**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 4

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5

1.1 Анализ предметной области 5

1.2 Постановка задачи 5

2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 7

2.1 Построение диаграммы «сущность-связь» в нотации П.Чена 7

2.2 Построение модели основанной на ключах 9

2.3 Построение полной атрибутивной модели 10

3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ 12

3.1 Выбор аппаратной и программной платформы для реализации БД 12

3.2 Реализация базы данных 13

3.3 Тестирование базы данных 13

3.4 Разграничение прав доступа 17

3.5 Расчет информационных параметров базы данных 17

4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 19

4.1 Обоснование выбора языка программирования 20

4.2 Разработка интерфейса пользователя 20

4.3 Алгоритм работы каждого из модулей 25

4.4 Тестирование работы приложения 26

ЗАКЛЮЧЕНИЕ 31

СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ 32

ПРИЛОЖЕНИЕ А SQL скрипт создания базы данных 33

ПРИЛОЖЕНИЕ Б Код клиентского приложения 35

**ВВЕДЕНИЕ**

На сегодняшний день, трудно представить любую информационную систему, которая не использует базы данных. Их развитие было очень тесно связано с развитием автоматизированной обработки данных и увеличением количества различных категорий пользователей, что послужило к построению более мощных СУБД, имеющие средства защиты информации, высокую отказоустойчивость и производительность.

Целью данного курсового проекта является проектирование и реализация базы данных «Библиотеки научно-исследовательского института». Для реализации данной работы, были проведены следующие этапы:

* анализ предметной области (определяются пользовательские системы, функциональные требования системы, выделяются основные сущности разрабатываемой системы и отношения между ними);
* разработка логической схемы базы данных (построение ER-диаграммы, создание модели, построенной на ключах, и нормализация отношений до третьей или четвертой формы);
* реализация базы данных (обоснование выбора СУБД, разработка физической схемы базы данных, реализация и тестирование базы данных, запросов к ней, разграничение прав доступа);
* исследование информационных параметров базы данных (расчет значений некоторых параметров БД и приложения в целом);
* разработка клиентского приложения (обоснование выбора языка программирования, разработка клиентского приложения, описание интерфейса пользователя, тестирование системы).

**1 АНАЛИТИЧЕСКАЯ ЧАСТЬ**

**1.1 Анализ предметной области**

Библиотека – это учреждение культуры, организующее сбор, хранение и общественное пользование произведениями печати и другими документами.

Основными направлениями работы любой библиотеки являются: комплектование и организация книжного фонда; обслуживание читателей.

При работе с системой библиотекарь должен иметь возможность решать следующие задачи:

* принимать новые книги и регистрировать их в библиотеке;
* относить книги к определенной категории и управлять информацией о новых издательствах и направлениях;
* ввести учёт выданных книг читателям;
* заполнять личную информацию о читателях.

Типичный пример работы библиотеки: читатель берет книгу на определенное время, библиотекарь оформляет читателя, затем читатель возвращает книгу.

**1.2 Постановка задачи**

Исходя из того, что пользователем приложения будет библиотекарь должно быть предусмотрено разграничение доступа к закрытой информации (персональная информация о читателях и записи о выдаче книг). Определим следующие функциональные требования для приложения:

* ограничение доступа к возможностям библиотекаря;
* хранение актуальной информации о книгах;
* возможность добавления, удаления и редактирования данных о авторах, направлениях и издательствах;
* возможность изменять список читателей, удаляя и добавляя в него читателей;
* возможность учёта выдачи книг читателям;
* интуитивно понятный интерфейс;
* предоставление общей информации о книгах, авторах, издательствах и направлениях, имеющихся в библиотеке.

Согласно вышеперечисленным функциональным требованиям, выделим сущности. Сущность – любой различимый объект, информацию о котором необходимо хранить в базе данных.

Для нашей задачи можно определить следующие сущности, которые в дальнейшем будут использоваться для построения внешней модели проектируемой базы данных: читатели, выдача книг, книги, авторы, направления, издательства.

Каждой из перечисленной сущности соответствует свой набор атрибутов:

* Читатели (ID читателя, ФИО, пол, дата рождения, телефон, адрес);
* Выдача книг (ID выдачи, ID книги, ID читателя, дата выдачи, дата возврата);
* Книги (ID книги, ID направления, ID издательства, ID авторов, название, количество страниц);
* Авторы (ID автора, ФИО, учёная степень, дата рождения);
* Направления (ID направления, описание);
* Издательства (ID издательства, название, адрес).

**2 РАЗРАБОТКА ЛОГИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

**2.1 Построение диаграммы «сущность-связь» в нотации П.Чена**

После того, как был проведен анализ предметной области и выделены основные сущности стало возможным построение диаграммы «сущность-связь» в нотации П.Чена, но для начала необходимо определить отношения между выделенными сущностями.

Была сформирована сложная сетевая структура, основанная на основных сущностях предметной области. Разработанная структура представлена на рисунке 1.

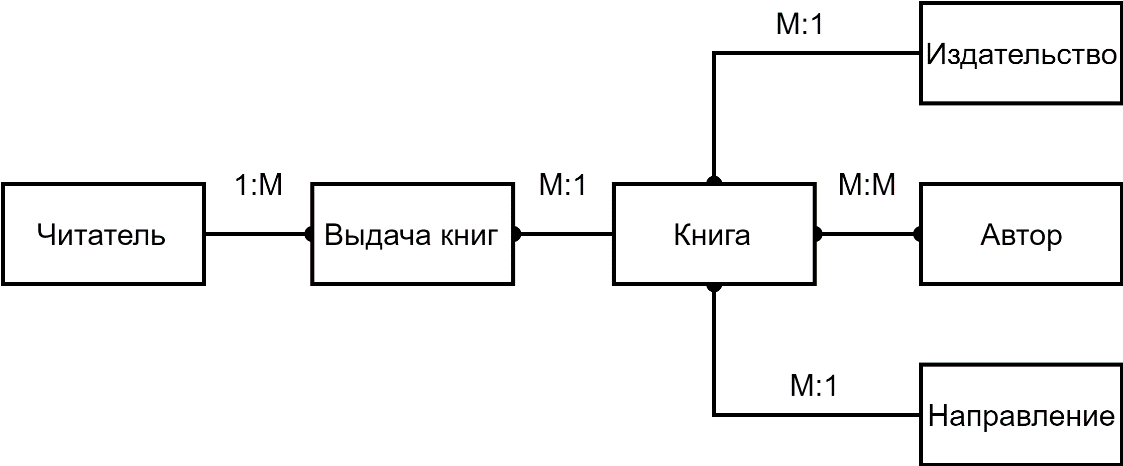


Рисунок 1 – Сложная сетевая структура

На основе сложной сетевой структуры была создана простая сетевая структура путём добавления промежуточных таблиц для того, чтобы связь многие ко многим трансформировать в связь один ко многим (рисунок 2).

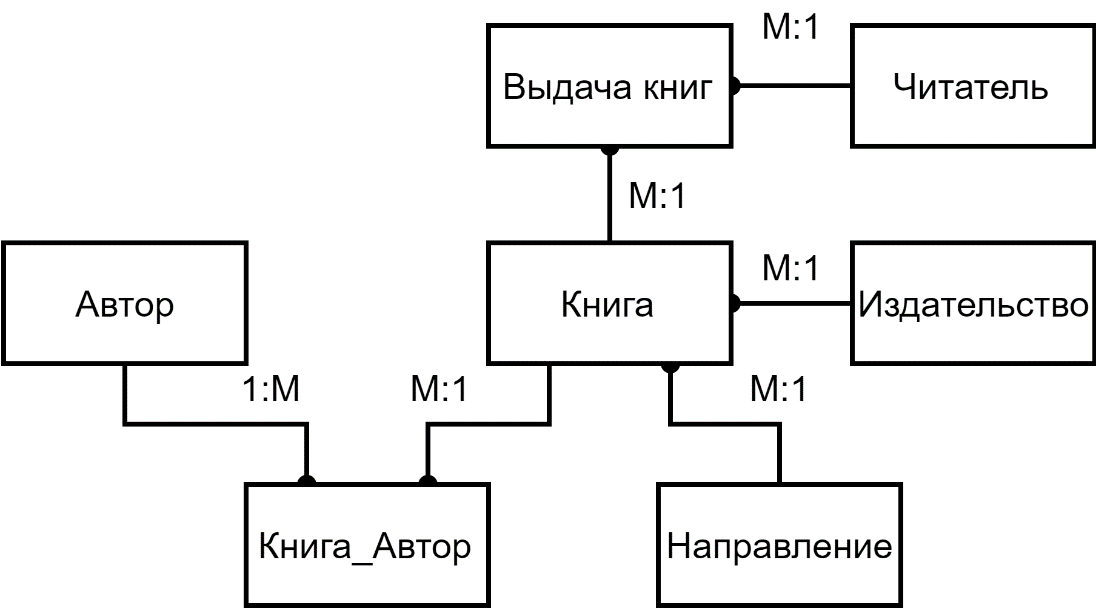


Рисунок 2 – Простая сетевая структура

Нормализация не может быть произведена в простой сетевой структуре, поэтому необходимо привести сетевую модель к иерархической.

Иерархическая (древовидная) структура не допускает наличия нескольких родителей у одного потомка. Поэтому для приведения простой сетевой структуры к древовидной введем избыточность, продублировав отношения, которые являются потомками. Полученная древовидная структура представлена на рисунке 3.

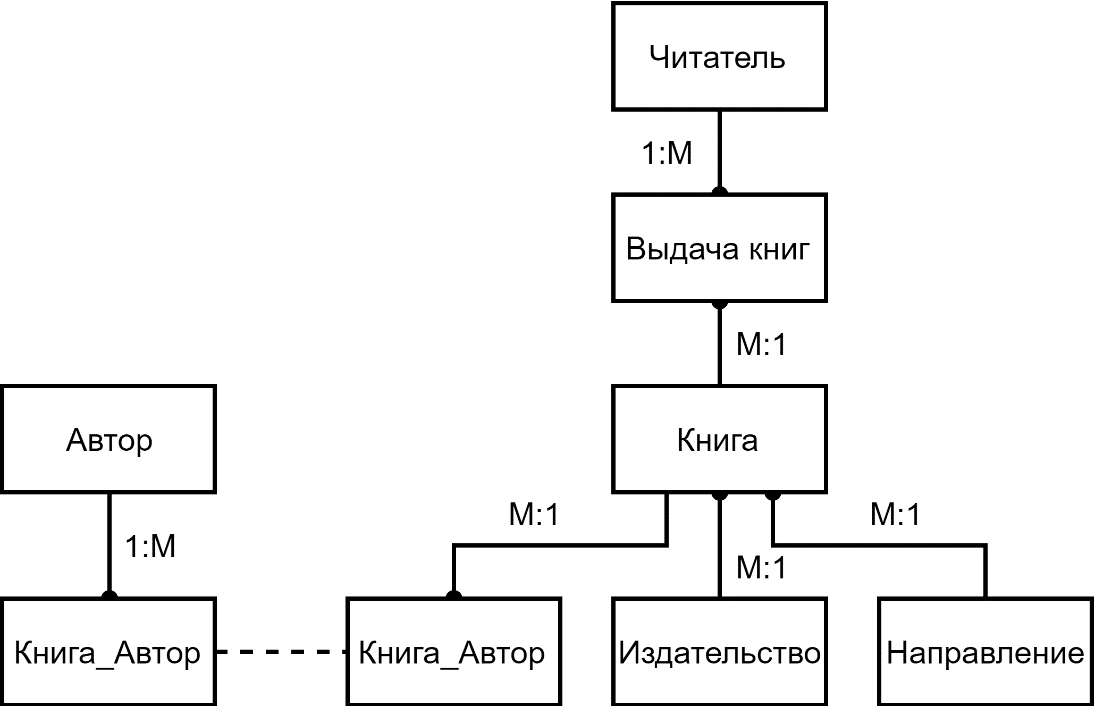


Рисунок 3 – Древовидная структура

Далее на основе прошлых таблиц была синтезирована ER-диаграмма (рисунок 4), отображающая взаимосвязь объектов, а также основные элементы взаимодействующих объектов.

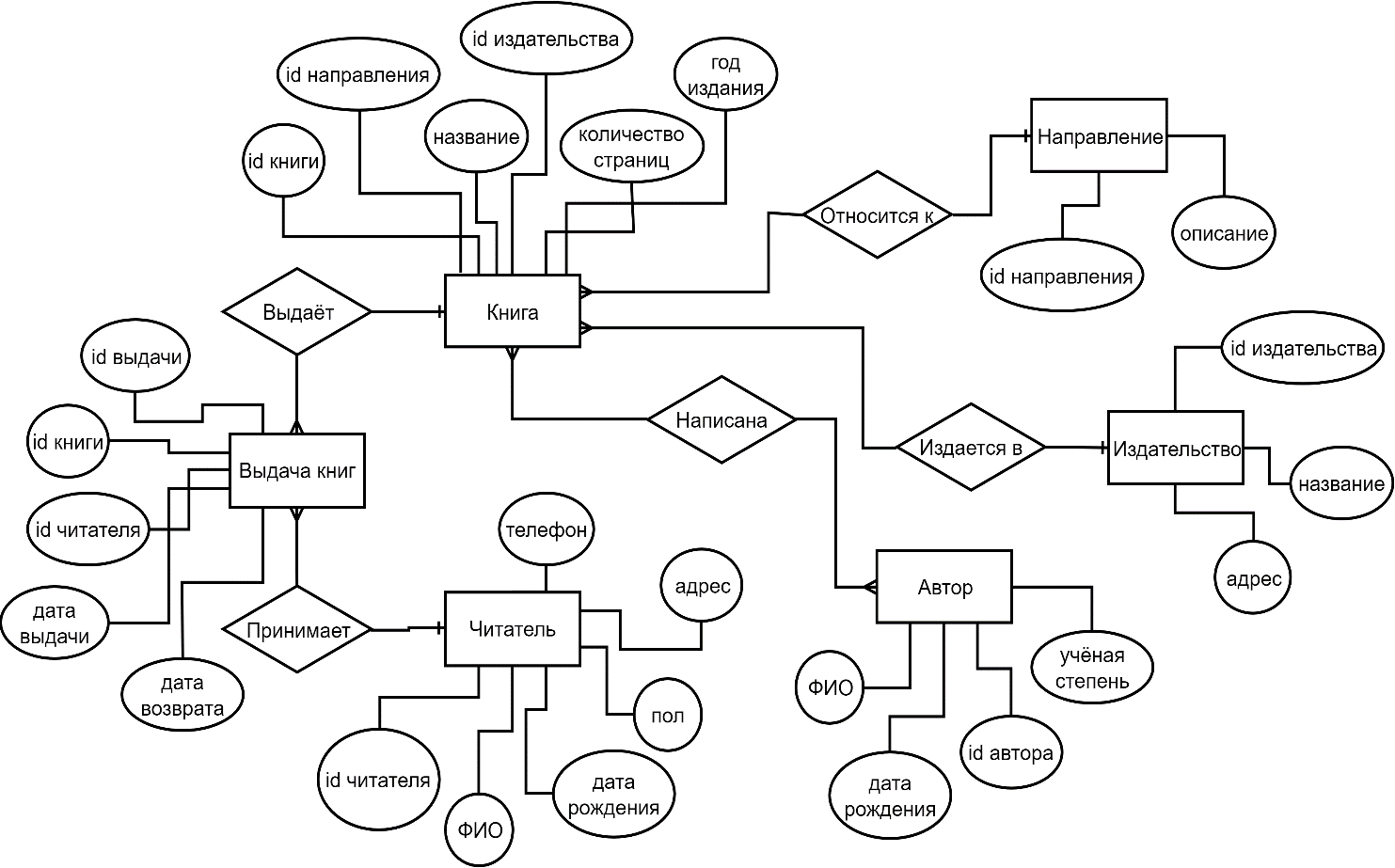


Рисунок 4 – ER диаграмма в нотации П.Чена

**2.2 Построение модели основанной на ключах**

Основной целью модели, основанной на ключах, является широкий обзор структур данных и ключей, нужных для поддержки определенной области. Эта модель определяет контекст, в котором могут быть созданы подробные модели, пригодные для конкретного воплощения.

Модель, основанная на ключах строится на основе ERD, но отображает больше деталей: ключевые поля, внешние ключи.

Разработанная модель представлена на рисунке 5.

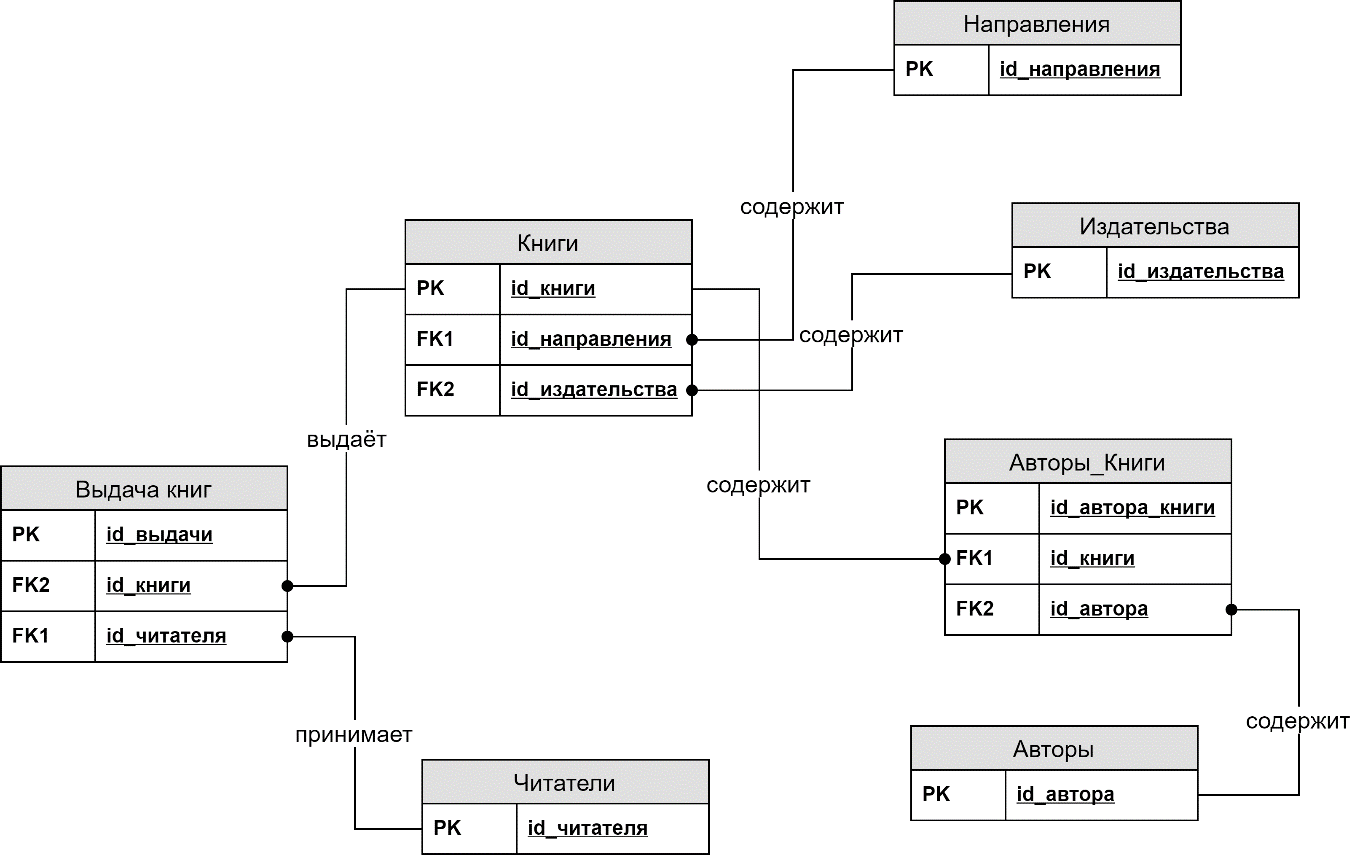


Рисунок 5 – Модель, основанная на ключах

**2.3 Построение полной атрибутивной модели**

Полная атрибутивная модель достигается нормализацией отношений до третьей или четвёртой нормальной формы.

Целью нормализации базы данных является сокращение избыточности. Нормализация предполагает последовательное приведение схемы базы данных к так называемым нормальным формам, каждая последующая из которых предъявляет более строгие требования по сравнению с предыдущей.

Рассмотрим все выделенные нами сущности, и проверим их на атомарность. В каждой из сущности соблюдается атомарность, следовательно, делаем вывод о том, что наша база данных находится в первой нормальной форме.

Так как наша база данных, находится в первой нормальной форме, рассмотрим её на удовлетворение второй нормальной форме (отношение R находится во второй нормальной форме в том и только в том случае, когда находится в 1НФ и каждый не ключевой атрибут полностью зависит от первичного ключа). Каждая сущность содержит уникальный, искусственно введенный, первичный ключ, не обладающий смысловой нагрузкой, от которого полностью зависят все не ключевые атрибуты, это удовлетворяет условиям второй нормальной форме, следовательно, делаем вывод о том, что база данных находится во второй нормальной форме.

Так как наша база данных находится во второй нормальной форме, рассмотрим на её удовлетворение третьей нормальной форме (отношение R находится в третьей нормальной форме в том и только в том случае, если находится в 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа). Во всех созданных сущностях транзитивные зависимости отсутствуют, что говорит о том, что наша база данных находится в третьей нормальной форме.

В ходе нормализации была построена полная атрибутивная модель в нотации IDEF1X, которая представлена на рисунке 6.

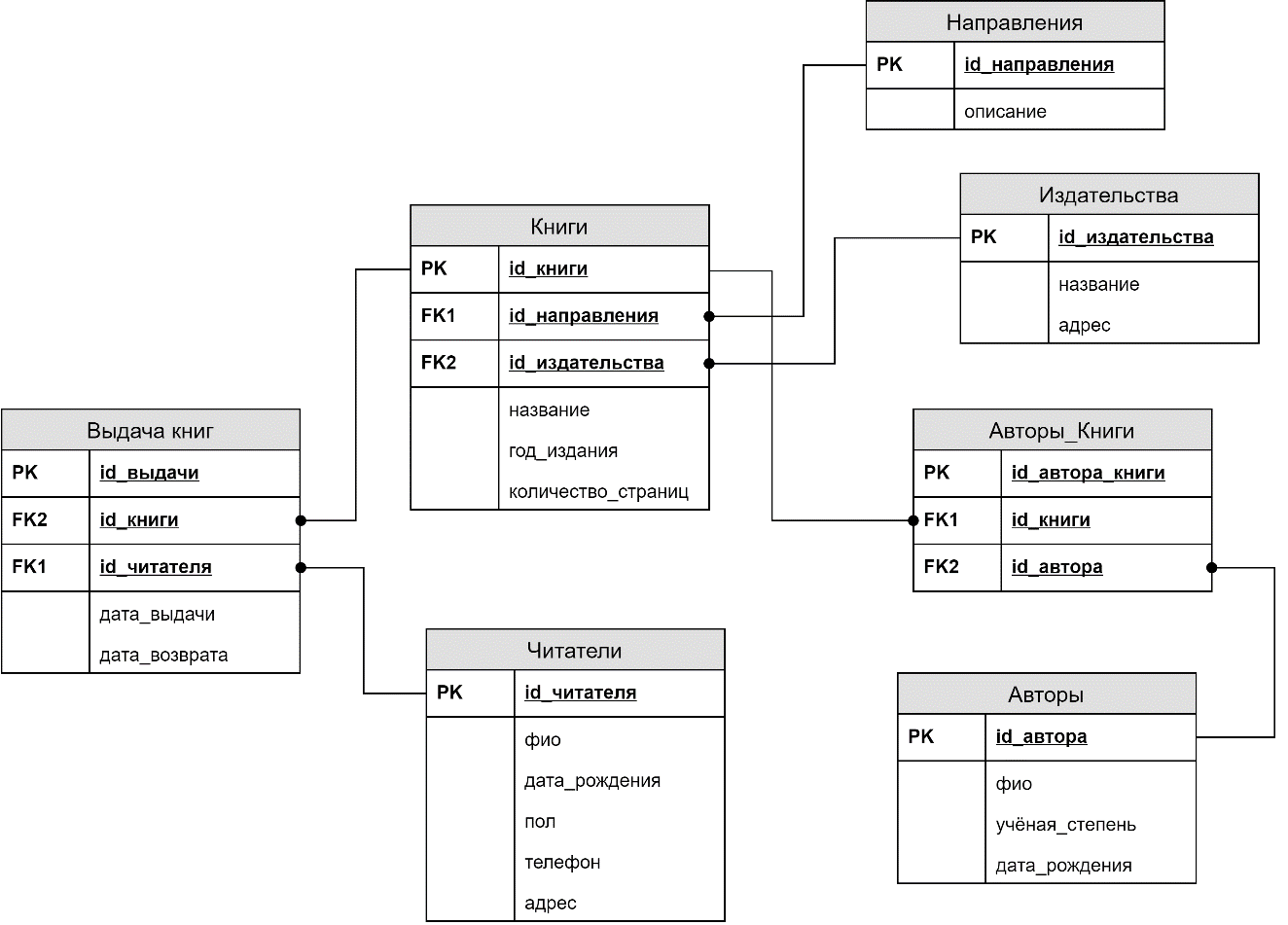


Рисунок 6 – Полная атрибутивная модель

**3 РАЗРАБОТКА ФИЗИЧЕСКОЙ МОДЕЛИ БАЗЫ ДАННЫХ**

**3.1 Выбор аппаратной и программной платформы для реализации базы данных**

Выбор системы управления базами данных представляет собой сложную многопараметрическую задачу и является одним из важных этапов при разработке приложений баз данных. Выбранная СУБД должна удовлетворять как текущим, так и будущим потребностям системы. В настоящее время существует достаточно много различных СУБД, но в данной курсовой работе была выбрана именно MySQL, т.к. она обладает следующими преимуществами:

* Быстродействие. СУБД MySQL является одной из самых быстрых баз данных из имеющихся на современном рынке.
* Простота использования. MySQL является высокопроизводительной и относительно простой в использовании СУБД.
* Возможности доступа. Сервер позволяет одновременно подключаться неограниченному количеству пользователей, одновременно работающих с базой данных.
* Аппаратная совместимость. MySQL отлично работает под управлением большинства операционных систем.
* Бесплатное использование. MySQL полностью бесплатна. Исходный код этой системы открыт и любой желающий может разработать на его базе собственные некоммерческие проекты, при условии соблюдения требований лицензии.

Все эти особенности сделали MySQL идеально подходящей СУБД под нашу базу данных.

**3.2 Реализация базы данных**

Так как у нас уже имеется логическая модель базы данных в третьей нормальной форме, а также выбрана СУБД, которая её будет реализовать, преобразуем логическую модель в физическую. На рисунке 7 представлена физическая модель нашей базы данных.

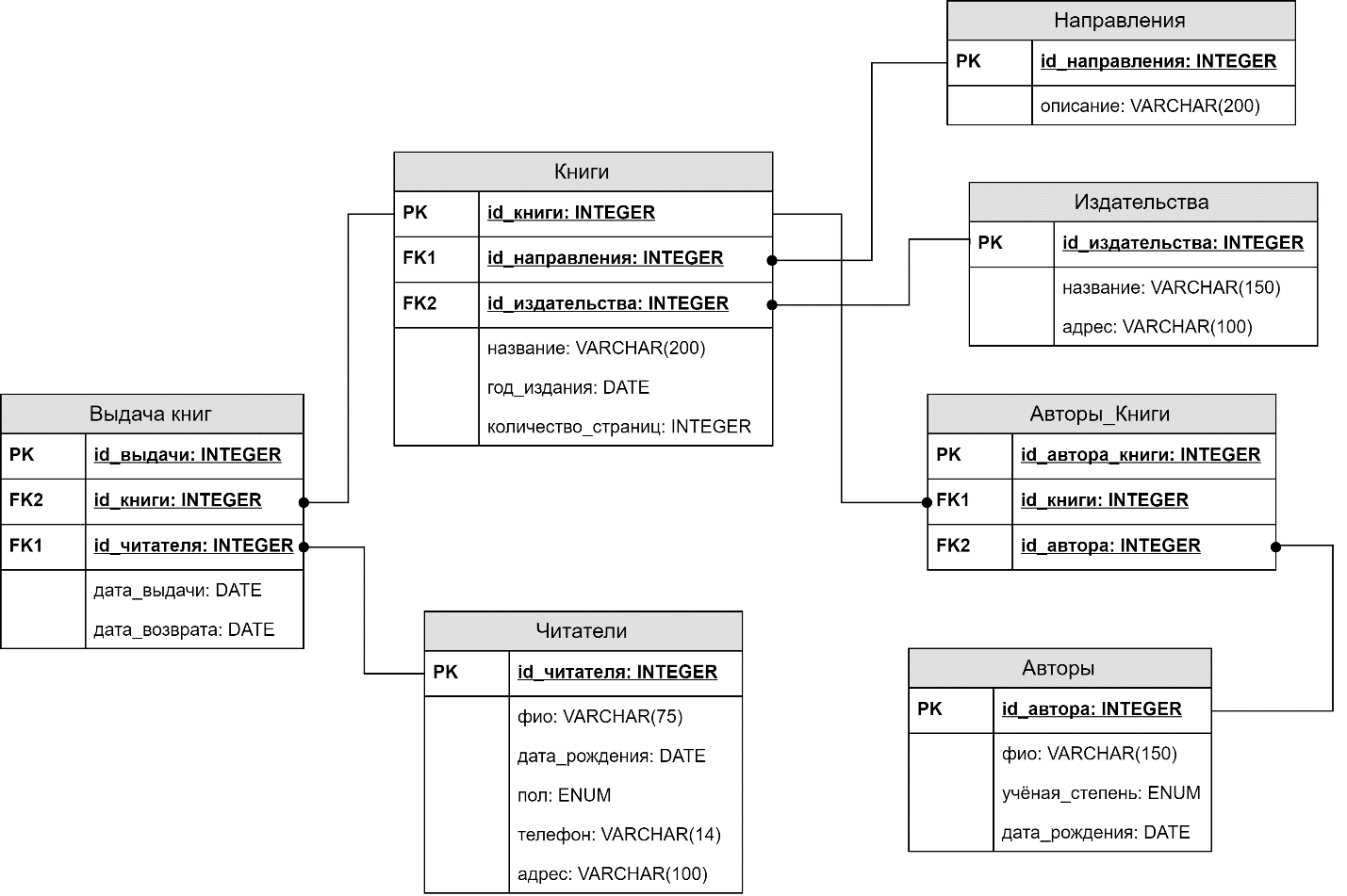


Рисунок 7 – Физическая модель базы данных

**3.3 Тестирование базы данных**

После создания таблиц и заполнения основными данными необходимо протестировать полученную базу данных. Для этого построим не менее 12 разноплановых запросов и удостоверимся что все разработанные запросы выполняются правильно.

Выполним просмотр всех книг, заменив id авторов, направлений и издательств на соответствующие им поля. Результат выполнения запроса представлен на рисунке 8.

SELECT книги.id\_книги, книги.название, GROUP\_CONCAT(distinct авторы.фио SEPARATOR ', ') as авторы, направления.описание 'направление', издательства.название 'издательство', книги.год\_издания, книги.количество\_страниц

FROM книги

LEFT JOIN авторы\_книги ON авторы\_книги.id\_книги = книги.id\_книги

LEFT JOIN авторы ON авторы\_книги.id\_автора = авторы.id\_автора

LEFT JOIN направления ON направления.id\_направления = книги.id\_направления

LEFT JOIN издательства ON издательства.id\_издательства = книги.id\_издательства

GROUP BY книги.id\_книги;

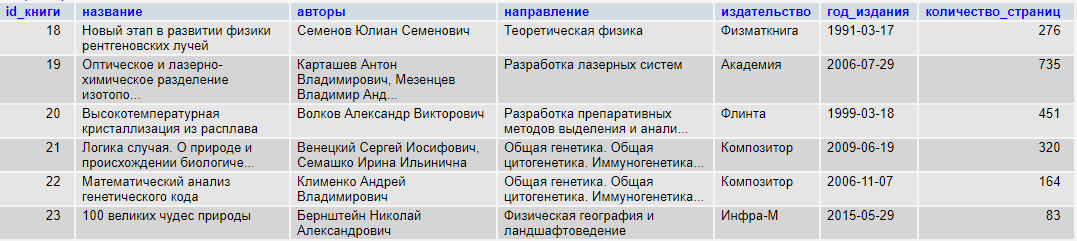


Рисунок 8 – Результат выполнения запроса для просмотра всех книг

Добавим новую книгу в базу данных, следующим SQL запросом:

INSERT INTO книги VALUES (NULL, 'Как выполнять тестирование баз данных', '72', '25', '2001-09-11', '228');

Выполним поиск добавленной книги по году, проверив на правильность выполнения сразу два запроса.

SELECT \* FROM книги WHERE год\_издания LIKE '%2001%';

Результат добавления книги и её поиск изображен на рисунке 9.



Рисунок 9 – Результат поиска добавленной книги по году

Аналогично добавим авторов для новой книги:

INSERT INTO авторы\_книги VALUES (NULL, '75', '25'), (NULL, '75', '29');

Результат добавления продемонстрирован на рисунке 10.

Также создадим SQL запрос, выполняющий поиск добавленной книги и заменяющий id книги на название, а id авторов на их ФИО (рисунок 10).

SELECT книги.название, GROUP\_CONCAT (distinct авторы.ФИО SEPARATOR ', ') as авторы

FROM авторы\_книги

LEFT JOIN книги ON авторы\_книги.id\_книги = книги.id\_книги

LEFT JOIN авторы ON авторы\_книги.id\_автора = авторы.id\_автора

WHERE авторы\_книги.id\_книги = '75'

GROUP BY книги.название;



Рисунок 10 – Результат добавления авторов новой книге и их вывод

Оформим нового читателя, выполнив следующий запрос:

INSERT INTO читатели VALUES (NULL, 'Фёдоров Мирон Янович', '1985-01-31', 'мужчина', '79358541097', 'г.Москва');

На рисунке 11 представлен результат добавления записи о новом читателе.



Рисунок 11 – Результат оформления нового читателя

Добавим новую выдачу книги читателю, которого мы оформили в прошлом запросе. Результат выполнения запроса продемонстрирован на рисунке 12.

INSERT INTO выдача\_книг VALUES (NULL, '75', '28', '2019-12-08', '2019-12-11');



Рисунок 12 – Результат добавления новой выдачи в базу данных

Выполним тестирование ссылочной целостности базы данных, каскадирования операций удаления и модификации атрибутов, являющихся внешними ключам.

Обновим поля “id\_книги” и “id\_читателя” в родительских таблицах. Результат обновления представлен на рисунке 13.

UPDATE читатели SET id\_читателя = '35' WHERE читатели.id\_читателя = 28;

UPDATE книги SET id\_книги = '96' WHERE книги.id\_книги = 75;



Рисунок 13 – Результат модификаций id в родительских таблицах

Удалим добавленную книгу в родительской таблице и выполним поиск по её id в таблице выдачи книг, чтобы удостовериться, что каскадные ограничения ссылочной целостности работают корректно (соответствующие строки обновляются или удаляются из ссылающейся таблицы, если данная строка обновляется или удаляется из родительской таблицы).

SQL запрос удаления в родительской таблице:

DELETE FROM книги WHERE книги.id\_книги = 96;

SQL запрос поиска по id удаленной книге в таблице выдачи книг, результат которого изображен на рисунке 14:

SELECT \* FROM выдача\_книг WHERE id\_книги = 96;



Рисунок 14 – Результат поиска выдачи по удаленной книге

Сравнив полученные результаты после обновления (рисунок 13 с рисунком 12) и результаты после удаления (рисунок 14), можно удостовериться, что при изменении первичных ключей в родительских таблицах, их внешние ключи меняются в соответствии с установленными каскадными ограничениями.

**3.4 Разграничения прав доступа**

Для работы с проектируемой базой данных было выделено 2 пользователя, имеющие различные уровни доступа – это администратор и читатель.

Администратор управляет выдачами книг и имеет полный доступ к остальным таблицам в базе данных.

Читатель имеет возможность просматривать и выполнять поиск по всем книгам, хранящимся в библиотеке, авторам, направлениям и издательствам.

**3.5 Расчет информационных параметров базы данных**

Длина логической записи j-ого файла определяется как сумма длин полей, её составляющих и указателей, если они организуются разработчиком

где – число групп полей в записях; – длина группы [байт].

Для переменных длин полей возьмем максимальную длину поля, следовательно, размер типов данных для хранения в памяти будет составлять:

1. INTEGER – 4 байта
2. VARCHAR(x) – x байт
3. DATE – 3 байта
4. ENUM – 2 байта

Вычислим:

Объем памяти, необходимый для размещения информационного фонда без учёта системных данных и указателей составит

где – число типов записей в информационном фонде; – количество записей -го файла.

Предположим, что в базе содержится:

20 записей о выдаче книг, 45 книг, 30 записей о читателях, 60 направлений, 15 издательств, 60 авторов и 45 связей авторы-книги, тогда

Приращение информационного фонда вычисляется как

где – число добавленных типов записей; – интенсивность добавления записей в файл -го типа.

Получаем:

Время заполнения информационного фонда определяется как

Получаем:

Время резервного копирования определяется интенсивностью отказов, сопровождающихся потерей данных

**4 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ**

**4.1 Обоснования выбора языка программирования**

В качестве языка программирования был выбран PHP. Это язык программирования, используемый на стороне WEB-сервера для динамической генерации HTML-страниц.

PHP один из немногих языков программирования, созданных специально для разработки веб-приложений. Поэтому он включает в себя все функции, необходимые именно для работы на веб-сервере, и при этом лишен избыточности, свойственной многим его конкурентам.

Одним из его преимуществ является доступность и бесплатность, а также богатый инструментарий и естественная поддержка многими популярными Web-серверами. Также приятной особенностью PHP является то, что его команды включаются в обычные HTML-страницы с помощью специальных тегов, создавая возможность чередовать простой HTML и PHP код.

**4.2 Разработка интерфейса пользователя**

В ходе разработки приложения были спроектированные следующие страницы: начальная страница с выбором уровня доступа; страница выдачи книг, доступная только администраторам; просмотр книг; страницы для управления данными читателей и авторов книг; страницы для просмотра, редактирования и удаления данных о направлениях и издательствах книг, находящихся в библиотеке. В связи с тем, что для работы с базой данных выделено 2 пользователя – это администратор и читатель, в приложении предусмотрено ограничение на редактирование и удаление данных: у читателей отсутствуют соответствующие кнопки редактирования и удаления, а страницы выдачи книг и личной информацией для читателей не доступна.

Для каждого поля ввода данных была добавлена валидация, а при удалении записи появляется диалоговое окно с подтверждением. Далее будут рассмотрены все перечисленные страницы на рисунках.

На рисунке 15 представлен процесс входа в базу данных, позволяющий выбрать уровень доступа.



Рисунок 15 – Начальная страница с выбором уровня доступа к базе данных

При входе с уровнем доступа “Читатель” открывается начальная страница “Книги”, представленная на рисунке 16. На этой странице отображаются все имеющиеся книги в библиотеке, где читатель имеет возможность выполнять поиск по названию книг, их авторам, направлениям и издательствам, а также по дате издания и количеству страниц в книге.



Рисунок 16 – Страница просмотра книг для читателей

При необходимости узнать больше дополнительных сведений о авторе, читатель имеет возможность перейти на страницу “Авторы”, находящуюся на боковой панели. Содержимое данной страницы представлено на рисунке 17.

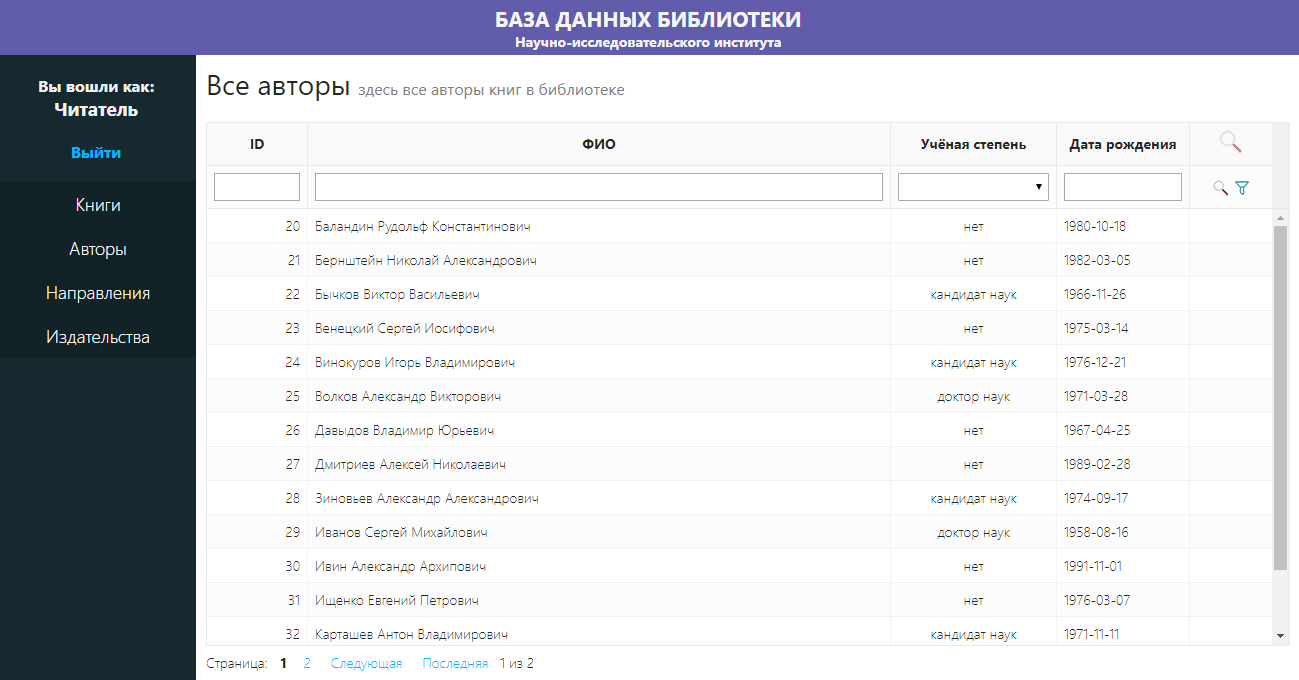


Рисунок 17 – Страница просмотра авторов книг

Если читателю необходимо узнать информацию о имеющихся направлениях или издательствах книг в библиотеке, он может перейти на соответствующую страницу и воспользоваться поиском. Страница “Направления” представлена на рисунке 18, а содержимое страницы “Издательства” на рисунке 19.

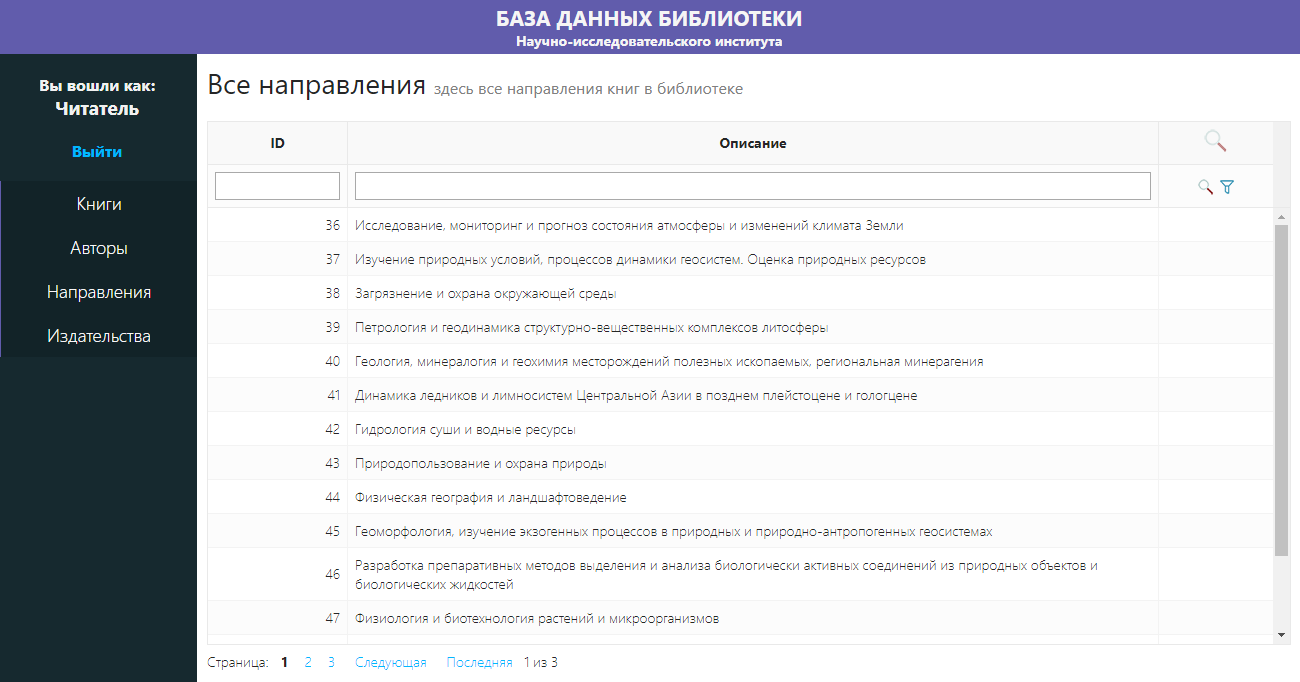


Рисунок 18 – Страница направлений книг

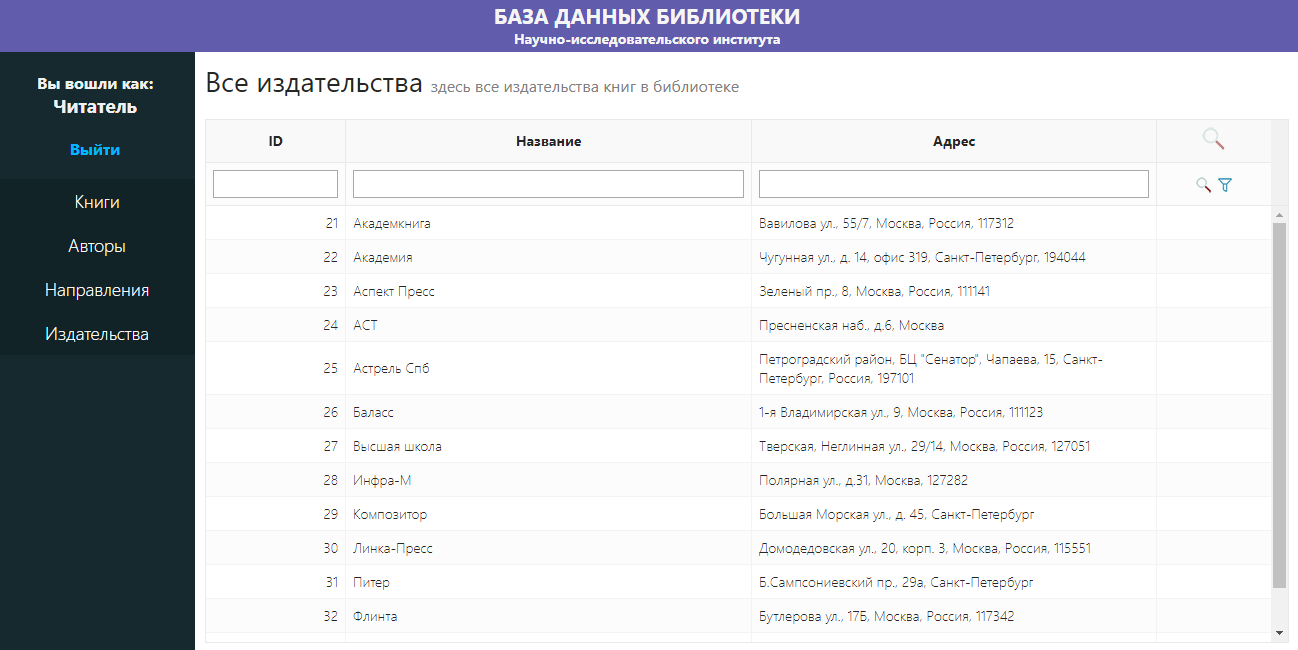


Рисунок 19 – Страница издательств книг

Рассмотрим возможности администратора. Администратор имеет полный доступ ко всем страницам, в его возможности входит добавление, удаление и редактирование любых данных.

Его основная страница “Выдачи книг”, представленная на рисунке 20, содержит записи о выдачах книг читателям. Каждая запись состоит из ID книги, которую читатель взял до определенного времени, ID данного читателя, даты выдачи и даты возврата.

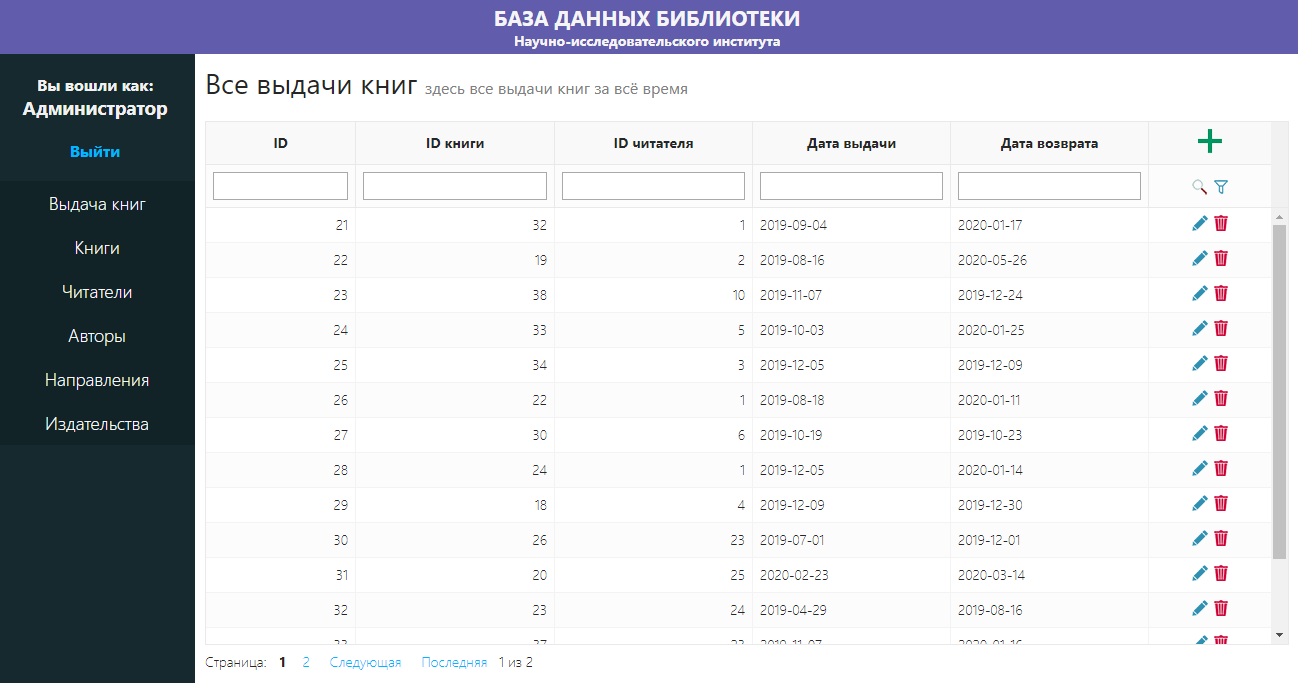


Рисунок 20 – Основная страница с записями о выдачах книг

В отличии от читателя, администратор имеет доступ к странице “Читатели”, на которой хранится личная информация читателей, такая как ФИО, дата рождения, пол, телефон и адрес. Описанная выше страница представлена на рисунке 21.

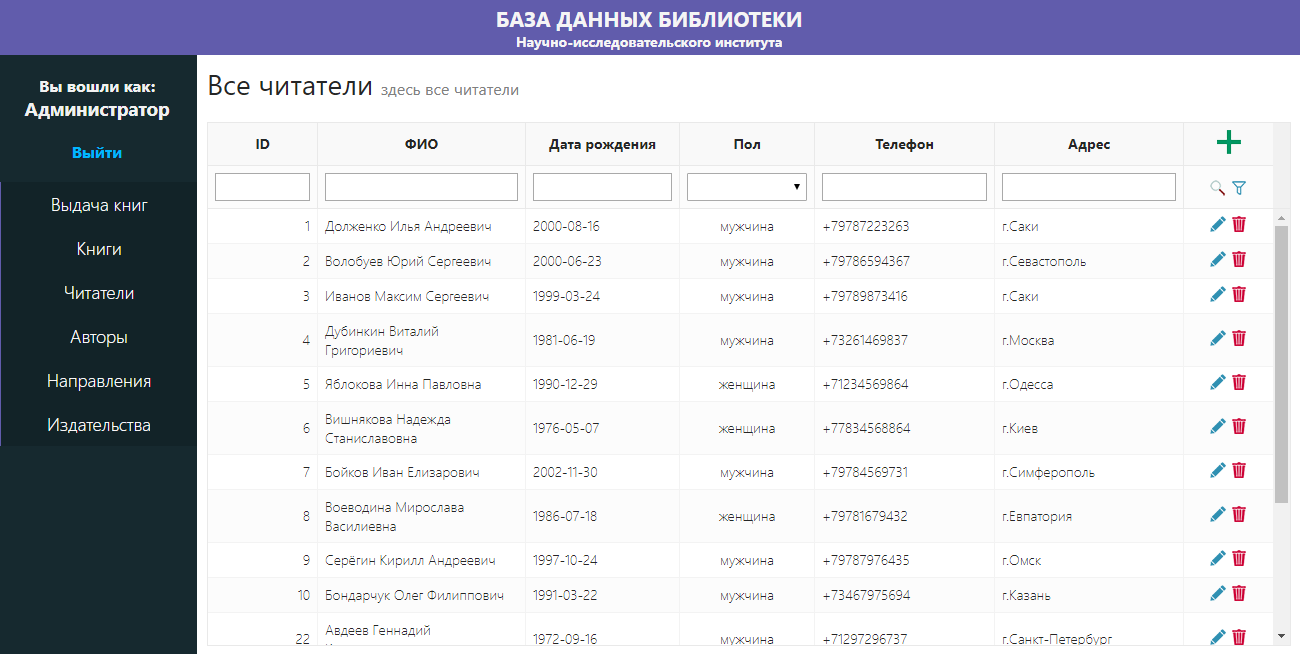


Рисунок 21 – Страница с личной информацией о читателях

На рисунках продемонстрированы основные страницы приложения. Страницы для администратора, содержащие информацию о авторах, книгах, направлениях и издательствах, аналогичны соответствующим страницам для читателей, но с возможностями добавления, редактирования и удаления.

**4.3 Алгоритм работы каждого из модулей**

При загрузке каждой страницы выполняется запрос к базе данных и на страницу выводится соответствующая таблица.

При нажатии на кнопку добавления записи становятся доступны поля для ввода и после подтверждения выполняется SQL запрос добавления, в полях которого содержатся введенные в таблице данные.

По аналогии с процессом добавления работает и возможность редактирования: при выборе записи, её поля становятся доступные для редактирования и после подтверждения в базу данных отправляется запрос с отредактированными полями.

Модуль удаления еще более прост: при нажатии на кнопку удаления записи, на экране появляется диалоговое окно с подтверждением, если действие было подтверждено, то выполняется запрос на удаление к базе данных по ключевому полю.

**4.4 Тестирование работы приложения**

Произведем процесс полного добавления информации, начиная от заполнения направлений и издательств, заканчивая главной записью о выдаче добавленной книги читателю.

Начнем с добавления направления для новой книги. Данный процесс представлен на рисунке 22.

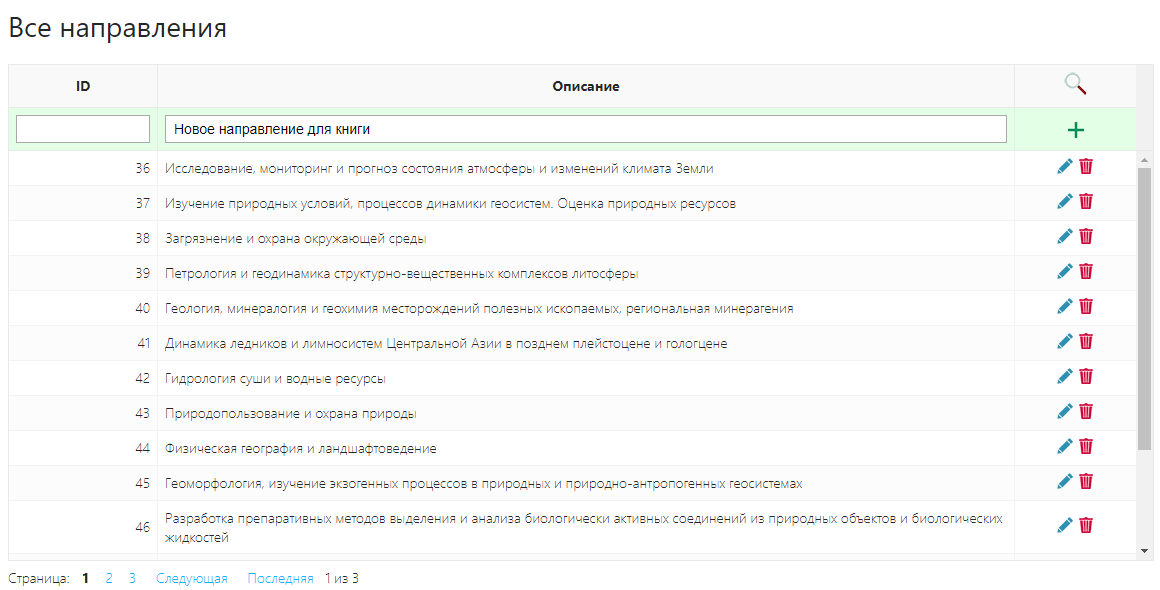


Рисунок 22 – Процесс добавления направления

Далее выполним добавление издательства (рисунок 23).

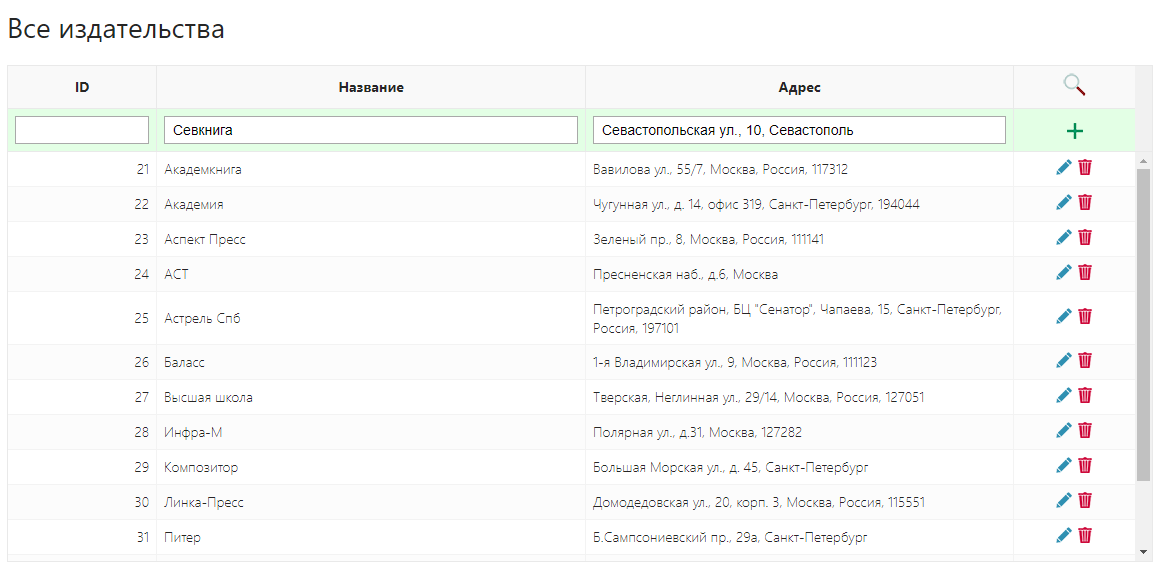


Рисунок 23 – Добавление нового издательства

Заполним информацию о авторе для будущей книги. На рисунке 24 представлен процесс добавления автора.

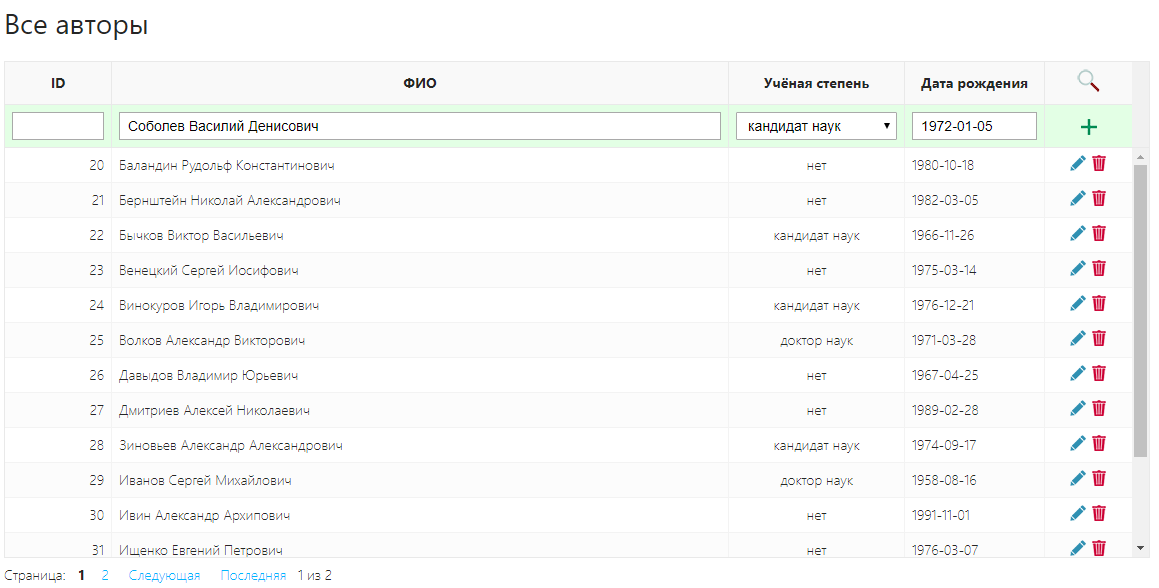


Рисунок 24 – Процесс добавления автора

После заполнения составляющих книги, перейдем к её добавлению. Добавление новой книги в базу данных представлено на рисунке 25.

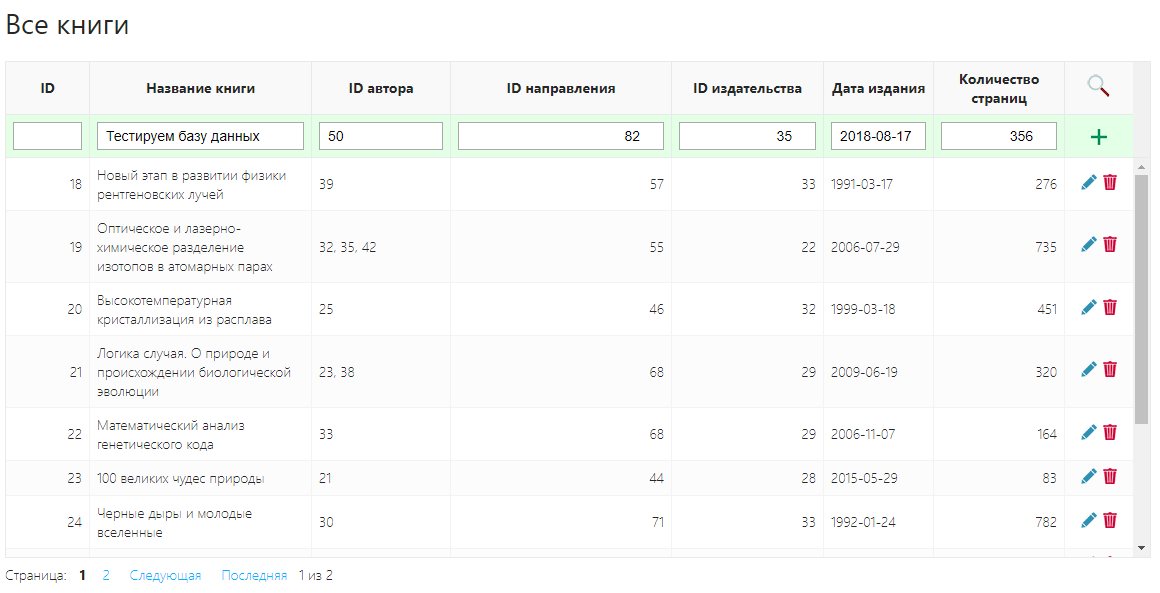


Рисунок 25 – Добавление новой книги в базу данных

Предположим, что администратор ошибся и забыл добавить еще одного автора для новой книги. Изменим поле авторов, добавив еще одного из уже имеющихся в базе данных. Процесс изменения представлен на рисунке 26.

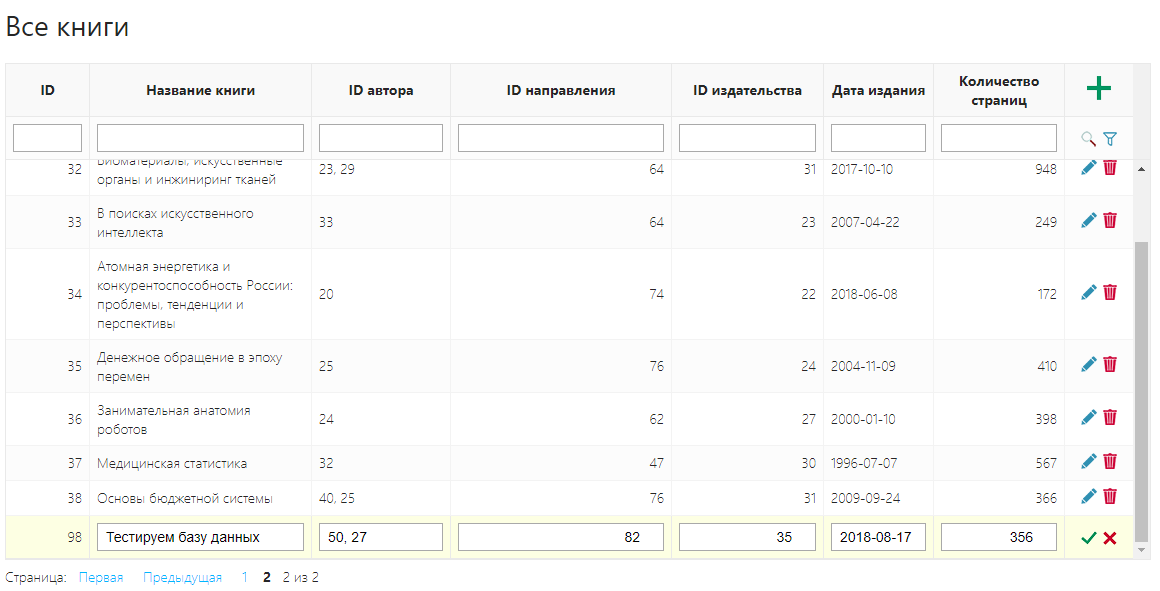


Рисунок 26 – Процесс редактирования записи

Для проверки результата редактирования и добавления в целом, зайдем в систему как читатель и выполним поиск по названию книги. Результат поиска изображен на рисунке 27.

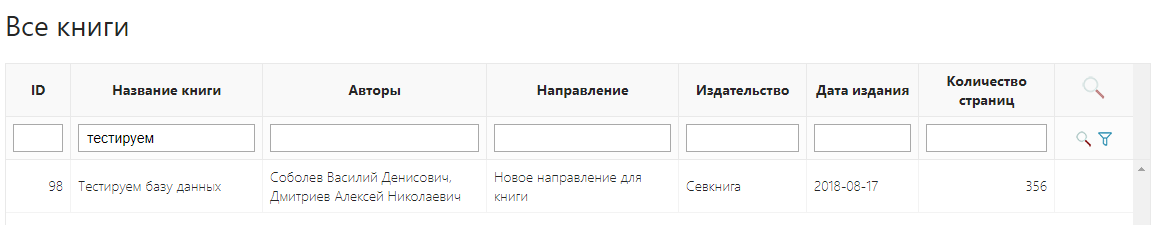


Рисунок 27 – Результат поиска добавленной книги

Продолжим оформление выдачи книги и перейдем к заполнению личной информации о читателе (рисунок 28).

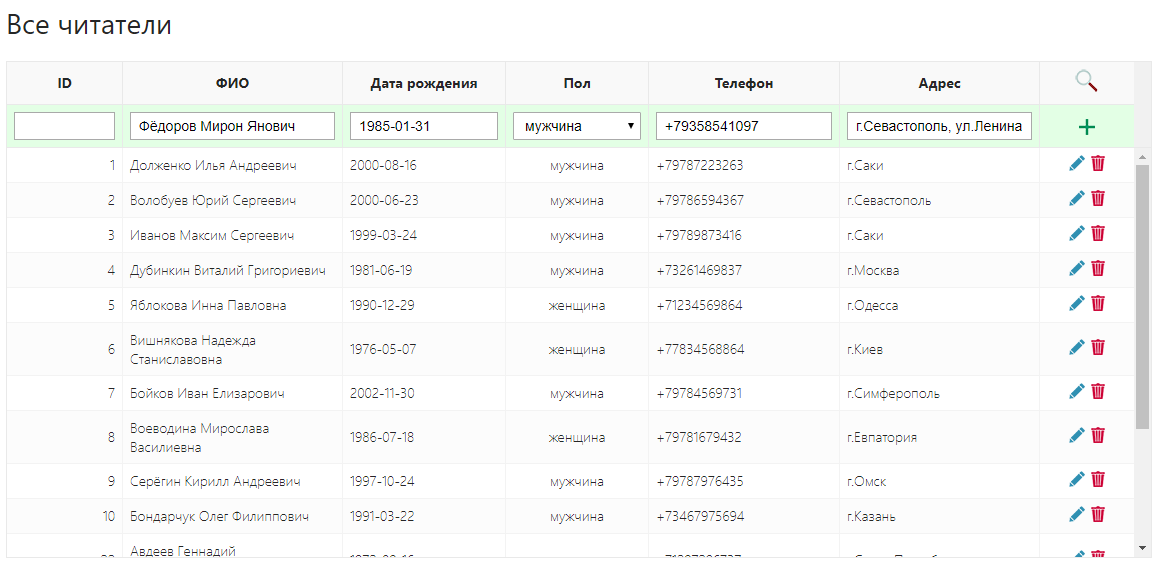


Рисунок 28 – Заполнение личной информации о читателе

Последним действием для тестирования будет оформление выдачи новой книги читателю. Данный процесс представлен на рисунке 29.

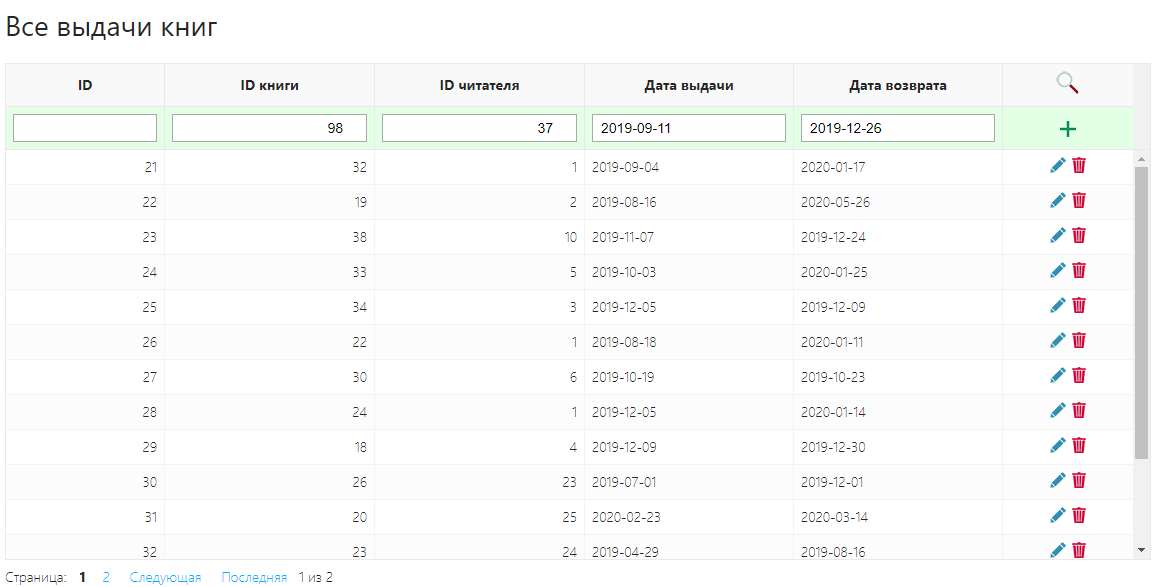


Рисунок 29 – Оформление выдачи книги

В результате тестирования приложения не было выявлено дефектов, вызывающих нарушение структуры базы данных, приложение выполняет необходимые действия по добавлению, изменению и удалению данных в таблицах.

**ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте была спроектирована реляционная база данных на тему «База данных библиотеки научно-исследовательского института». Написано приложение на языке PHP, реализующее работу с базой данных. Также был разработан простой и удобный интерфейс для работы с базой данных. Кроме того, была обеспечена защита базы данных, путём разграничения прав доступа на прикладном уровне.

Для спроектированной логической модели базы данных была проведена нормализация до третьей нормальной формы, которая позволила уменьшить избыточность данных, а также сохранить их целостность. Реализована физическая модель базы данных.

Были закреплены навыки проектирования логической модели базы данных, проведения их нормализации, построения физической модели и разработки программного обеспечения.

В результате выполнения курсового проекта был получен программный продукт для использования в библиотеках научно-исследовательских институтов, реализующий систему учёта выдач книг читателям и веб-интерфейс для администратора и пользователей.

**СПИСОК ЛИТЕРАТУРЫ И ИНФОРМАЦИОННЫХ РЕСУРСОВ**

1. Линн Бейли, Майкл Моррисон. Изучаем PHP и MySQL. - Москва: Эксмо, 2010. - 768 с.

2. Гарсиа-Молина Г. Системы баз данных. Полный курс: Пер. с англ./Г.Гарсиа-Молина, Д.Д.Ульман, Д.Уидом. – М.: Издательский дом «Вильямс», 2004. – 1088с.

3. Дейт К.Дж. Введение в системы баз данных, 7-е издание: Пер. с англ./ К.Дж.Дейт. – М.: Издательский дом «Вильямс», 2002. – 1072с.

4. Ульман Д. Основы систем баз данных/Д.Ульман. – М.: Финансы и статистика, 1983. – 334 с.

5. Кузнецов С.Д. Основы баз данных; Бином. Лаборатория знаний, Интернет-университет информационных технологий – Москва, 2010. – 488 с.

**ПРИЛОЖЕНИЕ А**

**SQL скрипт создания базы данных**

CREATE TABLE выдача\_книг (

id\_выдачи int NOT NULL AUTO\_INCREMENT,

id\_книги int NOT NULL,

id\_читателя int NOT NULL,

дата\_выдачи date NOT NULL,

дата\_возврата date NOT NULL,

PRIMARY KEY(id\_выдачи)

);

ALTER TABLE выдача\_книг ADD FOREIGN KEY(id\_читателя) REFERENCES читатели(id\_читателя) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE выдача\_книг ADD FOREIGN KEY(id\_книги) REFERENCES книги(id\_книги) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE читатели (

id\_читателя int NOT NULL AUTO\_INCREMENT,

фио varchar(75) NOT NULL,

дата\_рождения date NOT NULL,

пол enum("мужчина", "женщина") NOT NULL,

телефон varchar(14) NOT NULL,

адрес varchar(100) NOT NULL,

PRIMARY KEY(id\_читателя)

);

CREATE TABLE книги (

id\_книги int NOT NULL AUTO\_INCREMENT,

название varchar(200) NOT NULL,

id\_направления int NOT NULL,

id\_издательства int NOT NULL,

год\_издания date NOT NULL,

количество\_страниц int NOT NULL,

PRIMARY KEY(id\_книги)

);

ALTER TABLE книги ADD FOREIGN KEY(id\_направления) REFERENCES направления(id\_направления) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE книги ADD FOREIGN KEY(id\_издательства) REFERENCES издательства(id\_издательства) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE TABLE направления (

id\_направления int NOT NULL AUTO\_INCREMENT,

описание varchar(200) NOT NULL,

PRIMARY KEY(id\_направления)

);

CREATE TABLE издательства (

id\_издательства int NOT NULL AUTO\_INCREMENT,

название varchar(150) NOT NULL,

адрес varchar(100) NOT NULL,

PRIMARY KEY(id\_издательства)

);

CREATE TABLE авторы (

id\_автора int NOT NULL AUTO\_INCREMENT,

фио varchar(150) NOT NULL,

учёная\_степень enum('нет','кандидат наук','доктор наук') NOT NULL,

дата\_рождения date NOT NULL,

PRIMARY KEY(id\_автора)

);

CREATE TABLE авторы\_книги (

id\_автора\_книги int NOT NULL AUTO\_INCREMENT,

id\_книги int NOT NULL,

id\_автора int NOT NULL,

PRIMARY KEY(id\_автора\_книги)

);

ALTER TABLE авторы\_книги ADD FOREIGN KEY(id\_книги) REFERENCES книги(id\_книги) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE авторы\_книги ADD FOREIGN KEY(id\_автора) REFERENCES авторы(id\_автора) ON DELETE CASCADE ON UPDATE CASCADE;

**ПРИЛОЖЕНИЕ Б**

**Код клиентского приложения**

AdminBooksRepository.php

<?php

*include* "AdminBooks.php";

class AdminBooksRepository {

    protected $db;

    public function \_\_construct(PDO $db) {

*$this*->db = $db;

    }

    private function read($row) {

        $result = new AdminBooks();

        $result->id = $row["id\_книги"];

        $result->name = $row["название"];

        $result->id\_authors = $row["id\_авторов"];

        $result->id\_categories = $row["id\_направления"];

        $result->id\_publish = $row["id\_издательства"];

        $result->date\_create = $row["год\_издания"];

        $result->number\_pages = $row["количество\_страниц"];

*return* $result;

    }

    public function getById($id) {

        $sql = "SELECT \* FROM книги WHERE id\_книги = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

        $rows = $q->fetchAll();

*return* *$this*->read($rows[0]);

    }

    public function getAll($filter) {

        $id = $filter["id"];

        $name = "%" . $filter["name"] . "%";

        $id\_authors = $filter["id\_authors"];

        $id\_categories = $filter["id\_categories"];

        $id\_publish = $filter["id\_publish"];

        $date\_create = "%" . $filter["date\_create"] . "%";

        $number\_pages = $filter["number\_pages"];

        $sql = "SELECT книги.id\_книги, книги.название,

                GROUP\_CONCAT(distinct авторы\_книги.id\_автора SEPARATOR ', ') as

id\_авторов,книги.id\_направления, книги.id\_издательства, книги.год\_издания,

книги.количество\_страниц

                FROM книги

                LEFT JOIN авторы\_книги ON авторы\_книги.id\_книги = книги.id\_книги

                WHERE (:id = 0 OR книги.id\_книги = :id)

                    AND книги.название LIKE :name

                    AND (:id\_authors = '' OR авторы\_книги.id\_автора = :id\_authors)

                    AND (:id\_categories = 0 OR книги.id\_направления = :id\_categories)

                    AND (:id\_publish = 0 OR книги.id\_издательства = :id\_publish)

                    AND книги.год\_издания LIKE :date\_create

                    AND (:number\_pages = 0 OR книги.количество\_страниц = :number\_pages)

                    GROUP BY книги.id\_книги";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id);

        $q->bindParam(":name", $name);

        $q->bindParam(":id\_authors", $id\_authors);

        $q->bindParam(":id\_categories", $id\_categories);

        $q->bindParam(":id\_publish", $id\_publish);

        $q->bindParam(":date\_create", $date\_create);

        $q->bindParam(":number\_pages", $number\_pages);

        $q->execute();

        $rows = $q->fetchAll();

        $result = array();

*foreach*($rows as $row) {

            array\_push($result, *$this*->read($row));

        }

*return* $result;

    }

    public function insert($data) {

        $sql = "INSERT INTO книги (id\_книги, название, id\_направления, id\_издательства, год\_издания, количество\_страниц)

                VALUES (:id, :name, :id\_categories, :id\_publish, :date\_create, :number\_pages)";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":id\_categories", $data["id\_categories"], PDO::PARAM\_INT);

        $q->bindParam(":id\_publish", $data["id\_publish"], PDO::PARAM\_INT);

        $q->bindParam(":date\_create", $data["date\_create"]);

        $q->bindParam(":number\_pages", $data["number\_pages"], PDO::PARAM\_INT);

        $q->execute();

        $idAuthors = explode(",", str\_replace(" ", "", $data["id\_authors"]));

*foreach*($idAuthors as $arr){

*$this*->insertAuthors($arr, $data["name"], $data["number\_pages"]);

        }

*return* *$this*->getById(*$this*->db->lastInsertId());

    }

    public function insertAuthors($arr, $authorsName, $numberPages) {

        $sql = "INSERT INTO авторы\_книги (id\_книги, id\_автора)

                SELECT книги.id\_книги, :id\_authors

                FROM книги

                WHERE книги.название = :name AND книги.количество\_страниц = :number\_pages";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":name", $authorsName);

        $q->bindParam(":number\_pages", $numberPages);

        $q->bindParam(":id\_authors", $arr, PDO::PARAM\_INT);

        $q->execute();

    }

    public function update($data) {

        $sql = "UPDATE книги

                SET id\_книги = :id, название = :name, id\_направления = :id\_categories, id\_издательства = :id\_publish, год\_издания = :date\_create, количество\_страниц = :number\_pages

                WHERE id\_книги = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":id\_categories", $data["id\_categories"], PDO::PARAM\_INT);

        $q->bindParam(":id\_publish", $data["id\_publish"], PDO::PARAM\_INT);

        $q->bindParam(":date\_create", $data["date\_create"]);

        $q->bindParam(":number\_pages", $data["number\_pages"], PDO::PARAM\_INT);

        $q->execute();

*$this*->deleteAuthors($data["id"]);

        $idAuthors = explode(",", str\_replace(" ", "", $data["id\_authors"]));

*foreach*($idAuthors as $arr){

*$this*->insertAuthors($arr, $data["name"], $data["number\_pages"]);

        }

    }

    public function deleteAuthors($id) {

        $sql = "DELETE FROM авторы\_книги WHERE id\_книги = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

    }

    public function remove($id) {

        $sql = "DELETE FROM книги WHERE id\_книги = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

    }

}

?>

ClientBooksRepository.php

<?php

*include* "ClientBooks.php";

class ClientBooksRepository {

    protected $db;

    public function \_\_construct(PDO $db) {

*$this*->db = $db;

    }

    private function read($row) {

        $result = new ClientBooks();

        $result->id = $row["id\_книги"];

        $result->name = $row["название"];

        $result->authors = $row["авторы"];

        $result->categories = $row["направление"];

        $result->publish = $row["издательство"];

        $result->date\_create = $row["год\_издания"];

        $result->number\_pages = $row["количество\_страниц"];

*return* $result;

    }

    public function getAll($filter) {

        $id = $filter["id"];

        $name = "%" . $filter["name"] . "%";

        $authors = "%" . $filter["authors"] . "%";

        $categories = "%" . $filter["categories"] . "%";

        $publish = "%" . $filter["publish"] . "%";

        $date\_create = "%" . $filter["date\_create"] . "%";

        $number\_pages = $filter["number\_pages"];

        $sql = "SELECT книги.id\_книги, книги.название,

                GROUP\_CONCAT(distinct авторы.фио SEPARATOR ', ') as авторы,

                направления.описание 'направление', издательства.название 'издательство', книги.год\_издания, книги.количество\_страниц

                FROM книги

                LEFT JOIN авторы\_книги ON авторы\_книги.id\_книги = книги.id\_книги

                LEFT JOIN авторы ON авторы\_книги.id\_автора = авторы.id\_автора

                LEFT JOIN направления ON направления.id\_направления = книги.id\_направления

                LEFT JOIN издательства ON издательства.id\_издательства = книги.id\_издательства

                WHERE (:id = 0 OR книги.id\_книги = :id) AND книги.название LIKE :name AND авторы.фио LIKE :authors

                    AND направления.описание LIKE :categories AND издательства.название LIKE :publish

                    AND книги.год\_издания LIKE :date\_create AND (:number\_pages = 0 OR книги.количество\_страниц = :number\_pages)

                GROUP BY книги.id\_книги";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id);

        $q->bindParam(":name", $name);

        $q->bindParam(":authors", $authors);

        $q->bindParam(":categories", $categories);

        $q->bindParam(":publish", $publish);

        $q->bindParam(":date\_create", $date\_create);

        $q->bindParam(":number\_pages", $number\_pages);

        $q->execute();

        $rows = $q->fetchAll();

        $result = array();

*foreach*($rows as $row) {

            array\_push($result, *$this*->read($row));

        }

*return* $result;

    }

}

?>

AuthorsRepository.php

<?php

*include* "Authors.php";

class AuthorsRepository {

    protected $db;

    public function \_\_construct(PDO $db) {

*$this*->db = $db;

    }

    private function read($row) {

        $result = new Authors();

        $result->id = $row["id\_автора"];

        $result->name = $row["фио"];

        $result->degree = $row["учёная\_степень"];

        $result->date = $row["дата\_рождения"];

*return* $result;

    }

    public function getById($id) {

        $sql = "SELECT \* FROM авторы WHERE id\_автора = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

        $rows = $q->fetchAll();

*return* *$this*->read($rows[0]);

    }

    public function getAll($filter) {

        $id = $filter["id"];

        $name = "%" . $filter["name"] . "%";

        $degree = $filter["degree"];

        $date = "%" . $filter["date"] . "%";

        $sql = "SELECT \* FROM авторы

                WHERE (:id = 0 OR id\_автора = :id)

                    AND фио LIKE :name

                    AND (:degree = '' OR учёная\_степень = :degree)

                    AND дата\_рождения LIKE :date";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id);

        $q->bindParam(":name", $name);

        $q->bindParam(":degree", $degree);

        $q->bindParam(":date", $date);

        $q->execute();

        $rows = $q->fetchAll();

        $result = array();

*foreach*($rows as $row) {

            array\_push($result, *$this*->read($row));

        }

*return* $result;

    }

    public function insert($data) {

        $sql = "INSERT INTO авторы (id\_автора, фио, учёная\_степень, дата\_рождения)

                VALUES (:id, :name, :degree, :date)";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":degree", $data["degree"]);

        $q->bindParam(":date", $data["date"]);

        $q->execute();

*return* *$this*->getById(*$this*->db->lastInsertId());

    }

    public function update($data) {

        $sql = "UPDATE авторы

                SET id\_автора = :id, фио = :name, учёная\_степень = :degree, дата\_рождения = :date

                WHERE id\_автора = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":degree", $data["degree"]);

        $q->bindParam(":date", $data["date"]);

        $q->execute();

    }

    public function remove($id) {

        $sql = "DELETE FROM авторы WHERE id\_автора = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

    }

}

?>

ReadersRepository.php

<?php

*include* "Readers.php";

class ReadersRepository {

    protected $db;

    public function \_\_construct(PDO $db) {

*$this*->db = $db;

    }

    private function read($row) {

        $result = new Readers();

        $result->id = $row["id\_читателя"];

        $result->name = $row["фио"];

        $result->date = $row["дата\_рождения"];

        $result->gender = $row["пол"];

        $result->tel = $row["телефон"];

        $result->address = $row["адрес"];

*return* $result;

    }

    public function getById($id) {

        $sql = "SELECT \* FROM читатели WHERE id\_читателя = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

        $rows = $q->fetchAll();

*return* *$this*->read($rows[0]);

    }

    public function getAll($filter) {

        $id = $filter["id"];

        $name = "%" . $filter["name"] . "%";

        $date = "%" . $filter["date"] . "%";

        $gender = $filter["gender"];

        $tel = "%" . $filter["tel"] . "%";

        $address = "%" . $filter["address"] . "%";

        $sql = "SELECT \* FROM читатели

                WHERE (:id = 0 OR id\_читателя = :id)

                AND фио LIKE :name

                AND дата\_рождения LIKE :date

                AND (:gender = '' OR пол = :gender)

                AND телефон LIKE :tel

                AND адрес LIKE :address";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id);

        $q->bindParam(":name", $name);

        $q->bindParam(":date", $date);

        $q->bindParam(":gender", $gender);

        $q->bindParam(":tel", $tel);

        $q->bindParam(":address", $address);

        $q->execute();

        $rows = $q->fetchAll();

        $result = array();

*foreach*($rows as $row) {

            array\_push($result, *$this*->read($row));

        }

*return* $result;

    }

    public function insert($data) {

        $sql = "INSERT INTO читатели (id\_читателя, фио, дата\_рождения, пол, телефон, адрес)

                VALUES (:id, :name, :date, :gender, :tel, :address)";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":date", $data["date"]);

        $q->bindParam(":gender", $data["gender"]);

        $q->bindParam(":tel", $data["tel"]);

        $q->bindParam(":address", $data["address"]);

        $q->execute();

*return* *$this*->getById(*$this*->db->lastInsertId());

    }

    public function update($data) {

        $sql = "UPDATE читатели

                SET id\_читателя = :id, фио = :name, дата\_рождения = :date, пол = :gender, телефон = :tel, адрес = :address

                WHERE id\_читателя = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $data["id"], PDO::PARAM\_INT);

        $q->bindParam(":name", $data["name"]);

        $q->bindParam(":date", $data["date"]);

        $q->bindParam(":gender", $data["gender"]);

        $q->bindParam(":tel", $data["tel"]);

        $q->bindParam(":address", $data["address"]);

        $q->execute();

    }

    public function remove($id) {

        $sql = "DELETE FROM читатели WHERE id\_читателя = :id";

        $q = *$this*->db->prepare($sql);

        $q->bindParam(":id", $id, PDO::PARAM\_INT);

        $q->execute();

    }

}

?>

admin/authors.js

$(function () {

    var degrees = [

        { degree: "", id: 0 },

        { degree: "нет", id: 1 },

        { degree: "кандидат наук", id: 2 },

        { degree: "доктор наук", id: 3 }

    ];

    $("#jsGrid").jsGrid({

        height: "90%",

        width: "100%",

        filtering: true,

        inserting: true,

        editing: true,

        sorting: true,

        paging: true,

        autoload: true,

        pageSize: 15,

        pageButtonCount: 5,

        deleteConfirm: "Вы действительно хотите удалить автора?",

        pagerFormat: "Страница: {first} {prev} {pages} {next} {last}    {pageIndex} из {pageCount}",

        pagePrevText: "Предыдущая",

        pageNextText: "Следующая",

        pageFirstText: "Первая",

        pageLastText: "Последняя",

        controller: {

            loadData: function (filter) {

*return* $.ajax({

                    type: "GET",

                    url: "/pages/authors/",

                    data: filter

                });

            },

            insertItem: function (item) {

*return* $.ajax({

                    type: "POST",

                    url: "/pages/authors/",

                    data: item

                });

            },

            updateItem: function (item) {

*return* $.ajax({

                    type: "PUT",

                    url: "/pages/authors/",

                    data: item

                });

            },

            deleteItem: function (item) {

*return* $.ajax({

                    type: "DELETE",

                    url: "/pages/authors/",

                    data: item

                });

            }

        },

        fields: [

            { name: "id", title: "ID", type: "number", width: 60, editing: false },

            { name: "name", title: "ФИО", type: "text", width: 350, validate: "required" },

            { name: "degree", title: "Учёная степень", type: "select", width: 100, items: degrees, valueField: "degree", textField: "degree", validate: "required" },

            {

                name: "date", title: "Дата рождения", type: "text", width: 80,

                validate: function (value, item) {

*return* /*^*(19[0-9]{2}|2[0-9]{3})\-(0[1-9]|1[0-2])\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])*$*/.test(value);

                }

            },

            { type: "control" }

        ]

    });

});

admin/rent.js

$(function () {

    $("#jsGrid").jsGrid({

        height: "90%",

        width: "100%",

        filtering: true,

        inserting: true,

        editing: true,

        sorting: true,

        paging: true,

        autoload: true,

        pageSize: 15,

        pageButtonCount: 5,

        deleteConfirm: "Вы действительно хотите удалить запись о выдаче?",

        pagerFormat: "Страница: {first} {prev} {pages} {next} {last}    {pageIndex} из {pageCount}",

        pagePrevText: "Предыдущая",

        pageNextText: "Следующая",

        pageFirstText: "Первая",

        pageLastText: "Последняя",

        controller: {

            loadData: function (filter) {

*return* $.ajax({

                    type: "GET",

                    url: "/pages/rent/",

                    data: filter

                });

            },

            insertItem: function (item) {

*return* $.ajax({

                    type: "POST",

                    url: "/pages/rent/",

                    data: item

                });

            },

            updateItem: function (item) {

*return* $.ajax({

                    type: "PUT",

                    url: "/pages/rent/",

                    data: item

                });

            },

            deleteItem: function (item) {

*return* $.ajax({

                    type: "DELETE",

                    url: "/pages/rent/",

                    data: item

                });

            }

        },

        fields: [

            { name: "id", title: "ID", type: "number", width: 60, editing: false },

            { name: "id\_books", title: "ID книги", type: "number", width: 80, validate: "required" },

            { name: "id\_readers", title: "ID читателя", type: "number", width: 80, validate: "required" },

            {

                name: "date\_rent", title: "Дата выдачи", type: "text", width: 80,

                validate: function (value, item) {

*return* /*^*(19[0-9]{2}|2[0-9]{3})\-(0[1-9]|1[0-2])\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])*$*/.test(value);

                }

            },

            {

                name: "date\_return", title: "Дата возврата", type: "text", width: 80,

                validate: function (value, item) {

*return* /*^*(19[0-9]{2}|2[0-9]{3})\-(0[1-9]|1[0-2])\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])*$*/.test(value);

                }

            },

            { type: "control" }

        ]

    });

});

admin/books.js

$(function () {

    $("#jsGrid").jsGrid({

        height: "90%",

        width: "100%",

        filtering: true,

        inserting: true,

        editing: true,

        sorting: true,

        paging: true,

        autoload: true,

        pageSize: 12,

        pageButtonCount: 5,

        deleteConfirm: "Вы действительно хотите удалить книгу?",

        pagerFormat: "Страница: {first} {prev} {pages} {next} {last}    {pageIndex} из {pageCount}",

        pagePrevText: "Предыдущая",

        pageNextText: "Следующая",

        pageFirstText: "Первая",

        pageLastText: "Последняя",

        controller: {

            loadData: function (filter) {

*return* $.ajax({

                    type: "GET",

                    url: "/pages/admin\_books/",

                    data: filter

                });

            },

            insertItem: function (item) {

*return* $.ajax({

                    type: "POST",

                    url: "/pages/admin\_books/",

                    data: item

                });

            },

            updateItem: function (item) {

*return* $.ajax({

                    type: "PUT",

                    url: "/pages/admin\_books/",

                    data: item

                });

            },

            deleteItem: function (item) {

*return* $.ajax({

                    type: "DELETE",

                    url: "/pages/admin\_books/",

                    data: item

                });

            }

        },

        fields: [

            { name: "id", title: "ID", type: "number", width: 60, editing: false },

            { name: "name", title: "Название книги", type: "text", width: 160, validate: "required" },

            { name: "id\_authors", title: "ID автора", type: "text", width: 100, validate: "required" },

            { name: "id\_categories", title: "ID направления", type: "number", width: 160, validate: "required" },

            { name: "id\_publish", title: "ID издательства", type: "number", width: 110, validate: "required" },

            {

                name: "date\_create", title: "Дата издания", type: "text", width: 80,

                validate: function (value, item) {

*return* /*^*(19[0-9]{2}|2[0-9]{3})\-(0[1-9]|1[0-2])\-(0[1-9]|1[0-9]|2[0-9]|3[0-1])*$*/.test(value);

                }

            },

            { name: "number\_pages", title: "Количество страниц", type: "number", width: 95, validate: "required" },

            { type: "control" }

        ]

    });

});

client/books.js

$(function () {

    $("#jsGrid").jsGrid({

        height: "90%",

        width: "100%",

        filtering: true,

        sorting: true,

        paging: true,

        autoload: true,

        pageSize: 6,

        pageButtonCount: 5,

        pagerFormat: "Страница: {first} {prev} {pages} {next} {last}    {pageIndex} из {pageCount}",

        pagePrevText: "Предыдущая",

        pageNextText: "Следующая",

        pageFirstText: "Первая",

        pageLastText: "Последняя",

        controller: {

            loadData: function (filter) {

*return* $.ajax({

                    type: "GET",

                    url: "/pages/client\_books/",

                    data: filter

                });

            }

        },

        fields: [

            { name: "id", title: "ID", type: "number", width: 40 },

            { name: "name", title: "Название книги", type: "text", width: 120, validate: "required" },

            { name: "authors", title: "Авторы", type: "text", width: 140, validate: "required" },

            { name: "categories", title: "Направление", type: "text", width: 120, validate: "required" },

            { name: "publish", title: "Издательство", type: "text", width: 80, validate: "required" },

            { name: "date\_create", title: "Дата издания", type: "text", width: 70, validate: "required" },

            { name: "number\_pages", title: "Количество страниц", type: "number", width: 85, validate: "required" },

            { type: "control", deleteButton: false, editButton: false }

        ]

    });

});