МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Применение технологии полнотекстового поиска для задачи

«Книжный магазин»

Выполнил студент Лобан Виталий Сергеевич

(Ф.И.О.)

Руководитель проекта ст. пр. Жигаровская С. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В. В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты ст. пр. Жигаровская С. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер ст. пр. Жигаровская С. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2018

**Реферат**

Пояснительная записка курсового проекта содержит 31 страниц, 2 рисунков, 2 листингов, 5 источников литературы, 2 приложения.

РЕПЛИКАЦИЯ, БАЗА ДАННЫХ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, SQL, T-SQL, MICROSOFT SQL SERVER

Основными целями курсового проекта являются: проектирование базы данных для магазина книг.

Пояснительная записка состоит из введения, семи разделов, заключения.

Во введении представлена общая информация, дающая представление о предстоящей работе, определены цели.

В первом раздели описываются поставленные задачи.

Во втором разделе рассматривается архитектура базы данных.

В третьем разделе представлен процесс и результаты разработки объектов базы данных.

В четвёртом разделе описывается создание процедур для экспорта и импорта.

В пятом разделе представлено тестирование производительности.

В шестом разделе описана технология полнотекстового поиска.

В седьмом разделе представлено руководство пользователя.

В заключении представлены итоги курсового проектирования и задачи, которые были решены в ходе проектирования и разработки базы данных.

**Содержание**

[Введение 5](#_Toc531787778)

[1. Постановка задачи 6](#_Toc531787779)

[2. Разработка модели базы данных 7](#_Toc531787780)

[3. Разработка необходимых объектов 9](#_Toc531787781)

[3.1. Таблицы 9](#_Toc531787782)

[3.2. Пользователи 9](#_Toc531787783)

[3.3. Процедуры и функции 9](#_Toc531787784)

[4. Описание процедур импорта и экспорта данных 12](#_Toc531787785)

[4.1. Описание процедуры импорта данных 12](#_Toc531787786)

[4.2. Описание процедуры экспорта данных 12](#_Toc531787787)

[5. Тестирование производительности 13](#_Toc531787788)

[5.1. Тестирование производительности базы данных 13](#_Toc531787789)

[6. Описание технологии 15](#_Toc531787790)

[6.1. Описание применения технологии 15](#_Toc531787791)

[7. Руководство пользователя 19](#_Toc531787792)

[Заключение 26](#_Toc531787793)

[Спсок используемых источников 27](#_Toc531787794)

[ПРИЛОЖЕНИЕ А 28](#_Toc531787795)

[ПРИЛОЖЕНИЕ Б 29](#_Toc531787796)

# Введение

На сегодняшний день применение баз данных приобрело весьма важное значение для многих организаций, которые для упрощения своей работы применяют компьютерные технологии.

Базы данных стали основой информационных систем и в корне изменили методы работы многих организаций. В частности, в последние годы развитие технологии баз данных привело к созданию весьма мощных и удобных в эксплуатации систем. Благодаря этому системы баз данных стали доступными широкому кругу пользователей.

Большинство, если не все бизнес-приложения предназначены для обработки бизнес-данных. Самые первые бизнес-решения так и назывались — программы обработки данных. Эффективное хранение, обработка и взаимодействие с данными — только важная составляющая управления предприятием, что компании инвестируют значительные средства в разработку компьютеризированных системы для эффективного решения этих задач. Один из способов повышения эффективности обработки данных — организовать их эффективное хранение и получение. Самый распространенный подход к хранению данных на сегодня — использовать реляционную базу данных.

БД – это, прежде всего, хранилище объектов данных, т.е. набор возможных понятий или событий, описываемых базой данных, с возможностью поиска этих объектов по признакам. Базой данных можно считать не только таблицы, индексирующие файлы со знаниями разных форматов, но и сами эти файлы, потому, что они являются не типизированными хранилищами знаний в такой базе данных. БД могут применяться как вспомогательное средство, позволяющее реализовать какую-то полезную функцию. Например, хранение настроек программы, Internet - адресов для рассылки рекламы и т.д.

# Постановка задачи

В настоящее время, несмотря на повышение компьютеризации общества, в сфере бизнеса и торговли до сих пор нет средств, позволяющих в достаточной мере автоматизировать процесс ведения документации и отчетности.

Одной из основных задач можно рассматривать проблему ведения отчетности, а также оперативную корректировку данных при возникновении необходимости в этом.

О своевременности и актуальности рассматриваемой проблемы говорит тот факт, что большую часть своего времени администрация магазина тратит на оформление различной документации и отчетов. Огромное количество магазинов и отсутствие предложений в данной сфере гарантирует высокую потребность в данном продукте.

Объектом для создания базы данных являлся книжный магазин. Данная база данных предлагает хранения, ввод и корректировку данных.

В соответствии с заданием курсового проекта следует создать базу данных (БД) для магазина книг с использованием программного продукта Microsoft SQL Server 2017. Так же надо создать клиентское приложение, представляющее собой интерфейс для взаимодействия с базой данных.

# Разработка модели базы данных

Для создания, хранения и управления данными, была использованная реляционная СУБД Microsoft SQL Server. Ниже на рисунке 2 .1 представлена модель базы данных (таблицы, отмеченные на них первичных и вторичных ключей) курсового проекта. База данных была подвергнута процедуре нормализации данных.

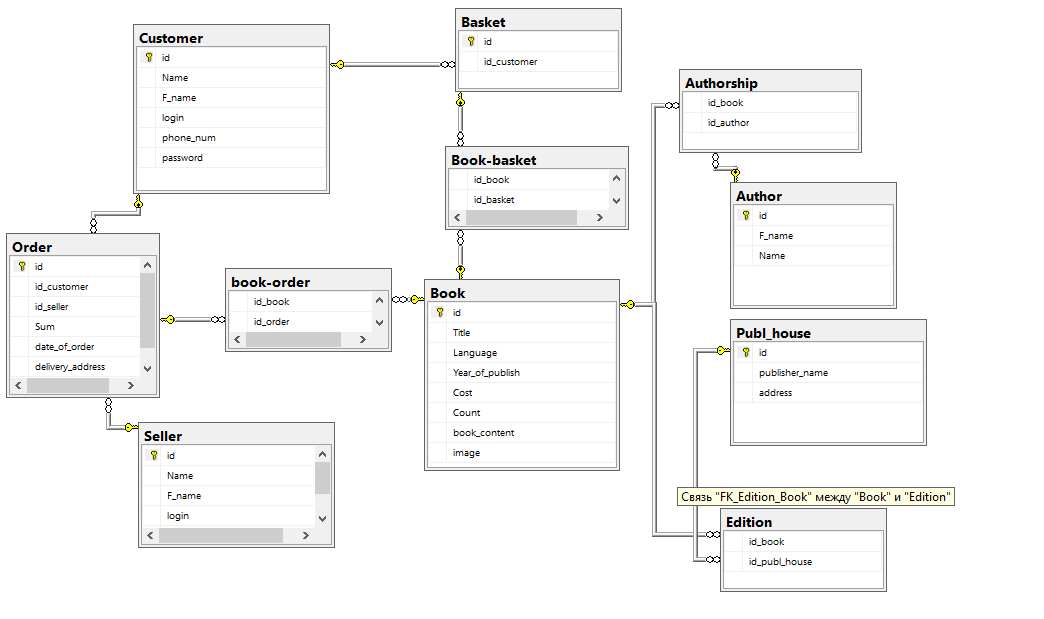


Рисунок 2.1 – Диаграмма модели базы данных

База данных на выше представленной схеме в полной мере описывает необходимые таблицы для решения поставленной задачи данного курсового проекта.

Таблица Book содержит информацию о книгах:

* Id – первичный ключ, идентификатор книг;
* Title – заголовок книги;
* language – язык;
* Year\_of\_bublish – год издания;
* Cost – стоимость;
* Count – количество;
* Book\_content – содержание.
* Image – ссылка на картинку;

Таблица Author содержит информацию об авторах:

* Id – первичный ключ, идентификатор автора;
* F\_name – фамилия;
* Name – имя;

Таблица Publ\_house содержит информацию об издательствах:

* Id – первичный ключ, идентификатор издательства;
* Publisher\_name – название издательства;
* Address – адрес;

Таблица Authorship содержит связи авторов и книг:

* Id\_book – идентификатор книги ;
* Id\_author – идентификатор автора;

Таблица Basket содержит данные о корзине пользователя:

* Id –первичный ключ, идентификатор корзины;
* Id\_customer – идентификатор пользователя;

Таблица Book-basket содержит связи корзин с книгами:

* Id\_book – идентификатор книги;
* Id\_basket – идентификатор корзины;
* Count – количество данных книг в корзине;

Таблица Book-order содержит связи заказов с книгами:

* Id\_book – идентификатор книги;
* Id\_order – идентификатор заказа;
* Count – количество данных книг в заказе;

Таблица Customer содержит данные о пользователях:

* Id – первичный ключ, идентификатор пользователя;
* Name – имя пользователя;
* F\_name – фамилия;
* Login – логин для входа;
* Phone\_num – номер телефона;
* Password – пароль для входа;

Таблица Edition содержит связи издательств и книг:

* Id\_book – идентификатор книги ;
* Id\_publ\_house – идентификатор издетельства;

Таблица Order содержит данные о заказах:

* Id– первичный ключ, идентификатор заказа;
* Id\_customer – идентификатор заказчика;
* Id\_seller – идентификатор продавца;
* Sum – стоимость заказа;
* Date\_of\_order – дата заказа;
* Delivery\_address – адрес доставки;
* Status – статус заказа;

Таблица Seller содержит данные о продавцах:

* Id – первичный ключ, идентификатор продавца;
* Name – имя продавца;
* F\_name – фамилия;
* Login – логин для входа;
* Phone\_num – номер телефона;
* Password – пароль для входа;

# Разработка необходимых объектов

База данных данного курсового проекта содержит следующие объекты: таблицы, хранимые процедуры, пользователи. Их подробное описание представлено ниже.

## Таблицы

База данных книжного магазина содержит 11 таблиц, которые описаны в главе 2.

## Пользователи

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. При проектировании базы данных понадобилось 2 пользователя (admin и user). У каждого пользователя есть свои права, предназначенные для выполнения соответствующих ему требований.

Пользователь admin наделен правами, имитирующими администратора, который может добавлять, изменять и удалять книги, подтверждать заказы.

Пользователь user наделен правами, имитирующего пользователя, который может покупать книги, а также смотреть историю заказов.

Скрипты назначения привилегий пользователям приведены в приложении А.

## Процедуры и функции

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Их использование в курсовом проекте предназначено для ограничения доступа пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение некоторых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

Процедуры, разработанные на данном курсовом проекте можно разделить на два типа: процедуры CRUD (Create, Read, Update, Delete) для каждой таблицы (используя DML - операторы) и процедуры выборки по таблицам.

Администратору доступны следующие процедуры и функции:

* Auth\_exist – проверка существует ли автор с указанным именем;
* Books\_from\_order – список книг из заказа;
* Change\_pass\_admin – изменить пароль;
* Check\_count\_books – проверить количество книг;
* Check\_password\_adm – аутентификация продавцов ;
* Check\_user\_adm – проверка существует ли админ с данным логином;
* Cust\_inf– информация о покупателе;
* Del\_book – удаление книги;
* Get\_auth\_on\_id – получить всех авторов соответствующей книги;
* Get\_books – получить список книг;
* Get\_curent\_adm– получить id текущего продавца;
* Get\_orders – список заказов;
* Get\_publ\_on\_id – получить всех издательств соответствующей книги;
* Get\_sellers – получить список продавцов из xml;
* Import\_sellers – импорт продавцов в xml;
* Ins\_authorship – добавить автора к книге;
* Ins\_edition – добавить издательство к книге;
* Ins\_adm – добавить продавца;
* Insert\_author– добавить автора;
* Insert\_book – добавить книгу;
* Insert\_p\_house – добавить издательство;
* isExist\_adm – проверка существует ли продавец с данным логином;
* p\_house\_exist – существует ли данное издательство;
* Update\_book – изменение книг;
* Update\_book\_count – изменить количество имеющихся книг;
* Book\_ord\_fromXml – импорт данных из xml;
* Export\_book\_from\_order\_ToXML – экспорт данных в xml;
* Update\_order – изменить статус заказа.

Пользователю доступны следующие процедуры и функции:

* Add\_book\_in\_order – добавить книгу в корзину;
* Add\_order – оформить заказ;
* Books\_from\_bask – список книг в корзине;
* Books\_fom\_order – список книг в заказе;
* Change\_pass – изменить пароль;
* Check\_password – проверить пароль;
* Check\_user – аутентификация пользователей;
* Create\_basket – создать корзину;
* Del\_from\_bask – удалить из корзины;
* Find\_auth – поиск автора;
* Find\_content – создание содержимого;
* Find\_title – поиск названия книги;
* Get\_auth\_on\_id – получить автора по id;
* Get\_book\_count – получить количество книг;
* Get\_book\_on\_id – получить книгу по id;
* Get\_books – список книг;
* Get\_curent – получить id текущего пользователя;
* Get\_curr\_bask – получить корзину данного пользователя;
* Get\_order\_by\_id – заказы текущего пользователя;
* Grt\_publ\_on\_id – список издательств данной книги;
* Insert\_customer – добавить пользователя;
* isExist – проверка существует ли пользователь с данным логином;
* seller\_inf – информация о продавце;
* Update\_cout\_in\_bask – изменить количество книг в корзине;

Листинг SQL-кода процедур находится в приложении Б.

# Описание процедур импорта и экспорта данных

В данной курсовой работе реализованы процедуры экспорта и импорта данных из xml файла в базу данных таблицы Book-order и наоборот. При данных операциях работает с файловой системой приложение, а разбором и генерацией xml занимается sql server.

## Описание процедуры импорта данных

Существует несколько вариантов процедуры импорта данных xml в таблицы MS SQL Server.

Были выполнены две цели, для выполнения преобразования данных из xml:

* импорт данных xml в MS SQL Server с помощью функции OPENROWSET;
* преобразование формата xml с помощью BULK.

Код создания процедуры импорта представлен на рисунке 4.1.

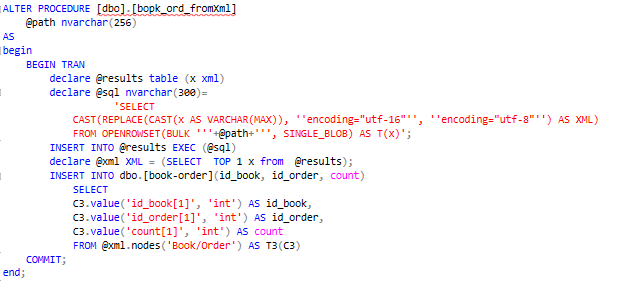


Рисунок 4.1 – процедура импорта из xml в таблицу Book-order

## Описание процедуры экспорта данных

Код создания процедуры экспорта данных таблицы в xml формат представлен на рисунке 4.2.

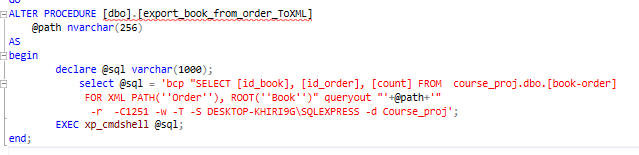


Рисунок 4.2 – процедура экспорта данных

# Тестирование производительности

Оптимизация запросов — процесс изменения запроса и/или структуры БД с целью уменьшения использования вычислительных ресурсов при выполнении запроса. Один и тот же результат может быть получен СУБД различными способами (планами выполнения запросов), которые могут существенно отличаться как по затратам ресурсов, так и по времени выполнения.

В MS SQL Server оптимизация запросом в основном заключается в построение индексов над таблицами, и изменением плана запроса. Индекс – это объект базы данных, предназначенный для ускорения запросов к данным в таблице базы данных. MSS поддерживает два типа индексов: кластеризованные и некластеризованные индексы.

При создании кластеризованного индекса данные индексируемой таблицы располагаются в физическом порядке, соответствующем индексу, и становятся частью кластеризованного индекса. Поэтому кластеризованный индекс для таблицы может быть создан только один.

Некластеризованный индекс – это отдельный объект, имеющий указатели на строки таблицы. Максимальное количество некластерированных индексов для одной таблицы не должно превышать 1000.

## Тестирование производительности базы данных

Для тестирования базы данных была выбрана таблица Author. В базе данных данного курсового проекта в каждой таблице находится поле с уникальным значением поля — id типа int, соответственно каждая таблица содержит кластеризованный индекс. База данных содержит множество процедур с выборкой содержащий оператор WHERE сравнивающий строки. Наиболее частое сравнение — это имя и фамилия автора для проверки на корректность запроса от пользователя. Даже при значительном заполнении базы данных (около 100000 записей в таблице) время выборки не занимает много времени, так как в таблице существует полнотекстовый индекс. Ниже на рисунке 5.1 представлена карта запроса при выборке в таблице Author.

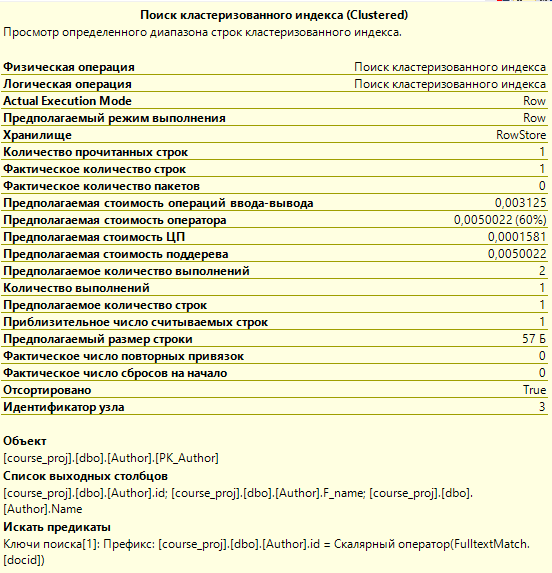


Рисунок 5.1 – карта запроса

# Описание технологии

В данном курсовом проекте была выбрана технология полнотекстового поиска. Полнотекстовый поиск — это поиск слов или фраз в текстовых данных. Обычно такой вид поиска используется для поиска текста в большом объёме данных, так как он значительно быстрей обычного поиска, который можно осуществить, используя конструкцию LIKE.

## Описание применения технологии

Полнотекстовый поиск подразумевает создание специального индекса текстовых данных, который представляет собой некий словарь слов, которые встречаются в этих данных. Так как помимо своей быстрой работы он обладает ещё и возможностью ранжировать найденные строки, т.е. выставлять ранг каждой найденной строке. Можно найти самые релевантные записи, т.е. самые подходящие под запрос.

Возможности полнотекстового поиска в MS SQL Server :

* в полнотекстовом поиске SQL сервера можно осуществлять поиск не только по отдельным словам или фразам, но и по префиксным выражениям, например, задать текст начала слова или фразы;
* можно искать слова по словоформам, например, различные формы глаголов или существительные в единственном и во множественном числе, т.е. по производным выражениям;
* можно построить запрос так, чтобы найти слова или фразы, находящиеся рядом с другими словами или фразами, т.е. выражения с учетом расположения;
* есть возможность искать синонимические формы конкретного слова.

Перед тем как приступать к созданию полнотекстового поиска, необходимо знать несколько важных моментов:

* для реализации полнотекстового поиска компонент Full-Text Search (Полнотекстовый поиск) должен быть установлен;
* у таблицы может быть только один полнотекстовый индекс;
* чтобы создать полнотекстовый индекс, таблица должна содержать один уникальный индекс, который включает один столбец и не допускает значений NULL. Рекомендовано использовать уникальный кластеризованный индекс (или просто первичный ключ), первый столбец которого должен иметь целочисленный тип данных;
* полнотекстовый индекс можно создавать на столбцах с типом данных: char, varchar, nchar, nvarchar, text, ntext, image, xml, varbinary или varbinary(max).

Для того чтобы создать полнотекстовый индекс сначала необходимо создать полнотекстовый каталог. Начиная с SQL Server 2008 полнотекстовый каталог – это логическое понятие, обозначающее группу полнотекстовых индексов.

Полнотекстовый каталог можно создать и в графическом интерфейсе Management Studio. Для этого открываем базу данных, переходим в папку Хранилище, затем Полнотекстовые каталоги и выбираем «Создать полнотекстовый каталог». Откроется окно создания, представленное на рисунке 6.1.

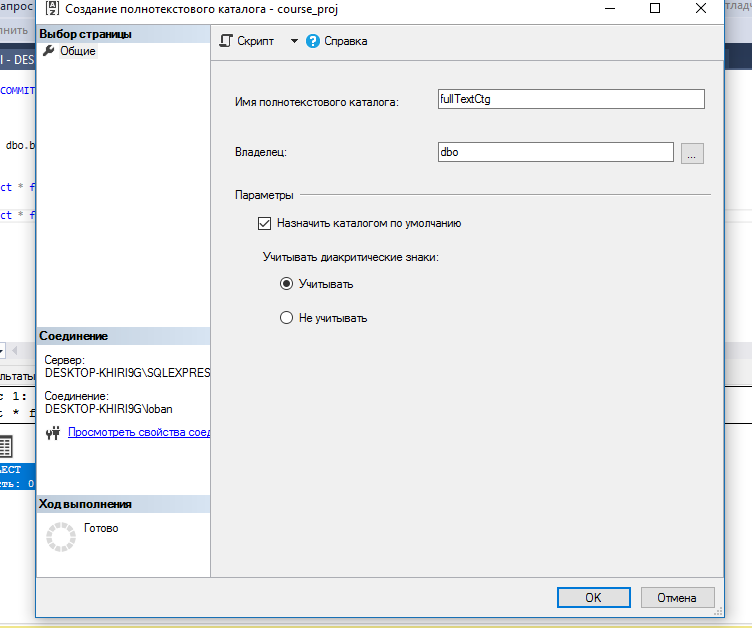


Рисунок 6.1 – окно создания полнотекстового каталога

После создания полнотекстового каталога можно начинать создавать в нем полнотекстовые индекса. В нашем случае мы хотим создать полнотекстовый индекс, в котором участвует поле F\_name и Name таблицы Author, а также Book\_content и Title таблицы Book.

Полнотекстовый индекс можно создать, используя и графические инструменты, для этого открываем свойства полнотекстового каталога и переходим в пункт «Таблицы или представления», выбираем нужную таблицу, поле, уникальный индекс и способ отслеживания изменений.

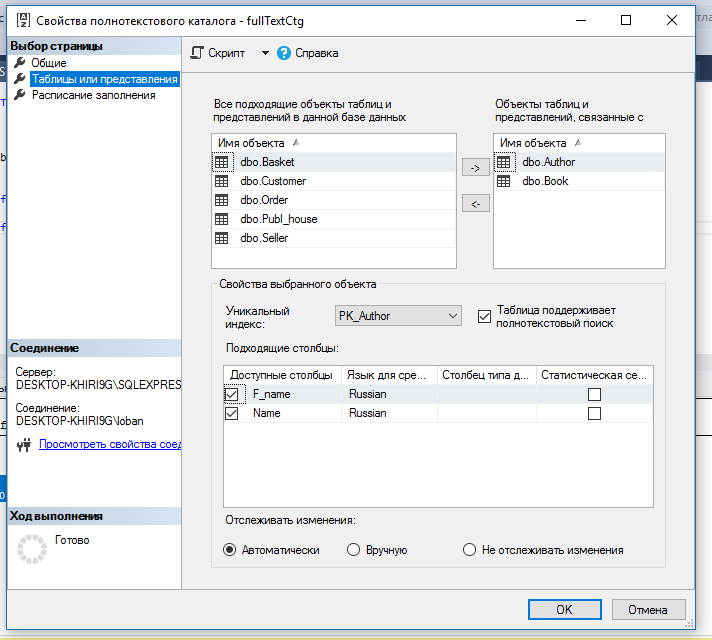


Рисунок 6.2 – создание полнотекстового индекса таблицы Author

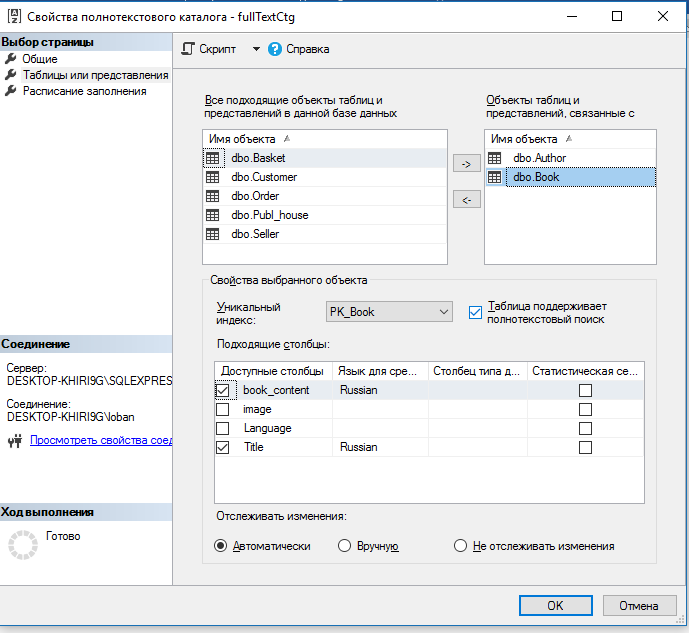


Рисунок 6.3 – создание полнотекстового индекса таблицы Book

Таким образом, мы выполнили создание полнотекстового каталога и индекса.

В качестве примера и подтверждения того, что наш полнотекстовый поиск работает, на рисунке 6.4 представлен пример выполнения полнотекстового запроса.

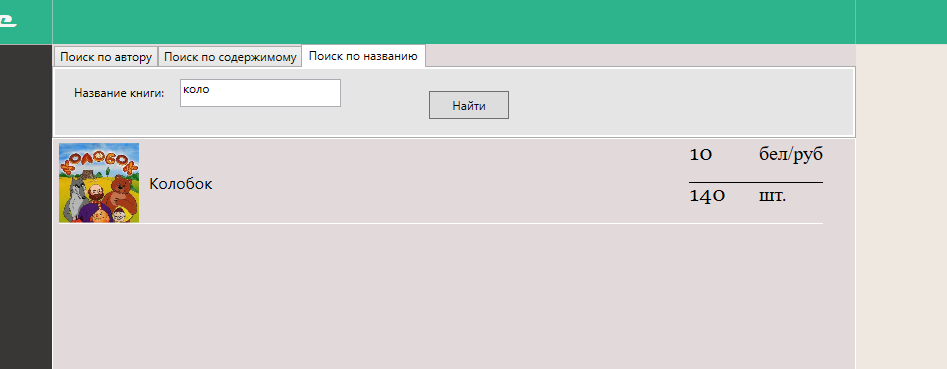


Рисунок 6.4 – пример полнотекстового запроса

Для осуществления запросов могут использоваться различные предикаты и функции. CONTAINS/CONTAINSTABLE и FREETEXT/FREETEXTTABLE полезны для различных типов сопоставления, поиска точных и неточных соответствий отдельных слов и фраз. Полнотекстовые предикаты CONTAINS и FREETEXT используются в предложении WHERE или HAVING инструкции SELECT. С помощью функций CONTAINS, CONTAINSTABLE, FREETEXT или FREETEXTTABLE можно найти все грамматические формы глаголов и существительных. Код запроса представлен на рисунке 6.5.

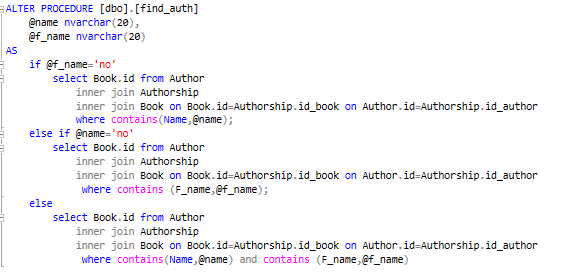


Рисунок 6.5 – пример кода полнотекстового запроса

# Руководство пользователя

Пользовательское приложение предоставляет собой десктоп приложение для взаимодействия с базой данных для пользователей администратор и пользователь. Оно было реализовано с помощью языка C#.

Начальной страницей приложения является окно авторизации, представленное на рисунке 7.1.

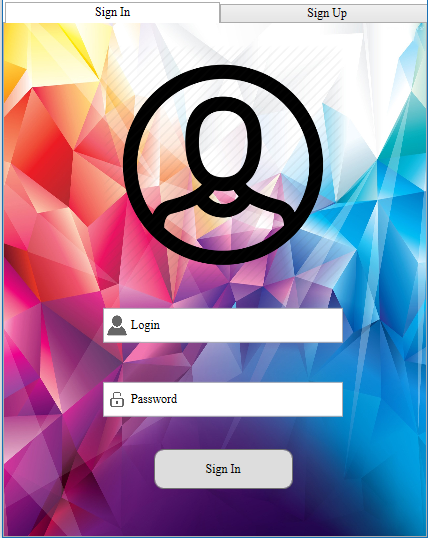


Рисунок 7.1 – Начальная страница

Если пользователь ещё не авторизован, то следует перейти на вкладку «Sign Up» и откроется окно, представленное на рисунке 7.2. После заполнения всех полей и нажатия кнопки «Sign Up» система перекинет пользователя на главное окно, представленное на рисунке 7.3.

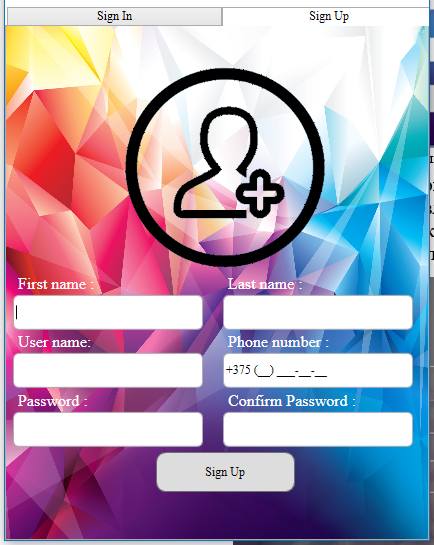


Рисунок 7.2 – Создание аккаунта

Далее, в зависимости от того, кто входит в приложение (пользователь или администратор), открывается соответствующее меню (рисунки 7.3– 7.4).

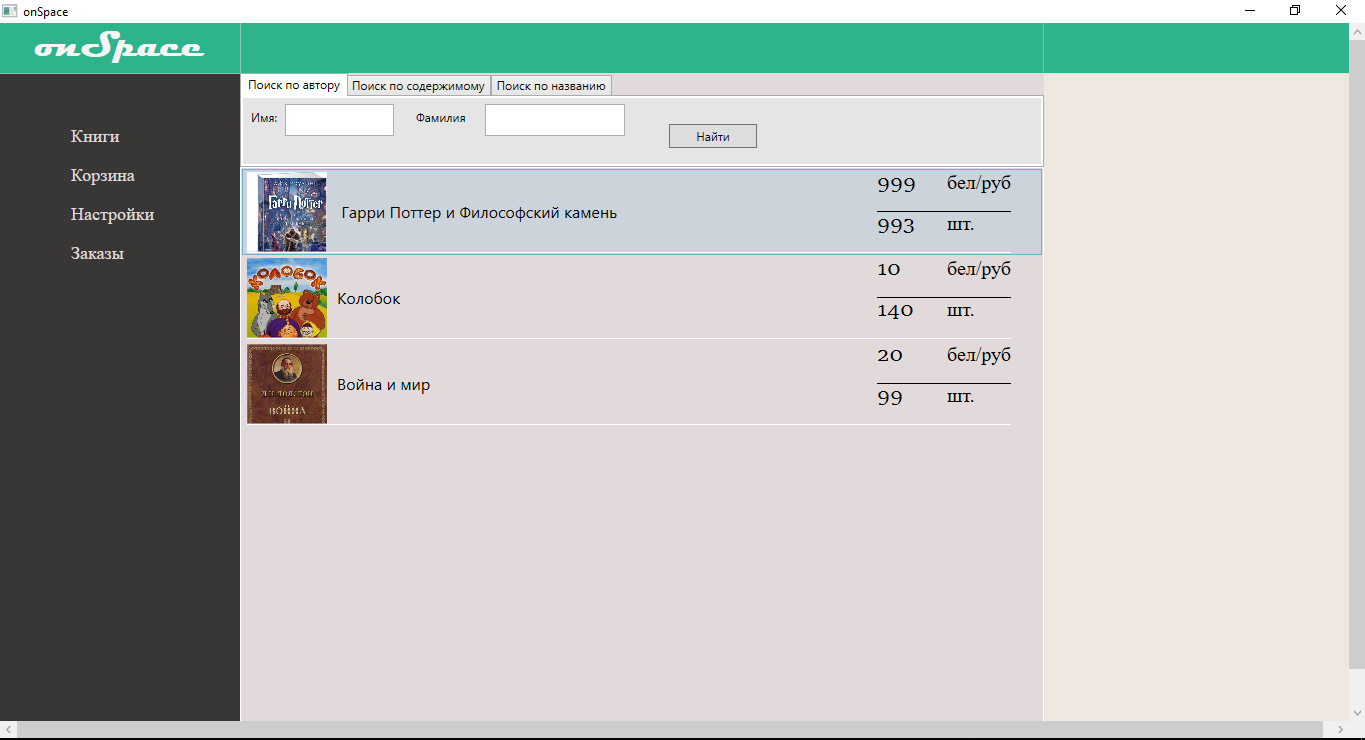


Рисунок 7.3 – Меню пользователя



Рисунок 7.4 – Меню администратора

Различие меню состоит в том, что пользователь может просматривать, искать, добавлять в корзину и заказывать книги, а также смотреть свою историю. Администратор может удалять, изменять и добавлять книги, а также подтверждать заказы. Далее подробно описано содержание каждого пункта меню.

Рассмотрим меню пользователя. Оно содержит в себе такие вкладки, как: «книги», «корзина», «настройки», «заказы».

Для поиска или просмотра всех книг магазина необходима нажать вкладку «Книги». Откроется окно, представленное на рисунке 7.5.

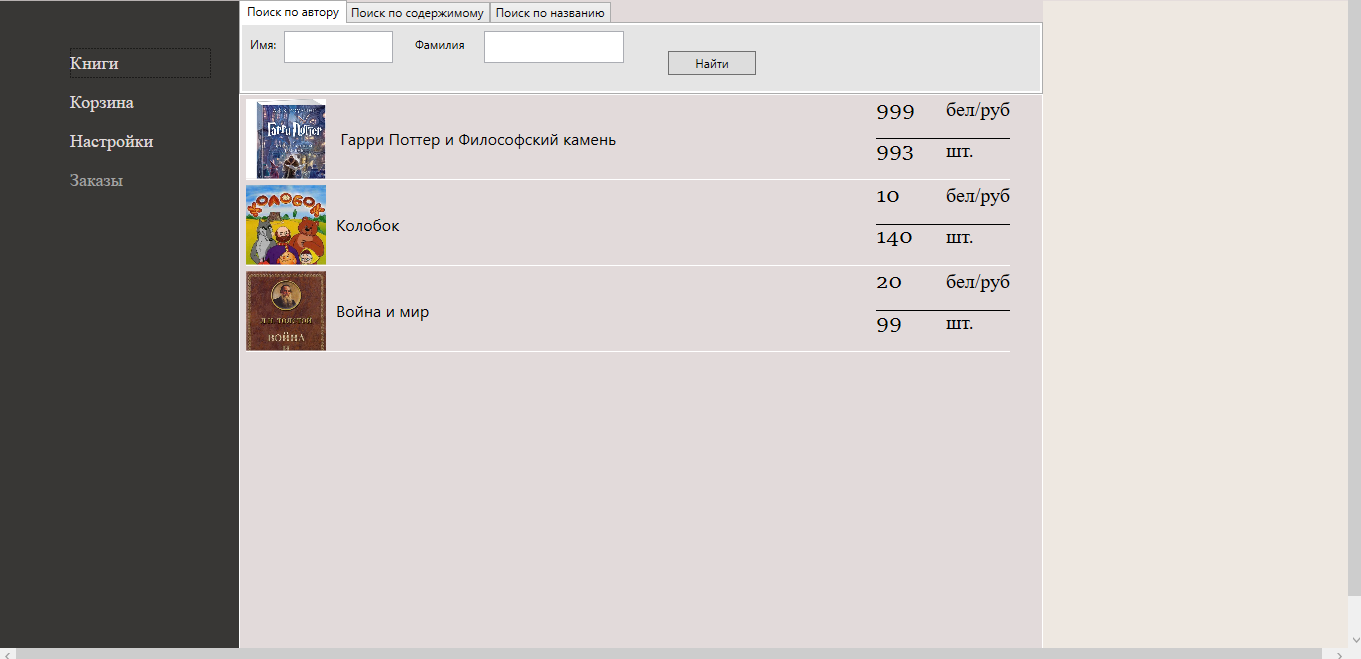


Рисунок 7.5 – Страница с книгами

При нажатии на книгу появится окно добавления книг в корзину (рисунок 7.6).

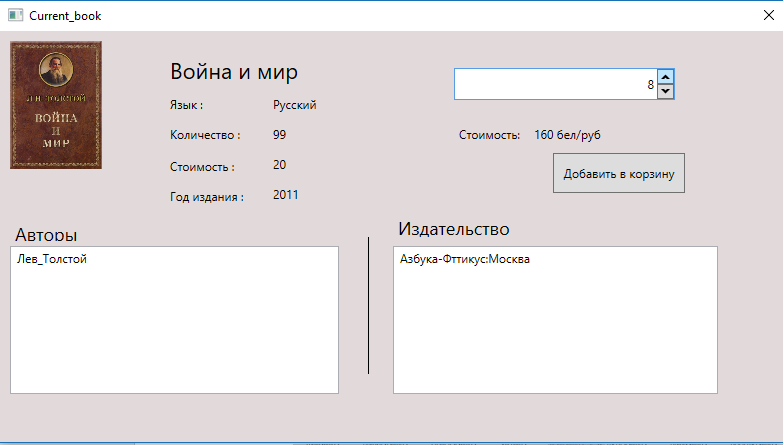


Рисунок 3.6 – Страница добавления в корзину

После того как пользователь добавит книги он может работать со своей корзиной. К примеру: он может удалить книги из корзины, изменить их количество или, указав свой адрес, оформить заказ (рисунок 7.8 – 7.9).

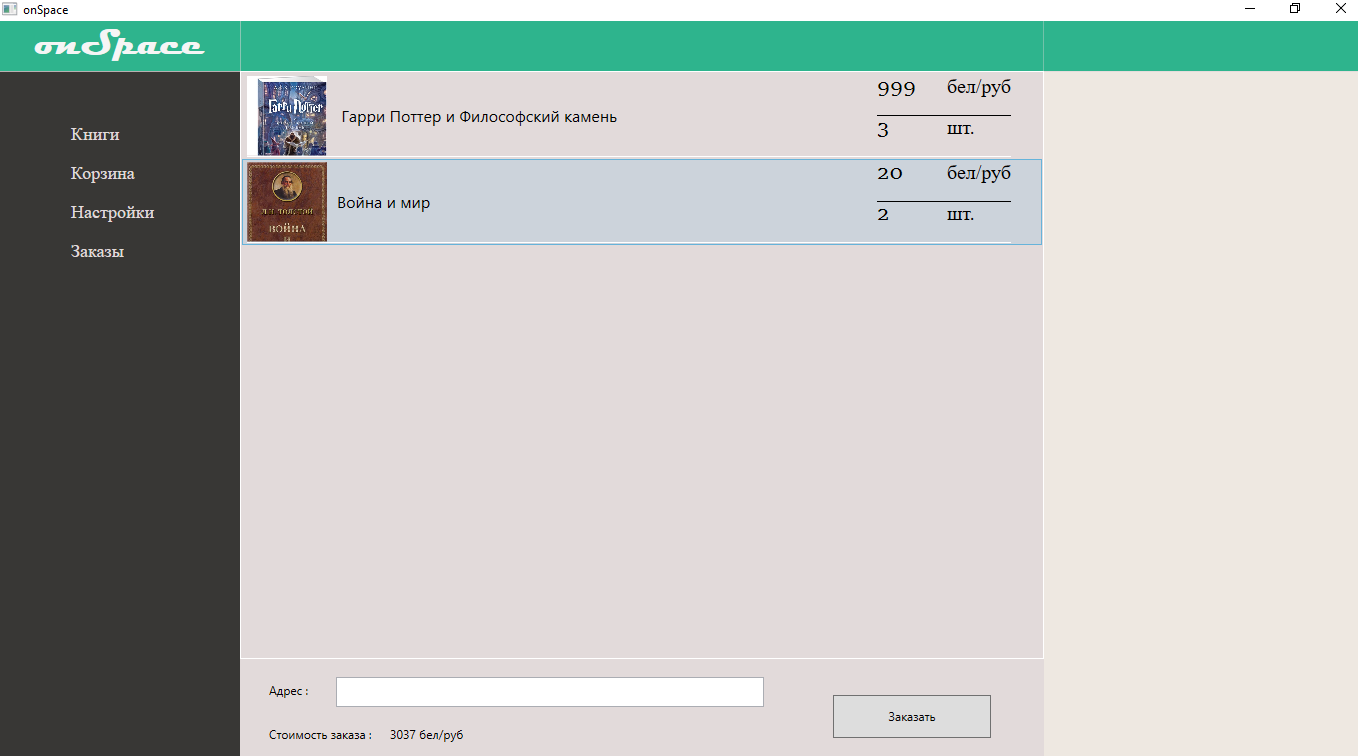


Рисунок 7.8 – Работа с корзиной

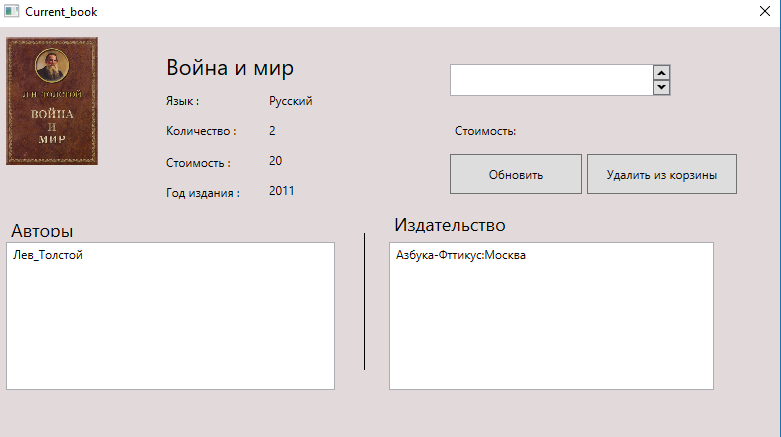


Рисунок 7.9 – Работа с корзиной

Для изменения текущего пароля пользователя необходимо нажать на вкладку «настройки», ввести свой старый пароль и новый. При желании на этой же странице можно изменить тему приложения рисунок 7.10.

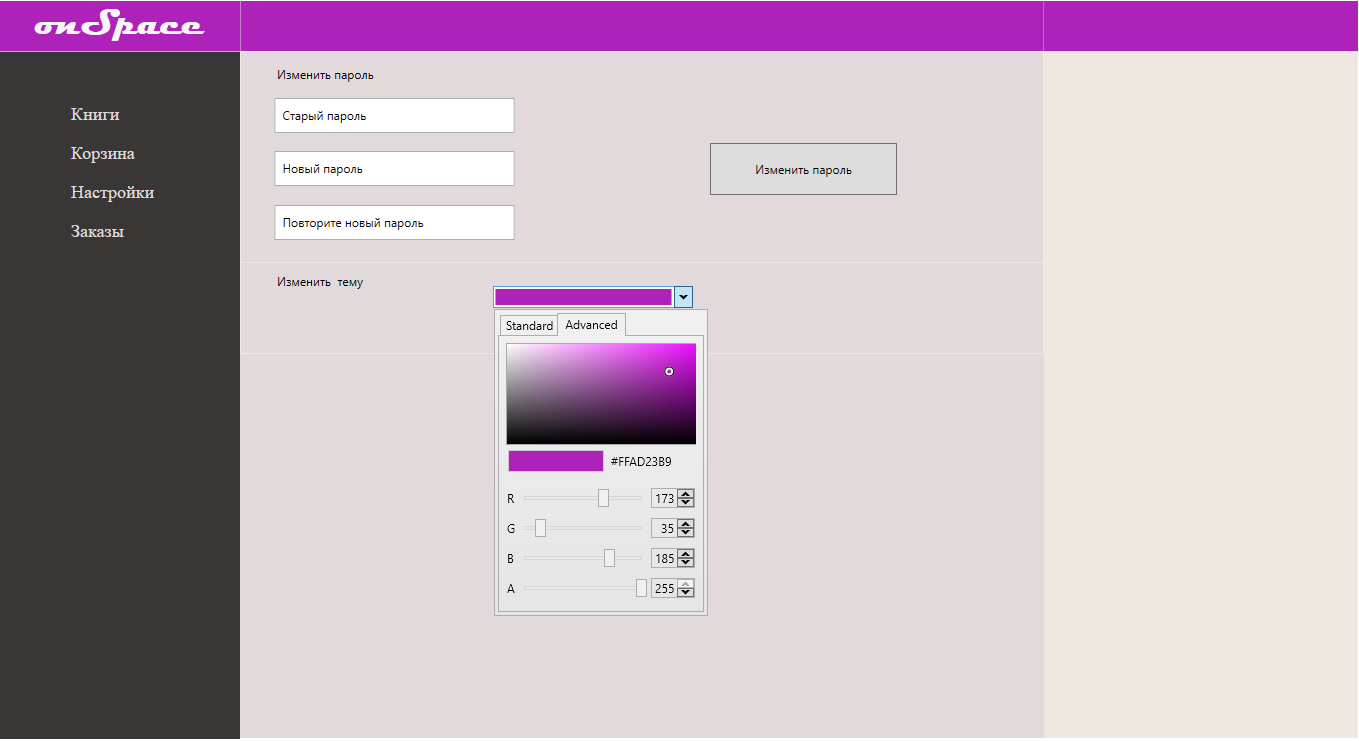


Рисунок 7.10 – Окно настройки

По нажатии на кнопку «Заказы» пользователю будет предоставлена история заказов, при выборе которых будет выведена полная информация о заказе (Рис. 7.11).

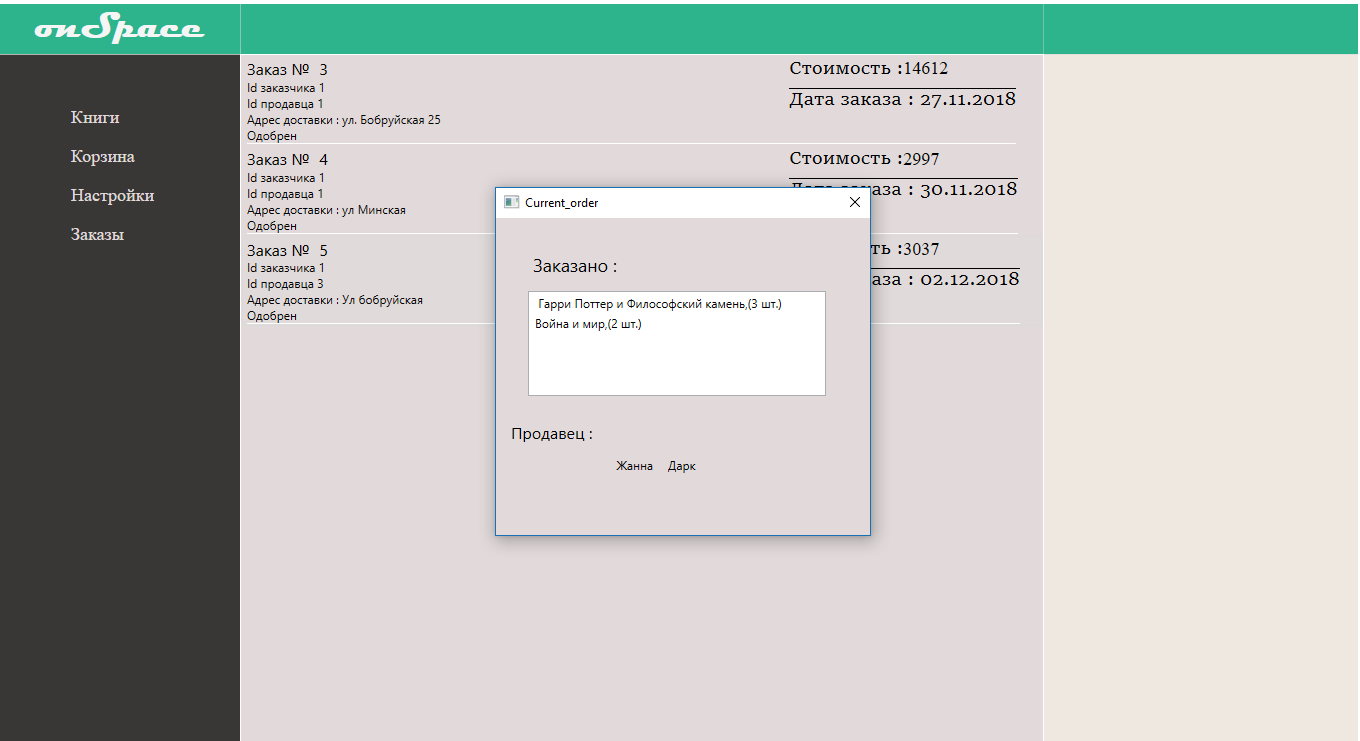


Рисунок 7.11 – Окно заказов

При входе в меню администратора добавляется одна новая вкладка «Добавить книги». При нажатии на неё откроется окно с пустыми полями, которые необходимо заполнить (Рис.7.12). Что бы изменить стандартное изображение книги следует нажать кнопку «Добавить изображение» (Рис. 7.13), после чего откроется диалоговое окно вставки изображения.

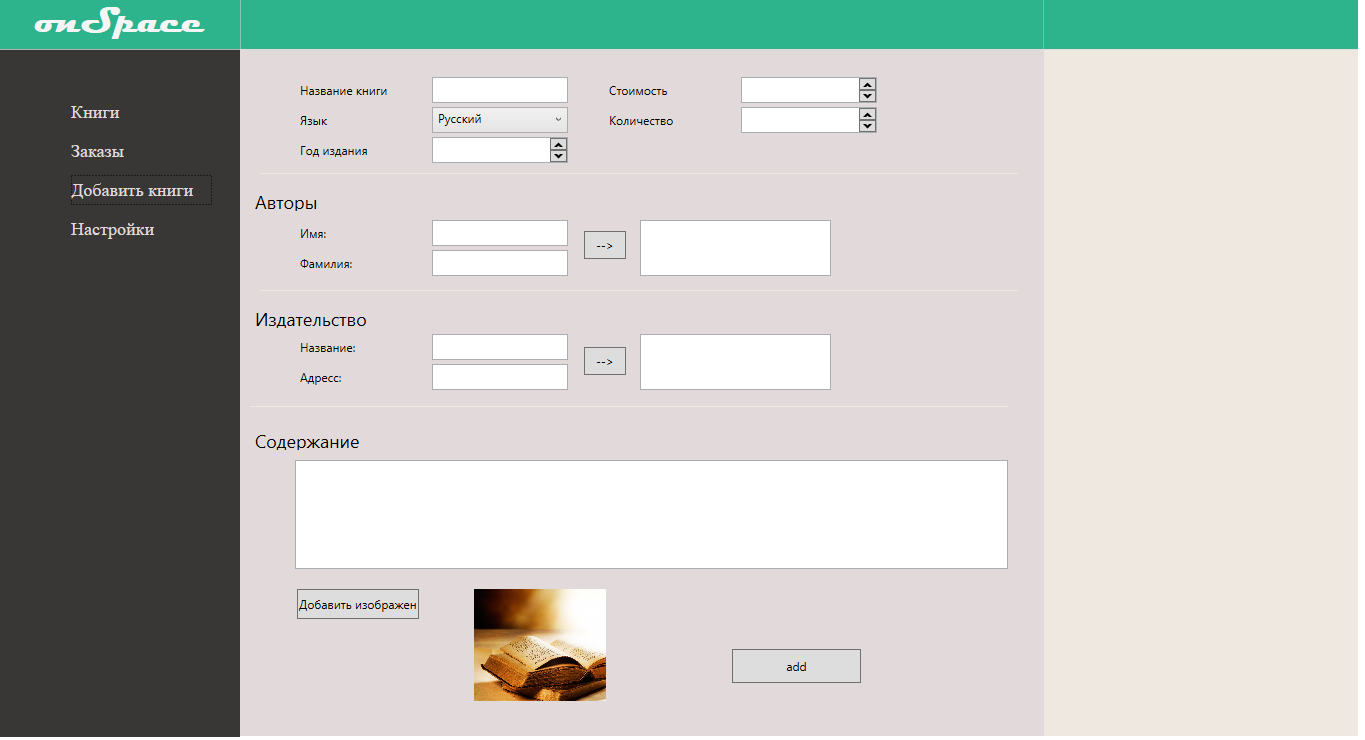


Рисунок 7.12 – Добавление книги

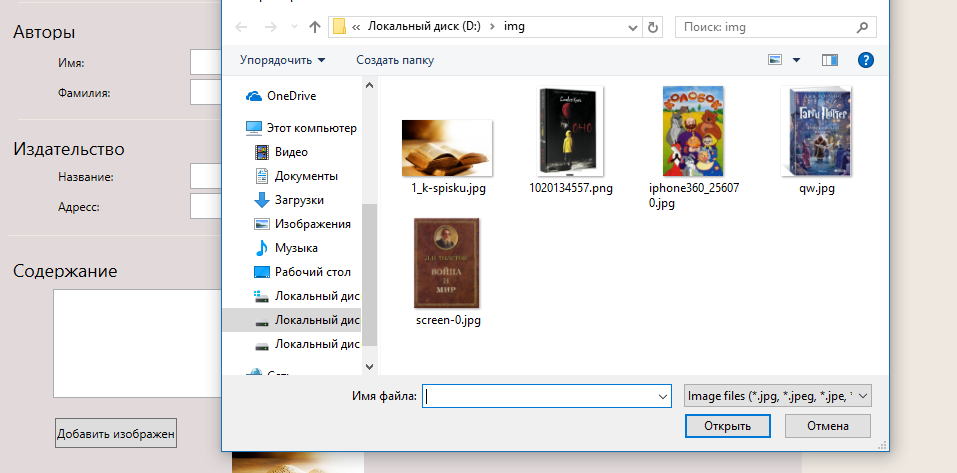


Рисунок 7.13 – Добавление изображения

При необходимости удалить или изменить информацию о существующей книге следует нажать кнопки «Книги» и выбрать нужную, после чего появится окно изменения количества и стоимости имеющихся книг (Рис.7.14).

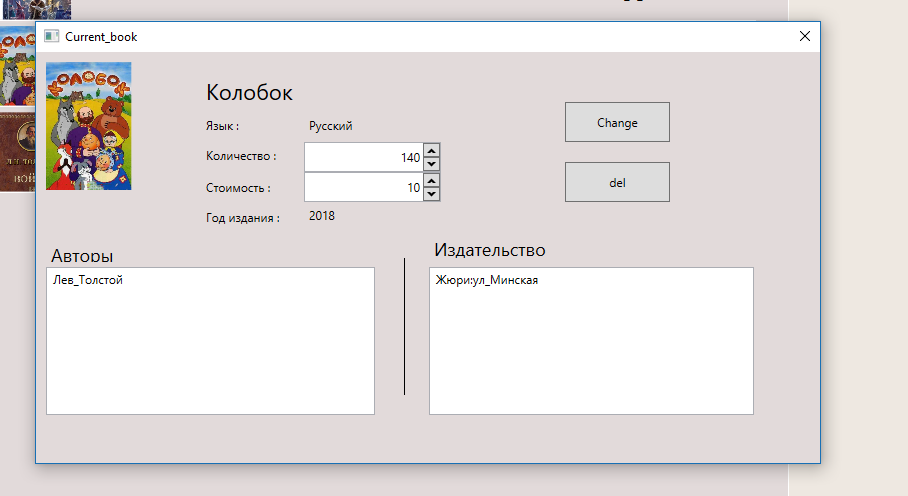


Рисунок 7.14 – Изменение и удаление книг

# Заключение

В данном курсовом проекте была разработана база данных для книжного магазина. Было разработано приложение для взаимодействия с базой данных. Использована технология полнотекстового поиска.

Разработка приложения – трудоемкое занятие, выполнение которого требует всестороннего знания той предметной области, к которой относится тема разрабатываемого приложения. Программная система имеет простой, интуитивно понятный интерфейс пользователя.

В соответствии с полученным результатом, можно сказать, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

.

# Спсок используемых источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова

2. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> – Дата доступа: 15.11.2018.

3. Примеры массового импорта и экспорта XML-документов (SQL Server) [Электронный ресурс] – режим доступа - [https://docs.microsoft.com/ru-ru/sql/relational-databases/import-export/examples-of-bulk-import-and-export-of-xml-documents-sql-server](https://docs.microsoft.com/ru-ru/sql/relational-databases/import-export/examples-of-bulk-import-and-export-of-xml-documents-sql-server%20Дата%20доступа%20–%2007.12.017)  Дата доступа 25.11.2018

4. METANIT.COM Сайт о программировании[Электронный ресурс] / Режим доступа: <https://metanit.com> Дата доступа 10.11.2018

5. Запрос с полнотекстовым поиском [Электронный ресурс]: https://docs.microsoft.com/ru-ru/sql/relational-databases/search/query-with-full-text-search?view=sql-server-2017 Дата доступа 26.11.2018.

# ПРИЛОЖЕНИЕ А

|  |
| --- |
| use course\_proj;  grant execute on dbo.add\_book\_in\_bask to [for\_user]  grant execute on dbo.Add\_book\_in\_order to [for\_user]  grant execute on dbo.Add\_order to [for\_user]  grant execute on dbo.Books\_from\_bask to [for\_user]  grant execute on dbo.Books\_fom\_order to [for\_user]  grant execute on dbo.Change\_pass to [for\_user]  grant execute on dbo.Check\_password to [for\_user]  grant execute on dbo.Check\_user to [for\_user]  grant execute on dbo.Create\_basket to [for\_user]  grant execute on dbo.Del\_from\_bask to [for\_user]  grant execute on dbo.Find\_auth to [for\_user]  grant execute on dbo.Find\_content to [for\_user]  grant execute on dbo.Find\_title to [for\_user]  grant execute on dbo.Get\_auth\_on\_id to [for\_user] |

Листинг 1 – наделение правами пользователей

# ПРИЛОЖЕНИЕ Б

|  |
| --- |
| ALTER PROCEDURE [dbo].[add\_book\_in\_bask]  @id\_bask int,  @id\_book int,  @count int  AS  if exists(select\*from [Book-basket] where id\_book=@id\_book and id\_basket=@id\_bask)  update [Book-basket] set count=count+@count where id\_book=@id\_book and id\_basket=@id\_bask  else  begin  INSERT INTO [Book-basket](id\_book,id\_basket,count)  VALUES (@id\_book,@id\_bask,@count);  SELECT SCOPE\_IDENTITY();  End  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[add\_book\_in\_order]  @id\_order int,  @id\_book int,  @count int  AS  if exists(select\*from [book-order] where id\_book=@id\_book and id\_order=@id\_order)  update [book-order] set count=count+@count where id\_book=@id\_book and id\_order=@id\_order  else  begin  INSERT INTO [book-order](id\_book,id\_order,count)  VALUES (@id\_book,@id\_order,@count);  SELECT SCOPE\_IDENTITY();  End  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[add\_order]  @id\_cust int,  @sum int,  @date date,  @adress nvarchar(20)  AS  INSERT INTO [Order](id\_customer,Sum,date\_of\_order,delivery\_address)  values(@id\_cust,@sum,@date,@adress);  SELECT SCOPE\_IDENTITY();  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[del\_book]  @id int  AS  delete from Authorship where id\_book=@id  delete from [book-order] where id\_book=@id  delete from [Book-basket] where id\_book=id\_book  delete from Edition where id\_book=@id  delete from Book where id=@id;  -----------------------------------------------------------------  ALTER procedure [dbo].[auth\_exist]  @name nvarchar(50),  @f\_name nvarchar(50)  as  select \* from Author where Name=@name and F\_name=@f\_name  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[books\_from\_bask]  @id\_bask int  AS  select Book.id,Book.Title,book.Language,Book.Year\_of\_publish,Book.Cost,[Book-basket].count,Book.image from [Book-basket]  inner join Basket on [Book-basket].id\_basket=Basket.id  inner join book on [Book-basket].id\_book=Book.id where Basket.id=@id\_bask  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[find\_auth]  @name nvarchar(20),  @f\_name nvarchar(20)  AS  if @f\_name='no'  select Book.id from Author  inner join Authorship  inner join Book on Book.id=Authorship.id\_book on Author.id=Authorship.id\_author  where contains(Name,@name);  else if @name='no'  select Book.id from Author  inner join Authorship  inner join Book on Book.id=Authorship.id\_book on Author.id=Authorship.id\_author  where contains (F\_name,@f\_name);  else  select Book.id from Author  inner join Authorship  inner join Book on Book.id=Authorship.id\_book on Author.id=Authorship.id\_author  where contains(Name,@name) and contains (F\_name,@f\_name)  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[get\_curent\_adm]  @login nvarchar(50)  AS  select id from Seller where login=@login  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[import\_sellers]  @name nvarchar(50),  @f\_name nvarchar(50),  @login nvarchar(50),  @phone\_num nvarchar(50),  @password nvarchar(50)  AS  if exists(select \* from Seller where login=@login)  UPDATE Seller set Name=@name, F\_name=@f\_name,phone\_num=@phone\_num,password=@password where login=@login  else  INSERT INTO Seller(Name,F\_name,login,password,phone\_num)  VALUES (@name, @f\_name,@login,@password,@phone\_num);  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[insert\_adm]  @name nvarchar(50),  @f\_name nvarchar(50),  @login nvarchar(50),  @phone\_num nvarchar(50),  @password nvarchar(50)  AS  INSERT INTO Seller(Name,F\_name,login,password,phone\_num)  VALUES (@name, @f\_name,@login,@password,@phone\_num);    SELECT SCOPE\_IDEN  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[insert\_book]  @title nvarchar(50),  @lang nvarchar(50),  @year int,  @cost int,  @count int,  @content nvarchar(max),  @imag nvarchar(50)  AS  INSERT INTO Book(Title,Language,Year\_of\_publish,Cost,Count,book\_content,image)  VALUES (@title, @lang,@year,@cost,@count,@content,@imag);  SELECT SCOPE\_IDENTITY();  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[insert\_customer]  @name nvarchar(50),  @f\_name nvarchar(50),  @login nvarchar(50),  @phone\_num nvarchar(50),  @password nvarchar(50)  AS  INSERT INTO Customer(Name,F\_name,login,phone\_num,password)  VALUES (@name, @f\_name,@login,@phone\_num,@password);    SELECT SCOPE\_IDENTITY()  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[update\_order]  @id\_order int,  @id\_seller int,  @status nvarchar(20)  AS  update [Order]  set id\_seller=@id\_seller,status=@status  where id=@id\_order  -----------------------------------------------------------------  alter PROCEDURE [dbo].[find\_auth\_2]  @name nvarchar(20)  AS  declare @ch\_name nvarchar(50)='FORMSOF(INFLECTIONAL,"'+@name+'")';  select name from author where contains(Name,@ch\_name);  GO  -----------------------------------------------------------------  ALTER procedure [dbo].[seller\_inf]  @id int  As  select Name,F\_name from Seller where id=@id  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[check\_password]  @pass nvarchar(50),  @id int  AS  select id from Customer where password=@pass and id=@id  -----------------------------------------------------------------  ALTER PROCEDURE [dbo].[export\_book\_from\_order\_ToXML]  @path nvarchar(256)  AS  begin  declare @sql varchar(1000);  select @sql = 'bcp "SELECT [id\_book], [id\_order], [count]  FROM course\_proj.dbo.[book-order] FOR XML PATH(''Order''),  ROOT(''Book'')" queryout "'+@path+'" -r -C1251 -w -T -S  DESKTOP-KHIRI9G\SQLEXPRESS -d Course\_proj';  EXEC xp\_cmdshell @sql;  end; |

Листинг 2 – процедуры для таблиц