Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Расчетно-графическая работа

по дисциплине «Базы данных»

Выполнил: студент 4 курса

ИВТ, гр. ИП-111

Кузьменок Д.В.

Работу проверил: ассистент кафедры ПМиК

Дьячкова Марина Сергеевна

Новосибирск 2024

# Задание

Тема базы данных: Книжный магазин

В ходе работы необходимо:

1. Спроектировать произвольную *базу данных*, соответствующую предметной области (согласно варианту) и позволяющую корректно хранить информацию об объектах этой предметной области.
2. Обеспечить пользовательский *веб-интерфейс* для добавления записей и просмотра содержимого БД.
3. Сформулировать и выполнить *запросы* к БД, имеющие смысл в контексте предметной области (согласно варианту). Обеспечить веб-интерфейс для запуска этих запросов.
4. Составить *отчет*, содержащий описание выполненной работы, в том числе некоторые этапы проектирования БД (см. ниже).

# Описание базы данных

2.1

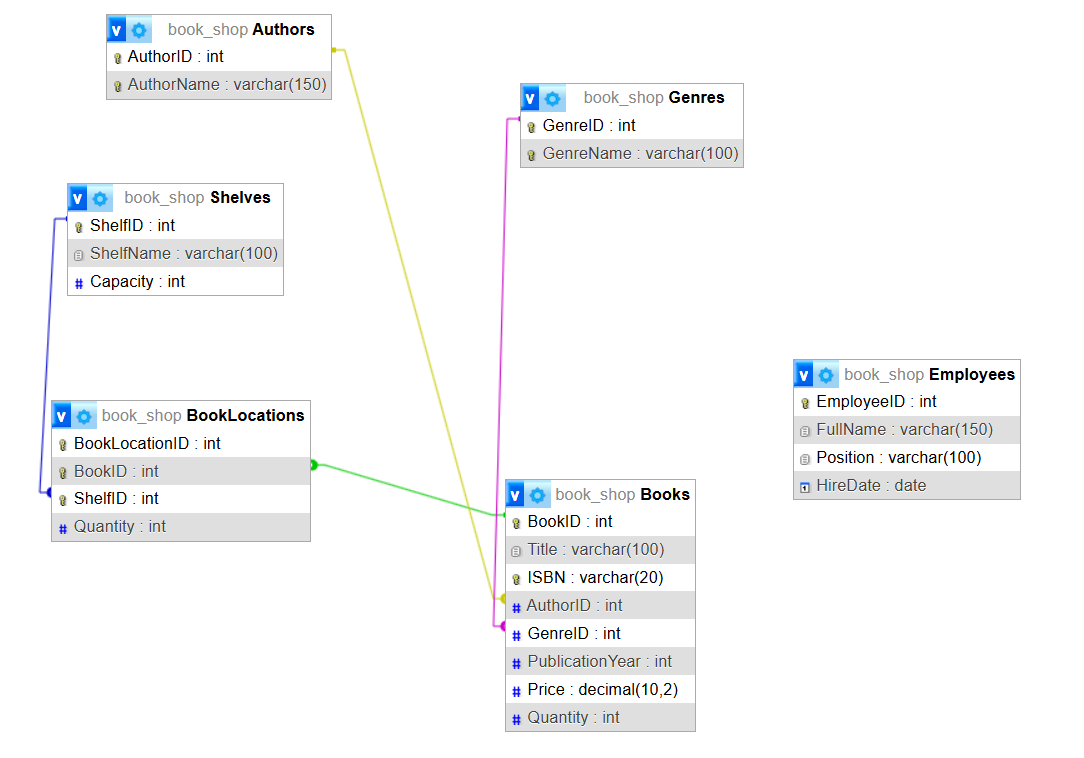


Рисунок 1. ER-диаграмма БД

Таблица 1. Сущности и связи

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность 1 | Сущность 2 | Тип связи (1:1, 1:M, M:N) | Описание связи |
| Жанры | Книги | 1:M | Один жанр может иметь множество книг, но каждая книга принадлежит только одному жанру. |
| Авторы | Книги | 1:M | Один автор может написать множество книг, но каждая книга написана только одним автором. |
| Книги | Расположение книг | 1:M | Одна книга может находиться на нескольких полках (если книга имеет сразу несколько жанров). |
| Полки | Расположение книг | 1:M | Одна полка может хранить множество книг, но каждая запись о расположении книги ссылается на одну конкретную полку. |
| Полки | Книги | M:N | Одна полка может содержать в себе множество книг, и каждая книга может находиться на нескольких полках (если книга имеет сразу несколько жанров). |

Таблица 2. Атрибуты сущностей

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Сущность, название таблицы в БД | Название атрибута | Название атрибута в БД (имя поля) | Тип данных | Not Null | Primary key | Foreign key | Дополнительные ограничения целостности |
| Жанры | уникальный идентификатор жанра | GenreID | int |  | P |  | AUTO\_INCREMENT |
| **genres** | название жанра | GenreName | varchar (100) | 1 |  |  | UNIQUE |
| Авторы | уникальный идентификатор автора | AuthorID | int |  | P |  | AUTO\_INCREMENT |
| **authors** | имя автора | AuthorName | varchar (150) | 1 |  | UNIQUE |
| Книги | уникальный идентификатор книги | BookID | int |  | P |  | AUTO\_INCREMENT  UNIQUE  REFERENCES Genres(GenreID)  REFERENCES Authors(AuthorID)  DEFAULT 1 |
| **books** | название книги | Title | varchar (100) | 1 |  |  |
|  | уникальный номер книжного издания | ISBN | varchar (20) |  |  | 1 |
|  | идентификатор жанра | GenreID | int | 1 |  |  |
|  | идентификатор автора  год публикации  цена  количество | AuthorID  PublicationYear  Price  Quantity | int  int  decimal (10,2)  int | 1  1 |  | 1 |
| Стеллажи | уникальный идентификатор стеллажа | ShelfID | int |  | P |  | AUTO\_INCREMENT |
| **shelves** | название полки | ShelfName | varchar (100) | 1 |  |  |
|  | вместимость | Capacity | int | 1 |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| Расположение книг  **booklocations** | уникальный идентификатор расположения  идентификатор книги  идентификатор полки    количество | BookLocationID  BookID  ShelfID    Quantity | int  int  int    int | 1  1    1 | P | 1  1 | AUTO\_INCREMENT  REFERENCES Books (BookID), UNIQUE KEY  REFERENCES Shelves (ShelfID), UNIQUE KEY    DEFAULT 1 |
|  |  |  |  |  |  |  |  |
| Сотрудники  **employees** | уникальный идентификатор сотрудника  имя сотрудника  должность    дата найма | EmployeeID  FullName  Position  HireDate | int  varchar (150)  varchar (100)  date | 1  1  1 | P |  | AUTO\_INCREMENT |

F1

2.2 Данные БД (содержимое таблиц) Таблица Genres

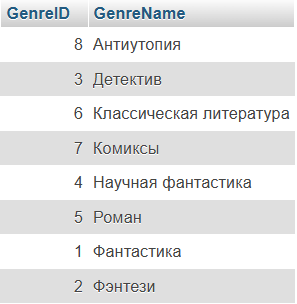


Таблица Authors



Таблица Books

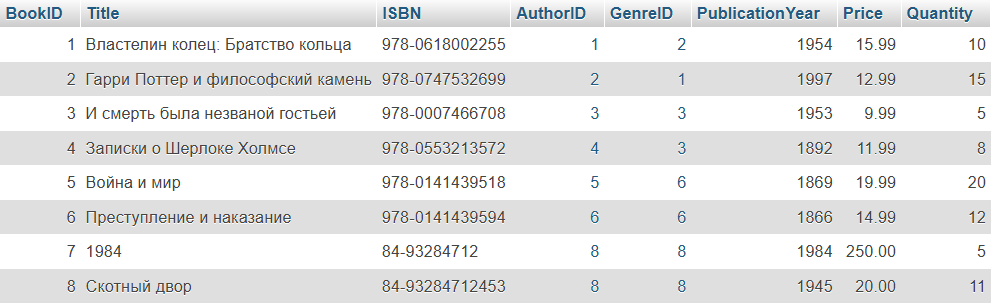


Таблица Shelves

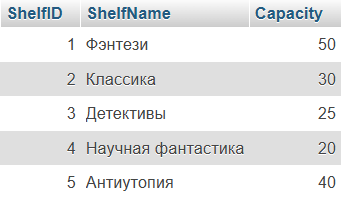


Таблица BookLocations

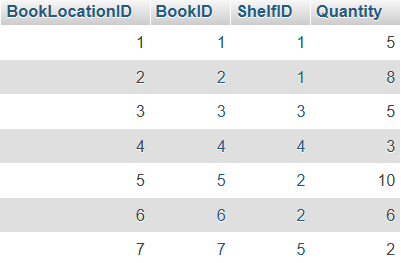
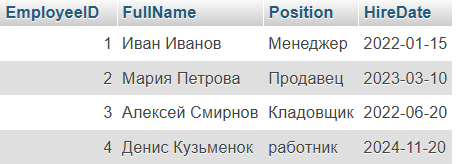


Таблица Employees



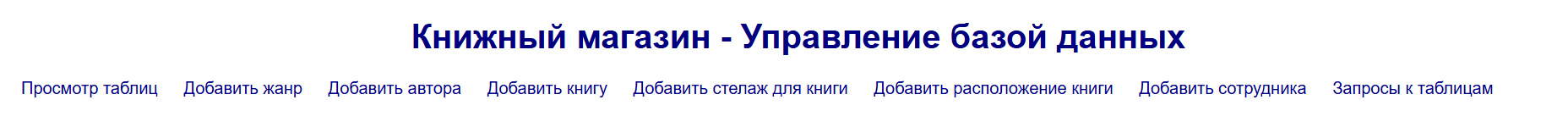
# Описание веб-интерфейса

Веб-интерфейс состоит из нескольких страниц, каждая из которых предоставляет доступ к определенному функционалу для управления данными в базе.

## Главная страница

*Функционал:*

* + - Содержит ссылки на страницы для добавления жанра, автора, книги, стеллажа, расположения книги, сотрудника.
    - Содержит ссылки на просмотр таблиц и выполнения запросов



## Просмотр данных таблиц

Эта страница предоставляет пользователю возможность просматривать данные из выбранной таблицы базы данных (таких как Genres, Authors, Books, Shelves, BookLocations, Employees).

Интерфейс выполнен с динамическим выбором и загрузкой содержимого таблицы на основе параметра, передаваемого через форму.

*Функционал:*

* + - Выбор таблицы для отображения: пользователь может выбрать таблицу, данные из которой он хочет просмотреть, с помощью выпадающего списка.
    - Просмотр содержимого выбранной таблицы: при выборе таблицы и нажатии на кнопку

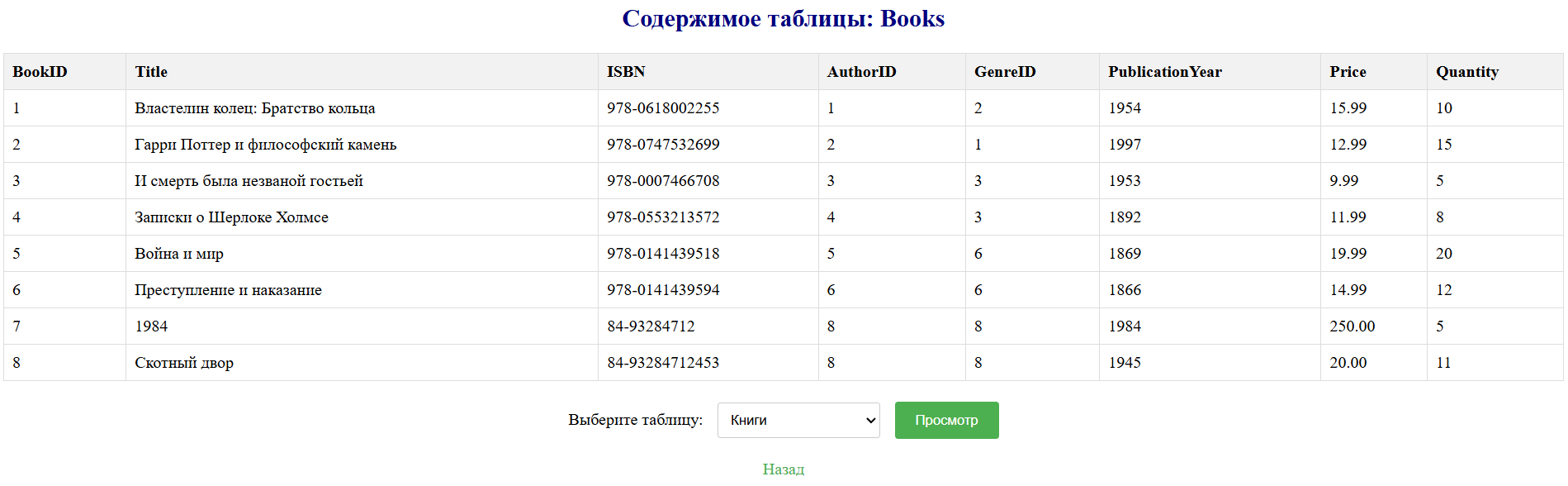
«Просмотр» загружается содержимое указанной таблицы в виде HTML-таблицы.

*Элементы интерфейса:*

* + - Форма выбора таблицы:
      * Поле «Выберите таблицу»: Выпадающий список с названиями таблиц.
      * Кнопка «Просмотр»: при нажатии отправляет запрос, чтобы загрузить данные из выбранной таблицы.
    - Таблица данных:
      * Если выбранная таблица существует, данные выводятся в таблице с колонками, соответствующими структуре таблицы.
      * Шапка таблицы: Заголовки автоматически генерируются и соответствуют полям в базе данных.
      * Строки данных: Данные из таблицы базы отображаются в строках. Каждая строка соответствует записи в таблице базы данных.
    - Кнопка «Назад»: возвращает пользователя на главную страницу.

*Проверки и ограничения:*

1. Проверка существования таблицы:
   * Если таблица с указанным именем отсутствует в базе данных, выводится сообщение об ошибке: «Таблица ... не найдена в базе данных.»



## Добавление Жанра

Эта страница позволяет пользователю добавлять новую запись в таблицу Genres.

*Функционал страницы:*

* + - Форма добавления жанра: Пользователь вводит название нового жанра.
    - Валидация данных: проводится проверка данных, что поле не пустое.
    - Запись данных в базу: при успешной проверке данных новая запись добавляется в таблицу Genres.

*Элементы интерфейса:*

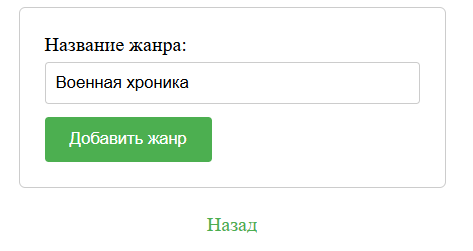
* + - Поле «Название жанра»:
      * Текстовое поле для ввода названия жанра. Поле обязательно для заполнения.
    - Кнопка «Добавить жанр»:
      * Кнопка отправки формы для добавления жанра в базу данных.
    - Кнопка «Назад»:
      * Ссылка для возврата на главную страницу.

*Проверки и ограничения:*

1. Проверка названия:
   * Поле «Название жанра» должно быть обязательно заполнено.

*Действия при отправке формы:*

1. Если данные успешно прошли все проверки, запись добавляется в базу данных.
2. При успешной записи выводится сообщение «Жанр успешно добавлен!».
3. Если добавление не удалось, на экран выводится сообщение с описанием ошибки.



## Добавление Автора

Эта страница предназначена для добавления нового автора в таблицу Authors.

Пользователь вводит информацию об авторе, включая имя. При корректном заполнении форма добавляет запись в базу данных.

*Функционал страницы*

* Форма добавления автора: Пользователь вводит полное имя автора.
* Валидация данных: Форма проводит проверки введённых данных на корректность.
* Запись данных в базу: при успешной проверке новая запись добавляется в таблицу Authors**.**

*Элементы интерфейса*

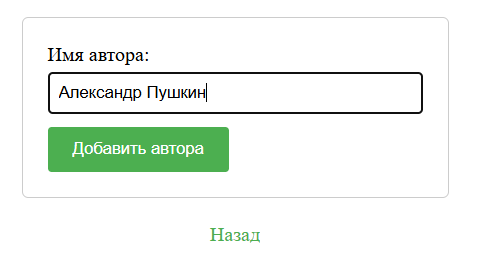
* Поле «Имя автора»**:**
  + Текстовое поле для ввода полного имени автора (обязательно для заполнения).
* Кнопка «Добавить автора»**:**
  + Кнопка отправки формы для добавления автора в базу данных.
* Кнопка «Назад»**:**
  + Ссылка для возврата на главную страницу.

*Проверки и ограничения*

1. Проверка имени**:**
   * Имя должно содержать одно или два слова и быть написано либо латинскими, либо кириллическими буквами, включая букву ё.

*Действия при отправке формы*

1. При успешном заполнении поля данные записываются в таблицу Authors.
2. Если запись была добавлена успешно, отображается сообщение «Автор успешно добавлен!».
3. Если происходит ошибка, на экран выводится сообщение с описанием ошибки.



## Добавление Книги

Эта страница позволяет пользователю добавить новую книгу в таблицу Books. Пользователь вводит информацию о книги, включая название, ISBN, автора, жанр, год публикации, цену, количество.

*Функционал страницы:*

* Форма добавления книги: Пользователь вводит название книги, ISBN, автора, жанр, год публикации, цену, количество.
* Валидация данных: Форма проводит проверки введённых данных на корректность.
* Запись данных в базу: при успешной проверке новая запись добавляется в таблицу Books**.**

*Элементы интерфейса:*

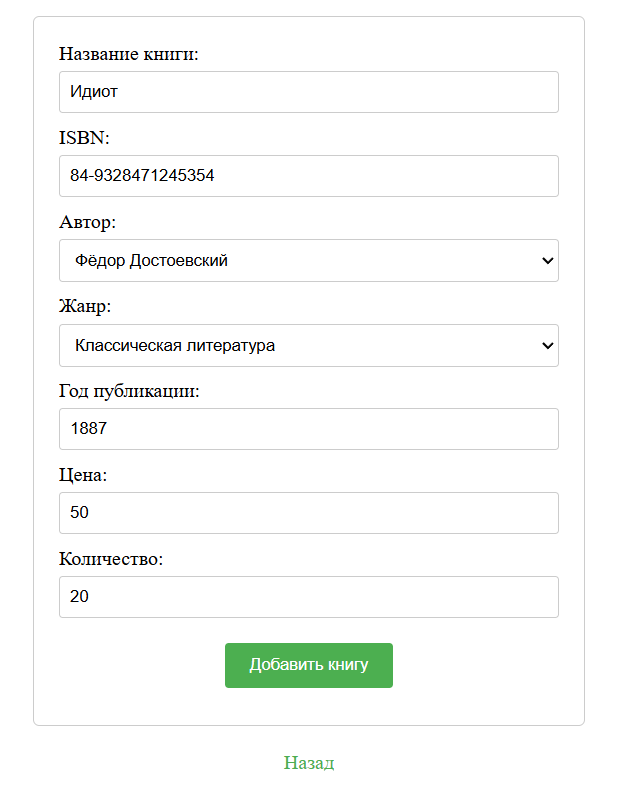
* + Поле «Название книги**»**:
    - Текстовое поле для ввода названия книги (обязательно для заполнения).
  + Поле «ISBN»**:**
    - Текстовое поле для ввода ISBN (обязательно для заполнения)
  + Выпадающий список «Автор»**:**
    - Список всех авторов, загруженных из таблицы Authors.
    - Если автор не выбран, пользователю будет показано сообщение с просьбой выбрать автора.
  + Выпадающий список «Жанр»**:**
    - Список всех жанров, загруженных из таблицы Genres.
    - Если жанр не выбран, пользователю будет показано сообщение с просьбой выбрать жанр.
  + Поле «Год публикации**»**:
    - Текстовое поле для ввода года публикации (обязательно для заполнения).
  + Поле «Цена**»**:
    - Текстовое поле для ввода цены (обязательно для заполнения).
  + Поле «Количество**»**:
    - Текстовое поле для ввода количества книг (обязательно для заполнения).
  + Кнопка «Добавить книгу»**:**
    - Отправляет форму для добавления книги в базу данных.
  + Кнопка «Назад»**:**
    - Ссылка для возврата на главную страницу.

*Проверки и ограничения:*

1. Проверка заполнения полей:
   * Поля «Название книги», «ISBN», «Автор», «Жанр», «Год публикации», «Цена», «Количество» обязательно должны быть заполнены. Причем поля «Год публикации», «Цена» и «Количество» могут содержать в себе лишь целые числа.

*Действия при отправке формы:*

1. Если все поля заполнены корректно, данные записываются в таблицу Books.
2. При успешном добавлении записи на странице выводится сообщение «Книга успешно добавлена!».
3. В случае, если какое-либо поле не заполнено, появляется сообщение с просьбой заполнить все поля.



## Добавление Стеллажа

Эта страница позволяет пользователю добавить новый стеллаж в таблицу Shelves. Пользователь заполняет название полки, вместимость полки. После успешного заполнения формы данные сохраняются в базе данных.

*Функционал страницы:*

1. Заполнение информации о стеллаже**:** Ввод названия стеллажа и вместимости стеллажа.
2. Добавление стеллажа**:** при корректном заполнении всех полей новый стеллаж добавляется в таблицу Shelves.

*Элементы интерфейса:*

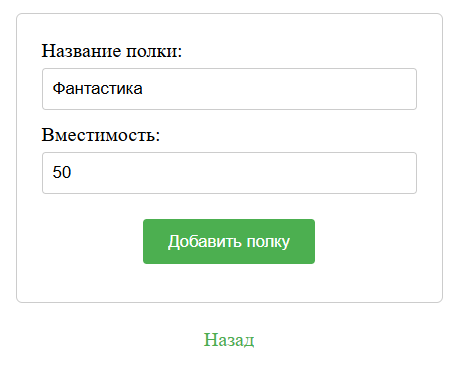
* + Поле «Название полки»**:**
    - Текстовое поле для ввода названия полки (обязательно для заполнения).
  + Поле «Вместимость»**:**
    - Текстовое поле для ввода максимального количества книг для данного стеллажа (обязательно для заполнения).
  + Кнопка «Добавить полку»**:**
    - Отправляет форму для добавления стеллажа в базу данных.
  + Кнопка **«**Назад»**:**
    - Ссылка для возврата на главную страницу.

*Проверки и ограничения:*

1. Проверка заполнения полей**:**
   * Поля «Название полки» и «Вместимость» являются обязательными. Если какое-либо из них не заполнено, появится сообщение с просьбой заполнить все поля. Причем поле «Название полки» может записываться только латиницей или кириллицей, а поле «Вместимость» может состоять только из целых чисел.

*Действия при отправке формы:*

1. Если все поля заполнены корректно, данные сохраняются в таблице Shelves.
2. При успешном добавлении записи выводится сообщение «Полка успешно добавлена!».
3. В случае, если какое-либо поле не заполнено, выводится соответствующее сообщение об ошибке.



## Добавление Расположения книги

Эта страница позволяет пользователю добавить новое расположение книги в таблицу BookLocations. Пользователь заполняет название книги, полку и количество. После успешного заполнения формы данные сохраняются в базе данных.

*Функционал страницы:*

1. Заполнение информации о расположении книги**:** Выбор книги, выбор стеллажа и количество книг.

2. Добавление расположения книги**:** при корректном заполнении всех полей новое расположение добавляется в таблицу BookLocations.

*Элементы интерфейса:*

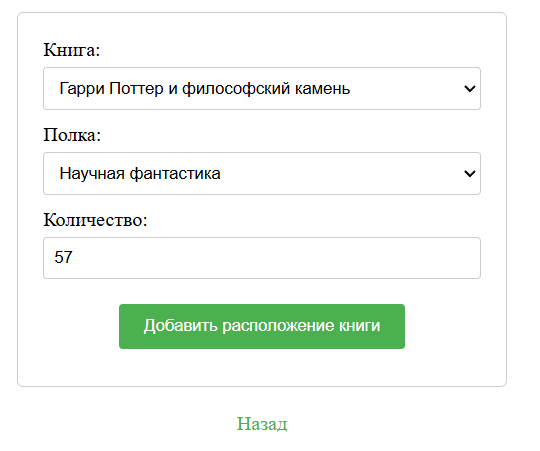
* + Выпадающий список «Книги»**:**
    - Список всех книг, загруженных из таблицы Books.
    - Если книга не выбрана, пользователю будет показано сообщение с просьбой выбрать книгу.
  + Выпадающий список «Полка»**:**
    - Список всех стеллажей, загруженных из таблицы Shelves.
    - Если стеллаж не выбран, пользователю будет показано сообщение с просьбой выбрать стеллаж.
  + Поле «Количество»**:**
    - Текстовое поле для ввода количества книг для данной полки (обязательно для заполнения).
  + Кнопка «Добавить расположение книги»**:**
    - Отправляет форму для добавления расположения в базу данных.
  + Кнопка **«**Назад»**:**
    - Ссылка для возврата на главную страницу.

*Проверки и ограничения:*

1. Проверка заполнения полей**:**
   * Поля «Книги», «Полка» и «Количество» являются обязательными. Если какое-либо из них не заполнено, появится сообщение с просьбой заполнить все поля. Причем поле «Количество» может состоять только из целых чисел.

*Действия при отправке формы:*

1. Если все поля заполнены корректно, данные сохраняются в таблице BookLocations.
2. При успешном добавлении записи выводится сообщение «Расположение книги успешно добавлено!».
3. В случае, если какое-либо поле не заполнено, выводится соответствующее сообщение об ошибке.



## Добавление Сотрудника

Эта страница позволяет пользователю добавить нового сотрудника в таблицу Employees. Пользователь заполняет полное имя, должность и дату найма. После успешного заполнения формы данные сохраняются в базе данных.

*Функционал страницы:*

1. Заполнение информации о сотруднике**:** Полное имя, должность и дата найма.

2. Добавление сотрудника**:** при корректном заполнении всех полей новый сотрудник добавляется в таблицу Employees.

*Элементы интерфейса:*

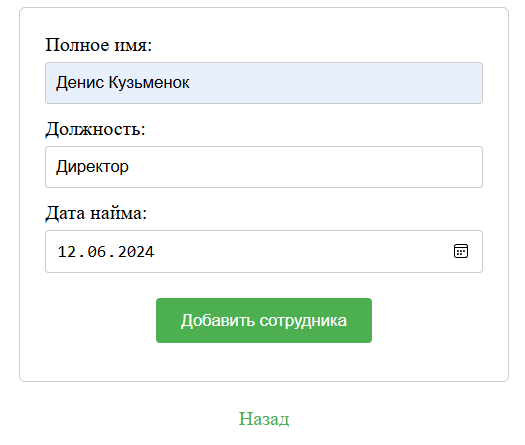
* + Поле «Полное имя»**:**
    - Текстовое поле для ввода ФИО сотрудника (обязательно для заполнения).
  + Поле «Должность»**:**
    - Текстовое поле для ввода должности сотрудника (обязательно для заполнения).
  + Поле «Дата найма»**:**
    - Поле типа date, позволяет вводить дату найма в формате ГГГГ-ММ-ДД (обязательно для заполнения).
  + Кнопка «Добавить сотрудника»**:**
    - Отправляет форму для добавления сотрудника в базу данных.
  + Кнопка **«**Назад»**:**
    - Ссылка для возврата на главную страницу.

*Проверки и ограничения:*

1. Проверка заполнения полей**:**
   * Поля «Полное имя», «Должность» и «Дата найма» являются обязательными. Если какое-либо из них не заполнено, появится сообщение с просьбой заполнить все поля. Причем поле «Полное имя» может записываться только латиницей или кириллицей.

*Действия при отправке формы:*

1. Если все поля заполнены корректно, данные сохраняются в таблице Employees.
2. При успешном добавлении записи выводится сообщение «Сотрудник успешно добавлен!».
3. В случае, если какое-либо поле не заполнено, выводится соответствующее сообщение об ошибке.



## Запросы

Эта страница позволяет пользователю выполнять три запроса к базе данных: просмотр книг, расположенных на определенном стеллаже, вывод общего количества книг по каждому жанру, вывод книг определенного автора.

*Функционал страницы*

1. Запрос 1: Книги, расположенные на определённом стеллаже — выводит список книг, расположенных на определённом стеллаже.
2. Запрос 2: Количество книг по каждому жанру — отображает таблицу с жанрами и количеством книг в данном жанре.
3. Запрос 3: Книги определенного автора — показывает книги выбранного автора.

*Элементы интерфейса*

1. Форма для Запроса 1: Книги, расположенные на определённом стеллаже**:**
   * Поле выбора «Выберите стеллаж»: выпадающий список, содержащий названия всех стеллажей, извлекаемых из таблицы Shelves.
   * Кнопка «Показать книги»: отправляет запрос для вывода списка книг, расположенных на указанном стеллаже.
2. Результаты для Запроса 1**:**
   * Если стеллаж выбран, результаты выводятся в виде таблицы, где указывается информация о книге (название, автор, жанр, количество), находящиеся на указанном стеллаже.
3. Таблица для Запроса 2: Количество книг по каждому жанру**:**
   * Отображает жанры и количество книг данного жанра.
   * Таблица состоит из двух колонок: «Жанр» и «Количество книг».
4. Форма для Запроса 3: Книги определенного автора**:**
   * Поле выбора «Выберите автора»: выпадающий список, содержащий всех авторов из таблицы Authors.
   * Кнопка «Показать книги»: выполняет запрос для отображения книг выбранного автора.
5. Результаты для Запроса 3**:**
   * При выборе автора выводится список книг с названием и жанром.
6. Кнопка «Назад»**:**
   * Ссылка для возврата на главную страницу.

*Обработка запросов и вывод данных*

### **Запрос 1: Книги, расположенные на определённом стеллаже**

**Описание:**

* Если пользователь выбирает стеллаж, выполняется запрос, который:
  + Объединяет сразу 5 таблиц (Genres, Authors, Books, BookLocations и Shelves), для получения информации о том, какие книги расположены на определенном стеллаже.
  + Результат выводится как таблица со столбцами название книги, автор, жанр и количество книг на данном стеллаже.

### **Запрос 2: Количество книг по каждому жанру**

**Описание:**

* Запрос выполняется автоматически и подсчитывает количество книг для каждого жанра.
* Используется LEFT JOIN между таблицами Genres и Books, чтобы отобразить все книги, включая те, где не указан жанр (если таковые имеются).
* Результат отображается в виде таблицы с названием жанра и количеством книг.

### **Запрос 3: Книги определенного автора**

**Описание:**

* Если пользователь выбирает автора, выполняется запрос, который:
  + Извлекает список книг определённого автора из таблицы Books, связывая их с таблицей Genres.
  + Выводит данные о книгах данного автора, включая название книги и жанр.

*Проверки и ограничения*

* Все формы требуют выбора значения (стеллаж или автор) из выпадающих списков.
* Если поле не выбрано, запрос не выполняется, и данные не выводятся.

# SQL-запросы

*Запрос 1: Книги, расположенные на определённом стеллаже*

Формулировка: получить список книг, которые находятся на выбранном стеллаже, с указанием их названия, автора, жанра и количеством.

SELECT b.Title, a.AuthorName, g.GenreName, s.ShelfName, bl.Quantity

FROM Books b

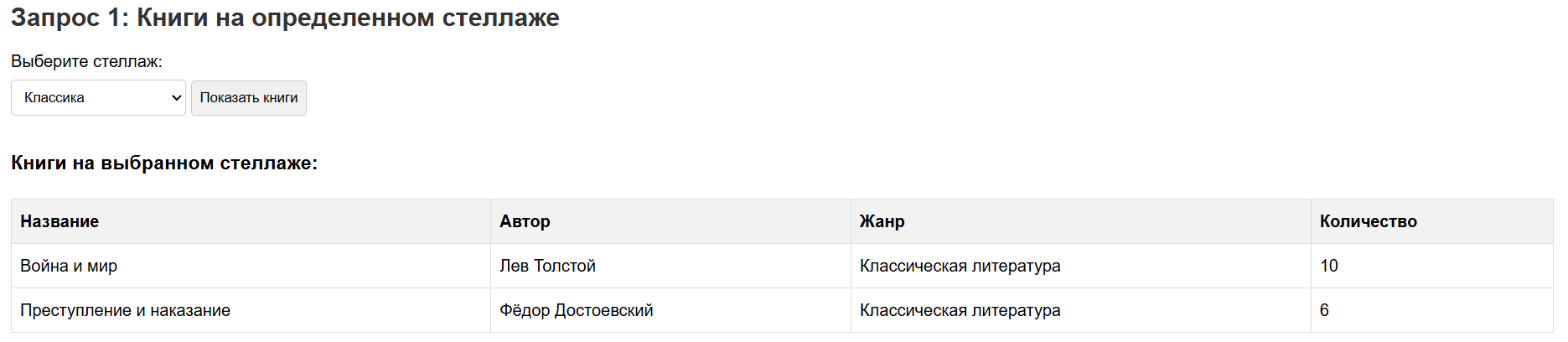
JOIN Authors a ON b.AuthorID = a.AuthorID

JOIN Genres g ON b.GenreID = g.GenreID

JOIN BookLocations bl ON b.BookID = bl.BookID

JOIN Shelves s ON bl.ShelfID = s.ShelfID

WHERE bl.ShelfID = ?



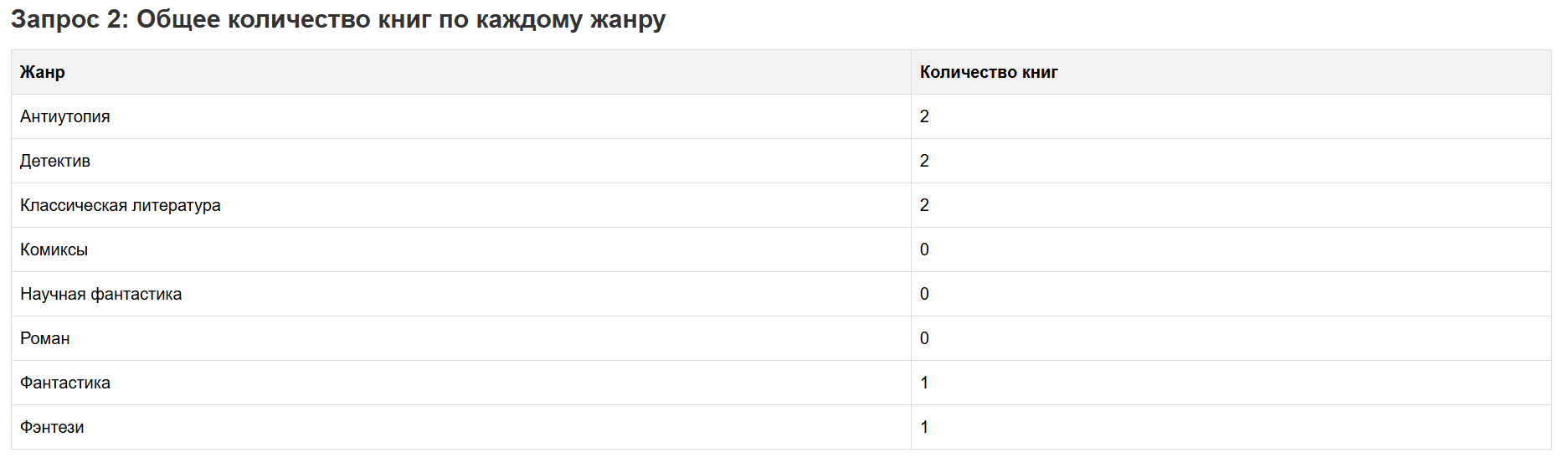
*Запрос 2: Количество книг по каждому жанру*

Формулировка: определить количество книг каждого жанра.

SELECT g.GenreName, COUNT(b.BookID) AS TotalBooks

FROM Genres g

LEFT JOIN Books b ON g.GenreID = b.GenreID

GROUP BY g.GenreName 

*Запрос 3: Книги определенного автора*

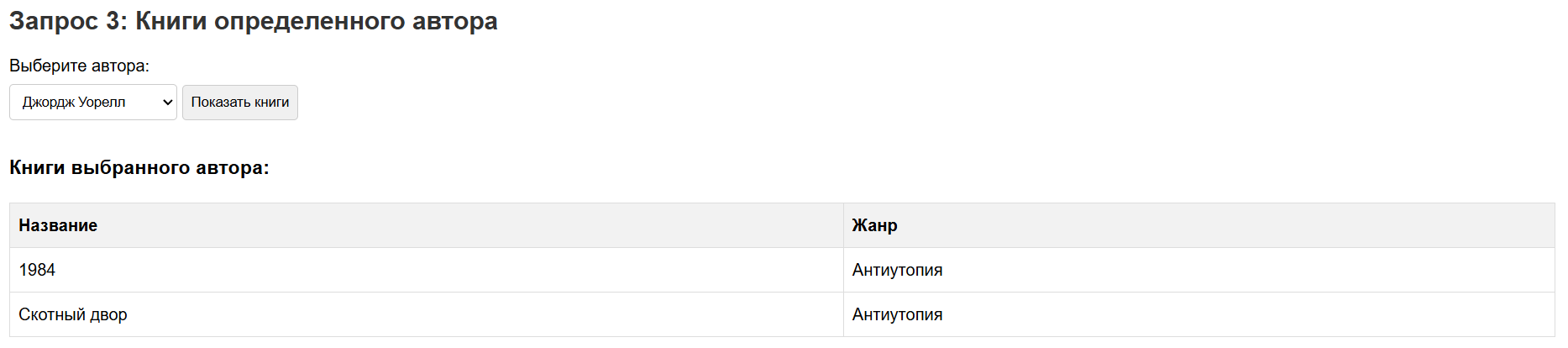
Формулировка**:** получить список книг выбранного автора с указанием их названия и жанра.

SELECT b.Title, g.GenreName

FROM Books b

JOIN Genres g ON b.GenreID = g.GenreID

WHERE b.AuthorID = ?



# Листинг

* 1. SQL-скрипты для создания таблиц, вставки данных Создание таблиц

CREATE TABLE Genres (

GenreID INT AUTO\_INCREMENT PRIMARY KEY,

GenreName VARCHAR(100) UNIQUE NOT NULL

);

CREATE TABLE Authors (

AuthorID INT AUTO\_INCREMENT PRIMARY KEY,

AuthorName VARCHAR(150) UNIQUE NOT NULL

);

CREATE TABLE Books (

BookID INT AUTO\_INCREMENT PRIMARY KEY,

Title VARCHAR(100) NOT NULL,

ISBN VARCHAR(20) UNIQUE,

AuthorID INT,

GenreID INT NOT NULL,

PublicationYear INT,

Price DECIMAL(10, 2) NOT NULL,

Quantity INT NOT NULL DEFAULT 1,

FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),

FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)

);

CREATE TABLE Shelves (

ShelfID INT AUTO\_INCREMENT PRIMARY KEY,

ShelfName VARCHAR(100) NOT NULL,

Capacity INT NOT NULL

);

CREATE TABLE BookLocations (

BookLocationID INT AUTO\_INCREMENT PRIMARY KEY,

BookID INT NOT NULL,

ShelfID INT NOT NULL,

Quantity INT NOT NULL DEFAULT 1,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (ShelfID) REFERENCES Shelves(ShelfID),

UNIQUE KEY (BookID, ShelfID)

);

CREATE TABLE Employees (

EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

FullName VARCHAR(150) NOT NULL,

Position VARCHAR(100) NOT NULL,

HireDate DATE NOT NULL

);

В**ставка данных**

INSERT INTO Genres (GenreName) VALUES

('Фантастика'), ('Фэнтези'), ('Детектив'), ('Научная фантастика'), ('Роман'), ('Классическая литература');

INSERT INTO Authors (AuthorName) VALUES

('Джон Толкиен'), ('Джоан Роулинг'), ('Агата Кристи'), ('Артур Конан Дойл'), ('Лев Толстой'), ('Фёдор Достоевский');

INSERT INTO Books (Title, ISBN, AuthorID, GenreID, PublicationYear, Price, Quantity) VALUES

('Властелин колец: Братство кольца', '978-0618002255', 1, 2, 1954, 15.99, 10),

('Гарри Поттер и философский камень', '978-0747532699', 2, 1, 1997, 12.99, 15),

('И смерть была незваной гостьей', '978-0007466708', 3, 3, 1953, 9.99, 5),

('Записки о Шерлоке Холмсе', '978-0553213572', 4, 3, 1892, 11.99, 8),

('Война и мир', '978-0141439518', 5, 6, 1869, 19.99, 20),

('Преступление и наказание', '978-0141439594', 6, 6, 1866, 14.99, 12);

INSERT INTO Shelves (ShelfName, Capacity) VALUES

('Фэнтези', 50), ('Классика', 30), ('Детективы', 25), ('Научная фантастика', 20);

INSERT INTO BookLocations (BookID, ShelfID, Quantity) VALUES

(1, 1, 5), (2, 1, 8), (3, 3, 5), (4, 3, 3), (5, 2, 10), (6, 2, 6), (7, 4, 7);

INSERT INTO Employees (FullName, Position, HireDate) VALUES

('Иван Иванов', 'Менеджер', '2022-01-15'),

('Мария Петрова', 'Продавец', '2023-03-10'),

('Алексей Смирнов', 'Кладовщик', '2022-06-20');

* 1. PHP-скрипты

db.php

<?php

$servername = 'mysql-8.0';

$dbname = 'book\_shop';

$username = 'root';

$password = '';

**try** {

$pdo = **new** PDO("mysql:host=$servername;dbname=$dbname;charset=utf8", $username, $password);

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

} **catch** (PDOException $e) {

**echo** "Ошибка подключения: " . $e->getMessage();

}

?>

index.php

<!DOCTYPE html>

<html>

<head>

<title>Book Shop</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>Книжный магазин - Управление базой данных</h1>

<ul>

<li><a href="view\_table.php?table=Authors">Просмотр таблиц</a></li>

<li><a href="add\_genre.php">Добавить жанр</a></li>

<li><a href="add\_author.php">Добавить автора</a></li>

<li><a href="add\_book.php">Добавить книгу</a></li>

<li><a href="add\_shelves.php">Добавить стелаж для книги</a></li>

<li><a href="add\_booklocation.php">Добавить расположение книги</a></li>

<li><a href="add\_employees.php">Добавить сотрудника</a></li>

<li><a href="queries.php">Запросы к таблицам</a></li>

</ul>

</body>

</html>

view\_table.php

<?php

**include** 'db.php';

$table = isset($\_GET['table']) ? $\_GET['table'] : 'Genres';

**echo** "<h2 style='text-align: center; color: navy;'>Содержимое таблицы: " . htmlspecialchars($table) . "</h2>";

**try** {

$stmt = $pdo->prepare("SHOW TABLES LIKE ?");

$stmt->execute(**array**($table));

**if** ($stmt->rowCount() == **0**) {

**echo** "<p style='color: red; text-align: center;'>Таблица " . htmlspecialchars($table) . " не найдена в базе данных.</p>";

} **else** {

$stmt = $pdo->query("SELECT \* FROM " . $table);

**echo** "<table style='width: 100%; border-collapse: collapse; margin: 20px auto;'>";

**echo** "<tr style='background-color: #f2f2f2;'>";

**for** ($i = **0**; $i < $stmt->columnCount(); $i++) {

$col = $stmt->getColumnMeta($i);

**echo** "<th style='padding: 8px; border: 1px solid #ddd; text-align: left;'>" . htmlspecialchars($col['name']) . "</th>";

}

**echo** "</tr>";

**while** ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

**echo** "<tr>";

**foreach** ($row **as** $value) {

**echo** "<td style='padding: 8px; border: 1px solid #ddd;'>" . htmlspecialchars($value) . "</td>";

}

**echo** "</tr>";

}

**echo** "</table>";

}

} **catch** (PDOException $e) {

**echo** "<p style='color: red; text-align: center;'>Ошибка: " . htmlspecialchars($e->getMessage()) . "</p>";

}

?>

<form method="get" style="text-align: center; margin-top: 20px;">

<label for="table" style="margin-right: 10px;">Выберите таблицу:</label>

<select name="table" id="table" style="padding: 8px; border: 1px solid #ccc; border-radius: 3px;">

<option value="Genres" <?php **if** ($table == 'Genres') **echo** 'selected'; ?>>Жанры книг</option>

<option value="Authors" <?php **if** ($table == 'Authors') **echo** 'selected'; ?>>Авторы</option>

<option value="Books" <?php **if** ($table == 'Books') **echo** 'selected'; ?>>Книги</option>

<option value="Shelves" <?php **if** ($table == 'Shelves') **echo** 'selected'; ?>>Стелажи книг</option>

<option value="BookLocations" <?php **if** ($table == 'BookLocations') **echo** 'selected'; ?>>Расположение книг</option>

<option value="Employees" <?php **if** ($table == 'Employees') **echo** 'selected'; ?>>Сотрудники</option>

</select>

<input type="submit" value="Просмотр" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer; margin-left: 10px;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration:none; color: #4CAF50;">Назад</a>

add\_genre.php

<?php

**include** 'db.php';

$genres = $pdo->query("SELECT GenreID, GenreName FROM Genres")->fetchAll(PDO::FETCH\_ASSOC);

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$genreName = trim($\_POST['genreName']);

**if** (!**empty**($genreName))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO Genres (GenreName) VALUES (?)");

$stmt->execute(**array**($genreName));

**echo** "<p style='color: green;'>Жанр успешно добавлен!</p>";

}

**catch** (PDOException $e)

{

**if** ($e->getCode() == **23000**)

{

**echo** "<p style='color: red;'>Жанр с таким названием уже существует.</p>";

} **else**

{

**echo** "<p style='color: red;'>Ошибка при добавлении жанра: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

}

**else**

{

**echo** "<p style='color: red;'>Поле 'Название жанра' не может быть пустым.</p>";

}

}

?>

<form method="post" style="width: 300px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="genreName" style="display: block; margin-bottom: 5px;">Название жанра:</label>

<input type="text" name="genreName" id="genreName" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить жанр" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration: none; color: #4CAF50;">Назад</a>

add\_author.php

<?php

**include** 'db.php';

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$authorName = trim($\_POST['authorName']);

**if** (!**empty**($authorName) && preg\_match('/^[A-Za-zА-Яа-яЁё]+(?:\s[A-Za-zА-Яа-яЁё]+)?$/u', $authorName))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO Authors (AuthorName) VALUES (?)");

$stmt->execute(**array**($authorName));

**echo** "<p style='color:green;'>Автор успешно добавлен!</p>";

}

**catch** (PDOException $e)

{

**if** ($e->getCode() == **23000**)

{

**echo** "<p style='color:red;'>Автор с таким именем уже существует.</p>";

}

**else**

{

**echo** "<p style='color:red;'>Ошибка при добавлении автора: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

}

**else**

{

**echo** "<p style='color:red;'>Поле 'Имя автора' не может быть пустым и должно содержать только буквы. Допускается одно или два слова.</p>";

}

}

?>

<form method="post" style="width:300px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="authorName" style="display:block; margin-bottom: 5px;">Имя автора:</label>

<input type="text" name="authorName" id="authorName" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить автора" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration:none; color: #4CAF50;">Назад</a>

add\_book.php

<?php

**include** 'db.php';

$authors = $pdo->query("SELECT AuthorID, AuthorName FROM Authors")->fetchAll(PDO::FETCH\_ASSOC);

$genres = $pdo->query("SELECT GenreID, GenreName FROM Genres")->fetchAll(PDO::FETCH\_ASSOC);

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$title = trim($\_POST['title']);

$isbn = trim($\_POST['isbn']);

$authorId = $\_POST['authorId'];

$genreId = $\_POST['genreId'];

$publicationYear = $\_POST['publicationYear'];

$price = $\_POST['price'];

$quantity = $\_POST['quantity'];

**if** (!**empty**($title) && !**empty**($isbn) && !**empty**($authorId) && !**empty**($genreId) && !**empty**($publicationYear) && !**empty**($price) && !**empty**($quantity) && preg\_match('/^[0-9]+$/',$quantity) && preg\_match('/^[0-9]+$/',$publicationYear) && preg\_match('/^[0-9.]+$/',$price))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO Books (Title, ISBN, AuthorID, GenreID, PublicationYear, Price, Quantity) VALUES (?, ?, ?, ?, ?, ?, ?)");

$stmt->execute([$title, $isbn, $authorId, $genreId, $publicationYear, $price, $quantity]);

**echo** "<p style='color: green;'>Книга успешно добавлена!</p>";

}

**catch** (PDOException $e)

{

**if** ($e->getCode() == **23000**)

{

**echo** "<p style='color: red;'>Ошибка: Нарушение уникальности (ISBN) или внешний ключ.</p>";

}

**else**

{

**echo** "<p style='color: red;'>Ошибка при добавлении книги: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

}

**else**

{

**echo** "<p style='color: red;'>Пожалуйста, заполните все поля корректно. Количество, год и цена должны быть числами.</p>";

}

}

?>

<form method="post" style="width: 400px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="title" style="display: block; margin-bottom: 5px;">Название книги:</label>

<input type="text" name="title" id="title" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="isbn" style="display: block; margin-bottom: 5px;">ISBN:</label>

<input type="text" name="isbn" id="isbn" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="authorId" style="display: block; margin-bottom: 5px;">Автор:</label>

<select name="authorId" id="authorId" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<option value="">Выберите автора</option>

<?php **foreach** ($authors **as** $author): ?>

<option value="<?php **echo** $author['AuthorID']; ?>">

<?php **echo** htmlspecialchars($author['AuthorName']); ?>

</option>

<?php **endforeach**; ?>

</select>

<label for="genreId" style="display: block; margin-bottom: 5px;">Жанр:</label>

<select name="genreId" id="genreId" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<option value="">Выберите жанр</option>

<?php **foreach** ($genres **as** $genre): ?>

<option value="<?php **echo** $genre['GenreID']; ?>">

<?php **echo** htmlspecialchars($genre['GenreName']); ?>

</option>

<?php **endforeach**; ?>

</select>

<label for="publicationYear" style="display: block; margin-bottom: 5px;">Год публикации:</label>

<input type="number" name="publicationYear" id="publicationYear" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="price" style="display: block; margin-bottom: 5px;">Цена:</label>

<input type="number" step="0.01" name="price" id="price" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="quantity" style="display: block; margin-bottom: 5px;">Количество:</label>

<input type="number" name="quantity" id="quantity" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить книгу" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer; display: block; margin: 10px auto;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration: none; color: #4CAF50;">Назад</a>

add\_shelves.php

<?php

**include** 'db.php';

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$shelfName = trim($\_POST['shelfName']);

$capacity = trim($\_POST['capacity']);

**if** (!**empty**($shelfName) && !**empty**($capacity) && preg\_match('/^[A-Za-zА-Яа-яЁё\s]+$/u', $shelfName) && preg\_match('/^[0-9]+$/', $capacity))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO Shelves (ShelfName, Capacity) VALUES (?, ?)");

$stmt->execute([$shelfName, $capacity]);

**echo** "<p style='color: green;'>Полка успешно добавлена!</p>";

}

**catch** (PDOException $e)

{

**echo** "<p style='color: red;'>Ошибка при добавлении полки: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

**else**

{

**echo** "<p style='color: red;'>Поле 'Название полки' не может быть пустым и должно содержать только буквы и пробелы. 'Вместимость' должна быть числом.</p>";

}

}

?>

<form method="post" style="width: 300px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="shelfName" style="display: block; margin-bottom: 5px;">Название полки:</label>

<input type="text" name="shelfName" id="shelfName" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="capacity" style="display: block; margin-bottom: 5px;">Вместимость:</label>

<input type="number" name="capacity" id="capacity" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить полку" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer; display: block; margin: 10px auto;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration: none; color: #4CAF50;">Назад</a>

add\_booklocation.php

<?php

**include** 'db.php';

$books = $pdo->query("SELECT BookID, Title FROM Books")->fetchAll(PDO::FETCH\_ASSOC);

$shelves = $pdo->query("SELECT ShelfID, ShelfName FROM Shelves")->fetchAll(PDO::FETCH\_ASSOC);

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$bookId = $\_POST['bookId'];

$shelfId = $\_POST['shelfId'];

$quantity = $\_POST['quantity'];

**if** (!**empty**($bookId) && !**empty**($shelfId) && !**empty**($quantity) && preg\_match('/^[0-9]+$/', $quantity))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO BookLocations (BookID, ShelfID, Quantity) VALUES (?, ?, ?)");

$stmt->execute([$bookId, $shelfId, $quantity]);

**echo** "<p style='color: green;'>Расположение книги успешно добавлено!</p>";

}

**catch** (PDOException $e)

{

**if** ($e->getCode() == **23000**)

{

**echo** "<p style='color: red;'>Ошибка: Нарушение уникальности (BookID, ShelfID) или внешний ключ.</p>";

}

**else**

{

**echo** "<p style='color: red;'>Ошибка при добавлении расположения книги: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

}

**else**

{

**echo** "<p style='color: red;'>Пожалуйста, заполните все поля корректно. Количество должно быть числом.</p>";

}

}

?>

<form method="post" style="width: 350px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="bookId" style="display: block; margin-bottom: 5px;">Книга:</label>

<select name="bookId" id="bookId" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<option value="">Выберите книгу</option>

<?php **foreach** ($books **as** $book): ?>

<option value="<?php **echo** $book['BookID']; ?>">

<?php **echo** htmlspecialchars($book['Title']); ?>

</option>

<?php **endforeach**; ?>

</select>

<label for="shelfId" style="display: block; margin-bottom: 5px;">Полка:</label>

<select name="shelfId" id="shelfId" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<option value="">Выберите полку</option>

<?php **foreach** ($shelves **as** $shelf): ?>

<option value="<?php **echo** $shelf['ShelfID']; ?>">

<?php **echo** htmlspecialchars($shelf['ShelfName']); ?>

</option>

<?php **endforeach**; ?>

</select>

<label for="quantity" style="display: block; margin-bottom: 5px;">Количество:</label>

<input type="number" name="quantity" id="quantity" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить расположение книги" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer; display: block; margin: 10px auto;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration: none; color: #4CAF50;">Назад</a>

add\_employees.php

<?php

**include** 'db.php';

**if** ($\_SERVER['REQUEST\_METHOD'] == 'POST')

{

$fullName = trim($\_POST['fullName']);

$position = trim($\_POST['position']);

$hireDate = $\_POST['hireDate'];

**if** (!**empty**($fullName) && !**empty**($position) && !**empty**($hireDate) && preg\_match('/^[A-Za-zА-Яа-яЁё]+ [A-Za-zА-Яа-яЁё]+$/u', $fullName))

{

**try**

{

$stmt = $pdo->prepare("INSERT INTO Employees (FullName, Position, HireDate) VALUES (?, ?, ?)");

$stmt->execute(**array**($fullName, $position, $hireDate));

**echo** "<p style='color: green;'>Сотрудник успешно добавлен!</p>";

}

**catch** (PDOException $e)

{

**echo** "<p style='color: red;'>Ошибка при добавлении сотрудника: " . htmlspecialchars($e->getMessage()) . "</p>";

}

}

**else**

{

**echo** "<p style='color: red;'>Пожалуйста, заполните все поля корректно. Полное имя должно состоять из двух слов.</p>";

}

}

?>

<form method="post" style="width: 350px; margin: 20px auto; padding: 20px; border: 1px solid #ccc; border-radius: 5px;">

<label for="fullName" style="display: block; margin-bottom: 5px;">Полное имя:</label>

<input type="text" name="fullName" id="fullName" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="position" style="display: block; margin-bottom: 5px;">Должность:</label>

<input type="text" name="position" id="position" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<label for="hireDate" style="display: block; margin-bottom: 5px;">Дата найма:</label>

<input type="date" name="hireDate" id="hireDate" required style="width: 100%; padding: 8px; margin-bottom: 10px; border: 1px solid #ccc; border-radius: 3px;">

<input type="submit" value="Добавить сотрудника" style="background-color: #4CAF50; color: white; padding: 10px 20px; border: none; border-radius: 3px; cursor: pointer; display: block; margin: 10px auto;">

</form>

<a href="index.php" style="display: block; text-align: center; margin-top: 20px; text-decoration: none; color: #4CAF50;">Назад</a>

queries.php

<?

**include** 'db.php';

$booksOnShelf = [];

**if** (isset($\_POST['shelf\_id']))

{

$shelfId = intval($\_POST['shelf\_id']);

**try**

{

$stmt = $pdo->prepare("

SELECT b.Title, a.AuthorName, g.GenreName, s.ShelfName, bl.Quantity

FROM Books b

JOIN Authors a ON b.AuthorID = a.AuthorID

JOIN Genres g ON b.GenreID = g.GenreID

JOIN BookLocations bl ON b.BookID = bl.BookID

JOIN Shelves s ON bl.ShelfID = s.ShelfID

WHERE bl.ShelfID = ?

");

$stmt->execute([$shelfId]);

$booksOnShelf = $stmt->fetchAll();

}

**catch** (PDOException $e)

{

error\_log("Ошибка в запросе 1: " . $e->getMessage() . "**\n**" . $e->getTraceAsString(), **3**, "db\_errors.log");

**echo** "<p style='color:red;'>Ошибка при выполнении запроса 1.</p>";

}

}

$booksPerGenre = [];

**try**

{

$stmt = $pdo->query("

SELECT g.GenreName, COUNT(b.BookID) AS TotalBooks

FROM Genres g

LEFT JOIN Books b ON g.GenreID = b.GenreID

GROUP BY g.GenreName

");

$booksPerGenre = $stmt->fetchAll();

}

**catch** (PDOException $e)

{

error\_log("Ошибка в запросе 2: " . $e->getMessage() . "**\n**" . $e->getTraceAsString(), **3**, "db\_errors.log");

**echo** "<p style='color:red;'>Ошибка при выполнении запроса 2.</p>";

}

$booksByAuthor = [];

**if** (isset($\_POST['author\_id']))

{

$authorId = intval($\_POST['author\_id']);

**try**

{

$stmt = $pdo->prepare("

SELECT b.Title, g.GenreName

FROM Books b

JOIN Genres g ON b.GenreID = g.GenreID

WHERE b.AuthorID = ?

");

$stmt->execute([$authorId]);

$booksByAuthor = $stmt->fetchAll();

}

**catch** (PDOException $e)

{

error\_log("Ошибка в запросе 3: " . $e->getMessage() . "**\n**" . $e->getTraceAsString(), **3**, "db\_errors.log");

**echo** "<p style='color:red;'>Ошибка при выполнении запроса 3.</p>";

}

}

$shelves = $pdo->query("SELECT ShelfID, ShelfName FROM Shelves")->fetchAll();

$authors = $pdo->query("SELECT AuthorID, AuthorName FROM Authors")->fetchAll();

?>

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Запросы к базе данных книг</title>

<style>

body {

font-family: sans-serif;

line-height: 1.6;

margin: 20px;

}

h1, h2 {

color: #333;

}

h2 {

margin-bottom: 10px;

}

form {

margin-bottom: 20px;

}

label {

display: block;

margin-bottom: 5px;

}

select, input[type="submit"] {

padding: 8px;

border: 1px solid #ccc;

border-radius: 4px;

margin-bottom: 10px;

}

table {

width: 100%;

border-collapse: collapse;

margin-bottom: 20px;

}

th, td {

border: 1px solid #ddd;

padding: 8px;

text-align: left;

}

th {

background-color: #f2f2f2;

}

.error {

color: red;

font-weight: bold;

}

</style>

</head>

<body>

<h1>Запросы к базе данных</h1>

<h2>Запрос 1: Книги на определенном стеллаже</h2>

<form method="post">

<label for="shelf\_id">Выберите стеллаж:</label>

<select id="shelf\_id" name="shelf\_id">

<option value="">Выберите стеллаж</option>

<?php **foreach** ($shelves **as** $shelf): ?>

<option value="<?php **echo** $shelf['ShelfID']; ?>">

<?php **echo** htmlspecialchars($shelf['ShelfName']); ?>

</option>

<?php **endforeach**; ?>

</select>

<input type="submit" value="Показать книги">

</form>

<?php **if** (!**empty**($booksOnShelf)): ?>

<h3>Книги на выбранном стеллаже:</h3>

<table>

<tr><th>Название</th><th>Автор</th><th>Жанр</th><th>Количество</th></tr>

<?php **foreach** ($booksOnShelf **as** $book): ?>

<tr>

<td><?php **echo** htmlspecialchars($book['Title']); ?></td>

<td><?php **echo** htmlspecialchars($book['AuthorName']); ?></td>

<td><?php **echo** htmlspecialchars($book['GenreName']); ?></td>

<td><?php **echo** htmlspecialchars($book['Quantity']); ?></td>

</tr>

<?php **endforeach**; ?>

</table>

<?php **endif**; ?>

<h2>Запрос 2: Общее количество книг по каждому жанру</h2>

<table>

<tr><th>Жанр</th><th>Количество книг</th></tr>

<?php **foreach** ($booksPerGenre **as** $genre): ?>

<tr>

<td><?php **echo** htmlspecialchars($genre['GenreName']); ?></td>

<td><?php **echo** htmlspecialchars($genre['TotalBooks']); ?></td>

</tr>

<?php **endforeach**; ?>

</table>

<h2>Запрос 3: Книги определенного автора</h2>

<form method="post">

<label for="author\_id">Выберите автора:</label>

<select id="author\_id" name="author\_id">

<option value="">Выберите автора</option>

<?php **foreach** ($authors **as** $author): ?>

<option value="<?php **echo** $author['AuthorID']; ?>">

<?php **echo** htmlspecialchars($author['AuthorName']); ?>

</option>

<?php **endforeach**; ?>

</select>

<input type="submit" value="Показать книги">

</form>

<?php **if** (!**empty**($booksByAuthor)): ?>

<h3>Книги выбранного автора:</h3>

<table>

<tr><th>Название</th><th>Жанр</th></tr>

<?php **foreach** ($booksByAuthor **as** $book): ?>

<tr>

<td><?php **echo** htmlspecialchars($book['Title']); ?></td>

<td><?php **echo** htmlspecialchars($book['GenreName']); ?></td>

</tr>

<?php **endforeach**; ?>

</table>

<?php **endif**; ?>

</body>

</html>

styles.css

body {

**font-family**: **sans-serif**;

}

h1 {

**text-align**: **center**;

**color**: navy;

}

ul {

**list-style-type**: **none**;

**padding**: **0**;

**margin**: **20px**;

}

li {

**display**: **inline**-**block**;

**margin-right**: **20px**;

}

a {

**text-decoration**: **none**;

**color**: darkblue;