

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

РАЗРАБОТКА АСОИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ C#, ASP.NET, SQL,
Bootstrap4 ДЛЯ МАГАЗИНА ПО ПРОДАЖЕ ГИТАР «AmDm.by»

Курсовое проектирование
по дисциплине «Проектирование автоматизированных систем»

КП.1-53 01 02.10030282

Исполнитель _____ Казымов Н.А., АСОИ-191

(подпись)

Руководитель _____ Крутолевич С.К.

(подпись)

Дата допуска к защите _____

Дата защиты _____

Оценка _____

Могилёв 2022

Факультет электротехнический
УТВЕРЖДАЮ
Зав. кафедрой «ПОИТ» Кутузов В.В.

«_10_»_сентября__2022_г.

ЗАДАНИЕ

по курсовому проектированию

Студенту гр. АСОИ-191 Казымову Никите Александровичу

1. **Тема проекта** Разработка АСОИ с использованием технологий C#, ASP.NET, SQL, Bootstrap4 для магазина по продаже гитар «AmDm.by».

2. **Сроки сдачи студентом законченного проекта** согласно графику учебного процесса.

3. **Исходные данные к проекту:** Входные и выходные документы учреждения, нормативно-справочная документация, методические указания, язык моделирования UML.

4. **Содержание расчетно-пояснительной записки:**

Введение;

1 Анализ бизнес-процессов. ; (Приводится структура формируемых документов в виде таблиц)

2 Проектирование структуры базы данных; (Приводится структура БД в виде таблиц)

3 Проектирование архитектуры проекта; (Приводится структура Интерфейсов, запросов и процедур в виде таблиц)

4 Управление процессом разработки программного обеспечения; (Таблицы трудоемкости и календарный график)

4.1 Определение трудоемкости разработки

4.2 Отчет о разработке программных компонентов; (Приводятся фрагменты разработанного кода программ)

5 Руководство пользователя (приводится описание процесса формирования документов с Формами, Отчетами (Копии экрана). все поля в формах и отчетах заполнены.

Заключение по проекту;

Список литературы.

В записке не приводятся диаграммы UML

5. **Перечень графического материала**

Диаграмма классов. Структура базы данных, 1 лист формата А3;

Диаграммы бизнес-процессов, 1 лист формата А3;

Диаграмма вариантов использования, 1 лист формата А4;

Диаграмма классов. Сущности , 1 лист формата А4;

Диаграмма классов. Контроллеры , 1 лист формата А3;

Диаграмма последовательности, 2 листа формата А3;

Диаграмма состояний, 1 лист формата А3;

7. **Дата выдачи задания** __10 сентября 2022__.

8. **Календарный график работы над проектом**

согласно графику учебного процесса.

Руководитель проекта _____

Задание принял к исполнению __10 сентября 2022__

(подпись студента) _____

Содержание

Введение	4
1 Анализ бизнес-процессов	5
1.1 Обоснование начала разработки АСОИ.....	5
1.2 Функциональные требования к АСОИ	5
1.3 Прочие требования к АСОИ.....	6
2 Проектирование структуры базы данных	7
3 Проектирование архитектуры проекта.....	10
3.1 Разработка диаграммы взаимодействия.....	10
3.2 Структура классов АСОИ.....	10
3.3 Диаграмма состояний.....	14
4 Управление процессом разработки программного обеспечения	15
4.1 Определение трудоёмкости разработки.....	15
4.2 Отчёт о разработке программных компонентов	16
5 Тестирование разработанного ПО	33
Заключение.....	41
Список использованных источников	42

					КП.1-53 01 02.10030282							
Изм.	Лист	№ докум.	Подпись	Дата								
Разраб.		Казымов Н.А.			Разработка АСОИ Курсовой проект				Лит.	Лист	Листов	
Провер.		Крутолевич С.К.									3	42
Реценз.									БРУ, гр. АСОИ-191			
Н. контр.												
Утверд.												

Введение

Темой курсового проектирования является разработка АСОИ для оптимизации рабочих процессов магазина по продаже гитар «AmDm.by».

В результате внедрения системы улучшится скорость и качество обслуживания клиентов, уменьшится время на оформление и комплектацию заказа, что, в свою очередь, способствует экономическому росту организации.

Пояснительная записка к курсовому проектированию содержит 6 разделов:

- анализ бизнес-процессов – приводится структура формируемых документов в виде таблиц;
- проектирование структуры базы данных – приводится структура БД в виде таблиц;
- проектирование архитектуры проекта – приводится структура интерфейсов, запросов и процедур в виде таблиц;
- управление процессом разработки программного обеспечения – таблицы трудоёмкости и календарный график;
- тестирование разработанного ПО – раздел включает в себя информацию по тестированию ПО;
- руководство пользователя – приводится описание процесса формирования документов с формами;
- заключение – содержит обобщение выполненной работы.

1 Анализ бизнес-процессов

1.1 Обоснование начала разработки АСОИ

По требованию заказчика в качестве объекта автоматизации был выбран магазин по продаже гитар «AmDm.by».

Система создаётся на основании следующих документов:

- отчёт по продажам;
- чек заказа.

Создание интернет-магазина – один из наиболее выгодных и перспективных инструментов онлайн-бизнеса, позволяющий при меньших затратах охватить большую аудиторию. Интернет-торговля на собственной платформе связана с существенно меньшим числом издержек, чем торговля в магазине. Такой вид бизнеса стал особенно актуален в условиях опасной эпидемиологической обстановки, когда обычные магазины сталкивались с ограничениями деятельности вплоть до закрытия

Проектируемую систему планируется использовать на рабочих местах сотрудников магазина музыкального оборудования.

В музыкальном магазине работают люди со средним и высшим образованием.

1.2 Функциональные требования к АСОИ

Назначение проекта – проектирование многопользовательской системы. Систему предполагается создать для улучшения качества обслуживания покупателей, учёта товарно-материальных ценностей и ускорения работы персонала магазина. Так как система позволяет увеличить скорость обслуживания, то возрастает число обслуживаемых покупателей.

Критерии оценки достижений целей системы:

- увеличение количества клиентов за счёт уменьшения времени обслуживания клиентов;
- улучшение качества обслуживания клиентов;
- увеличение скорости обработки информации о движении тмц для создания отчётов.

Система должна быть адаптивной к изменениям и простой в использовании, для большего ее распространения.

Система должна удовлетворять следующим требованиям:

- надёжность хранения данных;
- безопасность хранения данных;
- доступность системы с любого компьютера корпоративной сети;
- защищённости информации, хранящейся в системе, от внешних воздействий, хакерских атак и других аварийных ситуаций;
- квалификация персонала (персонал должен быть обучен работе с ИС).

В качестве функциональных требований выступают требования по формированию документов «Чек заказа» и «Отчёт по продажам».

Таблица 1.1 – Поля документа «Чек заказа»

Поле в документе	Обозначение
Номер заказа	[Id]
Покупатель	[Customer.Surname]
Стоимость	[TotalCost]
Тип оплаты	[PaymentType.Name]
Оформивший сотрудник	[Employee.Surname]
Дата оформления	[OrderDate]
Дата оплаты	[PaymentDate]

Таблица 1.2 – Поля документа «Отчёт по продажам»

Поле в документе	Обозначение
Артикул товара	[ProductNumber]
Количество проданных экземпляров / Кол-во	[SalesCount]
Цена	[Price]
Выручка	[Gain]
Наименование	[Name]

1.3 Прочие требования к АСОИ

Минимальные системные требования:

- процессор: Intel Core i3-4160 или соответствующий ему аналог от AMD;
- оперативная память: 2 ГБ;
- жёсткий диск: 512 ГБ HDD;
- видеокарта: GeForce 8800 (с 512 МБ видеопамяти) или Radeon HD3850 (с 512 МБ видеопамяти);
- жёсткий диск: 1 ТБ;

Рекомендуемые системные требования:

- процессор: Intel Core i5 9700KF 3.6GHz;
- оперативная память: 8 ГБ (для Windows 10);
- видеокарта: GeForce GTX 1050ti (с 4 ГБ видеопамяти);
- жесткий диск: 1 ТБ HDD.

2 Проектирование структуры базы данных

На основании всей предоставленной документации была разработана структура базы данных. Все таблицы приведены к третьей нормальной форме. Ниже приведена структура каждой из таблиц базы данных.

Таблица 2.1 – Структура таблицы Users

Key	Name	Type	Not Null	Unique	Len
PK	u_id	int	+	+	-
	u_login	varchar	+	+	30
	u_password	varchar	+	-	18

Таблица 2.2 – Структура таблицы ProductTypes

Key	Name	Type	Not Null	Unique	Len
PK	pt_id	int	+	+	-
	pt_name	nvarchar	+	+	100

Таблица 2.3 – Структура таблицы Products

Key	Name	Type	Not Null	Unique	Len
PK	p_number	int	+	+	-
FK	p_type	int	+	-	-
	p_name	nvarchar	+	+	200
	p_producer	nvarchar	+	-	200
	p_price	int	+	-	-
	p_description	nvarchar(max)	-	-	300
	p_amount	int	+	-	-
	p_is_deleted	bit	+	-	-

Таблица 2.4 – Структура таблицы Customers

Key	Name	Type	Not Null	Unique	Len
PK	c_id	int	+	+	-
	c_surname	nvarchar	+	-	50
	c_name	nvarchar	+	-	50
	c_patronymic	nvarchar	+	-	50
	c_phone	varchar	+	-	13

Таблица 2.5 – Структура таблицы OrderStatuses

Key	Name	Type	Not Null	Unique	Len
PK	os_id	int	+	+	-
	os_name	nvarchar	+	+	-

Таблица 2.6 – Структура таблицы ShoppingCarts

Key	Name	Type	Not Null	Unique	Len
PFK	sc_order	int	+	+	-
PFK	sc_product	int	+	+	-
	sc_count	int	+	-	-

Таблица 2.7 – Структура таблицы Staff

Key	Name	Type	Not Null	Unique	Len
PK	s_personnel_number	int	+	+	-
FK	s_user	int	-		-
	s_surname	nvarchar	+	-	50
	s_name	nvarchar	+	-	50
	s_patronymic	nvarchar	+	-	50
	s_post	nvarchar	+	-	50
	s_is_fired	bit	+	-	-

Таблица 2.8 – Структура таблицы PaymentTypes

Key	Name	Type	Not Null	Unique	Len
PK	pt_id	int	+	+	-
	pt_name	nvarchar	+	+	75

Таблица 2.9 – Структура таблицы Orders

Key	Name	Type	Not Null	Unique	Len
PK	o_id	int	+	+	-
FK	o_customer	int	-	-	-
FK	o_employee	int	-	-	-
FK	o_status	int	+	-	-
FK	o_payment_type	int	+	-	-
	o_total_cost	int	+	-	-
	o_order_date	datetime	+	-	-
	o_payment_date	datetime	-	-	-

Для отображения информационной модели рассматриваемого процесса используются следующие сущности:

— «Customer» – хранение информации о покупателе: id покупателя, фамилия, имя, отчество, контактный телефон;

— «Employee» – хранение информации о сотруднике магазина: табельный номер, id пользователя в системе, фамилия, имя, отчество, должность, флаг, уволен ли сотрудник;

— «Order» – хранение информации о заказе: id заказа, id покупателя, id сотрудника, принявшего заказ, id статуса заказа, id типа оплаты, общая стоимость заказа,

дата заказа, дата оплаты заказа.

— «OrderStatus» – хранение информации о статусах заказов: id статуса, наименование статуса заказа;

— «PaymentType» – хранение информации о типе оплаты: id типа оплаты, наименование типа оплаты;

— «Product» – хранение информации о товаре: артикул, id типа товара, наименование, производитель, цена, описание, количество на складе, флаг, удалён ли товар;

— «ProductType» – хранение информации о типе товара: id типа товара, наименование типа товара;

— «User» – хранение информации о пользователе системы: id пользователя, логин, пароль.

					КП.1-53 01 02.10030282	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

3 Проектирование архитектуры проекта

3.1 Разработка диаграммы взаимодействия

Для описания процессов, происходящих в клиентской части web-приложения, была разработана диаграмма взаимодействия. Она описывает все на данный момент реализованные варианты использования приложения, которые доступны пользователю. Диаграмма взаимодействия представлена в графической части.

Архитектура АСОИ представлена паттерном Model-View-Controller, где:

- модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние;
- представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели;
- контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Формами или интерфейсами взаимодействия с пользователем в данном приложении являются JSP-страницы, написанные на HTML и CSS, с использованием фреймворка Bootstrap4. Для связи приложения с базой данных используется библиотека Entity Framework Core.

В таблице 3.1 перечислены классы, реализованные в ходе разработки приложения.

Таблица 3.1 – Классы АСОИ

Имя класса	Стереотип класса
Customers	«boundary»
Employees	«boundary»
Payment types	«boundary»
Orders	«boundary»
Order details	«boundary»
Products	«boundary»
Shopping carts	«boundary»
CustomersController	«controller»
EmployeesController	«controller»
PaymentTypesController	«controller»
OrdersController	«controller»
ProductsController	«controller»
ShoppingCartsController	«controller»

3.2 Структура классов АСОИ

Диаграмма классов – это диаграмма, которая демонстрирует общую структуру классов, их атрибутов и взаимосвязей между ними.

Атрибуты класса определяют состав и структуру данных, которые хранятся в объектах этого класса. Каждый атрибут имеет имя и тип, определяющий, какие данные он представляет.

Таблица 3.2 – Интерфейс класса CustomersController

Класс CustomersController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.3 – Интерфейс класса OrdersController

Класс OrdersController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
SalesReport	Task
Details	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.4 – Интерфейс класса EmployeesController

Класс EmployeesController	
Поля	
Имя	Тип данных
_context	ApplicationContext

Продолжение таблицы 3.4

Методы	
Имя	Тип данных
Index	Task<IActionResult>
Details	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.5 – Интерфейс класса OrderStatusController

Класс OrderStatusController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.6 – Интерфейс класса PaymentTypesController

Класс PaymentTypesController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.7 – Интерфейс класса ProductsController

Класс ProductsController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Details	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.8 – Интерфейс класса ProductTypesController

Класс ProductTypesController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.9 – Интерфейс класса ShoppingCartsController

Класс ShoppingCartsController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>

Продолжение таблицы 3.9

1	2
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Таблица 3.10 – Интерфейс класса UsersController

Класс UsersController	
Поля	
Имя	Тип данных
_context	ApplicationContext
Методы	
Имя	Тип данных
Index	Task<IActionResult>
Create	IActionResult
Create	Task<IActionResult>
Edit	Task<IActionResult>
Edit	Task<IActionResult>
Delete	Task<IActionResult>
DeleteConfirmed	Task<IActionResult>

Перечисленные выше классы, которые представляют собой контроллеры, содержат всю основную бизнес логику данного приложения. Они позволяют просматривать, изменять, добавлять и удалять информацию.

3.3 Диаграмма состояний

Диаграмма состояний используется для описания взаимодействия между программой и пользователем. Диаграмма моделирует переходы между диалоговыми формами и список внутренних действий в форме. На переходах между формами отображается имя кнопки, вызвавшей событие перехода.

Диаграмма состояний похожа на диаграмму деятельности, но деятельность в случае диаграммы состояний заменена состоянием, переходы символизируют действия.

Состояние содержит имя или имя и список внутренних действий. Список внутренних действий содержит перечень действий или деятельности, которые выполняются во время нахождения объекта в данном состоянии. Данный список фиксированный.

Диаграмма состояний приведена в графической части документа.

4 Управление процессом разработки программного обеспечения

4.1 Определение трудоёмкости разработки

В таблице 4.1 указаны приблизительные трудозатраты разработки элементов АСОИ в виде количества часов, затраченного на создание того или иного элемента

Таблица 4.1 – Трудоёмкость разработки программного обеспечения

Компонент	Трудоёмкость разработки, ч
CustomersController	1
EmployeesController	1
OrdersController	2.5
OrderStatusController	1
PaymentTypesController	1
ProductsController	1
ProductTypesController	1
ShoppingCartscontroller	3
UsersController	1
Views: Customers	2.6
Views: Employees	3.2
Views: Orders	3.2
Views: OrderStatus	1.3
Views: PaymentTypes	1.4
Views: Products	3.5
Views: ProductTypes	1.5
Views: ShoppingCarts	2
Views: Users	1.5
Views: Shared	1
Views: Home	1
database.Entities	15
database.ApplicationContext	10
Итого:	58.7

Таблица 4.2 – Календарный план разработки

Компонент	Даты
database.Entities	20.09.2022 – 01.10.2022
database.ApplicationContext	05.10.2022 – 14.10.2022
CustomersController	16.10.2022
EmployeesController	18.10.2022
OrdersController	20.10.2022 – 22.10.2022
OrderStatusController	23.10.2022
PaymentTypesController	25.10.2022

Продолжение таблицы 4.2

1	2
ProductsController	26.10.2022
ProductTypesController	28.10.2022
ShoppingCartsController	30.10.2022 – 1.11.2022
UsersController	3.11.2022
Views: Shared	5.11.2022 – 6.11.2022
Views: Customers	7.11.2022– 11.11.2022
Views: Users	13.11.2022 – 16.11.2022
Views: Employees	17.11.2022 – 20.11.2022
Views: OrderStatus	22.11.2022 – 24.11.2022
Views: PaymentTypes	25.11.2022 – 27.11.2022
Views: ProductTypes	28.11.2022 – 30.11.2022
Views: Orders	1.12.2022 – 5.12.2022
Views: Products	6.12.2022 – 10.12.2022
Views: ShoppingCarts	12.12.2022 – 15.12.2022

4.2 Отчёт о разработке программных компонентов

В разработке применялся следующий стек технологий:

- платформа ASP.NET Core для кроссплатформенной разработки веб-приложений;
- язык C#;
- ORM-технология Entity Framework Core от компании Microsoft для доступа к базе данных;
- Microsoft SQL Server 2019 как СУБД;
- фреймворк Bootstrap4.

В разработке приложения использовались ORM Entity Framework Core и подход Code-First, которые позволяют непосредственно в коде C# создавать сущности базы данных, а также и контекст этой базы данных.

Фрагмент кода, в котором происходит создание контроллеров для всех сущностей базы данных и контекста базы данных, приведён ниже.

Контроллер CustomersController 16.10.2022

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MusicShop.DbContexts;
using MusicShop.Models;
```

```
namespace MusicShop.Controllers
```

```
{
```

```
    public class CustomersController : Controller
```



```

{
    private readonly ApplicationContext _context;

    public CustomersController(ApplicationContext context)
    {
        _context = context;
    }

    public async Task<IActionResult> Index()
    {
        return View(await _context.Customers.ToListAsync());
    }

    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("Id,Surname,Name,Patronymic,PhoneNumber")] Customer customer)
    {
        if (ModelState.IsValid)
        {
            _context.Add(customer);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(customer);
    }

    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Customers == null)
        {
            return NotFound();
        }

        var customer = await _context.Customers.FindAsync(id);
        if (customer == null)

```

```

    {
        return NotFound();
    }
    return View(customer);
}

```

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id,
[Bind("Id,Surname,Name,Patronymic,PhoneNumber")] Customer customer)
{
    if (id != customer.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(customer);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!CustomerExists(customer.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(customer);
}

public async Task<IActionResult> Delete(int? id)

```

```

    {
        if (id == null || _context.Customers == null)
        {
            return NotFound();
        }

        var customer = await _context.Customers
            .FirstOrDefaultAsync(m => m.Id == id);
        if (customer == null)
        {
            return NotFound();
        }

        return View(customer);
    }

    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.Customers == null)
        {
            return Problem("Entity set 'ApplicationContext.Customers' is null.");
        }
        var customer = await _context.Customers.FindAsync(id);
        if (customer != null)
        {
            _context.Customers.Remove(customer);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool CustomerExists(int id)
    {
        return _context.Customers.Any(e => e.Id == id);
    }
}

```

Контроллер EmployeesController 18.10.2022

					КП.1-53 01 02.10030282	Лист
						19
Изм.	Лист	№ докум.	Подпись	Дата		

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using MusicShop.DbContexts;
using MusicShop.Models;

namespace MusicShop.Controllers
{
    public class EmployeesController : Controller
    {
        private readonly ApplicationDbContext _context;

        public EmployeesController(ApplicationContext context)
        {
            _context = context;
        }

        public async Task<IActionResult> Index()
        {
            return View(await _context.Staff.ToListAsync());
        }

        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Staff == null)
            {
                return NotFound();
            }

            var employee = await _context.Staff
                .FirstOrDefaultAsync(m => m.Id == id);
            if (employee == null)
            {
                return NotFound();
            }

            return View(employee);
        }

        public IActionResult Create()
        {
            return View();
        }
    }
}

```

```

    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult>
Create([Bind("Id,UserId,Surname,Name,Patronymic,Post,IsFired")] Employee employee)
    {
        if (ModelState.IsValid)
        {
            _context.Add(employee);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        return View(employee);
    }

    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Staff == null)
        {
            return NotFound();
        }

        var employee = await _context.Staff.FindAsync(id);
        if (employee == null)
        {
            return NotFound();
        }
        return View(employee);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id,
[Bind("Id,UserId,Surname,Name,Patronymic,Post,IsFired")] Employee employee)
    {
        if (id != employee.Id)
        {
            return NotFound();
        }
    }

```

```

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(employee);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!EmployeeExists(employee.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        return View(employee);
    }

    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null || _context.Staff == null)
        {
            return NotFound();
        }

        var employee = await _context.Staff
            .FirstOrDefaultAsync(m => m.Id == id);
        if (employee == null)
        {
            return NotFound();
        }

        return View(employee);
    }

    [HttpPost, ActionName("Delete")]

```

```

[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id)
{
    if (_context.Staff == null)
    {
        return Problem("Entity set 'ApplicationContext.Staff' is null.");
    }
    var employee = await _context.Staff.FindAsync(id);
    if (employee != null)
    {
        _context.Staff.Remove(employee);
    }

    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

private bool EmployeeExists(int id)
{
    return _context.Staff.Any(e => e.Id == id);
}
}

```

Контроллер OrdersController 20.10.2022 – 22.10.2022

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MusicShop.DbContexts;
using MusicShop.Models;
using System.Text;

namespace MusicShop.Controllers
{
    public class OrdersController : Controller
    {
        private readonly ApplicationContext _context;

        public OrdersController(ApplicationContext context)
        {
            _context = context;
        }
    }
}

```

```

public async Task<IActionResult> Index()
{
    var applicationContext = _context.Orders.Include(o => o.Customer).Include(o
=> o.Employee).Include(o => o.PaymentType).Include(o => o.Status);
    return View(await applicationContext.ToListAsync());
}

[HttpGet]
public async Task SalesReport()
{
    var headers = new StringBuilder("Артикул    товара\tКол-
во\tЦена\tВыручка\tНаименование\n");
    var data = _context.Products
        .GroupJoin(_context.ShoppingCarts,
            p => p.Id,
            sc => sc.ProductId,
            (p, sc) => new
            {
                ProductNumber = p.Id,
                Name = p.Name,
                SalesCount = sc
                    .Where(item => item.ProductId == p.Id)
                    .Sum(s => s.Count),
                Price = p.Price,
                Gain = p.Price * sc
                    .Where(item => item.ProductId == p.Id)
                    .Sum(s => s.Count),
            });
    var salesStatistics = data.ToList();

    var report = new StringBuilder("");
    report.AppendLine(headers.ToString());

    foreach (var item in salesStatistics)
    {
        report.Append($" \t{item.ProductNumber} \t");
        report.Append($" {item.SalesCount} \t");
        report.Append($" {item.Price} \t");
        report.Append($" {item.Gain} \t");
        report.Append($" {item.Name} \t");
    }
}

```



```

        report.AppendLine();
    }

    await Response.WriteAsync(report.ToString(), Encoding.Unicode);
}

public async Task<IActionResult> Details(int? id)
{
    if (id == null || _context.Orders == null)
    {
        return NotFound();
    }

    var order = await _context.Orders
        .Include(o => o.Customer)
        .Include(o => o.Employee)
        .Include(o => o.PaymentType)
        .Include(o => o.Status)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (order == null)
    {
        return NotFound();
    }

    return View(order);
}

public IActionResult Create()
{
    ViewData["CustomerId"] = new SelectList(_context.Customers, "Id",
"Surname");
    ViewData["EmployeeId"] = new SelectList(_context.Staff, "Id", "Surname");
    ViewData["PaymentTypeId"] = new SelectList(_context.PaymentTypes, "Id",
"Name");
    ViewData["StatusId"] = new SelectList(_context.OrderStatuses, "Id", "Name");
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]

```

```

        public async Task<IActionResult>
Create([Bind("Id,CustomerId,EmployeeId,StatusId,PaymentTypeId,TotalCost,OrderDate,Pay
mentDate")] Order order)
    {
        if (ModelState.IsValid)
        {
            _context.Add(order);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }
        ViewData["CustomerId"] = new SelectList(_context.Customers, "Id",
"Surname", order.CustomerId);
        ViewData["EmployeeId"] = new SelectList(_context.Staff, "Id", "Surname",
order.EmployeeId);
        ViewData["PaymentTypeId"] = new SelectList(_context.PaymentTypes, "Id",
"Name", order.PaymentTypeId);
        ViewData["StatusId"] = new SelectList(_context.OrderStatuses, "Id", "Name",
order.StatusId);
        return View(order);
    }

    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Orders == null)
        {
            return NotFound();
        }

        var order = await _context.Orders.FindAsync(id);
        if (order == null)
        {
            return NotFound();
        }
        ViewData["CustomerId"] = new SelectList(_context.Customers, "Id",
"Surname", order.CustomerId);
        ViewData["EmployeeId"] = new SelectList(_context.Staff, "Id", "Surname",
order.EmployeeId);
        ViewData["PaymentTypeId"] = new SelectList(_context.PaymentTypes, "Id",
"Name", order.PaymentTypeId);
        ViewData["StatusId"] = new SelectList(_context.OrderStatuses, "Id", "Name",
order.StatusId);
    }

```

```

        return View(order);
    }

    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id,
[Bind("Id,CustomerId,EmployeeId,StatusId,PaymentTypeId,TotalCost,OrderDate,PaymentDate")] Order order)
    {
        if (id != order.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(order);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!OrderExists(order.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["CustomerId"] = new SelectList(_context.Customers, "Id",
"Surname", order.CustomerId);
        ViewData["EmployeeId"] = new SelectList(_context.Staff, "Id", "Surname",
order.EmployeeId);
        ViewData["PaymentTypeId"] = new SelectList(_context.PaymentTypes, "Id",
"Name", order.PaymentTypeId);
    }

```

```

        ViewData["StatusId"] = new SelectList(_context.OrderStatuses, "Id", "Name",
order.StatusId);
        return View(order);
    }

```

```

public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Orders == null)
    {
        return NotFound();
    }

```

```

    var order = await _context.Orders
        .Include(o => o.Customer)
        .Include(o => o.Employee)
        .Include(o => o.PaymentType)
        .Include(o => o.Status)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (order == null)
    {
        return NotFound();
    }

```

```

    return View(order);
}

```

```

[HttpPost, ActionName("Delete")]

```

```

[ValidateAntiForgeryToken]

```

```

public async Task<IActionResult> DeleteConfirmed(int id)

```

```

{
    if (_context.Orders == null)
    {
        return Problem("Entity set 'ApplicationContext.Orders' is null.");
    }
    var order = await _context.Orders.FindAsync(id);
    if (order != null)
    {
        _context.Orders.Remove(order);
    }

```

```

    await _context.SaveChangesAsync();

```

```

        return RedirectToAction(nameof(Index));
    }

    private bool OrderExists(int id)
    {
        return _context.Orders.Any(e => e.Id == id);
    }
}

```

Контроллер ShoppingCartsController 30.10.2022 – 1.11.2022

```

using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using MusicShop.DbContexts;
using MusicShop.Models;

```

```

namespace MusicShop.Controllers

```

```

{
    public class ShoppingCartsController : Controller
    {
        private readonly ApplicationDbContext _context;

```

```

        public ShoppingCartsController(ApplicationDbContext context)
        {
            _context = context;
        }

```

```

        public async Task<IActionResult> Index()
        {
            var applicationContext = _context.ShoppingCarts.Include(s
s.Order).Include(s => s.Product);
            return View(await applicationContext.ToListAsync());
        }

```

```

        public IActionResult Create()
        {
            ViewData["OrderId"] = new SelectList(_context.Orders, "Id", "Id");
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Name");
            return View();
        }

```

```

[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create([Bind("OrderId,ProductId,Count")]
ShoppingCart shoppingCart)
{
    if (ModelState.IsValid)
    {
        _context.Add(shoppingCart);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["OrderId"] = new SelectList(_context.Orders, "Id", "Id",
shoppingCart.OrderId);
    ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Name",
shoppingCart.ProductId);
    return View(shoppingCart);
}

public async Task<IActionResult> Edit(int? orderid, int? productid)
{
    if (orderid == null || productid == null || _context.ShoppingCarts == null)
    {
        return NotFound();
    }

    var shoppingCart = await _context.ShoppingCarts.FindAsync(orderid,
productid);
    if (shoppingCart == null)
    {
        return NotFound();
    }
    ViewData["OrderId"] = new SelectList(_context.Orders, "Id", "Id",
shoppingCart.OrderId);
    ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Name",
shoppingCart.ProductId);
    return View(shoppingCart);
}

[HttpPost]
[ValidateAntiForgeryToken]

```

```

        public async Task<IActionResult> Edit(int orderid, int productid,
[Bind("OrderId,ProductId,Count")] ShoppingCart shoppingCart)
        {
            if (orderid != shoppingCart.OrderId && productid != shoppingCart.ProductId)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(shoppingCart);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ShoppingCartExists(shoppingCart.OrderId, shoppingCart.ProductId))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["OrderId"] = new SelectList(_context.Orders, "Id", "Id",
shoppingCart.OrderId);
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Name",
shoppingCart.ProductId);
            return View(shoppingCart);
        }

        public async Task<IActionResult> Delete(int? orderid, int? productid)
        {
            if (orderid == null || productid == null || _context.ShoppingCarts == null)
            {
                return NotFound();
            }
        }

```

```

        var shoppingCart = await _context.ShoppingCarts
            .Include(s => s.Order)
            .Include(s => s.Product)
            .FirstOrDefaultAsync(m => m.OrderId == orderid && m.ProductId ==
productid);
        if (shoppingCart == null)
        {
            return NotFound();
        }

        return View(shoppingCart);
    }

    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int orderid, int productid)
    {
        if (_context.ShoppingCarts == null)
        {
            return Problem("Entity set 'ApplicationContext.ShoppingCarts' is null.");
        }
        var shoppingCart = await _context.ShoppingCarts.FindAsync(orderid,
productid);
        if (shoppingCart != null)
        {
            _context.ShoppingCarts.Remove(shoppingCart);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool ShoppingCartExists(int orderid, int productid)
    {
        return _context.ShoppingCarts.Any(e => e.OrderId == orderid && e.ProductId
== productid);
    }
}

```


5 Тестирование разработанного ПО

Для нормальной работы системы необходимо 512 Мбайт оперативной памяти, Windows 10, MS SQLServer 2019.

Результатом тестирования в данном случае может послужить безошибочное оформление заказа, добавление товара в корзину, печати чека заказа и получения отчёта продаж товаров.

После тестирования получили результаты:

- информационная система поддерживает многопользовательский режим;
- выполнение запросов прошло успешно.

В курсовом проекте разработана база данных с клиентскими приложениями, которая хранится на SQL сервере. Составлены запросы и формы. Формы составлены на основе запросов и таблиц, и используются для занесения и модификации информации в базе данных.

После открытия приложения пользователь находится на главной странице, представленной на рисунке 5.1:

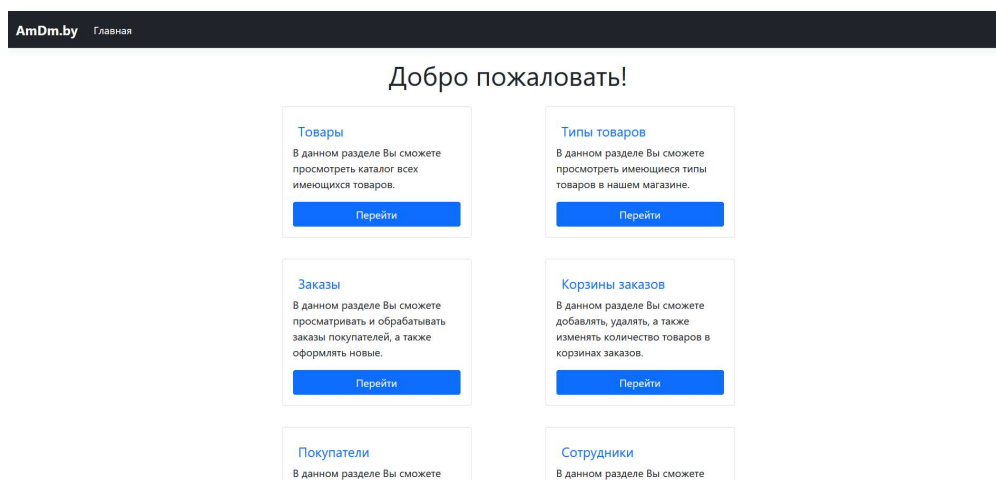


Рисунок 5.1 – Главная страница

Карта сайта представлена в виде карточек с заголовком страницы, описанием списка действий пользователя и кнопками «Перейти» для перехода к требуемой странице. Карточка представлена на рисунке 5.2.

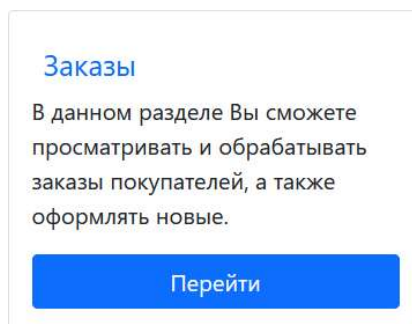


Рисунок 5.2 – Карточка с информацией о разделе «Заказы»

На странице «Сведения о заказах» (рисунок 5.3) отображается следующая информация о заказах: номер заказа, код клиента, статус и стоимость заказа.

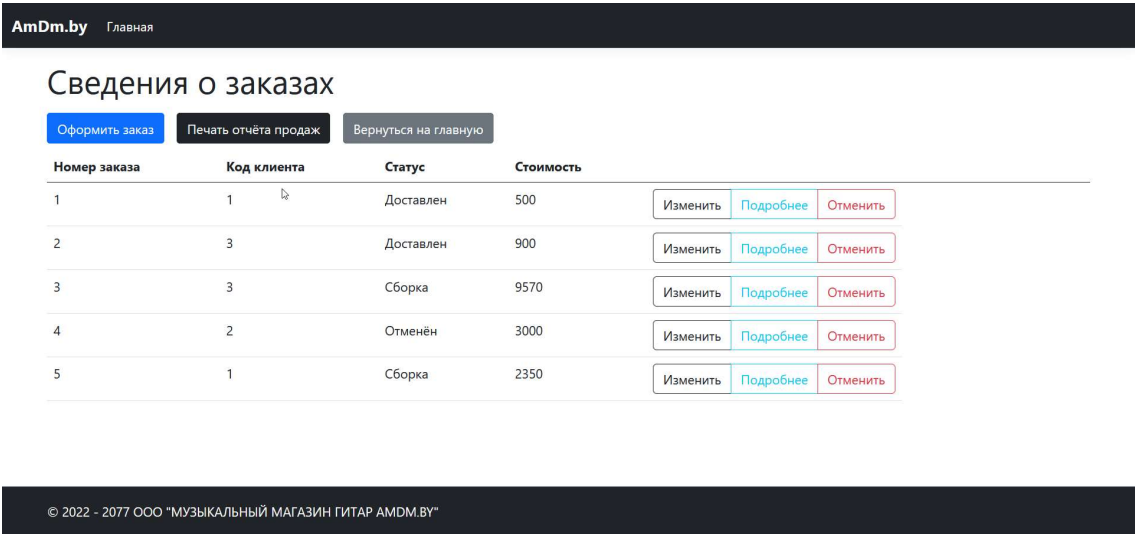


Рисунок 5.3 – Страница «Сведения о заказах»

Кроме того, на странице присутствуют кнопки для оформления заказа («Оформить заказ»), вывод в файл отчёта по продажам («Печать отчёта продаж»), а также перехода на главную страницу («Вернуться на главную»).

Напротив каждой записи присутствует группа кнопок, определяющих действия над ней: изменение («Изменить»), просмотр подробной информации («Подробнее») и отмена заказа («Отменить»).

После нажатия на кнопку «Печать отчёта продаж» на локальный компьютер будет скачан файл SalesReport, в котором будет отчёт по продажам товаров (рисунок 5.4):

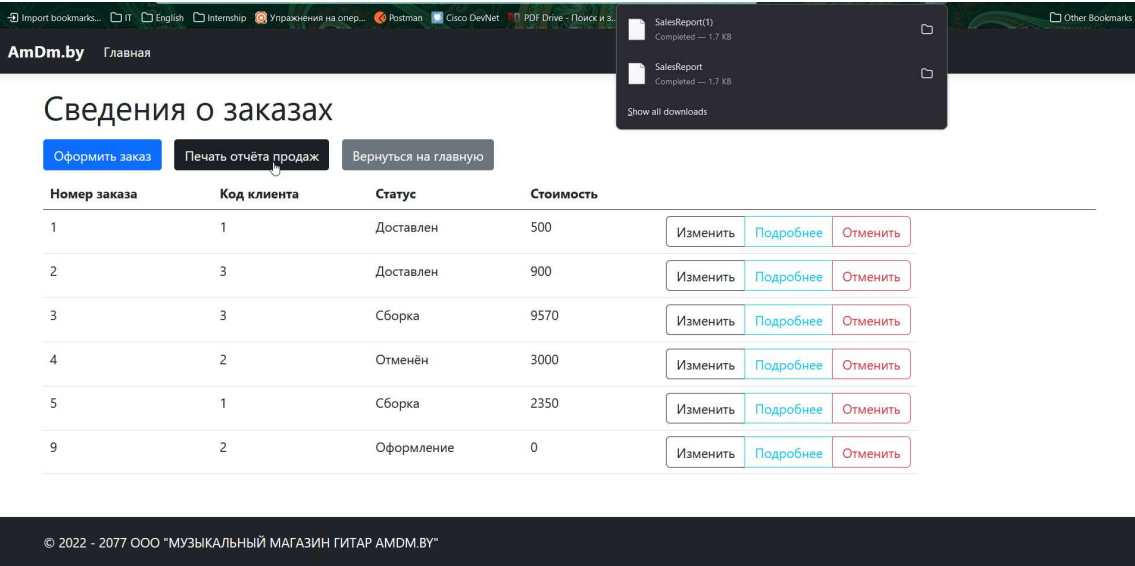


Рисунок 5.4 – Скачивание отчёта

Откроем загрузившийся файл (рисунок 5.5)

SalesReport.txt – Блокнот

Файл	Правка	Формат	Вид	Справка	
Артикул	товара	Кол-во	Цена	Выручка	Наименование
1	8	358	2864		Акустическая гитара Cort AD810 OP
2	0	988	0		Акустическая гитара Sigma Guitars 000MTRUEST+
3	0	832	0		Электрогитара Epiphone LES PAUL SPECIAL VE Ebony Vintage A064659
4	0	1326	0		Электрогитара Ibanez IRON RG421-MOL Mahogany 011
5	0	2	0		Медиатор Planet Waves 70BUS
6	0	1	0		Медиатор Dunlop 417R.71
7	1	32	32		Каподастр Planet Waves PW-CPTRUE6
8	2	26	52		Струны для акустической гитары Ernie Ball P02144
9	0	50	0		Струны для электрогитары D'Addario EC625
10	1	57	57		Чехол для акустической гитары ЧПУП ВРАЛАН ГЧ265 Классика
11	0	240	0		Кейс для акустической гитары Crossrock CRW50058BK
12	0	1157	0		Комбоусилитель гитарный Marshall Code 50
13	1	4680	4680		Ламповый гитарный комбоусилитель Fender Hot Rod Deluxe IV Black
14	0	300	0		Cort Ad810M

Рисунок 5.5 – Содержимое файла SalesReport

После нажатия на кнопку «Оформить заказ» открывается форма «Оформление заказа», представленная на рисунке 5.6:

AmDm.by Главная

Оформление заказа

Покупатель

Видюхин

Сотрудник

Савельев

Тип оплаты

Предоплата

Дата оформления

12 / 15 / 2022, 10:14 AM

Оформить

Очистить

Вернуться к заказам

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР AMDM.BY"

Рисунок 5.6 – Форма «Оформление заказа»

Данная форма имеет четыре поля для ввода информации. Первое поле «Покупатель» необходимо для выбора покупателя из выпадающего списка. Второе («Сотрудник») и третье поле («Тип оплаты») также представляют собой выпадающие списки для выбора оформляющего сотрудника и способа оплаты заказа соответственно. Четвёртое поле «Дата оформления» необходимо для выбора даты оформления заказа.

Кроме того, на форме имеются три кнопки: «Оформить» – для подтверждения оформления, «Очистить» – для очистки полей формы и «Вернуться к заказам» для возвращения на страницу «Сведения о заказах».

После нажатия кнопки «Оформить» сведения о заказе будут добавлены в базу данных, и пользователь сможет наблюдать их на странице «Сведения о заказах» (рисунок 5.7).

9	2	Оформление	0	Изменить	Подробнее	Отменить
---	---	------------	---	----------	-----------	----------

Рисунок 5.7 – Запись оформленного заказа

При нажатии кнопки «Изменить» открывается форма для изменения сведений о заказе (рисунок 5.8):

AmDm.by Главная

Сведения о заказе

Клиент
Видюхин

Сотрудник
Савельев

Статус заказа
Оформление

Тип оплаты
Предоплата

Стоимость заказа
832

Дата оформления
12 / 15 / 2022 , 10 : 14 AM

Дата оплаты
mm / dd / yyyy

Сохранить

Вернуться к заказам

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР АМДМ BY"

Рисунок 5.8 – Форма «Сведения о заказе»

Далее добавим в корзину заказа товар. Для этого необходимо нажать на кнопку «Вернуться на главную» и далее выбрать нажать на кнопку «Перейти» карточки «Корзины заказов» (рисунок 5.9):

AmDm.by Главная

Добро пожаловать!

Товары

В данном разделе Вы сможете просмотреть каталог всех имеющихся товаров.

Перейти

Типы товаров

В данном разделе Вы сможете просмотреть имеющиеся типы товаров в нашем магазине.

Перейти

Заказы

В данном разделе Вы сможете просматривать и обрабатывать заказы покупателей, а также оформлять новые.

Перейти

Корзины заказов

В данном разделе Вы сможете добавлять, удалять, а также изменять количество товаров в корзинах заказов.

Перейти

<https://localhost:7089/ShoppingCarts>

Рисунок 5.9 – Карточка «Корзины заказов»

На странице «Корзины заказов» (рисунок 5.10) отображается следующая информация о заказах: номер заказа, наименование товара и его количество.

AmDm.by Главная

Корзины заказов

Добавить товар в корзину Вернуться на главную

Номер заказа	Наименование товара	Кол-во	Изменить	Удалить
1	Акустическая гитара Cort AD810 OP	3	Изменить	Удалить
1	Струны для акустической гитары Ernie Ball P02144	2	Изменить	Удалить
2	Акустическая гитара Cort AD810 OP	4	Изменить	Удалить
2	Чехол для акустической гитары ЧПУП ЮРАЛИАН ГЧ265 Классика	1	Изменить	Удалить
3	Каподастр Planet Waves PW-CPTRUE6	1	Изменить	Удалить
4	Акустическая гитара Cort AD810 OP	1	Изменить	Удалить
5	Ламповый гитарный комбоусилитель Fender Hot Rod Deluxe IV Black	1	Изменить	Удалить

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР AMDM.BY"
https://localhost:7089/ShoppingCarts/Create

Рисунок 5.10 – Страница «Корзины заказов»

Здесь присутствуют кнопки для добавления товара в корзину заказа («Добавить товар в корзину»), перехода на главную страницу («Вернуться на главную»), а также список действий (кнопки «Изменить», «Удалить») напротив каждой заявки о товаре.

При нажатии на кнопку «Добавить товар в корзину» открывается форма «Добавление товара в корзину» (рисунок 5.11).

AmDm.by Главная

Добавление товара в корзину

Номер заказа

9

Наименование товара

Электрогитара Epiphone LES PAUL SPECIAL VE Ebony Vintage A064659

Кол-во товара

2

Добавить Очистить

Вернуться к каталогу

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР AMDM.BY"

Рисунок 5.11 – Форма «Добавления товара в корзину»

Далее необходимо выбрать из выпадающего списка заказ (поле «Номер заказа»), необходимый товар (поле «Наименование товара») и указать его количество (поле «Кол-во товара»). При нажатии кнопки «Добавить» товар будет добавлен в корзину заказа.

При нажатии на кнопку «Изменить» открывается форма «Заявка на товар» для изменения количества товара, представленная на рисунке 5.12:

AmDm.by Главная

Заявка на товар

Номер заказа
9

Наименование
3

Кол-во
1

Сохранить

Вернуться к каталогу

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР AMDM.BY"

Рисунок 5.12 – Форма «Заявка на товар»

При нажатии на кнопку «Сохранить» изменения в заявке на товар будут применены в базе.

Для удаления товара из корзины на странице «Корзины заказов» нужно нажать напротив соответствующей записи кнопку «Удалить». После нажатия открывается форма «Удаление товара из корзины» (рисунок 5.13).

AmDm.by Главная

Вы уверены, что хотите удалить этот товар из корзины?

Номер заказа
9

Наименование товара
Электрогитара Epiphone LES PAUL SPECIAL
VE Ebony Vintage A064659

Кол-во
1

Удалить

Вернуться к корзинам

© 2022 - 2077 ООО "МУЗЫКАЛЬНЫЙ МАГАЗИН ГИТАР AMDM.BY"

Рисунок 5.13 – Форма удаления товара из корзины

После добавления товара в корзину необходимо получить чек заказа. Для этого требуется перейти на страницу «Сведения о заказах» (рисунок 5.3) и нажать кнопку «Подробнее» для открытия страницы с деталями заказа, представленной на рисунке 5.14:

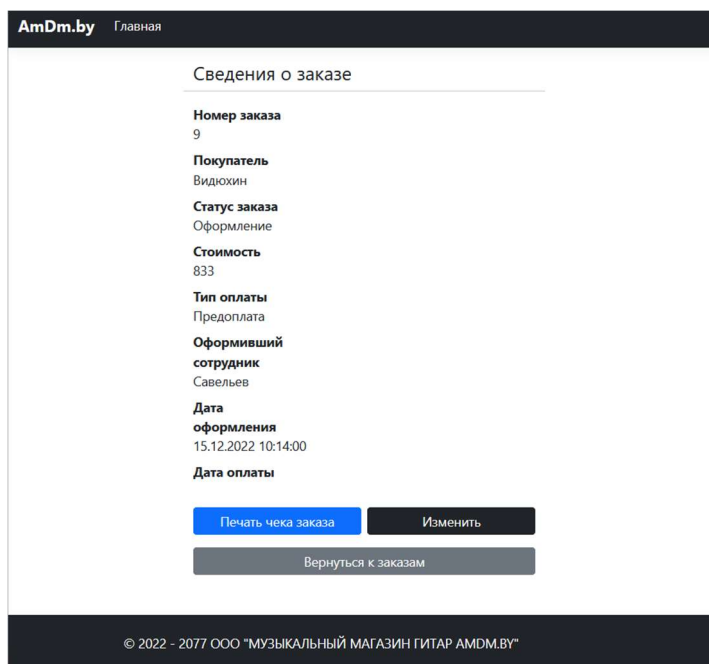


Рисунок 5.14 – Страница «Сведения о заказе»

На этой странице присутствуют три кнопки: «Печать чека заказа», «Изменить» – для изменения информации о заказе и «Вернуться к заказам». При нажатии на кнопку «Печать чека заказа» инициируется вывод информации о заказе в PDF-файл (рисунок 5.15):

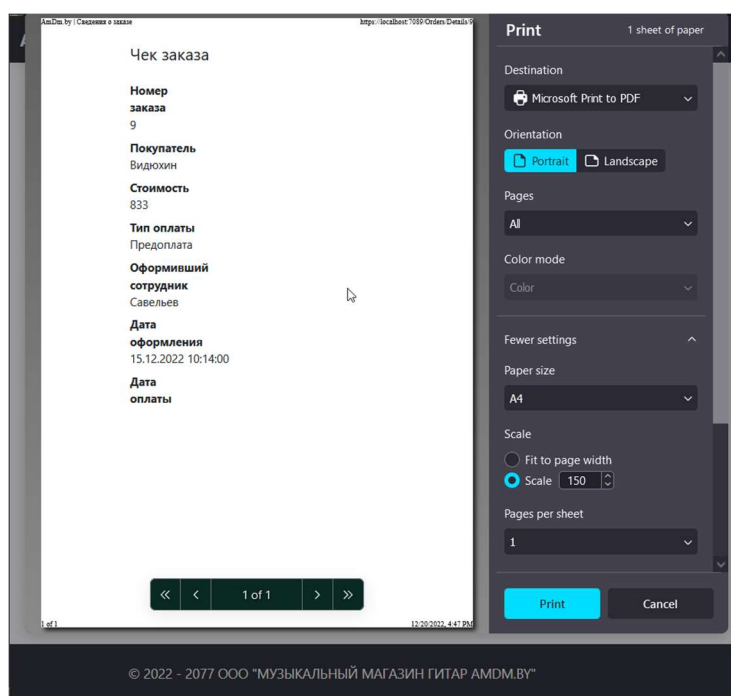
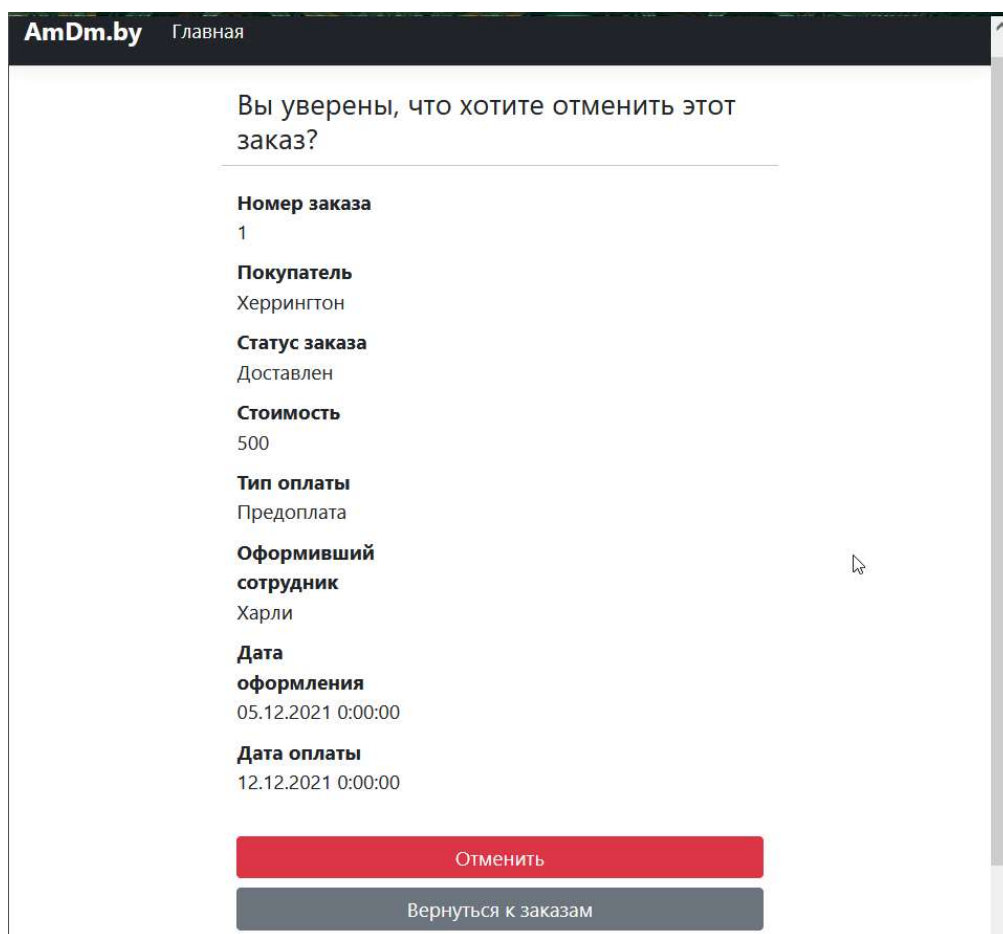


Рисунок 5.15 – Печать чека заказа

Для отмены заказа необходимо вернуться на страницу «Сведения о заказах» и нажать на кнопку «Отменить» напротив необходимого заказа (рисунок 5.16):



AmDm.by Главная

Вы уверены, что хотите отменить этот заказ?

Номер заказа
1

Покупатель
Херрингтон

Статус заказа
Доставлен

Стоимость
500

Тип оплаты
Предоплата

Оформивший сотрудник
Харли

Дата оформления
05.12.2021 0:00:00

Дата оплаты
12.12.2021 0:00:00

Отменить

Вернуться к заказам

Рисунок 5.16 – Форма «Отмена заказа»

При нажатии на кнопку «Отменить» заказ будет отменён.

Заключение

Автоматизация и информатизация необходима для интернет-магазинов, так как она позволяет обрабатывать заказы в реальном времени, что в конечном счёте приводит к оптимизации торгового процесса.

Во время выполнения курсового проектирования была разработана многопользовательская информационная система магазина по продаже гитар и аксессуаров к ним.

При создании АСОИ использовался следующий стек технологий:

- язык C# и платформа ASP.NET;
- ORM-технология Entity Framework Core 7;
- фреймворк Bootstrap4;
- CASE-средство Sparx System Enterprise Architect 15.2 Build 1560 05-Nov-2021;
- СУБД Microsoft SQL Server 2019;
- HTML5/CSS3.

Были разработаны такие элементы проектирования систем, как: диаграмма вариантов использования, диаграмма бизнес-процессов, диаграмма классов АСОИ, диаграмма классов БД, диаграмма последовательности, диаграмма состояний.

Проведено тестирование реализованных систем. Проведенное тестирование показало целостность и правильность составленного кода взаимодействия с созданной базой данных.

Все пункты технического задания курсового проектирования были выполнены. Таким образом, система выполняет поставленную перед ней задачу.

					КП.1-53 01 02.10030282	Лист
						41
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Буч, Г. Язык UML. Руководство пользователя: пер. с англ. / Г. Буч, Д. Рамбо, И. Якобсон; – 2-е изд.–М.: ДМК Пресс, 2007. – 496с.
2. Куликов, С. С. Реляционные базы данных в примерах : практическое пособие для программистов и тестировщиков / С. С. Куликов. Минск: Четыре четверти, 2020. — 424 с.
3. Чамберс Джеймс, Пэкетт Дэвид, Тиммс Саймон ASP.NET Core. Разработка приложений. — СПб.: Питер, 2018. — 464 с.: ил. — (Серия «Для профессионалов»).
4. Сильвио Морето Bootstrap в примерах. / Пер. с англ. Рагимов Р.Н. / Науч.ред. Киселев А.Н. – М.: ДМК Пресс, 2017 – 314 с.: ил.
5. Смит Дж. П. Entity Framework Core в действии: пер. с англ. / Д.А. Беликова. – М.: ДМК Пресс, 2022. – 690 с.