

**МИНОБРНАУКИ РОССИИ**  
федеральное государственное автономное образовательное учреждение  
высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ»)  
Университетский политехнический колледж

Утверждаю  
Зам. директора по УМР  
\_\_\_\_\_ Е.Г. Конакина  
«\_\_» \_\_\_\_ 2019 г.

**ОТЧЕТ**  
**по учебной практике**

по профессиональному модулю

ПМ.02 «Участие в интеграции программных модулей»

(код и наименование)

Специальность 09.02.03 «Программирование в компьютерных системах»

(код и наименование специальности)

Студента 4 курса 42928/2 группы

\_\_\_\_\_ Тимушева Федора Алексеевича \_\_\_\_\_  
(Фамилия, имя, отчество)

Место прохождения практики \_\_\_\_\_

ФГАОУ ВО «СПбПУ» УПК ВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «14» января 2019 г. по «02» февраля 2019 г.

Руководитель(и) практики:

\_\_\_\_\_  
(подпись)                      Зернова Е.Н.  
(Ф.И.О. расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_

Санкт-Петербург  
2019

## Задание на учебную практику

по профессиональному модулю

ПМ.03 «Участие в интеграции программных модулей»

(код и наименование)

Специальность 09.02.03 «Программирование в компьютерных системах»

(код и наименование специальности)

Студенту 4 курса 42928/2 группы

Тимушеву Федору Алексеевичу  
(фамилия, имя, отчество)

Место прохождения практики

ФГАОУ ВО «СПбПУ» УПК ВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики  
с «14» января 2019 г. по «02» февраля 2019 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ)*

- 1 Разработка технического задания
- 2 Этапы проектирования программного комплекса
- 3 Исследовательские работы. Обоснование принципиальной возможности решения задачи
- 4 Участие в выработке требований к программе, утверждение технического задания
- 5 Разработка схемы алгоритма программного продукта
- 6 Определение форм представления входных и выходных данных
- 7 Разработка структуры программы
- 8 Составление схем программы в соответствии с ЕСПД
- 9 Разработка пояснительной записки
- 10 Разработка схем алгоритмов модулей
- 11 Приведение схемы алгоритма модуля к структурному виду
- 12 Написание структурных модулей и их отладка
- 13 Разработка текста программы
- 14 Разработка интерфейса пользователя
- 15 Разработка методики испытаний
- 16 Отладка и тестирование программы
- 17 Оптимизация программы

- 18 Разработка технологической документации
- 19 Проведение приемо-сдаточных испытаний

### Индивидуальное задание

1. Анализ предметной области <>
2. Построение диаграмм этапа проектирования
3. Проектирование базы данных SQL Server Management Studio
4. Разработка веб-приложения ASP.NET MVC с помощью архитектуры проектирования MVC
5. Отладка и тестирование приложения
6. Выпуск приложения

Задание выдал «14» января 2019г.

\_\_\_\_\_  
(подпись)

Зернова Е.Н.  
(Ф.И.О.)

С заданием ознакомлен

«14» января 2019г.

\_\_\_\_\_  
(подпись)

\_\_\_\_\_  
(Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**  
федеральное государственное автономное образовательное учреждение  
высшего образования  
«Санкт-Петербургский политехнический университет Петра Великого»  
(ФГАОУ ВО «СПбПУ»)  
Университетский политехнический колледж

**ДНЕВНИК**  
**прохождения учебной практики**

по профессиональному модулю

ПМ.03 «Участие в интеграции программных модулей»

(код и наименование)

Специальность 09.02.03 «Программирование в компьютерных системах»  
(код и наименование специальности)

Студенту 4 курса 42928/2 группы

Тимушеву Федору Алексеевичу  
(фамилия, имя, отчество)

Место прохождения практики

ФГАОУ ВО «СПбПУ» УПК ВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «14» января 2019 г. по «02» февраля 2019 г.

Руководитель(и) практики:

Зернова Е.Н.  
(подпись) (Ф.И.О. расшифровка подписи)

Санкт-Петербург

2019

### Содержание дневника

Дата	Виды выполненных работ и заданий по программе практик	Подпись руководителя практики
1	2	3
14.01.2019	Анализ предметной области. Построение ER-диаграммы	
15.01.2019	Разработка базы данных в среде SQL Server Management Studio.	
	Определение первичных и внешних ключей, связей таблиц. Заполнение базы данных записями.	
17.01.2019	Разработка двух хранимых процедур и двух функций, возвращающих табличное и скалярное значения	
16.01.2019	Разработка Use Case диаграммы (диаграммы вариантов использования)	
18.01.2019	Изучение паттерна ASP.NET MVC	
	Разработка схемы алгоритма программного продукта	
18.01.2019	Подключение Entity Data Model к проекту	
	Создание представлений (View) для вывода информации	
20.01.2019	Разработка системы авторизации	
	Разработка системы регистрации	
21.01.2019	Составление схем программы в соответствии с ЕСПД	
	Разработка Sequence Diagrams (диаграммы последовательности)	
22.01.2019	Разработка схем алгоритмов модулей	
	Доработка классов контроллеров (Controller)	
24.01.2019	Приведение схемы алгоритма модуля к структурному виду	
	Доработка классов моделей (Model)	
25.01.2019	Написание структурных модулей и их отладка	
	Доработка и оформление представлений (View) в удобном для пользователя виде	
28.01.2019	Разработка методики испытаний	
	Написание мануальных тест-кейсов к программе	
	Отладка программы	
30.01.2019	Написание user-stories	
	Отладка программы	
	Оптимизация программы	
02.02.2019	Разработка технологической документации	

## СОДЕРЖАНИЕ

Введение .....	7
1 Теоретические основы разработки .....	7
1.1 Описание предметной области .....	7
1.2 Обзор средств программирования .....	7
2 Практическая часть .....	11
2.1 Участие в выработке требований программного обеспечения.....	11
2.2 Участие в проектировании программного обеспечения с.....	12
2.3 Участие в интеграции программных модулей.....	18
Заключение.....	28

## **ВВЕДЕНИЕ**

### **1 Теоретические основы разработки**

#### **1.1 Описание предметной области**

Музыкальный магазин имеет большой выбор музыкальных инструментов, оборудования и сопутствующих товаров. В штате магазина трудятся работники: менеджеры и продавцы.

Магазин использует базу данных для учета товаров. Для этого требуется многопользовательская база данных, поскольку продажами могут одновременно или посменно заниматься несколько служащих. Кроме того, менеджер также должен иметь доступ к базе данных, чтобы определить момент, когда необходимо будет заказать новую партию того или иного товара. При этом только менеджер имеет право регистрировать новых продавцов.

База данных должна содержать информацию о клиентах, товарах, поставках и сотрудниках.

#### **1.2 Обзор средств программирования**

Платформа .NET Framework - это один из компонентов системы Windows. Он позволяет создавать и использовать приложения нового поколения. Назначение платформы .NET Framework :

- создание целостной объектно-ориентированной среды программирования допускающей различные варианты реализации: код может храниться и выполняться локально; выполняться локально, а распространяться через Интернет; или выполняться удаленно;
- предоставление среды выполнения кода, в которой число конфликтов при развертывании программного обеспечения и управлении версиями будет сведено к минимуму;
- обеспечение безопасности выполнения кода в среде - в том числе кода, созданного неизвестным разработчиком или разработчиком с частичным доверием;
- предоставление среды выполнения кода, позволяющей устранить проблемы, связанные с производительностью сред на основе сценариев или интерпретации;
- унификация работы разработчиков в совершенно разных приложениях: как в приложениях Windows, так и в веб-приложениях;

- использование промышленных стандартов во всех областях обмена данными и, как следствие, обеспечения совместимости кода, созданного в .NET Framework, с другими программами.

Платформа .NET Framework состоит из двух основных компонентов: среды CLR и библиотеки классов .NET Framework. Среда CLR - это фундамент платформы .NET Framework. Это своеобразный агент, управляющий кодом во время его выполнения, предоставляющий ключевые службы, связанные с такими процессами, как управление памятью, потоками и удаленными операциями, а также обеспечивающий безопасность типов и другими способами контролирующий правильность кода, гарантируя безопасность и стабильность приложений. Понятие управления кодом является для среды основополагающим. Код, созданный для среды, называется управляемым. Любой другой код называется неуправляемым кодом. Библиотека классов, второй основной компонент платформы .NET Framework, является обширным объектно-ориентированным набором типов, которые можно использовать для разработки самых различных приложений - от классических приложений с интерфейсом командной строки или графическим интерфейсом пользователя до новейших приложений на базе технологий ASP.NET, например веб-форм и веб-служб XML.

Платформа .NET Framework может располагаться на неуправляемом компоненте, который загружает среду CLR в собственные процессы и инициирует выполнение управляемого кода - тем самым создавая среду приложений, в которой может выполняться как управляемый, так и неуправляемый код. Платформа .NET Framework сама предоставляет несколько хост-приложений и поддерживает хост-приложения сторонних разработчиков.

Рассмотрим, к примеру, следующую ситуацию: платформа ASP.NET предоставляет масштабируемую среду для управляемого кода на стороне сервера. ASP.NET непосредственно взаимодействует со средой, обеспечивая работу приложений ASP.NET и веб-служб XML (речь о них пойдет ниже).

Обозреватель Internet Explorer является примером неуправляемого приложения, в котором располагается среда (в виде расширения типа MIME). Размещение среды в обозревателе Internet Explorer позволяет встраивать управляемые компоненты и элементы управления Windows Forms в документы HTML. Такое размещение среды делает возможным использование управляемого мобильного кода (схожего с элементами управления Microsoft ActiveX), предоставляя при этом расширенные



возможности, характерные исключительно для управляемого кода, к примеру выполнение при частичном доверии или изолированное хранение файлов.

Библиотека классов .NET Framework - это набор стандартных типов, тесно связанных со средой CLR. Библиотека классов является объектно-ориентированной. В ней содержатся типы, на основании которых ваш управляемый код может выводить нужные функции. Это не только обеспечивает простоту использования типов в .NET Framework, но и сокращает количество времени, необходимое для изучения новых возможностей платформы. Кроме того, компоненты сторонних разработчиков могут полностью интегрироваться в библиотеку классов .NET Framework.

К примеру, классы коллекций в .NET Framework позволяют реализовать ряд интерфейсов, которые затем вы можете использовать для создания собственных классов. Классы коллекций, которые вы создадите, будут тесно интегрированы в набор классов .NET Framework.

Поскольку библиотека классов является объектно-ориентированной, типы в .NET Framework позволяют выполнять ряд стандартных операций программирования: управление строками, сбор данных, подключение к базам данных, доступ к файлам. В библиотеке также есть типы, поддерживающие самые разнообразные специализированные ситуации, с которыми вы сталкиваетесь при разработке приложений. Платформу .NET Framework можно использовать для создания следующих приложений и служб:

- консольных приложений;
- приложений с графическим интерфейсом пользователя для системы Windows (Windows Forms);
- приложений ASP.NET;.
- мобильных приложений;.
- Веб-служб XML;
- служб Windows.

Например, классы Windows Forms - это обширный набор стандартных типов, значительно упрощающих разработку графического интерфейса под Windows. При создании веб-форм ASP.NET можно использовать классы Web Forms.

#### Разработка клиентских приложений

Клиентские приложения больше всего напоминают классические приложения для системы Windows. Это приложения, отображающие на экране компьютера окна

или формы, позволяющие пользователю выполнять те или иные задачи. Клиентским приложением может быть текстовый редактор, электронная таблица, а также бизнес-приложения: средства ввода данных, создания отчетов и т. д. В клиентских приложениях обычно используются окна, меню, кнопки и другие элементы графического интерфейса. Очень часто они осуществляют доступ к локальным ресурсам, например к файловой системе, и периферийным устройствам, например к принтерам.

Другая разновидность клиентских приложений - это традиционные элементы управления ActiveX (теперь на смену им пришли элементы управления Windows Forms), развертываемые через Интернет в виде веб-страниц. Они мало чем отличаются от других клиентских приложений: работают как готовые приложения, осуществляют доступ к локальным ресурсам и имеют графический интерфейс.

Раньше такие приложения создавались либо с использованием С или С++ и классов Microsoft Foundation (MFC), либо при помощи среды быстрой разработки приложений, к примеру Microsoft Visual Basic. Платформа .NET Framework объединила в себе возможности этих продуктов, предоставив целостную среду разработки, в значительной степени упрощающую создание клиентских приложений.

Классы Windows Forms, имеющиеся в .NET Framework, предназначены для разработки графического интерфейса пользователя. Они упрощают создание командных окон, кнопок, меню, панелей инструментов и других графических элементов, обеспечивая гибкость, необходимую для удовлетворения меняющихся потребностей бизнеса.

Например, в .NET Framework есть ряд простых свойств, при помощи которых можно изменить атрибуты, определяющие внешний вид форм. В некоторых случаях система, в которой разрабатывается приложение, не позволяет менять эти атрибуты напрямую. Тогда платформа .NET Framework создает форму заново. Это лишь один пример того, как в платформе .NET Framework осуществляется интеграция интерфейса разработки, упрощается и систематизируется написание кода.

В отличие от элементов управления ActiveX элементы Windows Forms осуществляют доступ к компьютеру пользователя в режиме частичного доверия. Это означает, что бинарный или готовый код к одним ресурсам системы пользователя может иметь доступ (например к элементам графического интерфейса и части файлов), а к другим не может. Благодаря управлению доступом для кода многие приложения, которые раньше приходилось устанавливать в системе пользователя,

теперь можно разворачивать через Интернет. Приложение может выполнять функции локальной программы, но при этом работать в виде веб-страницы.

## 2 Практическая часть

### 2.1 Участие в выработке требований программного обеспечения

2.1.1 Use Case Diagram / Диаграмма вариантов использования  
Диаграмма вариантов использования (англ. use case diagram) в UML — диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. В нашем случае рассматриваются отношения между «продавцами», «менеджерами» и надлежащих им прецедентов. На диаграмме (рисунок 1) видно, что актер «менеджер» наследуется от «продавца».

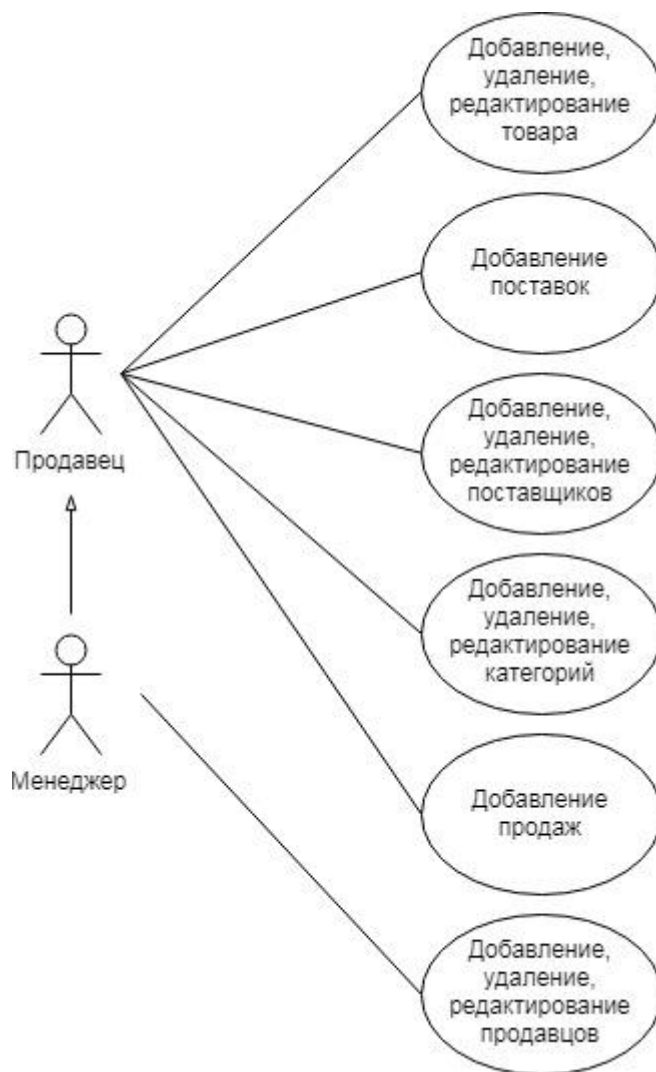


Рисунок 1 – Диаграмма вариантов использования

## 2.2 Участие в проектировании программного обеспечения с

### 2.2.1 Построение диаграмм этапа проектирования

2.2.1.1 Sequence Diagram / Диаграмма последовательности - диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта и взаимодействие актёров ИС в рамках какого-либо определённого прецедента. В нашем случае описан процесс авторизации и работы с данными Б/Д (рисунок 2)

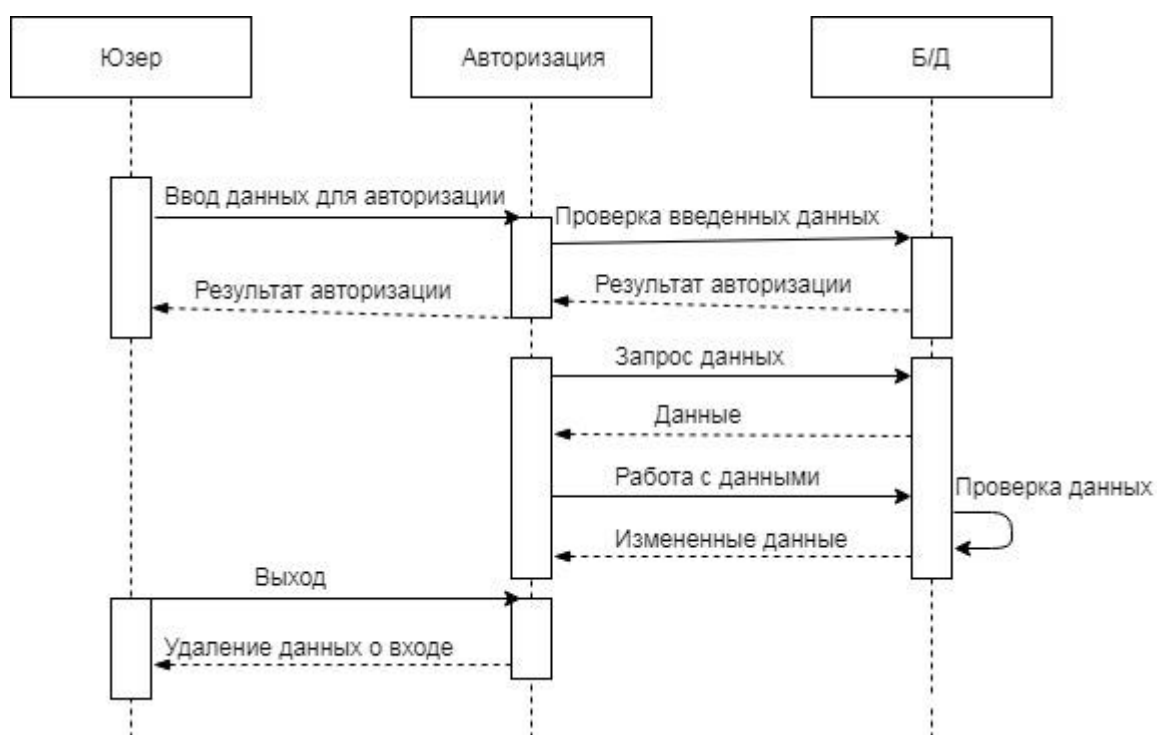


Рисунок 2 - Диаграмма последовательности

2.2.1.2 Statechart Diagram / Диаграмма состояний - определяет все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. На нашем примере это будет процесс авторизации пользователя (рисунок 3).

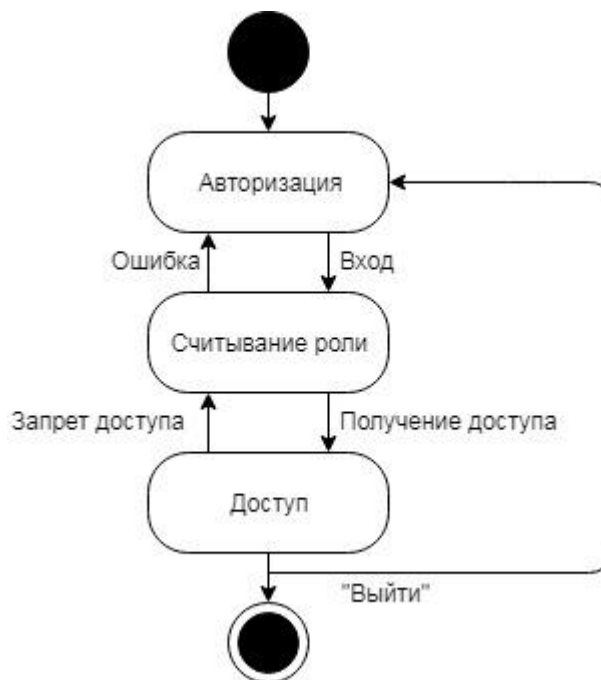


Рисунок 3 - Диаграмма состояний

2.2.1.3 Activity Diagram / Диаграмма активности – является представлением алгоритмов неких действий (активностей), выполняющихся в системе. На данном примере это добавление продавца (рисунок 4).

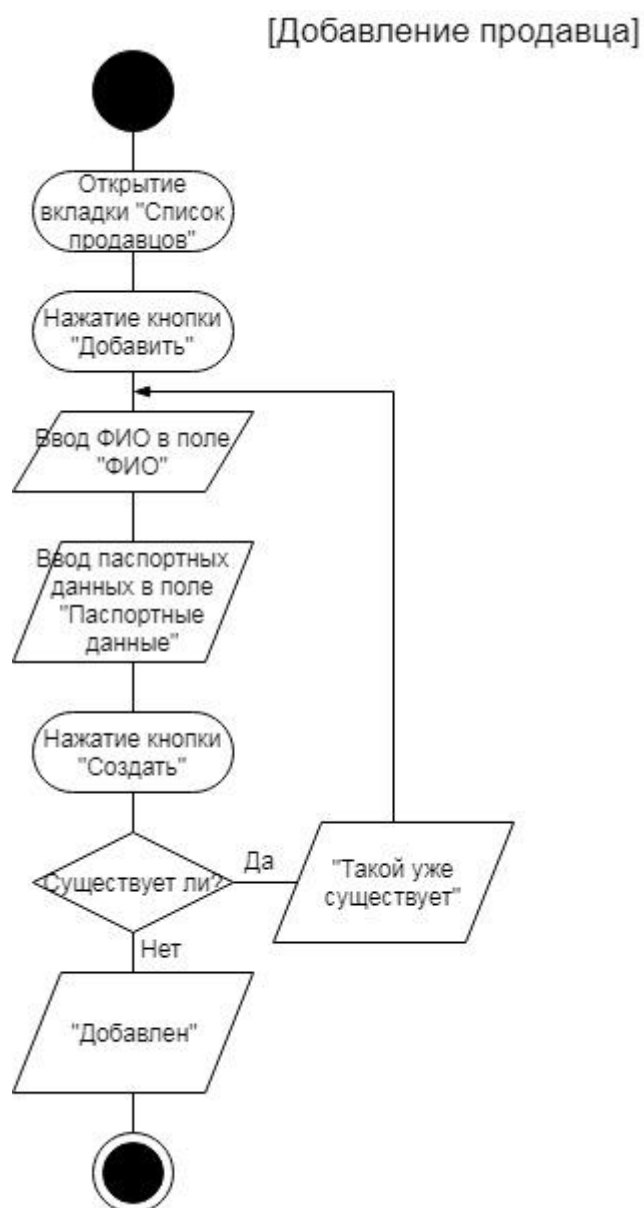


Рисунок 4 - Диаграмма активности

## 2.2.2 Проектирование базы данных SQL Server в Visual Studio (Server Explorer) и SQL Server Management Studio.

2.2.2.1 В качестве средства проектирования базы данных «music\_store» была выбрана SQL Server Management Studio (рисунок 5).

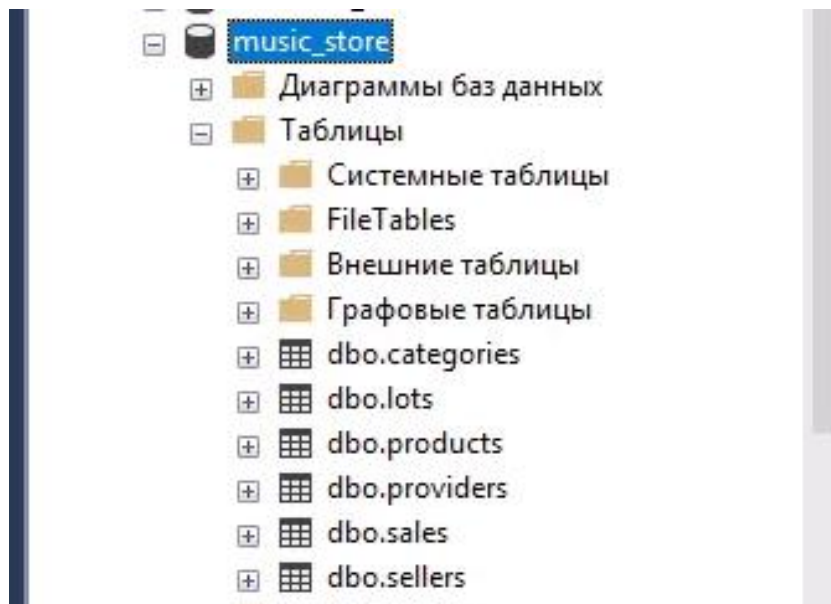


Рисунок 5 – База данных «music\_store» в SQL MS

### 2.2.2.2 Схема данных «music\_store» (рисунок 6).

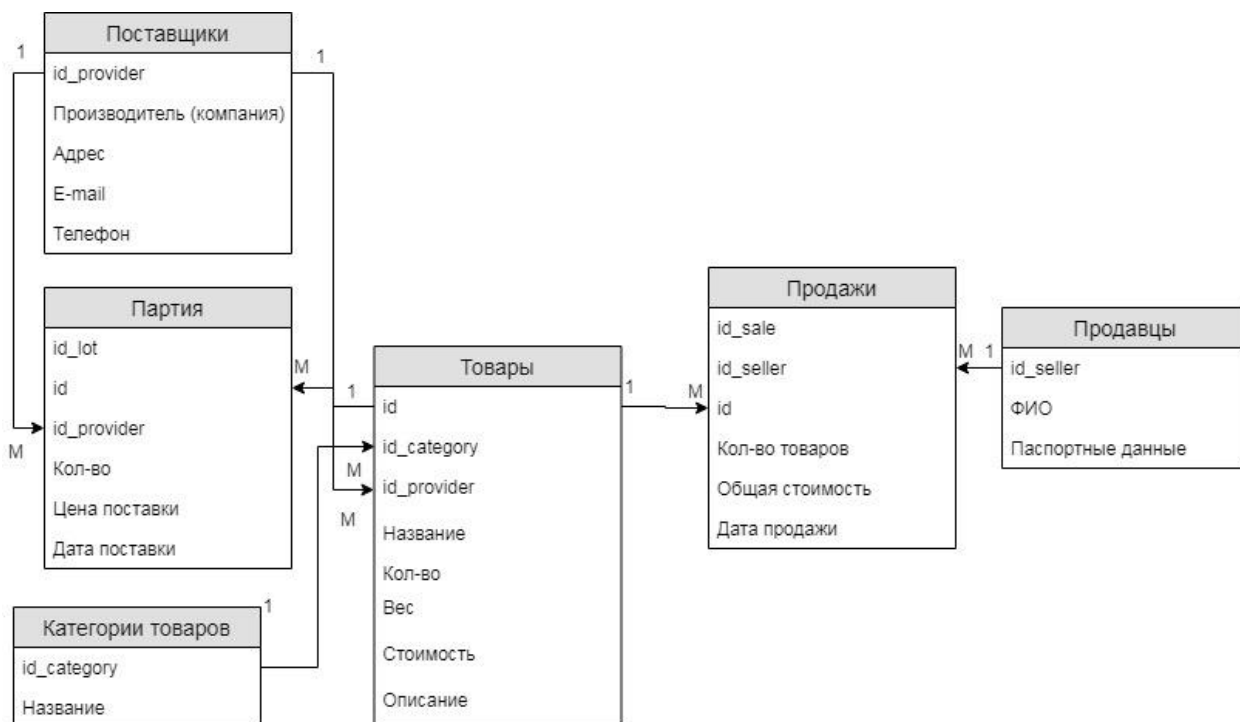


Рисунок 6 – Схема данных «music\_store»

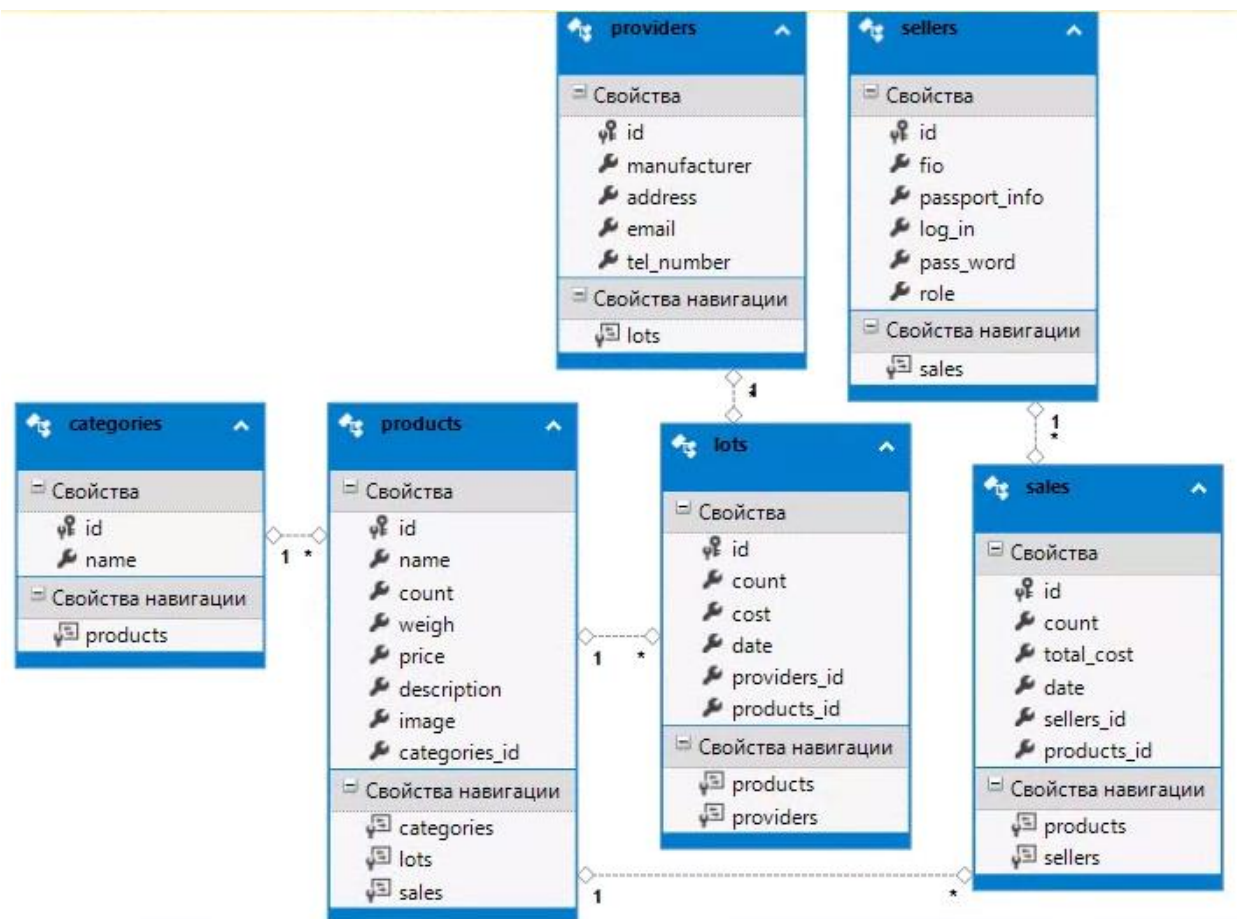


Рисунок 7 – ER модель в VS



2.2.2.3 Хранимые процедуры и функции. Процедура - объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В хранимых процедурах могут выполняться стандартные операции с базами данных (как DDL, так и DML). В качестве процедур было решено создать процесс регистрации (рисунок 8) и авторизации (рисунок 9), за функции (записанные как триггеры) были взяты процессы поставки товара на склад (рисунок 10) и их исчерпывание по мере продаж (рисунок 11).

```
USE [music_store]
GO
/***** Object: StoredProcedure [dbo].[Insert_User]    Script Date: 25.01.2019 11:14:07 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[Insert_User] @login NVARCHAR(30), @password NVARCHAR(20), @role NVARCHAR(20)
AS
BEGIN
    SET NOCOUNT ON;
    IF EXISTS(SELECT log_in FROM sellers WHERE log_in = @login)
    BEGIN
        SELECT -1 AS UserId -- Username exists.
    END
    ELSE
    BEGIN
        INSERT INTO [sellers]([log_in],[pass_word],[role])
        VALUES(@login,@password,@role)
        select 1
        --SELECT SCOPE_IDENTITY() AS UserId -- UserId
    END
END
```

Рисунок 8 – ХП регистрации

```
USE [music_store]
GO
/***** Object: StoredProcedure [dbo].[LoginByUs
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[LoginByUsernamePassword]
@login varchar(50),
@password varchar(50)
AS
BEGIN
SELECT log_in, pass_word
FROM sellers
WHERE log_in = @login
AND pass_word = @password
END
```

Рисунок 9 – ХП авторизации

```
USE [music_store]
GO
/***** Object: Trigger [dbo].[trigger_ModifyBudget]    Script Date:
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[trigger_ModifyBudget]
ON [dbo].[lots] AFTER INSERT
AS
BEGIN
UPDATE products
SET products.[count] = [inserted].[count] + products.[count]
FROM products
JOIN inserted
ON products.id = inserted.products_id
END
```

Рисунок 10 – Триггер пополнения товара после поставки

```
USE [music_store]
GO
/***** Object: Trigger [dbo].[trigger_ModifySALES]    Script Date:
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[trigger_ModifySALES]
ON [dbo].[sales] AFTER INSERT
AS
BEGIN
UPDATE products
SET products.[count] = [products].[count] - [inserted].[count]
FROM products
JOIN inserted
ON products.id = inserted.products_id
END
```

Рисунок 11 – Триггер убыли товара после продажи

## 2.3 Участие в интеграции программных модулей

### 2.3.1 Архитектура проектирования MVC

Model-view-controller (MVC)— шаблон проектирования, с помощью которого его компоненты (модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем) разделены на три отдельных так, что модификация одного из них оказывает минимальное воздействие на остальные. Модель предоставляет данные и

методы работы с ними, реагирует на запросы, изменяя своё состояние. Представление отвечает за визуализацию. Часто в качестве представления выступает форма с графическими элементами. Контроллер обеспечивает ввод данных пользователем и использует модель и представление для реализации необходимой реакции

Главным преимуществом концепции MVC является разделение логики управления приложения, получения данных и их отображения.

В ходе исследования необходимо изучить методы построения архитектуры программной системы и создать эффективную программную систему с использованием шаблона Model-View-Controller.

Для достижения поставленной цели, необходимо решение следующих исследовательских задач:

- Разработка формального описания функционирования программной системы на основе шаблона MVC.
- Разработка программной системы с использованием шаблона MVC.
- Оценка эффективности программной архитектуры с использованием шаблона MVC.

2.3.2 Разработка интерфейса приложения View. На основе сгенерированных страниц html и привязанным к ним css объектов был изменен дизайн приложения (рисунок 12, 13, 14).

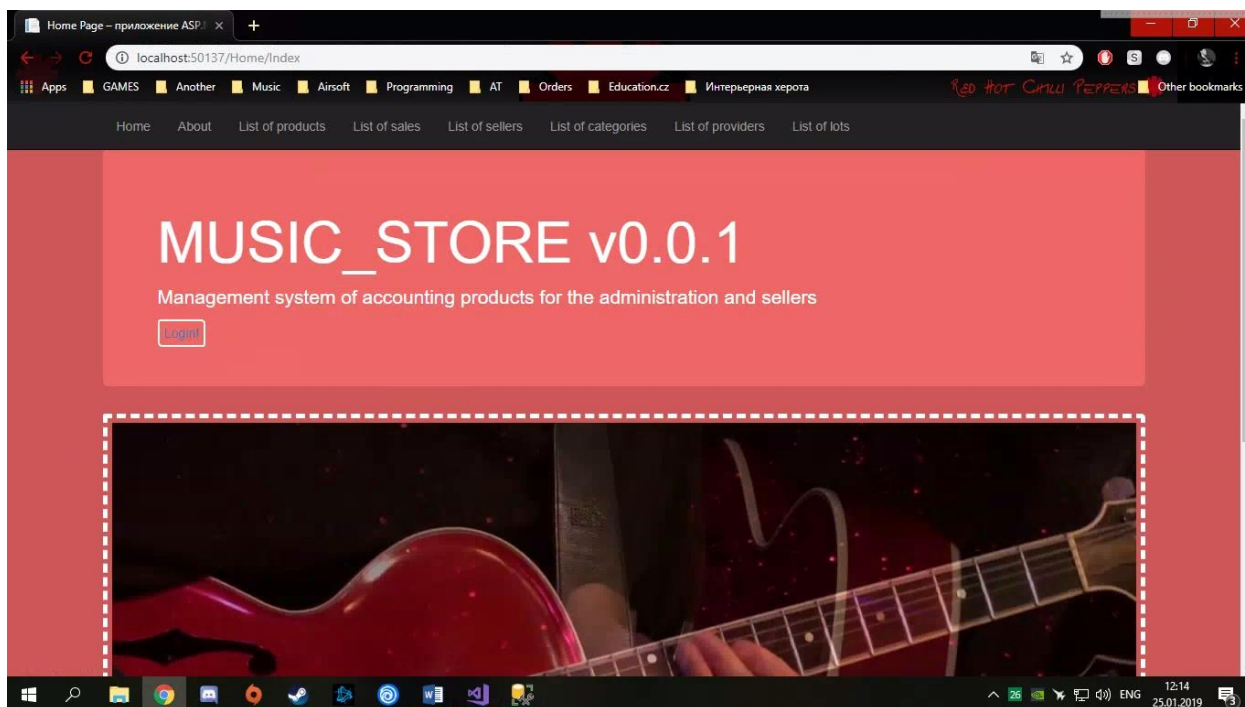


Рисунок 12 – Дизайн стартовой страницы

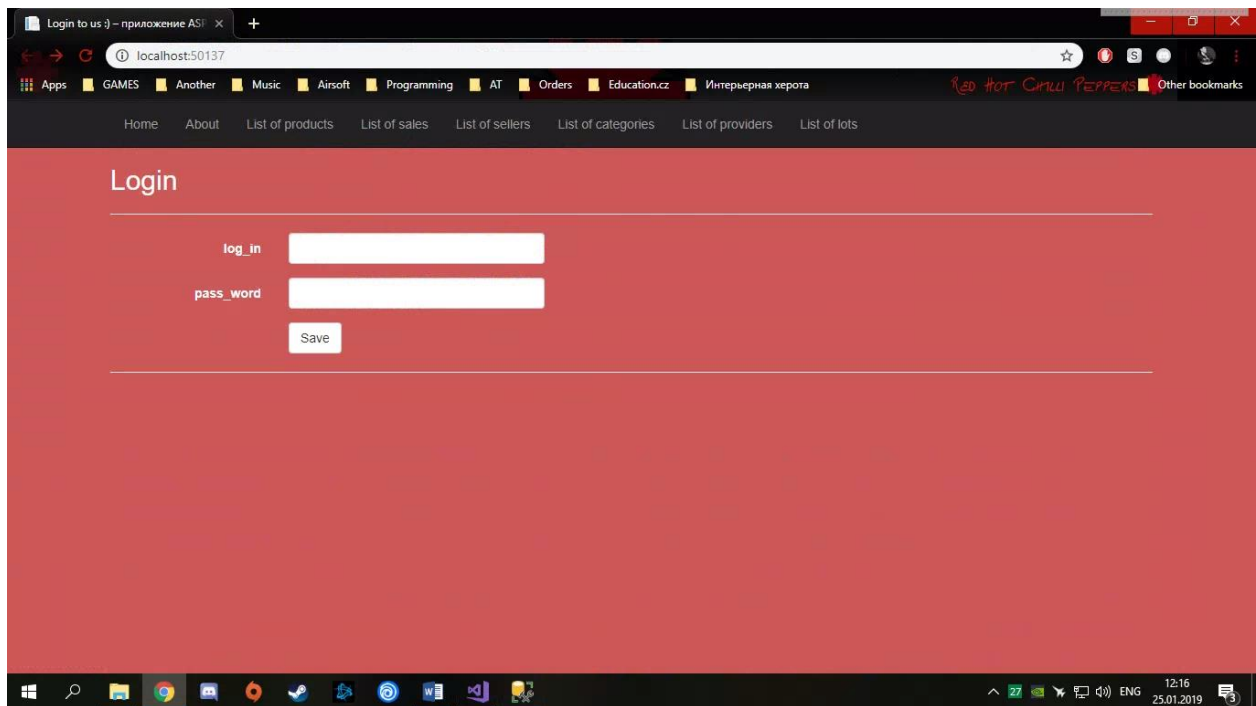


Рисунок 13 – Дизайн страницы авторизации

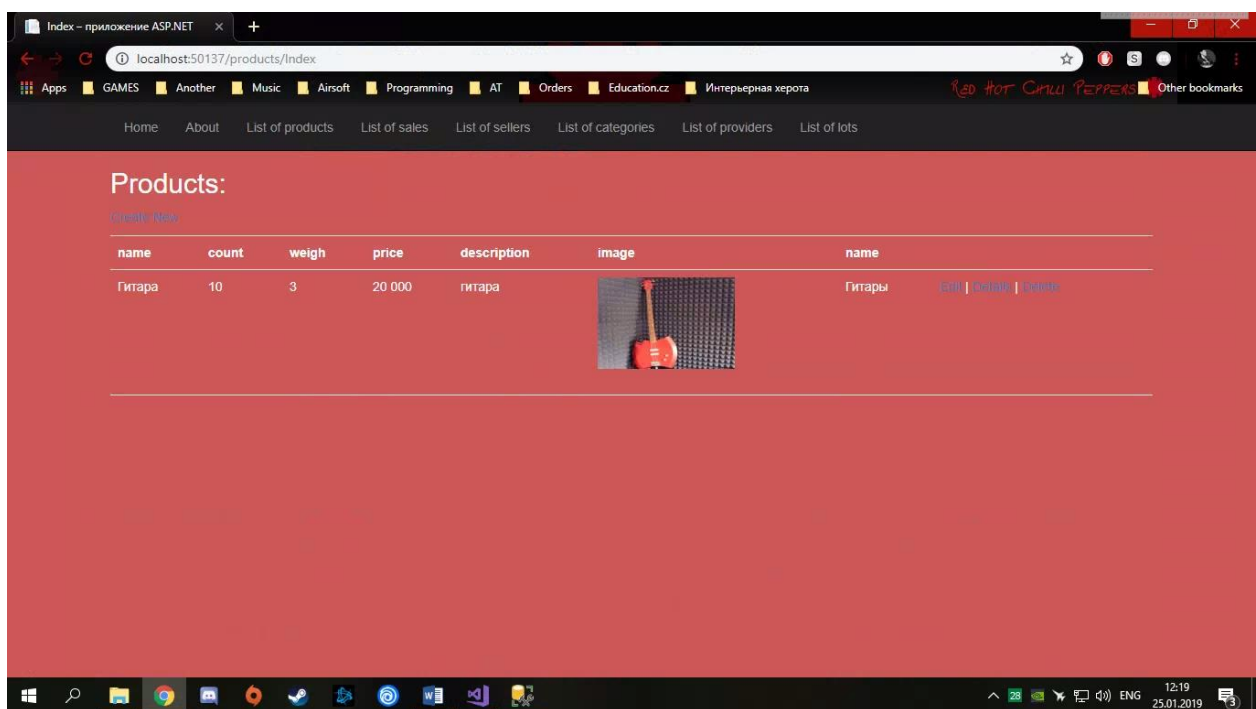
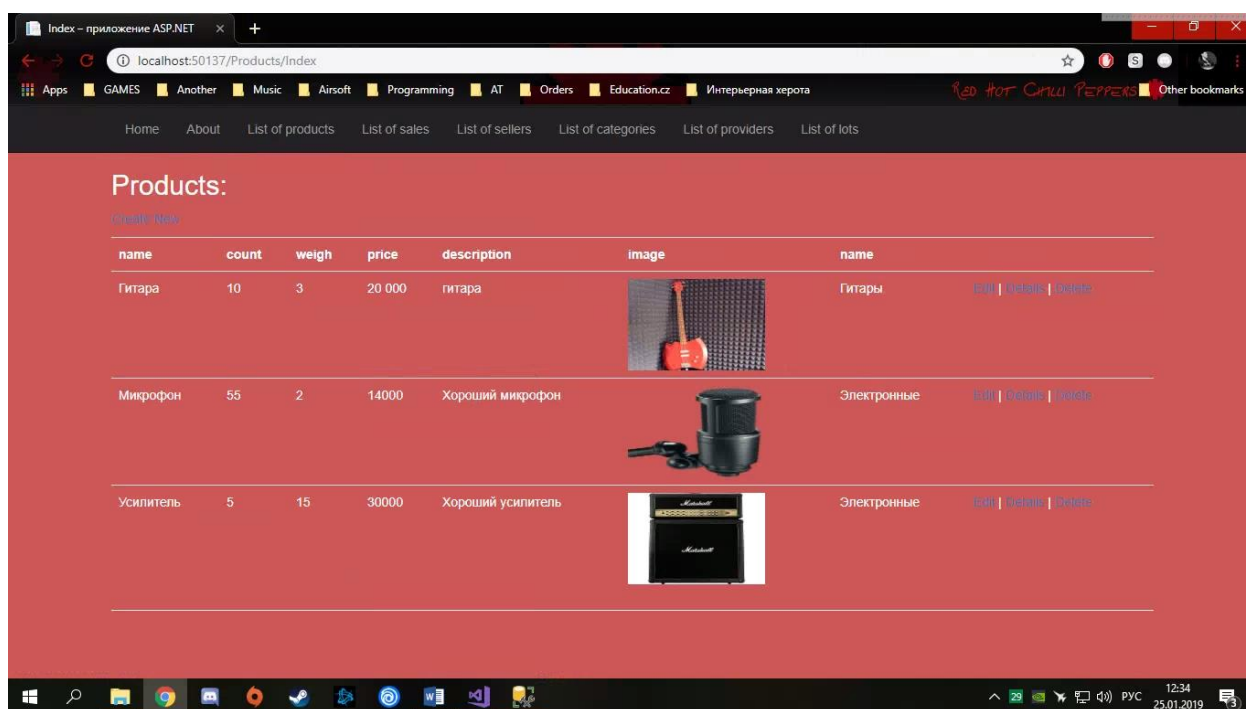


Рисунок 14 – Дизайн страницы товаров

### 2.3.3 Разработка Controller:

2.3.3.1 Вывод информации из всех таблиц базы данных. Из таблицы товаров (рисунок 15), продаж (рисунок 16), продавцов (рисунок 17), категорий (рисунок 18), поставщиков (рисунок 19), поставок (рисунок 20), а также смежной таблицы отношения даты и кол-ву продаж в этот день (рисунок 21).






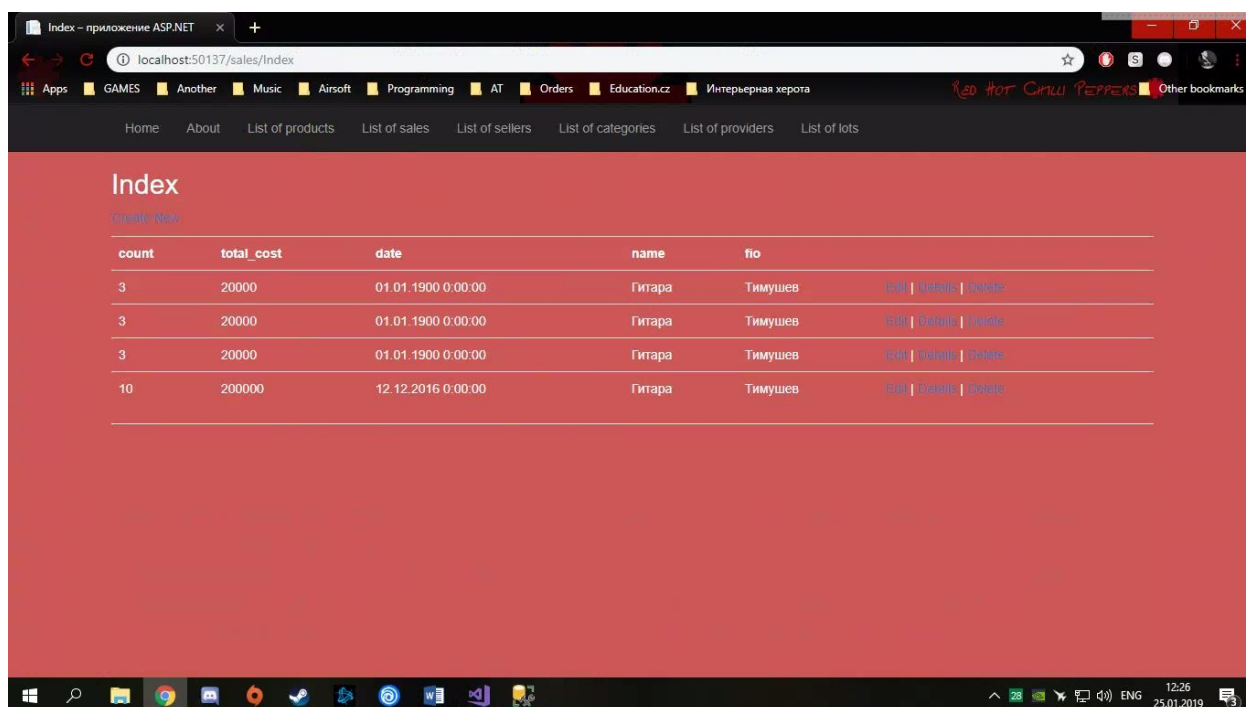
name	count	weigh	price	description	image	name
Гитара	10	3	20 000	гитара		Гитары
Микрофон	55	2	14000	Хороший микрофон		Электронные
Усилитель	5	15	30000	Хороший усилитель		Электронные

Рисунок 15 – Данные таблицы товаров



count	total_cost	date	name	fio
3	20000	01.01.1900 0:00:00	Гитара	Тимушев
3	20000	01.01.1900 0:00:00	Гитара	Тимушев
3	20000	01.01.1900 0:00:00	Гитара	Тимушев
10	200000	12.12.2016 0:00:00	Гитара	Тимушев

Рисунок 16 – Данные таблицы продаж



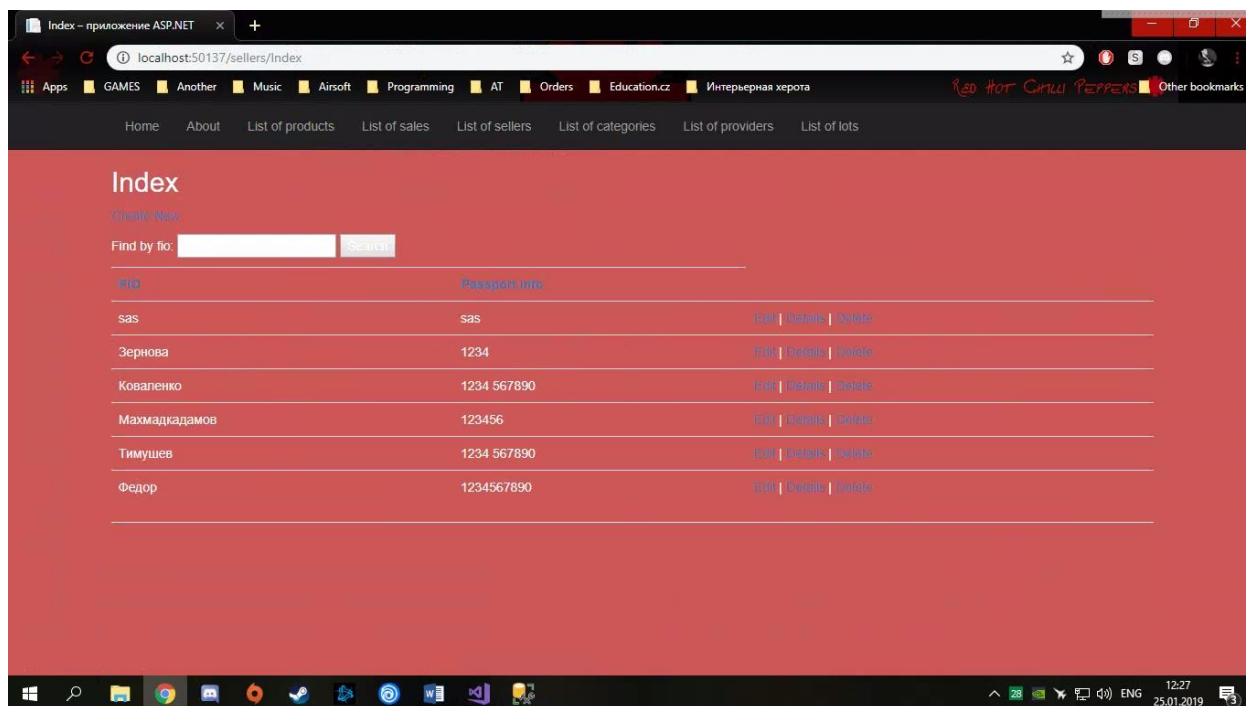


Рисунок 17 – Данные таблицы продавцов

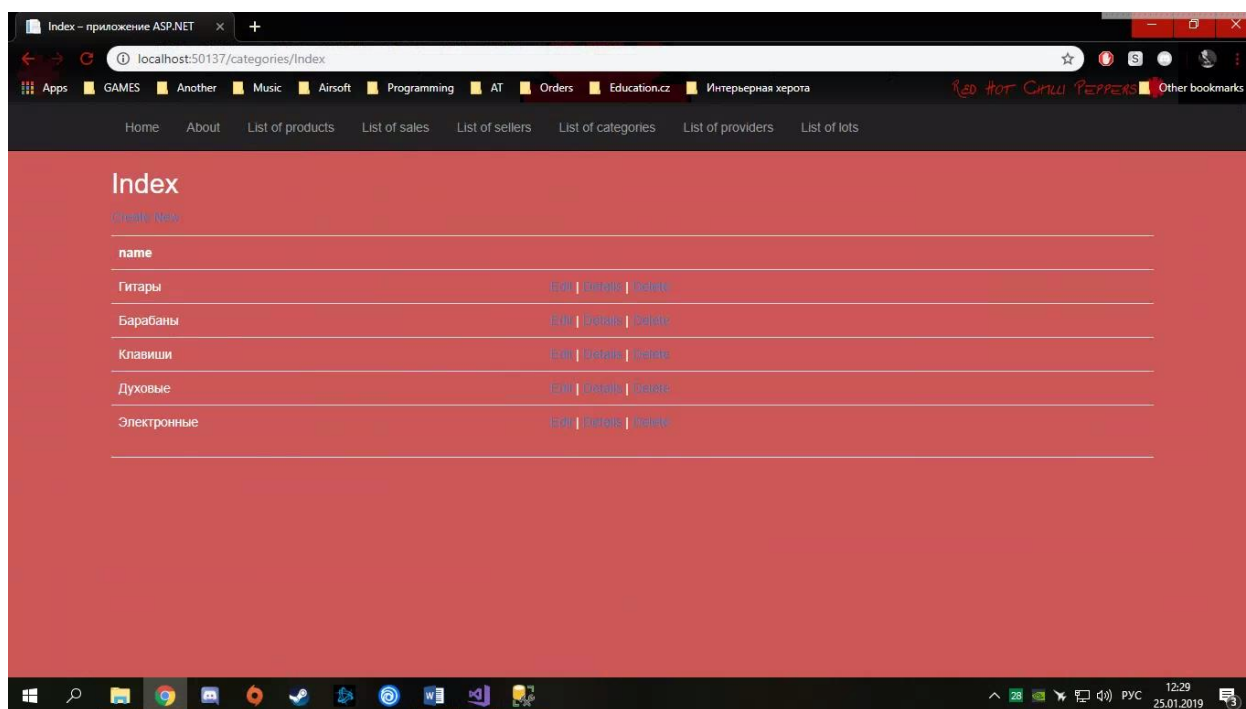


Рисунок 18 – Данные таблицы категорий

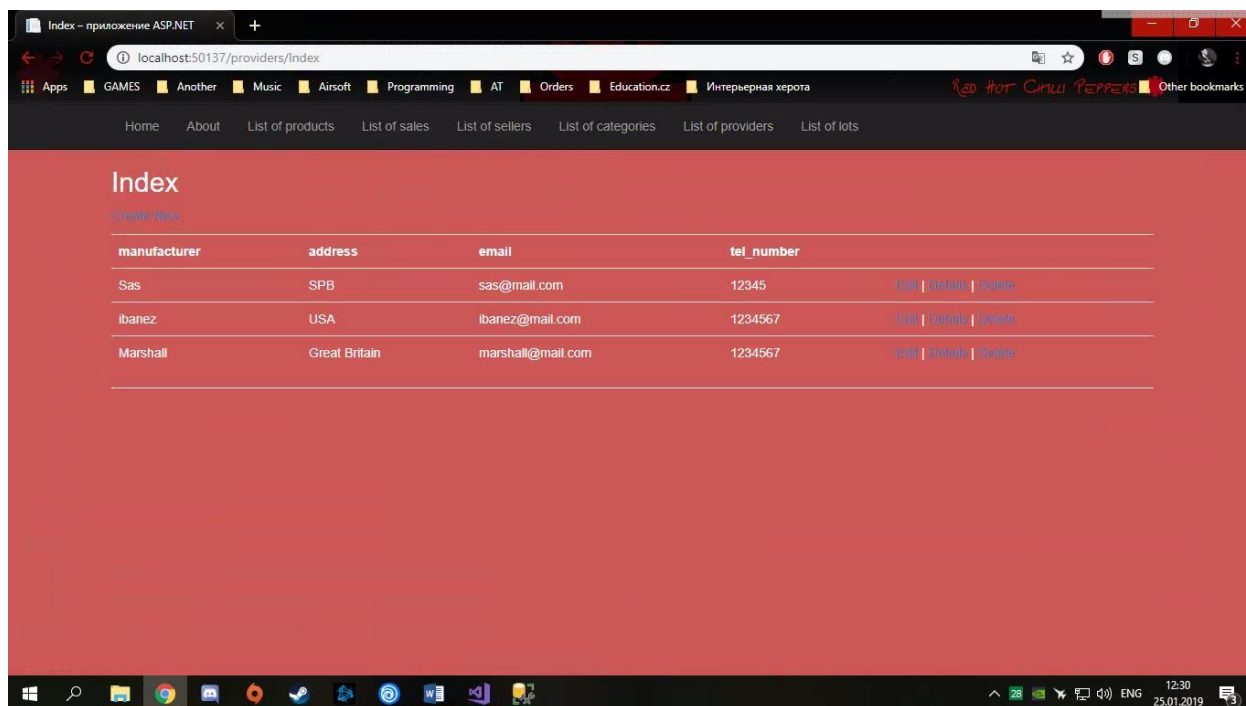


Рисунок 19 – Данные таблицы поставщиков

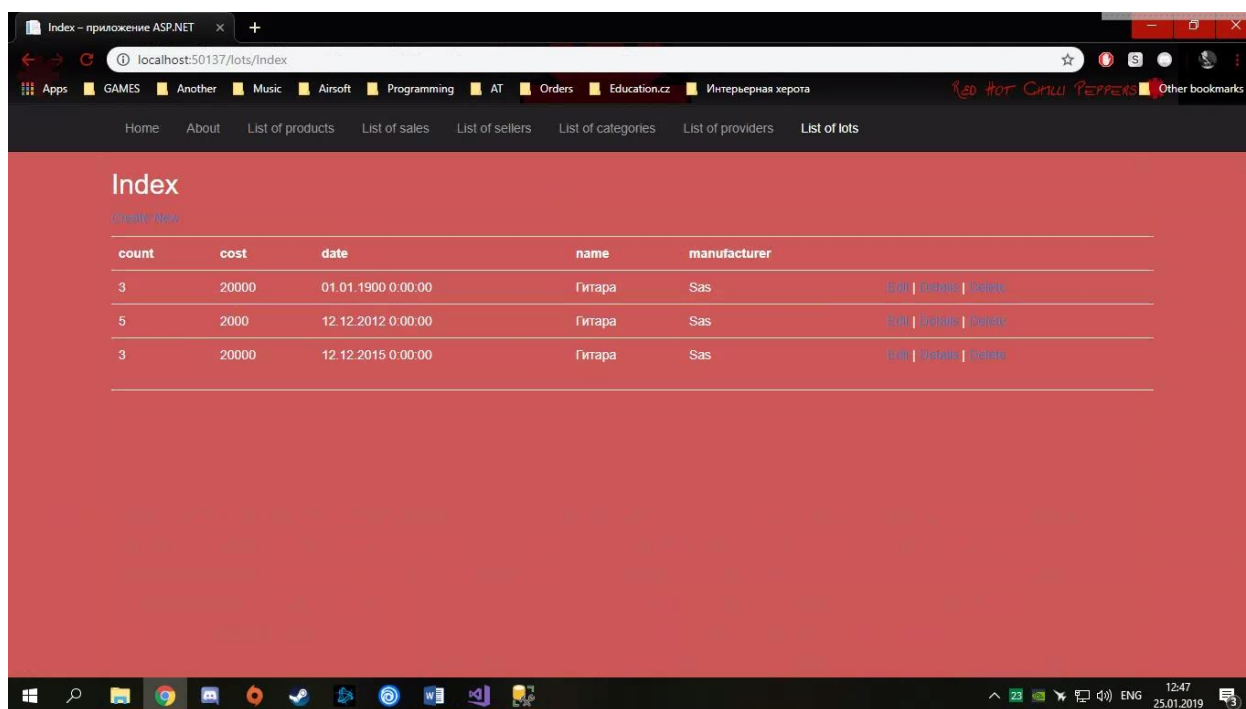


Рисунок 20 – Данные таблицы поставок

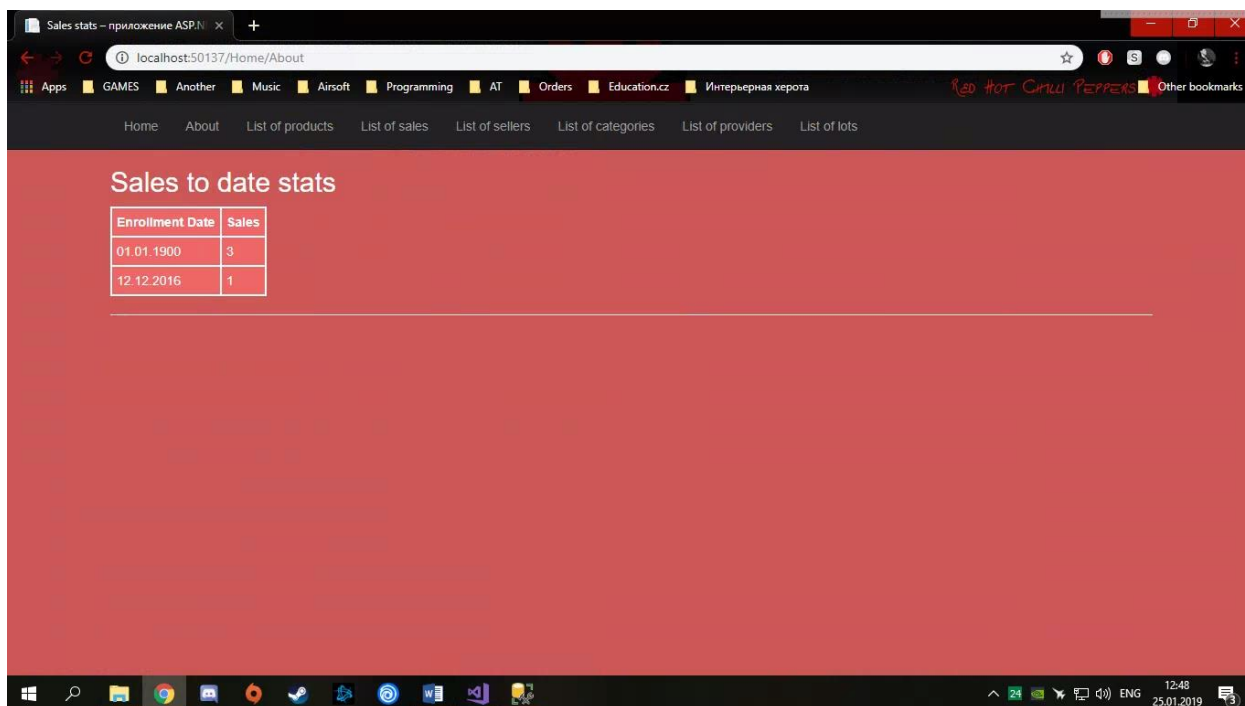


Рисунок 21 – Смежная таблица количества продаж на дни

2.3.3.2 Обновление, редактирование, удаление записей. Рассмотрим все упомянутые действия на примере товаров. Добавление новой записи в исходную таблицу (рисунок 22) данных происходит при помощи кнопки «Create» с последующим переходом на страницу добавления, внесением необходимых данных (рисунок 23) и нажатием на кнопку «Create». Теперь можно увидеть новую запись (рисунок 24).

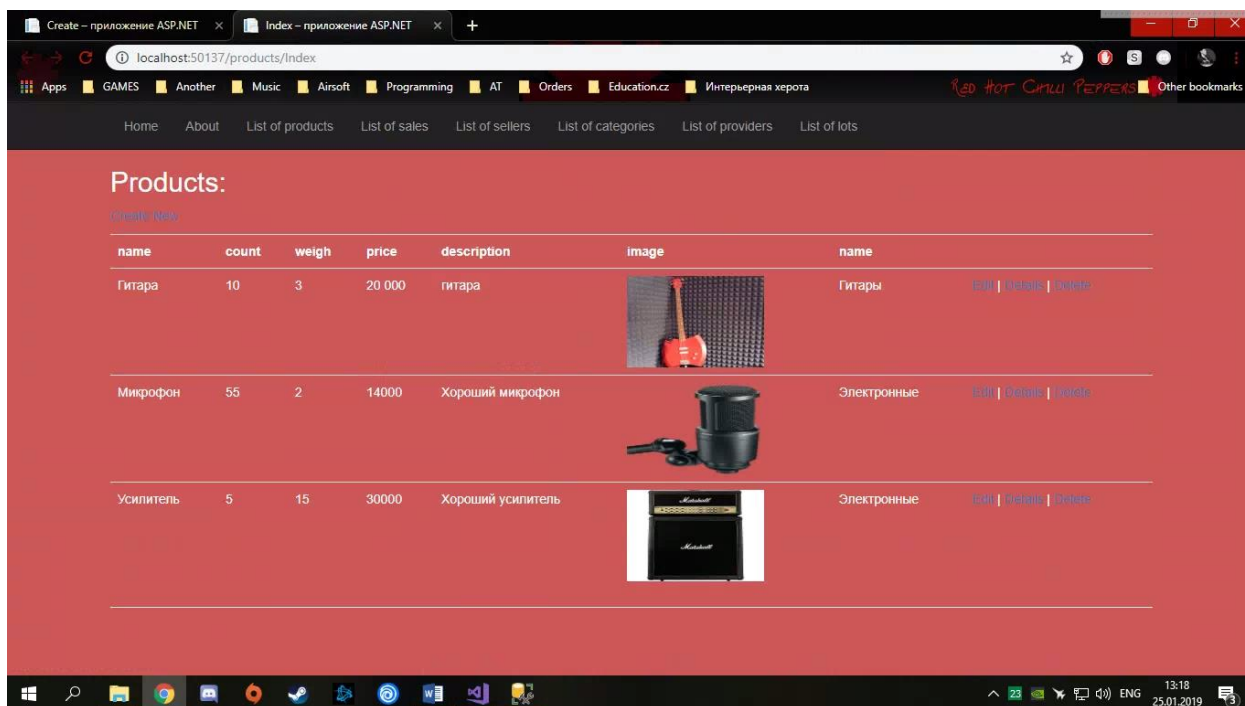


Рисунок 22 – Исходная таблица данных таблицы «товары»



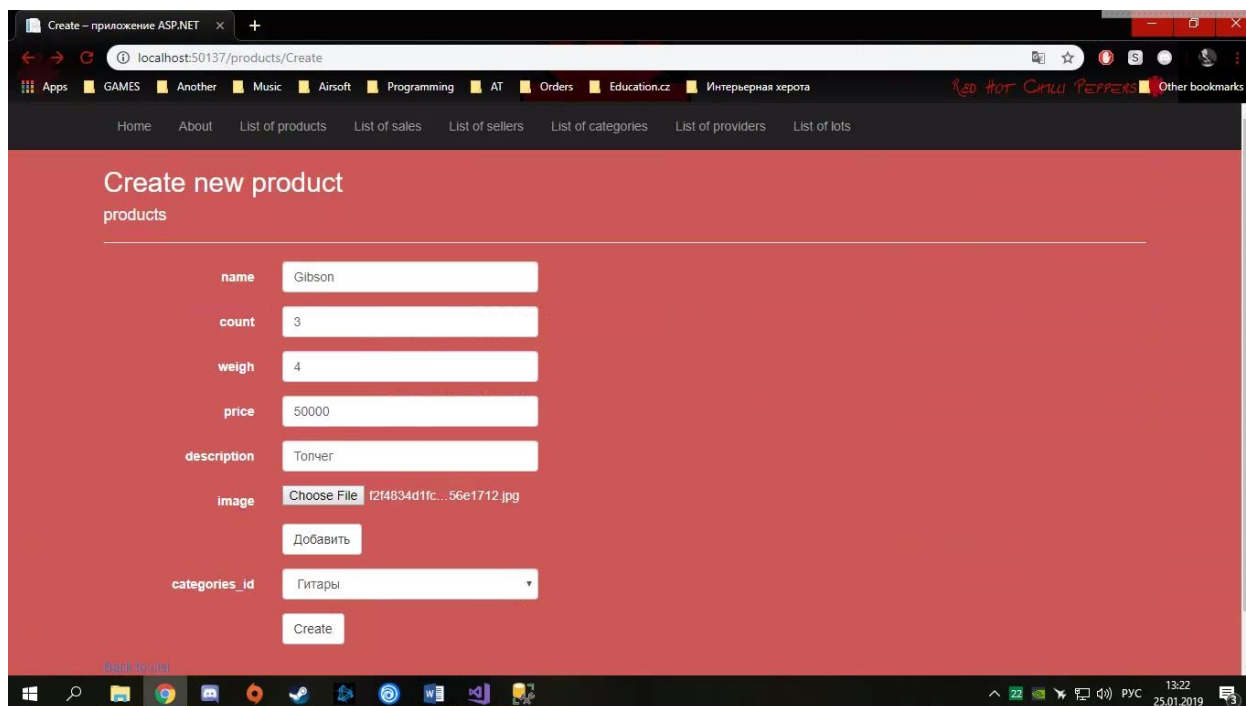


Рисунок 23 – Заполнение полей данными

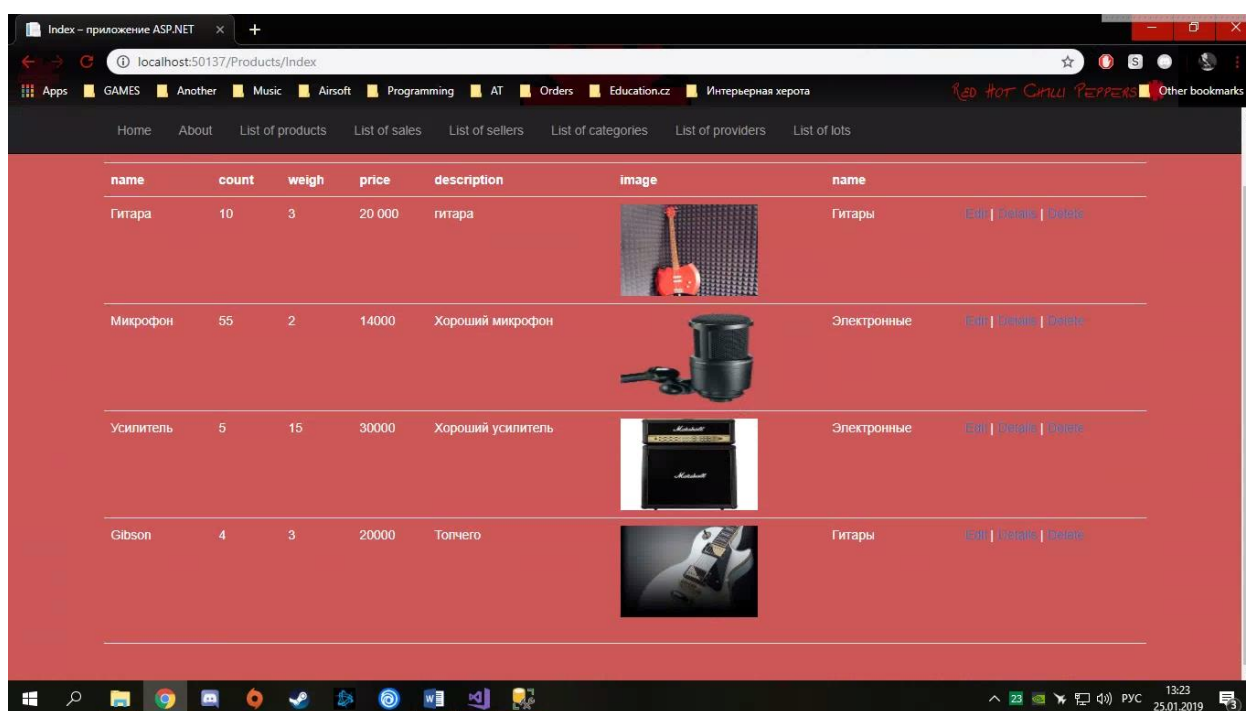
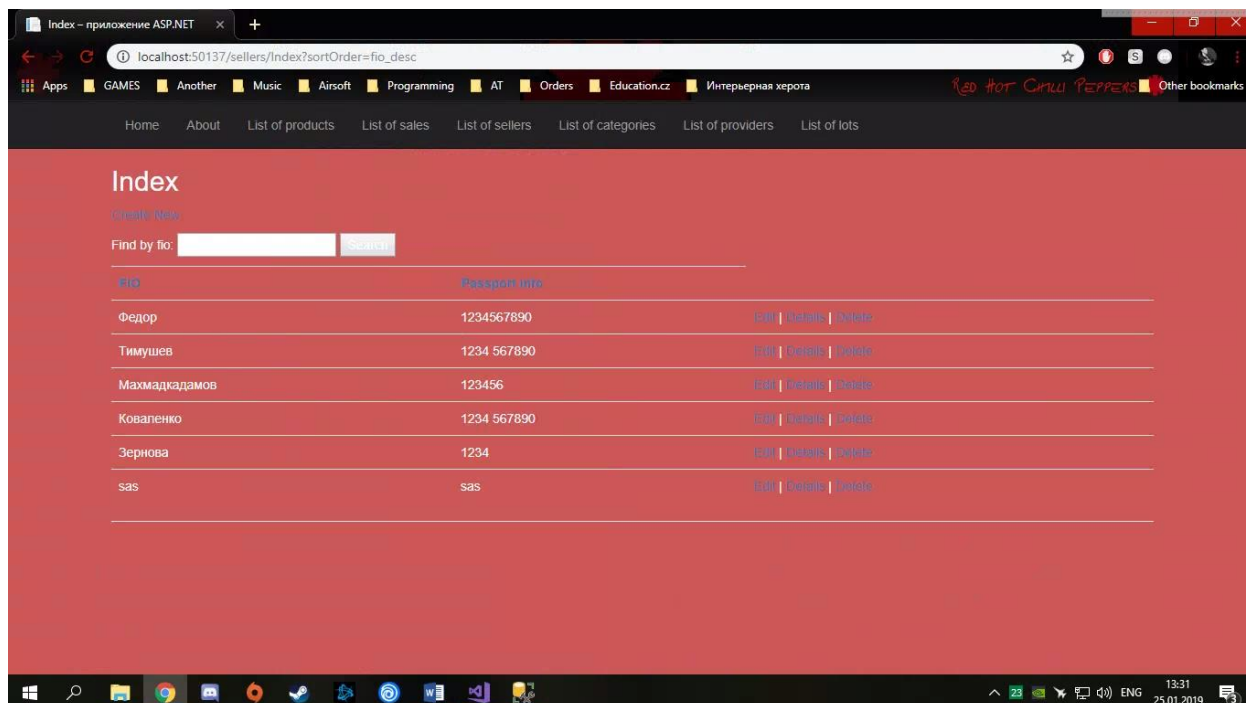


Рисунок 24 – Новая запись в таблице

2.3.3.3 Фильтрацию и сортировку мы рассмотрим на примере таблицы «продавцы». Вид исходной таблицы (рисунок 25) будет изменен нажатием на название поля (рисунок 26). Сортировка произойдет после ввода в поле поиска необходимых данных и нажатием на кнопку «Search» (рисунок 27).



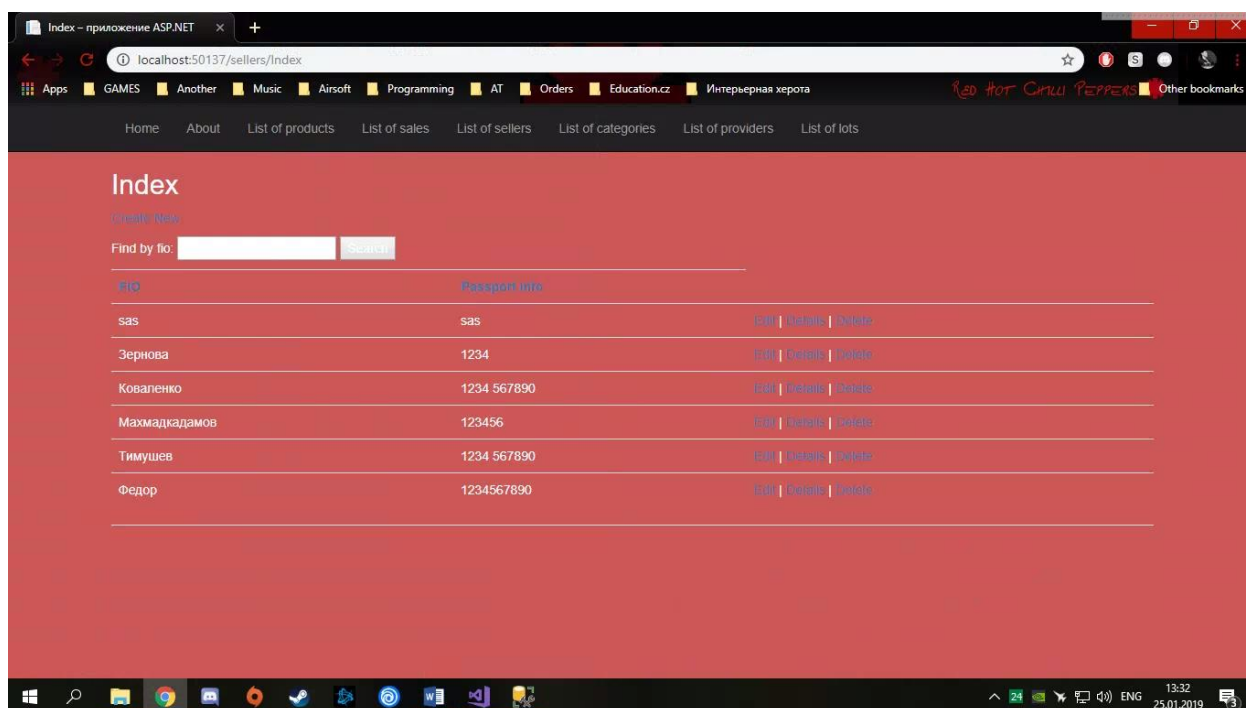
Index

Create New

Find by fio:

ID	Passport info	
Федор	1234567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Тимушев	1234 567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Махмадқадамов	123456	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Коваленко	1234 567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Зернова	1234	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
sas	sas	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

Рисунок 25 – Исходный формат таблицы



Index

Create New

Find by fio:

ID	Passport info	
sas	sas	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Зернова	1234	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Коваленко	1234 567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Махмадқадамов	123456	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Тимушев	1234 567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>
Федор	1234567890	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

Рисунок 26 – Отфильтрованная таблица

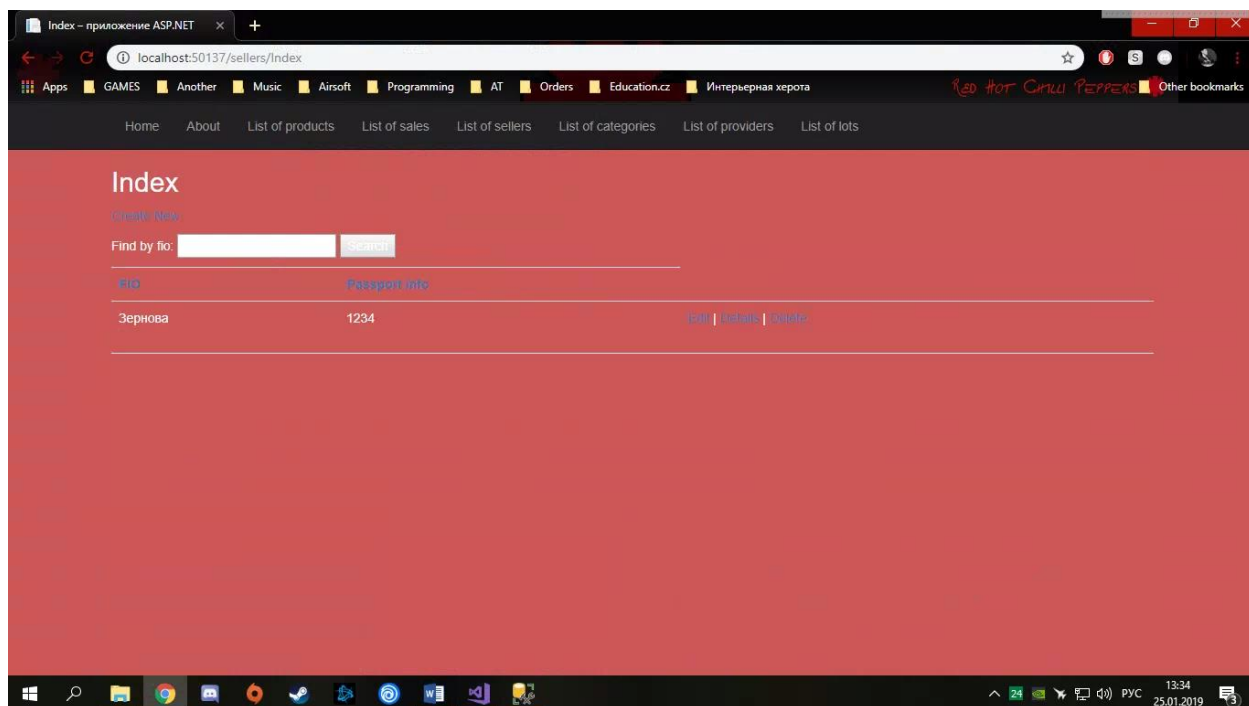


Рисунок 27 – Отсортированная таблица

## **ЗАКЛЮЧЕНИЕ**

Разработанная в ходе прохождения учебной практики программа удовлетворяет всем требованиям технического задания, что подтверждается протоколом испытаний, получены навыки работы с ASP.NET и MVC.

Разработанная программа может быть использована в обучающих и коммерческих целях.

