Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет Информационных Технологий и Управления

Кафедра Информационных Технологий Автоматизированных Систем

Дисциплина: Базы и Банки Данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ИНТЕРНЕТ-МАГАЗИН САДОВОГО ИНСТРУМЕНТА

БГУИР КП 1-53 01 02 023 ПЗ

Студент: гр. 820601 Судаков Б.Д.

Руководитель: Трофимович А.Ф.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики   
и радиоэлектроники»

Факультет информационных технологий и управления

УТВЕРЖДАЮ

Заведующий кафедрой

*––––––––––––––––––––*

(подпись)

–––––– –––––––––2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту    *Судакову Богдану Дмитриевичу–––––––––––––––––––*

1. Тема проекта   *Интернет-магазин садового инструмента.*

2. Срок сдачи студентом законченного проекта– –*4 мая 2021 г.–*

3. Исходные данные к проекту *создание сервиса, содержащего список товаров. На домашней странице должны присутствовать: ссылки на формы регистрации и входа в систему, корзина для оплаты покупок.При переходе в корзину необходимо иметь кнопку оплаты, а также страницу оплаты. Оплаченные продукты должны сохраняться в истории заказов, которая доступна каждому зарегестрированному пользователю. Пользователь-администратор может удалять, редактировать и добавлять информацию о товарах, также у него должна иметься кнопка для экспорта данных из БД в файл формата json.*

*Устройство должно быть подключено к сети. Скорость передачи информации должна быть выше 0.5 мегабит в секунду. На устройстве требуется иметь браузер для просмотра сайта. Требование к браузеру: браузер Internet Explorer 10 и выше, все остальные браузеры последних либо предпоследних версий.*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке) *Введение. 1. Анализ предметной области. 2. Проектирование системы. 3. Описание процесса реализации. 4. Руководство пользователя. Приложение А (справочное). Приложение Б (справочное).*

5. Консультант по проекту (с обозначением разделов проекта) *А.Ф. Трофимович–*

7. Дата выдачи задания –––– –*3 февраля 2021 г.–––––––––––––––––––*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*раздел 1-2 к 22.02 – 30 %; ––––––––––––––––– ––––––––––      –\_*

*раздел 3-4 к 23.03 – 50 %; –– –––––––––––––– ––––––––––––––––––*

*оформление пояснительное записки к 20.04 – 80 %; –– ––––––––––––––*

*Защита курсового проекта с 04.05 по 18.05.2021г.––––––––––––––––––*

Руководитель*– –––––––– А.Ф. Трофимович*

(подпись)

Задание принял к исполнению *–––––––\_\_\_\_\_\_\_\_\_\_\_ –– Б. Д. Судаков*

(подпись)

РЕФЕРАТ

ИНТЕРНЕТ-МАГАЗИН САДОВОГО ИНСТРУМЕНТА: курсовой проект / Б. Д. Судаков. – Минск: БГУИР, 2021, – п.з. – 54 с.

Основной задачей курсового проекта является создание интернет магазина садового инструмента, где пользователь может просматривать каталог товаров, добавлять и удалять товары из корзины и оформлять заказы.

Для разработки серверной и клиентской частей проекта системы был выбран *С#*, который является языком программирования и поддерживает объектно-ориентированный, императивный и функциональный стили.

В результате выполнения курсовой работы было разработано приложение, а также написан глоссарий. В первом разделе описывается текущая ситуация на рынке решений для контроля качества, приводится описание аналогов системы, а также достоинств и недостатков приложений.

Во втором разделе отображена бизнес-логика приложения, а также приведены диаграммы вариантов использования для каждой роли. Приведена схема базы данных.

В третьем разделе описан процесс реализации, начиная с выбора языка программирования, заканчивая примерами кодам.

В четвёртом разделе представлено руководство пользователя, где пошагово расписаны возможные варианты работы в приложении.

Заключение включает в себя основные выводы о проделанной работе.

содержание

Введение 7

1 Анализ предметной области 9

1.1 Рынок 9

1.2 Аналоги 9

1.3 Постановка задачи 11

2 Проектирование системы 13

2.1 Проектирование пользовательского интерфейса 13

2.2 Проектирование бизнес-логики 14

2.3 Проектирование базы данных 16

3 Описание процесса реализации 20

3.1 Выбор СУБД 20

3.2 Выбор языка программирования 24

3.3 Реализация интернет-магазина 25

4 Руководство пользователя 37

Заключение 42

Список использованных источников 43

Приложение А (справочное) Выдержка из кода 44

Приложение Б (справочное) Таблицы созданной базы данных 53

Ведомость курсового проекта 54

# Введение

Интернет-магазины – это форма электронной торговли, которая позволяет потребителям напрямую покупать товары или услуги у продавца через Интернет, используя веб-браузер. Подобная форма значительно упрощает процесс приобретения товаров, их заказ и связь с продавцом. Однако немаловажную часть играет доставка непосредственно покупателю. Исходя из большого количества городов и мест, куда товар должен быть доставлен в кратчайшие сроки, возникает вопрос о создании компании, которая будет осуществлять доставку в установленное время и с оптимальными затратами для реализации подобного рода услуг.

Несмотря на наличие многих бесплатных ресурсов в Интернете, интернет-магазины по-прежнему пользуются спросом у покупателей, так как обычно они гарантируют качество предлагаемой продукции. На примере постоянно обновляемого ассортимента интернет-магазинов, изменения в количестве пользователей можно достаточно полно изучить принципы создания динамических сайтов.

Веб-сайт интернет-магазина представляет собой систему, ориентированную в основном на клиента. Она должна предоставлять услуги по просмотру товаров, удобный поиск товаров, а также простоту и удобство при заказе товаров. В то же время, система должна быть ориентирована на персонал магазина и обращать внимание на новые поступления, заказы и продажи. Следует отметить, что сотрудники магазина не всегда являются квалифицированными программистами.

Практически каждая организация сегодня имеет личный сайт, на котором размещается информация о предлагаемых ею товарах и услугах. Это очень удобно, так как позволяет пользователю познакомиться с деятельностью компании и предлагаемым спектром услуг, не выходя из дома. Поэтому в настоящее время для многих организаций, индивидуальных предпринимателей и прочих, одной из главных задач является разработка такой информационной системы, которая смогла бы обеспечить максимально простой способ взаимодействия представителя компании и клиента через интернет. Люди приходят в интернет за покупками из-за того, что в онлайн-магазинах низкие цены и широкий выбор товаров [1].

В это же время актуальным становится вопрос о создании сайта для магазина садового инструмента. Это вызвано сезонным спросом на товары данной сфере, поэтому необходима система, в которой покупатель сможет сделать заказ удаленно и в кратчайшие сроки с целью получения необходимых инструментов.

Данный курсовой проект направлен на создание подобной базы данных и приложения, которые в последующем могут быть интегрированы в систему магазинов садового инструмента с последующей автоматизацией оформления заказов и упрощения коммуникации между клиентом и магазином в удаленном режиме.

# 1 Анализ предметной области

## Рынок

На сегодняшний день Интернет-торговля (*e-commerce*) во всем мире, а также в Беларуси демонстрирует высокие темпы роста. Дополнительный стимул для открытия интернет-магазина создает падение платежеспособности клиентов. В результате большинство потребителей ищут лучшее предложение в Интернете. В настоящее время, все большее количество людей приобретают продукцию в интернет-магазинах. Причинами являются низкие цены, удобство и возможность получить свой заказ дома. Основной задачей при открытии интернет-магазина является создание функционального сайта.

Количество интернет-магазинов в Беларуси за последние пять лет увеличилось более чем на 10 тыс. (или на 94%). По состоянию на 1 июля в Торговом реестре зарегистрирован 24021 интернет-магазин.

По оценкам *BusinessStat,* В 2015-2019 годах продажа садовых инструментов в Беларуси увеличилась на 49,9%. При этом наибольший прирост продаж наблюдался в 2017 году – на 35,4% к уровню предыдущего года.

## Аналоги

В наше время количество интернет-магазинов и потребность к онлайн шоппингу значительно растет. И это неудивительно, ведь людям намного проще зайти на сайт и посмотреть каталог товаров. Поэтому количество, а соответственно и конкуренция, растут. Однако в Беларуси существует небольшое количество интернет-магазинов садового инвентаря. Ниже перечислены самые популярные интернет-магазины, которыми сейчас часто пользуются.

**1.2.1** *Mdregion.by*

Это интернет магазин садового инвентаря, который позволяет просматривать каталог товаров, разделенный на категории. Есть возможность зарегистрироваться и добавить товары в корзину. Пользовательский интерфейс показан на рисунке 1.1



Рисунок 1.1 – Интернет магазин «mdregion.by»

**1.2.2** *21vek.by*

Это многофункциональный интернет магазин различных товаров, который позволяет просматривать каталог товаров, авторизоваться, добавить товар в корзину и оформит заказ. Пользовательский интерфейс показан на рисунке 1.2

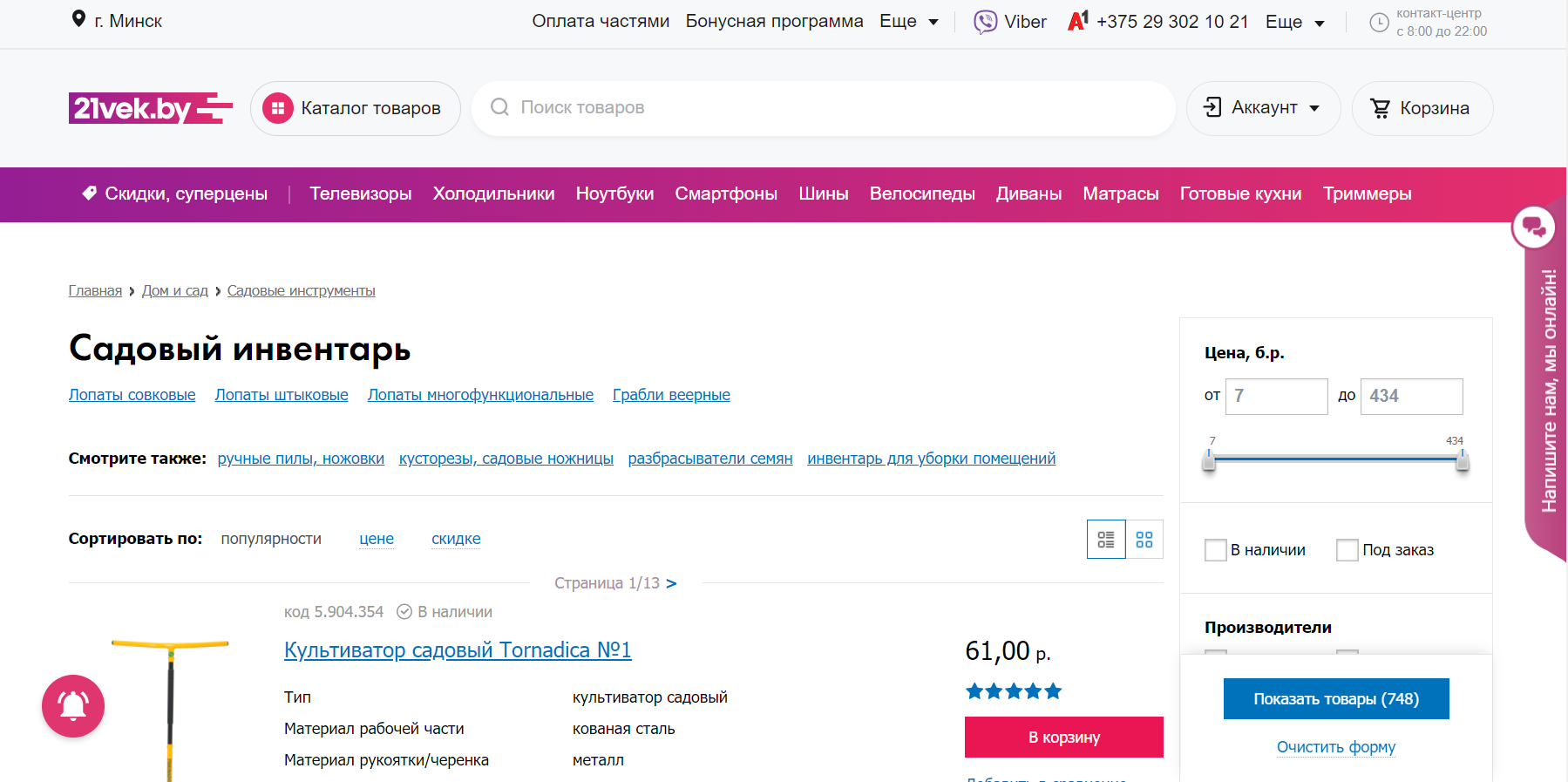


Рисунок 1.2 – Интернет магазин 21vek.by

**1.2.3** *Deal.by*

Это интернет магазин, в котором доступен просмотр каталога товаров, поиска необходимых товаров и добавление в корзину. На этом сайте предусмотрена возможность личного кабинета и входа в личный аккаунт. Пользовательский интерфейс этого музыкального интернет магазина представлен на рисунке 1.3.

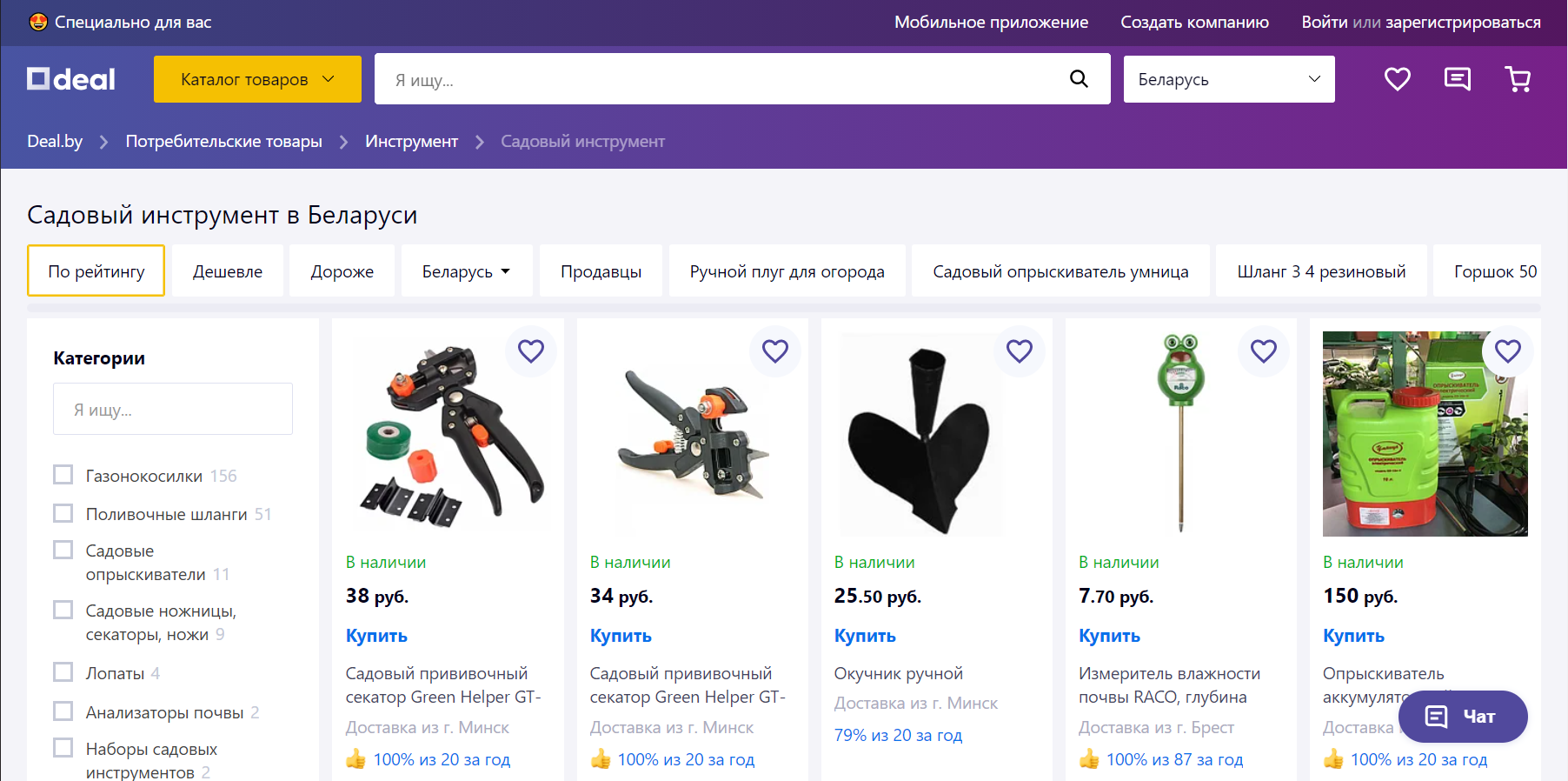


Рисунок 1.1 – Интернет магазин deal.by

## Постановка задачи

Постановка задачи важный этап создания сайта или интернет магазина. На этом этапе определяются основные параметры и задается направление разработки.

При разработке сайта или интернет-магазина важно учитывать следующие моменты:

* структура сайта должна быть грамотной и понятной для пользователя
* анонимность пользователей
* четкая информация о товарах (характеристика, параметры, цена, информация о изготовителе), магазине (расположение, время работы, контактные телефоны)
* возможность складывания товар в корзину
* возможность оформить заказ

В данном проекте любой зарегистрированный пользователей будет иметь роль *user.* Роль зарегистрированного пользователя предполагает собой возможность выполнения следующих действий:

* регистрация
* авторизация
* просмотр полного списка доступных товаров
* фильтрация товара по категориям
* поиск товара
* подробный просмотр каждого товара по отдельности
* добавление товара в корзину
* редактирование товара в корзине
* удаление товара из корзины
* просмотр полного списка товаров в корзине
* оформление заказа

Владелец интернет магазина имеет роль *admin*. Роль администратора предполагает собой возможность выполнения следующих действий:

* наполнение сайта (загрузка новых товаров, проверка характеристик)
* редактирование каталога товаров
* просмотр заказов
* обслуживание клиентов
* отслеживание заказов
* управление статусами заказов

У неавторизированных пользователей имеется ограниченный доступ к сайту, а именно только просмотр каталога товаров.

# 2 Проектирование системы

Данное программное обеспечение включает в себе три части:

* бизнес-логика
* серверная часть, *Back-end*
* клиентская часть, пользовательский интерфейс, *Front-end*

Взаимодействие программного обеспечения между клиентом и сервером происходит по архитектуре «клиент-сервер». Веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер (в широком смысле).

Основная часть приложения, как правило, находится на стороне веб-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный пользователю.

Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты (потребители этих функций) [1].

## 2.1 Проектирование пользовательского интерфейса

Пользовательский интерфейс (*user interface* или сокращенно *UI*) – это интерфейс, с помощью которого пользователь может управлять программным обеспечением или аппаратным оснащением. *UI* должен быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне [2].

Дизайн пользовательского интерфейса является фактором, оказывающим влияние на три основных показателя качества программного продукта: его функциональность, дизайн и производительность.

Чтобы интерфейс был эффективен и удобен, он должен обладать следующими характеристиками:

* назначение должно быть очевидно для пользователя
* процесс взаимодействия должен быть последовательным, простым и логичным
* дизайн должен быть узнаваем, для удобства пользователей.

В данной системе всё это реализуется за счет привычного и узнаваемого для пользователей интерфейса

## 2.2 Проектирование бизнес-логики

Бизнес-логика — в разработке информационных систем — совокупность правил, принципов, зависимостей поведения объектов предметной области (области человеческой деятельности, которую система поддерживает). Иначе можно сказать, что бизнес-логика — это реализация правил и ограничений автоматизируемых операций. Является синонимом термина «логика предметной области» (англ. *domain logic*) [3].

Логика сайта — фактически та же структура (структура сайта - скрытая от глаз посетителя часть. Отвечает за взаимодействие между собой различных модулей и компонентов сайта. Разрабатывая внутреннюю структуру, исполнитель принимает решение о том, как организовать информацию в базе и на сервере, каким образом всё это сгруппировать и разделить, какие модули для этого применять. На этом этапе решается каким образом реализовать тот или иной функционал.), только уже видимая посетителю и отвечает за взаимодействие и связи друг с другом различных разделов, категорий, страниц. То есть это уже структура выводимой информации. Именно во время этой работы становится ясно какими должны быть навигация, насколько понятен будет сайт посетителю [4].

Бизнес-логика веб-сайта может включать в себя такой рабочий процесс, как:

* система позволяет пользователю пройти регистрацию или авторизацию
* система проверяет данные, вводимые при авторизации на соответствие хранимых в базе данных логинов и паролей пользователей
* система производит валидацию вводимых параметров
* система позволяет просматривать товары, добавлять, редактировать и удалять товары из корзины
* система позволяет оформлять заказы
* для администратора, система позволяет добавлять, редактировать, удалять товары
* для администратора, система позволяет управлять статусами заказов
* для администратора, система позволяет просматривать заказы.

Логика интернет-магазина строится на основе требований, указанных выше и вся функциональность реализует их. На диаграммах описаны основные требования к системе для пользователя и для администратора. С помощью данных диаграмм можно понять, каким функционалом должно обладать приложение, какие действующие лица могут осуществлять взаимодействие с ним. Это поможет при составлении последующих диаграмм и моделировании всей системы. Также, это поможет для реализации кода программного обеспечения для разработчика. Более того, исходя из диаграмм, можно определить, на каких средствах будет размещена система. Основная логика взаимодействия пользователя с интернет-магазином отображается на рисунке 2.1.

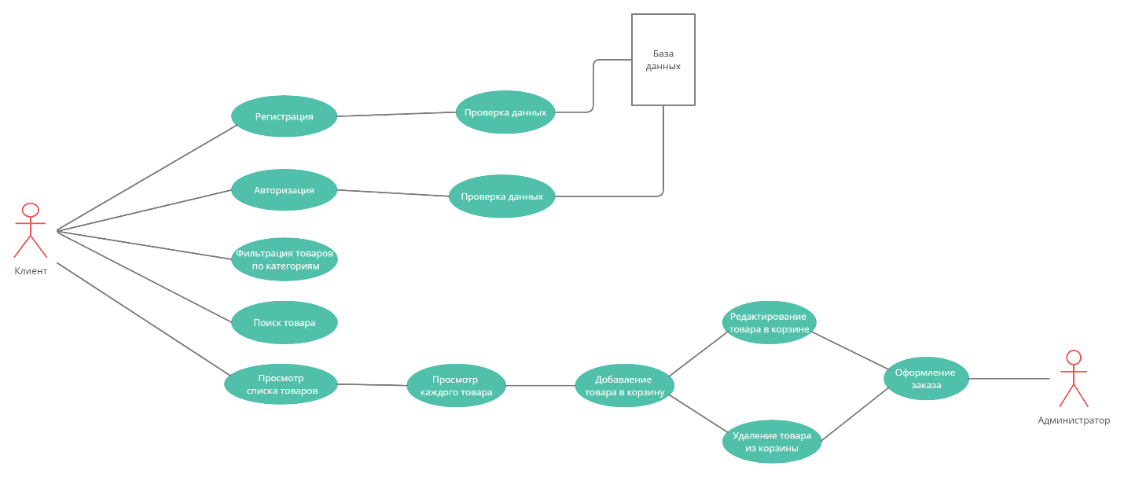


Рисунок 2.1 – Диаграмма вариантов использования пользователя

Основная логика взаимодействия администратора с интернет-магазином отображается на рисунке 2.2.

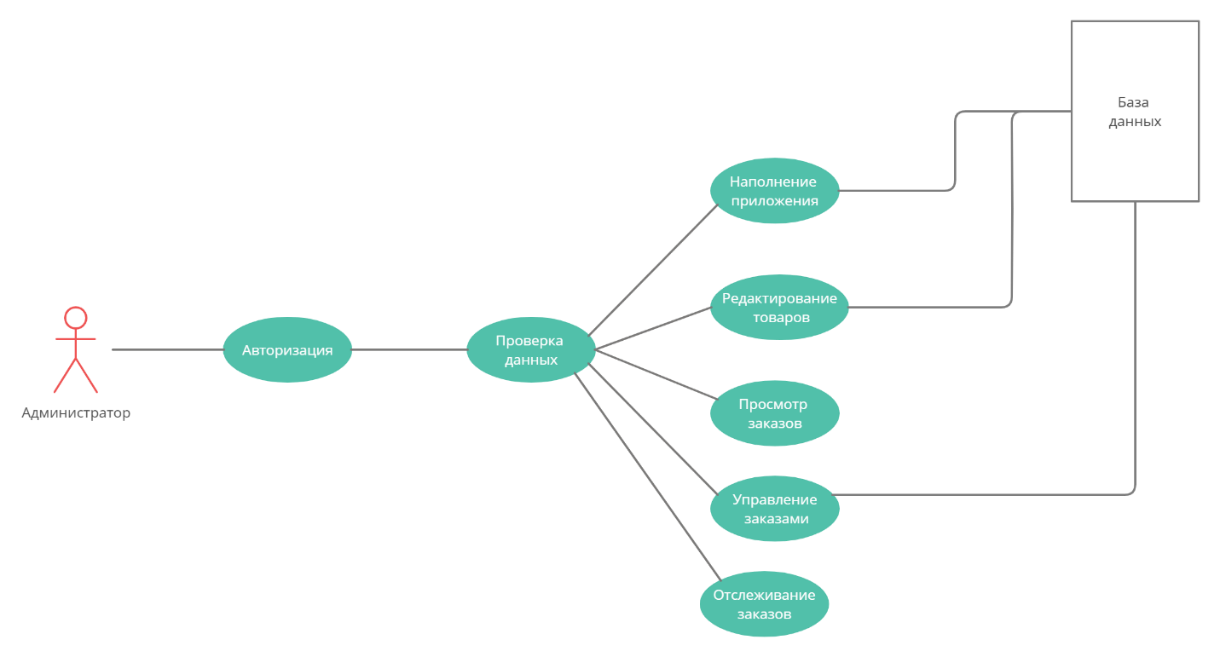


Рисунок 2.2 – Диаграмма вариантов использования администратора

Основная логика отображения устройств, на которых будет находится приложение, отображается на рисунке 2.3.

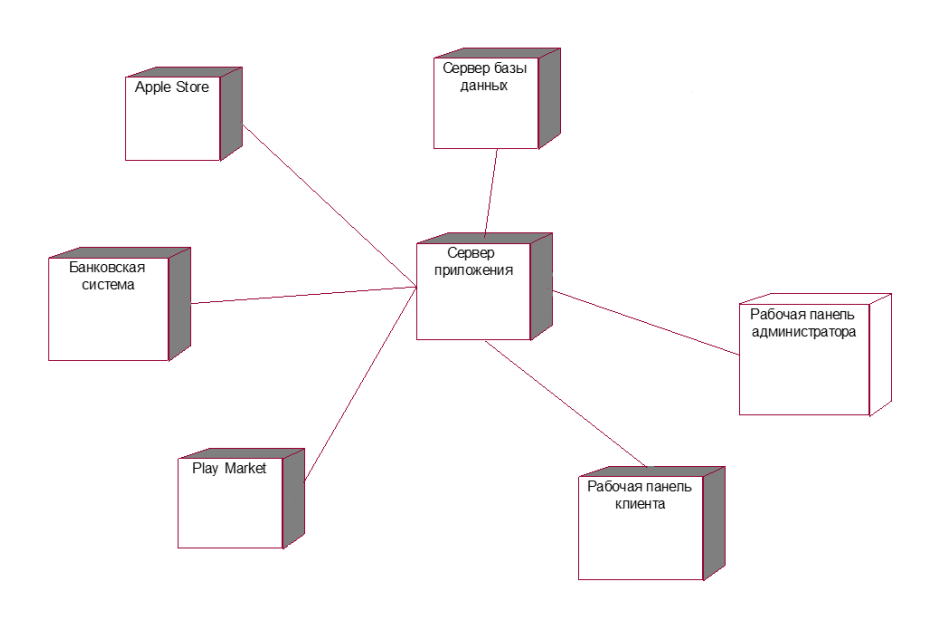


Рисунок 2.3 – Диаграмма размещения приложения

## 2.3 Проектирование базы данных

База данных - совокупность связанных данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. База данных является информационной моделью предметной области. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционная база данных – это набор данных с предопределенными связями между ними. Эти данные организованны в виде набора таблиц, состоящих из столбцов и строк. В таблицах хранится информация об объектах, представленных в базе данных. В каждом столбце таблицы хранится определенный тип данных, в каждой ячейке – значение атрибута. Каждая стока таблицы представляет собой набор связанных значений, относящихся к одному объекту или сущности. Каждая строка в таблице может быть помечена уникальным идентификатором, называемым первичным ключом, а строки из нескольких таблиц могут быть связаны с помощью внешних ключей [5].

Термин «реляционный» означает, что теория основана на математическом понятии отношение (*relation*). В качестве неформального синонима термину «отношение» часто встречается слово таблица. Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как абстрактное понятие, а визуальное представление отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию. Наиболее частая ошибка состоит в рассуждениях о том, что реляционная модель данных имеет дело с «плоскими», или «двумерными» таблицами, тогда как таковыми могут быть только визуальные представления таблиц.

Для описания сущностей и их атрибутов будем использовать таблицы:

Таблица 2.1 – Описание сущности «*User»*

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор пользователя |
| 2 | *Name* | Имя клиента |
| 3 | *Email* | Электронная почта клиента |
| 4 | *Login* | Логин пользователя |
| 5 | *Password* | Пароль пользователя |

Таблица 2.2 – Описание сущности «*Category*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор категории |
| 2 | *CategoryName* | Название категории |
| 3 | *Description* | Описание категории |
| 4 | *GardeningItems* | Товары, относящиеся к данной категории |

Таблица 2.3 – Описание сущности «*GardeningItem*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор товара |
| 2 | *Name* | Название товара |
| 3 | *ShortDesc* | Краткое описание товара |
| 4 | *LongDesc* | Подробное описание товара |
| 5 | *Img* | Изображение товара |
| 6 | *Price* | Цена товара |
| 7 | *IsFavourite* | Проверка на добавление товара в избранное |
| 8 | *Available* | Проверка на наличие товара в продаже |
| 9 | *CategoryID* | Уникальный идентификатор категории, к которой относится товар |
| 10 | *Category* | Название категории, к которой относится товар |

Таблица 2.4 – Описание сущности «*Order*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор заказа |
| 2 | *Name* | Имя заказчика |
| 3 | *Surname* | Фамилия заказчика |
| 4 | *Address* | Адрес доставки |
| 5 | *Phone* | Контактный телефон заказчика |
| 6 | *Email* | Электронная почта заказчика |
| 7 | *OrderTime* | Дата оформления заказа |
| 7 | *OrderDetail* | Подробности заказа |

Таблица 2.5 – Описание сущности «*OrderDetail*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор подробности заказа |
| 2 | *OrderID* | Уникальный идентификатор заказа |
| 3 | *ItemID* | Уникальный идентификатор товара |
| 4 | *Price* | Стоимость товара |
| 5 | *Item* | Название товара |
| 6 | *Order* | Ссылка на заказ |

Стоит отметить, что в таблице 2.6 для создания большей безопасности реализовано шифрование.

Таблица 2.6 – Описание сущности «*ShopCart*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *ShopCartId* | Уникальный идентификатор корзины |
| 2 | *listShopItems* | Список товаров в корзине |

Таблица 2.7 – Описание сущности «*ShopCartItem*»

|  |  |  |
| --- | --- | --- |
| № | Имя атрибута | Описание атрибута |
| 1 | *Id* | Уникальный идентификатор товара в корзине |
| 2 | *Item* | Название товара в корзине |
| 3 | *Price* | Стоимость товара в корзине |
| 4 | *ShopCartId* | Уникальный идентификатор корзины, в которой находится товар |

Требуется соблюсти соответствие базы данных трём нормальным формам. Нормальная форма – требование, предъявляемое к структуре таблицы теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц).

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК). Для соблюдения этой нормальной формы в таблицах «*User*», «*Order*», «*ShopCart*», «*OrderDetail*», «*Category*» и «*GardeningItem*» создаются внешние ключи, а их сочетание составляет уникальный индекс. Таким образом, невозможна функциональная зависимость не ключевого атрибута от части первичного ключа.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых относится к нескольким записям таблицы, в отдельные таблицы. Иными словами, неключевые столбцы не должны пытаться играть роль ключа в таблице, т.е. они действительно должны быть неключевыми столбцами, такие столбцы не дают возможности получить данные из других столбцов, они дают возможность посмотреть на информацию, которая в них содержится, так как в этом их назначение.

Схема базы данных представлена на рисунке 2.3.

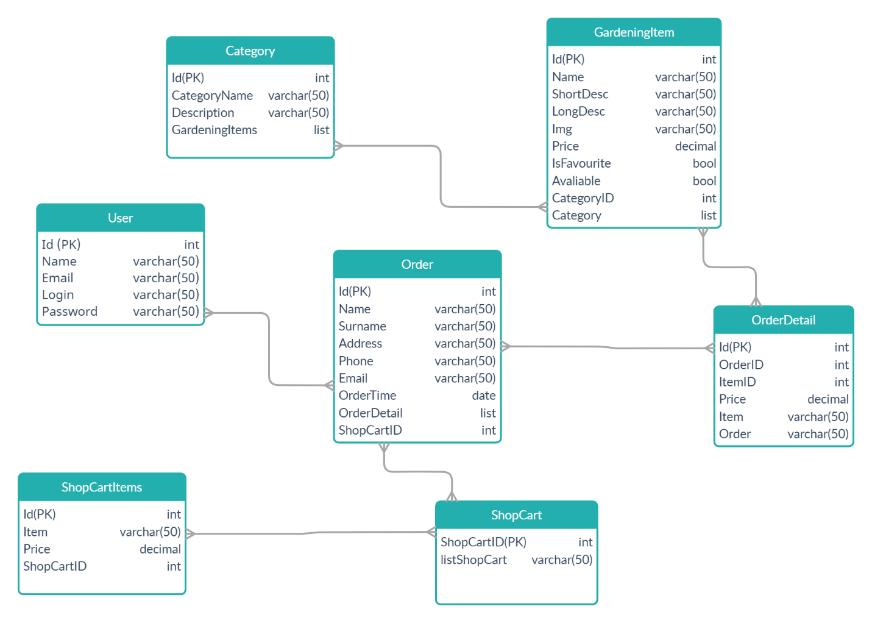


Рисунок 2.3 – Схема базы данных ShopGarden

Исходя из спроектированной базы данных нашего магазина, ставится задача нормализации, заполнение необходимыми данными и обеспечения необходимых связей между таблицами, что позволяет сохранить целостность данных и предотвратить потерю части данных .

# 3 Описание процесса реализации

## 3.1 Выбор СУБД

Несмотря на то, что все системы управления базами данных выполняют одну и ту же основную задачу (т.е. дают возможность пользователям создавать, редактировать и получать доступ к информации, хранящейся в базах данных), сам процесс выполнения этой задачи варьируется в широких пределах. Кроме того, функции и возможности каждой СУБД могут существенно отличаться. Различные СУБД документированы по-разному: более или менее тщательно. По-разному предоставляется и техническая поддержка.

При сравнении различных популярных баз данных, следует учитывать, удобна ли для пользователя и масштабируема ли данная конкретная СУБД, а также убедиться, что она будет хорошо интегрироваться с другими продуктами, которые уже используются. Кроме того, во время выбора следует принять во внимание стоимость системы и поддержки, предоставляемой разработчиком.

Сравниваются некоторые из популярных СУБД и на результате данного анализа выбирается подходящая для данного проекта.

**3.1.1** *Microsoft SQL Server*

Это система управления базами данных, движок которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов одновременно. Вскоре после выпуска *Microsoft SQL Server 2016*, *Microsoft* адаптировала продукт для операционной системы *Linux*, а на *Windows*-платформе он работал изначально. Одной из уникальных особенностей версии 2016 года является *temporal data support* (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия *Microsoft SQL*-сервер поддерживает *dynamic data masking* (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства:

* продукт очень прост в использовании;
* текущая версия работает быстро и стабильно;
* движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов;
* вы сможете получить доступ к визуализации на мобильных устройствах;
* он очень хорошо взаимодействует с другими продуктами *Microsoft*.

Недостатки:

* цена для юридических лиц оказывается неприемлемой для большей части организаций;
* даже при тщательной настройке производительности корпорация *SQL Server* способен занять все доступные ресурсы;
* сообщается о проблемах с использованием службы интеграции для импорта файлов;
* идеально подходит для крупных организаций, которые уже используют ряд продуктов *Microsoft*.

**3.1.2** *MongoDB*

Бесплатная база данных, которая имеет коммерческую версию - *MongoDB*, она предназначена для приложений, которые используют как структурированные, так и неструктурированные данные. Ядро является очень гибким и работает при подключении базы данных к приложениям через драйверы *MongoDB*. Существует широкий выбор доступных драйверов, поэтому легко найти драйвер, который будет работать с требуемым языком программирования.

Поскольку изначально система *MongoDB* не была разработана для обработки моделей реляционных данных (хотя может это выполнять), могут возникнуть проблемы производительности, если вы попытаетесь использовать её таким образом. Однако, движок предназначен для обработки различных данных, которые нельзя отнести к реляционным, и может хорошо справляться там, где другие движки работают медленно или бессильны.

*MongoDB* 3.2 - имеет новую подключаемую систему движков хранения. Документы могут быть проверены в процессе обновления или выполнения вставок, а функции текстового поиска были улучшены. Новая способность частичного индексирования может привести к более высокой производительности, уменьшая размер индексов.

Достоинства:

* скорость и простота в использовании;
* движок поддерживает *json* и другие традиционные документы *NoSQL*;
* данные любой структуры могут быть сохранены/прочитаны быстро и легко.

Недостатки:

* *SQL* не используется в качестве языка запросов;
* инструменты для перевода *SQL*-запросов в *MongoDB* доступны, но их следует рассматривать именно как дополнение;
* программа установки может занять много времени;
* подходит для организаций, работающих с разнородными данными, которые тяжело поддаются классификации. Для внедрения потребуются высококлассные специалисты.

**3.1.3** *MySQL*

*MySQL* - одна из самых популярных баз данных для веб-приложений. Фактически, является стандартом для веб-серверов, которые работают под управлением операционной системы *Linux*. *MySQL* - это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством и недостатком - в зависимости от области внедрения.

Разработку и поддержку *MySQL* осуществляет корпорация *Oracle*, получившая права на торговую марку вместе с поглощённой *Sun Microsystems*, которая ранее приобрела шведскую компанию *MySQL* *AB*. Продукт распространяется как под *GNU General Public License*, так и под собственной коммерческой лицензией [6]. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Эта СУБД позволяет выбирать различные движки для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. Гибкость СУБД *MySQL* обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа *MyISAM*, поддерживающие полнотекстовый поиск, так и таблицы *InnoDB*, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД *MySQL* поставляется со специальным типом таблиц *EXAMPLE*, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и *GPL*-лицензированию, в СУБД *MySQL* постоянно появляются новые типы таблиц. Она также имеет простой в использовании интерфейс, и пакетные команды, которые позволяют удобно обрабатывать огромные объемы данных. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства:

* распространяется бесплатно;
* прекрасно документирована;
* предлагает много функций, даже в бесплатной версии;
* пакет *MySQL* включен в стандартные репозитории наиболее распространённых дистрибутивов операционной системы *Linux*, что позволяет устанавливать её элементарно;
* поддерживает набор пользовательских интерфейсов;
* может работать с другими базами данных, включая *DB2* и *Oracle*.

Недостатки:

* придётся потратить много времени и усилий, чтобы заставить *MySQL* выполнять несложные задачи, хотя другие системы делают это автоматически, например, создавать инкрементные резервные копии;
* отсутствует встроенная поддержка *XML* или *OLAP*;
* для бесплатной версии доступна только платная поддержка;
* идеально подходит для: организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

## 3.2 Выбор языка программирования

Данный проект реализуется на языках программирования *С#* (клиентская часть) и фреймворка *ASP .NET Core* (серверная часть)*.* Также применяется язык разметки *HTML.* Ниже приведены описания данных технологий

**3.2.1 *С#***

*C#* — это объектно- и компонентно-ориентированный язык программирования. *C#* предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому *C#* подходит для создания и применения программных компонентов. С момента создания язык *C#* обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

Программы *C#* выполняются в *.NET*, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (*CLR*) и набор библиотек классов. Среда *CLR* — это реализация общеязыковой инфраструктуры языка (*CLI*), являющейся международным стандартом, от корпорации Майкрософт. *CLI* является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке *C#* компилируется в промежуточный язык (*IL*), который соответствует спецификациям *CLI*. Код на языке *IL* и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением *.dll*. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки.

**3.2.2 *ASP .NET Core***

*ASP.NET Core* может работать поверх кросс-платформенной среды *.NET Core*, которая может быть развернута на основных популярных операционных системах: *Windows*, *Mac OS*, *Linux*. И таким образом, с помощью *ASP.NET Core* мы можем создавать кросс-платформенные приложения. И хотя *Windows* в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь уже мы не ограничены только этой операционной системой. То есть мы можем запускать веб-приложения не только на ОС *Windows*, но и на *Linux* и *Mac OS*. А для развертывания веб-приложения можно использовать традиционный *IIS*, либо кросс-платформенный веб-сервер *Kestrel*.

Благодаря модульности фреймворка все необходимые компоненты веб-приложения могут загружаться как отдельные модули через пакетный менеджер *Nuget*. Кроме того, в отличие от предыдущих версий платформы нет необходимости использовать библиотеку *System.Web.dll*.

*ASP.NET Core* включает в себя фреймворк *MVC*, который объединяет функциональность *MVC*, *Web API* и *Web Pages*. В предыдущих версии платформы данные технологии реализовались отдельно и поэтому содержали много дублирующей функциональности. Сейчас же они объединены в одну программную модель *ASP.NET Core MVC*. А *Web Forms* полностью ушли в прошлое [7].

## 3.3 Реализация интернет-магазина

Данный проект базируется на архитектуре «клиент-сервер», главная идея которой состоит в разделении сетевого приложения на несколько компонентов, каждый из которых реализует специфический набор сервисов. Компоненты такого приложения могут выполняться на разных компьютерах, выполняя серверные и/или клиентские функции. Это позволяет повысить надежность, безопасность и производительность сетевых приложений и сети в целом. Архитектура «клиент-сервер» определяет общие принципы организации взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты, потребители этих функций.

Практические реализации такой архитектуры называются клиент-серверными технологиями. Каждая технология определяет собственные или использует имеющиеся правила взаимодействия между клиентом и сервером, которые называются протоколом обмена (протоколом взаимодействия).

**3.3.1 База данных**

Подключение базы данных к серверу осуществляется с помощью микросервиса:

*{*

*"ConnectionStrings":*

*{*

*"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=ShopGarden;Trusted\_Connection=True;MultipleActiveResultSets=true"*

*}*

*}*

**3.3.2 Модель таблиц базы данных**

Здесь приведено описание класса *Category*, в котором хранится информация о видах товаров.

*public class Category*

*{*

*public int Id { get; set; }*

*public string CategoryName { get; set; }*

*public string Desription { get; set; }*

*public List<GardeningItem> GardeningItems { get; set; }*

*}*

Ниже приведено описание класса *GardeningItem*, в котором хранится информация о имеющихся товарах.

*public class GardeningItem*

*{*

*public int Id { set; get; }*

*public string Name { set; get; }*

*public string ShortDesc { set; get; }*

*public string LongDesc { set; get; }*

*public string Img { set; get; }*

*public double Price { set; get; }*

*public bool IsFavourite { set; get; }*

*public bool Available { set; get; }*

*public int CategoryID { set; get; }*

*public virtual Category Category { get; set; }*

*}*

*}*

Ниже приведено описание класса *Order*, в котором хранится информация о сформированном заказе.

*public class Order*

*{*

*[BindNever]*

*public int id { get; set; }*

*[Required(ErrorMessage = "The field must be filled")]*

*[Display(Name = "Enter your name")]*

*[StringLength(30, MinimumLength = 3, ErrorMessage = "Line length must be between 3 and 30 characters")]*

*public string Name { get; set; }*

*[Display(Name = "Enter your surname")]*

*[StringLength(35, MinimumLength = 3, ErrorMessage = "Line length must be between 3 and 35 characters")]*

*[Required(ErrorMessage = "The field must be filled")]*

*public string Surname { get; set; }*

*[Display(Name = "Enter your adress")]*

*[StringLength(50, MinimumLength = 3, ErrorMessage = "Line length must be between 3 and 50 characters")]*

*[Required(ErrorMessage = "The field must be filled")]*

*public string Adress { get; set; }*

*[Display(Name = "Enter your phone number")]*

*[StringLength(30,MinimumLength = 7, ErrorMessage = "Line length must be between 7 and 30 characters")]*

*[DataType(DataType.PhoneNumber)]*

*[Required(ErrorMessage = "The field must be filled")]*

*public string Phone { get; set; }*

*[Display(Name = "Enter your Email")]*

*[StringLength(50, ErrorMessage = "the field should be no more than 50 characters")]*

*[DataType(DataType.EmailAddress)]*

*[Required(ErrorMessage = "The field must be filled")]*

*public string Email { get; set; }*

*[BindNever]*

*[ScaffoldColumn(false)]*

*public DateTime OrderTime { get; set; }*

*public List<OrderDetail> OrderDetails { get; set; }*

*public class OrderDetail*

*{*

*public int id { get; set; }*

*public int OrderID { get; set; }*

*public int ItemID { get; set; }*

*public double Price { get; set; }*

*public virtual GardeningItem Item { get; set; }*

*public virtual Order Order { get; set; }*

*}*

*}*

Ниже приведено описание класса *ShopCart*, который содержит информацию о выбранных пользователем товарах в рамках одной сессии, а также описаны методы взаимодействия с корзиной пользователя.

*public class ShopCart*

*{*

*AppDBContent appDBContent;*

*public ShopCart(AppDBContent appDBContent)*

*{*

*this.appDBContent = appDBContent;*

*}*

*public string ShopCartId { get; set; }*

*public List<ShopCartItem> listShopItems { get; set; }*

*public static ShopCart GetCart(IServiceProvider services)*

*{*

*ISession session = services.GetRequiredService<IHttpContextAccessor>()?.HttpContext.Session;*

*var context = services.GetService<AppDBContent>();*

*//CartId (key)*

*string shopCartId = session.GetString("CartId") ?? Guid.NewGuid().ToString();*

*session.SetString("CartId", shopCartId);*

*return new ShopCart(context) { ShopCartId = shopCartId };*

*}*

*public void AddToCart(GardeningItem item)*

*{*

*appDBContent.ShopCartItem.Add(new ShopCartItem*

*{*

*ShopCartId = ShopCartId,*

*Item = item,*

*Price = item.Price*

*});*

*appDBContent.SaveChanges();*

*}*

*public void DeleteToItem(ShopCartItem Item)*

*{*

*var item = Item;*

*appDBContent.ShopCartItem.Remove(item);*

*appDBContent.SaveChanges();*

*}*

*public List<ShopCartItem> getShopCartItems()*

*{*

*return appDBContent.ShopCartItem.Where(c => c.ShopCartId == ShopCartId).Include(s => s.Item).ToList();*

*}*

*public ShopCartItem getShopCartItem (int id)*

*{*

*return appDBContent.ShopCartItem.First(c => c.ShopCartId == ShopCartId && c.Item.Id == id);*

*}*

*}*

*}*

Ниже приведено описание класса *ShopCartItem*, который содержит информацию о товаре, находящийся в корзине.

*public class ShopCartItem*

*{*

*public int id { get; set; }*

*public GardeningItem Item { get; set; }*

*public double Price { get; set; }*

*public string ShopCartId { get; set; }*

*}*

Ниже приведено описание класса *User*, который содержит данные для личного кабинета пользователя и его контактные данные.

*public class User*

*{*

*public int Id { get; set; }*

*public string Name { get; set; }*

*public string Email { get; set; }*

*public string Password { get; set; }*

*public string Login { get; set; }*

*}*

**3.3.3 Авторизация пользователей**

При заполнении формы входа и нажатии кнопки *«Login»* будет вызван метод:

*public async Task<IActionResult> Login(LoginViewModel model)*

*{*

*if (ModelState.IsValid)*

*{*

*User user = \_iaccount.AllAccounts.FirstOrDefault(u => u.Login == model.Login && u.Password == model.Password);*

*if (user != null)*

*{*

*await Authenticate(model.Login); // аутентификация*

*return RedirectToAction("Index", "Home");*

*}*

*ModelState.AddModelError("", "Некорректные логин и(или) пароль");*

*}*

*return View(model);*

*}*

Если пользователь ввел некорректные данные, он получит уведомление об этом и останется на заполнении формы, пока не заполнит ее верно, либо пока не решит прекратить попытки войти в аккаунт.

Затем, если пользователь ввел корректные данные, он попадёт на следующую форму, после заполнения которой и нажатии кнопки *«Confirm»* будет вызван метод:

*private async Task Authenticate(string userName)*

*{*

*var claims = new List<Claim>*

*{*

*new Claim(ClaimsIdentity.DefaultNameClaimType, userName)*

*};*

*ClaimsIdentity id = new ClaimsIdentity(claims, "ApplicationCookie", ClaimsIdentity.DefaultNameClaimType, ClaimsIdentity.DefaultRoleClaimType);*

*await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(id));*

*}*

В случае, если у пользователя еще нет аккаунта, при нажатии кнопки регистрации вызывается метод:

*public async Task<IActionResult> Register(RegisterViewModel model)*

*{*

*if (ModelState.IsValid)*

*{*

*User user = \_iaccount.AllAccounts.FirstOrDefault(u => u.Login == model.Login);*

*if (user == null)*

*{*

*await \_iaccount.createUser(model);*

*await Authenticate(model.Login);*

*return RedirectToAction("Index", "Home");*

*}*

*else*

*ModelState.AddModelError("", "Некорректные логин и(или) пароль");*

*}*

*return View(model);*

*}*

В случае, если пользователь хочет выйти из своего аккаунта, при нажатии кнопки *Logout* вызывается метод:

*public async Task<IActionResult> Logout()*

*{*

*await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme;*

*return RedirectToAction("Login", "Account");*

*}*

**3.3.4 Добавление товара в корзину**

При просмотре товаров и дальнейшем их добавлении вызываются следующие методы:

*public ShopCartController(IItems itemRep, ShopCart shopCart)*

*{*

*\_itemRep = itemRep;*

*\_shopcart = shopCart;*

*}*

*public ViewResult Index()*

*{*

*ViewBag.Title = "basket";*

*var items = \_shopcart.getShopCartItems();*

*\_shopcart.listShopItems = items;*

*var obj = new ShopCartViewModel*

*{*

*shopCart = \_shopcart*

*};*

*return View(obj);*

*}*

*public RedirectToActionResult addToCart(int id)*

*{*

*var item = \_itemRep.AllGardeningItems.FirstOrDefault(i => i.Id == id);*

*if (item != null)*

*{*

*\_shopcart.AddToCart(item);*

*}*

*return RedirectToAction("Index");*

*}*

*public RedirectToActionResult deleteToItem(int id)*

*{*

*var item = \_shopcart.getShopCartItem(id);*

*if (item != null)*

*{*

*\_shopcart.DeleteToItem(item);*

*}*

*return RedirectToAction("Index");*

*}*

*}*

**3.3.5 Оформление заказа**

При нажатии кнопки *checkout* пользователь перенаправляется в форму оформления заказа, заполняет необходимые данные и при нажатии кнопки *complete order* вызываются следующие методы:

*public OrderController(IOrders orders, ShopCart shopCart)*

*{*

*this.orders = orders;*

*this.shopCart = shopCart;*

*}*

*public IActionResult Checkout()*

*{*

*return View();*

*}*

*[HttpPost]*

*public IActionResult Checkout(Order order)*

*{*

*shopCart.listShopItems = shopCart.getShopCartItems();*

*if (shopCart.listShopItems.Count == 0)*

*{*

*ModelState.AddModelError("order", "you must have goods");*

*ViewBag.Message = "you must have goods";*

*}*

*if (ModelState.IsValid)*

*{*

*orders.createOrder(order);*

*return RedirectToAction("Complete");*

*}*

*return View(order);*

*}*

*public ViewResult Complete()*

*{*

*ViewBag.Message = "order processed successfully";*

*return View();*

*}*

При этом происходит проверка корректности введённых данных, а именно проверка номера телефона на соответствующую длину, а также валидация введенной электронной почты.

# 4 Руководство пользователя

Ниже приведены все необходимые средства для работы пользователя с готовым продуктом.

Доступ к сайту получаем, произведя развертывание его на *localhost*.

Пользователь переходит на начальную страницу для авторизации или регистрации, приведенную на рисунке 4.1.

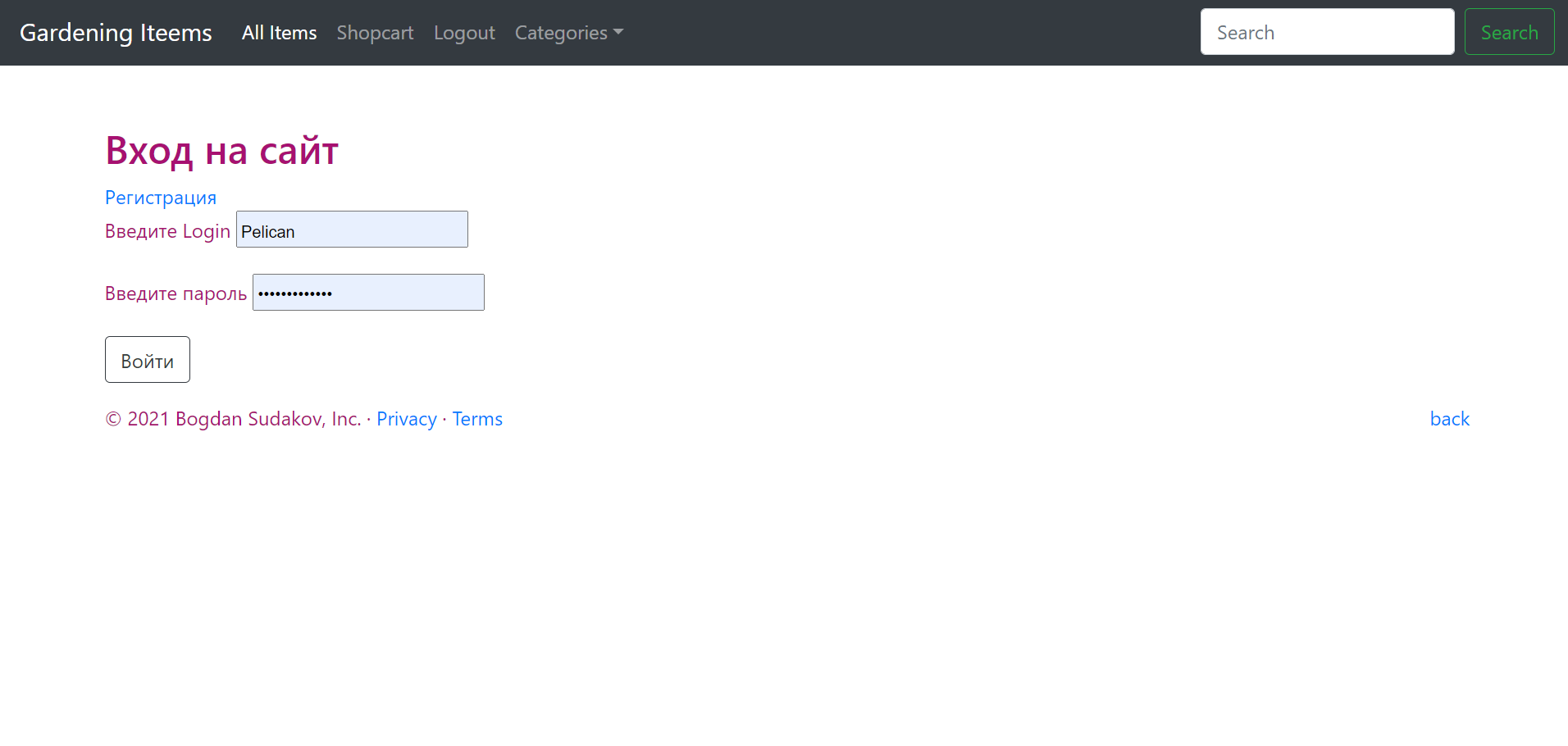


Рисунок 4.1 – Начальная страница

Если у пользователя ещё нет аккаунта, при нажатии кнопки «Регистрация» он перейдет на форму создания аккаунта, приведенную на рисунке 4.2.

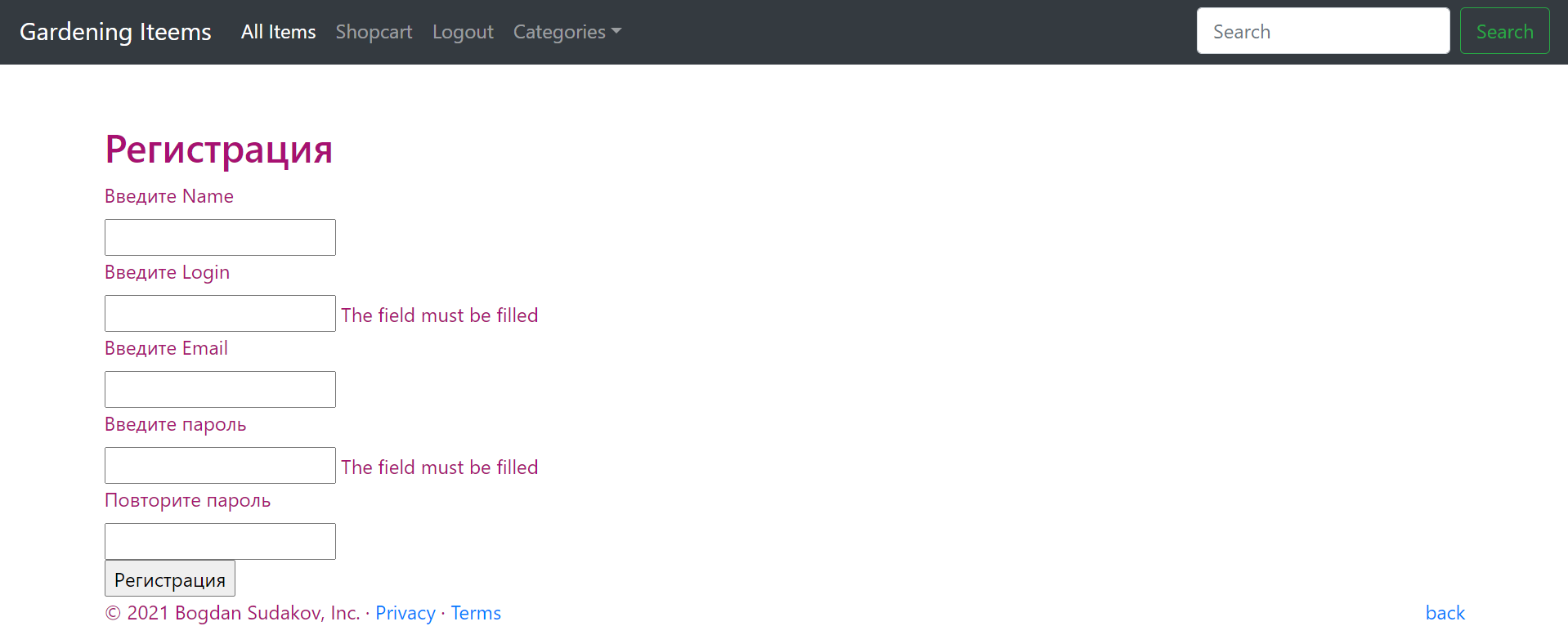


Рисунок 4.2 – Форма регистрации

При регистрации также предусмотрено соответствие логина и пароля определенно длине для увеличения безопасности личных данных пользователя. В случае ввода некорректных данных будет показано сообщение о необходимости выбрать другое название логина, пароля или почты, как показано на рисунке 4.3.

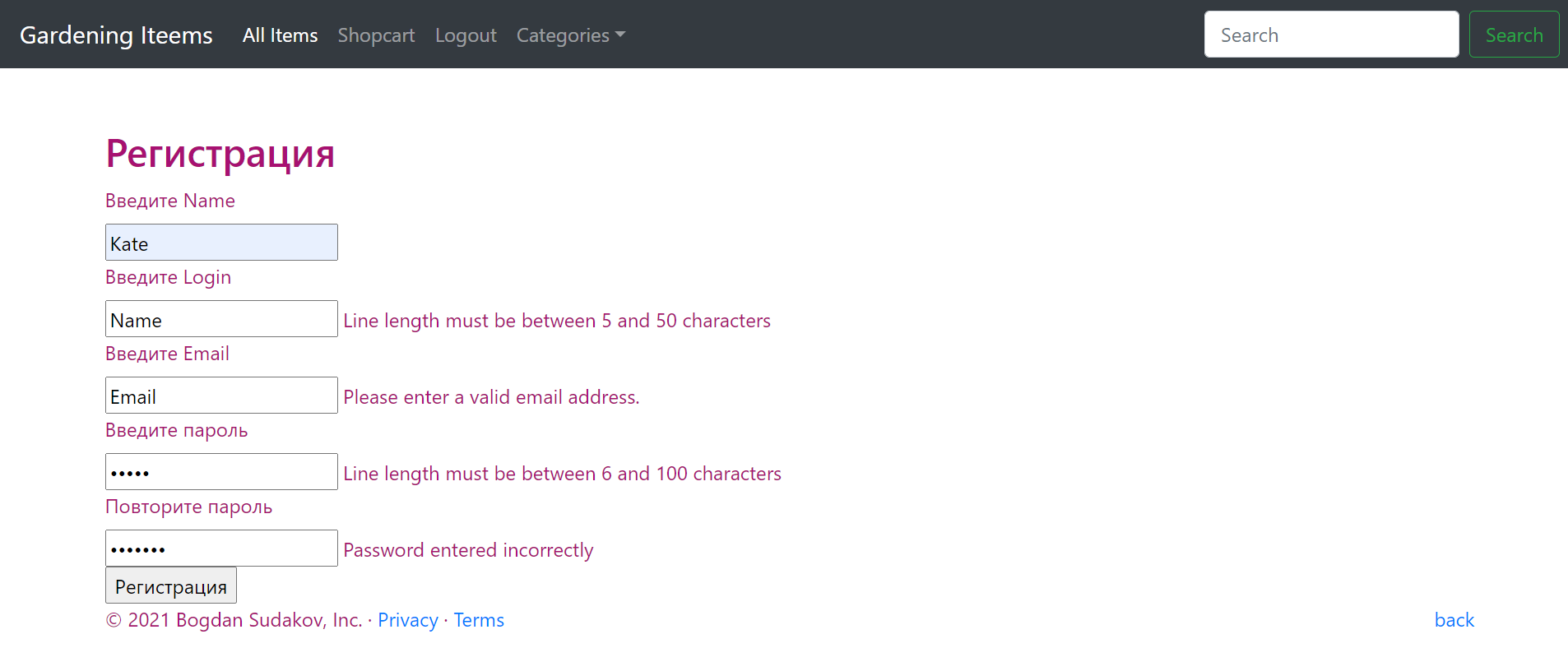


Рисунок 4.3 – Форма регистрации с сообщением о некорректных данных

После авторизации или перехода по вкладке *All Items* пользователь переходит к каталогу всех товаров, как показано на рисунке 4.4.

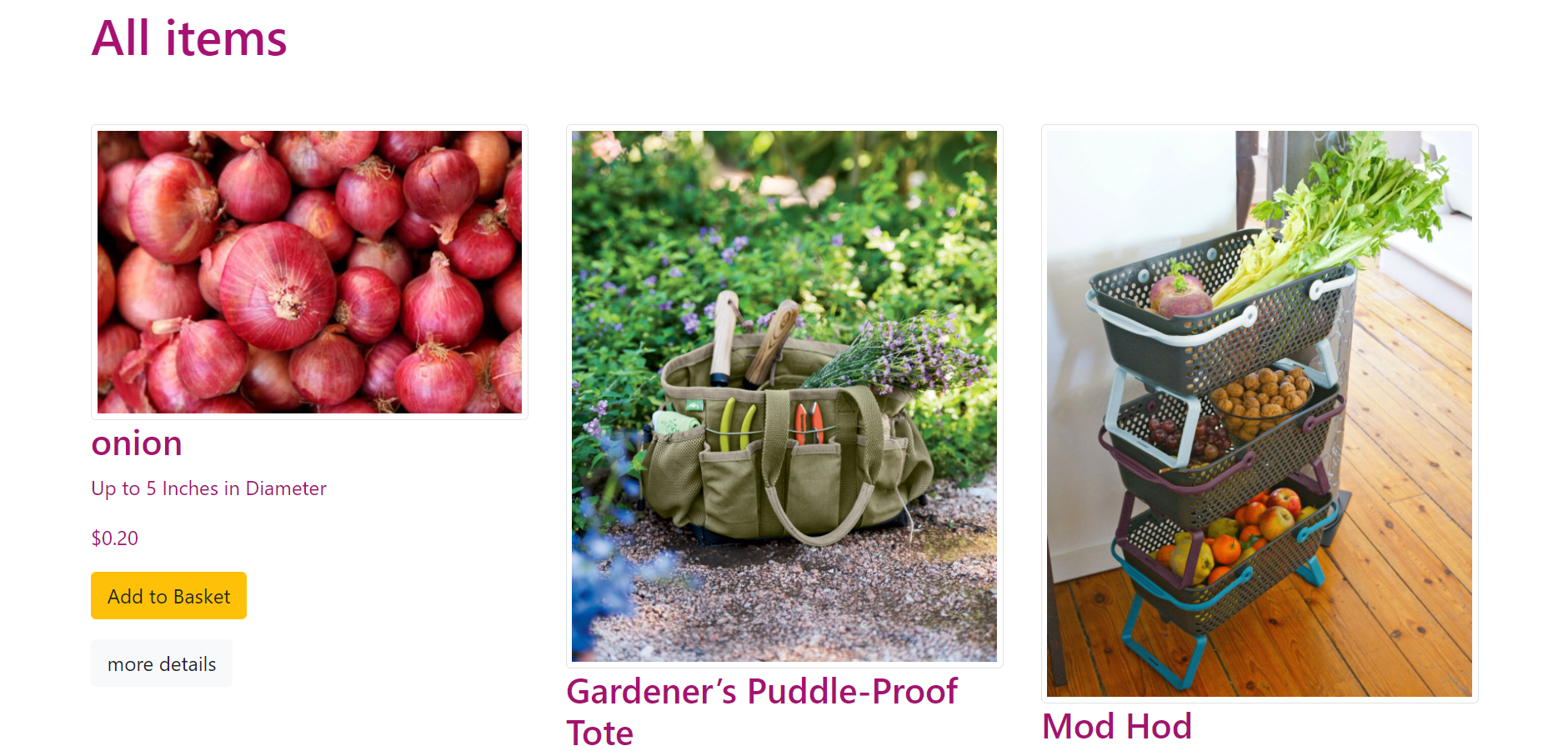


Рисунок 4.4 – Основной каталог товаров

В шапке сайта расположены кнопки, показанные на рисунке 4.5, для выбора категории товара для более точного поиска товаров.

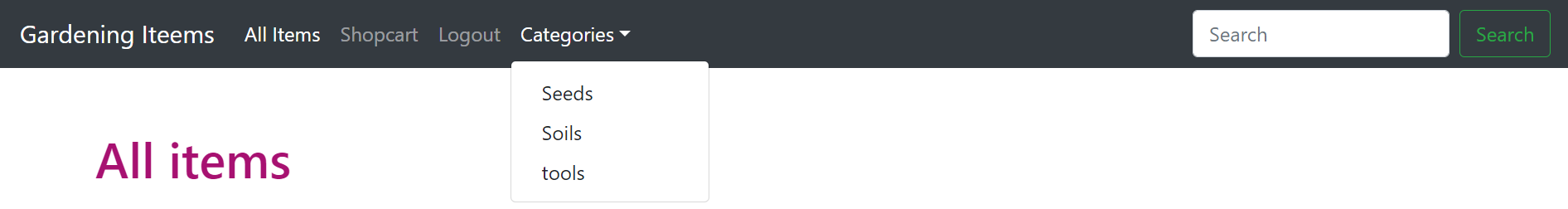


Рисунок 4.5 – Кнопки выбора категории товара

При выборе конкретной категории будут показаны товары, относящиеся к ней. Пример каталога для категории *Seeds* приведён на рисунке 4.6.

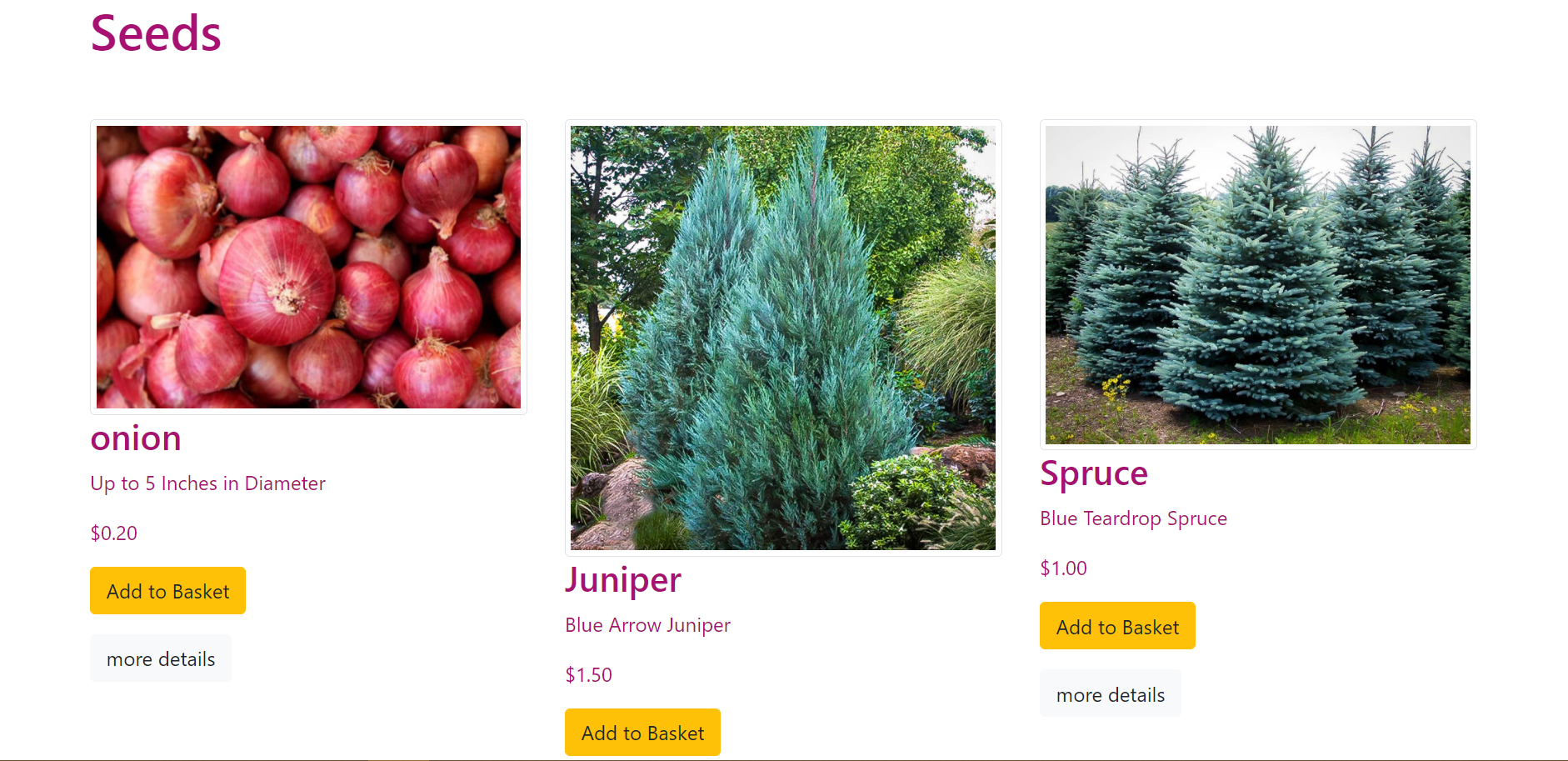


Рисунок 4.6 – Каталог товаров категории Seeds

В случае, если пользователь хочет увидеть подробную информацию о товаре, следует нажать кнопку *more details*, и в этом случае появится все необходимые данные. Пример подробной информации о товаре *Juniper* показана на рисунке 4.7.



Рисунок 4.7 – Подробная информация товара Juniper

Если пользователь хочет после просмотра подробной информации о товаре, он может сразу добавить его к себе в корзину, нажав на кнопку *Add to Basket*. В этом случае он будет перенаправлен в корзину своих товаров, как показано на рисунке 4.8.

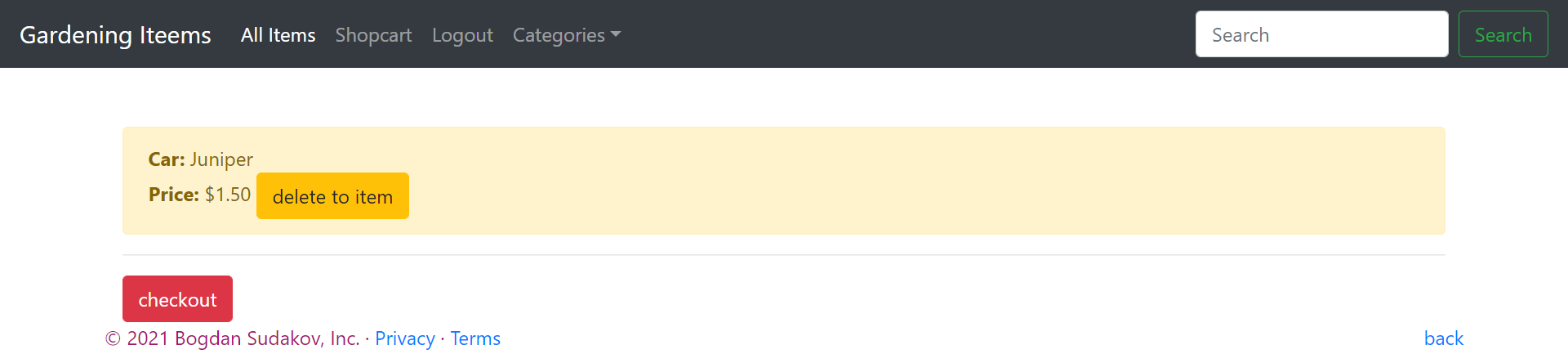


Рисунок 4.8 – Корзина пользователя

Затем пользователь может вернуться к каталогу и производить дальнейший выбор товара и последующее добавление его в корзину. Причем добавление в корзину возможно и без просмотра подробностей. Для этого кнопка *Add to Basket* находится возле каждого товара, приведенного в каталоге (см. рисунок 4.9).

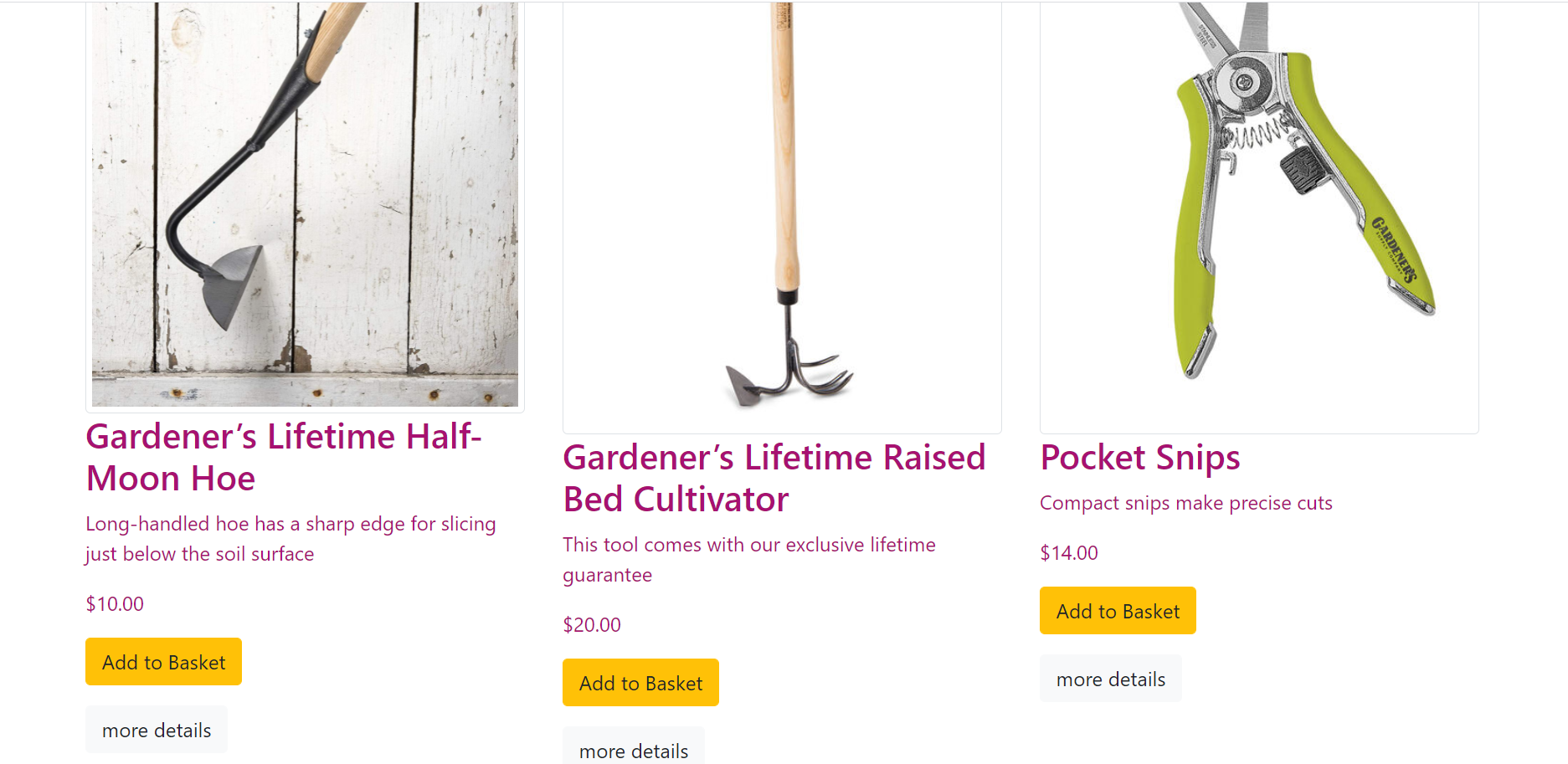


Рисунок 4.9 – Наличие добавления в корзину возле каждого товара

После того, как пользователь выбрал всё необходимое, при нажатии кнопки *Shopcart* он возвращается в корзину для проверки наличия всех выбранных им товаров и при нажатии кнопки checkout происходит переход к оформлению заказа и указанию всех необходимых данных для дальнейшей доставки и связи с покупателем (см. рисунок 4.10)



Рисунок 4.9 – Форма создания заказа

Чтобы окончательно оформить заказ следует нажать на кнопку *complete order*, после чего будет выдано сообщение об успешном выполнении процедуры оформления заказа.

По окончании сессии пользователь может выйти из своего аккаунта, нажав на кнопку Logout, после чего он вернется к странице входа в аккаунт.

Таким образом, функционал сайта довольно понятен и интуитивен, что позволяет пользователю легко ориентироваться по сайту.

# Заключение

В результате проделанной работы был создано интернет приложение, которое способно значительно упростить поиск нужных товаров. Данный проект представляет собой пользовательский интерфейс, предназначенный для просмотра и изменения данных из базы. Логика сайта создана в соответствии с требованиями, представленными выше.

Соблюдены правила создания базы данных, разработанная база содержит оптимальное количество таблиц для простого интернет-приложения, данные в таблицах приведены к третьей нормальной форме.

Так же соблюдались предписания по созданию пользовательского интерфейса, интерфейс получился простым, интуитивно понятным.

Использовались современные технологии по созданию веб-ресурсов (*С#*, *HTML*), что обеспечило широкое отображение ресурсов, практически без ограничений по браузерной части.

Пояснительная записка оформлена в соответствии с требованиями стандарта СТП 01-2017

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Святослав Куликов. Реляционные базы данных в примерах – EPAM Systems, RD Dep, 2021. – 420 с.
2. К. Дж. Дейт. Введение в системы баз данных/ К. Дж. Дейт - Вильямс, 2017. – 1328 с.
3. Вендров А. М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000.
4. Обзор программных средств для реализации БД [Электронный ресурс] – Режим доступа: <https://car-tesla.ru/kompjutery/obzor-programmnyh-sredstv-dlia-realizacii-bazy-dannyh/>, свободный.
5. СТП 01-2017. Стандарт предприятия. Дипломные проекты(работы). Общие требования. – Минск: БГУИР, 2017. – 169
6. Томас Конолли. Базы данных. Проектирование, реализация и сопровождение. Теория и практик/ Томас Конолли, Карролин Бегг – Вильямс, 2003. – 1430 с
7. Фримен Адам *asp.NET Core MVC 2* с примерами на *C#* для профессионалов.

# ПРИЛОЖЕНИЕ А

**(справочное)**

**Выдержка из кода**

Представление главной страницы сайта

*<form asp-action="Login" asp-controller="Account" asp-anti-forgery="true">*

*<div class="validation" asp-validation-summary="ModelOnly" />*

*<div>*

*<div class="form-group">*

*<label asp-for="Login">Введите Login</label>*

*<input type="text" asp-for="Login" />*

*<span asp-validation-for="Login" />*

*</div>*

*<div class="form-group">*

*<label asp-for="Password">Введите пароль</label>*

*<input asp-for="Password" />*

*<span asp-validation-for="Password" />*

*</div>*

*<div class="form-group">*

*<input type="submit" value="Войти" class="btn btn-outline-dark" />*

*</div>*

*</div>*

*</form>*

*<form asp-action="Register" asp-controller="Account" asp-anti-forgery="true">*

*<div class="validation" asp-validation-summary="ModelOnly" />*

*<div>*

*<div>*

*<label asp-for="Name">Введите Name</label><br />*

*<input type="text" asp-for="Name" />*

*<span asp-validation-for="Name" />*

*</div>*

*<div>*

*<label asp-for="Login">Введите Login</label><br />*

*<input type="text" asp-for="Login" />*

*<span asp-validation-for="Login" />*

*</div>*

*<div>*

*<label asp-for="Email">Введите Email</label><br />*

*<input asp-for="Email" />*

*<span asp-validation-for="Email" />*

*</div>*

*<div>*

*<label asp-for="Password">Введите пароль</label><br />*

*<input asp-for="Password" />*

*<span asp-validation-for="Password" />*

*</div>*

*<div>*

*<label asp-for="ConfirmPassword">Повторите пароль</label><br />*

*<input asp-for="ConfirmPassword" />*

*<span asp-validation-for="ConfirmPassword" />*

*</div>*

*<div>*

*<input type="submit" value="Регистрация" />*

*</div>*

*</div>*

*</form>*

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta name="viewport" content="width=device-width" />*

*<title>@ViewBag.Title</title>*

*<link href="~/css/bootstrap.min.css" rel="stylesheet" type="text/css">*

*<link href="~/css/Style.css" rel="stylesheet" type="text/css">*

*</head>*

*<body>*

*<nav class="navbar navbar-expand-md navbar-dark bg-dark">*

*<a class="navbar-brand" asp-controller="Home" asp-action="Index">Gardening Iteems</a>*

*<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">*

*<span class="navbar-toggler-icon"></span>*

*</button>*

*<div class="collapse navbar-collapse" id="navbarsExampleDefault">*

*<ul class="navbar-nav mr-auto">*

*<li class="nav-item active">*

*<a class="nav-link" href="/Items/List">All Items <span class="sr-only">(current)</span></a>*

*</li>*

*<li class="nav-item">*

*<a class="nav-link" href="/ShopCart/">Shopcart</a>*

*</li>*

*<li class="nav-item">*

*<a class="nav-link" asp-controller="Account" asp-action="Logout">Logout</a>*

*</li>*

*<li class="nav-item dropdown">*

*<a class="nav-link dropdown-toggle" href="" id="dropdown01" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Categories</a>*

*<div class="dropdown-menu" aria-labelledby="dropdown01">*

*<a class="dropdown-item" asp-controller="Items" asp-action="List" asp-route-category="Seeds">Seeds</a>*

*<a class="dropdown-item" asp-controller="Items" asp-action="List" asp-route-category="Soils">Soils</a>*

*<a class="dropdown-item" asp-controller="Items" asp-action="List" asp-route-category="tools">tools</a>*

*</div>*

*</li>*

*</ul>*

*<form class="form-inline my-2 my-lg-0">*

*<input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">*

*<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>*

*</form>*

*</div>*

*</nav>*

*<div class="container mt-5">*

*@RenderBody()*

*</div>*

*<footer class="container">*

*<p class="float-right"><a href="#">back</a></p>*

*<p>© 2021 Bogdan Sudakov, Inc. · <a href="#">Privacy</a> · <a href="#">Terms</a></p>*

*</footer>*

*<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>*

*<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>*

*<script>window.jQuery || document.write('<script src="/docs/4.5/assets/js/vendor/jquery.slim.min.js"><\/script>')</script>*

*<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.bundle.min.js"></script>*

*</body>*

*</html>*

*@model GardeningItem*

*<div class="col-lg-4">*

*<img class="img-thumbnail " src="@Model.Img" alt="@Model.Name" />*

*<h3>*

*@Model.Name*

*</h3>*

*<p>*

*@Model.ShortDesc*

*</p>*

*<p>*

*@Model.Price.ToString("C", new CultureInfo("en-US"))*

*</p>*

*<p>*

*<a class="btn btn-warning" asp-controller="ShopCart" asp-action="addToCart"*

*asp-route-Id="@Model.Id"> Add to Basket </a>*

*</p>*

*<p>*

*<a class="btn btn-light" asp-controller="Items" asp-action="MoreDetails"*

*asp-route-Id="@Model.Id"> more details </a>*

*</p>*

*</div>*

Представление оформления заказа

*@model Order*

*<h3>checkout</h3>*

*<div class="editor-label">*

*@ViewBag.Message*

*</div>*

*<div class="card-body">*

*<form asp-action="Checkout" method="post" class="form-horizontal">*

*<div class="form-group">*

*<label asp-for="Name"></label>*

*<div class="col-12">*

*<input asp-for="Name" class="form-control" />*

*<span asp-validation-for="Name" class="text-danger" />*

*</div>*

*</div>*

*<div class="form-group">*

*<label asp-for="Surname"></label>*

*<div class="col-12">*

*<input asp-for="Surname" class="form-control" />*

*<span asp-validation-for="Surname" class="text-danger" />*

*</div>*

*</div>*

*<div class="form-group">*

*<label asp-for="Adress"></label>*

*<div class="col-12">*

*<input asp-for="Adress" class="form-control" />*

*<span asp-validation-for="Adress" class="text-danger" />*

*</div>*

*</div>*

*<div class="form-group">*

*<label asp-for="Phone"></label>*

*<div class="col-12">*

*<input asp-for="Phone" class="form-control" />*

*<span asp-validation-for="Phone" class="text-danger" />*

*</div>*

*</div>*

*<div class="form-group">*

*<label asp-for="Email"></label>*

*<div class="col-12">*

*<input asp-for="Email" class="form-control" />*

*<span asp-validation-for="Email" class="text-danger" />*

*</div>*

*</div>*

*<input class="btn btn-success" type="submit" value="complete order" />*

*</form>*

*<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>*

*<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.17.0/jquery.validate.min.js"></script>*

*<script src="https://ajax.aspnetcdn.com/ajax/jquery.validation.unobtrusive/3.2.10/jquery.validate.unobtrusive.min.js"></script>*

*</div>*

Представление получения подробной информации товара

*@model GardeningItem*

*<h3>@Model.Category.CategoryName</h3>*

*<div class="col-lg-4">*

*<img class="img-thumbnail " src="@Model.Img" alt="@Model.Name" />*

*<h3>*

*@Model.Name*

*</h3>*

*<p>*

*@Model.ShortDesc*

*</p>*

*<p>*

*@Model.Price.ToString("C", new CultureInfo("en-US"))*

*</p>*

*<p>*

*@Model.LongDesc*

*</p>*

*<p>*

*<a class="btn btn-warning" asp-controller="ShopCart" asp-action="addToCart"*

*asp-route-id="@Model.Id"> Add to Basket </a>*

*</p>*

*</div>*

# ПРИЛОЖЕНИЕ Б

**(справочное)**

**Таблицы разработанной базы данных**

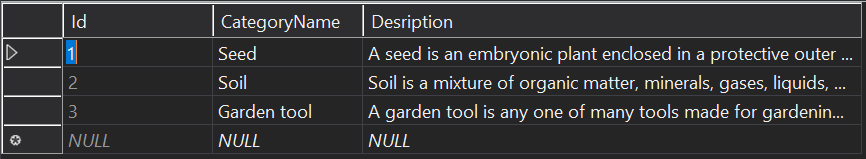


Рисунок Б1 – Таблица *Сategory*

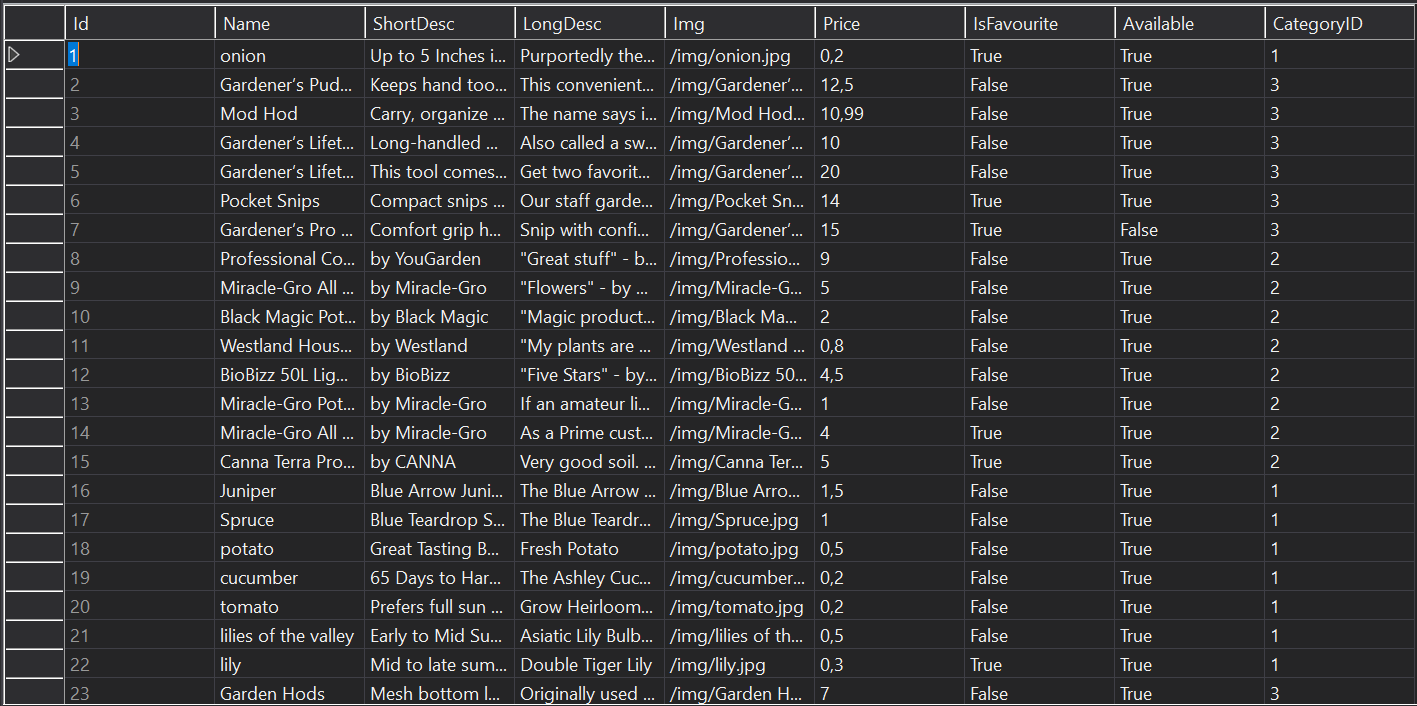


Рисунок Б2 – Таблица *GardeningItems*

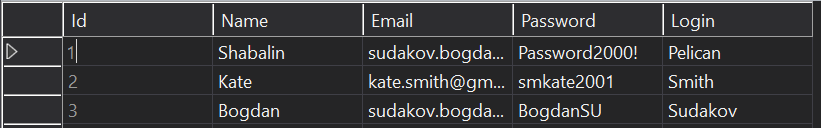


Рисунок Б3 – Таблица *Users*

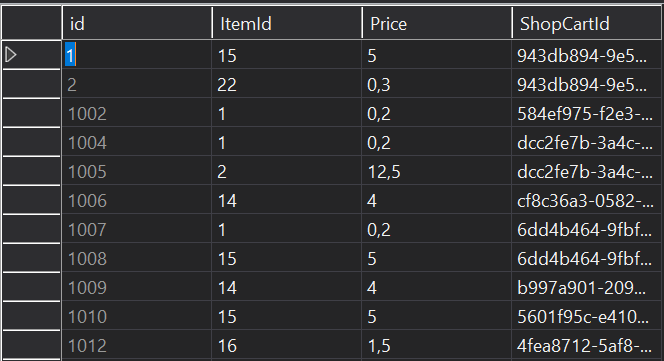


Рисунок Б4 – Таблица *ShopCartItem*

# Ведомость курсового проекта

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | |
|  | | | | | *Текстовые документы* |  | | | | |
| *БГУИР КП 1-53 01 02 06 023 ПЗ* | | | | |  |  | | | | |
|  | | | | | *Пояснительная записка* | *54с.* | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | *БГУИР КП 1-53 01 02 06 023 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л* | *№ докум.* | *Подп.* | *Дата* | *Интернет-магазин садового инструмента*  *Ведомость курсового проекта* | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Судаков* |  |  |  | *Т* |  | *54* | *54* |
| *Провер.* | | *Трофимович* |  |  | *Кафедра ИТАС*  *гр. 820601* | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |