскільки при такій силі струму можливо самостійне звільнення осіб від контакту з струмоведучими частинами. Для змінного струму промислової частоти сила не відпускає струму знаходиться в межах 6-20 мА. Постійний струм не викликає не відпускає ефекту, але призводить до сильних больових відчуттів, сила такого струму 15-80 мА і більше [17].

Недостатня освітленість робочого місця. Правильно спроектоване і виконане висвітлення в приміщенні, де працюють оператори, забезпечує високу працездатність, надає позитивний психологічний вплив на тих, хто працює, сприяє підвищенню продуктивності роботи.

Рекомендована освітленість для роботи для роботи з екраном дисплея становить 400-750 лк. Рекомендовані яскравості в полі зору операторів повинні лежати в межах 1:5-1:10.

Можливість виникнення пожежі. Приміщення, в якому розміщені ЕОМ за категоріями пожежної небезпеки відноситься до категорії "В". Зазвичай в ньому знаходиться велика кількість можливих джерел спалаху: кабельні лінії, що використовуються для живлення ЕОМ від мережі змінного струму напругою 220В, електронно-променева трубка монітора, яка є вибухонебезпечною без додаткового захисту, устаткування, меблі з горючих матеріалів, папір.

4.3 Розробка заходів по зменшенню дії шкідливих факторів

При розробці заходів, які забезпечують охорону праці та безпеки життедіяльності, враховують всі шкідливі і небезпечні фактори, зменшуючи кожен з них до допустимих значень. Тим не менше, тільки дотримання правил і норм технічної безпеки та охорони праці забезпечить гарну робочу обстановку і запобіжить нещасні випадки на робочому місці.

При придбанні моніторів, слід вибирати монітори узгоджених з рекомендаціями щодо зменшення електричних і магнітних полів, а також що підтримують вертикальну частоту розгортки 75 Гц і більше.

Вимоги до мікроклімату в робочому приміщенні.

Для підтримки необхідних мікрокліматичних параметрів необхідно встановлювати кондиціоноване обладнання. Для зменшення кількості пилу встановлюють періодичність вологого прибирання в приміщенні.

Вимоги до електробезпеки.

Велике значення для запобігання електротравматизму має правильна організація обслуговування діючих електроустановок, проведення ремонтних та профілактичних робіт, що здійснюється за допомогою наступних заходів: допуск до роботи, нагляд під час роботи, виробництво відключень під час ремонту, вивішування попереджувальних плакатів і знаків безпеки, перевірка відсутності напруги, накладення заземлення.

Для забезпечення електробезпеки обслуговуючого персоналу перед-бачених заземлювальні пристрої, до яких підключені всі металеві частини робочого обладнання. У зв'язку з необхідністю розміщення проводів електроживлення обладнання без зайвих витрат на переобладнання приміщення, доцільно влаштовувати підлога з підпіллям.

Для зниження величин виникаючих статичних зарядів в обчислювальних центрах застосовують покриття технологічне з одношарового антистатичного лінолеуму марки ASN. Можна застосовувати загальне і місцеве зволоження повітря. Одним з нових методів зменшення статичної напруги в приміщенні є

нейтралізація електрики іонізованим газом.

Вимоги до освітленості.

В дисплейних залах, звичайно, застосовують одностороннє природне бічне освітлення. З метою уникнення прямого сонячного світла використовують приміщення з вікнами з північної, північно-східній або північно-західною орієнтацією. Монітори розташовують подалі від вікон і так, щоб вікна були збоку. Якщо екран монітора розташований до вікна, необхідні спеціальні екранують пристрої.

Для штучного освітлення приміщень слід використовувати люмінес-процентні лампи, оскільки в них висока світлова віддача (до 75 лм/Вт і більше), тривалий термін служби (до 10000 годин), мала яскравість поверхні, яка світиться, близький до природного спектральний склад випромінюваного світла , який забезпечує хороше перенесення кольорів. Найбільш прийнятними для дисплейних приміщень є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (білого-тепло-білого світла) потужністю 40, 80 Вт.

Для виключення засвічена екранів дисплеїв прямими світловими потоками світильники загального освітлення розташовують збоку від робочого місця, паралельно лінії зору оператора і стіні з вікнами. Таке розміщення світильників дозволяє проводити їх послідовне включення залежно від величини природної освітленості і виключає роздратування очей смугами світла і тіні, що чергуються, виникають при поперечному розташуванні світильників.

Вимоги до пожежної безпеки.

Для запобігання виникнення пожежі необхідно передбачити заходи пожежної профілактики: дотримання протипожежних вимог під час проектування і експлуатації систем вентиляції; дотримання умов пожежної безпеки електроустановок згідно вимог; наявність засобів сповіщення:

– пожежні сповіщувачі (ЛІП-1, ІП-105 2/1 і т.д.);

– установки пожежогасінні (АУП);

– інструкції по заходам протипожежної безпеки, план евакуації людей і технічних засобів.

Для поліпшення умов пожежної безпеки в приміщенні має бути встановлений підлога з негорючих матеріалів, технологічно знімний; папір зберігається в металевій шафі; в наявності два вогнегасники типу OU-5, а також два димових датчика.

Загальні вимоги з техніки безпеки

Кожен користувач ЕОМ зобов'язаний:

- дотримуватися правил внутрішнього трудового розпорядку;
- не вносити на територію підприємства і не розпивати спиртні напої, палити лише у відведеному для цієї мети місці.

На території підприємства необхідно виконувати наступні вимоги:

Рухатися тільки по тротуарах, у разі пересування по проїжджій частині дороги потрібно йти по лівій стороні назустріч рухомого транспорту, виконуючи заходи обережності.

При вимушеній зупинці на середині дороги не кидатися в сторони, а стояти на одному місці, щоб дати можливість водієві об'їхати Вас з тієї чи іншої сторони.

Обходити на безпечній відстані місця, де ведуться висотні роботи.

До основних небезпек, які можуть призвести до травм при роботі оператора ЕОМ, відносять небезпеку ураження електричним струмом та іонізуючим випромінюванням.

Необхідно стежити за тим, щоб підлога в машинному залі була рівна і неслизький, все люки, кабелю, проводу повинні бути закриті або захищені.

Суворо дотримуватися правил пожежної безпеки на робочому місці. У разі загоряння відключити електроживлення і зателефонувати в пожежну частину, пояснивши, що і де горить.

Вимоги до безпеки перед початком роботи.

Прийнявши ЕОМ від змінника, необхідно перевірити, чи добре прибране робоче місце, ознайомитися з неполадками, які були в попередній зміні, в роботі ЕОМ і з прийнятими заходами по їх усуненню. Також перед початком роботи необхідно:

- отримати завдання у керівника роботи, а також інструктаж з техніки безпеки;
- підготувати необхідні інструменти, прилади. Перевірити їх справність, якщо необхідно, отримати захисні пристосування, запчастини, радіодеталі, матеріали;
- перевірити надійність кабелів в місцях їх підключення до джерел живлення;
- перевірити відсутність замикання між земляними ланцюгами та ланцюгами живлення напруги;
- перевірити наявність, справність і відповідність по струму запобіжників в блоках ремонтованої ЕОМ або пристроїв;

Вимоги до безпеки під час роботи:

- не дозволяється залишати ЕОМ, що знаходиться під напругою, без спостереження;
- не дозволяється замінювати знімні елементи і проводити переміщення і перемонтування на включеній ЕОМ;
- не дозволяється користуватися несправною апаратурою та інструментами;
- не дозволяється поєднувати і роз'єднувати розетки і вилки роз'ємів, які знаходяться під напругою;
- не дозволяється знімати кришки та щити, які закривають доступ до струмоведучих частин, при включенні обладнання;
- не дозволяється змінювати запобіжники під напругою;
- при заміні запобіжників в блоках і пристроях, суворо дотримуватись керівництвом маркування по струму;
- не дозволяється користуватися паяльником з незаземленим корпусом;
- при ремонті знімних блоків системи електроживлення їх корпус необхідно заземлити;
- всі операції, пов'язані з установкою переносних приладів і вимірами, повинні виключати дотик струмоведучих частин;

- при проведенні робіт необхідно вимагати присутності другої людини, допущеного до роботи з електричними установкам і з напругою до 1000 вольт.

Вимоги до безпеки по закінченню робіт:

- вимкніть електроживлення від ЕОМ і обладнання, яке забезпечує роботу ЕОМ;
- приведіть у порядок робоче місце;
- повідомте керівника робіт про закінчення роботи та її результати;
- зробіть запис в «Журналі зауважень про роботу ЕОМ».

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка програмного забезпечення для автоматизації роботи бібліотеки-філіалу, яка передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов. Розроблене програмне забезпечення дозволить працівникам бібліотеки-філіалу спростити та пришвидшити свою роботу та полегшити ведення звітності.

Для досягнення поставленої мети було виконано:

– В першому розділі проаналізовано інформаційну систему бібліотеки-філіалу та визначено специфіку її роботи; розглянуто існуючі програмні рішення; здійснено постановку задачі на розробку програмного забезпечення для автоматизації роботи бібліотеки-філіалу.

– В другому розділі виконано ескізний, технічний та робочий проекти програмного забезпечення для автоматизації роботи бібліотеки-філіалу, розроблена необхідна програмна документація.

– В третьому розділі представлені результати розробки програмного забезпечення для автоматизації роботи бібліотеки-філіалу.

– В четвертому розділі розглянуті питання охорони праці.

Мета роботи досягнута, усі поставлені задачі виконано. Подальша робота може бути пов’язана з удосконаленням роботи програмного забезпечення для автоматизації роботи бібліотеки-філіалу та збільшенням його функціональності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Моя библиотека. URL: http://www.softportal.com/software-6658-moya-domashnyaya – biblioteka.html (дата звернення: 20.04.2022).

2. Calibre 3.35.0. URL: https://www.bestfree.ru/soft/office/cataloger-of-books.php (дата звернення: 20.04.2022).

3. MyLib 1.0. URL: https://www.bestfree.ru/soft/office/accounting-books.php (дата звернення: 20.04.2022).

4. Опис роботи бібліотеки. URL: http://conference.nbuv.gov.ua/report/view/id/177 (дата звернення: 20.04.2022).

5. Буч Г., Рамбо Д., Якобсон И. Краткая история UML, язык UML. Руководство пользователя. 2-е издание. 2006. 496 с.

6. Діаграма варіантів використання. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28086 (дата звернення: 25.04.2022).

7. Елементи моделі «сутність-зв'язок». URL: http://easy-code.com.ua/2012/09/elementi-modeli-sutnist-zvyazok-integraciya-dodatkiv-i-danix-bazi-danix-statti/ (дата звернення: 25.04.2022).

8. Діаграма станів. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=2808710 (дата звернення: 27.04.2022).

9. Інтерфейс користувача. URL: http://sven.ua/ua/service/glossary/ukr/11/ (дата звернення: 27.04.2022).

10. Діаграма класів. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 (дата звернення: 29.04.2022).

11. Діаграма послідовності. URL: http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?r=16538 (дата звернення: 29.04.2022).

12. Блох Д. Java. Эффективное программирование = Effective Java. М.: Лори, 2002. 224 с.

13. MySQL. Руководство администратора. М.: Издательский дом «Вильямс», 2005. 624 с.

14. Про охорону праці : Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 12.05.2022).

15. Поняття, мета і завдання охорони праці. URL: http://ncpn.net.ua/oxrana_truda.html (дата звернення: 12.05.2022).

16. ГОСТ 12.1.005-88. Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – Введ. 1989-01-01 – М.: Стандартинформ, 2000. – 4 с.

17. ГОСТ 12.1.038-82. Система стандартов безопасности труда. Электробезопасность. Предельно допустимые значения напряжений прикосновения и токов. – Введ. 1983-07-03 – М.: Стандартинформ, 01.11.1998. – 10 с.

ДОДАТОК А – Технічне завдання

Вступ

Назва розроблюваного проекту: «Програмне забезпечення для автоматизації роботи бібліотеки-філіалу». Галузь застосування – бібліотека-філіал.

1 Підстава для розробки

Розробка програмного забезпечення виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

2 Призначення розробки

2.1 Експлуатаційне призначення

Дане програмне забезпечення призначене для зберігання інформації про читачів та книги у базі даних і оперування цією інформацією для ведення обліку бібліотеки-філіалу.

2.2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація роботи працівників бібліотеки-філіалу, покращення та легкість ведення даних, зберігання та використання інформації, а також автоматизація робочого місця бібліотекаря і як наслідок, зменшення часу обслуговування читачів; забезпечення оперативного отримання інформації про наявність книг; формування читацького квитку; оформлення видачі книги.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

1) оформлення читацького квитка;
2) оформлення видачі книги;

3) оформлення бронювання книги;
4) перегляд інформації про книги та читацький квиток;
5) занесення та редагування даних про книгу;
6) видалення інформації про книги;
7) звіт «Видані книги»:
8) звіт «Інвентаризація»;
9) звіт «Планове повернення»;
10) звіт «Заборгованість».

3.1.2 Вимоги до організації вхідних та вихідних даних

Дані зберігаються в файлі БД з довільним доступом типу .myd.

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

3.2 Вимоги до надійності

Програмний продукт повинен функціонувати при безперебійній роботі комп’ютера. При виникненні збоїв в роботі комп’ютера, відновлення нормальної роботи повинне проводитися після перезавантаження програмного продукту.

Для забезпечення надійності інформації повинна використовуватися СКБД, що забезпечує цілісність транзакцій і несе відповідальність за цілісність інформації.

У разі введення користувачем некоректної інформації програмний продукт повинен повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навики в користуванні комп’ютером. Комп’ютер призначений для роботи в закритому

опалювальному приміщені.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

- CPU: Intel(R) Celeron(R) – 2.16 МГц;
- RAM: 2 GB;
- HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумістності

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Необхідна наявність встановленого Java Runtime Environment (JRE 1.8).

3.6 Вимоги до маркування та пакування

Програмне забезпечення буде упаковано в електронний архів і мати найменування Biblioteka.zip та містити:

- виконуваний файл Biblioteka.jar ;
- файл, який містить структуру БД Biblioteka.sql;
- файл з описом ReadMe.txt.

4 Вимоги до програмної документації

Програмна документація повинна містити:

- технічне завдання;
- текст програми;
- опис програми;
- інструкцію користувача;
- програму і методику випробувань ПЗ.

5 Стадії та етапи розробки

Стадії та етапи розробки представлені в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

Стадії розробки	Етапи робіт	Вміст робіт	Контрольні точки
Технічне завдання	Обумовлення необхідності розробки	Постановка задачі. Збір початкових матеріалів. Вибір і обумовлення критеріїв ефективності та якості розроблюваного ПЗ	02.03.2022 -15.03.2022
	Розробка та затвердження технічного завдання	Визначення вимог до ПЗ. Узгодження і затвердження технічного завдання.	16.03.2022 -31.03.2022
Ескізний проект	Розробка ескізного проекту	Попередня розробка структури вхідних та вихідних даних. Уточнення методів рішення задачі. Розробка загального опису алгоритму розв'язання задачі	01.04.2022 -15.04.2022
	Затвердження ескізного проекту	Розробка пояснювальної записки. Узгодження і затвердження ескізного проекту	16.04.2022 -27.04.2022
Технічний проект	Розробка технічного проекту	Уточнення структури вхідних та вихідних даних. Розробка алгоритму рішення задачі. Розробка структури ПЗ	28.04.2022 -03.05.2022
	Затвердження технічного проекту	Розробка пояснювальної записки. Узгодження і затвердження технічного проекту	04.05.2022 - 06.05.2022
Робочий проект	Розробка ПЗ	Програмування та налагодження програми	07.05.2022 -10.05.2022
	Розробка програмної документації	Розробка програмної документації відповідно до вимог ГОСТ 19.101-77	11.05.2022 -13.05.2022
	Випробування ПЗ	Розробка, узгодження та затвердження порядку і методики випробувань. Проведення попередніх випробувань. Коригування ПЗ і програмної документації за результатами випробувань.	14.05.2022 -15.05.2022

6 Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути

представлена в зазначені строки та узгоджена із замовником.

Прийом проводять відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б Текст програми

Main.java

```
package sample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;
public class Main extends Application {@Override
public void start(Stage primaryStage) throws Exception{
Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
primaryStage.setTitle("Главное");
primaryStage.getIcons().add(new Image("img/book.png"));
primaryStage.setScene(new Scene(root));
primaryStage.show();
}
public static void main(String[] args) {
launch(args);
}
}
```

Controller.java

```
package sample;

import java.net.URL;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeParseException;
import java.util.List;
import java.util.Optional;
import java.util.ResourceBundle;

import dao.ReaderDAO;
import dao.BookDAO;
import dao.BookingDAO;
import dao.TransferDAO;
import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import tables.Reader;
import tables.Book;
import tables.Booking;
import tables.Transfer;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Controller {

@FXML private ResourceBundle resources;
@FXML private URL location;
@FXML private TextField titleBookTextField;
```

```
@FXML private TextField performerBookTextField;
@FXML private TextField genreBookTextField;
@FXML private TextField ageRestrictionBookTextField;
@FXML private TextField cityBookTextField;
@FXML private TextField yyyyBookTextField;
@FXML private Button addBookRecordButton;
@FXML private Label addBookResultLabel;
@FXML private TextField ddTransferTextField;
@FXML private TextField mmTransferTextField;
@FXML private TextField yyyyTransferTextField;
@FXML private TextField placeTransferTextField;
@FXML private TextField TransferStatusTextField;
@FXML private Button addTransferRecordButton;
@FXML private Label addTransferResultLabel;
@FXML private TextField secondNameBookingTextField;
@FXML private TextField firstNameBookingTextField;
@FXML private TextField middleNameBookingTextField;
@FXML private TextField ddBirthdayBookingTextField;
@FXML private TextField mmBirthdayBookingTextField;
@FXML private TextField yyyyBirthdayBookingTextField;
@FXML private TextField phoneBookingTextField;
@FXML private TextField ddHiringBookingTextField;
@FXML private TextField mmHiringBookingTextField;
@FXML private TextField yyyyHiringBookingTextField;
@FXML private Button addBookingRecordButton;
@FXML private Label addBookingResultLabel;
@FXML private TextField secondNameReaderTextField;
@FXML private TextField firstNameReaderTextField;
@FXML private TextField middleNameReaderTextField;
@FXML private TextField ddReaderTextField;
@FXML private TextField mmReaderTextField;
@FXML private TextField yyyyReaderTextField;
@FXML private TextField TransferNumReaderTextField;
@FXML private TextField BookTitleReaderTextField;
@FXML private Button addReaderRecordButton;
@FXML private  Label  addReaderResultLabel;
@FXML private TableView <Book> BookTable;
@FXML private TableColumn<Book, Integer> idBookColumn;
@FXML private TableColumn<Book, String> BookTitleColumn;
@FXML private TableColumn<Book, String> performerColumn;
@FXML private TableColumn<Book, String> genreColumn;
@FXML private TableColumn<Book, String> BookCityColumn;
@FXML private TableColumn<Book, LocalDate> BookDateColumn;
@FXML private Button showAllBooksButton;
@FXML private Button editSelectedBookRecordButton;
@FXML private Button deleteSelectedBookRecordButton;
@FXML private TableView<Reader> ReaderTable;
@FXML private TableColumn<Reader, Integer> idReaderColumn;
@FXML private TableColumn<Reader, String> secondNameReaderColumn;
@FXML private TableColumn<Reader, String> firstNameReaderColumn;
@FXML private TableColumn<Reader, String> middleNameReaderColumn;
@FXML private TableColumn<Reader, LocalDate> birthdayReaderColumn;
@FXML private TableColumn<Reader, Integer> TransferNumReaderColumn;
@FXML private TableColumn<Reader, String> BookTitleReaderColumn;
@FXML private Button showAllReadersButton;
@FXML private Button editReaderSelectedRecordButton;
@FXML private Button deleteReaderSelectedRecordButton;
@FXML private TableView<Booking> BookingTable;
@FXML private TableColumn<Booking, Integer> idBookingColumn;
@FXML private TableColumn<Booking, String> secondNameBookingColumn;
@FXML private TableColumn<Booking, String> firstNameBookingColumn;
@FXML private TableColumn<Booking, String> middleNameBookingColumn;
@FXML private Button showAllBookingButton;
@FXML private Button editSelectedBookingRecordButton;
```

```
@FXML private Button deleteSelectedBookingRecordButton;
@FXML private TableView<Transfer> TransferTable;
@FXML private TableColumn<Transfer, Integer> idTransferColumn;
@FXML private TableColumn<Transfer, LocalDate> dateTransferColumn;
@FXML private TableColumn<Transfer, LocalTime> timeTransferColumn;
@FXML private TableColumn<Transfer, String> TransferStatusColumn;
@FXML private Button showAllTransfersButton;
@FXML private Button editSelectedTransfersRecordButton;
@FXML private Button deleteSelectedTransfersRecordButton;

private List<Reader> Readers;
private List<Book> Books;
private List<Booking> Bookings;
private List<Transfer> Transfers;
private ReaderDAO ReaderDAO;
private BookDAO BookDAO;
private BookingDAO BookingDAO;
private TransferDAO TransferDAO;
private EntityManagerFactory entityManagerFactory;
private EntityManager em;
private AlertWindow alertWindow = new AlertWindow();

@FXML
void initialize() {

setCellValueFactories();
entityManagerFactory = Persistence.createEntityManagerFactory("MyPU");
em = entityManagerFactory.createEntityManager();

ReaderDAO = new ReaderDAO(em);
BookDAO = new BookDAO(em);
BookingDAO = new BookingDAO(em);
TransferDAO = new TransferDAO(em);

addBookRecordButton.setOnAction(event -> addBook());
addTransferRecordButton.setOnAction(event -> addTransfer());
addBookingRecordButton.setOnAction(event -> addBooking());
addReaderRecordButton.setOnAction(event -> addReader());
showAllBooksButton.setOnAction(event -> showAllBooks());
editSelectedBookRecordButton.setOnAction(event -> editBook());
deleteSelectedBookRecordButton.setOnAction(event ->deleteBook());
showAllReadersButton.setOnAction(event -> showAllReaders());
editReaderSelectedRecordButton.setOnAction(event -> editReader());
deleteReaderSelectedRecordButton.setOnAction(event ->deleteReader());
showAllBookingButton.setOnAction(event -> showAllBookings());
editSelectedBookingRecordButton.setOnAction(event ->editBooking());
deleteSelectedBookingRecordButton.setOnAction(event ->deleteBooking());
showAllTransfersButton.setOnAction(event -> showAllTransfers());
editSelectedTransfersRecordButton.setOnAction(event -> editTransfer());
deleteSelectedTransfersRecordButton.setOnAction(event ->deleteTransfer());
}

private void deleteTransfer() {
Transfer Transfer = TransferTable.getSelectionModel().getSelectedItem();if
(Transfer != null &&
alertWindow.confirmationAlert("Записать",Transfer.toString())) {
TransferDAO.delete(Transfer);

TransferTable.setItems(FXCollections.observableList(TransferDAO.findAll()));
}
}

private void editTransfer() {
Transfer Transfer = TransferTable.getSelectionModel().getSelectedItem();if
```

```
(Transfer == null)
return;

Dialog<R> dialog = new Dialog<Object>();
var window = addReaderRecordButton.getScene().getWindow();
dialog.initOwner(window);
dialog.setTitle("Информация по книге");

ButtonType okButtonType = new ButtonType("Редактировать",
ButtonType.OK.getButtonData());
ButtonType cancelButtonType = new ButtonType("Отмена",
ButtonType.CANCEL.getButtonData());
dialog.getDialogPane().getButtonTypes().addAll(okButtonType,cancelButtonType);
GridPane grid = new GridPane();grid.setHgap(10); grid.setVgap(10);

grid.setPadding(new Insets(20, 20, 10, 10));

TextField BookDate = new
TextField(Transfer.getBookDate().toString());
BookDate.setPromptText("Дата");
TextField seat = new TextField(Transfer.getSeat());
seat.setPromptText("Читательский билет");
TextField TransferStatus = new TextField(Transfer.getTransferStatus());
TransferStatus.setPromptText("Статус");

grid.add(new Label("Дата"), 0, 0);
grid.add(BookDate, 1, 0);
grid.add(new Label("Читательский билет"), 0, 5);
grid.add(TransferStatus, 1, 5);
grid.add(new Label("Статус"), 0, 7);
grid.add(seat, 1, 7);
dialog.getDialogPane().setContent(grid);
dialog.setResultConverter(dialogButton -> {
if (dialogButton == okButtonType) {
try {
int indexSelected = Transfers.indexOf(Transfer);
Transfer.setBookDate(LocalDate.parse(BookDate.getText().trim()));
Transfer.setBookTime(LocalTime.parse(BookTime.getText().trim()));
Transfer.setBookAddress(BookAddress.getText().trim());
Transfer.setBookCity(BookCity.getText().trim());
Transfer.setBookPlace(BookPlace.getText().trim());
Transfer.setTransferStatus(TransferStatus.getText().trim());

Transfer.setPrice(Double.parseDouble(price.getText().trim()));
Transfer.setSeat(seat.getText().trim());

Transfers.set(indexSelected, Transfer);
TransferDAO.editRecord(Transfer);

TransferTable.setItems(FXCollections.observableArrayList(Transfers));
TransferTable.refresh();

} catch (NullPointerException e) {
alertWindow.errorAlert("Ошибка! Введены неправильные данные!");


} catch (NumberFormatException e) {
alertWindow.errorAlert("Ошибка! Введен текст вместо числа!");


} catch (DateTimeParseException e) {
alertWindow.errorAlert("Ошибка! Введена некорректная дата!");
}
}
```

```
alertWindow.errorAlert("Ошибка! Одно или несколько полей пусты!");}

return null;
});
Optional<Transfer> result = dialog.showAndWait();
}

private void showAllTransfers() {
try {
Transfers = TransferDAO.findAll();
if (Transfers.isEmpty()) {
alertWindow.errorAlert("Записи не найдены!");return;
}
TransferTable.getItems().clear();
TransferTable.setItems(FXCollections.observableList(Transfers));
}
catch (Exception ex) {
alertWindow.errorAlert(ex.getMessage());
}
}

private void deleteBooking() {Booking Booking =
BookingTable.getSelectionModel().getSelectedItem();if (Booking != null &&
alertWindow.confirmationAlert("Записать",Booking.briefInfo())) {
BookingDAO.delete(Booking);

BookingTable.setItems(FXCollections.observableList(BookingDAO.findAll()));
}
}

private void editBooking() {Booking Booking =
BookingTable.getSelectionModel().getSelectedItem();
if (Booking == null)
return;

Dialog<Booking> dialog = new Dialog<>();
var window = addReaderRecordButton.getScene().getWindow();
dialog.initOwner(window);
dialog.setTitle("Бронирование книги");

ButtonType okButtonType = new ButtonType("Редактировать",
ButtonType.OK.getButtonData());
ButtonType cancelButtonType = new ButtonType("Отмена",
ButtonType.CANCEL.getButtonData());
dialog.getDialogPane().getButtonTypes().addAll(okButtonType,cancelButtonType);
GridPane grid = new GridPane();grid.setHgap(10); grid.setVgap(10);

grid.setPadding(new Insets(20, 20, 10, 10));

TextField secondName = new TextField(Booking.getSecondName());
secondName.setPromptText("Фамилия");
TextField firstName = new TextField(Booking.getFirstName());
firstName.setPromptText("Имя");
TextField middleName = new TextField(Booking.getMiddleName());
middleName.setPromptText("Отчество");
TextField birthday = new TextField(Booking.getBirthday().toString());
birthday.setPromptText("Дата рождения");
TextField phone = new TextField(Booking.getPhone());
phone.setPromptText("Контактный телефон");
TextField hiringDate = new TextField(Booking.getHiringDate().toString());
hiringDate.setPromptText("Дата");

grid.add(new Label("Фамилия"), 0, 0);
grid.add(secondName, 1, 0);
```

```
grid.add(new Label("Имя"), 0, 1);
grid.add(firstName, 1, 1);
grid.add(new Label("Отчество"), 0, 2);
grid.add(birthday, 1, 2);
grid.add(new Label("Дата рождения"), 0, 3);
grid.add(middleName, 1, 3);
grid.add(new Label("Контактный телефон"), 0, 6);
grid.add(phone, 1, 6);
grid.add(new Label("Дата"), 0, 7);
grid.add(occupation, 1, 7);

dialog.getDialogPane().setContent(grid);
dialog.setResultConverter(dialogButton -> {
if (dialogButton == okButtonType) {
try {
int indexSelected = Bookings.indexOf(Booking);
Booking.setSecondName(secondName.getText().trim());
Booking.setFirstName(firstName.getText().trim());

Booking.setHiringDate(LocalDate.parse(hiringDate.getText().trim()));
Booking.setMiddleName(middleName.getText().trim());
Booking.setCity(city.getText().trim());
Booking.setOccupation(occupation.getText().trim());
Booking.setPhone(phone.getText().trim());
Booking.setOccupation(occupation.getText().trim());

Bookings.set(indexSelected, Booking);
BookingDAO.editRecord(Booking);

BookingTable.setItems(FXCollections.observableArrayList(Bookings));
BookingTable.refresh();
} catch (NullPointerException e) {
BookingTable.getItems().clear();
BookingTable.setItems(FXCollections.observableList(Bookings));
}
catch (Exception ex) {
alertWindow.errorAlert(ex.getMessage());
}
}

private void deleteReader() {
Reader Reader = ReaderTable.getSelectionModel().getSelectedItem();if (Reader !=
null &&
alertWindow.confirmationAlert("Читательский билет",Reader.briefInfo())) {
ReaderDAO.delete(Reader);

ReaderTable.setItems(FXCollections.observableList(ReaderDAO.findAll()));
}
}

private void editReader() {
Reader Reader = ReaderTable.getSelectionModel().getSelectedItem();if (Reader ==
null)
return;

Dialog<Reader> dialog = new Dialog<>();
var window = addReaderRecordButton.getScene().getWindow();
dialog.initOwner(window);
dialog.setTitle("Выдача");

ButtonType okButtonType = new ButtonType("Редактировать",
ButtonType.OK.getButtonData());
ButtonType cancelButtonType = new ButtonType("Отмена",
ButtonType.CANCEL.getButtonData());
```

```
dialog.getDialogPane().getButtonTypes().addAll(okButtonType,cancelButtonType);
GridPane grid = new GridPane();grid.setHgap(10); grid.setVgap(10);
grid.setPadding(new Insets(20, 20, 10, 10));

TextField secondName = new TextField(Reader.getSecondName());
secondName.setPromptText("Фамилия");
TextField firstName = new TextField(Reader.getFirstName());
firstName.setPromptText("Имя");
TextField middleName = new TextField(Reader.getMiddleName());
middleName.setPromptText("Отчество");
TextField birthday = new TextField(Reader.getBirthday().toString());
birthday.setPromptText("Дата рождения");
TextField BookTitle = new TextField(Reader.getBookTitle());
BookTitle.setPromptText("Контактный телефон");

grid.add(new Label("Фамилия"), 0, 0);
grid.add(secondName, 1, 0);
grid.add(new Label("Имя"), 0, 1);
grid.add(firstName, 1, 1);
grid.add(new Label("Отчество"), 0, 2);
grid.add(birthday, 1, 2);
grid.add(new Label("Дата рождения"), 0, 3);
grid.add(middleName, 1, 3);
grid.add(new Label("Контактный телефон"), 0, 6);
grid.add(BookTitle, 1, 6);

dialog.getDialogPane().setContent(grid); dialog.setResultConverter(dialogButton ->
{
if (dialogButton == okButtonType) {
try {
int indexSelected = Readers.indexOf(Reader);
Reader.setSecondName(secondName.getText().trim());
Reader.setFirstName(firstName.getText().trim());

Reader.setBirthday(LocalDate.parse(birthday.getText().trim()));
Reader.setMiddleName(middleName.getText().trim());
Reader.setCity(city.getText().trim());

Reader.setTransferNum(Integer.parseInt(TransferNum.getText().trim()));
Reader.setBookTitle(BookTitle.getText().trim());

Readers.set(indexSelected, Reader);
ReaderDAO.editRecord(Reader);

ReaderTable.setItems(FXCollections.observableArrayList(Readers));
ReaderTable.refresh();
} catch (NullPointerException e) {
ReaderTable.getItems().clear();
ReaderTable.setItems(FXCollections.observableList(Readers));
}
catch (Exception ex) {
alertWindow.errorAlert(ex.getMessage());
}
}

private void deleteBook() {
Book Book = BookTable.getSelectionModel().getSelectedItem();if (Book != null &&
alertWindow.confirmationAlert("Запись",Book.briefInfo())) {
BookDAO.delete(Book);

BookTable.setItems(FXCollections.observableList(BookDAO.findAll()));
}
}
```

```
private void editBook() {
Book Book = BookTable.getSelectionModel().getSelectedItem();if (Book == null)
return;

Dialog<Book> dialog = new Dialog<>();
var window = addReaderRecordButton.getScene().getWindow();
dialog.initOwner(window);
dialog.setTitle("Читательский билет");

ButtonType okButtonType = new ButtonType("Редактировать",
ButtonType.OK.getButtonData());
ButtonType cancelButtonType = new ButtonType("Отмена",
ButtonType.CANCEL.getButtonData());
dialog.getDialogPane().getButtonTypes().addAll(okButtonType,cancelButtonType);
GridPane grid = new GridPane();grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(20, 20, 10, 10));

TextField title = new TextField(Book.getTitle());
title.setPromptText("Название книги");
TextField performer = new TextField(Book.getPerfomer());
performer.setPromptText("Автор");
TextField genre = new TextField(Book.getGenre());
genre.setPromptText("Жанр");
TextField ageRestriction = new TextField(Book.getAgeRestriction());
BookCity.setPromptText("Город");
TextField BookTime = new TextField(Book.getBookTime().toString());
BookTime.setPromptText("Год");

grid.add(new Label("Название книги"), 0, 0);
grid.add(title, 1, 0);
grid.add(new Label("Автор"), 0, 1);
grid.add(performer, 1, 1);
grid.add(new Label("Жанр"), 0, 2);
grid.add(genre, 1, 2);
grid.add(new Label("Город"), 0, 4);
grid.add(BookCity, 1, 4);
grid.add(new Label("Год"), 0, 5);
grid.add(BookTime, 1, 5);

dialog.getDialogPane().setContent(grid);
dialog.setResultConverter(dialogButton -> {
if (dialogButton == okButtonType) {
try {
int indexSelected = Books.indexOf(Book);
Book.setTitle(title.getText().trim());
Book.setPerfomer(performer.getText().trim());

Book.setAgeRestriction(ageRestriction.getText().trim());
Book.setGenre(genre.getText().trim());
Book.setBookCity(BookCity.getText().trim());

Book.setBookTime(LocalTime.parse(BookTime.getText().trim()));
Book.setTransferPrice(Double.parseDouble(TransferPrice.getText().trim()));
Books.set(indexSelected, Book);
BookDAO.editRecord(Book);

BookTable.setItems(FXCollections.observableArrayList(Books));
BookTable.refresh();
} catch (NullPointerException e) {
BookTable.getItems().clear();
BookTable.setItems(FXCollections.observableList(Books));
}
catch (Exception ex) {
```

```
alertWindow.errorAlert(ex.getMessage());
}
}

private void addReader() {
if(secondNameReaderTextField.getText().isEmpty() ||
firstNameReaderTextField.getText().isEmpty() ||
middleNameReaderTextField.getText().isEmpty()
|| ddReaderTextField.getText().isEmpty() ||mmReaderTextField.getText().isEmpty()
|| yyyyReaderTextField.getText().isEmpty()
|| cityReaderTextField.getText().isEmpty() ||
TransferNumReaderTextField.getText().isEmpty() ||
BookTitleReaderTextField.getText().isEmpty()) {
try { addReaderResultLabel.setText("");Reader Reader = new Reader();

Reader.setFirstName(firstNameReaderTextField.getText().trim());
Reader.setSecondName(secondNameReaderTextField.getText().trim());
Reader.setMiddleName(middleNameReaderTextField.getText().trim());
Reader.setCity(cityReaderTextField.getText().trim());

Reader.setTransferNum(Integer.parseInt(TransferNumReaderTextField.getText().trim(
)));
Reader.setBookTitle(BookTitleReaderTextField.getText().trim());
Reader.setBirthday(LocalDate.parse(yyyyReaderTextField.getText().trim() + "-"
+ mmReaderTextField.getText().trim() + "-" +
ddReaderTextField.getText().trim()));

ReaderDAO.add(Reader); addReaderResultLabel.setTextFill(Color.web("#00FF00"));
addReaderResultLabel.setText("Запись успешно добавлена!");

if (!ReaderTable.getItems().isEmpty())

ReaderTable.setItems(FXCollections.observableList(ReaderDAO.findAll()));

firstNameReaderTextField.clear();
secondNameReaderTextField.clear();
middleNameReaderTextField.clear();
cityReaderTextField.clear();
TransferNumReaderTextField.clea();
BookTitleReaderTextField.clear();
yyyyReaderTextField.clear(); mmReaderTextField.clear();
ddReaderTextField.clear();
} catch (NullPointerException e) {
}
}

private void addBooking() {
if(secondNameBookingTextField.getText().isEmpty() ||
firstNameBookingTextField.getText().isEmpty() ||
middleNameBookingTextField.getText().isEmpty()
|| ddBirthdayBookingTextField.getText().isEmpty() ||
mmBirthdayBookingTextField.getText().isEmpty()
|| yyyyBirthdayBookingTextField.getText().isEmpty() ||
phoneBookingTextField.getText().isEmpty()
|| ddHiringBookingTextField.getText().isEmpty() ||
mmHiringBookingTextField.getText().isEmpty()
|| yyyyHiringBookingTextField.getText().isEmpty() ||
occupationBookingTextField.getText().isEmpty()
|| cityBookingTextField.getText().isEmpty()) {
try { addBookingResultLabel.setText("");Booking Booking = new Booking();


Booking.setFirstName(secondNameBookingTextField.getText().trim());
Booking.setSecondName(firstNameBookingTextField.getText().trim());
```

```
Booking.setMiddleName(middleNameBookingTextField.getText().trim());
Booking.setPhone(phoneBookingTextField.getText().trim());
Booking.setCity(cityBookingTextField.getText().trim());

Booking.setOccupation(occupationBookingTextField.getText().trim());

Booking.setBirthday(LocalDate.parse(yyyyBirthdayBookingTextField.getText().trim()
+ "-" + mmBirthdayBookingTextField.getText().trim() + "-" +
ddBirthdayBookingTextField.getText().trim()));

Booking.setHiringDate(LocalDate.parse(yyyyHiringBookingTextField.getText().trim()
+ "-" + mmHiringBookingTextField.getText().trim() + "-" +
ddHiringBookingTextField.getText().trim()));

BookingDAO.add(Booking); addBookingResultLabel.setTextFill(Color.web("#00FF00"));
addBookingResultLabel.setText("Запись успешно добавлена!");

if (!ReaderTable.getItems().isEmpty())

ReaderTable.setItems(FXCollections.observableList(ReaderDAO.findAll()));
secondNameBookingTextField.clear();
firstNameBookingTextField.clear();
middleNameBookingTextField.clear();
phoneBookingTextField.clear();
cityBookingTextField.clear();
occupationBookingTextField.clear();
yyyyBirthdayBookingTextField.clear();
mmBirthdayBookingTextField.clear();
ddBirthdayBookingTextField.clear();
yyyyHiringBookingTextField.clear();
mmHiringBookingTextField.clear();
ddHiringBookingTextField.clear();
} catch (NullPointerException e) {
}
}

private void addTransfer() {
if(ddTransferTextField.getText().isEmpty() ||
mmTransferTextField.getText().isEmpty() ||
yyyyTransferTextField.getText().isEmpty()
|| hoursTransferTextField.getText().isEmpty() ||
minutesTransferTextField.getText().isEmpty()
|| cityTransferTextField.getText().isEmpty() ||
addressTransferTextField.getText().isEmpty()
|| placeTransferTextField.getText().isEmpty() ||
seatTransferTextField.getText().isEmpty()
|| priceTransferTextField.getText().isEmpty() ||
TransferStatusTextField.getText().isEmpty()) {
try { addTransferResultLabel.setText("");
Transfer Transfer = new Transfer();
Transfer.setTransferStatus(TransferStatusTextField.getText().trim());
Transfer.setPrice(Double.parseDouble(priceTransferTextField.getText().trim()));
Transfer.setSeat(seatTransferTextField.getText().trim());
Transfer.setBookPlace(placeTransferTextField.getText().trim());

Transfer.setBookAddress(addressTransferTextField.getText().trim());
Transfer.setBookCity(cityTransferTextField.getText().trim());

Transfer.setBookDate(LocalDate.parse(yyyyTransferTextField.getText().trim() +
"-" + mmTransferTextField.getText().trim() + "-" +
ddTransferTextField.getText().trim()));

Transfer.setBookTime(LocalTime.parse(hoursTransferTextField.getText().trim() +
":" + minutesTransferTextField.getText().trim()));
```

```
TransferDAO.add(Transfer);
addTransferResultLabel.setTextFill(Color.web("#00FF00"));
addTransferResultLabel.setText("Запись про читательский билет успешно
добавлена!");

if (!TransferTable.getItems().isEmpty())

TransferTable.setItems(FXCollections.observableList(TransferDAO.findAll()));

TransferStatusTextField.clear();
priceTransferTextField.clear();
seatTransferTextField.clear();
placeTransferTextField.clear();
addressTransferTextField.clear();
cityTransferTextField.clear();
yyyyTransferTextField.clear();
mmTransferTextField.clear();
ddTransferTextField.clear();
hoursTransferTextField.clear();
minutesTransferTextField.clear();
} catch (NullPointerException e) {
}
}

private void addBook() {
if(titleBookTextField.getText().isEmpty() ||
performerBookTextField.getText().isEmpty() ||
genreBookTextField.getText().isEmpty()
|| ageRestrictionBookTextField.getText().isEmpty() ||
cityBookTextField.getText().isEmpty()
|| yyyyBookTextField.getText().isEmpty() ||
mmBookTextField.getText().isEmpty()
|| ddBookTextField.getText().isEmpty() ||
hoursBookTextField.getText().isEmpty()
|| minutesBookTextField.getText().isEmpty() ||
minTransferPriceBookTextField.getText().isEmpty()) {
try { addBookResultLabel.setText("");
Book Book = new Book();

Book.setTitle(titleBookTextField.getText().trim());
Book.setPerfomer(performerBookTextField.getText().trim());
Book.setGenre(genreBookTextField.getText().trim());

Book.setAgeRestriction(ageRestrictionBookTextField.getText().trim());
Book.setBookCity(cityBookTextField.getText().trim());

Book.setTransferPrice(Double.parseDouble(minTransferPriceBookTextField.getT
ext().trim()));

Book.setBookDate(LocalDate.parse(yyyyBookTextField.getText().trim()
+ "-" + mmBookTextField.getText().trim() + "-" +
ddBookTextField.getText().trim()));

Book.setBookTime(LocalTime.parse(hoursBookTextField.getText().trim()
+ ":" + minutesBookTextField.getText().trim()));

BookDAO.add(Book); addBookResultLabel.setTextFill(Color.web("#00FF00"));
addBookResultLabel.setText("Запись про книгу успешно добавлена!");

if (!BookTable.getItems().isEmpty())

BookTable.setItems(FXCollections.observableList(BookDAO.findAll()));
```

```
titleBookTextField.clear();
performerBookTextField.clear();
genreBookTextField.clear();
ageRestrictionBookTextField.clear();
cityBookTextField.clear();
minutesBookTextField.clear();
yyyyBookTextField.clear();
mmBookTextField.clear();
ddBookTextField.clear();
hoursBookTextField.clear();
minutesBookTextField.clear();
minTransferPriceBookTextField.clear();
} catch (NullPointerException e) {
}
}

private void setCellValueFactories() {
idBookColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
BookTitleColumn.setCellValueFactory(new PropertyValueFactory<>("title"));
performerColumn.setCellValueFactory(new PropertyValueFactory<>("perfomer"));
genreColumn.setCellValueFactory(new PropertyValueFactory<>("genre"));
ageRestrictionColumn.setCellValueFactory(new
PropertyValueFactory<>("ageRestriction"));
BookCityColumn.setCellValueFactory(new PropertyValueFactory<>("BookCity"));
BookDateColumn.setCellValueFactory(new PropertyValueFactory<>("BookDate"));
BookTImeColumn.setCellValueFactory(new PropertyValueFactory<>("BookTime"));
minPriceBookColumn.setCellValueFactory(new
PropertyValueFactory<>("TransferPrice"));

idReaderColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
secondNameReaderColumn.setCellValueFactory(new
PropertyValueFactory<>("secondName"));
firstNameReaderColumn.setCellValueFactory(new
PropertyValueFactory<>("firstName"));
middleNameReaderColumn.setCellValueFactory(new
PropertyValueFactory<>("middleName"));
birthdayReaderColumn.setCellValueFactory(new PropertyValueFactory<>("birthday"));
cityReaderColumn.setCellValueFactory(new PropertyValueFactory<>("city"));
TransferNumReaderColumn.setCellValueFactory(new
PropertyValueFactory<>("TransferNum"));
BookTitleReaderColumn.setCellValueFactory(new
PropertyValueFactory<>("BookTitle"));

idBookingColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
secondNameBookingColumn.setCellValueFactory(new
PropertyValueFactory<>("secondName"));
firstNameBookingColumn.setCellValueFactory(new
PropertyValueFactory<>("firstName"));
middleNameBookingColumn.setCellValueFactory(new
PropertyValueFactory<>("middleName"));
birthdayBookingColumn.setCellValueFactory(new PropertyValueFactory<>("birthday"));
cityBookingColumn.setCellValueFactory(new PropertyValueFactory<>("city"));
occupationBookingColumn.setCellValueFactory(new
PropertyValueFactory<>("occupation"));
phoneBookingColumn.setCellValueFactory(new PropertyValueFactory<>("phone"));
hiringDateBookingColumn.setCellValueFactory(new
PropertyValueFactory<>("hiringDate"));

idTransferColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
dateTransferColumn.setCellValueFactory(new PropertyValueFactory<>("BookDate"));
timeTransferColumn.setCellValueFactory(new PropertyValueFactory<>("BookTime"));
cityTransferColumn.setCellValueFactory(new PropertyValueFactory<>("BookCity"));
addressTransferColumn.setCellValueFactory(new
PropertyValueFactory<>("BookAddress"));
```

```
placeTransferColumn.setCellValueFactory(new PropertyValueFactory<>("BookPlace"));
seatTransferColumn.setCellValueFactory(new PropertyValueFactory<>("seat"));
priceTransferColumn.setCellValueFactory(new PropertyValueFactory<>("price"));
TransferStatusColumn.setCellValueFactory(new
PropertyValueFactory<>("TransferStatus"));
}
}
```

Додаток В Опис програми

1 Загальні відомості

Розроблене програмне забезпечення призначене для автоматизації роботи бібліотеки-філіалу та представляє собою програму для співробітника бібліотеки, який займається формуванням та зберіганням звітної інформації, а також веденням обліку книг та читачів бібліотеки.

2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація роботи працівників бібліотеки-філіалу, покращення та легкість ведення даних, зберігання та використання інформації, а також автоматизація робочого місця бібліотекаря і як наслідок, зменшення часу обслуговування читачів; забезпечення оперативного отримання інформації про наявність книг; формування читацького квитку; оформлення видачі книги.

3 Опис логічної структури

Програма розроблена на модульній основі, тобто кожний компонент (модуль) програми відповідає за конкретну дію, наприклад: формування звітів по виданим книгам (відповідає за формування та виведення кількості виданих книг читачам).

Програмні дані зберігаються у нормалізованих таблицях бази даних.

4 Технічні та програмні засоби

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) Celeron(R) – 2.16 МГц;

– RAM: 2 GB;

– HDD: 10 GB вільного місця.

Для повноцінного функціонування системи необхідно використовувати ОС Windows версії не нижче 7.

Необхідна наявність встановленого Java Runtime Environment (JRE 1.8).

5 Виклик та завантаження

Для того, щоб почати працювати з програмою, потрібно запустити виконуваний файл Biblioteka.jar, який знаходиться на робочому столі або в меню «Пуск». Після чого з'явиться вікно авторизації. Після проходження авторизації з’явиться головне меню програми.

6 Вхідні та вихідні дані

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатури та миші). Дані або вводяться вручну, або обираються із представлених списків.

Вхідні данні:

- інформація про читацький квиток (код квитка, ПІБ, адреса, телефон.);
- інформація про книги (код книги, заголовок, тип видання, номер тома, бібліотечний шифр, код видавництва, код автора, код мови, код художника, код перекладача, код палітурки, об’єм книги, дата отримання, ціна);
- документ «Бронювання книги» (код бронювання, код книги, код квитка, код видачі, дата замовлення бронювання);
- документ «Видача книги» (код квитка, код книги, дата видачі, термін, дата повернення).

Вихідні данні:

– звіт «Видані книги»:
– звіт «Інвентаризація»;
– звіт «Планове повернення»;
– звіт «Заборгованість».

Програмні дані зберігаються у нормалізованих таблицях бази даних.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

ДОДАТОК Г Інструкція користувача

Програмне забезпечення для автоматизації роботи бібліотеки-філіалу можна запустити з виконуваного файлу Biblioteka.jar, який знаходиться на робочому столі або в меню «Пуск»

Після цього з’являється вікно авторизації користувача, яке показано на рисунку Г.1. Сюди треба вводити свої дані, а саме ім’я та пароль для роботи у програмі. Якщо логін або пароль було ведено не вірно, з’являється попередження (див. рис. Г.2).

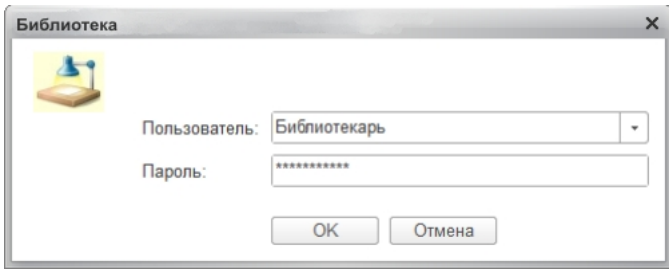

Рисунок Г.1 – Форма авторизації

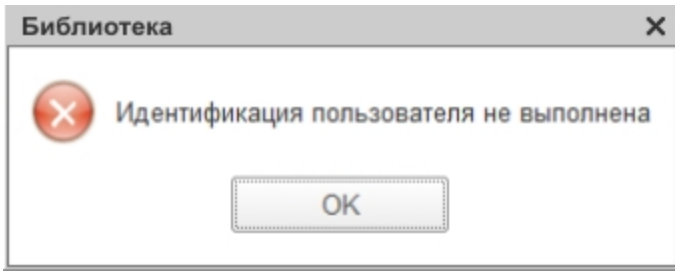

Рисунок Г.2 – Попередження

Після авторизації, з’являється головне вікно програми, яке представлено на рисунку Г.3.

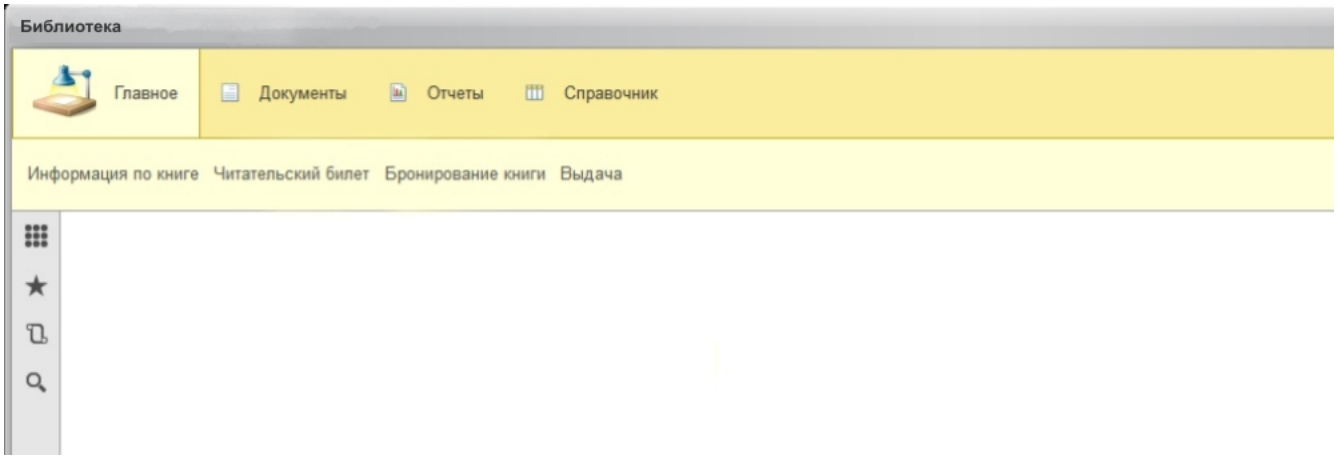

Рисунок Г.3 – Головне вікно програми

На головне меню «Главное» винесено основні довідники: «Информация по книге» та «Читательский билет», а також документи: «Бронирование книг» та «Выдача».

Для вибіру іншого довідника потрібно перейти на вкладку «Справочник» (див. рис. Г.4).

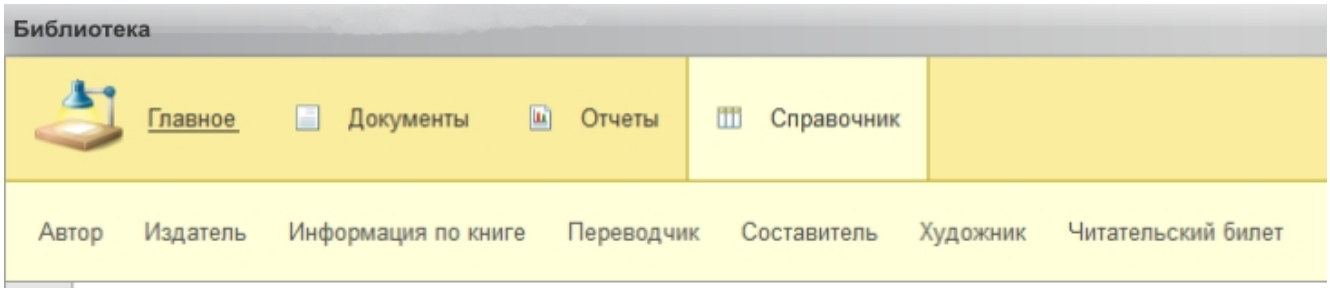

Рисунок Г.4 – Список всіх довідників

Для вибору іншого документа або звіту, потрібно виконати аналогічні дії, що і для вибору довідника.

Потрібно заповнити довідник «Информация по книге». Це відбувається наступним чином:

1. Перейти на вкладку «Главное» або «Справочники».
2. Обираємо «Информация по книге».
3. Для створення нового запису потрібно натиснути кнопку «Создать».

З’явилась нова форма для створення нового запису, потрібно заповнити

необхідні реквізити та натиснути кнопку «Записать и закрыть» (див. рис. Г.5).

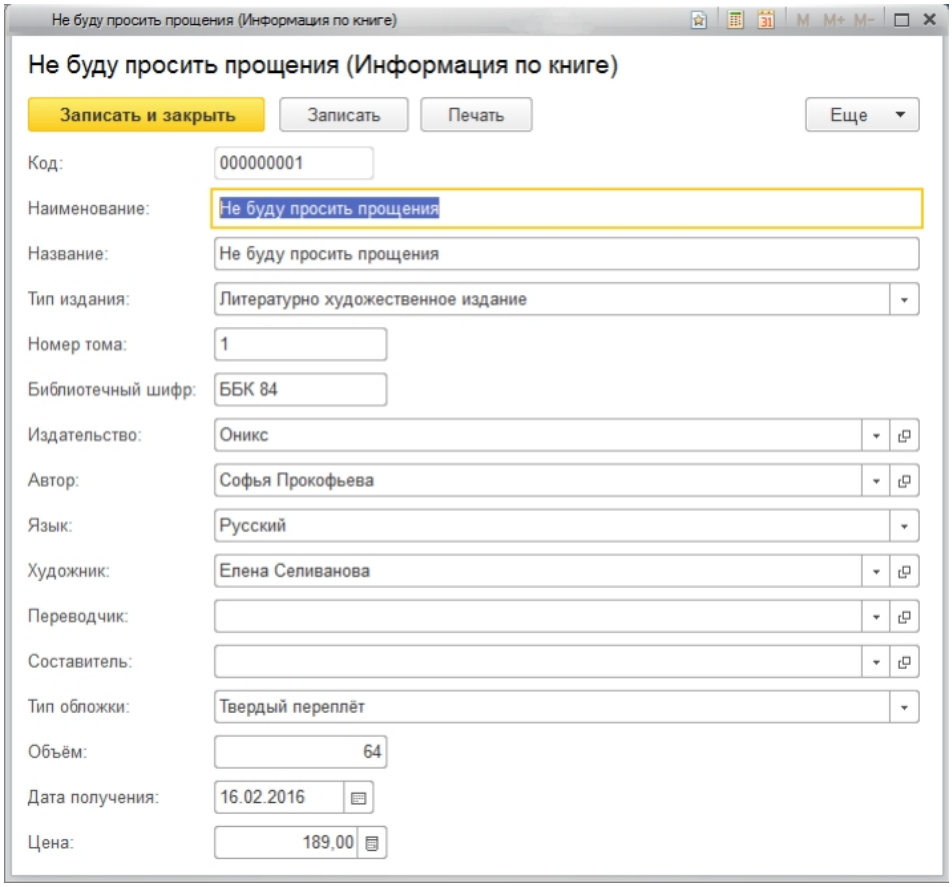

Рисунок Г.5 – Форма для заповнення реквізитів «Информация по книге».

На рисунку Г.6 зображено список всіх книг, які знаходяться у системі.

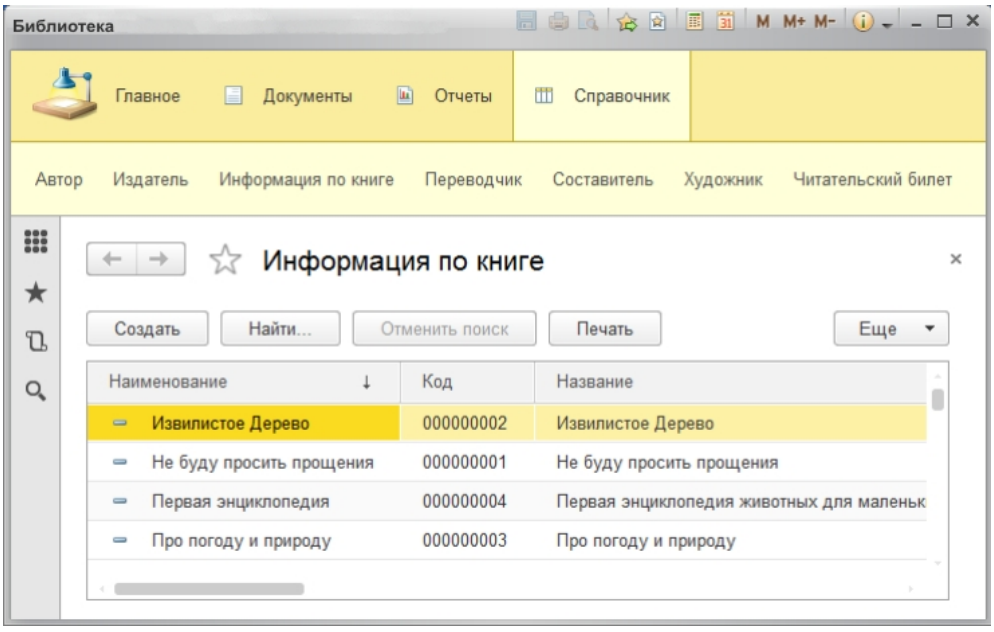

Рисунок Г.6 – Список книг

Для заповнення довідника «Читательский билет», необхідно виконати наступні дії:

1. Перейти на вкладку «Главное» або «Справочники».
2. Обираємо «Читательский билет».
3. Для створення нового запису потрібно натиснути кнопку «Создать».

З’явилась нова форма для створення нового запису, потрібно заповнити необхідні реквізити та натиснути кнопку «Записать и закрыть» (див. рис. Г.7).

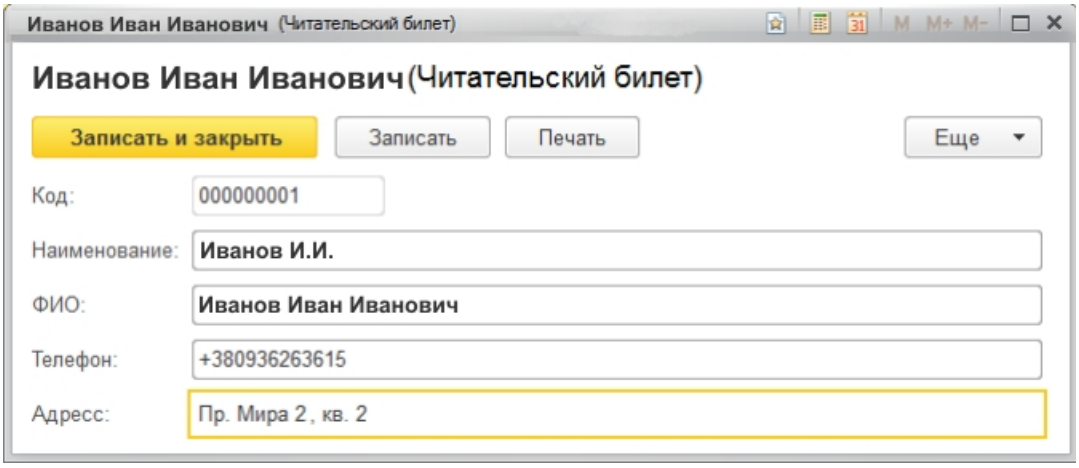

Рисунок Г.7 – Форма для заповнення реквізитів «Читательский билет»

На рисунку Г.8 відображено список всіх читачів у системі.

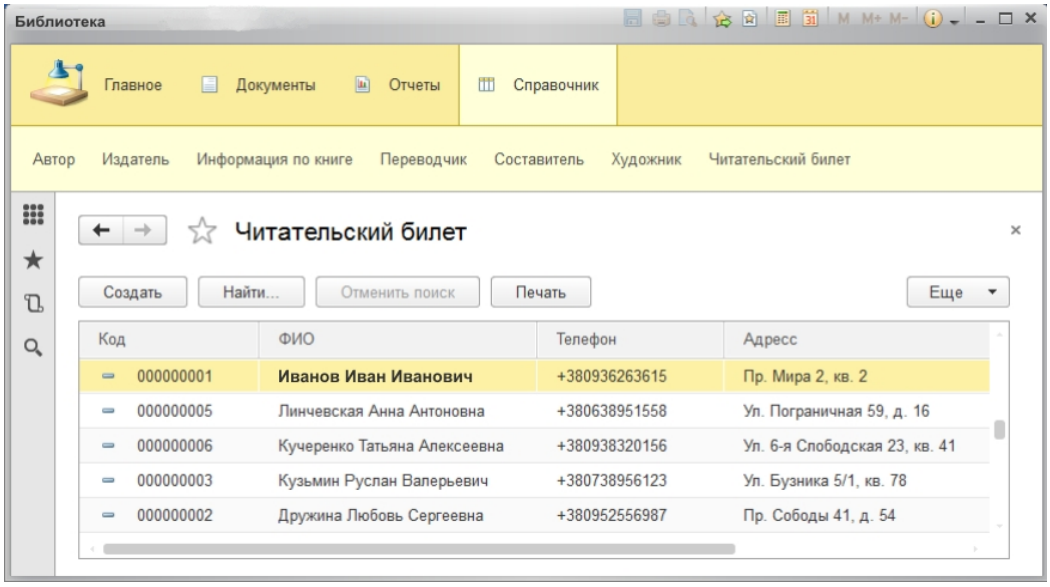

Рисунок Г.8 – Список читачів

Для заповнення інших довідників виконує аналогічні дії.

Для створення нового документу «Выдача» виконуємо наступі дії:

1. Перейти на вкладку «Главное» або «Документ».
2. Обираємо документ «Выдача».
3. Для створення нового запису потрібно натиснути кнопку «Создать».

З’явилась нова форма для створення нового запису, потрібно заповнити необхідні реквізити та натиснути кнопку «Провести и закрыть» (див. рис. Г.9).

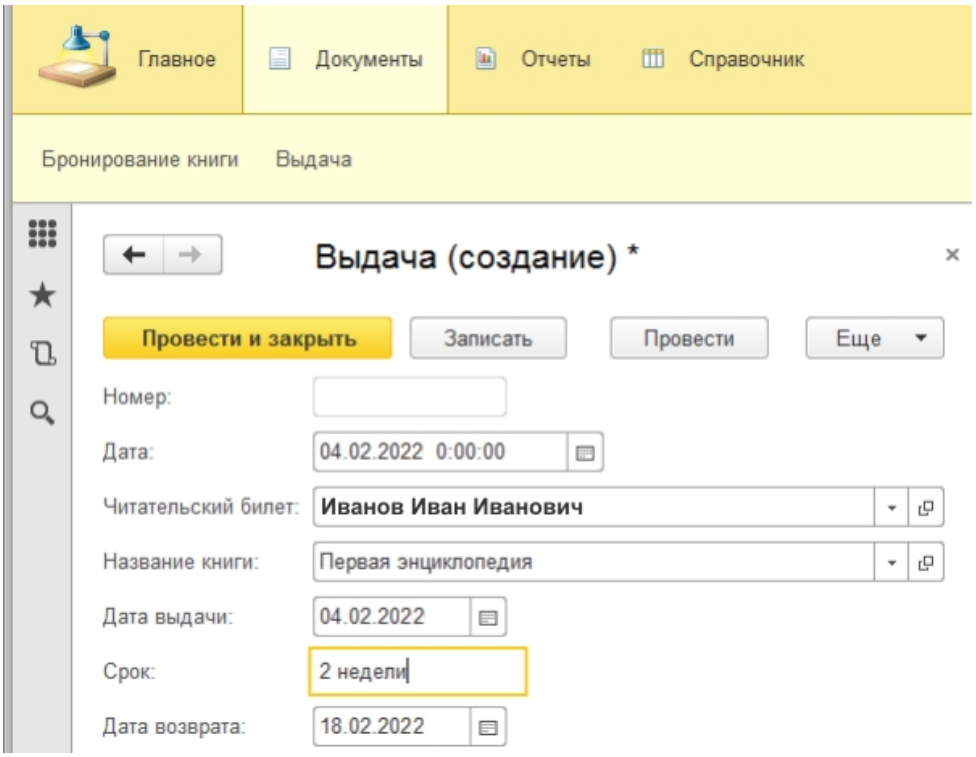

Рисунок Г.9 – Форма для заповнення резквізитів «Выдача»

Документ «Бронирование книги» заповнюється аналогічно.

Для формування звіту «Выдание книги» необхідно виконати наступні дії:

1. Перейти на вкладку «Отчеты».
2. Обрати звіт «Выданные книги».
3. Обрати період.
4. Натиснути кнопку «Сформировать» (див. рис. Г.10).

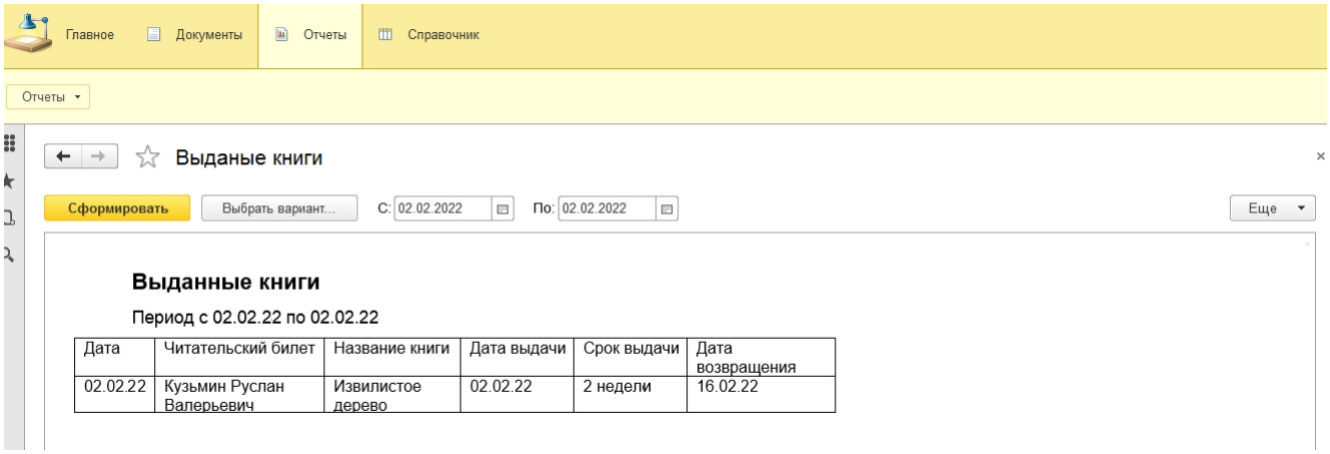

Рисунок Г.10 – Звіт «Выданные книги»

Якщо сформований звіт потрібно роздрукувати, натискаємо кнопку «Еще» та обираємо «Печать».

Додаток Д Програма та методика випробування програмного забезпечення

1 Об’єкт випробування

1.1 Назва об’єкту

Повна назва об’єкта випробувань: «Програмне забезпечення для автоматизації роботи бібліотеки-філіалу».

Коротка назва: програма.

1.2 Область застосування

Програма призначена для зберігання інформації про читачів та книги у базі даних і оперування цією інформацією для ведення обліку бібліотеки-філіалу.

Программа забезпечує виконання таких функцій:

- авторизація;
- пошук необхідної інформації;
- оформлення читацького квитка;
- оформлення видачі книги;
- оформлення бронювання книги;
- перегляд інформації про книги та читацький квиток;
- занесення та редагування даних про книгу;
- видалення інформації про книги;
- формування звіту «Видані книги»;
- формування звіту «Інвентаризація;
- формування звіту «Планове повернення»;
- формування звіту «Заборгованість».

Мета випробування

Мета проведення випробування: оцінка експлуатаційних характеристик програми, перевірка і підтвердження працездатності програми в умовах, максимально наближених до умов реальної експлуатації.

2 Вимоги до програмного забезпечення

Вимоги до програми, що підлягали перевірці, представлені у технічному завданні, яке наведено у Додатку А.

3 Вимоги до програмної документації

Для проведення випробування програми повинна бути надана наступна документація:

- технічне завдання;
- текст програми;
- програма і методика випробувань ПЗ;
- керівництво користувача.
- опис програми

4 Засоби та порядок випробувань

Випробування програми проводилося на цільовому обладнанні в тій конфігурації, яка заявлена в Технічному завданні.

Під час випробування було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу. Функції тестуються у наступному порядку: проходження авторизації, додавання нового запису, редагування запису, видалення запису, формування бронювання книги.

Всі виявлені недоліки програми повинні бути зафіксовані в протоколах і усуваються розробником до моменту впровадження програми в дію.

5 Методи випробування

Випробування було проведено за стратегією «чорного ящика». За рахунок дуже великої кількості функцій, які потрібно було випробовувати, представлено

тільки декілька результатів випробування функцій програми.

1) Функція «Оформлення та редагування читацького квитка»

На головному вікні обрали довідник «Читательский билет» та натиснули кнопку «Добавить». З’явилось відповідне вікно у якому обираємо читача, внесли необхідні реквізити та натиснули кнопку «Закрыть и сохранить». У результаті отримали сформований читацьких квиток або відредагований вже існуючий.

2) Функція «Оформлення бронювання книги»

На головному вікні обрали таблицю «Бронирование книги», та натиснули кнопку «Добавить». З’явилось відповідне вікно у якому обираємо читача, книгу та натискаємо кнопку «Закрыть и сохранить». У результаті отримали сформоване бронювання на книгу.

3) Функція «Оформлення видачі книги»

На головному вікні обрали таблицю «Выдача», та натиснули кнопку «Добавить». З’явилось відповідне вікно у якому обираємо читача, книгу та натискаємо кнопку «Закрыть и сохранить». У результаті отримали сформоване видачу на книгу.

4) Функція «Формування звіту «Видані книги»»

В головному меню натиснули на кнопку «Отчеты». З’явилося вікно з такими кнопками: «Выданные книги». Вибрали звіт та виконали встановлені початкові дії (вибрали відповідні параметри звіту). В результаті сформувався відповідний звіт.