

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ

РАБОТА БАКАЛАВРА

на тему

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АПТЕЧНОГО СКЛАДА

Студент Зин 14.06.2018 Зиновьев Андрей Владимирович
(подпись, дата)

Руководитель Попов 14.06.18 Попов К.В.
(подпись, дата)

Нормоконтролёр Кучин 17.06.18 Кучин А.В.
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от 25.05.2018

№ 38-09-12)

Заведующий кафедрой

Пашенко Пашенко Д.В.
(подпись)

Работа защищена с отметкой отлично (протокол заседания ГАК от 19.06.18 № 7)

Секретарь ГАК

Гурин Гурин Е.И.
(подпись)

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Д.В. Пашенко
«_____» _____ 2018 г.

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Зиновьев Андрей Владимирович гр. 14ВВ1 факультета вычислительной техники.

2. Тема работы: «Программное обеспечение аптечного склада»

Тема утверждена приказом ПГУ № 398 а/о от "03" мая 2018 г.

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Попов К.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Назначение разработки – возможность управления аптечным складом удаленно.
Исходные данные – техническое задание.

Разработать программное обеспечение, которое должно осуществлять следующие функции: учет товара на складе, учет движение товара, учет аптечных пунктов, формирование заявок на товар. Формирование отчетной документации: отчет по движению товаров, отчет по заказам, сводный отчет по прибыли.

Система должна состоять из: подсистема «Аптечный пункт» и «подсистема Склад».

Также данное программное обеспечение должно обладать приятным и интуитивно понятным для пользователя графическим интерфейсом.

5. Перечень подлежащих разработке вопросов.

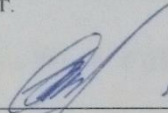
Требуется разработать программное обеспечение подсистем «Аптечного пункта» и «Склада», которые позволят менеджерам Аптечных пунктов оформлять заявки на пополнение товарами их аптеки со склада. Склад, в свою очередь, может оформлять заявки на пополнения склада товарами от поставщиков. Использовать платформу для

разработки – ASP.NET.
б. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	09.05.18	
2.	Выбор технологии для программирования веб-приложения	10	16.05.18	
3.	Проектирование модели «Аптечного склада»	20	16.05.18	
4.	Программирование модуля для администратора «Аптечного склада»	10	23.05.18	
5.	Программирование модуля для менеджеров «Аптечных пунктов»	10	23.05.18	
6.	Отладка веб-приложения	10	31.05.18	
7.	Оформление выпускной работы	30	05.06.18	

Дата выдачи задания "22" февраля 2018 г.

Руководитель бакалаврской работы


(подпись, дата)

14.06.18

Попов К.В.

Задание к исполнению принял студент

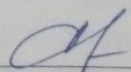


14.06.2018

(подпись, дата)

Зиновьев А.В.

Работу к защите допустить
Декан факультета



15.06.2018

(подпись, дата)

Фионова Л.Р.

РЕФЕРАТ

Бакалаврская работа содержит 84 листа, 19 рисунков, 6 источников, 2 приложения.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, АПТЕЧНЫЙ СКЛАД, ASP.NET, ПРИЛОЖЕНИЕ НА ASP.NET, ВЕБ-ПРИЛОЖЕНИЕ, ДИНАМИЧЕСКИЙ САЙТ, WEB FORMS.

Цель работы – разработать программное обеспечение «Аптечного склада».

Объектом разработки является программный интерфейс для веб-приложения, который позволит менеджерам «Аптечных пунктов» оформлять заявки для доставки товара в аптеку, а администратору совершать заказы на пополнение склада от поставщиков, а также отвечать на заявки и отправлять товары, которые заказали менеджеры.

В результате проведенной работы создано программное обеспечение, удовлетворяющий поставленной задаче. А именно, создано веб-приложение, выполняющее функцию взаимодействия между аптечным пунктом и складом, что позволяет оформлять заявки, как на пополнения склада от поставщиков, так и на пополнения аптечного пункта из склада.

Данный проект рекомендуется использовать на аптечных складах. Проект может быть внедрен во все аптечные пункты и их обеспечивающие склады.

ПГУ 1.090301.10.001 ПЗ				
м.	Лист	№ докум.	Подпись	Дата
Заб.		Зиновьев А.В.		14.08.18
Ювер.		Попов К.В.		14.08.18
Контр.		Кучин А.В.		17.08.18
Верд.		Пашенко Д.В.		

Лит.	Лист	Листов
	4	84

Программное обеспечение
«Аптечного склада»
Пояснительная записка

ФВТ, каф. ВТ
гр. 14ВВ1

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ	8
1.1. Постановка задачи	8
1.2. Выбранный инструмент для решения поставленной задачи.	8
1.3. Почему же ASP.NET работает быстрее?	10
1.4. Почему же был выбран ASP.NET, а не PHP?	13
1.4.1. Общие принципы работы ASP.NET и PHP	14
1.4.2. Языки программирования ASP.NET и PHP	18
1.4.3. Маленькие и большие проекты на ASP.NET и PHP	21
1.4.4. Вывод из сравнения ASP.NET и PHP	22
2. ОПИСАНИЕ МОДЕЛИ АПТЕЧНОГО СКЛАДА, ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И СВЯЗЬ С ТАБЛИЦАМИ БАЗЫ ДАННЫХ	23
2.2. Описание разработанного программного обеспечения	34
2.3. Связь с таблицами базы данных	36
3. РУКОВОДСТВО ПОДСИСТЕМЫ АПТЕЧНОГО ПУНКТА И ПОДСИСТЕМЫ СКЛАД	39
ЗАКЛЮЧЕНИЕ	47
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	48
ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММЫ	49
ПРИЛОЖЕНИЕ В ПРЕЗЕНТАЦИЯ	73

ВВЕДЕНИЕ

Прикладное программное обеспечение (ППО) составляют программы конечного пользователя. Это самый обширный класс программного обеспечения. В настоящее время в большинстве сфер человеческой деятельности разработаны и применяются прикладные программные продукты. Везде, где требуется выполнить большие математические расчеты или производится обработка больших объемов разнообразных данных, или требуется быстрый анализ ситуации с принятием управляющего решения, – компьютеры под управлением прикладного программного обеспечения с успехом заменяют человека.

Прикладное программное обеспечение делятся на 3 категории: ПО общего назначения, ПО специального назначения и ПО профессионального назначения.

С точки зрения данной выпускной квалификационной работы рассмотрим ПО специального назначения. Прикладное программное обеспечение специального назначения

Разработчики создают специальные программные системы целевого назначения для специалистов в некоторой предметной области. Такие программы называют авторскими инструментальными системами.

В ПО специально назначения нас интересуют информационные системы, которые предоставляют широкие возможности:

- в управлении предприятием, например, склад, документооборот офиса (1С: Предприятие);

- в бухгалтерском учете, например, системы, имеющие функции текстовых, табличных редакторов и СУБД. Предназначены для автоматизации подготовки начальных бухгалтерских документов предприятия и их учета, регулярных отчетов по итогам производственной, хозяйственной и финансовой

деятельности в форме, приемлемой для налоговых органов, внебюджетных фондов и органов статистического учета (1С: бухгалтерия);

- в анализе экономической и финансовой деятельности; их используют в банковских и биржевых структурах. Они позволяют контролировать и прогнозировать ситуацию на финансовых, торговых рынках и рынках сырья, выполнять анализ текущих событий, готовить отчеты.

Проектирование информационных систем представляет собой сложную, трудоемкую и длительную работу, требующую высокой квалификации участвующих в ней специалистов.

В данной работе было разработано как раз программное обеспечение прикладного уровня специального назначения, а именно информационная система аптечного склада, которое позволяет решать конечному пользователю буквально в два клика такие задачи, как оформление заказа определенных препаратов с указанием количества по уже указанному адресу, а также отслеживание состояния заказа. А администратору склада также в два действия пополнять склад товарами от поставщиков, следить за прибылью аптечных пунктов и создавать новые аптеки в базе данных.

Таким образом, данное программное обеспечение облегчает жизнь администратору склада и менеджерам аптечных пунктов, которым без этого ПО, необходимо было бы собрать огромное количество бумаг, связаться с поставщиками, проверить наличие товара на складе, возможность доставки и следить за прибылью. А это означает трату колоссального количества времени, сил и энергии, что мешает отличному выполнению поставленной задачи, а также быстрой доставки конкретного препарата, который может спасти кому-то жизнь.

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

1.1. Постановка задачи

Требуется разработать программное обеспечение, которое должно осуществлять следующие функции:

- Учет товара на складе
- Учет движение товара
- Учет аптечных пунктов
- Формирование заявок на товар
- Формирование отчетной документации:
 - 1) Отчет по движению товаров
 - 2) Отчет по заказам
 - 3) Сводный отчет по прибыли

Система должна состоять из:

- Подсистема Аптечный пункт
- Подсистема Склад.

Также данное программное обеспечение должно обладать приятным и интуитивно понятным для пользователя графическим интерфейсом.

1.2. Выбранный инструмент для решения поставленной задачи.

В связи с поставленной задачей был использован такой инструмент программирования веб-сайтов, как ASP.NET.

Начало истории появления было положено с выходом первоначальной версии .NET Framework примерно десяти-одиннадцати лет назад в области программного обеспечения. Мотивированными наилучшими возможностями и способностями Java, COM и веб-технологий и наученные на ошибках и пределах

прежних технологий, программистами и разработчиками в Microsoft было принято решение полностью обновить свою платформу для разработки.

И благодаря этому появилось множество удивительных и совершенных технологий для выполнения таким простых вещей как построения приложений Windows и заканчивая выполнением сложных запросов в базах данных, и специально направленных на разработку веб-сайтов платформа под названием ASP.NET.

В настоящий момент ASP.NET пользуется огромной популярностью, но сильно революционной технологией более не является. Стоит заметить, что базовые функциональные способности, которые лежат в основе ASP.NET, к изумлению, остаются такими же, как и десять лет назад, разработчики Microsoft добавили и будут добавлять к ним ещё множество дополнительных средств и абстракций кодирования более высокого уровня.

Обратим внимание, что в конкуренцию стандартному и традиционному программированию с использованием ASP.NET появилось как минимум одно новое направление, получившее название ASP.NET Model-View-Controller.

В момент первого выпуска платформы ASP.NET, от предшествующих её продуктов Microsoft и платформ, составляющих ей конкуренцию, её отличали следующие особенности: как описывалось выше, разработчики Microsoft полностью перестроили ASP.NET, базируясь на Common Language Runtime (CLR), которая является базой всех приложений фреймворка Microsoft .NET. Благодаря чему, программисты могут писать код для ASP.NET, пользуясь практически всеми языками программирования, которые входят в комплект .NET Framework (а это: C#, JScript.NET, Visual Basic.NET). Одно из главных преимуществ платформы ASP.NET является скорость по сравнению со скриптовыми технологиями, поскольку при первом обращении код компилируется и хранится в специальном кэш, благодаря чему только выполняется, не требуя затрат времени на оптимизацию, парсинг, и т.д.

Так как программное обеспечение аптечного склада создавалось с нуля и прочитав о возможных аналогах, таких как PHP или фреймворк Ruby on Rail, было выявлено, что лучшим инструментом будет выбранная технология. Так как, во-первых, мы можем использовать язык, который изучали на курсах в университете, такой как C#. Во-вторых, нам нужна быстрота исполнения нашего веб-приложения, а ASP.NET по сравнению с аналогами компилирует код, поэтому он исполняется намного быстрее.

1.3. Почему же ASP.NET работает быстрее?

Подобно всем приложениям .NET, приложения ASP.NET всегда компилируются. На самом деле выполнение кода на C# или Visual Basic без предварительной компиляции просто невозможно.

Приложения ASP.NET в действительности проходят через два этапа компиляции.

Главной причиной, по которой .NET могут использовать самые разные языки, как раз является первый этап, в котором написанный код на C# компилируется в код на, так называемом, промежуточном языке, который имеет такое название, как MSIL (Microsoft Intermediate Language – промежуточный язык Microsoft), или просто IL.

Это возможно благодаря тому, что все языки .NET (в этом числе и C#, и Visual Basic и многие другие), по сути, компилируется в почти идентичный код MSIL. Выше описанный этап компиляции может осуществляться как автоматически при первоначальном запросе страницы, так и выполняться заранее (так называемая предварительная компиляция). После этого этапа образуется скомпилированный файл с кодом на MSIL, который представляет собой сборку.

На втором этапе код IL компилируется в код на низкоуровневом машинном языке, этот этап компиляции осуществляется непосредственно перед фактическим выполнением страницы. Этот этап называется, как оперативная компиляция (JIT - Just-In-Time) и выглядят одинаково для абсолютно всех приложений, разработанных с помощью фреймворка .NET.

					ПГУ 1.090301.10.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

На следующем рисунке показан этот состоящий из двух этапов процесс компиляции:

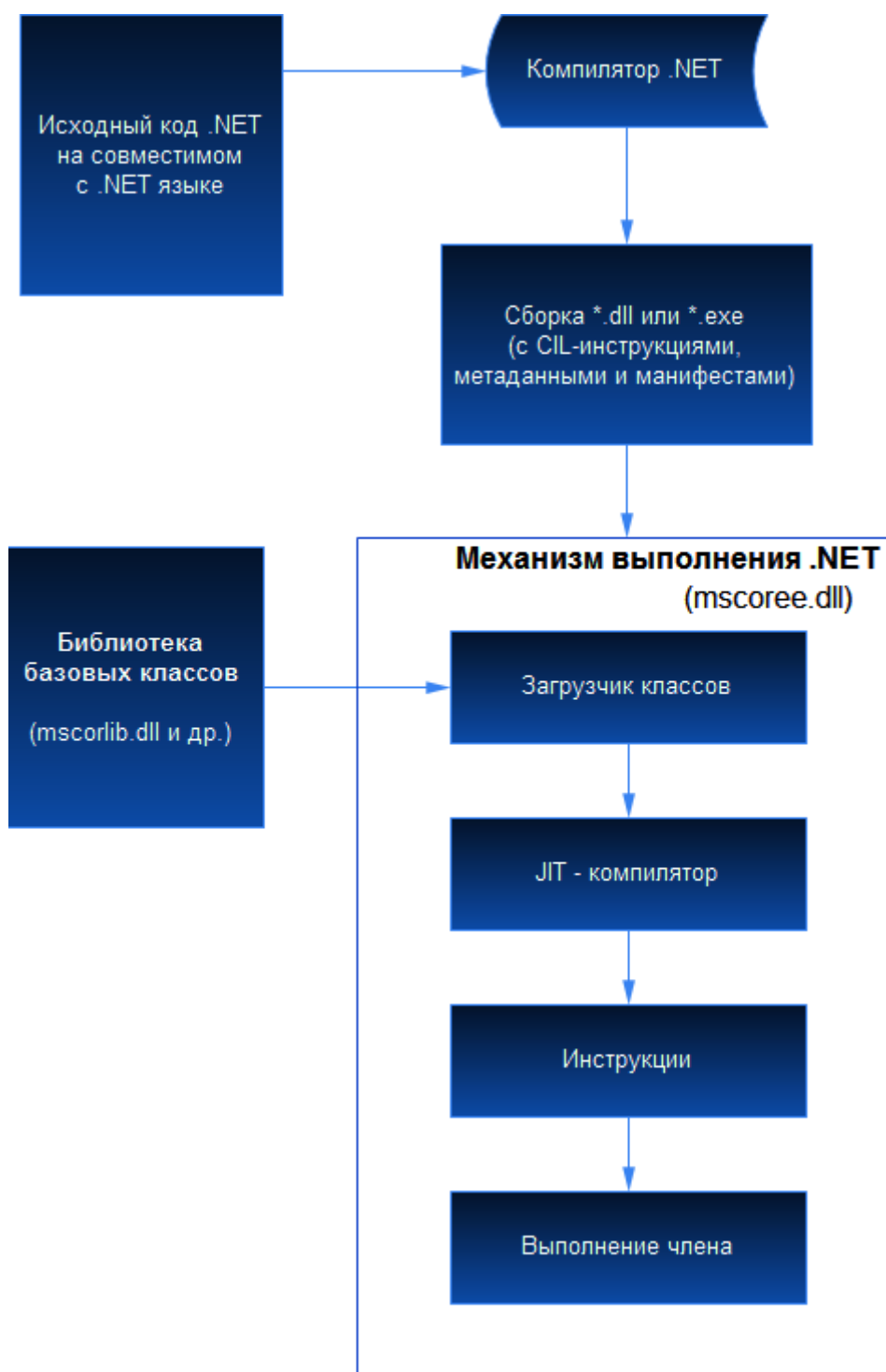


Рисунок 1 – процесс компиляции

Именно для предоставления большего удобства и мобильности программистам процесс компиляции в .NET разделен на два этапа. Сначала, компилятору необходимо знать, в какой операционной системе и на каком базовом оборудовании будет исполняться приложение (к примеру, будь то 32-х или 64-х разрядная ОС Windows), только потом создается код на низкоуровневом

Изм.	Лист	№ докум.	Подпись	Дата

машинном языке. И благодаря исполнению этих двух этапов компиляции, можно создать скомпилированное приложение с кодом .NET и все также распространять её среди более, чем одной платформы.

Конечно, этап оперативной компиляции, к слову, не был настолько продуктивен, если бы его нужно было бы выполнять при абсолютно каждом запросе какой-либо страницы веб-сайта. К радости программистом, веб-приложениям, написанных на ASP.NET, нет необходимости выполнять компиляцию при каждом запрашивании веб-страницы. Вместо этого, код на MSIL в них создается лишь один раз и компилируется заново лишь только тогда, когда происходит изменения исходного кода, а файлы, которые содержат машинный код, кэшируются в системном каталоге, доступ к которому можно осуществить, перейдя в папку, путь которого выглядит примерно так:

C:\Windows\Microsoft.NET\Framework\[Номер версии]\Temporary ASP.NET Files.

Тот момент, когда код компилируется MSIL, зависит только от того, каким способом создается и разворачивается веб-приложение. При создании веб-приложения в виде веб-проекта в Visual Studio код компилируется в IL, к примеру, при компиляции проекта. А в случае, когда веб-приложение выглядит в виде облегченного веб-сайта без проекта, для каждой страницы код компилируется при первом запросе конкретной ему страницы. Но, стоит отметить, что и в том, и другом случае из IL в машинный код, второй этап компиляции, код проходит лишь только при первом его запуске.

Отметим, что в ASP.NET включены также и инструменты, которые позволяют выполнить предварительную компиляцию, с помощью которых можно сделать так, чтобы веб-приложение сразу же компилировалось в машинный код, прямо после его разворачивания на производственном веб-сервере. Благодаря этому можно избежать накладных расходов, которые связаны с выполнением первого этапа компиляции при разворачивании готового приложения. Также исключает возможности для подделки или изменения кода другими людьми.

1.4. Почему же был выбран ASP.NET, а не PHP?

Многие web-программисты для создания своих проектов повсеместно используют PHP. Это объясняется довольно просто: поддержка множеством хостингов, легкость базового изучения, возможность что-то быстро подправить, огромное количество специалистов, и, абсолютно точно, крупнейший выбор готовых CMS. Например, огромная часть веб-программистов используют для большинства своих проектов WordPress, который как раз-таки и написан на PHP.

Но, стоит заметить, что если же создаете движок с нуля, а не используете уже готовое решение, как, например, данное программное обеспечение, то многие преимущества PHP отходят на второй план. Для того, чтобы разобраться с этим, необходимо рассмотреть эту тему подробнее.

Преимущества PHP

1. Простота. Но это же и является его минусом, потому что при разработке сложного программного обеспечения это преимущество обращается в недостаток, так как неполная реализации объектности, отсутствие строгой типизации и т.д., которые присущи полноценным языкам, приводит к тому, что начинаются проблемы в развитии и поддержки сложной системы.

2. Большое количество хостингов. Но если мы разрабатываем веб-приложение с нуля, то стоимость хостинга уходит на второй план в сравнении со стоимостью разработки.

3. Абсолютно бесплатная платформа и вообще всё. Огромный плюс, особенно если вы разрабатываете масштабируемую систему, которая позже будет работать на десятке, а то и сотни серверах. А это возможность сэкономить множество денежных средств, если использовать бесплатную ОС. Но если же веб-приложению достаточно одного сервера, то существенно сэкономить уже не получится. Так же относится и к виртуальному хостингу.

Преимущества ASP.NET

1. Полноценный язык программирования. Кроме того, есть возможность использовать абсолютно любой .NET-совместный язык, благодаря одноименной платформе (к примеру, C# и VB.NET, а также Python, Delphi и множество других).

Отметим, что имеется удобный объектно-ориентированный синтаксис базовых библиотек.

2. Компиляция программ. Как уже говорилось выше, в .NET код компилируется, в отличие от PHP, из-за этого код выполняется намного быстрее, а это несомненно очень важно в больших проектах.

3. Идентичность среды разработки и деплоя. К примеру, если программист использует Windows, то при разработке веб-приложения на PHP, ваша среда будет отличаться от серверной, потому что там стоит Linux. А вот при разработке на ASP.NET используется одинаковая среда как на девелоперских компьютерах, так и на серверах в Интернете.

4. Полная поддержка юникода. Абсолютно все сайты, написанные на ASP.NET, всегда поддерживают UTF-8, чего не скажешь о веб-приложениях на PHP.

5. Имеются огромные возможности, которые включены в состав стандартных библиотек. А это очень важно. Так, например, в PHP многие аналогичные вещи реализуются лишь сторонними библиотеками, а это чревато таким проблемам, как: отсутствия таких же библиотек на хостинги и необходимость изучения.

6. MS SQL против MySQL. Программисты, которые пишут на PHP используют в большинстве своём MySQL, в отличие от ASP.NET программистов, которые выбирают MS SQL. Отметим, что последний обладает куда более широкими возможностями, чем MySQL.

По правде говоря, непонятно как на PHP создаются такие сложные системы, как, например, WordPress. На ASP.NET все делается куда более проще и понятнее.

1.4.1. Общие принципы работы ASP.NET и PHP

PHP

В чистом виде язык PHP позволяет динамически выводить HTML-разметку и иную информацию (к примеру, заголовки для Cookie и т.п.)

Например:

```
<?php
print "<p>Hello, world!</p>";
?>
```

Так же возможно вставить PHP-код в обычную статическую HTML-разметку:

```
<body>
<table>
<tr><td>
<?php
print "I am content!";
?>
</td></tr>
</table>
</body>
```

В следствие этого, программист сам заботится о всей формируемой разметке.

ASP.Net Webforms

Принцип работы в ASP.Net совершенно иной, в отличии от PHP: каждая страница состоит представляет собой два файла: файл с HTML-разметкой и ASP.NET-контроллерами, файл (такой файл обладает расширением aspx) и файла с серверной логикой (расширение vb или cs). Суть заключается в следующем: в файле разметки фактически описание только внешнего вида страница, в него могут быть вписаны также некоторые скрипты, написанные на javascript, к примеру. В файле логики – только логика, то есть программный код. Стоит заметить, как я уже упомянул выше, можно добавлять программный код в файл разметки, и в файл логики добавлять HTML-разметку (по сути, использовать подход, который использовался в PHP), поступать так, разумеется, не рекомендуется, потому что нарушается общая идея, а именно, отделение внешнего вида страницы от логики.

Например, так выглядит файл с разметкой:

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs"
    Inherits="WebApplication1._Default" %>
DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="MyLabel" runat="server"
Text="">asp:Label<br />
            <asp:Button ID="MyButton" runat="server"
Text="Нажми меня!

"OnClick="MyButton_Click" />
        </div>
    </form>
</body>
</html>

```

Соответствующий ей файл с бизнес-логикой:

```

namespace WebApplication1
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void MyButton_Click(object sender, EventArgs e)
        {
            MyLabel.Text = "Кнопку нажали";
        }
    }
}

```

Обратим внимание, что на странице имеются теги, которые не свойственны HTML. Это – так называемы контролы ASP.NET. При компиляции они преобразуются в обычную HTML-разметку. К примеру, asp:CheckBox или asp:TextBox будут преобразованы в <input type = “text” />. Asp:Hyperling – в и т.д. Каждый серверный контрол, разумеется, обладает несколькими свойствами, например, стилями, содержимым и т.п.

Конечно, все серверные контролы, которые были созданы на странице доступны в файле с бизнес-логикой. Например, выше описано, как в

программном плане происходить доступ к элементу MyLabel и задается его содержимое. Как говорилось ранее, это происходит на этапе компиляции страницы и уже к пользователю в браузере приходит готовая страница с HTML-разметкой.

Сравнение

По объему написанного кода для каждой из технологий уже видно, что разобраться в принципах работы PHP гораздо проще, нежели в ASP.NET. Кажется, это одна из главных причин, из-за чего PHP так популярен и распространён среди разработчиков. «Порог вхождения» у PHP значительно ниже, и новичку значительно легче понять принципы его работы, чем сложную структуру ASP.NET. И, стоит отметить, что одновременно в данной ситуации для PHP это является и недостатком, и достоинством.

Достоинство – это простота изучения технологии, но она же одновременно является и недостатком, что приводит к тому, что низкий уровень программистом в PHP встречается значительно чаще, чем в ASP.NET. К примеру, если программист недостаточно опытен или его знания находятся на низком уровне – то язык PHP он все равно освоит, нежели при освоении ASP.Net, в процессе которого у него скорее всего возникнут трудности. А это означает то, что, обращаясь к ASP.Net-разработчику вы понижаете шанс получить низкокачественный продукт, чем ежели вы обратитесь к PHP-разработчику.

К существенным достоинствам PHP можно отнести его «низкоуровневость». ASP.Net – это попытка разработчиками из Microsoft применить при разработке Web-сайтов такие же принципы, как и при разработке Windows приложений, в отличии от того же PHP. К примеру, на странице создается кнопка, к ней пишется обработчик событий, который при нажатии на неё, например, изменит цвет текста у кого-нибудь контроля или загрузит картинку на задний фон, и тому подобное. Но такая модель, стоит отметить, не совсем применима к проектированию веб-приложений, потому что протокол HTTP – это протокол сеанса, это значит, что он работает по принципу запрос-ответ. А приложения на Windows работают постоянно и если провести аналогию,

то работает, используя постоянное соединения. По этой причине для реализации такого же подхода было необходимо применить большее количество абстракций и приемов. Все эти методы были призваны для того, чтобы скрыть «сеансовость» протокола HTTP. Их особенностью является то, что процесс разработки веб-приложения значительно ускоряется. Вероятно, если начинающий программист начнет изучение веб-разработки с ASP.NET, то, весьма вероятно, что он не сразу сможет понять истинные принципы работы веб-приложений. К примеру, необходимо получить от пользователя сайта какие-либо данные и обработать их, к примеру, считать или записать в базу данных. PHP-разработчику будет необходимо создать html-форму, скрипт, который получит все необходимые данные через форму, и вернет посетителю ответ, в виде html-разметки, именно так и работают все веб-приложения: запрос-ответ. Другое дело, если такую же задачу решает ASP.Net-программист, то форма за него уже сделана, он лишь добавит поле для ввода и кнопку, на которую будет прикреплен обработчик события, в котором выполнятся, при нажатии, нужные действия по обработке введенных данных.

Это очень напоминает то, как бы задача была написана для Windows-приложений. И начинающий программист, скорее всего, и не будет знать, что на самом деле при нажатии на кнопку случится отправка данных с формы, которые будут автоматически обработаны, после чего выполнится его логика, которая помещена в обработчики события, а затем так же автоматически сформируется ответная HTML-разметка, которую вскоре получит пользователь.

Другими словами, если начать изучения сферы веб-проектирования с PHP разработчик раньше поймет и увидит реальные принципы работы Web-приложений, нежели программист ASP.Net, от которого эти принципы пытаются спрятать.

1.4.2. Языки программирования ASP.NET и PHP

PHP

PHP – это и есть сам по себе язык программирования, в котором используется си-подобный синтаксис, отсутствует строгая типизация переменных, а в 5ой

версии гораздо лучше работает поддержка объектно-ориентированного программирования.

ASP.Net

При разработке веб-сайтов на ASP.Net, как уже говорилось выше, можно использовать любой язык, который входит в состав .Net такие, как: C# или Visual Basic.Net. Доступно множество библиотек из .Net Framework.

Сравнение

Здесь, абсолютно точно, с большим перевесом выигрывает ASP.Net.

Во-первых, по способностям и возможностям PHP очень далеко от .Net-языков. Например, объектно-ориентированное программирование, хотя, как было отмечено выше, в PHP5 было улучшено, но PHP по-прежнему проигрывает .Net языкам, где они (по крайней мере, C# и Visual Basic.Net) являются полностью объектными. В PHP нельзя просто создать функцию, а только лишь метод класса, также нельзя просто описать переменную – только атрибут класса и т.д. Также в PHP не имеются перечисления (enum), нет событие, а так называемое «ручное создание» нельзя считать событиями, потому что по сути встроенной возможности разработки с использованием модели событий нет. Не имеется возможности перегрузить методы, так как нет типизации в языке. И это лишь несколько неприятных особенностей, с которыми придется столкнуться программистам при разработке на PHP, если они раньше серьезно использовали объектно-ориентированные языки. В PHP нет многопоточности, и некоторые проблемы с поддержкой Unicode. Да, будет честным сказать, что она есть, но абсолютна неудобна для разработчика и эту проблему обещают исправить в PHP6.

Во-вторых, в отличие от .Net-языков, PHP гораздо менее строгий язык. Это обусловлено, в первую очередь, из-за отсутствия строгой типизации. Поэтому переменные объявлять необязательно. А это приводит к тому, что множество опаснейших конструкции, которые не скомпилируются даже в компиляторе Си, в PHP будут исполнены, конечно, с непредсказуемым результатом. Некоторые из ошибок, те же неинициализированные переменные, можно наблюдать, если

включить режим отображения всех ошибок, но совершенно точно стоит об этом знать и помнить. Если начинающий программист начинает изучать программирование с PHP, то далеко не факт, что он будет информирован обо всех тонкостях и необходимости той же «дополнительной настройки». Также начинающему разработчику может просто надоесть постоянные сообщения об ошибках, и он их просто на просто отключает. И, возможно, сайт в таких вот случаях работает, и может быть даже правильно, но рано или поздно, пропущенная ошибка, может привести к серьёзным сбоям в работе сайта или же взломе, мало из бед - это конечно некорректность работы. В .Net-языках же такое не произойдет, пока не будет исправлена ошибка, потому что программа просто не скомпилируется и всё.

Более того, в PHP нет исключения, которые выкидываются самим интерпретатором. Например, если в .Net вы попытаетесь обратиться к объекту, который не создали, т.е. ссылка на него есть, а вызвать new забыли, то у вас произойдет исключение. В PHP только лишь вы увидите предупреждение, и то, если не забыли настроить их отображение. Главное отличие заключается в том, что исключения в .NET вы не пропустите никак – вместо страницы вы получите сообщение об ошибке, где она возникла и что она собой представляет. В PHP же к исходной странице просто будет добавлено где-нибудь сообщение с предупреждением. Да и его вообще можно легко не заметить, к примеру, если сообщение появится в малозаметном месте, или же где-нибудь внизу страницы, куда вы редко заглядываете.

Если даже PHP-разработчик опытен и знает обо всех тонкостях, которые были описаны выше, все равно существует человеческий фактор. Например, расстался разработчик вчера с девушкой, но работать надо. Однако во время работы программист не сконцентрирован на программе, мысли совсем о другом – повышается вероятность того, что он забудет где-то что-то проверить. Во многих случаях это может остаться незамеченным – интерпретация в PHP произойдет успешно. В ASP.Net же ошибку никак не пропустить, потому что она либо не пройдет этап компиляции, либо произойдет исключение.

PHP даёт программисту огромное количество вольностей, которые потенциально опасны, что рекомендуется не начинать изучения программирования, именно программирования, а не веб-проектирования, с PHP. Стоит сначала начать с более серьезного типизированного языка, к примеру Си, а только лишь потом перейти на PHP, имея уже полезные привычки такие, как: всегда инициализировать и объявлять переменные, не игнорировать предупреждение интерпретатора и т.д.

То, что нужно держать в уме, когда программируешь на PHP, в .Net языках за вас проверяет компилятор и пока ошибка не будет исправлена – программа просто не скомпилируется.

Затронем вопросы производительности. В ASP.NET исходных код компилируется лишь один раз в исполняемый байт-код. В PHP же интерпретация выполняется каждый раз. Исходя из этого, ASP.Net должен работать быстрее, но если затронуть реальные тесты, то они показывают примерно равную производительность. В некоторых случаях какая-то из платформ вырывается вперед, но в большинстве своем, можно сказать, что технологии примерно равны. Не стоит забывать, что время генерации страницы не всегда зависит только лишь от языка программирования. Потому что это ещё и время выполнения запросов в базу данных, которое по правде говоря может быть на много больше времени интерпретации.

1.4.3. Маленькие и большие проекты на ASP.NET и PHP

Обычно принято считать, что PHP подходит больше для разработки маленьких проектов, а ASP.Net – для больших веб-приложений. Однако, это неправильно думать так. Как известно, под PHP существует большое количество шаблонизаторов и фреймворков, к примеру, Zend Framework, Smarty и множество других, которые приближают PHP к ASP.Net с точки зрения архитектуры веб-сайта, в том смысле, что происходит отделение программного кода от HTML-разметы, имеется наличие ASP.Net-контролов и т.д. И примеров огромных проектов, которые написаны на PHP, достаточно: vk.com, photofile.ru

и другие. И в то же время, на ASP.Net можно с легкостью создать простой веб-сайт, может быть даже быстрее, чем на PHP.

К крупнейшему преимуществу ASP.Net перед PHP отнесем то, что он использует общую платформу .Net. Что, во-первых, означает, что если вы – разработчик .Net, то не нужно изучать новый язык и новые библиотеки. А, во-вторых, этим открывается широкий простор для интеграции Windows и ASP.Net-приложений. К примеру, класс, который вы ранее написали для настольного приложения можно абсолютно без исправлений использовать и в веб-проекте на ASP.Net. PHP же используют только для разработки веб-страниц. С его помощью можно написать скрипт для командой строки, но такое возникает довольно редко.

1.4.4. Вывод из сравнения ASP.NET и PHP

Какой же инструмент разработки программного обеспечения в связи с исходными требованиями будет наиболее лучшим и продуктивным. Стараясь составить сравнения максимально объективно, подведем итог. Создавать проекты на .Net получается быстрее, с меньшим количеством ошибок, и работа в Visual Studio гораздо более удобна, чем в PHP-редакторе. И создавать программное обеспечение, тем более огромное на платформе .Net в разы лучше, чем в PHP. Именно поэтому был выбран такой язык программирования как ASP.NET.

2. ОПИСАНИЕ МОДЕЛИ АПТЕЧНОГО СКЛАДА, ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И СВЯЗЬ С ТАБЛИЦАМИ БАЗЫ ДАННЫХ

2.1. Описание модели аптечного склада и пользовательского интерфейса

В данном проекте реализован лишь один из модулей огромного проекта аптечного склада, который представляет собой два подмодуля:

- Аптечный пункт
- Склад

Стоит заметить, что реализация модуля работы с поставщиком и модуля оплаты товаров не реализована в данном программном обеспечении, они будут добавлены позже.

Модель сайта «Программное обеспечение Аптечного склада» предполагает наличие следующих разделов (компонентов):

«Страница авторизации», где пользователь получает возможность войти на сайт с выделенными ему полномочиями (администратор склада либо менеджер аптеки).

Последующие разделы будут зависеть от того, с правами какого пользователя вы вошли на сайт, поэтому разделим следующие страницы на две группы:

Администратор:

1. «Управление заказами», раздел, где администратор может увидеть количество заказчиков, информацию по ним, какое количество заказов было сделано и на какую стоимость.

2. «Персональные заказы», на этой странице есть возможность просмотреть и, при необходимости, обработать заказы у выбранного заказчика.

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

3. «Управление пользователями», здесь администратор может обновить данные о менеджере определенной аптеки (предположим, если менеджер сменился) и вовсе удалить пользователя (например, если аптека закрылась).

4. «Редактирование препаратов», в этом разделе можно изменить данные об уже существующих препаратах (например, цену или количество), удалить наименование (если в выбранном лекарстве аптеки более не нуждаются), а также добавить новый препарат.

5. «Создание франшизы аптеки», страница для добавления пользователя отвечающего за новую аптеку, которая будет заказывать препараты у нашего склада. Здесь мы вводим все данные, необходимые для обслуживания новой франшизы (Логин менеджера, имя и фамилия менеджера, адрес аптеки, город, так же мы сами задаем пароль новому пользователю).

На основе изложенного выше можно составить UML-диаграммы вариантов использования для администратора:

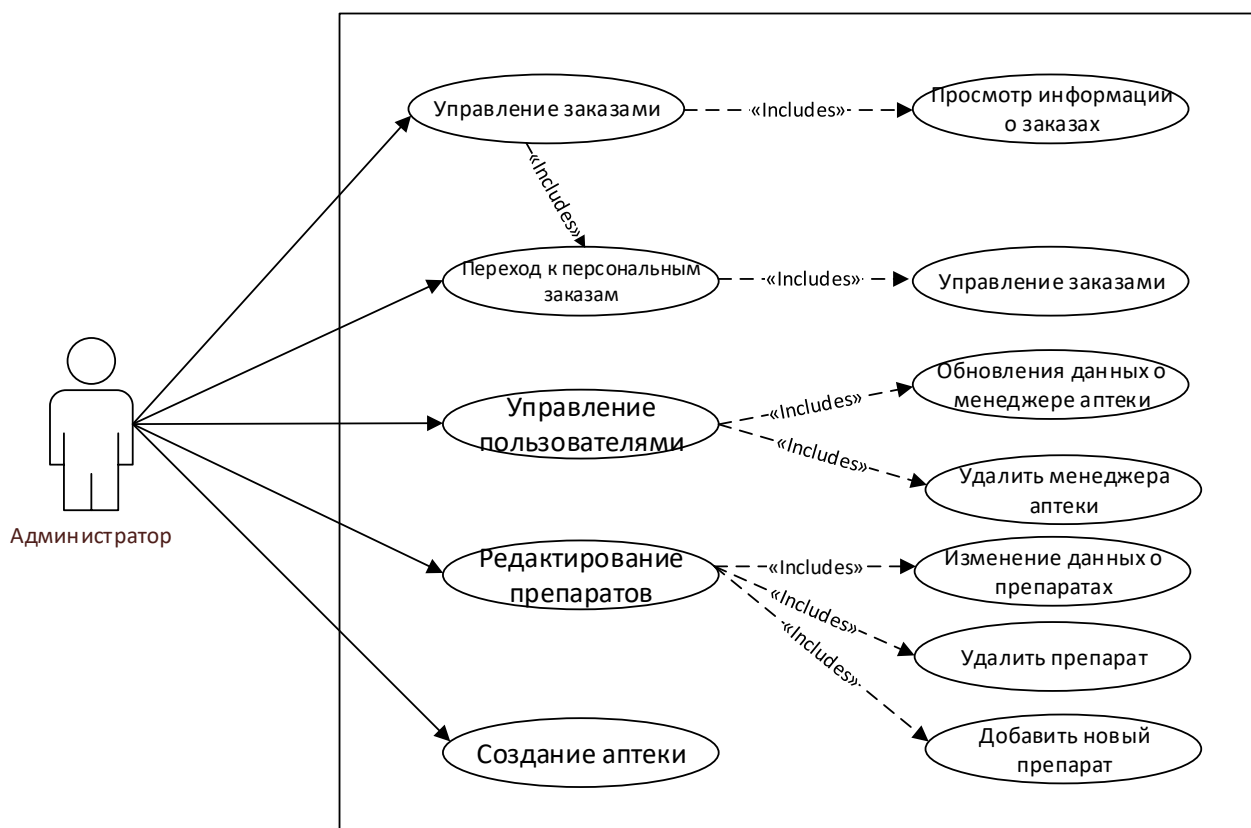


Рисунок 2 - UML-диаграмма вариантов использования для администратора.

Менеджер аптеки:

1. «Список товаров», в этом разделе заказчик может просмотреть всю продукцию, которую он может заказать, а также перейти на страницу с подробным описанием интересующего его товара. На этой странице присутствует фильтр продукции по категориям (Все товары, мочеполовая система, неврология, обезболивающее, офтальмология, спортивное питание).

2. «Описание товара», здесь располагается подробная информация о конкретном товаре, выбранном пользователем (категория, описание, кол-во на складе). Так же менеджер аптеки на этой странице может добавить товар в корзину для дальнейшего его заказа.

3. «Корзина», этот компонент предоставляет пользователю возможность просмотреть все выбранные им для заказа товары, а также, при необходимости, удалить ненужные наименования. На этой странице пользователь имеет возможность либо перейти на странице «Список товаров» (с помощью кнопки «Продолжить покупки»), либо перейти на страницу заказа всех товаров, расположенных в корзине.

4. «Оформить заказ», раздел для непосредственно заказа продукции. Здесь пользователь может убедиться, что данные, по которым заказ должен быть отправлен, верны и совершить покупку.

5. «История заказов», в этом разделе менеджер может увидеть информацию о своих предыдущих заказах, к тому же здесь можно отслеживать статус своих текущих заказов (отправлен заказ или же он еще в обработке).

С помощью всех этих компонентов реализована связь между несколькими аптечными пунктами и складом препаратов.

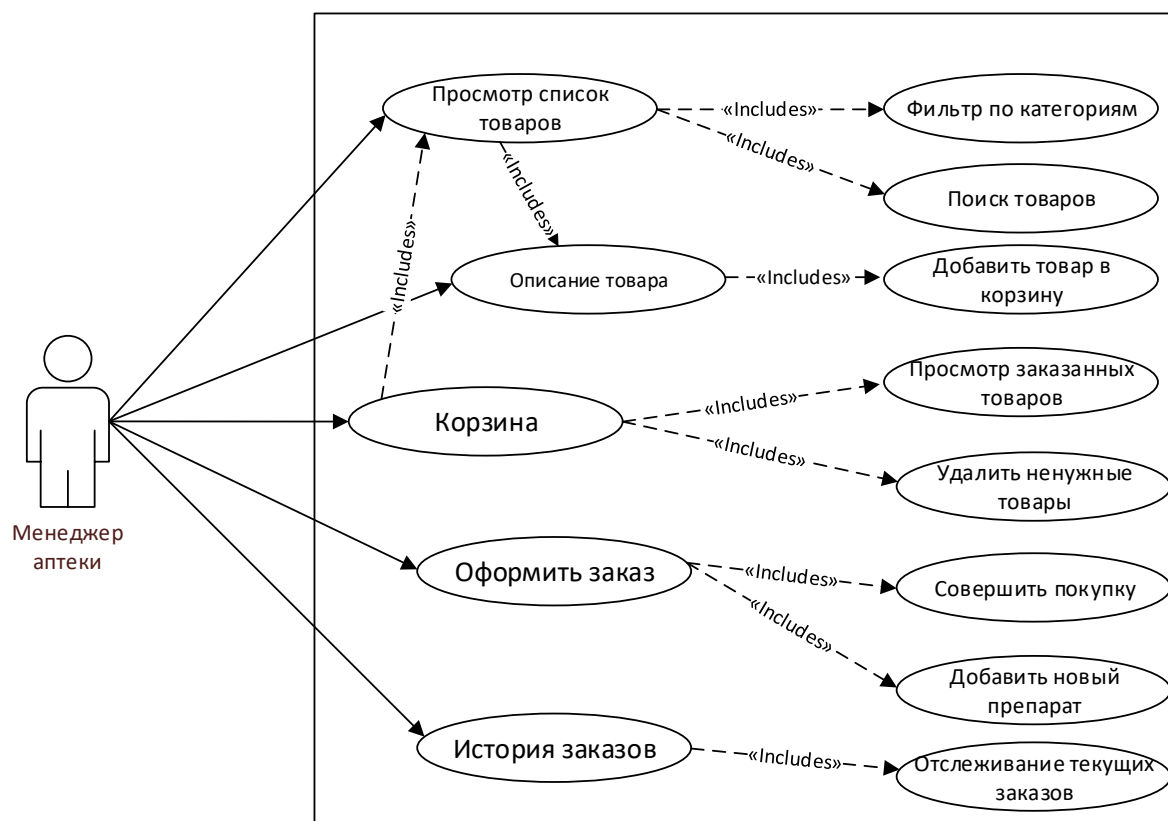


Рисунок 3 - UML-диаграмма вариантов использования для менеджера аптеки.

Как уже было сказано выше, программное обеспечение состоит из страницы входа (Login.aspx), который включает в себя файл Login.aspx.cs, где находится функция авторизации пользователя, которая выглядит следующим образом:

```

protected void SignIn(object sender, EventArgs e)
{
    string name = Request.Form["uname"];
    string password = Request.Form["psw"];

    var userStore = new UserStore<IdentityUser>();
    var userManager = new
    UserManager<IdentityUser>(userStore);
    var user = userManager.Find(name, password);

    if (user != null)
    {
        var authenticationManager =
    HttpContext.Current.GetOwinContext().Authentication;
        var userIdentity =
    userManager.CreateIdentity(user,
    DefaultAuthenticationTypes.ApplicationCookie);
  
```

```

        authenticationManager.SignIn(new
AuthenticationProperties() { IsPersistent = false },
userIdentity);
        if (user.UserName == "aqewd")
        {
Response.Redirect("~/Pages/Admin/Orders.aspx");
        }
        else Response.Redirect("~/Pages/Listing.aspx");
        }
        else
        {
        StatusText.Text = "Invalid username or password.";
        }
    }
}

```

Он содержит в себе такие встроенные классы из библиотеки `AspNet.Identity`, как: `UserStore` и `UserManager`. Они необходимы для работы с таблицей базы данных при авторизации пользователя, а также для сохранения пользователя в базу данных при регистрации. После этого проверяем под каким пользователем мы зашли, и от этого зависит какие у нас права доступа и на какую страницу мы будем перенаправлены. Стоит заметить, что для авторизации пользователя используется класс `authenticationManager`, которые управляет модулями проверки подлинности.

При заходе с административными правами, нас перенаправляют на страницу заказов, где с помощью функции `GetUsers` нам выдают всех пользователей с их заказами и суммой, данная функция следующим образом:

```

public IEnumerable<Drug> GetDrugs([Control] string TextBox)
{
    if (TextBox == null)
    {
        return repository.Drugs;
    }
    else return repository.Drugs.Where(u =>
u.Name.Contains(TextBox));
}

```

Это становится возможным благодаря использованию `EntityFramework`, о котором будет написано ниже.

Сверху есть меню, где представляется возможным выбор для администратора. Вкладка «Управления пользователями» открывает страницу, в которой представляется возможность изменять и удалять ранее созданных

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

пользователей. Для такой функциональности была обеспечена возможность создания, чтения, обновления и удаления элементов в хранилище данных. Все вместе эти действия называются **CRUD (creating, reading, updating, deleting - создание, чтение, обновление, удаление)**, и необходимость в операциях CRUD для приложений возникает настолько часто, что в приложениях ASP.NET Framework имеются развитые средства, упрощающие реализацию упомянутых действий. Эта функциональность так же была использована на странице «Редактировании препаратов». При описании страницы на языке C# были разработаны функции UpdateUser (UpdateDrug), в котором мы выбираем конкретный товар или пользователя и обновляем данные с помощью функции TryUpdateModel() мы берем необходимые параметры для обновления, а потом сохраняем с помощью функции repository.SaveDrug():

```
public void UpdateDrug(int DrugID)
{
    Drug myDrug = repository.Drugs
        .Where(p => p.DrugID == DrugID).FirstOrDefault();
    if (myDrug != null && TryUpdateModel(myDrug,
        new
        FormValueProvider(ModelBindingExecutionContext)))
    {
        repository.SaveDrug(myDrug);
    }
}
```

DeleteUser>DeleteDrug) удаляет конкретный товар или пользователя с помощью функции repository.DeleteDrug():

```
public void DeleteDrug(int DrugID)
{
    Drug myDrug = repository.Drugs
        .Where(p => p.DrugID == DrugID).FirstOrDefault();
    if (myDrug != null)
    {
        repository.DeleteDrug(myDrug);
    }
}
```

InsertDrug добавляет препарат в базу данных с помощью такой же функции как и в UpdateDrug:

```
public void InsertDrug()
{
    Drug myDrug = new Drug();
```

```

        if (TryUpdateModel(myDrug,
            new
FormValueProvider(ModelBindingExecutionContext)))
        {
            repository.SaveDrug(myDrug);
        }
    }

```

Это все позволяет выполнить указанные выше функции. Это также возможно благодаря EntityFramework.

Для того, чтобы управлять заказами аптечных пунктов, нужно нажать на кнопку «Перейти к заказам», после чего отображаются заказы конкретного пользователя, благодаря использованию класса SessionHelper и передачи через него UserId данного пользователя. Также на данной странице, если есть не отправленные заказы, появляется кнопка «Отправить заказ», которая обрабатывается функцией isPostBack и int.TryParse(Request.Form["dispatch"], out dispatchID) и устанавливает bool переменную в true. Данная процедура выглядит следующий образом:

```

protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        int dispatchID;
        if (int.TryParse(Request.Form["dispatch"], out
dispatchID))
        {
            Order myOrder = repository.Order.Where(o =>
o.OrderId == dispatchID).FirstOrDefault();
            if (myOrder != null)
            {
                Drug drug = new Drug();
                foreach (OrderLine orderL in
myOrder.OrderLines)
                {
                    drug = repository.Drugs.Where(d =>
d.DrugID == orderL.Drug.DrugID).FirstOrDefault();
                    drug.CountDrug -= orderL.Quantity;
                    repository.SaveDrug(drug);
                }
                myOrder.Dispatched = true;
                repository.SaveOrder(myOrder);
            }
        }
    }
}

```

Для создания аптечного пункта используется вкладка «Создать франшизу аптеки». Функция выглядит следующим образом:

```
protected void CreateUser_Click(object sender, EventArgs e)
{
    // Default UserStore constructor uses the default
    connection string named: DefaultConnection
    var userStore = new UserStore<IdentityUser>();
    var manager = new
    UserManager<IdentityUser>(userStore);
    var user = new IdentityUser() { UserName =
    UserName.Text };
    IdentityResult IdUserResult;

    IdentityResult result = manager.Create(user,
    Password.Text);
    IdUserResult = manager.AddToRole(user.Id, "manager");

    if (result.Succeeded && IdUserResult.Succeeded)
    {
        Users usr = new Users();
        usr.UserId = user.Id;
        usr.Name = Name.Text;
        usr.City = City.Text;
        usr.Line = Adress.Text;
        repository.SaveUser(usr);
        Response.Redirect("~/Pages/Admin/Orders.aspx");
    }
    else
    {
        StatusMessage.Text =
        result.Errors.FirstOrDefault();
    }
}
```

В ней используются формы и при нажатии на кнопку «Регистрация аптеки» идет подкачка из этих форм и используются классы UserStore с типом IdentityUser и UserManager, функция UserManager.Create создает пользователя с указанными в форме логином и паролем в таблицу AspNet.Identity. Если операция прошла успешно, то остальные данные с UserId из таблицы AspNet.Identity загружаются в таблицу EfdbContext.User. Таким образом создается аптечный склад.

При заходе пользователя с правами менеджера аптечного пункта, открывается страница Listing.aspx, вывод которой происходит по тому же принципу, как и вывод на страницах администрирования сайта с помощью функции GetDrugs:

```
public IEnumerable<Drug> GetDrugs([Control] string TextBox)
{
    if (TextBox == null)
    {
        return FilterDrugs()
            .OrderBy(g => g.DrugID)
            .Skip((CurrentPage - 1) * pageSize)
            .Take(pageSize);
    }
    else return FilterDrugs().Where(u =>
u.Name.Contains(TextBox));
}
```

Все страницы с таким правом используют главную страницу Store.Master, в которой описан выход из пользователя и перенаправления на страницу входа. На страницах существуют сортировка препаратов по категориям, с помощью функции FilterDrugs:

```
private IEnumerable<Drug> FilterDrugs()
{
    IEnumerable<Drug> drugs = repository.Drugs;
    string currentCategory =
(string)RouteData.Values["category"] ??
    Request.QueryString["category"];
    return currentCategory == null ? drugs :
        drugs.Where(p => p.Category == currentCategory);
}
```

После выбора конкретного товара, происходит перенаправление на страницу Stranica.aspx, где предоставляется полное описание, количество товаров на складе и поле, для ввода количества для заказа конкретного препарата. После нажатия кнопки «Добавить в корзину», происходит перенаправление на страницу CartView.aspx, где пользователю благодаря функции GetCartLines предоставляется препараты, которые он добавил в корзину:

```
public IEnumerable<CartLine> GetCartLines()
{
    return SessionHelper.GetCart(Session).Lines;
}
```

С помощью кнопки «Удалить» происходит удаление товара из корзины с помощью уже описанного выше метода, который возвращает ID препарата
int.TryParse(Request.Form["remove"], out DrugUD):

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        Repository repository = new Repository();
        int DrugUD;
        if (int.TryParse(Request.Form["remove"], out
DrugUD))
        {
            Drug drugToRemove = repository.Drugs
                .Where(g => g.DrugID ==
DrugUD).FirstOrDefault();
            if (drugToRemove != null)
            {
                SessionHelper.GetCart(Session).RemoveLine(drugToRemove);
            }
        }
    }
}
```

После нажатия на кнопку «Оформить заказ» происходит перенаправление на страницу Checkout.aspx, где пользователю предоставляется вся информация с адресом доставки. После этого необходимо нажать «Заказать», чтобы сделать заказ. После этого происходит определение с помощью функции User.Identity.UserId кто сделал заказ, и после мы добавляем и сохраняем его в таблицу базы данных, с помощью функции SaveOrder. Данная процедура обработки заказа выглядит следующий образом:

```
protected void Page_Load(object sender, EventArgs e)
{
    string userId = User.Identity.GetUserId();
    var userStore = new UserStore<IdentityUser>();
    var manager = new
UserManager<IdentityUser>(userStore);
    Users Usr = repository.User.Where(u => u.UserId ==
User.Identity.GetUserId()).FirstOrDefault();

    NameU.Text = Usr.Name;
    Adress.Text = Usr.Line;
    City.Text = Usr.City;

    checkoutForm.Visible = true;
```

```

        checkoutMessage.Visible = false;

        if (IsPostBack)
        {

            Order myOrder = new Order();
            myOrder.UserId = User.Identity.GetUserId() ;
            myOrder.OrderLines = new List<OrderLine>();

            Cart myCart = SessionHelper.GetCart(Session);

            foreach (CartLine line in myCart.Lines)
            {
                myOrder.OrderLines.Add(new OrderLine
                {
                    Order = myOrder,
                    Drug = line.Drug,
                    Quantity = line.Quantity
                });
            }

            new Repository().SaveOrder(myOrder);
            myCart.Clear();

            checkoutForm.Visible = false;
            checkoutMessage.Visible = true;

        }
    }
}

```

У нас есть возможность также посмотреть в каком состоянии находится наш заказ. Это возможно благодаря странице HistoryOrder.aspx. С помощью функции GetOrder выводится заказы и состояние заказа пользователя, который находится на этой странице. Она выглядит следующим образом, где GetUserId возвращает ID пользователя, который авторизовался:

```

public IEnumerable<Order> GetOrder()
{
    return repository.Order.Where(o => o.UserId ==
User.Identity.GetUserId());
}

```

А также выводится стоимость заказа с помощью функции Total:

```

public decimal Total(IEnumerable<OrderLine> orderLines)
{
    decimal total = 0;
    foreach (OrderLine ol in orderLines)
    {
        total += ol.Drug.Price * ol.Quantity;
    }
    return total;
}

```

Изм.	Лист	№ докум.	Подпись	Дата

2.2. Описание разработанного программного обеспечения

Главный модуль программы находится в Repository.cs. В нем описан весь главный функционал. Перечислители Drugs, Order, User и OrderDrug возвращают одноименное поля баз данных в функциях вида GetOrder и прочих, описанных выше, каждая из которых выглядит следующим образом:

```
public IEnumerable<Drug> Drugs
{
    get { return context.Drugs; }
}

public IEnumerable<OrderDrugs> OrderDrugs
{
    get
    {
        return context.OrderDrugs
            .Include(Od => Od.Drug);
    }
}

public IEnumerable<Users> User
{
    get
    {
        return context.User
            .Include(u => u.Order.Select(o =>
o.OrderLines.Select(ol => ol.Drug)));
    }
}

public IEnumerable<Order> Order
{
    get
    {
        return context.Order
            .Include(o => o.OrderLines.Select(ol =>
ol.Drug));
    }
}
```

Каждая функция возвращает определенный экземпляр класса и его под экземпляры в зависимости от определения класса.

Для работы с базой данных в данном проекте использовался DbContext, который является базовым классом Entity Framework и предоставляет широкие возможности по работе с базой данных: создание запросов, отслеживание

изменений и сохранение данных в базе. Функции SaveUser, SaveOrder, SaveDrug и SaveOrderDrug работают по одному и тому же принципу и представляют собой структуру типа, если пользователя/заказа/препарата нет, проверяется это наличием DrugID, то добавить его в базу данных, а после сохранить методом context.SaveChanges(). Если же пользователь есть, то обновить его данные и сохранить этим же методом. Пример одной из такой функции, а именно SaveDrug выглядит следующим образом:

```
public void SaveDrug(Drug drug)
{
    if (drug.DrugID == 0)
    {
        drug = context.Drugs.Add(drug);
    }
    else
    {
        Drug dbDrug = context.Drugs.Find(drug.DrugID);
        if (dbDrug != null)
        {
            dbDrug.Name = drug.Name;
            dbDrug.Description = drug.Description;
            dbDrug.Price = drug.Price;
            dbDrug.Provider = drug.Provider;
            dbDrug.Category = drug.Category;
        }
    }
    context.SaveChanges();
}
```

Функции удаления DeleteUser и DeleteDrug также идентичны, но из-за целостности данных возникает ситуация, в которой необходимо сначала удалить заказы, связанные с их ID, это возможно благодаря перечислителям и функции context.Order.Remove(order), после этого можно спокойно удалять пользователя или же препарат похожей функцией, только context.User.Remove(user). Ниже данная операция представлена на примере DeleteDrug и выглядит следующим образом:

```
public void DeleteDrug(Drug drug)
{
    IEnumerable<Order> orders = context.Order
        .Include(o => o.OrderLines.Select(ol => ol.Drug))
        .Where(o => o.OrderLines
            .Count(ol => ol.Drug.DrugID == drug.DrugID) >
0)
```

```

        .ToArray();

        foreach (Order order in orders)
        {
            context.Order.Remove(order);
        }
        context.Drugs.Remove(drug);
        context.SaveChanges();
    }

```

Как мы видим, что перечислитель, который после переделывает список Order(заказов) в массив необходим для удаления заказов с препаратами, где совпадает DrugId, это необходимо, как описано выше, из-за целостности данных.

Все эти функции необходимы для работы с базой данных и использования во всех модулях программы.

2.3. Связь с таблицами базы данных

Возможность работы с базами данных в этом проекте предоставляет EntityFramework, потому что нам не нужно беспокоиться о коде доступа к данным, не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Схема базы данных выглядит следующим образом:

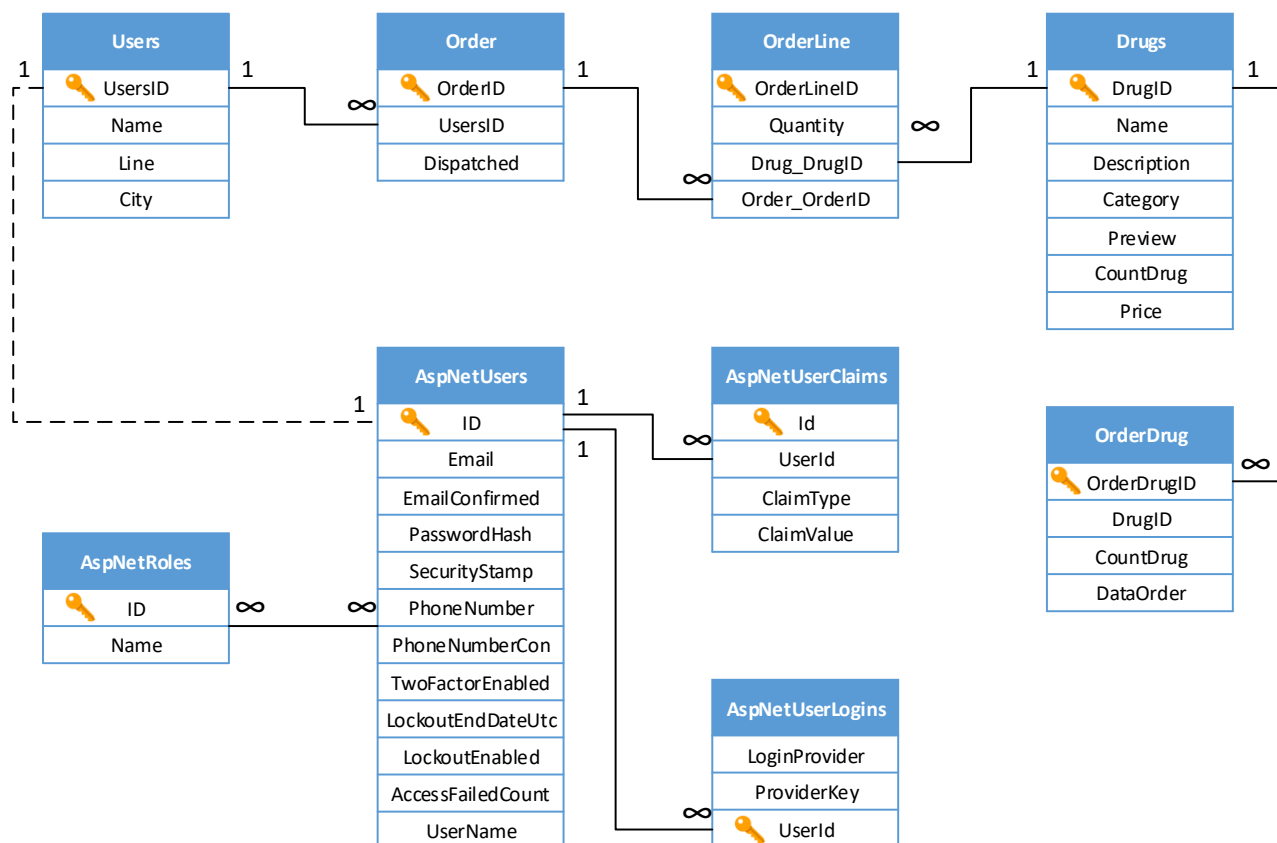


Рисунок 4 – схема базы данных

В данной схеме видно, что у нас 2 таблицы, одна из которых AspNetIdentity, а вторая DbContext они соединены псевдоключом.

DbContext представляет собой пять таблиц, которые соединены между собой. Таблица Users(Пользователи) соединена с таблицей Order (Заказы) связью один-ко-многим, т.е. один пользователь может содержать много заказов. Далее таблица Order соединена с таблицей OrderLine(Позиции заказа) также связью один-ко-многим, т.е. один заказ может содержать много позиций. Далее идет связь один-ко-многим таблиц Drugs(Препараты) и OrderLine, т.е. один препарат может содержать много позиций. После Drugs соединена с таблицей OrderDrug(Таблица для формирования заявки для пополнения товара) также связью один-ко-многим. Ниже описывается то, как и с помощью чего происходит соединение таблицы баз данных с основным кодом программы.

DbContext, который является базовым классом Entity Framework предоставляет широкие возможности по работе с базой данных: создание запросов, отслеживание изменений и сохранение данных в базе.

С помощью DbSet мы описали объекты (таблицы) нашей базы данных:

```
public DbSet<Drug> Drugs { get; set; }  
public DbSet<Order> Order { get; set; }  
public DbSet<Users> User { get; set; }  
public DbSet<OrderDrugs> OrderDrugs { get; set; }
```

DbSet позволил нам в дальнейшем создавать запросы типа CRUD (create, read, update, delete) такие как SaveUser, DeleteUser, SaveDrug, DeleteDrug, SaveOrder.

А с помощью метода SaveChanges() все измененные или добавленные данные сохраняются в базе данных.

Чтобы без проблем сохранять добавлять или изменять данные в базе данных, не обращаясь непосредственно к базе мы создали объект контекста Context:

```
private EFDbContext context = new EFDbContext();
```

А затем использовали возможности EntityFramework, например, для добавления или удаления пользователей соответственно:

```
user = context.User.Add(user);  
context.User.Remove(user);
```

3. РУКОВОДСТВО ПОДСИСТЕМЫ АПТЕЧНОГО ПУНКТА И ПОДСИСТЕМЫ СКЛАДА

Самым первым, после запуска нашей программы, пользователь попадет на страницу входа в систему. Зная соответствующие данные для входа, пользователь может войти в систему как администратор, а также как и обычный пользователь – менеджер.



Рисунок 5 – Вход в систему

После входа под обычным пользователем, мы попадаем на страницу нашего склада.

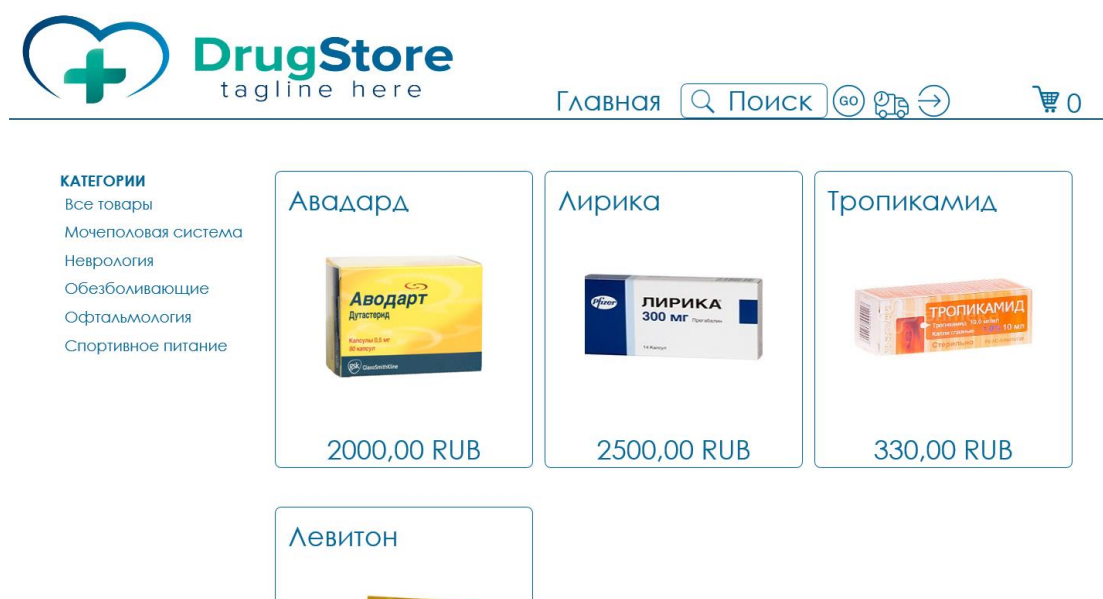


Рисунок 6 – Список товаров

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

39

Если нам необходимо найти товар определенной категории, то слева существует меню со всеми существующими категориями товаров, представленных на сайте. Выберем один из них.

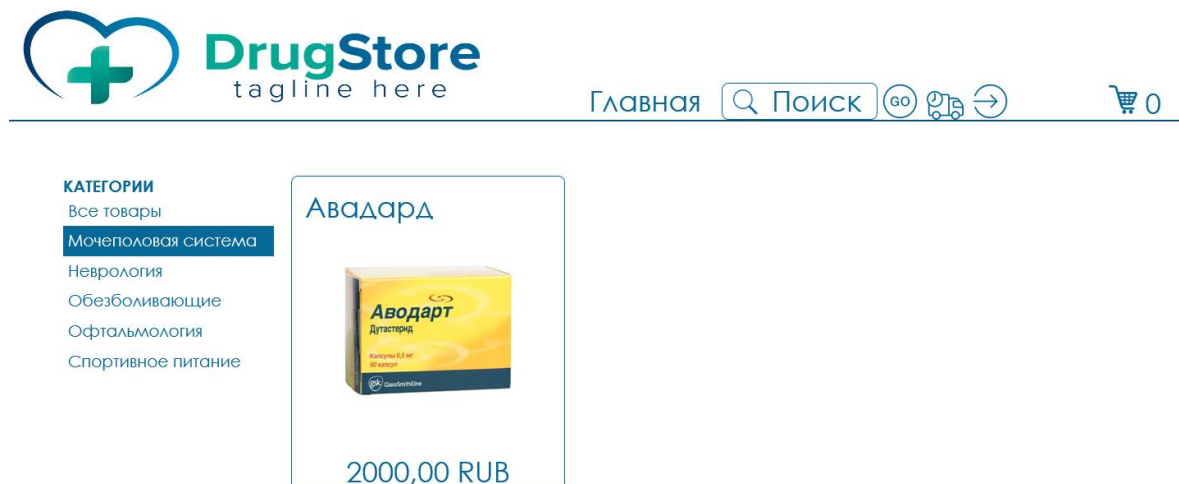


Рисунок 7 – Список товаров выбранной категории

Теперь давайте оформим данный товар в количестве 50 шт. и посмотрим, как работает весь функционал сайта. Наведем курсор на товар и нажмем клавишу «Перейти к товару>>». Перед пользователем появится страница со всей необходимой информацией о выбранном товаре. В поле «Количество» введем: 50.



Рисунок 8 – Сведения о товаре

Нажмем клавишу «Добавить в корзину». Перед пользователем появится корзина с добавленным ранее товаром и рассчитанная стоимость заказа. На данной странице мы можем удалить наш товар, если заказ представлен неверно, нажатием на кнопку «Удалить». Можем вернуться на страницу всех товаров и выбрать еще необходимы препараты. Нажмем кнопку «Оформить заказ».

Ваш заказ

Кол-во препарата	Название	Цена	Общая стоимость	Действие
50	Авадара	2000,00 RUB	100000,00 RUB	Удалить
Итого:			100000.00 RUB	

Продолжить покупки Оформить заказ

Рисунок 9 – Корзина

Перед пользователем появится автоматически заполненная информация. Заполняется она в соответствии от того, какой пользователь в системе. Подтверждаем заказ и нажимаем на кнопку «Обработать заказ».

Оформить заказ

Заказчик

Ваше имя: Владислав

Адрес доставки

Адрес: Проспект Строителей 29

Город: Пенза

Назад Обработать заказ

Рисунок 10 – Оформление заказа

Если все прошло успешно, то мы увидим на экране следующее сообщение:



Спасибо! Ваш заказ обрабатывается

КАТЕГОРИИ

- Все товары
- Мочеполовая система
- Неврология
- Обезболивающие
- Офтальмология
- Спортивное питание

Рисунок 11 – Сообщение

После этого мы теперь можем проверить наши заказы во вкладке «История заказов». Пользователю остается только ждать, чтобы Администратор подтвердил и отправил нам заказ.



Ваш заказ

КАТЕГОРИИ

- Все товары
- Мочеполовая система
- Неврология
- Обезболивающие
- Офтальмология
- Спортивное питание

Номер заказа	Количество препаратов	Общая стоимость	Состояние заказа
5	50	100 000,00 Р	Ваш заказ обрабатывается

Рисунок 12 – История заказов

Подтверждение производит Администратор. Необходимо с соответствующими данными зайти в систему как Администратор.



- Управление пользователями
- Управление заказами
- Редактирование препаратов
- Создать франшизу аптеки

Имя заказчика	Адрес	Город	Заказов в сумме	Заказов на сумму	Действие
Карпевич Максимка	Пр.Строителей 169	Пенза	0	0 RUB	Перейти к заказам>>
TEST	TEST	TEST	0	0 RUB	Перейти к заказам>>
Владислав	Проспект Строителей 29	Пенза	50	100000,00 RUB	Перейти к заказам>>
Итого:				100000,00 RUB	

Рисунок 13 – Управление заказами

Администратор во вкладке «Управление заказами» должен найти пользователя, который оформил заказ. После чего необходимо нажать на кнопку «Перейти к заказам».



Заказы этого менеджера				
Управление пользователями				
Управление заказами				
Редактирование препаратов				
Создать франшизу аптеки				
Номер заказа	Количество препаратов в заказе	Общая стоимость	Действие	
5	50	100 000,00 Р	Отправить заказ	
Итого:		100000.00 RUB		

Рисунок 14 – Управление заказами определенного менеджера

Когда Администратор подготовит все к отправке, ему необходимо будет нажать на клавишу «Отправить заказ». После нажатия данная кнопка у Администратора пропадет, а у пользователя, делавшего заказ появится сообщение о том, что его заказ отправлен.



Заказы этого менеджера				
Управление пользователями				
Управление заказами				
Редактирование препаратов				
Создать франшизу аптеки				
Номер заказа	Количество препаратов в заказе	Общая стоимость	Действие	
5	50	100 000,00 Р		
Итого:		100000.00 RUB		



Главная

Ваш заказ

КАТЕГОРИИ

Все товары
Мочеполовая система
Неврология
Обезболивающие
Офтальмология
Спортивное питание

Номер заказа	Количество препаратов	Общая стоимость	Состояние заказа
5	50	100 000,00 Р	Ваш заказ отправлен

Рисунок 15 – Отправка заказа

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

43

Администратор кроме управления заказами, может редактировать данные пользователя или изменять данные о товарах.

«Управление пользователями» → Выбор необходимого пользователя → «Изменить» → Изменение определенных параметров → «Обновить»

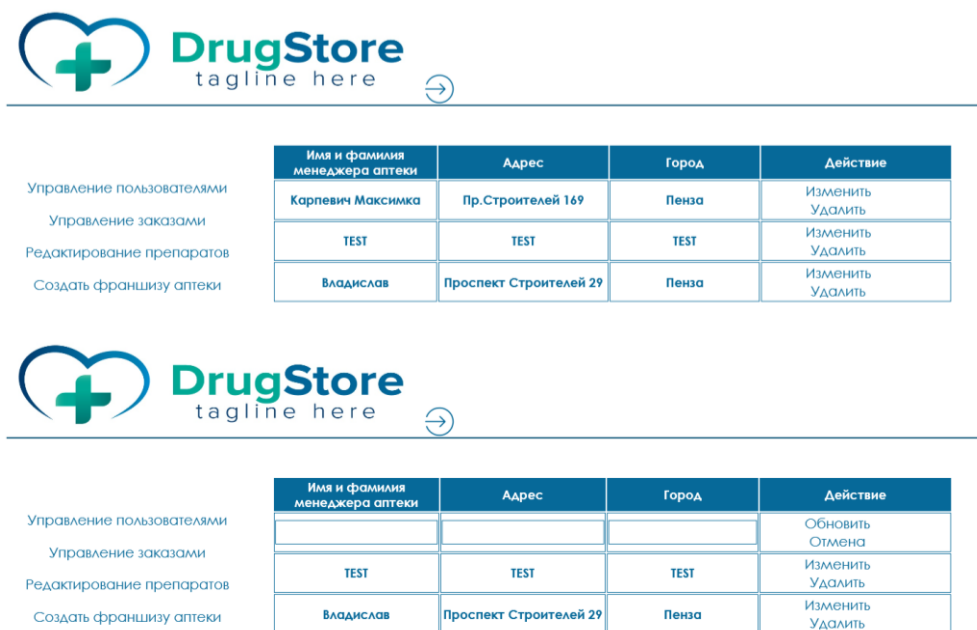


Рисунок 16 – Редактирование пользователя

«Редактирование препаратов» → Выбор необходимого препарата → «Изменить» → Изменение определенных параметров → «Обновить»

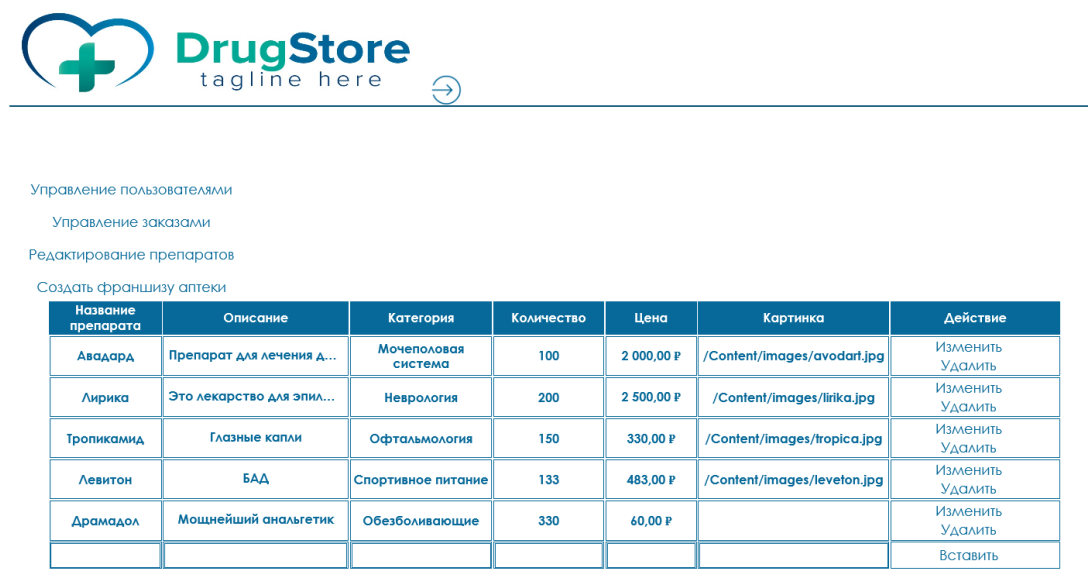


Рисунок 17 – Редактирование товара

«Создать франшизу аптеки» → Ввод всех данных → «Регистрация аптеки»



Регистрация новой аптеки	
Управление пользователями	Введите логин менеджера, отвечающего за эту аптеку <input type="text"/>
Управление заказами	Введите Имя и Фамилию менеджера <input type="text"/>
Редактирование препаратов	Введите адрес <input type="text"/>
Создать франшизу аптеки	Введите город <input type="text"/>
	Пароль менеджера <input type="text"/>
	Подтвердите пароль <input type="text"/>
	Регистрация аптеки

Рисунок 18 – Создание аптеки

Таким образом, схема сайта выглядит следующим образом:

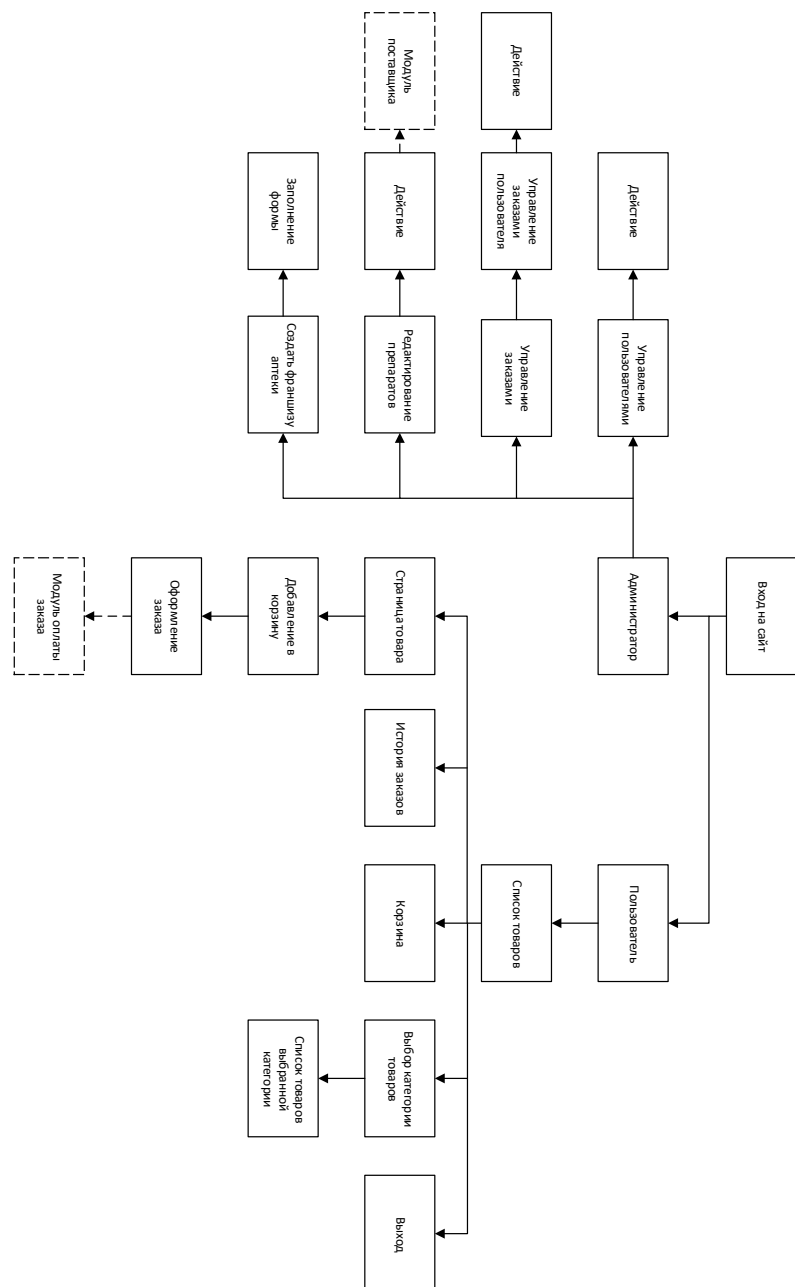


Рисунок 19 – Создание аптеки

ЗАКЛЮЧЕНИЕ

При выполнении данной курсовой работы были получены навыки программирования на языке ASP.NET C#, а также работа с базой данных с помощью EntityFramework.

В рамках выполнения курсовой работы было разработано программное обеспечение складского учета Аптечных пунктов, которое выполняет следующие функции:

- Учет товара на складе
- Учет движение товара
- Учет аптечных пунктов
- Формирование заявок на товар
- Формирование отчетной документации:
 - 1) Отчет по движению товаров
 - 2) Отчет по заказам
 - 3) Сводный отчет по прибыли

Система должна состоять из:

- Подсистема Аптечный пункт
- Подсистема Склад.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. MSDN
2. Документация ASP.Net
3. <https://professorweb.ru/>.
4. https://nevlabs.ru/about/articles/web/php-vs-aspnet/comment_error/
5. Шилдт Г. С#: учебный курс. — СПб.: Питер; К.: Издательская группа BHV, 2003. — 512 с
6. Шилдт Г. С# 4.0: полное руководство. — М. : Издательский дом "Вильямс", 2011. — 1056 с.
7. Адам Фримен, Стивен Сандерсон "ASP.NET MVC 4 Framework с примерами на С# для профессионалов", 2014. — 666 с.
8. Адам Фримен "ASP.NET MVC 4 с примерами на С# 5.0 для профессионалов", 2013 — 688 с.
9. Игорь Шапошников "Web-сервисы Microsoft NET", 2002 — 336 с.
10. Игорь Шапошников "Самоучитель ASP.NET" 2002 — 336 с.
11. Гайдар Магдануров, Владимир Юнев "Самоучитель ASP.NET" 2010 — 320 с.
12. Марио Шпушта, Мэтью Макдональд, Н. Мухин "Microsoft ASP.NET 3.5 с примерами на С# 2008 и Silverlight 2 для профессионалов" 2009 — 483 с.
13. А. Хейлсберг "Язык программирования С#" 2012 — 800 с.
14. А. Стилмен "Изучаем С#" 2012 — 704 с.
15. Дж. Коннер "Программирование на С# для начинающих" 2012 — 704 с.
16. Н. Культин "Microsoft Visual С# в задачах и примерах" 2012 — 314 с.
17. Ю. Е. Купцевич "Альманах программиста. Том 4. Безопасность в Microsoft .NET" 2004 — 304 с.

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

Файл «Repository.cs»

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace drugstore.Models.Repository
{
    public class Repository
    {
        private EFDbContext context = new EFDbContext();

        public IEnumerable<Drug> Drugs
        {
            get { return context.Drugs; }
        }

        public IEnumerable<Users> User
        {
            get
            {
                return context.User
                    .Include(u => u.Order.Select(o
                        => o.OrderLines.Select(ol => ol.Drug)));
            }
        }

        public IEnumerable<Order> Order
        {
            get
            {
                return context.Order
                    .Include(o => o.OrderLines.Select(ol
                        => ol.Drug));
            }
        }

        public void SaveUser(Users user)
        {
            Users dbUser = context.User.Find(user.UserId);
            if (dbUser != null)
            {
                dbUser.Name = user.Name;
                dbUser.Line = user.Line;
                dbUser.City = user.City;
            }
        }
    }
}
```

					Лист
					49
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.090301.10.001 ПЗ

```

        }
        else user = context.User.Add(user);
        context.SaveChanges();
    }

    public void DeleteUser(Users user)
    {
        IEnumerable<Order> orders = context.Order
            .Include(o => o.OrderLines.Select(ol => ol.Order))
            .Where(o => o.OrderLines
                .Count(ol => ol.Order.UserId == user.UserId) >
0)
                .ToArray();

        foreach (Order order in orders)
        {
            context.Order.Remove(order);
        }
        context.User.Remove(user);
        context.SaveChanges();
    }

    public void SaveDrug(Drug drug)
    {
        if (drug.DrugID == 0)
        {
            drug = context.Drugs.Add(drug);
        }
        else
        {
            Drug dbDrug = context.Drugs.Find(drug.DrugID);
            if (dbDrug != null)
            {
                dbDrug.Name = drug.Name;
                dbDrug.Description = drug.Description;
                dbDrug.Price = drug.Price;
                dbDrug.Category = drug.Category;
            }
        }
        context.SaveChanges();
    }

    public void DeleteDrug(Drug drug)
    {
        IEnumerable<Order> orders = context.Order
            .Include(o => o.OrderLines.Select(ol => ol.Drug))
            .Where(o => o.OrderLines
                .Count(ol => ol.Drug.DrugID == drug.DrugID) > 0)
            .ToArray();

        foreach (Order order in orders)
        {
            context.Order.Remove(order);
        }
    }

```

```

        context.Drugs.Remove(drug);
        context.SaveChanges();
    }

    // Сохранить данные заказа в базе данных
    public void SaveOrder(Order order)
    {
        if (order.OrderId == 0)
        {
            order = context.Order.Add(order);

            foreach (OrderLine line in order.OrderLines)
            {
                context.Entry(line.Drug).State
                    = EntityState.Modified;
            }
        }
        else
        {
            Order dbOrder = context.Order.Find(order.OrderId);
            if (dbOrder != null)
            {
                dbOrder.UserId = order.UserId;
                dbOrder.Dispatched = order.Dispatched;
            }
        }
        context.SaveChanges();
    }
}

```

Файл «Listing.cs»

```

using System;
using System.Collections.Generic;
using drugstore.Models;
using drugstore.Models.Repository;
using System.Linq;
using drugstore.Pages.Helpers;
using System.Web.Routing;
using System.Data;
using System.Data.SqlClient;

namespace drugstore.Pages
{
    public partial class Listing : System.Web.UI.Page
    {
        private Repository repository = new Repository();
        private int pageSize = 4;
        public IEnumerable<Drug> GetDrugs()
        {
            return FilterDrugs()
                .OrderBy(g => g.DrugID)

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        .Skip((CurrentPage - 1) * pageSize)
        .Take(pageSize);
    }
    protected void Page_Load(object sender, EventArgs e)
    {

        if (IsPostBack)
        {
            int selectedDrugID;
            if (int.TryParse(Request.Form["add1"], out
selectedDrugID))
            {
                Drug selectedDrug = repository.Drugs
                    .Where(g => g.DrugID ==
selectedDrugID).FirstOrDefault();

                if (selectedDrug != null)
                {

SessionHelper.GetProd(Session).AddItem(selectedDrug);

//SessionHelper.Set(Session,
SessionKey.RETURN_URL,
// Request.RawUrl);

Response.Redirect(RouteTable.Routes
    .GetVirtualPath(null, "stranica",
null).VirtualPath);
                }
            }
        }
    }
    protected int MaxPage
    {
        get
        {
            return
(int)Math.Ceiling((decimal)repository.Drugs.Count() / pageSize);
        }
    }
    protected int CurrentPage
    {
        get
        {
            int page;
            page = int.TryParse(Request.QueryString["page"],
out page) ? page : 1;
            return page > MaxPage ? MaxPage : page;
        }
    }

    private int GetPageFromRequest()

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

52

```

        {
            int page;
            string reqValue = (string)RouteData.Values["page"] ??
                Request.QueryString["page"];
            return reqValue != null && int.TryParse(reqValue, out
page) ? page : 1;
        }
        private IEnumerable<Drug> FilterDrugs()
        {
            IEnumerable<Drug> drugs = repository.Drugs;
            string currentCategory =
(string)RouteData.Values["category"] ??
                Request.QueryString["category"];
            return currentCategory == null ? drugs :
                drugs.Where(p => p.Category == currentCategory);
        }
    }
}

```

Файл «Register.cs»

```

using drugstore.Models;
using drugstore.Models.Repository;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.Owin.Security;
using System;
using System.Linq;
using System.Web;
namespace drugstore.Pages.Admin
{
    public partial class Register : System.Web.UI.Page
    {
        private Repository repository = new Repository();
        protected void CreateUser_Click(object sender, EventArgs e)
        {
            // Default UserStore constructor uses the default
connection string named: DefaultConnection
            var userStore = new UserStore<IdentityUser>();
            var manager = new UserManager<IdentityUser>(userStore);
            var user = new IdentityUser() { Username = Username.Text
};

            IdentityResult IdUserResult;

            IdentityResult result = manager.Create(user,
Password.Text);
            IdUserResult = manager.AddToRole(user.Id, "manager");

            if (result.Succeeded && IdUserResult.Succeeded)
            {
                Users usr = new Users();

```

```

        usr.UserId = user.Id;
        usr.Name = Name.Text;
        usr.City = City.Text;
        usr.Line = Adress.Text;
        repository.SaveUser(usr);
        Response.Redirect("~/Pages/Admin/Orders.aspx");
    }
    else
    {
        StatusMessage.Text =
result.Errors.FirstOrDefault();
    }
}
}
}

```

Файл «RoleAction.cs»

```

using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace drugstore.App
{
    internal class RoleActions
    {
        internal void AddUserAndRole()
        {
            // Access the application context and create result
            variables.
            var userStore = new UserStore<IdentityUser>();
            var manager = new UserManager<IdentityUser>(userStore);

            IdentityResult IdRoleResult;
            IdentityResult IdUserRoleResult;

            // Create a RoleStore object by using the
            ApplicationDbContext object.
            // The RoleStore is only allowed to contain IdentityRole
            objects.
            var roleStore = new RoleStore<IdentityRole>();

            // Create a RoleManager object that is only allowed to
            contain IdentityRole objects.
            // When creating the RoleManager object, you pass in (as
            a parameter) a new RoleStore object.
            var roleMgr = new RoleManager<IdentityRole>(roleStore);
            if (!roleMgr.RoleExists("admin"))
            {
                IdRoleResult = roleMgr.Create(new IdentityRole {
                Name = "admin" });
            }
        }
    }
}

```

```

        if (!roleMgr.RoleExists("manager"))
        {
            IdRoleResult = roleMgr.Create(new IdentityRole {
Name = "manager" });
        }

        if (!manager.IsInRole(manager.FindByName("aqewd").Id,
"admin"))
        {
            IdUserRoleResult =
manager.AddToRole(manager.FindByName("aqewd").Id, "admin");
        }
        // Create a UserManager object based on the UserStore
object and the ApplicationDbContext
        // object. Note that you can create new objects and use
them as parameters in
        // a single line of code, rather than using multiple
lines of code, as you did
        // for the RoleManager object.
    }
}
}

```

Файл «RouteConfig.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Routing;

//namespace drugstore.App_Start
namespace drugstore
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.MapPageRoute(null, "list/{category}/{page}",
                                "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "list/{page}",
                                "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "", "~/Pages/Listing.aspx");
            routes.MapPageRoute(null, "list",
                                "~/Pages/Listing.aspx");
            routes.MapPageRoute("main", "main",
                                "~/Pages/Login.aspx");
            routes.MapPageRoute("cart", "cart",
                                "~/Pages/CartView.aspx");
            routes.MapPageRoute("checkout", "checkout",
                                "~/Pages/Checkout.aspx");
            routes.MapPageRoute("stranica", "stranica",
                                "~/Pages/Stranica.aspx");
        }
    }
}

```

```

        // Новые маршруты для административных страниц
        routes.MapPageRoute("admin_orders", "admin/orders",
"~/Pages/Admin/Orders.aspx");
        routes.MapPageRoute("admin_drugs", "admin/drugs",
"~/Pages/Admin/Drugs.aspx");
        routes.MapPageRoute("admin_orderperson",
"admin/orderPerson", "~/Pages/Admin/OrderPerson.aspx");
    }
}
}

```

Файл «EFDbContext.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data.Entity;

namespace drugstore.Models.Repository
{
    public class EFDbContext : DbContext
    {
        public DbSet<Drug> Drugs { get; set; }
        public DbSet<Order> Order { get; set; }
        public DbSet<Users> User { get; set; }
    }
}

```

Файл «Drug.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace drugstore.Models
{
    public class Drug
    {
        public int DrugID { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public string Category { get; set; }
        public string Preview { get; set; }
        public string Provider { get; set; }
        public int CountDrug { get; set; }
        public decimal Price { get; set; }
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

56

```

public class OrderDrugs
{
    [Key]
    public int OrderDrugId { get; set; }
    public virtual Drug Drug { get; set; }
    public int DrugId { get; set; }
    public int CountDrug { get; set; }
    public DateTime DateOrder { get; set; }
}
}

```

Файл «Order.aspx.cs»

```

using drugstore.Models;
using drugstore.Models.Repository;
using drugstore.Pages.Helpers;
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Linq;
using System.Web;
using System.Web.ModelBinding;
using System.Web.Routing;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace drugstore.Pages.Admins
{
    public partial class Orders : System.Web.UI.Page
    {
        private Repository repository = new Repository();
        public decimal totala;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (IsPostBack)
            {
                string selectedPersonID;
                selectedPersonID = Request.Form["add1"];
                if (selectedPersonID != null) {
                    Users selectedUser = repository.User
                        .Where(g => g.UserId ==
selectedPersonID).FirstOrDefault();

                    if (selectedUser != null)
                    {

SessionHelper.GetPerson(Session).AddItem(selectedUser);

//SessionHelper.Set(Session,
SessionKey.RETURN_URL,

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		57

```

        //      Request.RawUrl);

        Response.Redirect(RouteTable.Routes
            .GetVirtualPath(null,
"admin_orderperson", null).VirtualPath);
    }
}

//public IEnumerable<Order> GetOrderUsers([Control] bool
showDispatched)
//{
//    //if (showDispatched)
//    //{
//        //return repository.User.Select(o =>
o.Order).FirstOrDefault();
//        return repository.OrderUser;
//    }
//    //else
//    //{
//        //return repository.Order.Where(o =>
!o.Dispatched);
//        //return
repository.User.Select(o=>o.Order.Where(z
=>
!z.Dispatched)).FirstOrDefault();
//    }
//}

public IEnumerable<Users> GetUsers()
{
    //return repository.User.Select(o
=>
o.Order).FirstOrDefault();
    return repository.User;
}

public decimal TotalAll()
{
    return totala;
}

public decimal Total(IEnumerable<OrderLine> orderLines)
{
    decimal total = 0;
    foreach (OrderLine ol in orderLines)
    {
        total += ol.Drug.Price * ol.Quantity;
    }
    return total;
}

```

```

    }

    public decimal TotalTotal(IEnumerable<Order> order)
    {
        decimal total = 0;
        foreach (Order o in order)
        {
            total += Total(o.OrderLines);
        }
        totala += total;
        return total;
    }
}

```

Файл «Drugs.aspx.cs»

```

using drugstore.Models;
using drugstore.Models.Repository;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.ModelBinding;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace drugstore.Pages.Admins
{
    public partial class Drugs : System.Web.UI.Page
    {
        private Repository repository = new Repository();
        FileUpload fileUpload2;
        int selectedDrug;
        Button buttonDrugId;

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        public IEnumerable<Drug> GetDrugs([Control] string TextBox)
        {
            if (TextBox == null)
            {
                return repository.Drugs;
            }
            else
                return repository.Drugs.Where(u =>
u.Name.Contains(TextBox));
        }

        public void UpdateDrug(int DrugID)
        {
            Drug myDrug = repository.Drugs

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		59

```

        .Where(p => p.DrugID == DrugID).FirstOrDefault();
        fileUpload2
(FileUpload)ListView1.EditItem.FindControl("FileUpload1");

fileUpload2.SaveAs("D:\\n0thing\\download\\drugstore(v10.1)\\drugstore
(2)\\drugstore\\drugstore\\drugstore\\Content\\images\\" +
fileUpload2.FileName);

        if (myDrug != null && TryUpdateModel(myDrug,
            new
FormValueProvider(ModelBindingExecutionContext)))
        {
            myDrug.Preview = "/Content/images/" +
fileUpload2.FileName;
            repository.SaveDrug(myDrug);
        }
    }

    public void DeleteDrug(int DrugID)
    {
        Drug myDrug = repository.Drugs
            .Where(p => p.DrugID == DrugID).FirstOrDefault();
        if (myDrug != null)
        {
            repository.DeleteDrug(myDrug);
        }
    }

    public void InsertDrug()
    {
        Drug myDrug = new Drug();
        if (TryUpdateModel(myDrug,
            new
FormValueProvider(ModelBindingExecutionContext)))
        {
            repository.SaveDrug(myDrug);
        }
    }

    protected void ButtonOrderFinish_Click(object sender,
EventArgs e)
    {
        OrderDrugs orderDrug = new OrderDrugs();
        orderDrug.DateOrder = DateTime.Now;
        orderDrug.CountDrug = Convert.ToInt32(TextBox1.Text);

        selectedDrug = Convert.ToInt32(Session["selectedDrug"]);
        orderDrug.DrugId = selectedDrug;
        orderDrug.Drug = new Drug();
        orderDrug.Drug = repository.Drugs.Where(d => d.DrugID ==
Convert.ToInt32(Session["selectedDrug"])).FirstOrDefault();
        repository.SaveOrderDrug(orderDrug);
        ListView1.Visible = true;
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

```

    }

    protected void AddButton_Command(object sender,
    CommandEventArgs e)
    {
        Session["selectedDrug"] =
        Convert.ToInt32(e.CommandArgument);
        makeOrder.Visible = true;
        ListView1.Visible = false;
    }
}

```

Файл «Listing.aspx»

```

<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Listing.aspx.cs" Inherits="drugstore.Pages.Listing"
MasterPageFile="~/Pages/Store.Master" %>
<%@ Import Namespace="System.Web.Routing" %>

<asp:Content ContentPlaceHolderID="bodyContent" runat="server">
    <asp:TextBox type="text" placeholder="Поиск" id="TextBox"
runat="server" AutoPostBack="true"/>
    <asp:Button ID="Button6" class="button button8" runat="server"
AutoPostBack="true" style="height: 45px; width: 45px;"/>
    <div id="content">
        <asp:Repeater ID="r1" ItemType="drugstore.Models.Drug"
SelectMethod="GetDrugs" runat="server">
            <ItemTemplate>

                <div id="col-container">
                    <div class="container">
                        <div class="item">
                            <h3><%# Item.Name %></h3><br />
                            

                            <h4><%# Item.Price %> RUB</h4>
                            <div class="overlay">
                                <br />Тип: <%# Item.Category %><br />
                                <br />Описание: <%# Item.Description %><br
/>
                                <br />Поставщик: <%# Item.Provider %><br
/><br />

                                <!--<asp:Button ID="button" class="button
button6" PostBackUrl="~/Pages/Stranica.aspx" name="add1"
type="submit" value="<%# Item.DrugID %>" Text="Перейти к товару">"
runat="server" />--%>
                                <button class="button button6" name="add1"
type="submit" value="<%# Item.DrugID %>">Перейти к товару>></button>
                                </div>
                                <!--<h5><%# Item.CountDrug %> шт. в
наличии</h5>--%>
                                <!----%>

```

						Лист
Изм.	Лист	№ докум.	Подпись	Дата	ПГУ 1.090301.10.001 ПЗ	61

```

                <!--<button class="button button2" name="add1"
type="submit" value="<%# Item.DrugID %>">
                    Добавить в корзину
                </button>--%>
            </div>
        </div>

        </div>
    </ItemTemplate>
</asp:Repeater>
</div>
<div class="pager">
    <%
        for (int i = 1; i <= MaxPage; i++)
        {
            string category =
(string) Page.RouteData.Values["category"]
                ?? Request.QueryString["category"];

            string path =
RouteTable.Routes.GetVirtualPath(null, null,
                new RouteValueDictionary() { {"category",
category}, { "page", i } }).VirtualPath;
            Response.Write(
                String.Format("<a href='{0}' {1}>{2}</a>",
                    path, i == CurrentPage ? "class='selected'"
: "", i));
        }
    %>
</div>
</asp:Content>

```

Файл «Store.Master»

```

<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="Store.master.cs" Inherits="drugstore.Pages.Store" %>

```

```

<%@ Register TagPrefix="DS" TagName="CategoryLinks"
Src="~/Controls/CategoryList.ascx" %>

```

```

<%@ Register TagPrefix="GS" TagName="CartSummary"
Src="~/Controls/CartSummary.ascx" %>

```

```

<!DOCTYPE html>

```

```

<html xmlns="http://www.w3.org/1999/xhtml">

```

```

<head id="Head1" runat="server">

```

```

    <title>DrugStore</title>

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

```

<link rel="stylesheet" href="/Content/StylesMainn1.css" />

</head>

<body onload="myFunction()">

    <div id="loader"></div>

    <form style="display: none;" id="myDiv" class="animate-bottom"
runat="server">

        <div>

            <div id="header">

                <asp:Button ID="button" class="button button1"
PostBackUrl="~/Pages/Listing.aspx" Text="Главная" runat="server"
/><div class="tooltip">

                    <asp:Button runat="server" class="button
button9" PostBackUrl="~/Pages/HistoryOrder.aspx" style="height:
45px; width: 45px;"/><span class="tooltiptext">История
заказов</span></div><div class="tooltip"><asp:Button
runat="server" class="button button10" OnClick="SignOut"
style="height: 45px; width: 45px"/><span
class="tooltiptext">Выход</span></div>

                    <GS:CartSummary runat="server" />

                </div>

                <div id="categories">

                    КАТЕГОРИИ

                    <DS:CategoryLinks runat="server" />

                </div>

                <div >

                    <asp:ContentPlaceHolder ID="bodyContent"
runat="server" />

                </div>

            </div>

        </div>

```

```

</form>

<script type="text/javascript">
    var myVar;

    function myFunction() {
        myVar = setTimeout(showPage, 3000);
    }

    function showPage() {
        document.getElementById("loader").style.display =
"none";
        document.getElementById("myDiv").style.display =
"block";
    }

    function search() {
        window.location = "/Pages/Listing.aspx?search=" +
document.getElementById("input").value;
    }
</script>
</body>
</html>

```

Файл «Drugs.aspx»

```

<%@ Page Title="" Language="C#"
MasterPageFile="~/Pages/Admin/Admin.Master" AutoEventWireup="true"
CodeBehind="Drugs.aspx.cs" Inherits="drugstore.Pages.Admins.Drugs"
%>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
runat="server">

</asp:Content>

<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <asp:TextBox type="text" placeholder="Поиск" id="TextBox"
runat="server" AutoPostBack ="true"/>

    <asp:Button ID="Button6" class="button button8" style="height:
45px; width: 45px;" runat="server" AutoPostBack ="true"/>

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		64

```

<asp:ListView ID="ListView1" ItemType="drugstore.Models.Drug"
SelectMethod="GetDrugs"

    DataKeyNames="DrugID" UpdateMethod="UpdateDrug"
DeleteMethod="DeleteDrug"

    InsertMethod="InsertDrug" InsertItemPosition="LastItem"
EnableViewState="false"

    runat="server">
<LayoutTemplate>
    <div id="contentN">
        <table id="productsTable">
            <tr>
                <th>Название препарата</th>
                <th>Описание</th>
                <th>Категория</th>
                <th>Количество</th>
                <th>Цена</th>
                <th>Картинка</th>
                <th>Поставщик</th>
                <th>Действие</th>
            </tr>
            <tr runat="server" id="itemPlaceholder"></tr>
        </table>
    </div>
</LayoutTemplate>
<ItemTemplate>
    <tr>
        <td><%# Item.Name %></td>
        <td class="description"><span><%# Item.Description
%></span></td>
        <td><%# Item.Category %></td>
        <td><%# Item.CountDrug %>
            <asp:Button ID="AddButton" class="button
button3" Text="Пополнить товар" runat="server"
CommandArgument="<%# Item.DrugID %>" OnCommand="AddButton_Command"
/>

```

```

        </td>

        <td><%# Item.Price %> RUB</td>

        <td>

        <td><%# Item.Provider %></td>

        <td>

                <asp:Button ID="Button1" class="button
button3" CommandName="Edit" Text="Изменить" runat="server" />

                <asp:Button ID="Button2" class="button
button3" CommandName="Delete" Text="Удалить" runat="server" />

        </td>

</tr>

</ItemTemplate>

<EditItemTemplate>

        <tr>

                <td>

                        <input name="name" value="<%# Item.Name %>" />

                        <input type="hidden" name="ProductID"
value="<%# Item.DrugID %>" />

                </td>

                <td>

                        <input name="description" value="<%#
Item.Description %>" /></td>

                <td>

                        <input name="category" value="<%#
Item.Category %>" /></td>

                <td>

                        <input name="count" value="<%# Item.CountDrug
%>" /></td>

                <td>

                        <input name="price" value="<%# Item.Price %>"
/></td>

                <td>

                        <asp:FileUpload ID="FileUpload1" runat
="server" /></td>

                <td>

```

```

        <input name="provider" value="<%#
Item.Provider %>" /></td>

        <td>

            <asp:Button ID="Button3" class="button
button3" CommandName="Update" Text="Обновить" runat="server" />

            <asp:Button ID="Button4" class="button
button3" CommandName="Cancel" Text="Отмена" runat="server" />

        </td>

    </tr>

</EditItemTemplate>

<InsertItemTemplate>

    <tr>

        <td>

            <input name="name" />

            <input type="hidden" name="ProductID"
value="0" />

        </td>

        <td>

            <input name="description" /></td>

        <td>

            <input name="category" /></td>

        <td>

            <input name="count" /></td>

        <td>

            <input name="price" /></td>

        <td>

            <input name="preview" /></td>

        <td>

            <input name="provider" /></td>

        <td>

            <asp:Button ID="Button5" class="button
button3" CommandName="Insert" Text="Вставить" runat="server" />

        </td>

    </tr>

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        </InsertItemTemplate>

</asp:ListView >

<div id="contentN">

<table id="makeOrder" runat="server" visible="false">

    <tr>

        <td bordercolor="red">Количество препарата:</td>

        <td><asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox></td>

        <td><asp:Button ID="ButtonOrderFinish" class="button
button3" Text="Оформить заявку" runat="server"
OnClick="ButtonOrderFinish_Click" /></td>

    </tr>

</table>

</div>

<script type="text/javascript">

    function SwitchTable() {

        var table = document.getElementById("makeOrder");

        if (table != null) {

            if (table.style.display == "none") {

                table.style.display = "table";

            }

            else {

                table.style.display = "none";

            }

        }

        return false;

    }

</script>

</asp:Content>

```

Файл «Drugs.aspx»

```

using drugstore.Models.Repository;

using drugstore.Pages.Helpers;

using drugstore.Models;

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace drugstore.Pages.Admin
{
    public partial class OrderPerson : System.Web.UI.Page
    {
        public decimal totalAll;
        Repository repository = new Repository();
        protected void Page_Load(object sender, EventArgs e)
        {
            if (IsPostBack)
            {
                int dispatchID;
                if (int.TryParse(Request.Form["dispatch"], out
dispatchID))
                {
                    Order myOrder = repository.Order.Where(o =>
o.OrderId == dispatchID).FirstOrDefault();
                    if (myOrder != null)
                    {
                        Drug drug = new Drug();
                        foreach (OrderLine orderL in
myOrder.OrderLines)
                        {
                            drug = repository.Drugs.Where(d =>
d.DrugID == orderL.Drug.DrugID).FirstOrDefault();
                            drug.CountDrug -= orderL.Quantity;
                            repository.SaveDrug(drug);
                        }
                    }
                }
            }
        }
    }
}

```

```

        myOrder.Dispatched = true;
        repository.SaveOrder(myOrder);
    }
}

}

public PersonL GetPersonLine()
{
    return SessionHelper.GetPerson(Session).Lines;
}

public IEnumerable<Order> GetOrder()
{
    return repository.Order.Where(o => o.UserId ==
GetPersonLine().User.UserId);
}

public decimal Total(IEnumerable<OrderLine> orderLines)
{
    decimal total = 0;
    foreach (OrderLine ol in orderLines)
    {
        total += ol.Drug.Price * ol.Quantity;
    }
    totalAll += total;
    return total;
}

public decimal Total1()
{
    return totalAll;
}
}
}

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

Файл «Profiles.aspx»

```

<%@ Page Title="" Language="C#"
MasterPageFile="~/Pages/Admin/Admin.Master" AutoEventWireup="true"
CodeBehind="Profiles.aspx.cs"
Inherits="drugstore.Pages.Admin.Profiles" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head"
runat="server">

</asp:Content>

<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <asp:ListView ID="ListView1"
ItemType="drugstore.Models.Users" SelectMethod="GetUsers"

        DataKeyNames="UserId" UpdateMethod="UpdateUser"
DeleteMethod="DeleteUser" EnableViewState="false"

        runat="server">

<LayoutTemplate>

    <div class="outerContainer">

        <table id="productsTable">

            <tr>

                <th>Имя и фамилия менеджера аптеки</th>

                <th>Адрес</th>

                <th>Город</th>

                <th>Действие</th>

            </tr>

            <tr runat="server" id="itemPlaceholder"></tr>

        </table>

    </div>

</LayoutTemplate>

<ItemTemplate>

    <tr>

        <td><%# Item.Name %></td>

        <td><%# Item.Line %></td>

        <td><%# Item.City %></td>

        <td bgcolor="white"><asp:Button class="button
button3" CommandName="Edit" Text="Изменить" runat="server" />

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

71

```

        <asp:Button class="button button3"
CommandName="Delete" Text="Удалить" runat="server" />

    </td>

</tr>

</ItemTemplate>

<EditItemTemplate>

    <tr>

        <td>

            <input name="name" value="<%# Item.Name %>" />

            <input type="hidden" name="UserId" value="<%#
Item.UserId %>" />

        </td>

        <td>

            <input name="description" value="<%# Item.Line
%>" /></td>

        <td>

            <input name="category" value="<%# Item.City
%>" /></td>

        <td>

            <asp:Button ID="Button3" class="button
button3" CommandName="Update" Text="Обновить" runat="server" />

            <asp:Button ID="Button4" class="button
button3" CommandName="Cancel" Text="Отмена" runat="server" />

        </td>

    </tr>

</EditItemTemplate>

</asp:ListView>

</asp:Content>

```

ПРИЛОЖЕНИЕ В
ПРЕЗЕНТАЦИЯ

ТЕМА РАБОТЫ:
«ПРОГРАММНОЕ
ОБЕСПЕЧЕНИЕ АПТЕЧНОГО
СКЛАДА»

АВТОР РАБОТЫ: СТУДЕНТ ГРУППЫ 14ВВ1 ЗИНОВЬЕВ А.В.

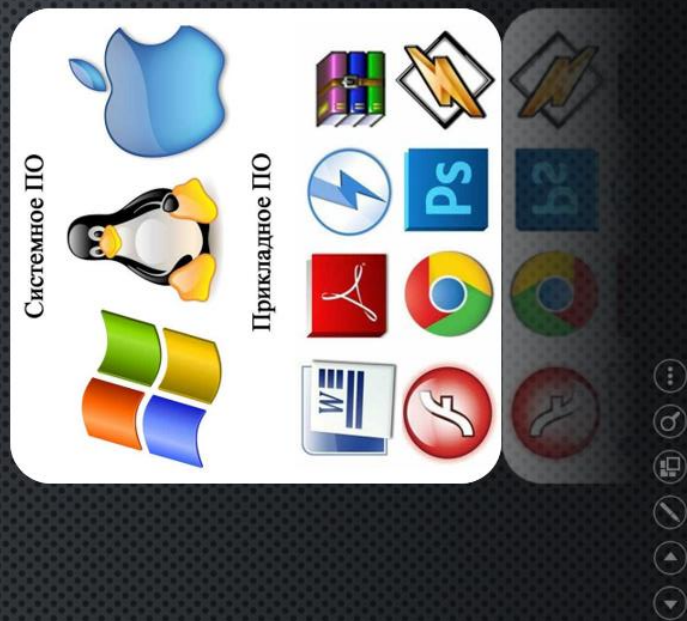
НАУЧНЫЙ РУКОВОДИТЕЛЬ: К.Т.Н., ДОЦЕНТ ПОПОВ К.В.

Изм.	Лист	№ докум.	Подпись	Дата

ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

ПРИКЛАДНАЯ ПРОГРАММА, ИЛИ ПРИЛОЖЕНИЕ, — ПРОГРАММА, ПРЕДНАЗНАЧЕННАЯ ДЛЯ ВЫПОЛНЕНИЯ ОПРЕДЕЛЁННЫХ ЗАДАЧ И РАССЧИТАННАЯ НА НЕПОСРЕДСТВЕННОЕ ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ.

ПРОГРАММЫ ОБРАБОТКИ ЗАКАЗОВ ИЛИ СОЗДАНИЯ СПИСКОВ РАССЫЛКИ — ПРИМЕР ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.

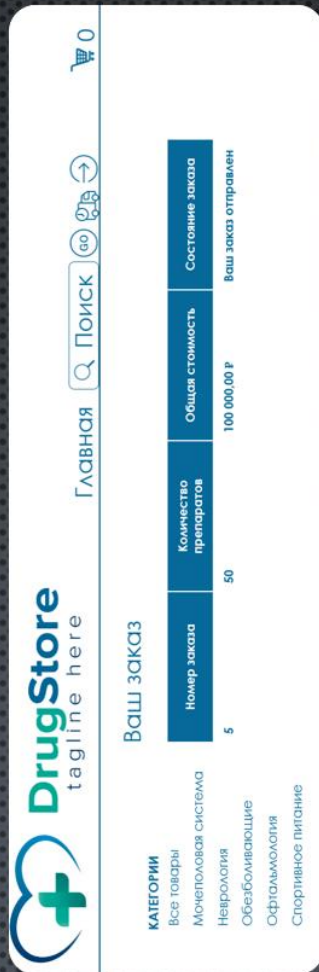


Изм.	Лист	№ докум.	Подпись	Дата

ЦЕЛЬ РАБОТЫ:

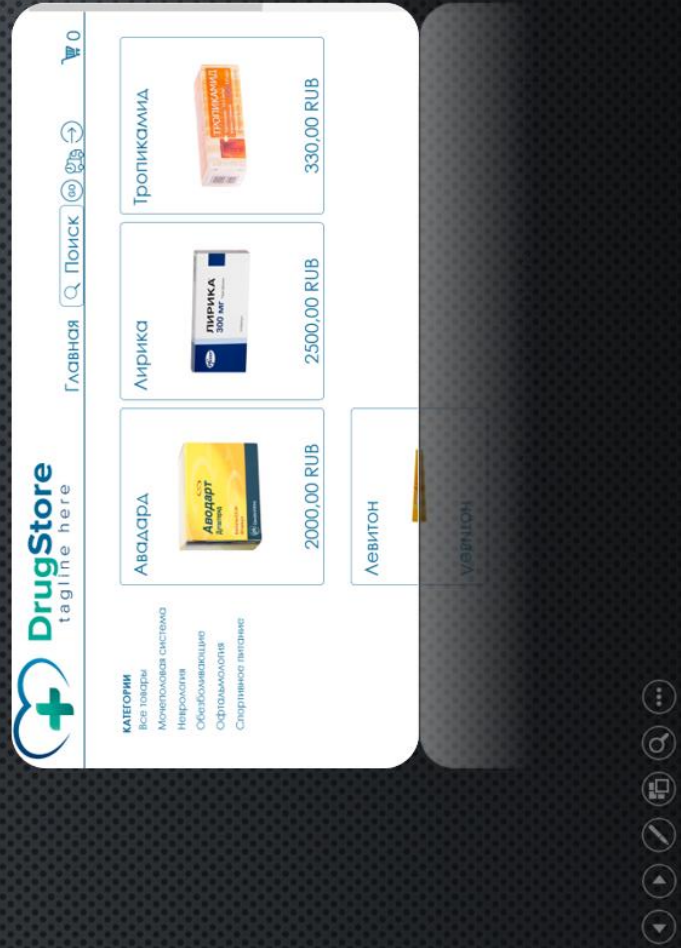
РАЗРАБОТАТЬ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, КОТОРОЕ ДОЛЖНО ОСУЩЕСТВЛЯТЬ СЛЕДУЮЩИЕ ФУНКЦИИ:

- УЧЕТ ТОВАРА НА СКЛАДЕ
- УЧЕТ ДВИЖЕНИЕ ТОВАРА
- УЧЕТ АПТЕЧНЫХ ПУНКТОВ
- ФОРМИРОВАНИЕ ЗАЯВОК НА ТОВАР
- ФОРМИРОВАНИЕ ОТЧЕТНОЙ ДОКУМЕНТАЦИИ



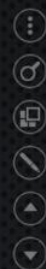
Скриншот программы

СИСТЕМА ДОЛЖНА СОСТОЯТЬ ИЗ: -ПОДСИСТЕМА АПТЕЧНЫЙ ПУНКТ -ПОДСИСТЕМА СКЛАД.



ПОДЗАДАЧИ:

- АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
- ОПИСАНИЕ МОДЕЛИ АППЕЧНОГО СКЛАДА, ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА И СВЯЗЬ С ТАБЛИЦАМИ БАЗЫ ДАННЫХ
- РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ



Изм.	Лист	№ докум.	Подпись	Дата

АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ:

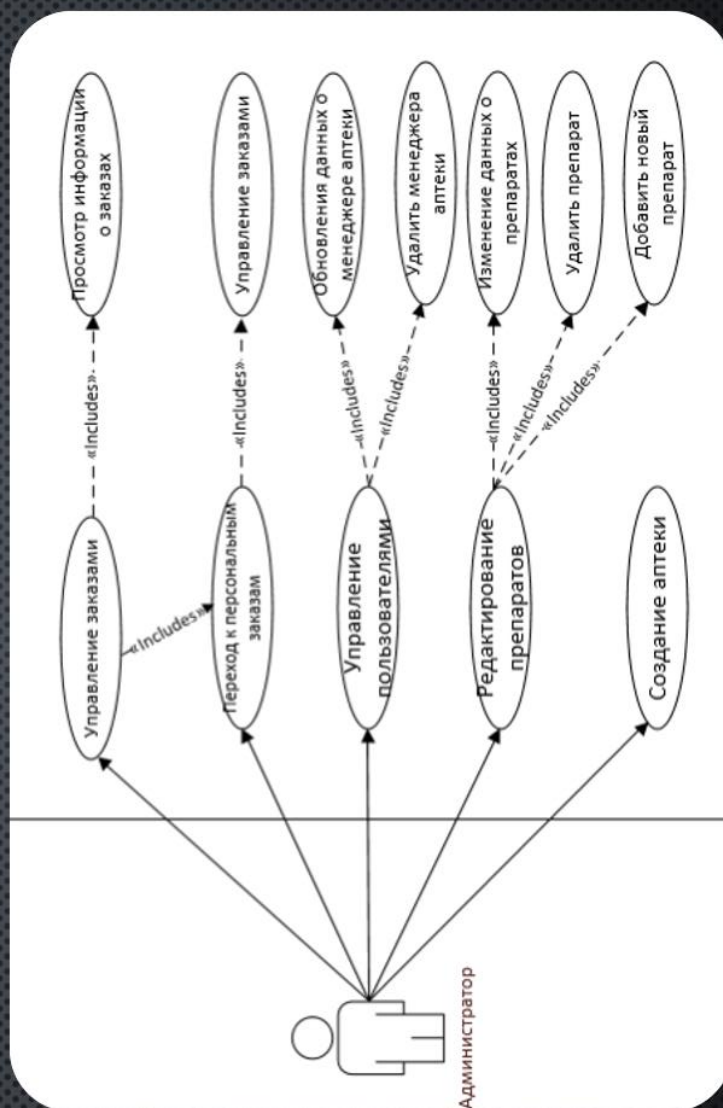


Изм.	Лист	№ докум.	Подпись	Дата

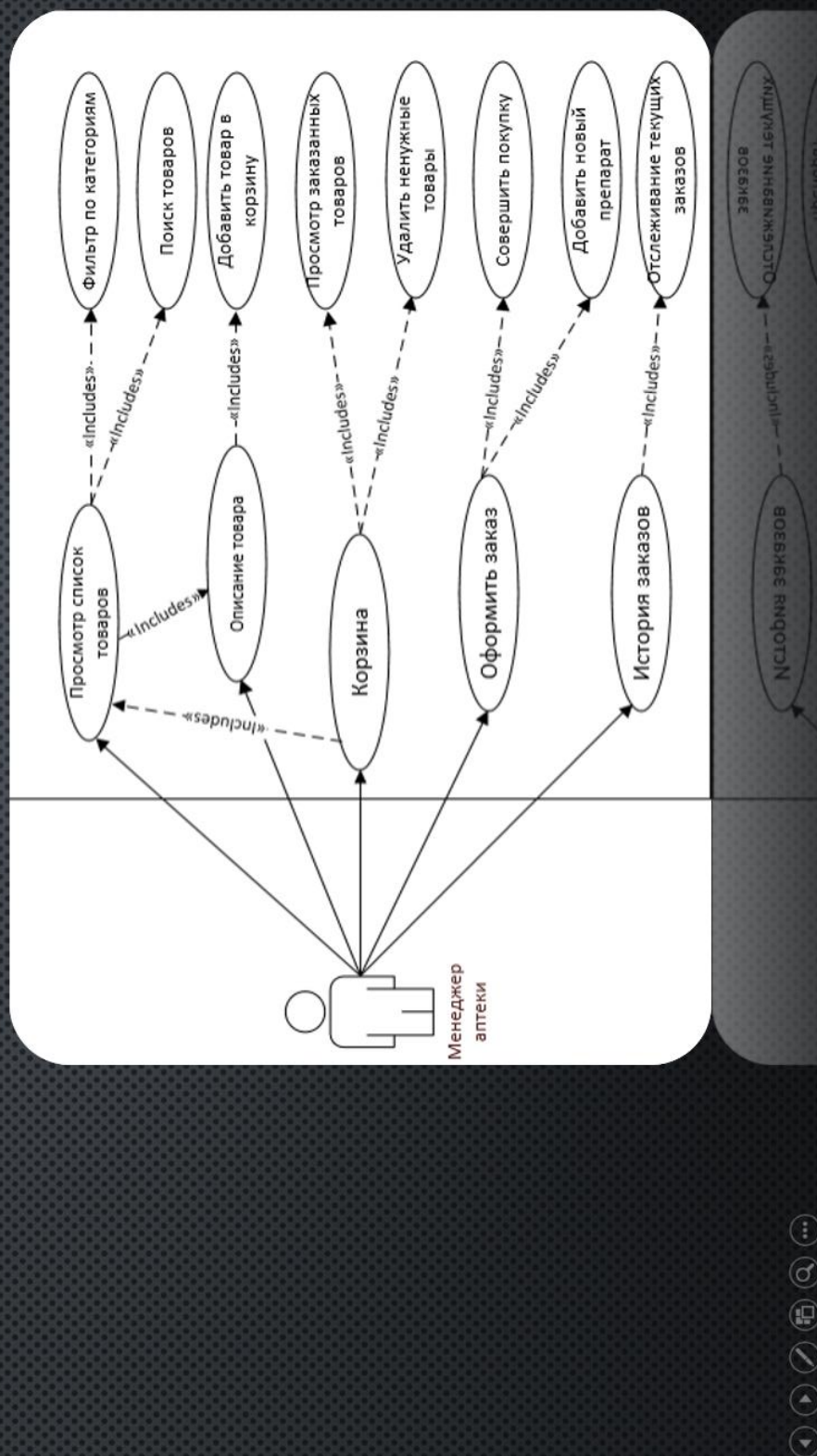
ПГУ 1.090301.10.001 ПЗ

МОДЕЛЬ АПТЕЧНОГО СКЛАДА:

• АДМИНИСТРАТОР СКЛАДА:



• МЕНЕДЖЕР АПТЕКИ:



Изм.	Лист	№ докум.	Подпись	Дата

СВЯЗЬ С ТАБЛИЦА БАЗЫ ДАННЫХ:

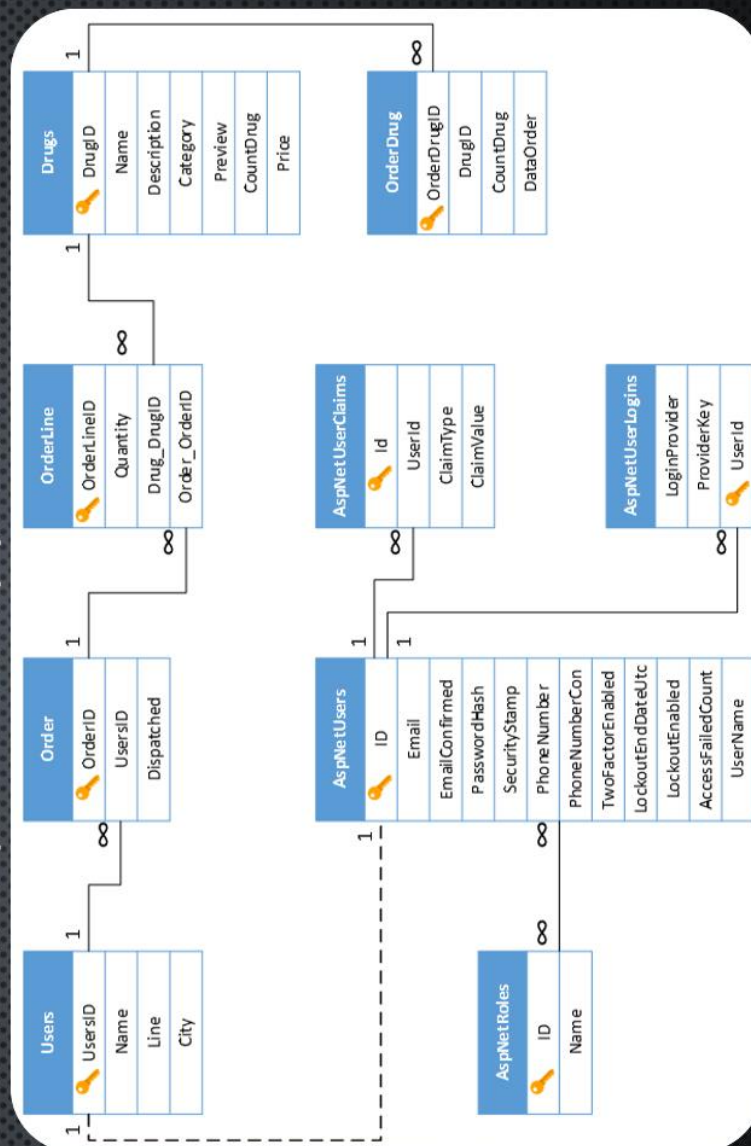
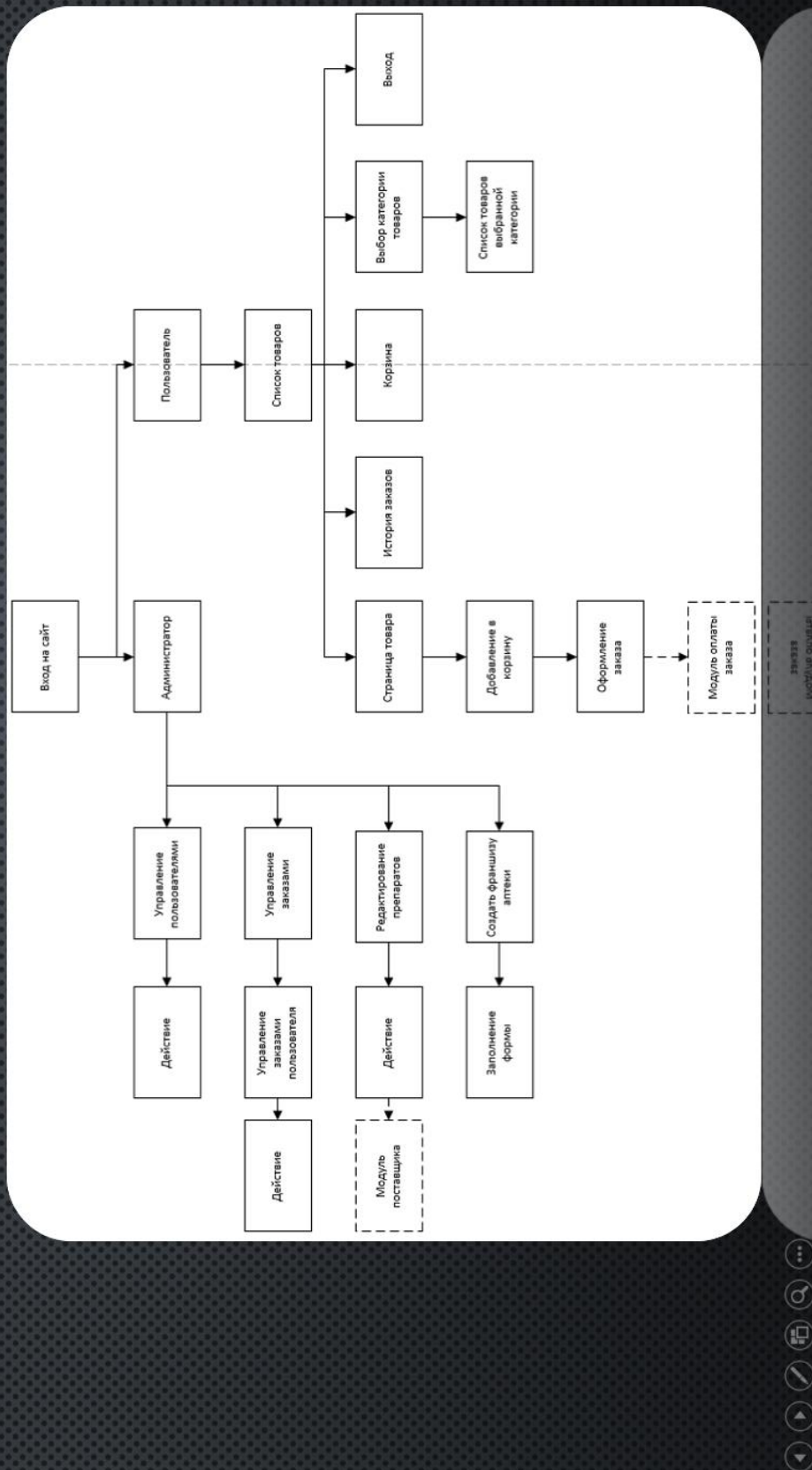


СХЕМА САЙТА ВЫГЛЯДИТ СЛЕДУЮЩИМ ОБРАЗОМ:



НЕСКОЛЬКО СКРИНШОТОВ САЙТА:



СПАСИБО ЗА ВНИМАНИЕ!

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

84