

**АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ  
ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ МОСКОВСКИЙ  
МЕЖДУНАРОДНЫЙ КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ  
«АКАДЕМИЯ ТОП»**

**ГРУППОВОЙ ПРОЕКТ**

Уровень профессионального образования:  
Среднее профессиональное образование

Квалификация: Программист

Учебный предмет: Технология доступа к базам данных ADO.NET

Тема: Система обслуживания читателей библиотеки.

Преподаватель:  
О. А. Рослова

Участники:  
Колобов Евгений  
БуМанько Аким  
Деменев Данил

Группа: 9/3-РПО-23/1

## **СОДЕРЖАНИЕ**

ВВЕДЕНИЕ.....	3
ГЛАВА 1. БАЗА ДАННЫХ.....	4
1.1 Описание сущностей.....	4
1.2 ER диаграмма.....	4
1.3 Нормализация.....	5
1.4 Описание финальных таблиц и их атрибутов.....	8
ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММЫ.....	10
3.1 Test case.....	10
3.2 Результаты тестирования.....	12
ЗАКЛЮЧЕНИЕ.....	13
ПРИЛОЖЕНИЕ.....	14

## **ВВЕДЕНИЕ**

Сейчас автоматизация рабочих процессов является ключевым фактором для повышения эффективности деятельности любой организации. Ручной учёт книг, читателей и операций по выдаче/возврату литературы отнимает много времени, увеличивает вероятность ошибок и в принципе бесполезен когда есть возможность в автоматизации.

Данный проект посвящен созданию базы данных, которая позволит решить эти проблемы путем автоматизации основных задач сотрудника библиотеки.

Цель проекта - разработать информационную систему «Библиотека», позволяющую автоматизировать: учёт книжного фонда, регистрацию читателей, а также контроль операций по выдаче и возврату книг. Система упростит работу сотрудников библиотеки и предоставит быстрый доступ к актуальной информации.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ предметной области и определить сущности библиотеки.
2. Спроектировать логическую и физическую структуру базы данных, определить таблицы, атрибуты, типы данных и связи между ними.
3. Создать базу данных с использованием SQL SSMS
4. Заполнить таблицы базы данных тестовыми данными для демонстрации работоспособности системы.
5. Реализовать консольный интерфейс при помощи C# и LINQ в Visual Studio
6. Реализовать основные функции системы:
  - Функцию логина в аккаунт.
  - Функции автора.
  - Функции читателя.
7. Провести тестирование разработанного программного обеспечения на предмет корректности выполнения операций и целостности данных.
8. Подготовить отчетную документацию по проекту.

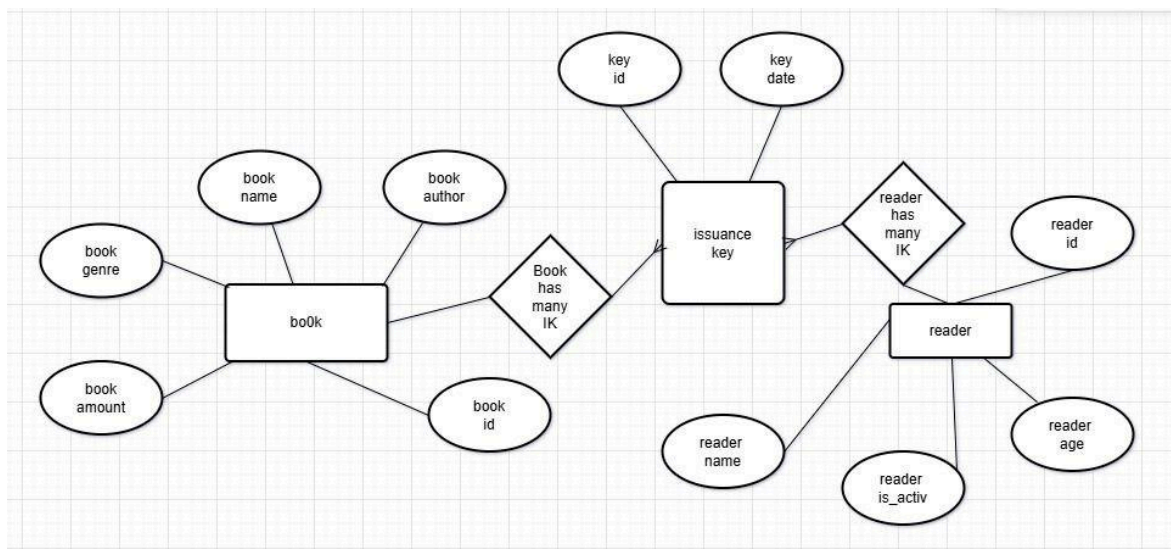
# ГЛАВА 1. БАЗА ДАННЫХ

## 1.1 Описание сущностей

Список таблиц (сущностей):

- Книги (BOOKS): кол-во книг, жанр, название, автор, id,
- Читатели (READERS): прозвище, статус, возраст, id,
- Ключ выдачи (ISSUANCE KEY): ключ, дата

## 1.2 ER диаграмма



Изображение 1: ER-диаграмма

### 1.3 Нормализация

Исходная таблица:

book_id	book_name	book_amount	book_author	book_genre
0	Война и Война	7	Худой Т. М.	Драма, Фэнтези
1	Как сварить яйцо	19	Картонков К. Д.	Кулинария, Советы
2	Сферы в Геометрии	5	Подсчётова Ю. Т	Наука, Хоррор
reader_id	reader_name	reader_birth_date	reader_active	
0	Микрочел	08.01.2010	0	
1	Денис	29.02.2002	1	
2	Infgotoinf	01.05.2005	0	
issuance_id	key_date	key_reader_id	key_book_id	key_closed
0	08.10.2025	1	1	0
1	27.08.2025	2	0	1

Таблица 1: Исходная таблица

1NF – Удаление повторов данных в таблицах

book_id	book_name	book_amount	book_author	book_genre
0	Война и Война	7	Худой Т. М.	Фэнтези
1	Как сварить яйцо	19	Картонков К. Д.	Кулинария
2	Сферы в Геометрии	5	Подсчётова Ю. Т	Наука
reader_id	reader_name	reader_birth_date	reader_active	
0	Микрочел	08.01.2010	0	
1	Денис	29.02.2002	1	
2	Infgotoinf	01.05.2005	0	
issuance_id id	key_date	key_reader_id	key_book_id	key_closed
0	08.10.2025	1	1	0
1	27.08.2025	2	0	1

Таблица 2: 1 нормализованная форма

2NF – Удаление избыточности данных, распределение данных по таблицам

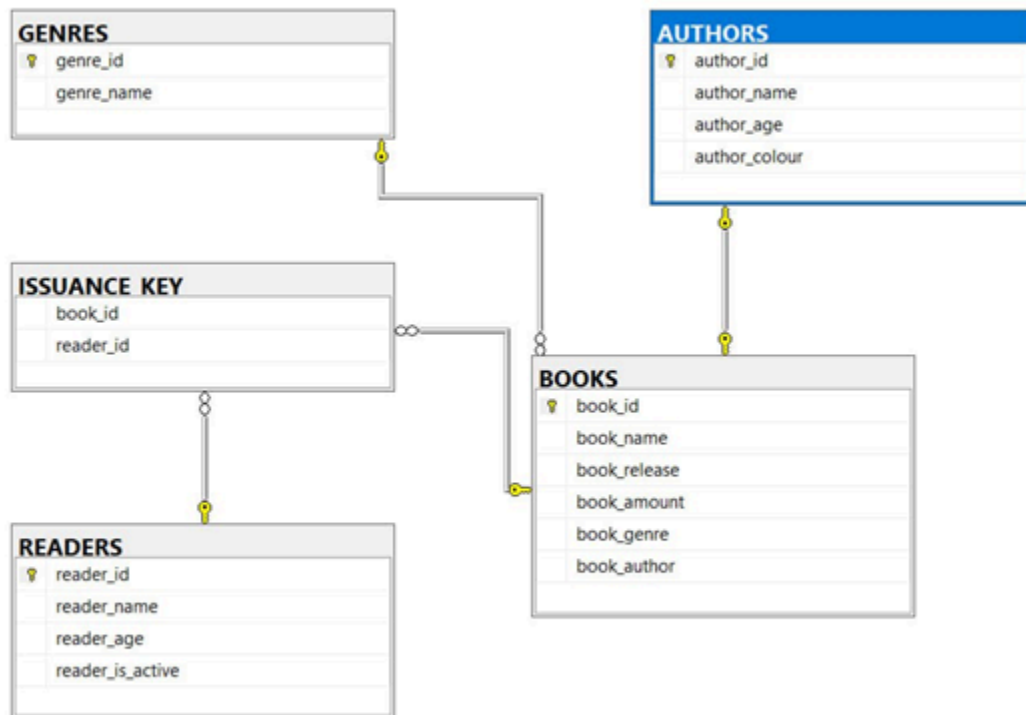
book_id	book_name	book_amount		
0	Война и Война	7		
1	Как сварить яйцо	19		
2	Сферы в Геометрии	5		
reader_id	reader_name	reader_birth_date	reader_active	
0	Микрочел	08.01.2010	0	
1	Денис	29.02.2002	1	
2	Infgotoinf	01.05.2005	0	
issuance_id	key_date	key_reader_id	key_book_id	key_closed
0	08.10.2025	1	1	0
1	27.08.2025	2	0	1
author_id	author_name	author_birth_date		
0	Худой Т. М.	19.09.1980		
1	Картонков К. Д.	03.06.1991		
2	Подсчётова Ю. Т	05.02.1844		
genre_id	genre_name			
0	Фэнтези			
1	Кулинария			
2	Наука			

Таблица 3: 2 нормализованная форма

3NF – всё соединено по ID и удалена транзитивная зависимость

book_id	book_name	book_amount	book_author	book_genre
0	Война и Война	7	0	0
1	Как сварить яйцо	19	1	1
2	Сферы в Геометрии	5	2	2
reader_id	reader_name	reader_birth_date	reader_active	
0	Микрочел	08.01.2010	0	
1	Денис	29.02.2002	1	
2	Infgotoinf	01.05.2005	0	
key_id	key_date	key_reader_id	key_book_id	key_closed
0	08.10.2025	1	1	0
1	27.08.2025	2	0	1
author_id	author_name	author_birth_date		
0	Худой Т. М.	19.09.1980		
1	Картонков К. Д.	03.06.1991		
2	Подсчётова Ю. Т	05.02.1844		
genre_id	genre_name			
0	Фэнтези			
1	Кулинария			
2	Наука			

Таблица 4: 3 нормализованная форма



Изображение 2: диаграмма из СУБД

#### 1.4 Описание финальных таблиц и их атрибутов

Атрибуты и ключи каждой таблицы:

1. Авторы: AUTHOR\_ID (PK);
2. Книги: BOOK\_ID (PK), BOOK\_GENRE (FK), BOOK\_AUTHOR (FK);
3. Читатели: READER\_ID (PK);
4. Жанры: GENRE\_ID (PK);
5. Ключ выдачи: ISSUENCE\_ID (PK), READER\_ID (FK), BOOK\_ID (FK).

AUTHORS:

- author\_id (PK) – int – NOT NULL
- author\_name – nvarchar (32) – NOT NULL
- author\_birth\_date – date

BOOKS:



- book\_id (PK) – int – NOT NULL
- book\_name – nvarchar (64) – NOT NULL
- book\_amount – int – NOT NULL
- book\_author (FK) – int – NOT NULL
- book\_genre (FK) – int – NOT NULL

#### ISSUANCE KEY:

- book\_id (FK) – int – NOT NULL
- reader\_id (FK) – int – NOT NULL

#### READERS:

- reader\_id (PK) – int – NOT NULL
- reader\_name – nvarchar (32) – NOT NULL
- reader\_birth\_date – date
- reader\_is\_active – bit – NOT NULL

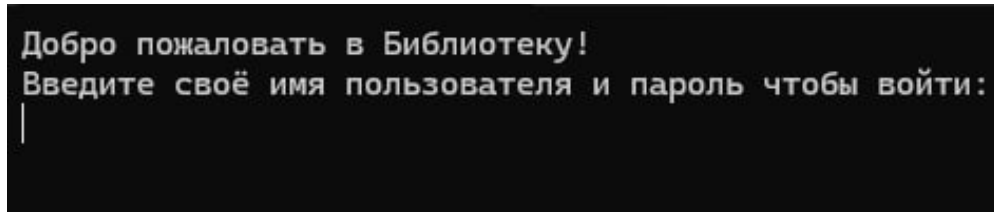
#### GENRES:

- genre\_id (PK) – int – NOT NULL
- genre\_name – nvarchar (32) – NOT NULL

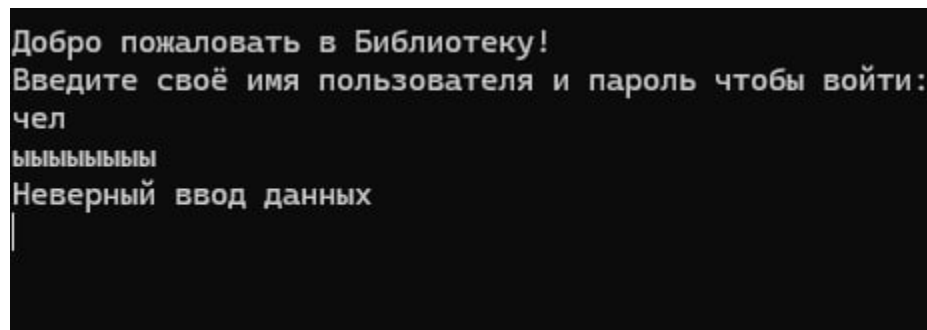
## ГЛАВА 2. ИНТЕРФЕЙС ПРОГРАММЫ

### 2.1 Общий интерфейс для пользователей

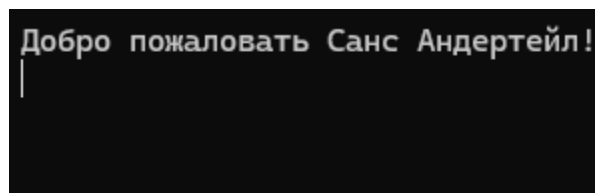
При входе в программу первое, на что мы обращаем внимание - это естественно её интерфейс, и так как мы делаем консольное приложение, то самое главное, чтобы с интерфейсом был удобен и понятен.



Изображение 3: Экран входа



Изображение 4: Экран неудачного входа



Изображение 5: Экран удачного входа

### 2.2 Интерфейс автора

Интерфейс автора позволяет создавать и изменять свои книги.

```
Введите, что вы хотите сделать:  
1: Посмотреть свои книги  
2: Добавить книгу  
3: Изменить книгу  
4: Посмотреть жанры  
0: Выйти  
|
```

Изображение 6: Интерфейс автора

```
1  
book id book name      genre  book ammount  
1      Программирование на языке C++  Шутка  11  
5      Гайд на попадание в тюрьму      Философия  25  
13     Лол      Дебаговый ад      5  
|
```

Изображение 7: Пример просмотра своих книг

```
Введите название новой книги:  
Крутая книга  
Введите айди жанра новой книги:  
6  
Введите количество этих книг:  
13  
Книга добавлена  
|
```

Изображение 8: Пример создания новой книги

```

Введите Id книги которую хотите изменить:
16
Хотите поменять название книги (Супер крутая книга)?[y/N]
y
Введите новое название книги:
Супер пупер крутая книга
Хотите поменять жанр книги (6)?[y/N]
y
Введите новое id жанра книги:
5
Хотите поменять кол-во книги (13)?[y/N]

Книга изменена
|

```

Изображение 9: Пример изменения книги

```

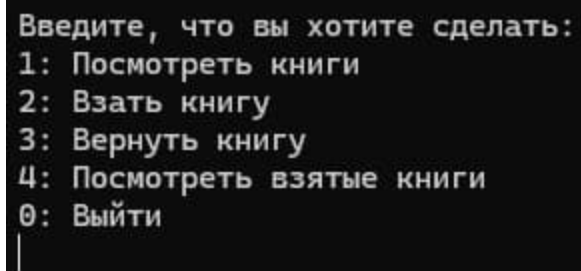
genre id      genre name
1      Библийский триллер
2      Библиотечный хоррор
3      Дебаговый ад
4      Хеллоуинские страшилки
5      Шутка
6      Сказки на ночь
7      Рецепты
8      Философия
|

```

Изображение 10: Пример вывода жанров

### 2.3 Интерфейс читателя

Интерфейс читателя позволяет получать и возвращать книги.



```
Введите, что вы хотите сделать:  
1: Посмотреть книги  
2: Взять книгу  
3: Вернуть книгу  
4: Посмотреть взятые книги  
0: Выйти  
|
```

Изображение 6: Интерфейс читателя

## ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

### 3.1 Test case

Для проверки корректности работы основной бизнес-логики был создан отдельный тестовый проект `fanfic.bible.tests`. Тесты используют технологию `MSTest` и базу данных в памяти (`In-Memory Database`) для изоляции от основной БД. Ниже приведены два сценария тестирования.

Название	Test Case: Успешная выдача доступной книги
Цель	Проверить, что функция <code>issue_book</code> корректно обрабатывает выдачу книги, которая есть в наличии.
Предусловия	<ol style="list-style-type: none"><li>1. В базе данных существует книга с <code>book_id = 1</code>.</li><li>2. У этой книги <code>book_amount &gt; 0</code>.</li></ol>
Шаги	<ol style="list-style-type: none"><li>1. Вызвать метод <code>dbMiddleMan.issue_book(reader_id: 1, book_id: 1)</code>.</li></ol>
Ожидаемый результат	<ol style="list-style-type: none"><li>1. Метод возвращает <code>true</code>.</li><li>2. Значение <code>book_amount</code> для книги с <code>book_id = 1</code> уменьшилось на 1.</li><li>3. В таблице <code>issuance_keys</code> создана новая запись о выдаче.</li></ol>

Таблица 5. Сценарий тестирования №1: Успешная выдача книги

<b>Название</b>	Test Case: Попытка выдачи книги, которой нет в наличии
Цель	Проверить, что функция <code>issue_book</code> не позволяет выдать книгу, которой нет в наличии.
Предусловия	<ol style="list-style-type: none"> <li>1. В базе данных существует книга с <code>book_id = 2</code>.</li> <li>2. У этой книги <code>book_amount = 0</code>.</li> </ol>
Шаги	<ol style="list-style-type: none"> <li>1. Вызвать метод <code>dbMiddleMan.issue_book(reader_id: 1, book_id: 2)</code>.</li> </ol>
Ожидаемый результат	<ol style="list-style-type: none"> <li>1. Метод возвращает <code>false</code>.</li> <li>2. Значение <code>book_amount</code> для книги с <code>book_id = 2</code> не изменилось.</li> <li>3. Новая запись в <code>issuance_keys</code> не создана.</li> </ol>

Таблица 6. Сценарий тестирования №2: Попытка выдачи отсутствующей книги

### 3.2 Результаты тестирования

Все реализованные модульные тесты были запущены с помощью отдельного тестового приложения. Все тесты успешно пройдены, что подтверждает корректность работы ключевых функций бизнес-логики программы.

```
Тестовый проект запущен!  
Версия .NET: 8.0.20  
  
Загружена сборка: fanfic.bible.tests, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null  
  
Running tests for: AddBookTests  
[PASS] AddBook_ShouldAddBook_WhenGenreExists  
[PASS] AddBook_ShouldNotAddBook_WhenGenreDoesNotExist  
Running tests for: GetAuthorIdByNameTests  
[PASS] GetAuthorIdByName_ShouldReturnCorrectId_WhenAuthorExists  
[PASS] GetAuthorIdByName_ShouldReturnZero_WhenAuthorDoesNotExist  
Running tests for: GetAuthorsBooksTests  
[PASS] GetAuthorsBooks_ShouldReturnCorrectBooks_WhenAuthorHasBooks  
[PASS] GetAuthorsBooks_ShouldReturnEmptyList_WhenAuthorHasNoBooks  
Running tests for: GetBooksTests  
[PASS] GetBooks_ShouldReturnAllBooks  
Running tests for: GetBookTests  
[PASS] GetBook_ShouldReturnBook_WhenBookExists  
[PASS] GetBook_ShouldReturnNull_WhenBookDoesNotExist  
Running tests for: GetGenresTests  
[PASS] GetGenres_ShouldReturnAllGenres  
Running tests for: GetReaderIdByNameTests  
[PASS] GetReaderIdByName_ShouldReturnCorrectId_WhenReaderExists  
[PASS] GetReaderIdByName_ShouldReturnZero_WhenReaderDoesNotExist  
Running tests for: GetUserTests  
[PASS] GetUser_ShouldReturnCorrectUserModel  
Running tests for: IssueBookTests  
[PASS] IssueBook_ShouldReturnFalse_WhenBookDoesNotExist  
[PASS] IssueBook_ShouldReturnFalse_WhenBookIsNotAvailable  
[PASS] IssueBook_ShouldReturnTrueAndDecreaseAmount_WhenBookIsAvailable  
Running tests for: UnIssueBookTests  
[PASS] UnIssueBook_ShouldReturnFalse_WhenKeyDoesNotExist  
[PASS] UnIssueBook_ShouldReturnTrueAndIncreaseAmount_WhenKeyExists  
Running tests for: UserIsInAuthorsTests  
[PASS] UserIsInAuthors_ShouldReturnFalse_WhenAuthorDoesNotExist  
[PASS] UserIsInAuthors_ShouldReturnTrue_WhenAuthorExists  
Running tests for: UserIsInReadersTests  
[PASS] UserIsInReaders_ShouldReturnFalse_WhenReaderDoesNotExist  
[PASS] UserIsInReaders_ShouldReturnTrue_WhenReaderExists  
  
TEST SUMMARY:  
Total tests: 22  
Passed: 22  
Failed: 0  
Success rate: 100.0%
```

Рисунок 3. Результаты выполнения тестов



## ЗАКЛЮЧЕНИЕ

Спроектирована и реализована реляционная база данных, отвечающая требованиям предметной области и нормализованная до 3НФ.

Создано консольное приложение на C# с использованием Entity Framework, которое предоставляет основной функционал для двух ролей пользователей: автора и читателя. Разработан и интегрирован отдельный модуль для автоматизированного тестирования, что позволило обеспечить высокое качество и надежность ключевых функций программы.

Трудности и их преодоление:

- Проблема: В процессе разработки возникла сложность с организацией тестового проекта. Первоначальная идея запускать тесты при каждом старте основного приложения приводила к циклическим зависимостям и ошибкам при загрузке сборок.
- Решение: Было принято решение полностью разделить основное и тестовое приложения. Тестовый проект был преобразован в отдельное консольное приложение. Это является стандартной практикой в индустрии и позволило изолировать тесты, упростить запуск и отладку как основного кода, так и тестов.
- Проект готов к демонстрации и дальнейшему развитию. Возможные направления для улучшения: переход на графический интерфейс (например, WPF или веб-интерфейс), расширение ролевой модели (например, добавление роли "Библиотекарь"), реализация более сложной логики поиска и фильтрации книг.

## ПРИЛОЖЕНИЕ

Ниже представлены листинги ключевых методов из класса dbMiddleMan, отвечающего за основную бизнес-логику и взаимодействие с базой данных.

### Метод GetAuthorsBooks(int user\_id)

Возвращает: List<book>

Комментарий: Метод принимает id автора и возвращает все его книги в виде листа.

```
// 1. Посмотреть свои книги
public List<book> GetAuthorsBooks(int user_id)
{
    var info = from book in db.books
                where book.book_author_id == user_id
                select book;
    return info.ToList();
}
```

### Метод GetGenres()

Возвращает: List<genre>

Комментарий: Возвращает лист со всеми существующими жанрами.

```
// 4. Посмотреть жанры
public List<genre> GetGenres()
{
    var info = from genre in db.genres
                select genre;
    return info.ToList();
}

public void PrintGenres()
{
    Console.WriteLine($"genre id\tgenre name");
    foreach (genre? genre in GetGenres())
    {
        Console.WriteLine($"{genre.genre_id}\t{genre.genre_name}");
    }
}
```

### Метод get\_books()

Возвращает: List<Book>

Комментарий:

Простая функция возвращающая список книг. Аналогична “SELECT \* FROM books”

```
public List<book> get_books()
{
    var info = from book in db.books
                select book;
    return info.ToList();
}
```

### Метод AddBook(book new\_book)

Возвращает: bool

Комментарий: Метод принимает объект book. Перед добавлением он проверяет, существует ли в базе данных жанр с указанным book\_genre\_id. Если жанр существует, книга добавляется в базу данных и изменения сохраняются.

```

public bool AddBook(book new_book)
{
    var info = from genre in db.genres
                where genre.genre_id == new_book.book_genre_id
                select genre;
    if (!info.Any())
    {
        return false;
    }
    else
    {
        db.books.Add(new_book);
        db.SaveChanges();
        return true;
    }
}

```

#### Метод issue\_book(int reader\_id, int book\_id)

Возвращает: bool

Комментарий: Метод находит книгу по ID. Если книга существует и есть в наличии (book\_amount > 0), её количество уменьшается на единицу и создается новая запись в таблице issuance\_keys со статусом "не закрыта" (ik\_closed = false).

```

public bool issue_book(int reader_id, int book_id)
{
    //book issuedBook = db.books
    // .Where(id => id.book_id == book_id)
    // .ToList()[0]; //Видимо это всё же не костыль, но get_user коммент слишком смешной чтобы его удалять

    // Заменяет верхнее потому что
    https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/sql/linq/how-to-update-rows-in-the-database
    book? issuedBook = (from book in db.books
                        where book.book_id == book_id
                        select book).FirstOrDefault(); //для работы теста

    if (issuedBook == null || issuedBook.book_amount <= 0)
    {
        return false;
    }
    issuedBook.book_amount--;

    issuance_key newKey = new issuance_key();

    newKey.ik_date = DateOnly.FromDateTime(DateTime.Now);
    newKey.ik_book_id = book_id;
    newKey.ik_reader_id = reader_id;
    newKey.ik_closed = false;

    db.issuance_keys.Add(newKey);
    db.SaveChanges();
    //db.issuance_keys.InsertOnSubmit(newKey);
    //db.SubmitChanges();
    return true;
}

```

#### Метод `un_issue_book(int ik_reader_id, int ik_book_id)`

Возвращает: bool

Комментарий: Метод находит запись о выдаче (`issuance_key`) по её ID. Если запись найдена, её статус меняется на "закрыта" (`ik_closed = true`), а количество экземпляров соответствующей книги увеличивается на единицу.

```
public bool un_issue_book(int ik_reader_id, int ik_book_id)
{
    issuance_key? key = (from issuance_key in db.issuance_keys
                        where issuance_key.ik_reader_id == ik_reader_id && issuance_key.ik_book_id == ik_book_id
                        select issuance_key).FirstOrDefault();

    if (key == null)
    {
        return false;
    }

    book? bookInfo = db.books.Find(key.ik_book_id);

    if (bookInfo == null)
    {
        return false;
    }

    key.ik_closed = true;
    bookInfo.book_amount += 1;

    db.SaveChanges();
    return true;
}
```

#### Метод `get_book(int book_id)`

Возвращает: book? ( book или null )

```
public List<book> get_books()
{
    var info = from book in db.books
               select book;
    return info.ToList();
}
```

Комментарий:

Поиск книги по id. Если книги с указанным id не существует, возвращается null

### Метод `get_user(int user_id)`

Возвращает: `UserModel`

```
public UserModel get_user(int user_id)
{
    UserModel model = new UserModel();

    model.readerInfo = db.readers
        .Where(i => i.reader_id == user_id)
        .ToList()[0]; // БОТ ЭТО КОСТЫЛЬ, а не эта всякая акимовская фигня

    model.issuances = db.issuance_keys
        .Where(ik => ik.ik_reader_id == user_id)
        .ToList();

    model.issuedBookIds = model.issuances
        .Select(i => i.ik_book_id)
        .ToList();

    return model;
}
```

Комментарий:

Получение модели пользователя(см. ниже) по id.

Ниже представлены модели данных использованные в проекте.

Модель(Структура) `UserModel`

Поля:

`readerInfo: reader;`

`issuances: List<issuance_key>`

`issuedBookIds: List<int>`

Комментарий:

Модель пользователя, содержащая всю касающуюся его информацию. Содержит: соответствующую строку из таблицы читателей, список строк из таблицы ключей выдачи (issuance keys), список id книг выданных пользователю.