**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**КУРСОВОЙ ПРОЕКТ**

### НА ТЕМУ

|  |  |
| --- | --- |
| РАЗРАБОТКА ПОДСИСТЕМЫ УЧЕТА | |
| ПРОДАЖ МАГАЗИНА МУЗЫКАЛЬНЫХ | |
| ИНСТРУМЕНТОВ | |
| Л109. 22КП01. 023 ПЗ | |

(Обозначение документа)

|  |
| --- |
| МДК.02.02 Технология разработки и защиты |
| баз данных |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ПКС-91 | | 14.05.2022 | Н.А. Шефов |
|  | (Группа) | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель |  |  | 14.05.2022 | Ю.С. Маломан |
|  |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2022

**СОДЕРЖАНИЕ**

Перечень сокращений и обозначений 3

Введение 4

1 Анализ и разработка требований 6

1.1 Назначение и область применения 6

1.2 Постановка задачи 6

1.3 Описание алгоритма функционирования системы 7

1.4 Выбор состава программных и технических средств 7

2 Разработка базы данных 10

2.1 Проектирование базы данных 10

2.2 Создание объектов базы данных 14

3 Разработка клиентского приложения 20

3.1 Разработка приложения для доступа к базе данных 20

3.2 Разграничение прав доступа пользователей 23

3.3 Разработка и экспорт отчетов 25

3.4 Тестирование разработанной системы 27

4 Руководство пользователя 30

4.1 Установка приложения 30

4.2 Инструкция по работе 30

Заключение 38

Список использованных источников 39

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

БД – база данных

ИС – информационная система

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

FK – внешний ключ

PK – первичный ключ

SQL – язык структурированных запросов

**ВВЕДЕНИЕ**

Актуальность курсового проекта заключается в том, что применение БД позволяет автоматизировать хранение, обработку и защиту больших объемов финансово- экономической информации. В современных СУБД имеется возможность не только хранить данные в своих структурах, а также разграничивать права доступа пользователей и хранить программный код, т.е. методы, с помощью которых происходит взаимодействие с потребителем. В настоящее время БД имеются или создаются во всех предприятиях малого, среднего и крупного бизнеса. Популярность БД обусловлена необходимостью повышения эффективности работы организаций и предприятий в части информационных и бизнес-процессов посредством проектирования, разработки и внедрения автоматизированных решений.

Система учета продаж магазина музыкальных инструментов – эффективное решение для автоматизации и упрощения работы персонала магазина и сбора статистики о продажах.

Целью курсового проектирования является разработка многопользовательской клиент-серверной ИС для учета продаж магазина музыкальных инструментов.

Для достижения поставленной цели требуется решить следующие задачи:

* изучить особенности предметной области «Учет продаж магазина музыкальных инструментов»,
* выполнить сбор требований к ИС для учета продаж магазина музыкальных инструментов,
* проанализировать результаты, полученные в ходе сбора требований,
* спроектировать концептуальную модель предметной области,
* спроектировать логическую модель предметной области,
* выбрать СУБД,
* спроектировать физическую модель предметной области для выбранной СУБД,
* разработать функции пользователя, хранимые процедуры, представления и триггеры в БД,
* реализовать возможность просмотра и отображения данных,
* реализовать возможность сортировки и фильтрации данных,
* реализовать возможность создания и выгрузки отчетов,
* разработать окно настройки строки подключения к БД,
* реализовать разграничение прав доступа,
* разработать функционал для роли гостя,
* разработать функционал для роли покупателя,
* разработать функционал для роли менеджера,
* разработать функционал для роли администратора,
* организовать развертывание системы,
* выполнить тестирование и отладку приложения,
* разработать руководство пользователя.

**1 Анализ и разработка требований**

**1.1 Назначение и область применения**

Музыкальный магазин «Music Shop» является коммерческой организацией и занимается продажами в сфере музыкальных инструментов. Создание информационной сети музыкального магазина необходимо для повышения продаж магазина, обеспечения контроля за товарами, заказами и продажами магазина и систематизированного взаимодействия между покупателями, менеджерами и администраторами.

**1.2 Постановка задачи**

Поставлена задача разработать многопользовательскую клиент-серверную ИС с оконным интерфейсом пользователя для организации продаж магазина музыкальных инструментов.

Информационная система должна выполнять следующий ряд задач:

* разграничивать права доступа пользователей,
* предоставлять полную информацию о товарах, с возможность фильтрации по категории, производителю, модели, цене и т.д.,
* позволять покупателям добавить товар в корзину, заказать,
* предоставлять менеджерам возможность добавить, изменить, удалить информацию о товаре,
* предоставлять администраторам возможность добавлять, изменять и удалять записи пользователей,
* предоставлять администраторам возможность изменять строку подключения к БД,
* автоматически заполнять информацию о заказе в формате docx,
* собирать статистику о продажах для отчетности.

Следует обратить внимание, что дизайн приложения должен соответствовать современным тенденциям, быть удобным и понятным.

Необходимо обеспечить защиту проекта от взлома, несанкционированного доступа к административным функциям.

**1.3 Описание алгоритма функционирования системы**

Для запуска системы нужно открыть исполняемый файл.

После запуска отображается главная форма, предоставляющая возможность авторизации под одной из категорий пользователей (администратор, менеджер и покупатель).

Пользователь, который не имеет учетную запись и не авторизовался в системе, имеет доступ к просмотру каталога товаров, взаимодействию с формами регистрации и авторизации.

Покупатель получает ряд следующих возможностей: добавление товаров в корзину, оформление заказа, оценивание товара, редактирование данных профиля.

Менеджеру доступны следующие возможности: добавление, редактирование и удаление товаров, сбор статистики о продажах.

Администратору системы доступны следующие функции:

* добавление, редактирование и удаление записей пользователей,
* настройка строки подключения к БД.

**1.4 Выбор состава программных и технических средств**

Согласно цели проекта требуется создать многопользовательскую клиент-серверную ИС для ведения учета продаж магазина музыкальных инструментов.

Работа с ИСбудет осуществляться на персональных компьютерах с установленной ОС Windows версии не ниже Windows 10.

В качестве системы управления базами данных выбрана СУБД Microsoft SQL Server Express 2019, так как имеет высокую производительность, встроенные функции классификации и защиты данных, мониторинга и передачи оповещений, гибкие возможности использования языка и платформы, позволяет организовать высокий уровень безопасности.

Приложение будет написано на языке программирования C#, так как C# – это объектно-ориентированный язык программирования, который позволяет разработчикам создавать многофункциональные приложения и из-за возможности работать с платформой .Net, C# позволяет быстрее, чем любой другой язык, разрабатывать программные решения для бизнеса.

Для разработки приложения будет использоваться интегрированная среда разработки программ Microsoft Visual Studio Community 2022, т.к. она позволяет работать с языком программирования C#. Microsoft Visual Studio Community 2022 имеет продвинутую систему IntelliCode и производительна.

Для функционирования системы на стороне сервера достаточны следующие программные и технические средства:

* жесткий диск: минимум 6 ГБ свободного места на диске,
* процессор: x64 с тактовой частотой не менее 1,4 ГГц,
* ОЗУ: 1 ГБ,
* ОС не ниже Windows 10,
* сервер БД: Microsoft SQL Server Express версии не ниже Microsoft SQL Server Express 2019,
* ПО для конфигурирования, управления и администрирования сервера БД: Microsoft SQL Server Management Studio 18,
* .Net Framework 4.8.

Для функционирования системы на стороне клиента достаточны следующие программные и технические средства:

* процессор: 1 ГГц,
* ОЗУ: 512 МБ,
* жесткий диск: минимум 4,5 ГБ свободного места на диске,

- ОС не ниже Windows 10,

- пакет Microsoft Office,

- .Net Framework 4.8.

**2 Разработка базы данных**

**2.1 Проектирование базы данных**

В БД требуется хранить информацию о товарах. У каждого товара указаны производитель, категория, наименование (уникально для каждого товара), цена, описание (необязательно), количество на складе.

Необходимо хранить информацию в БД о пользователях для разграничения прав доступа и идентификации личности пользователя. У каждого пользователя указан уникальный логин, пароль, номер телефона, фамилия, имя и отчество (необязательно).

Также у товаров может потребоваться хранить в БД несколько изображений и оценки (оценивается по 10 бальной системе), оставленные пользователями, которые заказали этот товар.

Требуется хранить информацию о заказах пользователей в БД. Каждый заказ должен содержать список заказанных товаров, количество товара в заказе не может превышать количества этого товара на складе. Также идентификатор пользователя, оформлявшего заказ, дату оформления заказа, дату доставки заказа (не ранее чем дата оформления заказа) и адрес, на который нужно доставить заказ.

Необходимо хранить информацию в БД о производителях и категориях товаров. У производителей указаны наименование (уникально) и страна. У категорий – наименование и идентификатор главной категории (необязательно и не должен быть равен идентификатору этой категории).

На рисунке 1 показана концептуальная модель предметной области в виде диаграммы «сущность-связь».

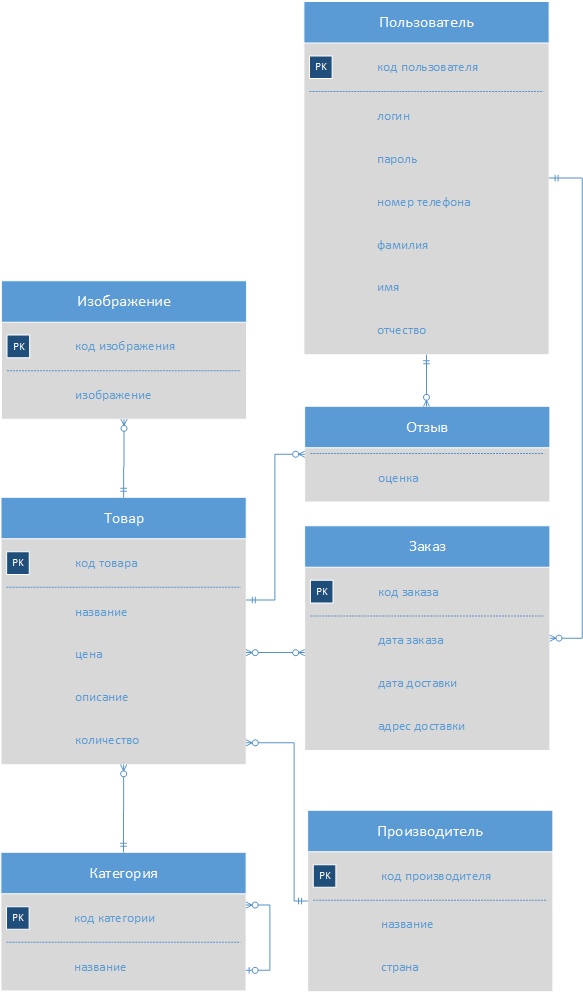


Рисунок 1 – Концептуальная модель

СУБД Microsoft SQL Server, выбранная для хранения данных, является реляционной, поэтому требуется преобразовать разработанную концептуальную модель в логическую с учетом правил преобразования ER-модели в реляционную модель данных:

* связь между сущностями Товар и Изображение 1:М и необязательная для сущности Товар, поэтому формируется 2 отношения с FK в отношении Изображение,
* связь между сущностями Товар и Отзыв 1:М и необязательная для сущности Товар, поэтому формируется 2 отношения с FK в отношении Отзыв,
* связь между сущностями Пользователь и Отзыв 1:М и необязательная для сущности Пользователь, поэтому формируется 2 отношения с FK в отношении Отзыв,
* связь между сущностями Пользователь и Заказ 1:М и необязательная для сущности Пользователь, поэтому формируется 2 отношения с FK в отношении Заказ,
* связь между сущностями Производитель и Товар 1:М и необязательная для сущности Производитель, поэтому формируется 2 отношения с FK в отношении Товар,
* связь между сущностями Заказ и Товар М:М и необязательная для обеих сущностей, поэтому формируется 3 отношения с FK в отношении Заказ товара,
* связь сущности Категория на саму себя 1:М и необязательная, поэтому формируется 1 отношение с FK в отношении Категория,
* связь между сущностями Товар и Категория 1:М и необязательная для сущности Категория, поэтому формируется 2 отношения с FK в отношении Товар.

На рисунке 2 показана логическая модель предметной области, полученная путем преобразования концептуальной модели [1, стр. 89].

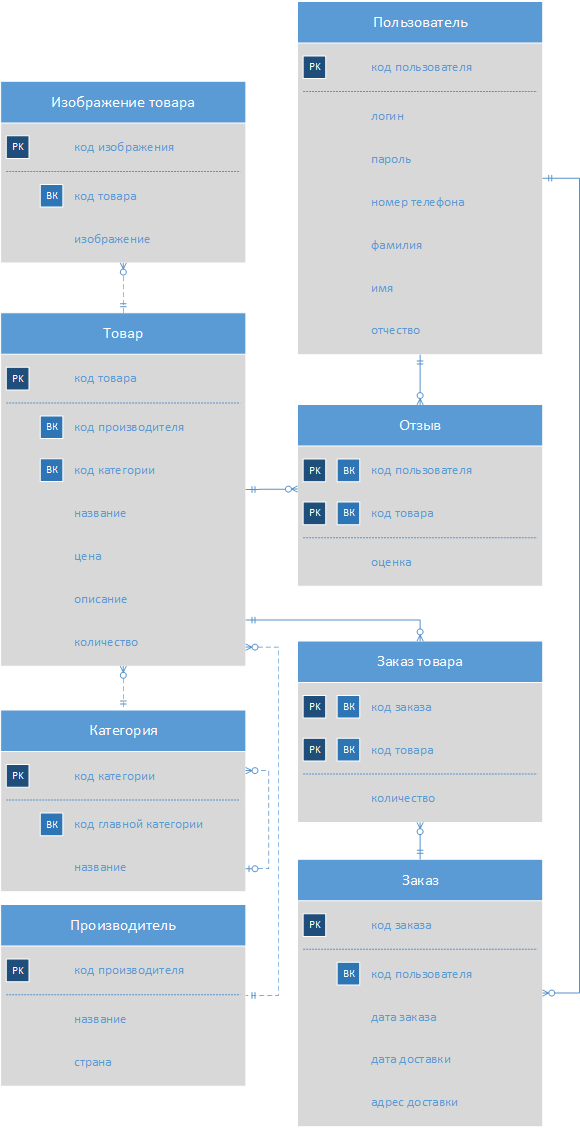


Рисунок 2 – Логическая модель

На рисунке 3 показана физическая модель предметной области, разработанная на основе логической модели для СУБД Microsoft SQL Server Express 2019 [2, стр. 48].

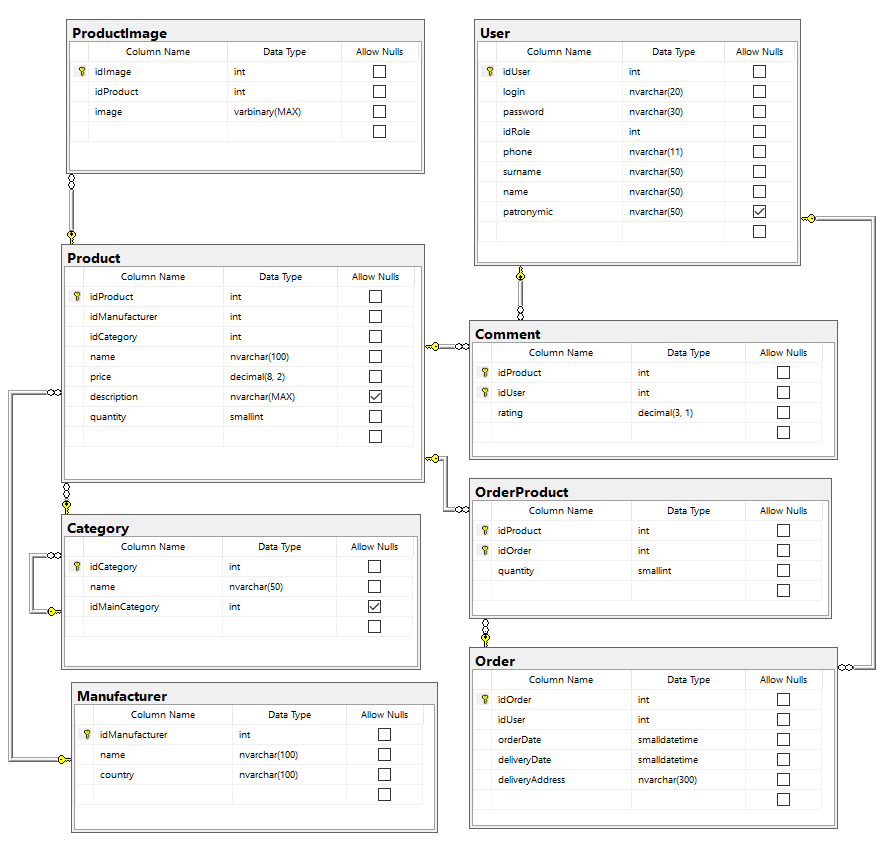


Рисунок 3 – Физическая модель

**2.2 Создание объектов базы данных**

В таблице 1 в виде словаря данных представлено описание созданных таблиц и ограничений целостности БД [3, стр. 46].

Таблица 1 – Словарь данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| User | | | | |
| PK | idUser | int | + | Автоинкрементное |
|  | login | nvarchar(20) | + | Уникальное |
|  | password | nvarchar(30) | + |  |
|  | phone | nvarchar(11) | + |  |
|  | surname | nvarchar(50) | + |  |
|  | name | nvarchar(50) | + |  |
|  | patronymic | nvarchar(50) | - |  |
| Manufacturer | | | | |
| PK | idManufacturer | int | + | Автоинкрементное |
|  | name | nvarchar(100) | + | Уникальное |
|  | country | nvarchar(100) | + |  |
| Category | | | | |
| PK | idCategory | int | + | Автоинкрементное |
| FK | idMainCategory | int | - | ≠ idCategory |
|  | name | nvarchar(50) | + |  |
| Product | | | | |
| PK | idProduct | int | + | Автоинкрементное |
| FK1 | idManufacturer | int | + |  |
| FK2 | idCategory | int | + |  |
|  | name | nvarchar(100) | + | Уникальное |
|  | price | decimal(8, 2) | + | > 0 |
|  | description | nvarchar(MAX) | - |  |
|  | quantity | smallint | + | ≥ 0 |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Поле | Тип данных | Обязательное | Примечание |
| Order | | | | |
| PK | idOrder | int | + | Автоинкрементное |
| FK | idUser | int | + |  |
|  | orderDate | smalldatetime | + | По умолчанию: GetDate() |
|  | deliveryDate | smalldatetime | + | ≥ orderDate |
|  | deliveryAddress | nvarchar(300) | + |  |
| OrderProduct | | | | |
| PK  FK1 | idOrder | int | + |  |
| PK  FK2 | idProduct | int | + |  |
|  | quantity | smallint | + | > 0 |
| Comment | | | | |
| PK  FK1 | idProduct | int | + |  |
| PK  FK2 | idUser | int | + |  |
|  | rating | decimal(3, 1) | + | ≥ 1 && ≤ 10 |
| ProductImage | | | | |
| PK | idImage | int | + | Автоинкрементное |
| FK | idProduct | int | + |  |
|  | image | varbinary(MAX) | + |  |

Для решения поставленных задач в БД созданы следующие представления:

- представление OrderView, возвращающее информацию о заказе, пользователе, оформившего заказ, и общую стоимость заказа,

- представление ProductView, возвращающее данные из таблиц Product, Manufacturer и использующее функции GetLastImage и GetAvgRating (код создания представления показан в листинге 1).

Листинг 1 – Код создания представления ProductView

|  |
| --- |
| --Код создания представления  CREATE VIEW [dbo].[ProductView]  AS  --Выборка данных из таблиц Product и Manufacturer  SELECT  –-последнее изображение товара  dbo.GetLastImage(dbo.Product.idProduct) AS image  ,dbo.Product.name –-наименование товара  –-наименование производителя товара  ,dbo.Manufacturer.name AS manufacturerName  ,dbo.Manufacturer.country –-страна производства товара  ,dbo.Product.price –-цена товара  –-средняя оценка товара  ,dbo.GetAvgRating(dbo.Product.idProduct) AS avgRating  ,dbo.Product.quantity –-количество товара на складе  ,dbo.Product.idProduct –-идентификатор товара  ,dbo.Product.idCategory --идентификатор категории товара  FROM dbo.Product LEFT OUTER JOIN dbo.Manufacturer  ON  dbo.Product.idManufacturer = dbo.Manufacturer.idManufacturer; |

Для решения поставленных задач в БД созданы следующие функции пользователя:

- табличная функция GetUserCartProduct, возвращающая данные о товарах в корзине пользователя на основе идентификатора пользователя (код создания функции представлен листингом 2),

- скалярная функция GetAvgRating, возвращающая среднюю оценку товара на основе идентификатора товара.

Листинг 2 – Код создания функции пользователя GetUserCartProduct

|  |
| --- |
| --Код создания табличной функции  CREATE FUNCTION [dbo].[GetUserCartProduct]  --входной параметр идентифитор пользователя  (@idUser int)  RETURNS TABLE AS RETURN  (  --Выборка данных из таблиц Cart и Product  SELECT Product.\* --все столбцы из таблицы Product  FROM Product JOIN Cart ON Product.idProduct = Cart.idProduct  –-проверка, совпадения идентификатора пользователя  WHERE Cart.idUser = @idUser  ); |

Для решения поставленных задач в БД созданы следующие хранимые процедуры:

- хранимая процедура ChangeRole, изменяющая роль пользователя на основе параметров идентификатор пользователя и идентификатор роли,

- хранимая процедура CreateOrderFromCart, добавляющая заказ и заполняющая список товаров заказа из корзины пользователя на основе параметров идентификатор пользователя, дата доставки и адрес доставки (код создания хранимой процедуры представлен листингом 3).

Листинг 3 – Код создания хранимой процедуры CreateOrderFromCart

|  |
| --- |
| --Код создания хранимой процедуры  CREATE PROCEDURE [dbo].[CreateOrderFromCart]  @idUser int, --Идентификатор пользователя  @deliveryDate smalldatetime, --Дата доставки  @deliveryAddress nvarchar(300) –Адрес доставки  AS  BEGIN  --Вставка данных в таблицу Order  INSERT INTO [Order] (idUser, deliveryDate, deliveryAddress) VALUES (@idUser, @deliveryDate, @deliveryAddress);  --Объявление переменной для хранения идентификатора заказа  DECLARE @idOrder int = scope\_identity();  --Вставка данных в таблицу OrderProduct из таблицы Cart  INSERT INTO OrderProduct  SELECT idProduct –-идентификатор товара  ,@idOrder –-переменная идентификатора заказа  ,quantity –количество товара  FROM Cart  –-проверка, совпадения идентификатора пользователя  WHERE Cart.idUser = @idUser;  END; |

Для решения поставленных задач в БД созданы следующие триггеры:

- триггер TR\_DecreaseInStockQuantity, выполняющийся перед операцией вставки в таблице OrderProduct и уменьшающий значение столбца, хранящее информацию о количестве товаров на складе в таблице Product, на количество заказанного товара (код создания триггера представлен листингом 4),

- триггер TR\_DeleteProduct, выполняющийся после операции удаления в таблице Product и записывающий все данные о товаре в идентичную по структуре таблицу DeletedProduct (код создания триггера представлен листингом 5).

Листинг 4 – Код создания триггера TR\_DecreaseInStockQuantity

|  |
| --- |
| CREATE TRIGGER [dbo].[TR\_DecreaseInStockQuantity]  ON [dbo].[OrderProduct] AFTER INSERT  AS  BEGIN  --Операция обновления данных  UPDATE Product  --Уменьшения значения столбца quantity в таблице Product  SET Product.quantity = Product.quantity - inserted.quantity  FROM Product JOIN inserted  ON Product.idProduct = inserted.idProduct  END; |

Листинг 5 – Код создания триггера TR\_DeleteProduct

|  |
| --- |
| CREATE TRIGGER [dbo].[TR\_DeleteProduct]  ON [dbo].[Product]  AFTER DELETE  AS  BEGIN  --Вставка данных об удаленном товаре в таблицу DeletedProduct  INSERT INTO DeletedProduct  SELECT \*  FROM deleted  END; |

**3 Разработка клиентского приложения**

**3.1 Разработка приложения для доступа к базе данных**

Для осуществления доступа к базе данных разработано оконное приложение Windows Forms. Разработана кнопка BlinkButton, которая меняет цвет при наведении на указанный в свойстве BlinkColor (код класса элемента управления представлен листингом 6).

Листинг 6 – Код класса элемента управления BlinkButton

|  |
| --- |
| public partial class BlinkButton : Button {  /// <summary>  /// Поле цвета кнопки при наведении  /// </summary>  private Color blinkColor;  /// <summary>  /// Свойство цвета кнопки при наведении  /// </summary>  public Color BlinkColor { get => blinkColor; set =>  blinkColor = value; }  /// <summary>  /// Поле для временного хранения цвета кнопки при наведении  /// </summary>  private Color tempColor;  public BlinkButton() { InitializeComponent(); }  //Обработчик события наведения курсора на кнопку  protected override void OnMouseEnter(EventArgs e) {  tempColor = BackColor;  BackColor = BlinkColor;  base.OnMouseEnter(e);  }  //Обработчик события перемещения курсора за границу кнопки  protected override void OnMouseLeave(EventArgs e) {  BackColor = tempColor;  base.OnMouseLeave(e);  }  } |

Для настройки стиля приложения используется родительская форма с выбранным цветом заднего фона, шрифтом и иконкой.

Для доступа к БД используется технология объектно-реляционного отображения EntityFramework, позволяющая организовать гибкую и удобную работу с объектами базы данных. Для работы с EntityFramework разработан класс контекста данных Core, представленный листингом 7.

Листинг 7 – Код класса контекста данных Core

|  |
| --- |
| /// <summary>  /// Класс контекста данных  /// </summary>  sealed class Core  {  /// <summary>  /// Поле контекста данных  /// </summary>  private static MusicShopEntities context;  /// <summary>  /// Свойство доступа к полю контекста данных  /// </summary>  public static MusicShopEntities Context  {  get  {  if (context == null)  {  context = new MusicShopEntities();  }  return context;  }  }  private Core() { }  } |

Получение данных товаров и сортировка реализованы при помощи автоматически реализуемого свойства SortType и свойства Products (код свойств представлен листингом 8) [5, стр. 140].

Листинг 8 – Код свойств SortType и Products

|  |
| --- |
| /// <summary>  /// Автореализуемое свойство хранящее тип сортировки  /// </summary>  public string SortType { get; set; }  /// <summary>  /// Товары из БД, отсортированные исходя от свойства SortType  /// </summary>  public List<Product> Products  {  get  {  switch (SortType)  {  case "По рейтингу":  return Core.Context.Products  .OrderByDescending(p => p.Comments.Average(c =>  c.Rating)).ToList();  case "Сначала дешевые":  return Core.Context.Products.OrderBy(p =>  p.Price).ToList();  case "Сначала дорогие":  return Core.Context.Products  .OrderByDescending(p => p.Price).ToList();  default:  return Core.Context.Products.ToList();  }  }  } |

Фильтрация реализована методом GetFilteredProducts с параметрами:

* products – список для фильтрации,
* name – наименование товара,
* minPrice – минимальная цена товара,
* maxPrice – максимальная цена товара,
* manufacturer – производитель товаров,
* isStock – фильтровать ли товары только в наличии.

Код метода GetFilteredProducts представлен листингом 9.

Листинг 9 – Код метода GetFilteredProducts

|  |
| --- |
| /// <summary>  /// Фильтрует список товаров по параметрам  /// </summary>  /// <param name="products">Список для фильрации</param>  /// <param name="name">Наименование для поиска совпадений в списке товаров</param>  /// <param name="minPrice">Минимальная цена товара</param>  /// <param name="maxPrice">Максимальная цена товара</param>  /// <param name="manufacturer">Производитель товаров</param>  /// <param name="isStock">Фильтровать ли товары только в наличии</param>  /// <returns>Отфильтрованный список товаров</returns>  public List<Product> GetFilteredProducts(List<Product> products, string name, decimal minPrice, decimal maxPrice, Manufacturer manufacturer, bool isStock)  {  var filteredProducts = products;  filteredProducts = filteredProducts  .Where(p => p.Name.ToLower().Contains(name.ToLower()))  .Where(p => p.Price >= minPrice &&  p.Price <= maxPrice).ToList();  filteredProducts = manufacturer.IdManufacturer == -1 ?  filteredProducts : filteredProducts.Where(p =>  p.Manufacturer == manufacturer).ToList();  filteredProducts = isStock ? filteredProducts.Where(p =>  p.Quantity > 0).ToList() : filteredProducts;  return filteredProducts;  } |

**3.2 Разграничение прав доступа пользователей**

Для организации разграничения прав доступа в БД созданы таблицы User и Role, а также глобальные статические поля для хранения учетной записи текущего пользователя, гостя и администратора приложения.

Для осуществления разграничения прав доступа в главной форме при изменении текущего пользователя, закрываются прошлые окна, открывается каталог товаров, доступный всем пользователям, и в зависимости от роли пользователя изменяется боковое меню с кнопками для перехода к окнам.

Разграничение прав доступа реализуется в методе SetRolesConstraint посредством проверки роли текущего пользователя при его обновлении. Код метода SetRolesConstraint представлен листингом 10.

Листинг 10 – Код метода SetRolesConstraint

|  |
| --- |
| /// <summary>  /// Метод разграничения прав доступа  /// </summary>  private void SetRolesConstraint()  {  //Очищаем все элементы в главном окне  foreach (TabPage tabPage in contentTabControl.TabPages)  {  foreach (Control control in tabPage.Controls)  {  control.Dispose();  }  }  if (Connection.CurrentUser != Connection.Admin)  {  CatalogForm catalogForm = new CatalogForm();  catalogTabPage.Tag = catalogForm;  catalogPresenter = new CatalogPresenter(catalogForm,  new CatalogModel());  catalogPresenter.Run();  Manager.ShowParentInTabPage(catalogTabPage);  }  switch (Connection.CurrentUser.Role.RoleName)  {  case "Покупатель":  menuTabControl.SelectedTab = customerTabPage;  CartForm cartForm = new CartForm();  cartTabPage.Tag = cartForm;  cartPresenter = new CartPresenter(cartForm,  new CartModel());  CustomerOrdersForm customerOrdersForm =  new CustomerOrdersForm();  customerOrdersTabPage.Tag = customerOrdersForm;  customerOrdersPresenter =  new CustomerOrdersPresenter(customerOrdersForm);  break;  case "Менеджер":  menuTabControl.SelectedTab = managerTabPage;  OrdersForm ordersForm = new OrdersForm();  ordersTabPage.Tag = ordersForm;  ordersPresenter = new OrdersPresenter(ordersForm,  new OrdersModel());  ManufacturersForm manufacturersForm =  new ManufacturersForm();  manufacturersTabPage.Tag = manufacturersForm;  manufacturersPresenter =  new ManufacturersPresenter(manufacturersForm);  CategoriesForm categoriesForm =  new CategoriesForm();  categoriesTabPage.Tag = categoriesForm;  categoriesPresenter =  new CategoriesPresenter(categoriesForm);  StatisticForm statisticForm = new StatisticForm();  statisticTabPage.Tag = statisticForm;  statisticPresenter =  new StatisticPresenter(statisticForm);  break;  case "Администратор":  menuTabControl.SelectedTab = adminTabPage;  UsersForm usersForm = new UsersForm();  usersTabPage.Tag = usersForm;  usersPresenter = new UsersPresenter(usersForm);  usersPresenter?.Run();  contentTabControl.SelectedTab = usersTabPage;  break;  default:  menuTabControl.SelectedTab = guestTabPage;  break;  }  loginLabel.Visible = Connection.CurrentUser ==  Connection.Guest;  logoutLabel.Visible = Connection.CurrentUser !=  Connection.Guest;  usernameLabel.Text = Connection.CurrentUser.Login;  } |

**3.3 Разработка и экспорт отчетов**

В приложении разработаны автоматические отчеты: автоматическое заполнение заказа покупателя в формате docx и окно с визуализацией данных продаж товаров в виде диаграмм и возможностью сохранить диаграмму на ПК в формате png.

Код обработчика события View\_SaveOrder, выполняющего автоматическое заполнения заказа покупателя в формате docx представлен листингом 11.

Листинг 11 – Код обработчика события View\_SaveOrder

|  |
| --- |
| private void View\_SaveOrder(object sender, EventArgs e)  {  //Диалоговое окно сохранения файла  SaveFileDialog saveFileDialog = new SaveFileDialog();  saveFileDialog.Filter = "docx|\*.docx";  if (saveFileDialog.ShowDialog() == DialogResult.OK)  {  try  {  var wordApp = new Word.Application();  //Открытие шаблона и замена строк на текущие данные  var doc = wordApp.Documents.Add($@"{Environment.CurrentDirectory}\Template\OrderTemplate.docx");  string fullName = $"{model.Order.User.Surname} {model.Order.User.Name} {model.Order.User.Patronymic}";  doc.Content.Find.Execute(FindText: "%fullName%", ReplaceWith: fullName, Replace: Word.WdReplace.wdReplaceAll);  doc.Content.Find.Execute(FindText: "%phone%", ReplaceWith: model.Order.User.Phone, Replace: Word.WdReplace.wdReplaceAll);  doc.Content.Find.Execute(FindText: "%address%", ReplaceWith: model.Order.DeliveryAddress, Replace: Word.WdReplace.wdReplaceAll);  doc.Content.Find.Execute(FindText: "%orderDate%", ReplaceWith: model.Order.OrderDate.ToShortDateString(), Replace: Word.WdReplace.wdReplaceAll);  doc.Content.Find.Execute(FindText: "%deliveryDate%", ReplaceWith: model.Order.DeliveryDate.ToShortDateString(), Replace: Word.WdReplace.wdReplaceAll);  doc.Content.Find.Execute(FindText: "%sum%", ReplaceWith: $"{model.Order.OrderProducts.Sum(op => op.Quantity \* op.Product.Price)} руб.", Replace: Word.WdReplace.wdReplaceAll);  //Заполнения таблицы со списком товаров  foreach (var orderProduct in model.Order.OrderProducts)  {  var row = doc.Tables[1].Rows.Add();  row.Cells[1].Range.Text =  orderProduct.Product.Name;  row.Cells[2].Range.Text =  orderProduct.Product.Price.ToString();  row.Cells[3].Range.Text =  orderProduct.Quantity.ToString();  }  doc.SaveAs(saveFileDialog.FileName);  wordApp.Quit();  view.ShowInfo($"Заказ успешно сохранен по пути: {saveFileDialog.FileName}");  }  catch (Exception ex)  {  view.ShowError(ex.Message);  }  }  } |

**3.4 Тестирование разработанной системы**

Для тестирования будет использоваться метод «черного ящика», не предоставляющий доступ к исходному коду приложения.

В таблице 2 показан набор тестов разработанного приложения.

В качестве корректных данных для тестирования используются:

* данные для входа администратора: логин «root», пароль «toor»,
* данные для входа менеджера: логин «manager2003», пароль «qwerty123»,
* данные для входа покупателя: логин «customer2110», пароль «password123»,
* данные подключения к БД: сервер «KIRILL\SQLEXPRESS», БД «MusicShop», логин «sa», пароль «1».

Таблица 2 – Набор тестов разработанного приложения

| Действие | Ожидаемый результат | Полученный результат |
| --- | --- | --- |
| Открыть окно авторизации, ввести «root» в поле ввода логина и «toor» в поле ввода пароля, нажать на кнопку «Войти» | Отображение модального окна с приветствием. Закрытие окна авторизации, отображение кнопок «Пользователи» и «Подключение» в боковом меню. Открытие списка пользователей. | Соответствует ожидаемому |
| Открыть окно авторизации, ввести «manager2003» в поле ввода логина и «qwerty123» в поле ввода пароля, нажать на кнопку «Войти» | Отображение модального окна с приветствием. Закрытие окна авторизации, отображение кнопок «Каталог», «Производители», «Категории», «Заказы» и «Статистика» в боковом меню. Открытие каталога товаров. | Соответствует ожидаемому |

Продолжение таблицы 2

| Действие | Ожидаемый результат | Полученный результат |
| --- | --- | --- |
| Открыть окно авторизации, ввести «customer2110» в поле ввода логина и «password123» в поле ввода пароля, нажать на кнопку «Войти» | Отображение модального окна с приветствием. Закрытие окна авторизации, отображение кнопок «Каталог», «Мои заказы», «Корзина» и «Мой профиль» в боковом меню. Открытие каталога товаров. | Соответствует ожидаемому |
| Открыть окно настройки подключения к БД, ввести «KIRILL\SQLEXPRESS» в поле ввода имени сервера, «MusicShop» в поле ввода БД, «sa» в поле ввода логина, «123» в поле ввода пароля, нажать на кнопку «ОК». | Вывод сообщения об ошибке подключения к БД | Соответствует ожидаемому |
| В окне «Каталог» из дерева категорий выбрать категорию «Гитары» | Вывод на экран списка продуктов только из категории «Гитары» и её подкатегорий | Соответствует ожидаемому |
| В окне «Каталог» из списка сортировки выбрать сортировку «Сначала дешевые» | Вывод на экран списка продуктов, отсортированных по возрастанию цены | Соответствует ожидаемому |
| Авторизоваться как менеджер, открыть окно «Статистика», нажать кнопку «Сохранить как…», выбрать путь сохранения файла | Появление файла с диаграммой по указанному при сохранении пути | Соответствует ожидаемому |
| В окне «Каталог» при пустом списке товаров нажать на кнопку «Открыть» | Отсутствие изменений | Вызов исключения «Отсутствует ссылка на объект» |

Во время тестирования приложения была обнаружена ошибка. При попытке открыть окно с информацией о товаре, при отсутствии товаров в списке, возникает исключение «Отсутствует ссылка на объект», так как свойство SelectedProduct, хранящее выделенный товар из списка, не имеет значения.

Код обработчика событий, содержащий ошибку представлен листингом 12. Исправленный код обработчика событий представлен листингом 13.

Листинг 12 – Код обработчика событий View\_OpenProduct, содержащий ошибку

|  |
| --- |
| private void View\_OpenProduct(object sender, EventArgs e)  {  //отсутствует проверка на наличие значения в свойстве  SelectedProduct  new ProductPresenter(  new ProductForm  {  FormBorderStyle = FormBorderStyle.None,  TopLevel = false,  Dock = DockStyle.Fill,  Parent = view.Parent  }, new ProductModel(view.SelectedProduct as  Product)).Run();  view.Hide();  } |

Листинг 13 – Код обработчика событий View\_OpenProduct с внесёнными исправлениями

|  |
| --- |
| private void View\_OpenProduct(object sender, EventArgs e)  {  Добавлена проверка значения свойства SelectedProduct  if (view.SelectedProduct == null)  {  //Завершение работы метода при отсутствии значения  return;  }  new ProductPresenter(  new ProductForm  {  FormBorderStyle = FormBorderStyle.None,  TopLevel = false,  Dock = DockStyle.Fill,  Parent = view.Parent  }, new ProductModel(view.SelectedProduct as  Product)).Run();  view.Hide();  } |

В ходе тестирования было выявлено, что ИС полностью выполняет поставленные задачи, обладает высокой производительностью, надежностью и отказоустойчивостью так как обрабатывает большинство возможных исключений.

**4 Руководство пользователя**

**4.1 Установка приложения**

Для установки ИС на стороне сервера необходимо:

* установить Microsoft SQL Server 2019 Express и Microsoft SQL Server Management Studio 18 [4, стр. 18],
* запустить Microsoft SQL Server Management Studio 18, подключиться к серверу и поочередно выполнить SQL-скрипты из папки Scripts на установочном диске (скрипты пронумерованы).

Для установки ИС на стороне клиента необходимо:

* распаковать архив с приложением в любое свободное пространство на жёстком диске,
* при необходимости загрузить пакет .Net Framework 4.8 с официального сайта Microsoft,
* запустить приложение и авторизоваться в качестве администратора системы, используя логин «root» и пароль «toor», задать необходимые настройки подключения к БД.

**4.2 Инструкция по работе**

Для запуска приложения требуется открыть исполняемый файл.

Пользователю откроется главное окно приложения с каталогом товаров и кнопкой «Войти в учетную запись». При нажатии на эту кнопку откроется окно авторизации, в котором необходимо указать учетные данные пользователя. Окно авторизации представлено на рисунке 4.

Если у пользователя отсутствует учетная запись, то он имеет возможность перейти к окну регистрации из окна авторизации (представлено на рисунке 5).

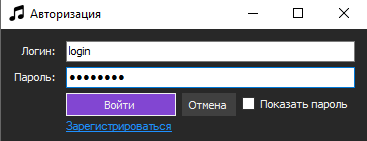


Рисунок 4 – Music Shop. Вид окна «Авторизация»

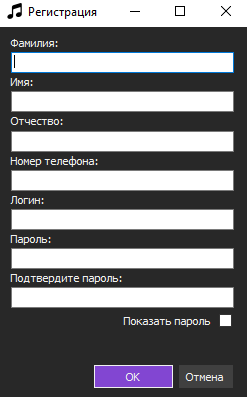


Рисунок 5 – Music Shop. Вид окна «Регистрация»

Для роли администратора приложения в боковом меню доступны кнопки «Пользователи» и «Подключение».

В окне настроек подключения (представлено на рисунке 6) отображаются текущие данные строки подключения к БД, которые при необходимости можно изменить.

Сигналом к успешному сохранению данных о подключении является вывод модального окна с сообщением, представленного на рисунке 7.

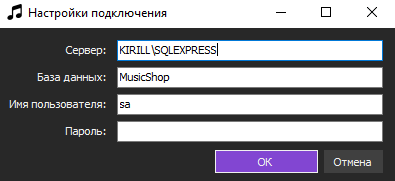


Рисунок 6 – Music Shop. Вид окна «Настройки подключения»

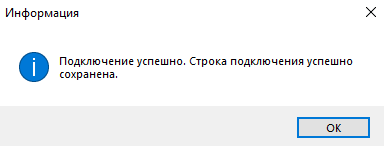


Рисунок 7 – Music Shop. Вид модального окна успешного сохранения строки подключения к БД

Каждый покупатель имеет возможность изменить данные своей учетной записи. Для этого необходимо нажать на кнопку «Мой профиль», после чего откроется окно с данными текущего покупателя и возможностью их редактирования (представлено на рисунке 8).

Если пользователь авторизуется в приложении в качестве покупателя, у него появится возможность оформления заказа. Для этого необходимо открыть каталог и выделить необходимый товар из списка, после нажать на кнопку «Открыть».

Для более быстрого и удобного поиска товаров можно ввести наименование товара в поле ввода, минимальную и максимальную цену, выбрать категорию и производителя (пример фильтрации списка товаров представлен на рисунке 9) или отсортировать товары, выбрав соответствующий параметр сортировки из выпадающего списка (пример сортировки списка товаров представлен на рисунке 10).

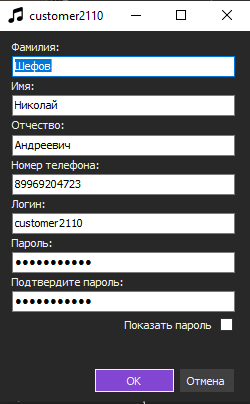


Рисунок 8 – Music Shop. Вид окна редактирования учетной записи

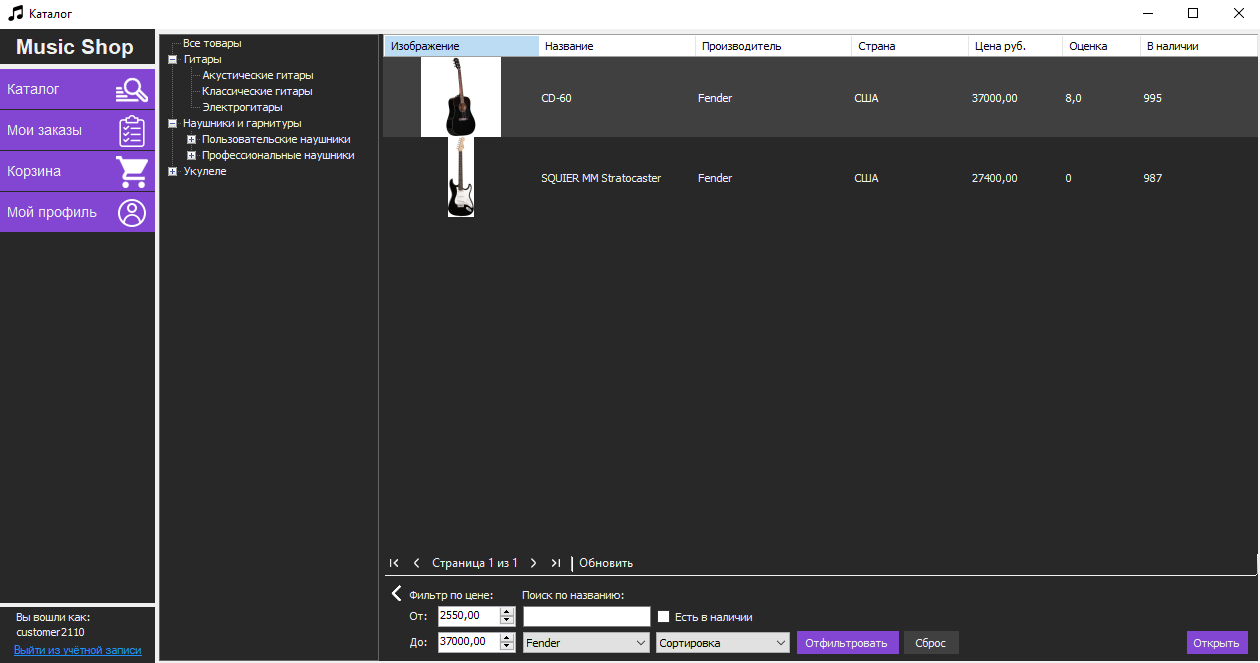


Рисунок 9 – Music Shop. Вид окна «Каталог». Пример фильтрации данных

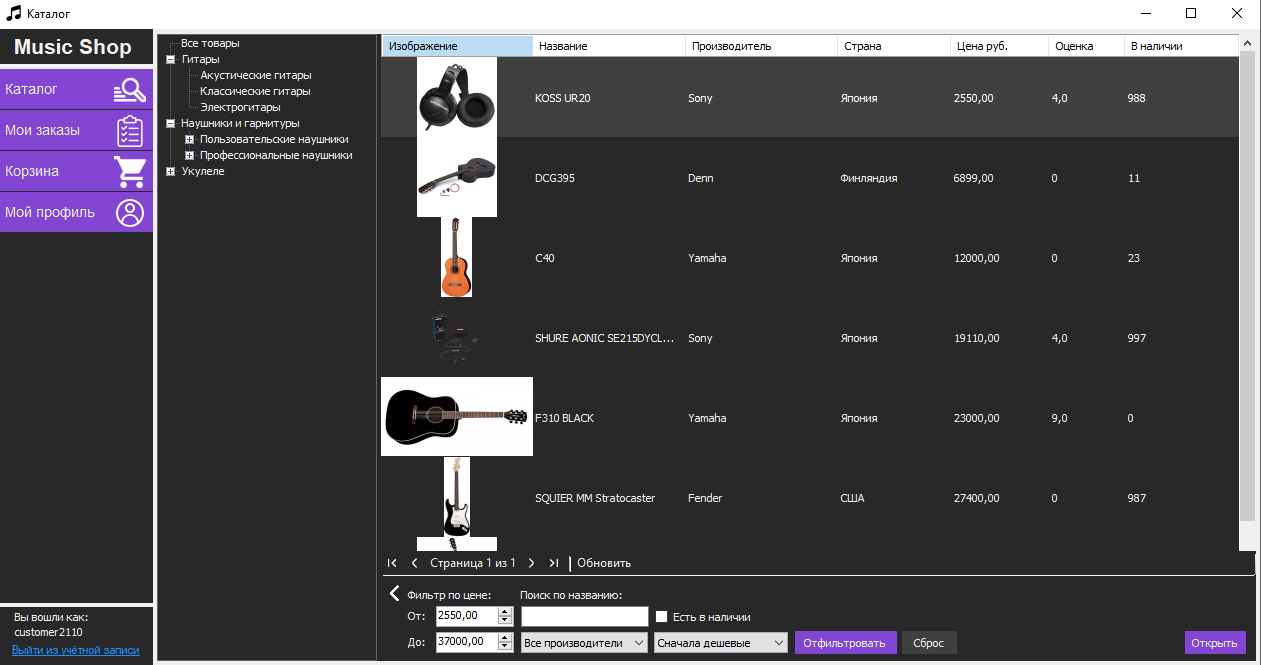


Рисунок 10 – Music Shop. Вид окна «Каталог». Пример сортировки данных

После открытия окна товара, в котором представлена более подробная информация о товаре, если товар есть в наличии, его можно добавить в корзину, нажав на кнопку «В корзину» (представлено на рисунке 11).

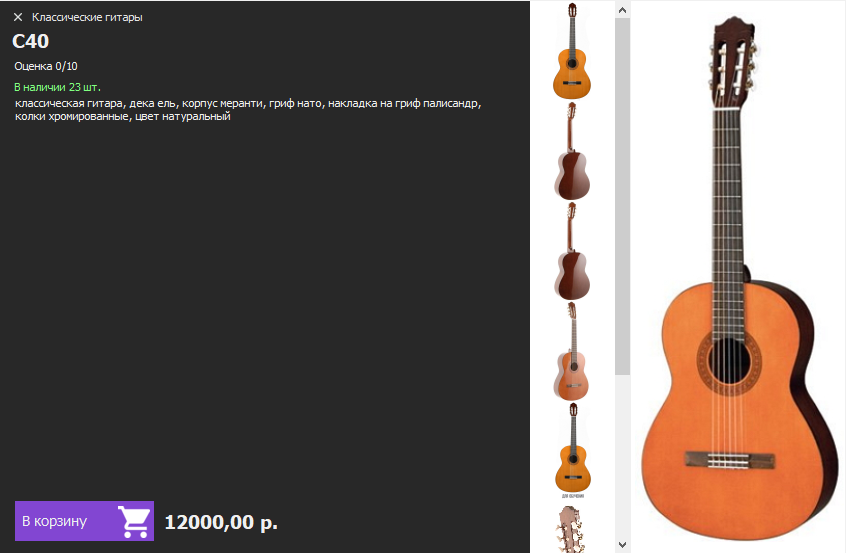


Рисунок 11 – Music Shop. Вид окна подробной информации о товаре

Для оформления заказа необходимо нажать на кнопку «Корзина», после чего откроется окно с информацией о составе текущего заказа и возможностью его редактирования (представлено на рисунке 12). Для оформления заказа необходимо заполнить поля дата доставки и адрес доставки, нажать на кнопку «Оформить заказ». При успешном оформлении появится всплывающее окно с уведомляющее об успешном оформлении заказа и заказ появляется в окне «Мои заказы».

Если пользователь авторизуется в приложении в качестве менеджера, у него появится возможность редактирования списка товаров, производителей и категорий, просмотр статистики по продажам товаров и печать заказов покупателей.

Для добавления записи товара в каталог товаров необходимо в окне «Каталог» нажать на кнопку «Добавить товар», после чего откроется соответствующее окно (представлено на рисунке 13), в котором нужно заполнить поля ввода и нажать на кнопку «ОК». Также для редактирования товара открывается окно, подобное окну «Добавить товар», с текущими данными товара и возможностью для их редактирования. Для удаления выделить нужный товар и нажать на кнопку «Удалить товар».

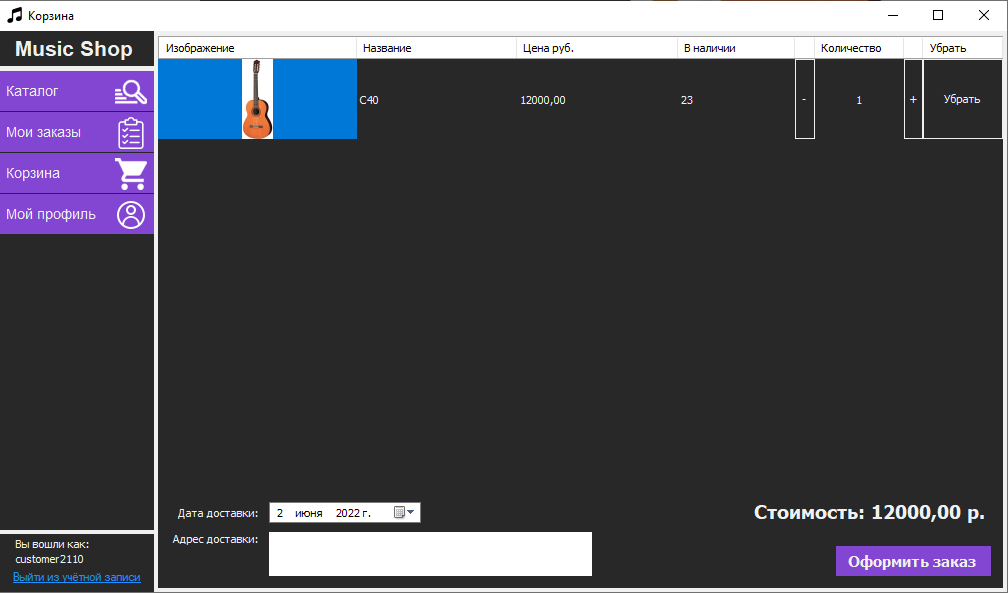


Рисунок 12 – Music Shop. Вид окна «Корзина»

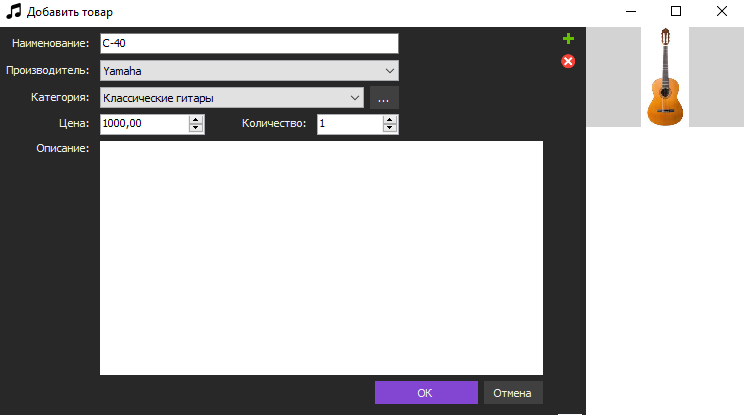


Рисунок 13 – Music Shop. Вид окна «Добавить товар»

При нажатии на кнопку «Заказы» открывается соответствующее окно с заказами пользователей. Каждый заказ можно просмотреть, отредактировать и удалить, а также получить информацию о заказе в формате docx, нажав на кнопку «Сохранить заказ» (представлено на рисунке 14). Образец сохраненного заказа предоставлен на рисунке 15

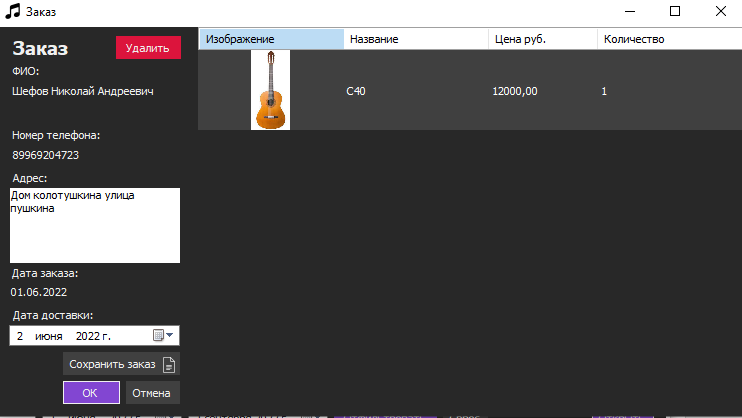


Рисунок 14 – Music Shop. Вид окна «Заказ»

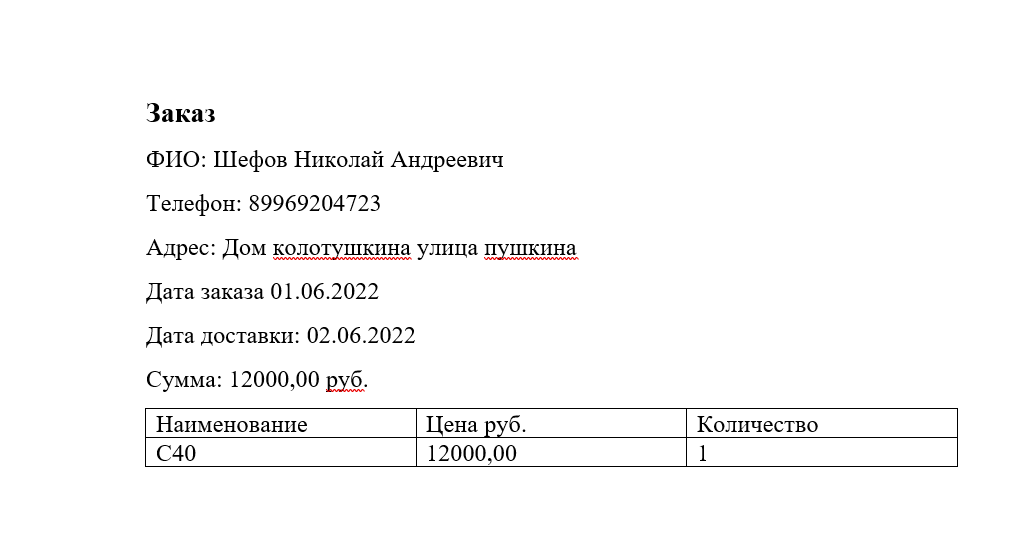


Рисунок 15 – Music Shop. Образец заказа в формате docx

Для просмотра и экспорта статистики о продажах товаров необходимо нажать на кнопку «Статистика». В открывшемся окне отображается диаграмма со статистикой о продажах товаров, также можно выбрать тип диаграммы и период продаж товаров в соответствующих выпадающих списках. Чтобы экспортировать диаграмму в формате png, необходимо нажать на кнопку «Сохранить как…» и выбрать путь сохранения. После успешного сохранения появится соответствующее всплывающее окно. Образец диаграммы представлен на рисунке 16.

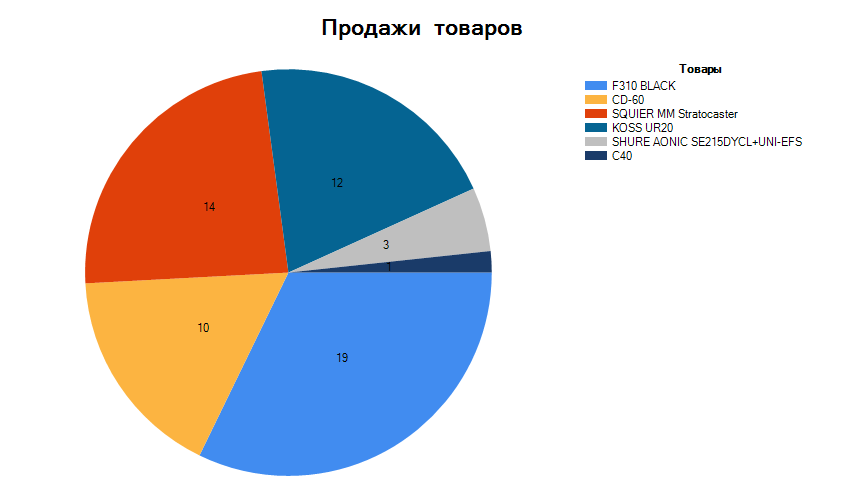


Рисунок 16 – Music Shop. Образец диаграммы со статистикой продаж

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсового проекта сформированы и закреплены навыки разработки БД и создания многопользовательских клиент-серверных ИС для работы с БД, а также сформированы навыки по разработке документаций ПО.

Цель работы достигнута: разработана многопользовательская клиент-серверная ИС для учета продаж магазина музыкальный инструментов.

Для достижения этой цели выполнены следующие задачи:

* изучены особенности предметной области «Учет продаж магазина музыкальных инструментов»,
* выполнен сбор требований к ИС для учета продаж магазина музыкальных инструментов,
* описана спецификация разрабатываемого ПО,
* спроектирована концептуальная, логическая и физическая модель предметной области,
* описан словарь данных,
* создана БД,
* в БД разработаны функции пользователя, хранимые процедуры, представления и триггеры,
* разработано приложение для доступа к данным БД,
* реализовано разграничение прав доступа в приложении,
* реализована возможность настройки подключения к БД,
* реализована возможность просмотра и отображения данных,
* реализована возможность сортировки и фильтрации данных,
* реализована возможность создания и выгрузки отчетов,
* выполнено тестирование и отладка разработанного приложения,
* разработано руководство пользователя.

Так как цель курсового проекта достигнута, его результат можно считать успешным.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Голицына, О. Л. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 4-е изд., перераб. и доп. – Москва : ФОРУМ : ИНФРА-М, 2020. – 400 с. – URL: https://znanium.com/catalog/document?id=362825. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
2. Голицына, О. Л. Основы проектирования баз данных : учебное пособие / О. Л. Голицына, Т. Л. Партыка, И. И. Попов. – 2-е изд., перераб. и доп. Москва : ФОРУМ : ИНФРА-М, 2021. – 416 с. – URL: https://znanium.com/catalog/document?id=364900. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
3. Дадян, Э. Г. Данные: хранение и обработка : учебник / Э. Г. Дадян. – Москва : ИНФРА-М, 2020. – 205 с. – URL: https://znanium.com/catalog/document?id=346013. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
4. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSOL-типа для применения проектирования информационных систем : Учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ИД «ФОРУМ» : ИНФРА-М, 2018. – 368 с. – URL: http://znanium.com/bookread2.php?book=926871. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.
5. Павловская, Т. А. C#. Программирование на языке высокого уровня: Учебник для вузов / Т. А. Павловская. – Санкт-Петербург : Питер, 2021. – 432 с. – URL: https://ibooks.ru/bookshelf/377952/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.