



Санкт-Петербургский горный университет
Кафедра информационных систем и вычислительной техники

Допускается к защите в ГЭК
Заведующий кафедрой ИСиВТ

доцент  Мазаков Е.Б.

“13” июня 2019 г.

ВЫПУСКНАЯ РАБОТА



(выпускная квалификационная работа бакалавра)

на тему: «Разработка web-приложения для обслуживания
клиентов в сети ресторанов быстрого питания»

Направление	<u>09.03.01</u> (шифр)	—	<u>Информатика и вычислительная техника</u> (наименование направления)
Автор: студент гр.	<u>ИАС-15</u> (шифр)	 (подпись)	/ <u>Ефимов Г.Л.</u> / (Ф.И.О.)
Руководитель:	<u>доцент</u> (должность)	 (подпись)	/ <u>Мазаков Е.Б.</u> / (Ф.И.О.)
Рецензент:	<u>доцент</u> (должность)	 (подпись)	/ <u>Кривцов А.Н.</u> / (Ф.И.О.)

КОНСУЛЬТАНТЫ

1. Кафедра иностранных
языков :

 (должность)	 (подпись)	<u>Гераимов А.С.</u> (Ф.И.О.)
--	---	----------------------------------

2. Кафедра ЭУиФ

 (должность)	 (подпись)	<u>Михайлов М.А.</u> (Ф.И.О.)
--	---	----------------------------------

Санкт-Петербург
2019 год

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования

Санкт-Петербургский горный университет

Кафедра информационных систем и вычислительной техники

Утверждаю

Заведующий кафедрой ИСиВТ

Доцент

Е.Б. Мазаков

« 17 » 12 2018г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

студенту группы

ИАС-15

(шифр группы)

Ефимов Г.Л.

(Ф.И.О.)

**Тема: «Разработка web-приложения для обслуживания
клиентов в сети ресторанов быстрого питания»**

Исходные данные: 1 Процесс обслуживания клиентов в сети ресторанов
быстрого питания; наборы тестовых данных;

2 Требования к организации хранения данных в web-приложении.

Тема специальной части: Разработка базы данных и web-приложения

*Требования к графической части проекта и пояснительной записке
содержатся в Методических указаниях по дипломному проектированию.*

Задание выдал (руководитель работы) доцент / Мазаков Е.Б. /
(должность) (подпись) (Ф.И.О.)

Задание принял к исполнению студент Ефимов Г.Л. /
(подпись) (Ф.И.О.)

Дата получения задания


« 8 » декабря 2018 г.



СПРАВКА

о результатах проверки текстового документа на наличие заимствований

Проверка выполнена в системе
Антиплагиат.ВУЗ

Автор работы	Ефимов Григорий Львович
Подразделение	Кафедра ИС и ВТ
Тип работы	Выпускная квалификационная работа
Название работы	Разработка web-приложения для обслуживания клиентов в сети ресторанов быстрого питания
Название файла	диплом антиплагиат.docx
Процент заимствования	4,83%
Процент цитирования	2,34%
Процент оригинальности	92,83%
Дата проверки	13:55:19 11 июня 2019г.
Модули поиска	Модуль поиска ИПС "Адилет"; Модуль выделения библиографических записей; Сводная коллекция ЭБС; Коллекция РГБ; Цитирование; Модуль поиска переводных заимствований; Коллекция eLIBRARY.RU; Коллекция ГАРАНТ; Модуль поиска Интернет; Коллекция Медицина; Модуль поиска перефразирований eLIBRARY.RU; Модуль поиска перефразирований Интернет; Коллекция Патенты; Модуль поиска "СПГУ"; Модуль поиска общеупотребительных выражений; Кольцо вузов
Работу проверил	Спиридонов Виктор Валентинович ФИО проверяющего
Дата подписи	11.06.2019 



АННОТАЦИЯ

Данная выпускная квалификационная работа посвящена разработке WEB-приложения для обслуживания клиентов сети ресторанов быстрого питания.

Первая глава работы посвящена анализу предприятия и процессов обслуживания клиентов ресторанов быстрого питания, исследованию текущей проблемы и недостатков существующей системы, предложению ее решения, а также выбору средств для реализации проекта.

Вторая глава посвящена разработке приложения, написанного на языке C# с использованием технологии ASP.NET архитектуры MVC(Model-View-Controller), созданию базы данных и структуры клиентской части Web-приложения.

Третья глава затрагивает экономический аспект работы. В ней рассчитаны затраты до реализации данной автоматизированной системы и после, также посчитан конечный результат и сделан вывод.

Выпускная квалификационная работа содержит пояснительную записку объемом 41 страниц, включая 36 рисунков, 1 таблицу, 1 приложения и список использованной литературы.

ABSTRACT

This final qualifying work is devoted to the development of a WEB application for serving customers of a fast food restaurant chain.

The first chapter of the work is devoted to the analysis of the enterprise and customer service processes of fast food restaurants, the study of the current problem and shortcomings of the existing system, suggestion of its solution, as well as the choice of funds for the project.

The second chapter is devoted to the development of an application written in C # using ASP.NET technology of the MVC architecture (Model-View-Controller), creating a database and the structure of the client part of a Web application.

The third chapter deals with the economic aspect of work. It calculates the costs before the implementation of this automated system and after, also the final result is counted and concluded.

The final qualifying paper contains an explanatory note with a volume of 41 pages, including 36 figures, 1 tables, 1 annexes and a list of references.

Оглавление

1	Анализ особенностей процессов обслуживания клиентов ресторанов быстрого питания	7
1.1	Характеристика предприятия	7
1.2	Анализ бизнес процесса	7
1.3	Постановка задачи.....	13
1.3.1	Постановка задачи и определение целей.....	13
1.3.2	Характеристика организации решения задачи.....	14
1.4	Выбор проектных решений.....	15
1.4.1	Выбор платформы будущего проекта.....	15
1.4.2	Выбор программных средств и языка программирования	16
2	Разработка web-приложения для обслуживания клиентов в сети ресторанов быстрого питания.....	17
2.1	База данных.....	20
2.1.1	Описание модели проекта	20
2.1.2	Описание таблиц	22
2.2	Серверная часть web-приложения.....	24
2.2.1	Описание контроллера для загрузки списка товаров.....	25
2.2.2	Описание контроллеров корзины	28
2.3	Клиентская часть web-приложения.....	32
2.3.1	Дополнительные средства разработки.....	33
2.3.2	Структура мастер-страницы. Работа с представлениями	34
3	Расчет оценки эффективности разработанного проектного решения	40
	Заключение	43
	Список использованной литературы.....	44
	Приложение	49

1 Анализ особенностей процессов обслуживания клиентов ресторанов быстрого питания

1.1 Характеристика предприятия

«Теремок» - это сеть ресторанов русской кухни, работающих по концепции «Фаст-Кэжуал». Заведения данной сети есть В Москве и московской области, Тюмени, Краснодаре, 2 ресторана в 2016 году открылись в Нью-Йорке, а по Санкт-Петербургу имеется 135 «Теремков».

Сеть «Теремок» имеет три формата заведений: рестораны в торговых центрах в ресторанных двориках, рестораны и кафе с собственным посадочным залом и уличные киоски.

1.2 Анализ бизнес процесса

На рисунке 1.1 представлена общая диаграмма организации предприятия, которая дает основное представление о примерной взаимосвязи отделов компании.

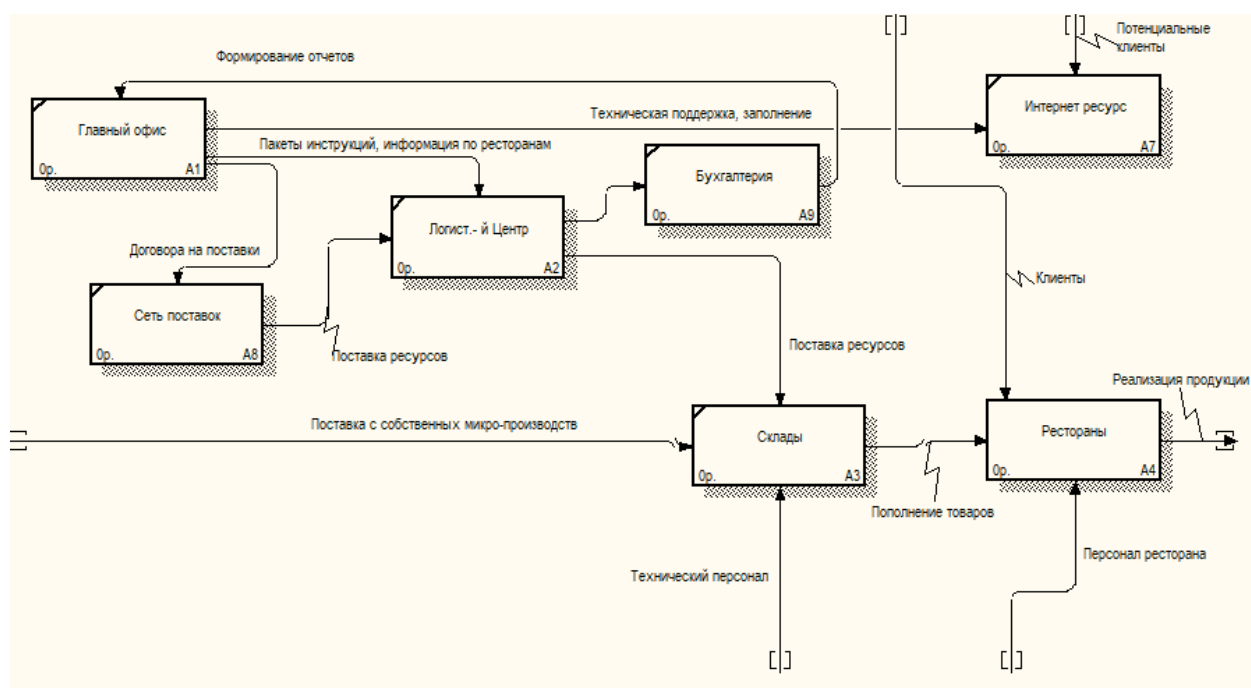


Рисунок 1.1 - Организация предприятия

1) Блок 1 характеризует главные офисы предприятия. Они занимаются основной аналитической работой и контролем всей компании, осуществляют поиск различных экономических решений и созданием сети с компаниями-поставщиками. Формируют и передают информацию об акциях, новинках и новых точках продаж в логистические центры. Контролируют работу дочерних компаний и нижестоящих отделов.

Ит-отделы главного офиса следят за работой интернет ресурсов. Обеспечивают их технической поддержкой, заполнением своевременным обновлением.

2) Блок 2 представлен логистическими центрами. Они контролируют пополнением складских помещений и складов ресторанов необходимыми ресурсами, контролируют логистику грузовых автомобилей. Обеспечивают постоянную работу между поставщиками, и собственными производствами.

3) Блок 3 представляет собой складские помещения. Они хранят все необходимые ресурсы и оборудование, для обеспечения ресторанов всем необходимым.

4) Блок 4 объединяет понятие всего множества существующих ресторанов. Они занимаются готовкой и реализацией готовой продукции конечному покупателю.

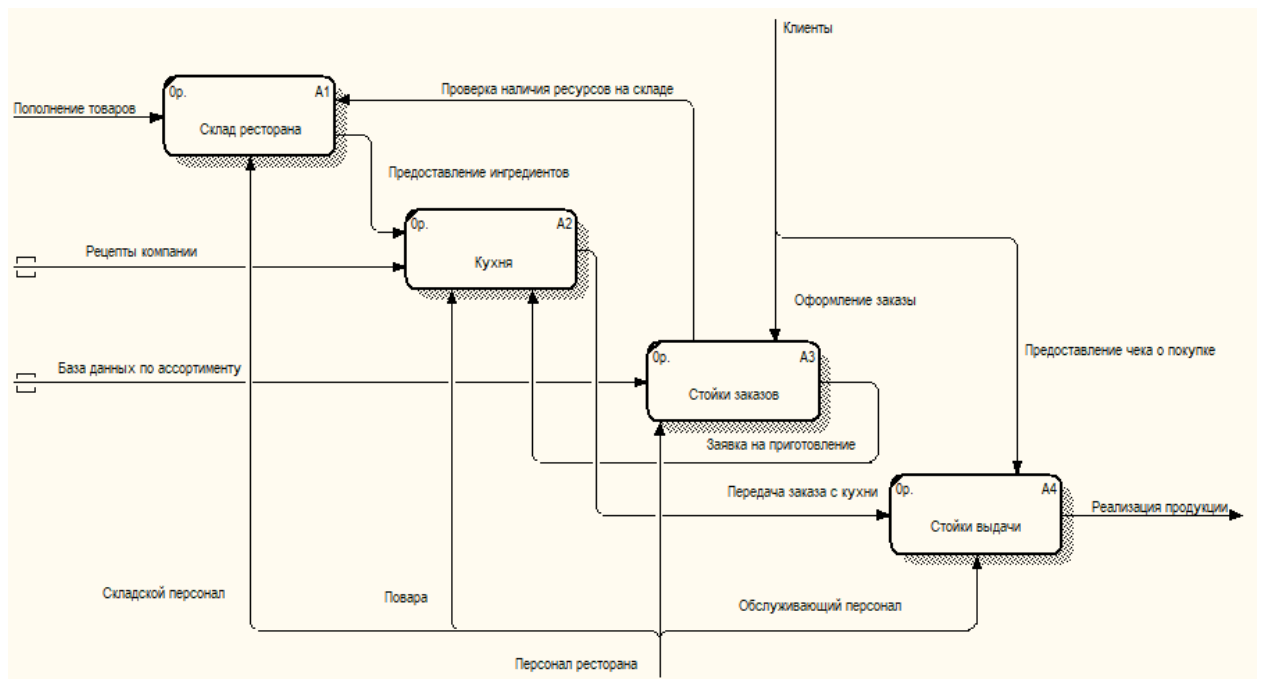


Рисунок 1.2 - Организация процесса заказа в ресторане

На рисунке 1.2 подробно рассмотрена структура работы ресторана.

1) Блок 1 характеризует склад предприятия, на котором хранятся продукты для кухни и другие ресурсы.

2) Блок 2 обозначен кухней. Входящие стрелки обозначены рецептами компаниями, по которым происходит весь процесс готовки. Заявки кухня получает от стойки заказов, а пополнение ресурсов осуществляется со склада. Исходящая стрелка обозначает движение готового заказа на стойку выдачи.

3) Блок 3 представляет собой кассовые стойки, где клиенты осуществляют оформление и оплату своих заявок. На каждой стойке имеется персональный компьютер с подключенной базой данных производства. Она предоставляет информацию по ассортименту, а также их наличию на складе. Позволяет осуществить проводку заказа и его расчет.

4) Блок 4 характеризует стойку выдачи, куда с кухни поступают подносы с готовыми заказами. Обслуживающий персонал сверяет номер чека с номером заказа, и передает его покупателю.

На рисунке 1.3 представлена модель организации стойки заказов.

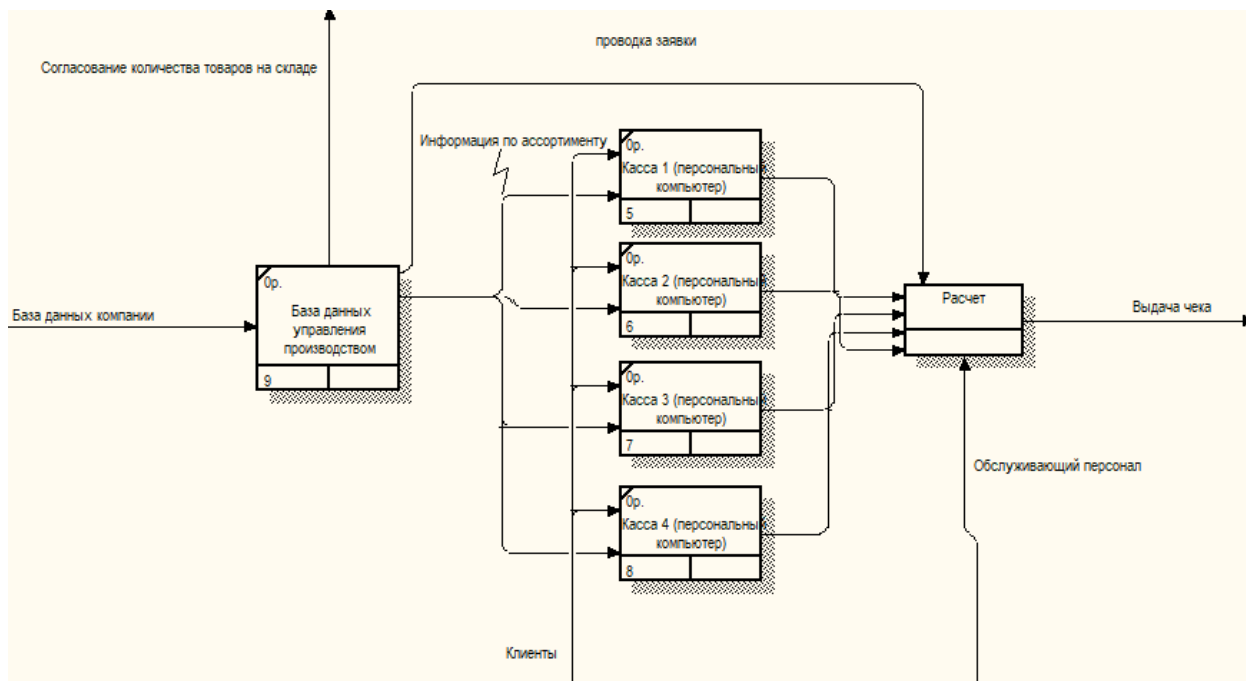


Рисунок 1.3 - Организация стойки заказов

Кассовые аппараты представлены по большей части персональными компьютерами с подключенной базой данных производства. БД предоставляет обслуживающему персоналу информацию по ассортименту, ценам и акциям, которые также проецируются на информационные табло. Клиент, обозначив заказ, рассчитывается и получает чек с номером своего заказа. После чего дожидается его у стоек выдачи.

Данная сеть уже имеет информационный интернет-ресурс, доступный всем пользователем.

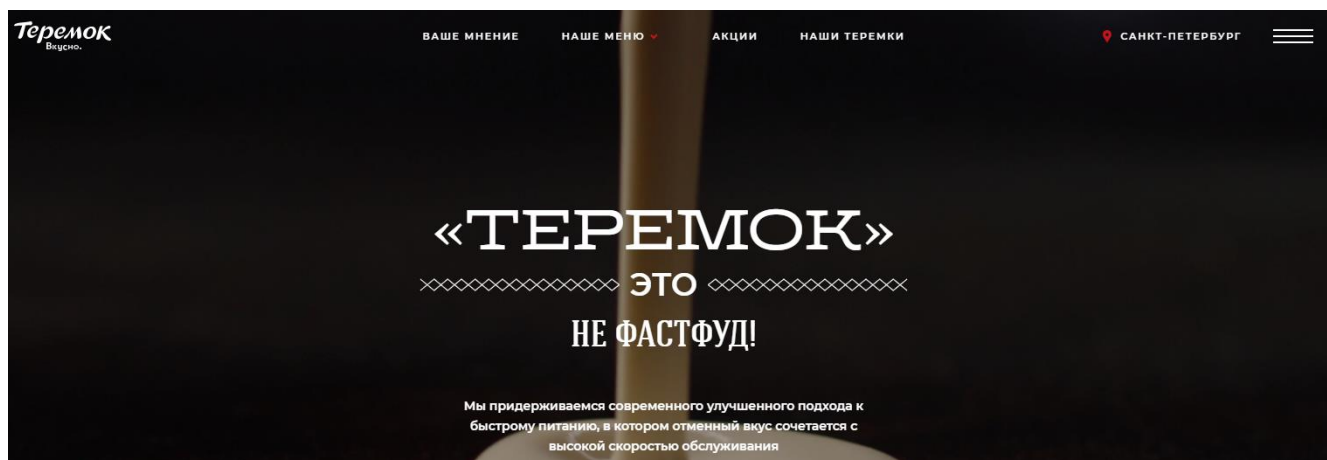


Рисунок 1.4 - Официальный сайт компании

Этот ресурс позволяет в первую очередь получить:

- доступ к форуму обсуждений, где каждый пользователь может оставить отзыв по тому или иному блюду, аспекту обслуживания, или высказать свое мнение в темах других пользователей. Данный форум позволяет аналитическим центрам главного офиса, основываясь на отзывах своих покупателей, контролировать работу различных ресторанов и своих сотрудников. Производить оценку новых блюд, или вносить изменения в существующие;
- информацию по ассортименту ресторанов, акциях и новинках. Для каждого ресторана можно отобразить стоимость блюда, общую информацию о составе и весе;
- информацию об адресах всех заведений в виде удобной интерактивной карты;
- также дополнительно можно ознакомиться с краткой информацией о компании, и о доступных на данный момент вакансиях.

Можно заметить, что ресурс решает только задачи по предоставлении информации пользователям, однако напрямую контактировать с рестораном они не могут. Моим предложением в данном случае является дополнение этого интернет-ресурса модулем, который позволял бы используя уже имеющуюся информацию об ассортименте и существующих ресторанах, удаленно формировать заказ. Таким образом клиенты получили бы возможность экономить время на его формирование, делая это заранее. А в случае его популяризации, компания смогла бы сэкономить на обучении и зарплатах обслуживающего персонала.

Целью данной выпускной дипломной работы служит предложение по разработке подобного приложения, которое позволит уменьшить время принятия заказов от клиентов, а также предоставить возможность делать это удаленно. Результатом проделанной работы станет готовый продукт

самообслуживания, который позволит уменьшить нагрузку на обслуживающий персонал, уменьшить количество персонала за кассой, и в следствии понизить расходы на обучение и заработные платы.

В данном случае стоит понимать, что многие аспекты для полного внедрения приложения нам не доступны. Это такие вещи, как: архитектура приложения управления производством, работа бухгалтерии и складских помещений и другие. Поэтому мы будем считать, что база данных приложения и его работа будет анализироваться отдельно от всего производства, т.е. не будет затрагивать работу существующих проектных решений, а будет работать параллельно с ними.

1.3 Постановка задачи

1.3.1 Постановка задачи и определение целей.

Задача проекта – разработать приложение для обслуживания клиентов, которое позволит им самостоятельно формировать и оплачивать заказ.

Целью реализации проекта является:

- 1) оптимизация принятия заказов от клиентов, тем самым позволив им экономить свое время;
- 2) оптимизировать штат сотрудников, уменьшив количество обслуживающего персонала;
- 3) разработка должна быть доступной в использовании для максимального числа людей.

Разработка приложений для компаний не нова в данной сфере. На сегодняшний день существует множество различных приложений, которые базируются как на базе персональных компьютеров, так и на базе мобильных приложений или сайтов. В последующих пунктах работы мы проведем анализ различных средств решения, и выберем наиболее оптимальный для нас вариант.

1.3.2 Характеристика организации решения задачи

Реализация будущего проекта будет выполняться по схеме, представленной на рисунке 1.5.

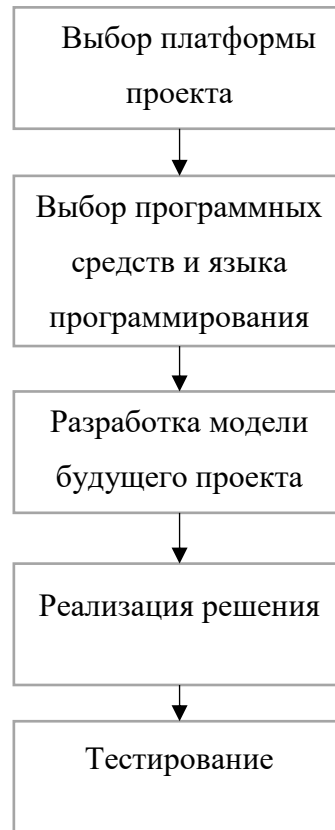


Рисунок 1.5 - Поэтапная схема реализации проекта

1.4 Выбор проектных решений.

1.4.1 Выбор платформы будущего проекта

Как было сказано ранее, для реализации данного проекта существует множество вариантов.

В нашем случае, при выборе платформы приложения мы будем придерживаться некоторых критериев:

1) Простота реализации. Так как целью нашего проекта является оптимизация процесса заказа, реализация проекта не должна занимать много времени и средств. Проект должен быть легко обслуживаем и расширяем в будущем.

2) Проект должен использовать распространенные языки программирования. Для дальнейшего обслуживания желательно базировать приложения на популярном языке, тем самым облегчить поиск специалистов и снизить стоимость обслуживания.

3) Стоимость. Желательно иметь наличие бесплатных сред разработки. Использовать уже имеющиеся наработки компании, дабы уменьшить общий объем работ.

4) Доступность. Клиенты должны иметь возможность легко получать доступ к приложению, независимо от платформ устройств и их типов. Плюсом будет отсутствие надобности установки приложения на устройства клиентов.

Под вышеперечисленные критерии отлично подходят web-решения. У компании уже имеется веб сайт, и его расширение станет удобным для нас решением. Они являются мульти-платформенными, а для их использования нужно лишь подключение к сети интернет. Их разработка не занимает много времени и средств, а их гибкость позволяет настроить заполнение под любые нужды.

1.4.2 Выбор программных средств и языка программирования

Мы определились, что реализация нашего проекта будет базироваться на web-решении. Теперь стоит определиться со средой, где будет происходить разработка данного решения.

Простые страничные сайты создаются на распространённом языке HTML, и могут писаться буквально в блокноте. Но они в основном не способны реализовывать запросы пользователей, и хранить информацию. Их расширением являются web-приложения. Использование веб-приложений приносит пользу как разработчикам веб-сайтов, так и их конечным пользователям, в общей сложности позволяют:

- быстро и легко находить нужную информацию на сайтах с большим объемом данных;
- собирать, анализировать и сохранять данные от посетителей;
- обновлять веб-сайты с периодически изменяющимся содержанием.

Логика таких приложений может программироваться на различных языках программирования.

Некоторые из критериев нашего проекта гласят, что нам желательно иметь бесплатную среду разработки и распространённый язык программирования. Под эти критерии идеально подходит фреймворк Visual Studio 2015 Enterprise, позволяющей разрабатывать приложения на языке программирования C#. Данная среда позволяет создавать различные проекты для всевозможных платформ, проста в освоении, а вся дополнительная документация находится в открытом доступе на официальном сайте.

Ко всему прочему Visual Studio имеет в себе инструменты для создания и работы с базами данных MySQL, что несомненно является важным фактором для реализации web-приложения.

2 Разработка web-приложения для обслуживания клиентов в сети ресторанов быстрого питания

В настоящее время существует множество технологий и средств создания web-сайта, все они позволяют сделать его уникальным и подогнать под рамки определенных нужд. Но в общей купе они все похожи, и весь процесс можно разделить на несколько этапов.

Во-первых, это серверная часть приложения. В независимости на каком языке она будет написана, она должна выполнять свою ключевую роль – обработку запросов пользователя, анализ и манипулирование информацией изменение содержимого на страницах. Таких языков на данный момент множество, но в данной работе мы будем использовать Asp.Net C#.

Во-вторых, неотъемлемая часть веб-приложения – база данных. В ней будет храниться вся информация сайта, такие как товары, заказы, записи пользователей и другие. На некоторых сайтах даже содержимое страниц может храниться в базе данных.

В-третьих, клиентская сторона. HTML(Hyper Markup Language) и CSS(Cascading Style Sheets) являются главными компонентами современного сайта, они задают внешний вид и стиль будущему проекту. Иначе говоря, HTML задает разметку, составляет содержимое и сообщает браузерам что отобразить на странице. CSS предоставляет форматы и стили, и задает внешний вид элементов сайта.

Разработка проекта для выпускной квалификационной работы будет происходить по нескольким этапам, отраженных на рисунке 2.1.



Рисунок 2.1 - План реализации web-приложения

В реализации данного web-приложения будет использоваться доступная в Visual Studio 2015 архитектура построения динамических сайтов Asp.Net MVC 5.

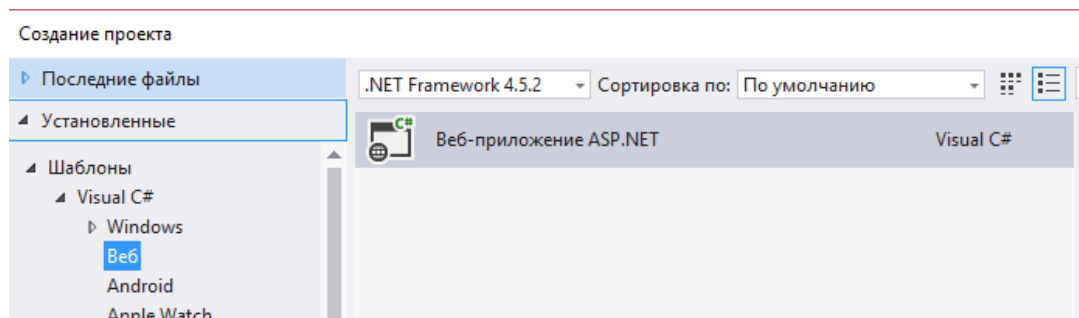


Рисунок 2.2 - Создание приложения ASP.NET

Core MVC предлагает различные функции, которые очень полезны для создания веб-приложений. Данный фреймворк использует парадигму MVC (Model – View – Controller) – то есть модель-представление-контроллер.

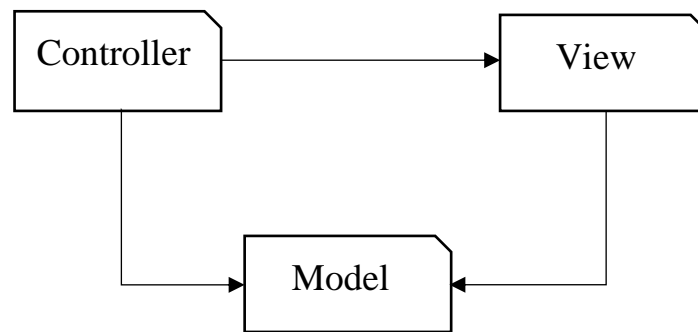


Рисунок 2.3 - Реализацию парадигмы MVC в рамках фреймворка ASP.NET MVC

Каждый из этих пакетов в рамках ASP.NET реализуется как отдельная сущность, поддерживаемая библиотеками. Так, Модель и Контроллер реализуются как классы, Представление – как HTML-файлы с вставками на языке C#, и имеют расширение .cshtml.

2.1 База данных

2.1.1 Описание модели проекта

Для записи различных данных как о клиентах, так и о товарах требуется база данных. В среде разработки Visual Studio 2015 присутствуют инструменты для работы с базами MySQL.

Для соответствия результатов поставленным задачам, нужно обозначить инфологическую модель проекта, описать логику действий контроллеров на действия пользователей.

Конечное web-приложение должно выполнять функции:

- 1) Давать представление об ассортименте ресторана, ценах и наличии.
- 2) Возможность составления корзины покупок и ее редактирования
- 3) Выбор типа покупки и ее оформление.
- 4) Предоставлять суперпользователям информацию из базы данных, сортировать ее для нужд администратора.

Для наглядного понимания, какие элементы должны содержаться в нашей будущей базы данных, мы составим морфологическую модель нашего проекта.

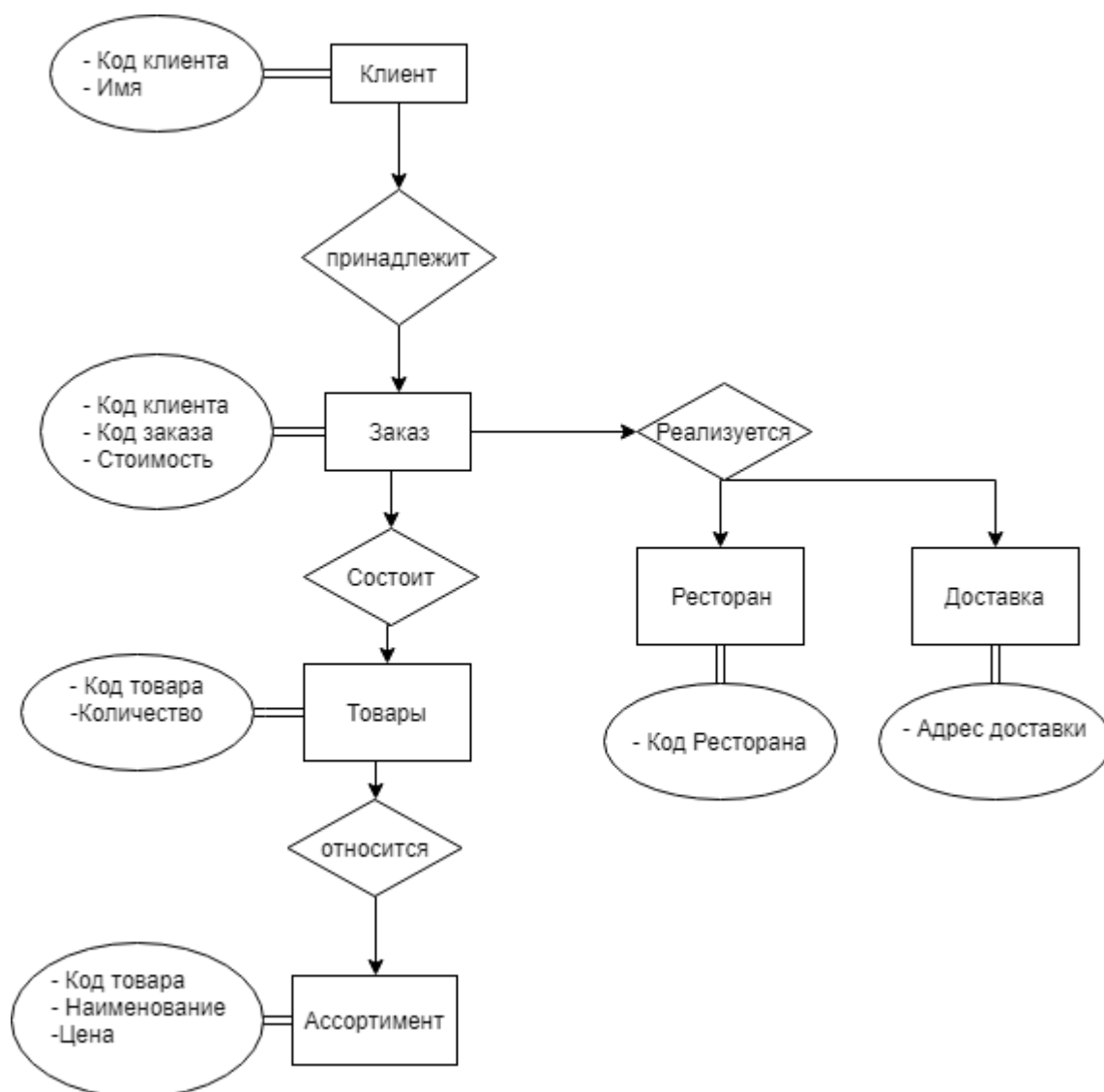


Рисунок 2.4 - Модель работы приложения

- 1) Пользователь просматривает ассортимент и составляет корзину из товаров ассортимента
- 2) Оформляет покупку и выбирает один из вариантов получения заказа
- 3) Пользователь оплачивает свой заказ
- 4) Происходит формирование заказа и запись информации в таблицы
- 5) Просмотр заказов для авторизованных пользователей осуществляется через выгрузку данных из таблиц.
- 6) Предполагается возможность экспорта данных в программы управления производством, формирование складских отчетностей.

Так как данная выпускная квалификационная работа является в первую очередь учебной, в ней не будет реализован весь перечень требований, который требуется для проектов такого масштаба. В этой работе мы отразим только основные пункты, необходимые для «web-приложения обслуживания клиентов в сети ресторанов быстрого питания».

2.1.2 Описание таблиц

Из описания функциональной модели мы видим, что для создания базы данных нам нужно обозначить несколько сущностей, которые характеризовали бы наш проект. Сущность – это объект, который может быть идентифицирован неким способом, отличающим его от других объектов. Сущность имеет множество поименованных свойств (атрибутов) и существует во множестве экземпляров.

Таблица 1 - Сущности и их описания

Сущность	Описание сущности	Атрибуты сущности
Товары	Хранит ассортимент товаров	Номер товара
		Название
		Тип товара
		Описание
		URL представления
		Цена единицы
Ресторан	Хранит информацию о ресторане	Номер ресторана
		Адрес ресторана
Заказы для ресторана	Хранит информацию по заказам для ресторанов	Номер Заказа
		Номер ресторана
		Сумма заказа
Товары для заказов ресторана	Дочерняя таблица для заказов ресторана. Хранит перечень товаров, входящих в заказ.	Номер заказа
		Номер товара
		Количество
		Стоимость товара

Продолжение таблицы 1

Покупатель	Хранит информацию о покупателях и информацию о заказе для доставки товаров	Номер Покупателя
		Фино покупателя
		Адрес
		Моб. телефон
		Номер заказа
		Сумма заказа
Товары для доставки	Дочерняя таблица для покупателя. Хранит перечень товаров, входящих в заказ.	Номер заказа
		Номер товара
		Количество
		Стоимость товара

Далее происходит формирование таблиц в среде MySQL фреймворка Visual Studio. Для этого к нашему проекту добавляется элемент данных Sql Server.

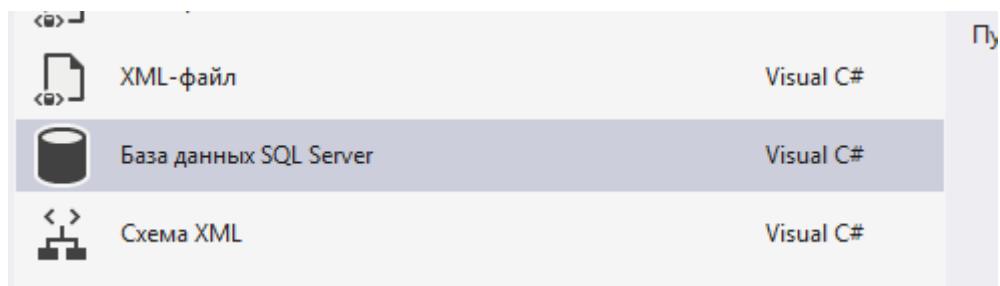


Рисунок 2.5 - Элемент баз данных SQL Server

После подключение элемента SQL сервера, мы имеем возможность добавлять и редактировать таблицы данных.

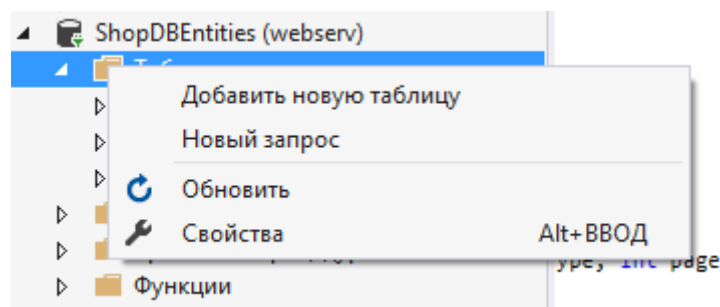


Рисунок 2.6 - Добавление таблиц данных

2.2 Серверная часть web-приложения

1) Обращения к базе данных

Для дальнейшей работы с данными и обращения к ним в архитектуре MVC требуется создать модель представления. Это осуществляется добавлением в проект нового элемента модели ADO.NET EDM. Для этого нажимаем правой кнопкой по папке с моделями и добавляем новый элемент.

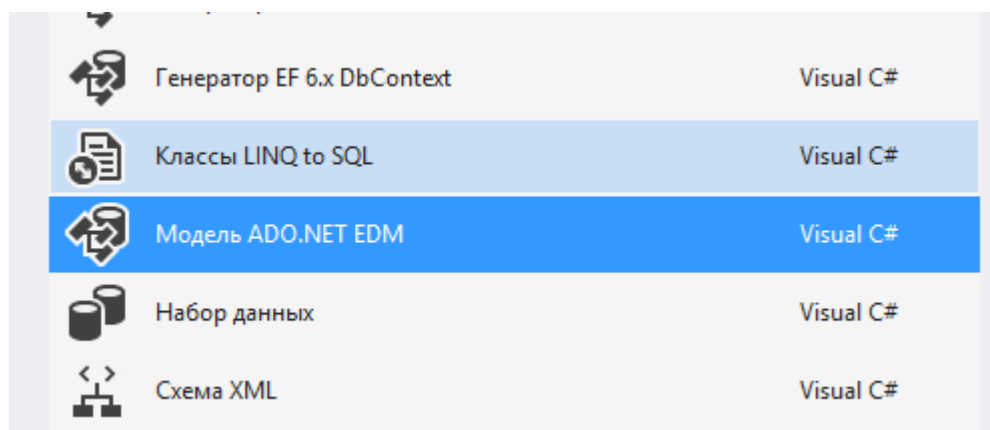


Рисунок 2.7 - Добавление модели ADO.NET

Присваиваем ей имя «ShopDBModel», а в роли источника данных указываем нашу базу данных и таблицы, с которыми требуется работа. Далее осуществляется автоматическое формирование кода, и теперь в нашем проекте существуют классы наших сущностей. Таким образом, теперь у нас есть сущности таблиц из нашей базы данных.

Данные нашего проекта хранятся в базе данных, мы уже имеем модель ее сущностей. Теперь для обращения к ним достаточно создать новую переменную для этого объекта:

```
private Models.ShopDBEntities db = new Models.ShopDBEntities(); .
```

Теперь для обращения к сущностям модели баз данных мы можем обращаться посредством переменной «db».

2) Создание контроллеров и методов

Реализация контроллера производится через унаследование класса от `System.Web.Mvc.Controller`. Обработка запросов обеспечивается отдельно

объявляемыми методами действий, которые вызываются при обработке запросов контроллером приложения. Модели класса описываются на языке, используемым сервером, в данном случае это C#, и предоставляются как отдельные классы.

Для нашего проекта, основными частями являются:

- каталог товаров;
- корзина;
- оформление заказа;
- панель администрирования.

Принцип их работы схож, и некоторые из них будут описаны ниже.

2.2.1 Описание контроллера для выгрузки списка товаров

Главным элементом нашего приложения является ассортимент доступных товаров. Покупатель не сможет оформить заказ, если ему не предоставить выбор товаров.

Для удобной работы, создадим новый котроллер `TovarsController`, в котором будем описывать методы для частичного представления листа товаров.

Подразумевается, что товаров у нас будет n -ое количество, и отображать их все на одной странице будет нецелесообразно. Поэтому стоит задуматься о разделении всего списка на страницы.

Создадим целочисленную переменную «pagesize», которая будет отвечать за количество товаров на странице. Задавая ей значение, мы сможем с легкостью задавать количество отображаемых элементов.

Опишем вспомогательные методы для определения ссылок будущим страницам. Для этого мы реализуем многократно используемый вспомогательный метод HTML, который будет генерировать разметку для требуемых навигационных ссылок. Создадим модель представления, которая будет передавать количество страниц. Этот новый класс «PagingInfo» будет передавать данные между контроллером и представлением.

```

public class PagingInfo
{
    public int TotalItems { get; set; } // общее количество элементов
    public int ItemsPerPage { get; set; } //элементов на странице
    public int CurrentPage { get; set; } //текущая страница
    public int TotalPages
    {
        get { return (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage); }
    }
}

```

Рисунок 2.8 - Класс «PagingInfo»

Теперь реализуем вспомогательный метод «PagingHelper». Этот метод, расширяющий «PageLinks()» генерирует HTML-разметку для набора ссылок на страницы с использованием информации в объекте «PagingInfo». Параметр «Func» принимает делегат, который применяется для генерации ссылок, облегчающих просмотр других страниц.

```

public static class PagingHelper
{
    public static MvcHtmlString PageLinks(this HtmlHelper html, PagingInfo paginginfo, Func<int, string> pageUrl)
    {
        StringBuilder result = new StringBuilder();
        for (int i = 1; i <= paginginfo.TotalPages; i++)
        {
            TagBuilder tag = new TagBuilder("a");
            tag.MergeAttribute("href", pageUrl(i));
            tag.InnerHtml = i.ToString();
            if (i == paginginfo.CurrentPage)
            {
                tag.AddCssClass("selected");
                tag.AddCssClass("btn-primary");
            }
            tag.AddCssClass("btn btn-default");
            result.Append(tag.ToString());
        }

        return MvcHtmlString.Create(result.ToString());
    }
}

```

Рисунок 2.9 - Класс «PagingHelper»

Теперь необходимо задуматься о передаче представлению экземпляра класса модели представления «PagingInfo». Можно использовать директивы «using», но удобнее всего поместить все данные, которые требуется передать из контроллера представлению, в один класс модели представления. Этим классом будет «TovarsListViewModel», который будет состоять из списка товаров и модели «PagingInfo».

```

public class TovarsListViewModel
{
    public IEnumerable<Tovar> Tovars { get; set; } //список товаров
    public PagingInfo PagingInfo { get; set; } //ссылки страниц
    public string CurrentType { get; set; } //свойство фильтра по типу товара
}

```

Рисунок 2.10 - Класс «TovarsListViewModel»

Когда приготовления закончены, можно приступить к описанию метода для модели конечного представления. Опишем новый метод результата представления «List». В данном методе будет формироваться выборка товаров на страницу и HTML-разметка для них. Внутри тела метода мы формируем новую переменную класса «TovarsListViewModel», которая получает объекты «Tovars», упорядоченные по категориям. При этом пропускаются товары располагаются до начала текущей страницы и выбирается количество равное указанному в поле «pagesize».

Передавая методу «View()» список объектов, мы снабжаем инфраструктуру данными, которыми заполняется объект представления в строго типизированном представлении.

```
public ActionResult List(string Type, int page = 1)
{
    TovarsListViewModel model = new TovarsListViewModel
    {
        Tovars = db.Tovars // обращение к базе данных
        .Where(b => Type == null || b.Type == Type) //сравнение типов для фильтрации
        .OrderBy(tovar => tovar.Type) //упорядочивание
        .Skip((page - 1) * pagesize) //пропуск товаров с предыдущих страниц
        .Take(pagesize), //выборка указанного количества
        PagingInfo = new PagingInfo
        {
            CurrentPage = page,
            ItemsPerPage= pagesize,
            TotalItems = Type==null ? db.Tovars.Count() : db.Tovars.Where ( tovar => tovar.Type == Type).Count()
        },
        CurrentType = Type //фильтрация типов
    };

    return View(model); //возврат модели в пользовательское представление
}
```

Рисунок 2.11 - Метод представления «List»

Теперь для этого метода можно сформировать представление, которое будет описано в главе ниже.

2.2.2 Описание контроллеров корзины

Процесс покупки в приложении должен быть понятен всем, кто когда-либо сталкивался с интернет магазинами. Этот процесс не должен вызывать проблем, и осуществляться на интуитивном уровне.

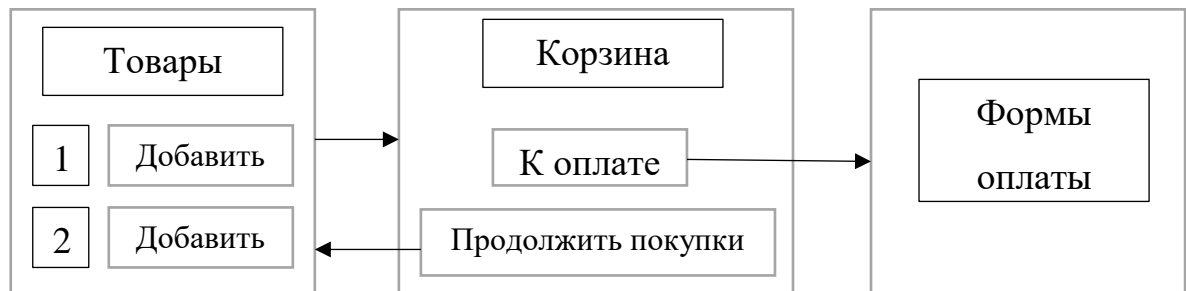


Рисунок 2.12 - Схема работы корзины

Для этого реализуем простую схему, когда для каждого товара будут элементы управления, добавляющие его в корзину, с переходом или без. В представлении самой корзины также будут элементы управления, позволяющие удалить ненужные позиции, вернуться к выбору или перейти к следующим формам оформления заказа. Для наглядного представления информации пользователям, необходимо будет осуществлять вывод наименований их выбора, количества и общей стоимости.

Корзина для покупок является частью предметной области нашего проекта, поэтому для ее представления требуется создать сущность в модели предметной области.

Создадим новые классы «Cart» и «CartLine». Класс «Cart» использует класс «CartLine», который осуществляет предоставление выбранного товара и его количество в заказе.

Для удобной работы с корзиной определим методы по добавлению, удалению и ее отчистки, а также вспомогательный метод по подсчету общей суммы по товару.

1) Метод добавления товара

```

public void Additem(Tovar tovar, int quantity) //метод добавления в корзину
{
    CartLine line = lineCollection //
        .Where(b => b.Tovar.Id == tovar.Id) //выдача первого соответствия этого товара в списке корзины
        .FirstOrDefault();
    if (line == null) // проверка наличия товара в корзине
    {
        lineCollection.Add(new CartLine { Tovar = tovar, Quantity= quantity }); //добавление, при условии отсутствия
    }
    else
    {
        line.Quantity += quantity; //увеличение количества, при наличии
    }
}

```

Рисунок 2.13 - Метод добавления товара в корзину

Данный метод принимает переменную сущности товаров и количества. В теле метода осуществляется создание новой переменной класса «CartLine», которая приравнивается к значению переменной в списке «lineCollection». При этом описываются дополнительные исключения, которые осуществляют поиск по списку по входным данным переменной «Tovar».

Далее происходит простое сравнение. Если поиск по списку обернулся неудачей, значение переменной останется пустым. В этом случае, мы добавляем к списку новую строку, содержащую данный товар и количество. Иначе, если такой товар в корзине имеется, мы увеличиваем его количество.

2) Метод удаления

```

public void Removeline(Tovar tovar) //метод удаления товара из корзины
{
    lineCollection.RemoveAll(l => l.Tovar.Id == tovar.Id);
}

```

Рисунок 2.14 - Метод удаления из корзины

Метод удаления работает схожим образом, и удаляет элемент по указанному индексу из коллекции.

3) Метод отчистки корзины базируется на встроенном методе отчистки списка

```

public void ClearCart() //Отчистка корзины
{
    lineCollection.Clear();
}

```

Рисунок 2.15 - Метод отчистки корзины

4) Подсчет общей суммы по товару. Осуществляется простое произведение цены товара на заказанное количество.

```
public decimal TovarsTotalValue() //подсчет общей суммы
{
    return lineCollection.Sum(e => e.Tovar.Price * e.Quantity);
}
```

Рисунок 2.16 - Метод подсчета общей стоимости заказа

Сущность корзины описана. Теперь стоит приступить к описанию контроллера, который обрабатывал бы запросы пользователей. Создадим новый контролер «Cart». Методы контроллера вызываются из представления, когда пользователь осуществляет какие-либо действия.

```
ССЫЛКА: 0
public RedirectToRouteResult AddToCart(Cart cart, int tovid, string returnUrl, int a)
{
    Tovar tov = new Tovar();
    tov = db.Tovars
        .FirstOrDefault(b => b.Id == tovid);
    if (tov != null)
    {
        cart.AddItem(tov, 1);
    }

    if (a == 1) return RedirectToAction("Index", new { returnUrl });
    else return RedirectToAction("List", "Tovars", new { returnUrl });
}
```

Рисунок 2.18 - Метод контроллера для добавления в корзину

На рисунке 2.18 представлено описание метода «AddToCart» контроллера корзины, который осуществляет добавление в нее. Входными данными является переменная класса корзины, объявляющая текущую сессию, а также целочисленный идентификационный номер требуемого товара, строка URL и целочисленный флаг. В теле метода осуществляется создание новой переменной класса «Tovar». Далее происходит обращение к базе данных, где происходит поиск товара по входным данным «tovid». В случае успешного поиска, вызывается метод добавления из класса «Cart», в котором входными данными являются наш товар и количество равное 1.

Методы «AddToCart()», «RemoveFromCart()» и другие методы контроллеров вызывают метод «RedirectToAction()», в результате чего

клиентскому браузеру отправляется инструкция по перенаправлению HTTP, заставляющая браузер запросить новый адрес. В случае, указанном на рисунке выше, метод «AddToCart()» в зависимости от значения флага запросит адрес, который вызывает метод действия «Index()» контроллера «Cart», или же «List()» контроллера «Tovars». Но в любом случае, данные методы используются для отображения содержимого контроллеров.

Представлению, которое будет отображать содержимое нашей корзины, необходимо передать информацию об объекте «Cart» и адресе URL, для возвращения по нажатию кнопки «Продолжить покупки». Для этого, как и в случае со списком товаров, создадим простой класс модели представления «CartIndexViewModel».

```
ссылка: 3
public class CartIndexViewModel
{
    ссылка: 3
    public Cart Cart { get; set; }
    ссылка: 3
    public string returnUrl { get; set; }
}
```

Рисунок 2.19 - Класс «CartIndexViewModel»

Для сохранения и извлечения объектов «Cart» применятся метод, использующий средство состояния сеанса ASP.NET. Эти средства позволяют ассоциировать множество различных запросов как единую группу от определенного пользователя, и формируют отдельный сеанс работы. Это решение позволяет разным пользователям иметь собственные корзины, которые сохранялись бы между запросами. Данные сеансов удаляются по истечению определенного времени, что означает отсутствие надобности в контроле хранения или жизненного цикла объектов корзин

Создадим новый класс, реализующий интерфейс «IModelBinder», и назовем его «CartModelBinder». В данном интерфейсе определен один метод «BindModel()», который принимает два параметра для создания модели предметной области. Параметр «ControllerContext» обеспечивает доступ ко всей информации класса контроллера, а второй, «ModelBindingContext»,

предоставляет сведения об объекте модели, который мы создаем. Первый параметр нам необходим в первую очередь по тому, что его свойство «HttpContext» содержит необходимые свойства сессии.

```
ССЫЛКА: 1
public class CartModelBinder: IModelBinder
{
    private const string sessionkey = "cart";
    ССЫЛОК: 0
    public object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
    {
        Cart cart = null;
        if (controllerContext.HttpContext.Session!=null)
        {
            cart = (Cart)controllerContext.HttpContext.Session[sessionkey];
        }
        if (cart==null)
        {
            cart = new Cart();
            if (controllerContext.HttpContext.Session != null)
            {
                controllerContext.HttpContext.Session[sessionkey] = cart;
            }
        }
        return cart;
    }
}
```

Рисунок 2.20 - Описание класса «CartModelBinder»

Объект «Cart» создается путем считывания ключа из данных сеанса и создания его экземпляра, если при проверке оказалось, что такого не существует. Теперь осталось только сообщить инфраструктуре MVC, что для создания экземпляра класса «Cart» она может пользоваться нашим новым классом.

```
ССЫЛКА: 0
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
    ModelBinders.Binders.Add(typeof(Cart), new CartModelBinder());
}
```

Рисунок 2.21 - Модификация структуры MVC Framework

Теперь наш контроллер корзины будет получать объекты «Cart», основываясь на сессиях пользователей.

2.3 Клиентская часть web-приложения

После того как контроллеры готовы, можно заняться их визуальным представлением.

По умолчанию, новое приложение уже имеет начальную страницу «_Layout.cshtml». Именно ее мы и будем считать мастер-страницей, т.е. ее элементы будут отображаться на всех страницах приложения. Места же других элементов будут заданы методом «@RenderBody()». Этот метод является определением позиции, на место которой другие представления будут подставлять свое содержимое. Таким образом сохраняется общий стиль, который поддерживает легкое добавление новых представлений.

Вся структура документа HTML состоит из тегов, по которым веб-браузер определяет, как должен выглядеть готовый документ. Из всех тегов можно выделить главные, которые должны присутствовать в документе.

Самым первым тегом в документе идет <HTML>. Он сообщает веб-браузеру на каком языке данных документ написан.

Далее следует теги <HEAD> и <BODY>. Они делят документ на две части: заголовок и основную, то есть тело. В теле размещается вся его текстовая и графическая часть.

И наконец тег <Title>, содержащий текст заголовка документа. Он указывается сразу после тега <HTML> и более нигде. Он представляет из себя обобщенное описание документа, и большинство браузеров отображают информацию заголовка в заголовке окна и в файле закладок.

Стоит отметить что все они являются парными и должны иметь закрывающие теги. Все они позволяют веб-браузерам уверенно делить заголовочную часть документа от смысловой.

2.3.1 Дополнительные средства разработки

Для дополнительной настройки стилей элементов, в проекте задействован Bootstrap v4.3.1. Это простой в использовании фреймворк готовых стилей, разметок и интерактивных компонентов, которые позволяют разрабатывать функциональные и красивые web-приложения и сайты за короткий промежуток времени. Данный инструментарий может добавить в свой проект любой желающий, а вся нужная документация присутствует на их сайте.

Для его подключения есть несколько вариантов. Мы же воспользуемся самым простым – через загрузку пакетов NuGet. Управление пакетами `nugget` находится в свойствах проекта VisualStudio, где осуществляется поиск и загрузка необходимых элементов.

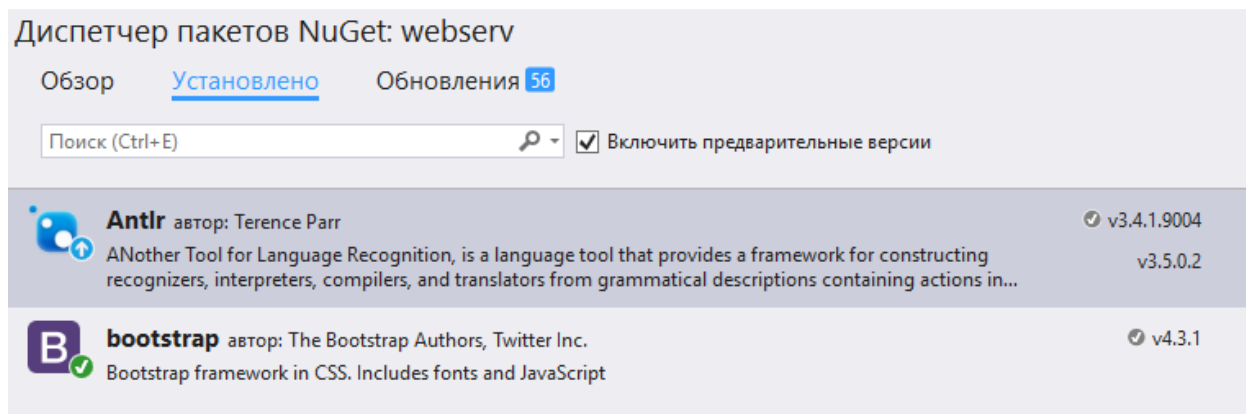


Рисунок 2.22 – Панель управления пакетами NuGet

После установки всех нужных элементов, в папке проекта «Content» появятся соответствующие элементы. Чтобы их задействовать в работе, достаточно просто перетащить их в заголовок мастер-страницы, а среда Visual Studio самостоятельно сформирует код.

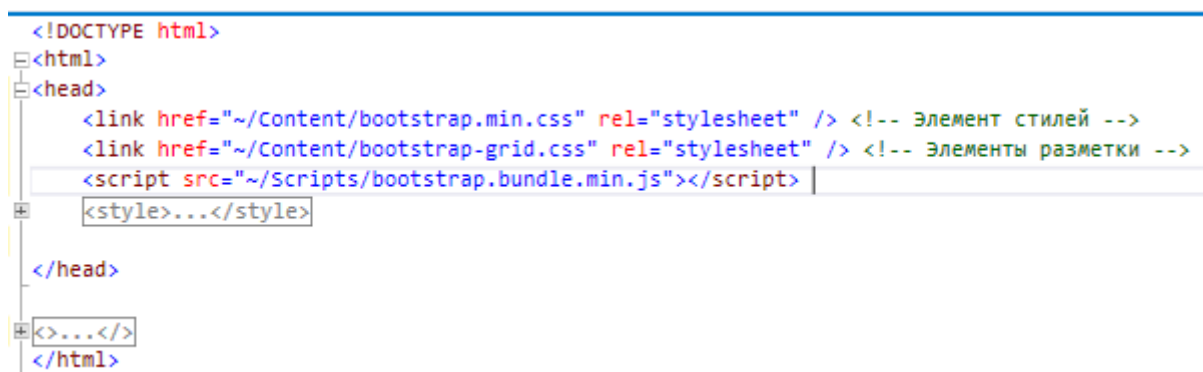


Рисунок 2.23 – Указание ссылок на элементы «Bootstrap»

2.3.2 Структура мастер-страницы. Работа с представлениями

Как говорилось ранее, элементы мастер-страницы будут отображаться на всех страницах. Это идеально подходит для различных навигационных элементов. При помощи блочных элементов `<div>`, и классов «Bootstrap» мы разделим тело приложения на несколько частей, указанных на рисунке 2.24.

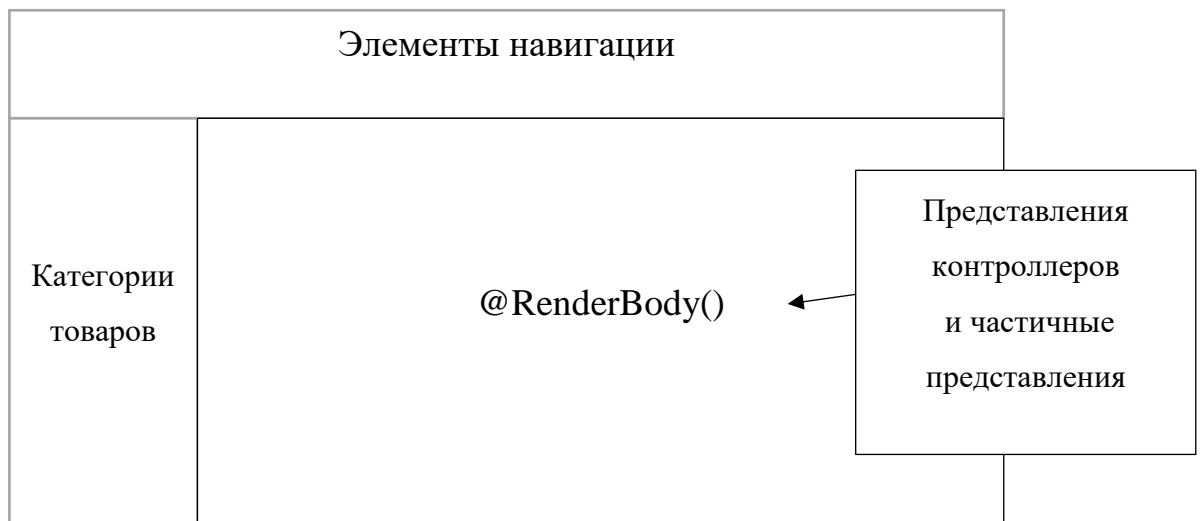


Рисунок 2.24 - Общая структура представления

Для элементов навигации будет использован класс «navbar», реализующий горизонтальное адаптивное меню.

```
<div class="container bg-light">
  <nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <button class="navbar-toggler collapsed">...</button>
    <div class="navbar-collapse collapse" id="navbarCollapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-justified">
            @Html.ActionLink(" Аккортимент ", "List", "Tovars"
              , new { area = "" } ,new { @class = "nav-brand" })
            <span class="sr-only"></span>
          </a>
        </li>
        <li class="nav-item active">
          <a class="nav-tabs ">
            @Html.ActionLink(" О программе ", "About", "Home"
              , new { area = "" } , new { @class = "nav-link " })
          </a>
        </li>
      </ul>
    </div>
  </nav>
</div>
```

Рисунок 2.25 – Часть разметки навигационного меню

На рисунке можно увидеть часть разметки мастер страницы. Элемент <div> объявляет контейнер для содержимого. Далее идет объявление раздела с ссылками для перехода, элементов и , характеризующих список и элементы списка. При помощи описания класса элементов, мы задаем им стиль и положение в сетке, используя свойства Bootstrap. В нем уже имеются различные настройки, такие как цвет фона, стили кнопок, элементов и множество другое. Со всеми настройками сеток и стилей Bootstrap можно ознакомиться на официальном сайте продукта. Помимо множества библиотек

по свойствам элементов, там можно ознакомиться с готовыми примерами, просмотреть их код и принципы работы.

Чтобы разделить нижнюю часть приложения на сегменты для категорий и тела, воспользуемся готовой сеткой. Система сетки «Bootstrap» использует ряд контейнеров, строк и столбцов для размещения и выравнивания содержимого. Таким образом используя свойства «row» и «col», а также указывая их положения на сетке, можно размечать множество различных блоков с содержимым, которых в нашем случае будет две.

```
<div class="row">
  <nav class="col-md-2 d-none d-md-block sidebar">
    <div class="container py-5">
      <div class="sidebar-sticky">
        <ul class="nav flex-column">
          @Html.Action("Menu", "Nav")
        </ul>
      </div>
    </div>
  </nav>
  <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">
    @RenderBody()
  </main>
</div>
```

Рисунок 2.26 – разделение колонок с содержимым

Как видно на рисунке 2.26, сначала происходит открытие контейнера с указанием классового свойства строки, а внутри содержатся два элемента со свойствами столбцов и дополнительными настройками положения и стиля.

Первый элемент вызывает метод «Menu» контроллера «Nav» для визуализации категорий, а второй столбец определяет позиции замещаемых элементов.

```
public PartialViewResult Menu(string types = null)
{
    ViewBag.SelectedCategory = types;

    IEnumerable<string> Type = db.Tovars
        .OrderBy(x => x.Type)
        .Select(x => x.Type)
        .Distinct().OrderBy(x => x);
    return PartialView(Type);
}
```

Рисунок 2.27 – Метод контроллера для навигации

Данный метод сканирует все имеющиеся категории товаров и составляет список. Далее реализуется частичное представление, которое выводит этот список в удобном и стилизованном виде.

```
@model IEnumerable<string>
@foreach (var link in Model)
{
    <li class="nav-item active">
        <a class="nav-pills">
            @Html.RouteLink(link, new
            {
                controller = "Tovars",
                action = "List",
                Type = link,
                page = 1
            }, new
            {
                @class = "btn btn-light btn-block"
                + (@link == ViewBag.SelectedCategory ? " btn-primary" : "")
            })
        </a>
    </li>
}
```

Рисунок 2.28 – Частичное представление метода «Menu»

Как можно заметить, помимо языка разметки HTML, можно использовать методы С# обозначив их символом «@». Основные функции навигации осуществляются при помощи операторов «@Html.ActionLink» или «@HTML.BeginForm», которые определяют ссылки на методы контроллеров. К примеру строка «@Html.ActionLink(" *Ассортимент* ", "List", "Tovars")» создает гиперссылку с текстом " *Ассортимент*", при нажатии по которой производится запрос к методу частичного представления «List» контроллера «Tovar».

В главе мы 2.2.1 мы описывали контроллер для вывода списка товаров. Метод «List» передает в модель представления объект класса «TovarsListViewModel». Чтобы облегчить работу со списком повторяющихся элементов, можно воспользоваться оператором «foreach» языка С#, которое будет внедрять частичное представление каждого товара, подобно шаблону.

```

@using webserv.HtmlHelpers
@model webserv.Controllers.TovarsListViewModel
@{
    ViewBag.Title = "Товары";
}
<div class="album py-5 bg-light">
    <div class="container">
        <div class="row">
            @foreach (var item in Model.Tovars)
            {
                @Html.Partial("_TovSummary", item)
            }
        </div>
    </div>
</div>
<div>
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("List", new { Page = x, Type = Model.CurrentType }))
</div>

```

Рисунок 2.29 – Модель представления

Частичное представление схоже с обычным, за исключением лишь того, что оно создает фрагмент разметки HTML, а не новый документ. Вызов частичного представления осуществляет вспомогательный метод «@HTML.Partial()», который в качестве параметров принимает имя представления и объект модели. В данном случае именем представления является «_TovSummary», а моделью каждый элемент «Товар».

```

@model webserv.Models.Tovar
<div class="col-md-4">
    <div class="card mb-4 shadow-sm bg-white">
        <svg class="bd-placeholder-img card-img-top">...</svg>
        <div class="card-body">
            <h3 id="tovname" class="text-justify">@Model.Name</h3>
            <h5 id="tovdesc" class="text-justify">@Model.Description </h5>
            <hr>
            <h4 class="text-right">@Model.Price py6</h4>
            <div id="cartbtn" class="row">
                @using (Html.BeginForm("AddToCart", "Cart", new { tovid = @Model.Id, returnUrl = Request.Url.Query }))
                {
                    <div class="d-flex justify-content-between align-items-center">
                        <div>...</div>
                    </div>
                }
                @using (Html.BeginForm("AddToCart", "Cart", new { tovid = @Model.Id }))
                {
                    <div class="d-flex justify-content-between align-items-center">
                        <div>...</div>
                    </div>
                }
            </div>
        </div>
    </div>
</div>

```

Рисунок 2.30 – Частичное представление товара

Для каждого товара мы создаем свое представление, а обращаясь к модели, получаем такие данные, как наименование, описание и цену. Используя Post методы, мы внедряем интерактивные кнопки, которые реализуют методы по добавлению товара в корзину.

Использование таких частичных представлений способствует уменьшению дублирования, и они особенно полезны, когда требуется визуализировать однородные данные в разных частях приложения.

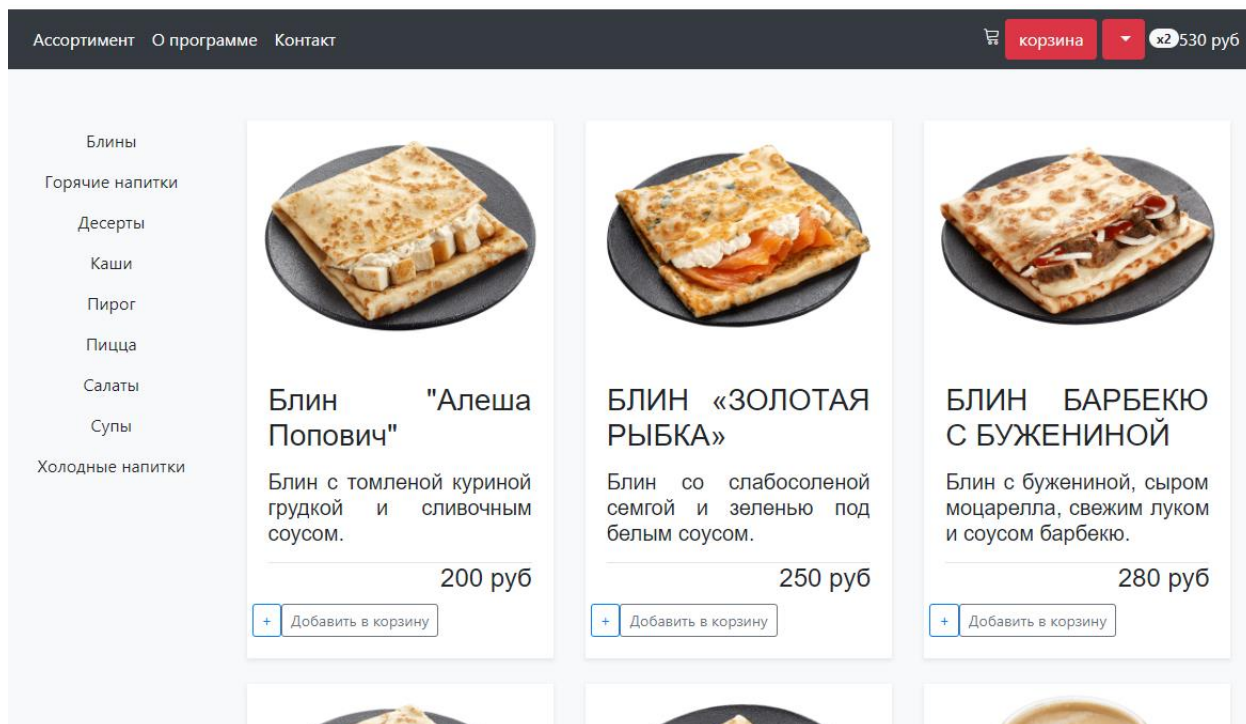


Рисунок 2.31 – Результат работы представления

3 Расчет оценки эффективности разработанного проектного решения

Итоговой целью написания экономического раздела выпускной квалификационной работы является определение целесообразности реализации предложенного приложения путем оценки его эффективности.

Конечной целью данной работы является создание web-приложения для обслуживания клиентов сети ресторанов быстрого питания. Необходимо рассчитать все затраты на разработку данного проекта, с учетом заработной платы сотрудников, а также сравнить их с затратами, которые были до внедрения данного решения.

При проектировании новых решений в сфере вычислительной техники затраты инвестиций определяются капитальными вложениями, то есть инвестициями, направленными на воспроизводство основных средств.

Так как предполагается, что проектирование проекта происходит в построенном здании, на имеющемся оборудовании, не производится подсчет капитальных вложений. Кроме того, не требуется переподготовка кадрового состава предприятия, отсутствует необходимость типовых разработок.

Эксплуатационные расходы представляют собой годовые издержки, связанные с реализацией мероприятия.

Эксплуатационные затраты рассчитываются по формуле:

$$З_{\text{эспл}} = З_{\text{зп}} + З_{\text{сн}},$$

Где $З_{\text{зп}}$ – затраты на заработную плату сотруднику, руб, $З_{\text{сн}}$ - страховые взносы обязательного социального страхования, руб.

Повременная оплата труда подразделяется на повременную и повременно-премиальную. В данной работе будет считаться, что в компании просто повременная оплата труда:

$$З_{\text{повр}} = C_p * T,$$

Где C_p – часовая тарифная ставка работника, руб, T – фактически потраченное работником время.

Страховые взносы обязательного социального страхования – это обязательные отчисления по установленным нормам (фонд социального страхования, пенсионный фонд, фонд обязательного медицинского страхования):

$$Z_{\text{сн}} = Z_{\text{пл}} * K_{\text{сн}},$$

Где $K_{\text{сн}}$ – ставка страховых взносов, доли, равной 0.3.

Так как основные статьи расходов после внедрения web-приложения не изменятся, достаточно посчитать выгоду в экономии заработной платы в случае уменьшения количества сотрудников.

В первую очередь нужно произвести расчеты затрат компании, которые есть до внедрения веб-приложения. Для начала обозначим некоторые входные данные:

У сети есть 5 точек. Для обслуживания клиентов в них работает 40 человек, по 20 человек на смену, используя 20 коммуникаторов, которые работают по 12 часов в день, и получают 150 р/час.

$$Z_{\text{повр}} = 150 * 40 * 12 * 365 = 26\,280\,000 \text{ руб./год}$$

Из этого следует, что за год в среднем тратиться 26 280 000 руб. на зарплату сотрудникам.

Расчет страховых взносов:

$$Z_{\text{сн}} = 26\,280\,000 * 0,3 = 7\,884\,000 \text{ руб./год}$$

$$Z_{\text{общ1}} = 26\,280\,000 + 7\,884\,000 = 34\,164\,000 \text{ руб./год}$$

Теперь следует провести расчет затрат компании после внедрения web-приложения по обслуживанию клиентов, которое позволит нам сократить общее число обслуживающего персонала.

Входные данные:

У сети есть 5 точек. Для обслуживания клиентов в них работает 30 человек, по 15 человек на смену, используя 15 коммуникаторов, которые работают по 12 часов в день, и получают 150 р/час.

$$З_{\text{повр}} = 150 * 30 * 12 * 365 = 19\,710\,000 \text{ руб./год}$$

Из этого следует, что за год в среднем тратиться 19 710 000 руб. на зарплату сотрудникам.

Расчет страховых взносов:

$$З_{\text{сн}} = 19\,710\,000 * 0,3 = 5\,913\,000 \text{ руб./год}$$

$$З_{\text{общ2}} = 19\,710\,000 + 5\,913\,000 = 25\,623\,000 \text{ руб./год}$$

Экономия:

$$Э = З_{\text{общ1}} - З_{\text{общ2}} = 34\,164\,000 - 25\,623\,000 = 8\,541\,000 \text{ руб./год.}$$

Из расчетов экономической части работы следует, что экономия от внедрения web-приложения составит 8 541 000 руб./год. Также стоит отметить, что компания не несла никаких дополнительных затрат на привлечение сторонних лиц для реализации этого приложения, так как оно разрабатывалось в рамках компании.

В результате проведения технико-экономических расчетов было определено, что разработка и внедрение web-приложения для обслуживания клиентов позволяет сократить годовые расходы, увеличить скорость обслуживания клиентов, а также улучшить доступность услуг компании для клиентов.

Заключение

Целью данной выпускной квалификационной работы являлась разработка web-приложения для обслуживания клиентов в сети ресторанов быстрого питания.

В первом разделе выпускной квалификационной работы представлен анализ бизнес процессов и организации предприятия, который показывает, что цель работы актуальна и реализация автоматизированной системы позволит оптимизировать процесс осуществления заказа клиентами. Был обоснован выбор в сторону web-решения, а также осуществлен выбор программных средств и языков программирования.

Во втором разделе описывается процесс создания web-приложения. Описание структуры базы данных и ее внедрения в проект. Описание контроллеров серверной и клиентской частей.

В последнем разделе был произведен расчет оценки эффективности разработанного проектного решения, которая показала, что экономия на сокращении обслуживающего персонала за год составит 8.5 млн. рублей.

Задачи, поставленные вначале данной работы выполнены: оптимизирован процесс принятия заказов от клиентов, приложение позволяет уменьшить количество обслуживающего персонала, разработка доступна любому пользователю с доступом к сети Интернет.

Список использованной литературы

1. Н.Комолова, Е. Яковлева: HTML[Текст]: Самоучитель. – СПб, 2011 – 288с.
2. Что такое web-приложение и динамические веб-страницы [Электронный ресурс]. – Режим доступа: <https://helpx.adobe.com/ru/dreamweaver/using/web-applications.html>, свободный (23.05.2019)
3. Шилдт, Герберт: С# 4.0 [Текст]: полное руководство.: Пер. с англ. – М, 2011 – 1056с.: ил. — Парал. тит. англ. - ISBN 978-5-8459-1684-6
4. Bootstrap в примерах.[Текст] / Пер. с англ. Рагимов Р. Н. / Науч. ред. Киселев А. Н. – М.: ДМК Пресс, 2017. – 314 с.: ил. - ISBN 978-5-97060-423-6
5. Bootstrap – официальный сайт [Электронный ресурс]. – Режим доступа: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
6. Справочник по С#[Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/>, свободный(23.05.2019)

ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИИ



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ»

Кафедра информационных систем и вычислительной техники

ОТЗЫВ

РУКОВОДИТЕЛЯ О ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студента: Ефимов Григорий Львович, группа: ИАС-15

Тема: *Разработка web- приложения для обслуживания клиентов в сети ресторанов быстрого питания*

Руководитель ВКР Заведующий кафедрой ИСиВТ Мазаков Евгений Борисович

ПОКАЗАТЕЛИ ОЦЕНКИ ВКР

№ п/п	Показатели	Оценка				
		5	4	3	2	0
1	Оригинальность и новизна полученных результатов, научных, конструкторских и технологических решений.		+			
2	Степень самостоятельности и творческого участия студента в работе.	+				
3	Уровень и новизна формулируемых задач исследования или разработки.		+			
4	Корректность использования в работе методов исследования, математического моделирования, инженерных расчётов.		+			
5	Степень комплексности работы. Применение в ней знаний естественнонаучных, социально-экономических, общепрофессиональных и специальных дисциплин.	+				
6	Использование информационных ресурсов Internet.	+				
7	Степень практического использования результатов.	+				
8	Степень полноты обзора состояния вопроса.		+			
9	Использование современных пакетов компьютерных программ и технологий.	+				
10	Наличие публикаций, участие в научно-технических конференциях, награды за участие в конкурсах.		+			
11	Ясность, чёткость, последовательность и обоснованность изложения.		+			
12	Качество оформления ВКР (общий уровень грамотности, стиль изложения, качество иллюстраций, соответствие требованиям ГОСТ к этим документам).		+			
13	Объём и качество выполнения графических материалов, его соответствие тексту записки и стандартам.	+				
ИТОГОВАЯ ОЦЕНКА		хорошо				

Достоинства работы:

Выпускная квалификационная работа посвящена разработке WEB-приложения для обслуживания клиентов сети ресторанов быстрого питания.

В работе проведен анализ бизнес-процессов предприятия, в том числе процессы обслуживания клиентов ресторанов быстрого питания, проанализированы основные проблемы и недостатки существующей системы, предложены пути их решения, проведен выбор средств для реализации проекта.

Результаты предложений реализованы в приложении, написанном на языке C# с использованием технологии ASP.NET архитектуры MVC(Model-View-Controller), спроектирована базы данных и структура клиентской части Web-приложения, что в целом является законченной частью программного средства, которое можно реализовывать на практике.

Кроме этого, с точки зрения экономической эффективности, было определено, что разработка и внедрение web-приложения для обслуживания клиентов позволяет сократить годовые расходы, увеличить скорость обслуживания клиентов, а также улучшить доступность услуг компании для клиентов.

Характеристика деловых качеств:

Студент Ефимов Г.Л. имеет хорошие инженерные знания по профессиональным предметам в области информатики и вычислительной техники, характеризуется положительно, достаточно ответственный, добросовестный, но иногда требует контроля, умеет самостоятельно организовать свою работу. Способен самостоятельно анализировать существующие проблемы и предлагать пути их решения. Умеет работать с технической литературой и Интернет-ресурсами для проведения исследования и разработки программных средств, выбирать средства реализации и доводить предложенные решения до программной реализации.

Характеристика работы над ВКР:

ВКР выполнена в соответствии с заданием, материал пояснительной записки изложен логически стройно, грамотно. Работа оформлена в соответствии с ГОСТ. В работе имеются отдельные стилистические неточности, что в целом не снижает качество самой работы.

Заключение:

В целом выпускная квалификационная работа выполнен на достаточно хорошем уровне, может быть допущена к защите, заслуживает оценки **«хорошо»**, а ее автор Ефимов Г.Л. заслуживает присвоения квалификации бакалавр по направлению «Информатика и вычислительная техника».

«10» июня 2019 г.

Руководитель: заведующий кафедрой ИСиВТ



Е.Б. Мазаков

ПЕРВОЕ ВЫСШЕЕ ТЕХНИЧЕСКОЕ УЧЕБНОЕ ЗАВЕДЕНИЕ РОССИИ



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ ГОРНЫЙ УНИВЕРСИТЕТ

РЕЦЕНЗИЯ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
студента Санкт-Петербургского горного университета

Студент: Ефимов Григорий Львович Группа ИАС-15
(Фамилия И. О.)

Факультет: Экономический
Кафедра: Информационных систем и вычислительной техники
Направление: 09.03.01 Информатика и вычислительная техника
Присваиваемая квалификация: бакалавр
Тема ВКР: Разработка web- приложения для обслуживания клиентов в сети ресторанов быстрого питания

ПОКАЗАТЕЛИ ОЦЕНКИ ВКР

	№ п/п	Показатели	Оценка				
			5	4	3	2	0*
Справочно-информационная	1	Соответствие представленного материала техническому заданию	+				
	2	Раскрытие актуальности тематики работы		+			
	3	Степень полноты обзора состояния вопроса, использование информационных ресурсов		+			
	4	Уровень и новизна постановки задачи исследования или разработки		+			
	5	Корректность использования в работе методов исследования, математического моделирования, инженерных расчетов		+			
	6	Степень комплексности работы. Применение знаний в естественнонаучных, социально-экономических, общепрофессиональных и специальных областях		+			
	7	Использование современных пакетов компьютерных программ и технологий	+				
Творческая	8	Обоснованность и достоверность основных положений и выводов		+			
	9	Оригинальность и новизна полученных результатов, научных, конструкторских и технологических решений		+			
	10	Ясность, чёткость, последовательность и обоснованность изложения	+				
Оформительская	11	Качество оформления ВКР:					
		– общий уровень грамотности		+			
		– стиль изложения		+			
		– качество иллюстраций и графического материала	+				
Итоговая оценка			хорошо				

(*)-не оценивается

Достоинства работы:

Выпускная квалификационная работа посвящена разработке WEB-приложения для обслуживания клиентов сети ресторанов быстрого питания.

В работе проведен анализ бизнес-процессов предприятия и модели «AS-IS», в том числе процессы обслуживания клиентов ресторанов быстрого питания, предложены пути их совершенствования путем внедрения WEB-приложения.

Проведен выбор средств для реализации проекта.

Результаты предложений реализованы в приложении, написанном на языке C# с использованием технологии ASP.NET архитектуры MVC(Model-View-Controller), спроектирована базы данных и структура клиентской части Web-приложения, что в целом является законченной частью программного средства, которое можно реализовывать на практике.

Кроме этого, с точки зрения экономической эффективности, было определено, что разработка и внедрение web-приложения для обслуживания клиентов может существенно сократить годовые расходы.

Недостатки работы:

- нет введения, в котором должны конкретизироваться цели и задачи дипломной работы. Вместо этого конкретизированы задачи для программного решения. Поэтому структурная логика исследований становится ясной только из контекста;
- бизнес- процессы описаны недостаточно подробно: представлена модель «AS-IS» и нет модели «TO-BE», что влияет на понимание роли и места представленного решения автоматизации;
- заявленные задачи оптимизации процессов и экономической эффективности внедрения обоснованы не в полной мере: приведены предполагаемые расчеты минимизации годовых расходов, основанные только на сокращении трудовых ресурсов и не учитывающее, при этом, других факторов (не только возможностей Интернет-технологий) обслуживания клиентов.

Заключение:

В целом выпускная квалификационная работа выполнена на достаточно хорошем уровне, может быть представлена к защите и заслуживает оценки «хорошо», а ее автор Ефимов Г.Л. заслуживает присвоения квалификации бакалавр по направлению 09.03.01 «Информатика и вычислительная техника».

Рецензент, доцент кафедры Безопасности информационных систем
Санкт-Петербургского государственного университета
телекоммуникаций им. М.А. Бонч-Бруевича

А.Н. Кривцов

10 июня 2019 г.
МП

Подпись

начальник отдела кадров - зам. начальника АКУ

/В.В. Новикова/ 10.06.2019 г.



Приложение

TovarsController.cs

```
namespace webserv.Controllers
{
    public class TovarsController : Controller
    {
        // GET: Tovars
        public ActionResult Index()
        {
            return View();
        }
        public int pagesize = 15; // переменная для обозначение количества товаров на странице
        private Models.ShopDBEntities db = new Models.ShopDBEntities();
        private EFTovarsRepository repository;
        public TovarsController(EFTovarsRepository repo)
        {
            repository = repo;
        }
        public TovarsController()
        {
        }
    }

    public ActionResult List(string Type, int page = 1)
    {
        TovarsListViewModel model = new TovarsListViewModel
        {
            Tovars = db.Tovars // обращение к базе данных
                .Where(b => Type == null || b.Type == Type) //сравнение типов для фильтрации
                .OrderBy(tovar => tovar.Type) //упорядочивание
                .Skip((page - 1) * pagesize) //пропуск товаров с предыдущих страниц
                .Take(pagesize), //выборка указанного количества
            PagingInfo = new PagingInfo
            {
                CurrentPage = page,
                ItemsPerPage = pagesize,
                TotalItems = Type == null ? db.Tovars.Count() : db.Tovars.Where ( tovar => tovar.Type == Type).Count()
            },
            CurrentType = Type //фильтрация типов
        };

        return View(model); //возврат модели в пользовательское представление
    }
}
```

```

    }
}

public class PagingInfo
{
    public int TotalItems { get; set; } // общее количество элементов
    public int ItemsPerPage { get; set; } //элементов на странице
    public int CurrentPage { get; set; } //текущая страница
    public int TotalPages
    {
        get { return (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage); }
    }
}

public class TovarsListViewModel
{
    public IEnumerable<Tovar> Tovars { get; set; } //список товаров
    public PagingInfo PagingInfo { get; set; } //ссылки страниц
    public string CurrentType { get; set; } //свойство фильтра по типу товара
}

```

PagingHelper.cs

```

namespace webserv.HtmlHelpers
{
    public static class PagingHelper
    {
        public static MvcHtmlString PageLinks(this HtmlHelper html, PagingInfo paginginfo, Func<int, string> pageUrl)
        {
            StringBuilder result = new StringBuilder();
            for (int i = 1; i <= paginginfo.TotalPages; i++)
            {
                TagBuilder tag = new TagBuilder("a");
                tag.MergeAttribute("href", pageUrl(i));
                tag.InnerHtml = i.ToString();
                if (i == paginginfo.CurrentPage)
                {
                    tag.AddCssClass("selected");
                    tag.AddCssClass("btn-primary");
                }
                tag.AddCssClass("btn btn-default");
                result.Append(tag.ToString());
            }

            return MvcHtmlString.Create(result.ToString());
        }
    }
}

```

Cart.cs

```

public class Cart //класс корзины
{
    private List<CartLine> lineCollection = new List<CartLine>();

    public IEnumerable<CartLine> Lines { get { return lineCollection; } } // свойство для обращение к содержимому списка

    public void Additem(Tovar tovar, int quantity) //метод добавления в корзину
    {
        CartLine line = lineCollection //
            .Where(b => b.Tovar.Id == tovar.Id) //выдача первого соответствия этого товара в списке корзины
            .FirstOrDefault();
        if (line == null) // проверка наличия товара в корзине
        {
            lineCollection.Add(new CartLine { Tovar = tovar, Quantity= quantity }); //добавление, при условии отсутствия
        }
        else
        {
            line.Quantity += quantity; //увеличение количества, при наличии
        }
    }

    public void Removeline(Tovar tovar) //метод удаления товара из корзины
    {
        lineCollection.RemoveAll(l => l.Tovar.Id == tovar.Id);
    }
    public decimal TovarsTotalValue() //подсчет общей суммы
    {
        return lineCollection.Sum(e => e.Tovar.Price * e.Quantity);
    }
    public void ClearCart() //Отчистка корзины
    {
        lineCollection.Clear();
    }
}

public class CartLine
{
    public Tovar Tovar { get; set; }
    public int Quantity { get; set; }
}

```

CartController.cs

```

namespace webserv.Controllers
{
    public class CartController : Controller
    {
        private ITovarRepository repository;
        private Models.ShopDBEntities db = new Models.ShopDBEntities();

        private OrderEnd ord = new OrderEnd();
        public CartController(ITovarRepository repo)
        {
            repository = repo;
        }
        public CartController()
        {
        }

        public ViewResult Index(Cart cart, string returnUrl)
        {
            return View(new CartIndexViewModel
            {
                Cart = cart,
                ReturnUrl = returnUrl
            });
        }

        public RedirectToRouteResult AddToCart(Cart cart, int tovid, string returnUrl, int a)
        {
            Tovar tov = new Tovar();
            tov = db.Tovars
                .FirstOrDefault(b => b.Id == tovid);
            if (tov != null)
            {
                cart.Additem(tov, 1);
            }
        }
    }
}

```



```

        if (a == 1) return RedirectToAction("Index", new { returnUrl });
        else return RedirectToAction("List", "Tovars", new { returnUrl });
    }

    public RedirectToRouteResult RemoveFromCart(Cart cart, int tovid, string returnUrl)
    {
        Tovar tov = new Tovar();
        tov = db.Tovars
            .FirstOrDefault(b => b.Id == tovid);
        if (tov != null)
        {
            cart.Removeline(tov);
        }

        return RedirectToAction("Index", new { returnUrl });
    }
    public PartialViewResult CartSumm(Cart cart)
    {
        return PartialView(cart);
    }

    public PartialViewResult CartinCheck(Cart cart)
    {
        return PartialView("CartinCheck", cart);
    }
    public PartialViewResult buyerlist()
    {
        var item = db.Buyers;
        return PartialView(item);
    }
    public PartialViewResult Tovars(int idzak)
    {
        var item = db.Zakaz_Tabl
            .Where(x => x.Id == idzak);

        return PartialView(item);
    }
    public ViewResult Checkout(Cart cart, ShippingDetails shippingdetails)
    {
        if (cart.Lines.Count() == 0)
        {
            //Обработка пустой корзины
            ModelState.AddModelError("", "Вы пытаетесь заказать пустую корзину!");
        }

        if (ModelState.IsValid)
        {
            //отправка заказа и сброс корзины
            ord.Order(cart, shippingdetails);
            cart.ClearCart();
            return View("Complited");
        }
        else
        {
            return View(new ShippingDetails());
        }
    }
    public ViewResult preorder()
    {
        return View();
    }
}

public class CartIndexViewModel
{
    public Cart Cart { get; set; }
    public string ReturnUrl { get; set; }
}

```

NavController.cs

```
namespace webserv.Controllers
{
    public class NavController : Controller
    {
        private Models.ShopDBEntities db = new Models.ShopDBEntities();
        public PartialViewResult Menu(string types = null)
        {
            ViewBag.SelectedCategory = types;

            IEnumerable<string> Type = db.Tovars
                .OrderBy(x => x.Type)
                .Select(x => x.Type)
                .Distinct().OrderBy(x => x);
            return PartialView(Type);
        }
    }
}
```

CartModelBinder.cs

```
namespace webserv.Infrastructure
{
    public class CartModelBinder: IModelBinder
    {
        private const string sessionkey = "cart";
        public object BindModel(ControllerContext controllerContext, ModelBindingContext bIndingContext)
        {
            Cart cart = null;
            if (controllerContext.HttpContext.Session!=null)
            {
                cart = (Cart)controllerContext.HttpContext.Session[sessionkey];
            }
            if (cart==null)
            {
                cart = new Cart();
                if (controllerContext.HttpContext.Session != null)
                {
                    controllerContext.HttpContext.Session[sessionkey] = cart;
                }
            }
            return cart;
        }
    }
}
```

OrderProces.cs

```
namespace webserv.Concreate
{
    public interface OrderProccess
    {
        void Order(Cart cart, ShippingDetails shippingdetails);
    }

    public class OrderEnd : OrderProccess
    {
        private Models.ShopDBEntities db = new Models.ShopDBEntities();

        public void Order(Cart cart, ShippingDetails shippingdetails)
        {
            int id = db.Buyers.Count() + 1;
            //происходит добавление нового заказа в базу
            foreach(var item in cart.Lines)
            {
                Models.Zakaz_Tabl b = new Models.Zakaz_Tabl
                {
                    Id = id,
                    id_tov = item.Tovar.Id,
                    Kolvo = item.Quantity,
                    price_tov = item.Quantity * item.Tovar.Price
                };
                db.Zakaz_Tabl.Add(b);
                db.SaveChanges();
            }
            Models.Buyer a = new Models.Buyer
            {
                Id=id,
                Name = shippingdetails.Name,
                Adres = shippingdetails.Adres,
                phone = '8'+ shippingdetails.Phone,
                idzak= db.Buyers.Count() + 1,
                totalprice = Convert.ToInt32(cart.TovarsTotalValue())
            };
            db.Buyers.Add(a);
            db.SaveChanges();
        }
    }
}
```

ShippingDetails.cs

```
namespace webserv.Concreate
{
    public class ShippingDetails
    {
        [Required(ErrorMessage = "Укажите ваше имя")]

        public string Name { get; set; }

        [Required(ErrorMessage = "Укажите ваш адрес")]
        public string Adres { get; set; }

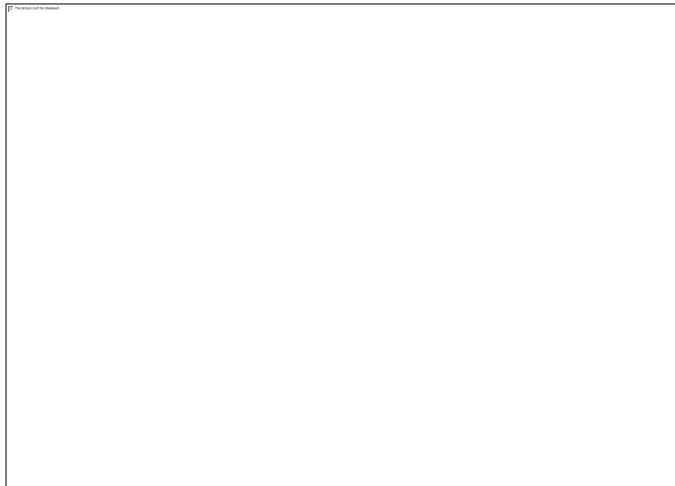
        [Required(ErrorMessage = "Укажите контактный телефон")]
        [MaxLength(10, ErrorMessage = "Номер телефона не может превышать 10 цифр. Введите его без +7")]
        public string Phone { get; set; }
    }
}
```

Buyer.cs

```
namespace webserv.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Buyer
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Adres { get; set; }
        public string phone { get; set; }
        public int idzak { get; set; }
        public int totalprice { get; set; }
    }
}
```

Tovar.cs



Zakaz_Table.cs

```
namespace webserv.Models
{
    using System;
    using System.Collections.Generic;

    public partial class Zakaz_Table
    {
        public int Id { get; set; }
        public int id_tov { get; set; }
        public int Kolvo { get; set; }
        public int price_tov { get; set; }
    }
}
```

AccountViewModel.cs

```
namespace webserv.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }
    }

    public class ExternalLoginListViewModel
    {
        public string ReturnUrl { get; set; }
    }

    public class SendCodeViewModel
    {
        public string SelectedProvider { get; set; }
        public ICollection<System.Web.Mvc.SelectListItem> Providers { get; set; }
        public string ReturnUrl { get; set; }
        public bool RememberMe { get; set; }
    }

    public class VerifyCodeViewModel
    {
        [Required]
        public string Provider { get; set; }

        [Required]
        [Display(Name = "Код")]
        public string Code { get; set; }
        public string ReturnUrl { get; set; }

        [Display(Name = "Запомнить браузер?")]
        public bool RememberBrowser { get; set; }

        public bool RememberMe { get; set; }
    }

    public class ForgotViewModel
    {
        [Required]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }
    }
}
```

```

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Display(Name = "Запомнить меня")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение пароля")]
        [Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ResetPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Адрес электронной почты")]
        public string Email { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "Значение {0} должно содержать не менее {2} символов.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Подтверждение пароля")]
        [Compare("Password", ErrorMessage = "Пароль и его подтверждение не совпадают.")]
        public string ConfirmPassword { get; set; }
    }

    public class ForgotPasswordViewModel
    {
        [Required]
        [EmailAddress]
        [Display(Name = "Почта")]
        public string Email { get; set; }
    }

```

Global.asax.cs

```

namespace webserve
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            ModelBinders.Binders.Add(typeof(Cart), new CartModelBinder());
        }
    }
}

```

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="~/Content/bootstrap.min.css" rel="stylesheet" /> <!-- Элемент стилей -->
<link href="~/Content/bootstrap-grid.css" rel="stylesheet" /> <!-- разметка -->
<script src="~/Scripts/bootstrap.bundle.min.js"></script>

<script src="~/Scripts/jquery-3.0.0.min.js"></script>
<title>@ViewBag.Title- приложение ASP.NET</title>
<link href="~/Content/Site.css" rel="stylesheet" />>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
<style>
    .bd-placeholder-img {
        font-size: 1.125rem;
        text-align: middle;
        -webkit-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
        user-select: none;
    }

</style>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<!-- Favicons -->
<link rel="shortcut icon" href="img/favicon.ico">
<!-- Fontawesome css -->
<link rel="stylesheet" href="css/font-awesome.min.css">
<!-- Ionicons css -->
<link rel="stylesheet" href="css/ionicons.min.css">
<!-- linearicons css -->
<link rel="stylesheet" href="css/linearicons.css">
<!-- Nice select css -->
<link rel="stylesheet" href="css/nice-select.css">
<!-- Jquery fancybox css -->
<link rel="stylesheet" href="css/jquery.fancybox.css">
<!-- Jquery ui price slider css -->
<link rel="stylesheet" href="css/jquery-ui.min.css">
<!-- Meanmenu css -->
<link rel="stylesheet" href="css/meanmenu.min.css">
```

```

<!-- Owl carousel css -->
<link rel="stylesheet" href="css/owl.carousel.min.css">
<!-- Bootstrap css -->
<link rel="stylesheet" href="css/bootstrap.min.css">
<!-- Custom css -->
<link rel="stylesheet" href="css/default.css">
<!-- Main css -->
<link rel="stylesheet" href="style.css">
<!-- Responsive css -->
<link rel="stylesheet" href="css/responsive.css">
<!-- Modernizer js -->
<script src="Scripts/modernizr-3.5.0.min.js"></script>
</head>

<body>
    <header>
        <div class="header-middle ptb-15">
            <div class="container">
                <div class="row align-items-center no-gutters">
                    <div class="col-lg-3 col-md-12">
                        <div class="logo mb-all-30">
                            <a href="#">![logo-image](/img/logo/logo.svg)

```


List.cshtml

```
@using webserv.HtmlHelpers
@model webserv.Controllers.TovarsListViewModel
@{
    ViewBag.Title = "Товары";
}
<div class="album py-5 bg-light">
    <div class="container">
        <div class="row">
            @foreach (var item in Model.Tovars)
            {
                @Html.Partial("_TovSummary", item)
            }
        </div>
    </div>
</div>
<div>
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("List", new { Page = x, Type = Model.CurrentType }))
</div>
```

_TovSummary.cshtml

```
@model webserv.Models.Tovar
<div class="col-md-4">
    <div class="card mb-4 shadow-sm bg-white">
        <svg class="bd-placeholder-img card-img-top" width="100%" height="225"
            xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
            preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder: Thumbnail">
            <image class="img-thumbnail" xlink:href="/img/products/@Model.Preview" width="100%" height="100%"></image>
        </svg>
        <div class="card-body">
            <h3 id="tovname" class="text-justify">@Model.Name</h3>
            <h5 id="tovdesc" class="text-justify">@Model.Description </h5>
            <hr>
            <h4 class="text-right">@Model.Price py6</h4>
            <div id="cartbtn" class="row">
                @using (Html.BeginForm("AddToCart", "Cart", new { tovid = @Model.Id, returnUrl = Request.Url.Query }))
                {
                    <div class="d-flex justify-content-between align-items-center">
                        <div>
                            @Html.Hidden("a", 0)
                            <input type="submit" class="btn btn-sm btn-outline-primary" value="+"/>
                        </div>
                    </div>
                }
                @using (Html.BeginForm("AddToCart", "Cart", new { tovid = @Model.Id }))
                {
                    <div class="d-flex justify-content-between align-items-center">
                        <div>
                            @Html.HiddenFor(x => x.Id)
                            @Html.Hidden("returnUrl", Request.Url.PathAndQuery)
                            @Html.Hidden("a", 1)
                            <input type="submit" class="btn btn-sm btn-outline-secondary" value="Добавить в корзину" />
                        </div>
                    </div>
                }
            </div>
        </div>
    </div>
</div>
```

_LoginPartial.cshtml

```
@using Microsoft.AspNet.Identity
@if (Request.IsAuthenticated)
{
    using (Html.BeginForm("LogOff", "Account", FormMethod.Post,
        new { id = "logoutForm", @class = "navbar-right" }))
    {
        @Html.AntiForgeryToken()

        <ul class="nav navbar-nav navbar-right">
            <li>
                @Html.ActionLink("Здравствуйте, " + User.Identity.GetUserName() + "!",
                    "Index", "Manage", routeValues: null, htmlAttributes: new { title = "Manage" })
            </li>
            <li><a href="javascript:document.getElementById('logoutForm').submit()">Выйти</a></li>
        </ul>
    }
}
else
{
    <ul class="nav navbar-nav navbar-right">
        <li>@Html.ActionLink("Регистрация", "Register", "Account",
            routeValues: null, htmlAttributes: new { id = "registerLink" })</li>
        <li>@Html.ActionLink("Выполнить вход", "Login", "Account",
            routeValues: null, htmlAttributes: new { id = "loginLink" })</li>
    </ul>
}
```

Menu.cshtml

```
@model IEnumerable<string>

@foreach (var link in Model)
{
    <li class="nav-item active">
        <a id="nav" class="nav-pills">
            @Html.RouteLink(link, new
            {
                controller = "Tovars",
                action = "List",
                Type = link,
                page = 1
            }, new
            {
                @class = "btn btn-light btn-block"
            } + (@link == ViewBag.SelectedCategory ? " btn-primary" : ""))
        </a>
    </li>
}
```

Index.cshtml

```
@model webserv.Controllers.CartIndexViewModel
@{
    ViewBag.Title = "Ваша корзина: ";
}
<h2>@ViewBag.Title</h2>
<table class="table table-borderless">
    <thead class="table-bordered border-primary">
        <tr>
            <th class="text-primary">Наименование</th>
            <th class="text-primary">Цена</th>
            <th class="text-primary">Количество</th>
            <th class="text-primary">Сумма</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var line in Model.Cart.Lines)
        {
            <tr>
                <td>@line.Tovar.Name</td>
                <td>@line.Tovar.Price.ToString("# pyб")</td>
                <td>@line.Quantity.ToString("x#")</td>
                <td>@((line.Quantity * line.Tovar.Price).ToString("# pyб"))</td>
                <td>
                    @using (Html.BeginForm("RemoveFromcart", "Cart", new { tovid = @line.Tovar.Id }))
                    {
                        @Html.Hidden("Id", line.Tovar.Id)
                        @Html.HiddenFor(x => x.ReturnUrl)
                        <input type="submit" class="btn btn-sm btn-outline-secondary" value="Удалить" />
                    }
                </td>
            </tr>
        }
    </tbody>
    <tfoot>
        <tr>
            <td>Итого: </td>
            <td></td>
            <td></td>
            <td>@Model.Cart.TovarsTotalValue().ToString("# pyб")</td>
        </tr>
    </tfoot>
</table>

<div class="text-center">
    <a class="btn btn-secondary" href="@Model.ReturnUrl">Продолжить покупку</a>
    @Html.ActionLink("Оформить заказ", "preorder", null, new { @class = "btn btn-primary" })
</div>
```

_icar.cshtml

```
@model webserv.Concreate.CartLine

<li class="list-group-item d-flex justify-content-between lh-condensed">
    <div>
        <h6 class="my-0">@Model.Tovar.Name</h6>
        <small class="text-muted">@Model.Tovar.Price.ToString("# pyб.") x @Model.Quantity</small>
    </div>
    <span class="text-muted">@ (Model.Tovar.Price * Model.Quantity)</span>
</li>
```

Cartincheck.cshtml

```
@model webserv.Concreate.Cart
<body>
    <h4 class="d-flex justify-content-between align-items-center mb-3">
        <span class="text-muted">Ваша корзина</span>
        <span class="badge badge-secondary badge-pill"></span>
    </h4>
    <ul class="list-group mb-3">
        @foreach (var item in Model.Lines)
        {
            @Html.Partial("_icar", item)
        }
        <li class="list-group-item d-flex justify-content-between">
            <span>Итого</span>
            <strong>
                @Model.TovarsTotalValue()
            </strong>
        </li>
    </ul>
</body>
```

CartSum.cshtml

```
@model webserv.Concreate.Cart
<div class="text-md-right">
    <a class="navbar-light">
        <i class="lnr lnr-cart active text-light"></i></a>
        @Html.ActionLink("корзина", "Index", "Cart",
            new { returnUrl = Request.Url.PathAndQuery },
            new { @class = "btn btn-danger" })
    <button type="button" class="btn btn-danger dropdown-toggle"
        data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
        <span class="sr-only">Toggle Dropdown</span>
    </button>
    <div id="cartin" class="dropdown-menu dropdown-menu-md-right col-md-4">
        <a class="dropdown-item">@Html.Action("CartinCheck", "Cart")</a>
    </div>
    <span class="badge badge-pill bg-light align-text-bottom">@Model.Lines.Sum(x => x.Quantity).ToString("x#")</span>
</div>
<div class="navbar-text text-md-right">
    <ul class="nav flex-column text-left">
        <li class="nav-item">
            <a class="navbar-light">@Model.TovarsTotalValue().ToString("# py6") </a>
        </li>
    </ul>
</div>
```

Preorder.cshtml

```
body>
<div class="container">
  <div class="row">
    <div class="col-md-4">
      <div class="card mb-4 shadow-sm">
        <svg class="bd-placeholder-img card-img-top" width="100%" height="225" xmlns="http://www.w3.org/2000/svg"
          preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder:
            Thumbnail"><title>Placeholder</title><rect width="100%"
              height="100%" fill="#55595c"/></rect><text x="50%" y="50%" fill="#eceeef" dy=".3em">Thumbnail</text></svg>
        <div class="card-body">
          <p class="card-text">
            <h3>Улица Пушкина</h3>
            какое-то описание
          </p>
          <div class="d-flex justify-content-between align-items-center">
            <div class="btn-group">
              <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
            </div>
            <small class="text-muted">9 mins</small>
          </div>
        </div>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card mb-4 shadow-sm">
        <svg class="bd-placeholder-img card-img-top" width="100%" height="225" xmlns="http://www.w3.org/2000/svg"
          preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder:
            Thumbnail"><title>Placeholder</title><rect width="100%"
              height="100%" fill="#55595c"/></rect><text x="50%" y="50%" fill="#eceeef" dy=".3em">Thumbnail</text></svg>
        <div class="card-body">
          <p class="card-text">
            <h3>Улица Наличная</h3>
            какое-то описание
          </p>
          <div class="d-flex justify-content-between align-items-center">
            <div class="btn-group">
              <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
            </div>
            <small class="text-muted">9 mins</small>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="col-md-4">
    <div class="card mb-4 shadow-sm">
      <svg class="bd-placeholder-img card-img-top" width="100%" height="225" xmlns="http://www.w3.org/2000/svg"
        preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder:
          Thumbnail"><title>Placeholder</title><rect width="100%"
            height="100%" fill="#55595c"/></rect><text x="50%" y="50%" fill="#eceeef" dy=".3em">Thumbnail</text></svg>
      <div class="card-body">
        <p class="card-text">
          <h3>Улица Пушкина</h3>
          какое-то описание
        </p>
        <div class="d-flex justify-content-between align-items-center">
          <div class="btn-group">
            <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
          </div>
          <small class="text-muted">9 mins</small>
        </div>
      </div>
    </div>
  </div>
  <div class="col-md-4">
    <div class="card mb-4 shadow-sm">
      <svg class="bd-placeholder-img card-img-top" width="100%" height="225" xmlns="http://www.w3.org/2000/svg"
        preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder:
          Thumbnail"><title>Placeholder</title><rect width="100%"
            height="100%" fill="#55595c"/></rect><text x="50%" y="50%" fill="#eceeef" dy=".3em">Thumbnail</text></svg>
      <div class="card-body">
        <p class="card-text">
          <h3>Улица Пушкина</h3>
          какое-то описание
        </p>
        <div class="d-flex justify-content-between align-items-center">
          <div class="btn-group">
            <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
          </div>
          <small class="text-muted">9 mins</small>
        </div>
      </div>
    </div>
  </div>
  <div class="col-md-4">
    <div class="card mb-4 shadow-sm">
      <svg class="bd-placeholder-img card-img-top" width="100%" height="225" xmlns="http://www.w3.org/2000/svg"
        preserveAspectRatio="xMidYMid slice" focusable="false" role="img" aria-label="Placeholder:
          Thumbnail"><title>Placeholder</title><rect width="100%"
            height="100%" fill="#55595c"/></rect><text x="50%" y="50%" fill="#eceeef" dy=".3em">Thumbnail</text></svg>
      <div class="card-body">
        <p class="card-text">
          <h3>Улица Пушкина</h3>
          какое-то описание
        </p>
        <div class="d-flex justify-content-between align-items-center">
          <div class="btn-group">
            <button type="button" class="btn btn-sm btn-outline-secondary">View</button>
          </div>
          <small class="text-muted">9 mins</small>
        </div>
      </div>
    </div>
  </div>

```



```

        <hr class="mb-4">
        <input class="btn btn-primary btn-lg btn-block" type="submit" value="Принять заказ" />
    </form>
</div>
</div>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script>window.jQuery || document.write('<script
src="/docs/4.3/assets/js/vendor/jquery-slim.min.js"></script>')</script>
<script src="/docs/4.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-xrRywqdh3PHs8keKZN+8zzc5TX0GRTLComivcblNJWm2rs5C8PRhcEn3czEjhA09o" crossorigin="anonymous"></script>
<script src="form-validation.js"></script>

```

Complited.cshtml

```

@{
    ViewBag.Title = "Complited";
}
<title>@ViewBag.Title</title>
<head>
    <script type="text/javascript">
        ...
        setTimeout('location.replace("/Home/Index")', 10000);
    </script>
</head>
<h2>Заказ принят</h2>

```

Tovars.cshtml

```

@model IEnumerable<webserv.Models.Zakaz_Tab1>

<ul class="nav flex-column">
    @foreach (var item in Model)
    {
        <li>
            ...
            <a> @item.id_tov</a>
        </li>
    }
</ul>

```

Login.cshtml

```
@using webserv.Models
@model LoginViewModel
@{
    ViewBag.Title = "Выполнить вход";
}
<h2>@ViewBag.Title.</h2>
<div class="row">
    <div class="col-md-8">
        <section id="loginForm">
            @using (Html.BeginForm("Login", "Account",
                new { returnUrl = ViewBag.ReturnUrl }, FormMethod.Post, new { @class = "form-horizontal", role = "form" }))
            {
                @Html.AntiForgeryToken()
                <h4>Используйте локальную учетную запись для входа.</h4>
                <hr />
                @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                <div class="form-group">
                    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })
                        @Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    @Html.LabelFor(m => m.Password, new { @class = "col-md-2 control-label" })
                    <div class="col-md-10">
                        @Html.PasswordFor(m => m.Password, new { @class = "form-control" })
                        @Html.ValidationMessageFor(m => m.Password, "", new { @class = "text-danger" })
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <div class="checkbox">
                            @Html.CheckBoxFor(m => m.RememberMe)
                            @Html.LabelFor(m => m.RememberMe)
                        </div>
                    </div>
                </div>
                <div class="form-group">
                    <div class="col-md-offset-2 col-md-10">
                        <input type="submit" value="Выполнить вход" class="btn btn-default" />
                    </div>
                </div>
            <p>
                @Html.ActionLink("Регистрация нового пользователя", "Register")
            </p>
        </section>
    </div>
    <div class="col-md-4">
        <section id="socialLoginForm">
            @Html.Partial("_ExternalLoginsListPartial", new ExternalLoginListViewModel { returnUrl = ViewBag.ReturnUrl })
        </section>
    </div>
</div>
<section Scripts>
    @Scripts.Render("~/bundles/jqueryval")
</section>
```

Index.cshtml

```
@model webserv.Models.IndexViewModel
@{
    ViewBag.Title = "Управление";
}

<h2>Таблица заказов</h2>

<p class="text-success">@ViewBag.StatusMessage</p>
<div class="row">
    <div class="col-md-4 order-md-2 mb-4 sticky-top">
        <h1> Darov dada</h1>
    </div>
    <div class="col-md-8 order-md-1">
        @Html.Action("buyerlist", "Cart")
    </div>
</div>
```