ВВЕДЕНИЕ

По мере развития информационных технологий в нашем мире происходит постоянное возрастание сложности информационных систем (ИС), которые создают и используют в различных областях экономики. У особенностям современных проектов ИС относятся такие как: требующие внимания моделирование и анализ данных: существование подсистем (объект системы, состязаний из элементов и представляющий собой отдельную систему) с определенными задачами и функциями; функционирование в неоднородной среде на нескольких аппаратных платформах. Создание ИС – это логически сложная, занимающая много времени и требующая сил работа, для которой необходимо обладать высокой квалификацией. Хотя довольно часто создание информационных систем нуждается не столько в навыках работы с информационными технологиями, сколько в выполнении задач при помощи интуиции и собственных основанных на искусстве и опыте методах.

Актуальность работы заключается в том, что информация и научные знания в последние годы играют все большую роль в жизни общества. Информация на данный момент стала одним из важнейших ресурсов научно-технического прогресса, модернизации общества, его развития во всех отраслях и достижения новых высот, так как появились средства вычислительной техники, которые помогают в этом. Самым приоритетным видом деятельности сегодня является формирование информационного общества, в котором процессы сбора, обработки, анализа, передачи информации, т.е. информационные и коммуникационные технологии занимают основное место в различных сферах человеческой деятельности.

Целью дипломной работы является разработка интернет-магазина живых цветов и семян на языке программирования Python с помощью фреймовика Django.

Объект исследования – магазин живых цветов и семян «Нотка».

Предмет исследования – средства и методы создания интернет-магазинов.

Исходя из цели, объекта и предмета были поставлены следующие задачи:

1. Провести аналитический обзор существующих методов решения поставленной задачи;
2. Разработать архитектуру разрабатываемого программного обеспечения;
3. Рассмотреть функциональные требования для проектирования программных средств и разработки внутренней архитектуры;
4. Реализовать проект;
5. Провести верификацию, валидацию и эксперимент созданного проекта.

Раздел 1 Аналитический обзор существующих методов решения поставленной задачи на основе изучения литературных и других источников по теме дипломной работы

1.1 Язык программирования Python

Python представляет собой высокоуровневый язык программирования, отличающийся эффективностью, простотой и универсальностью использования. Он широко применяется в разработке веб-приложений и прикладного программного обеспечения, а также в машинном обучении и обработке больших данных. За счет простого и интуитивно понятного синтаксиса является одним из распространенных языков для обучения программированию.

Питон – язык не новый. Его разработка началась ещё в конце восьмидесятых годов. Релиз первой версии языка произошёл в феврале 1991 года.

Отцом-основателем Питона и, на протяжении многих лет, главным разработчиком являлся голландский программист Гвидо Ван Россум. На момент создания языка Гвидо работал в центре математики и информатики в Нидерландах. В качестве основы для Пайтона Россум взял язык программирования ABC, в разработке которого когда-то участвовал. Почему выбрали Python
Нет. Он не назван в честь опасной змеи. Россум был фанатом комедийного сериала в конце 70-х. Название “Python” было взято из этого же сериала “Monty Python’s Flying Circus” (Летающий цирк Монти Пайтона).

У Python короткий и выразительный синтаксис, особенно в сравнении с такими императивными языками, как C++, Java, C#. Изюминкой синтаксиса является то, что вложенность обозначается отступами слева, а не фигурными скобками или другими знаками. Можно сказать, что благодаря этому язык

диктует хороший стиль оформления кода. Так же существует единый стандарт оформления (PEP-8) и во многих средах программирования можно привести код к этому стандарту при помощи нажатия одной комбинации клавиш.

Простота отчасти обусловлена тем, что Питон написан на основе языка ABC, который использовался для обучения программированию.

Питон можно абсолютно свободно использовать в любом проекте, даже в коммерческом. То, что у этого языка открытый исходный код, а на его будущее сильно влияет мнение широкой общественности – дополнительные драйверы развития.

Программа, написанная на Пайтоне, может быть запущенна почти на любой операционной системе. Перенести скрипт с одной платформы на другую – дело нескольких кликов.

Python позволяет с лёгкостью использовать код, написанный на других языках (особенно, на C). Это даёт возможность ускорить Вашу программу в критически важных местах.

Язык берёт на себя многие нюансы низкого уровня. Главные из них это «сборка мусора», работа с памятью, работа с конкурентностью. Это освобождает от головной боли, но и навязывает некоторые архитектурные решения.

Как уже говорилось, в Питоне есть обширная стандартная библиотека и множество сторонних библиотек. Для их установки и подключения предусмотрены удобные синтаксические конструкции и менеджер пакетов PIP.

1.2 Веб-фреймворк Django

Django — это высокоуровневый Python веб-фреймворк, который позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

Созданный опытными разработчиками, Django берёт на себя большую часть хлопот веб-разработки, поэтому вы можете сосредоточиться на написании своего веб-приложения без необходимости изобретать велосипед. Он бесплатный и с открытым исходным кодом, имеет растущее и активное сообщество, отличную документацию и множество вариантов как бесплатной, так и платной поддержки.

Django помогает писать программное обеспечение, которое будет:

- Полным. Django следует философии «Всё включено» и предоставляет почти всё, что разработчики могут захотеть сделать «из коробки». Поскольку всё, что вам нужно, является частью единого «продукта», всё это безупречно работает вместе, соответствует последовательным принципам проектирования и имеет обширную и актуальную документацию.
- Разносторонним. Django может быть (и был) использован для создания практически любого типа веб-сайтов — от систем управления контентом и wiki до социальных сетей и новостных сайтов. Он может работать с любой клиентской средой и может доставлять контент практически в любом формате (включая HTML, RSS-каналы, JSON, XML и т. д.).

  Хотя Django предоставляет решения практически для любой функциональности, которая вам может понадобиться (например, для нескольких популярных баз данных, шаблонизаторов и т. д.), внутренне он также может быть расширен сторонними компонентами, если это необходимо.
- Безопасным. Django помогает разработчикам избежать многих распространённых ошибок безопасности, предоставляя фреймворк, разработанный чтобы «делать правильные вещи» для автоматической защиты сайта. Например, Django предоставляет безопасный способ управления учётными записями пользователей и паролями, избегая распространённых ошибок, таких как размещение информации о

сеансе в файлы cookie, где она уязвима (вместо этого файлы cookie содержат только ключ, а фактические данные хранятся в базе данных) или непосредственное хранение паролей вместо хэша пароля.

Хэш пароля — это значение фиксированной длины, созданное путём обработки пароля через криптографическую хэш-функцию Django может проверить правильность введённого пароля, пропустив его через хэш-функцию и сравнив вывод с сохранённым значением хэша. Благодаря «одностороннему» характеру функции, даже если сохранённое хэш-значение скомпрометировано, злоумышленнику будет сложно определить исходный пароль.

Django, по умолчанию, обеспечивает защиту от многих уязвимостей, включая SQL-инъекцию, межсайтовый скриптинг, подделку межсайтовых запросов и кликджекинг.

- Масштабируемым. Django использует компонентную “shared-nothing” архитектуру (каждая её часть независима от других и, следовательно, может быть заменена или изменена, если это необходимо). Чёткое разделение частей означает, что Django может масштабироваться при увеличении трафика, путём добавления оборудования на любом уровне: серверы кеширования, серверы баз данных или серверы приложений. Одни из самых загруженных сайтов успешно масштабировали Django (например, Instagram и Disqus, если назвать только два из них).
- Удобным в сопровождении. Код Django написан с использованием принципов и шаблонов проектирования, которые поощряют создание поддерживаемого и повторно используемого кода. В частности, в нём используется принцип «Don't Repeat Yourself» (DRY, «не повторяйся»), поэтому нет ненужного дублирования, что сокращает объём кода. Django также способствует группированию связанных функциональных возможностей в повторно используемые «приложения» и, на более низком уровне, группирует связанный код в

модули (в соответствии с шаблоном Model View Controller (MVC) (en-US)).

- Переносным. Django написан на Python, который работает на многих платформах. Это означает, что вы не привязаны к какой-либо конкретной серверной платформе и можете запускать приложения на многих версиях Linux, Windows и Mac OS X. Кроме того, Django хорошо поддерживается многими веб-хостингами, которые часто предоставляют определённую инфраструктуру и документацию для размещения сайтов Django.

Django был разработан в период с 2003 по 2005 год командой, которая занималась созданием и обслуживанием газетных веб-сайтов. После создания нескольких сайтов, команда начала повторно использовать множество общего кода и шаблонов проектирования. Этот общий код эволюционировал в веб-фреймворк, который превратился в проект "Django" с открытым исходным кодом в июле 2005 года.

Django продолжает расти и улучшаться с момента его первого релиза (1.0) в сентябре 2008 года до недавно выпущенной версии 3.1 (2020). В каждой версии добавлены новые функциональные возможности и исправлены ошибки, начиная от поддержки новых типов баз данных, шаблонизаторов и кеширования, до добавления «общих» функций просмотра и классов (уменьшающих объём кода, который разработчики должны писать для ряда программных задач).

Веб-приложения, написанные на Django, обычно группируют код, который обрабатывает каждый из этих шагов, в отдельные файлы (Рисунок 1).

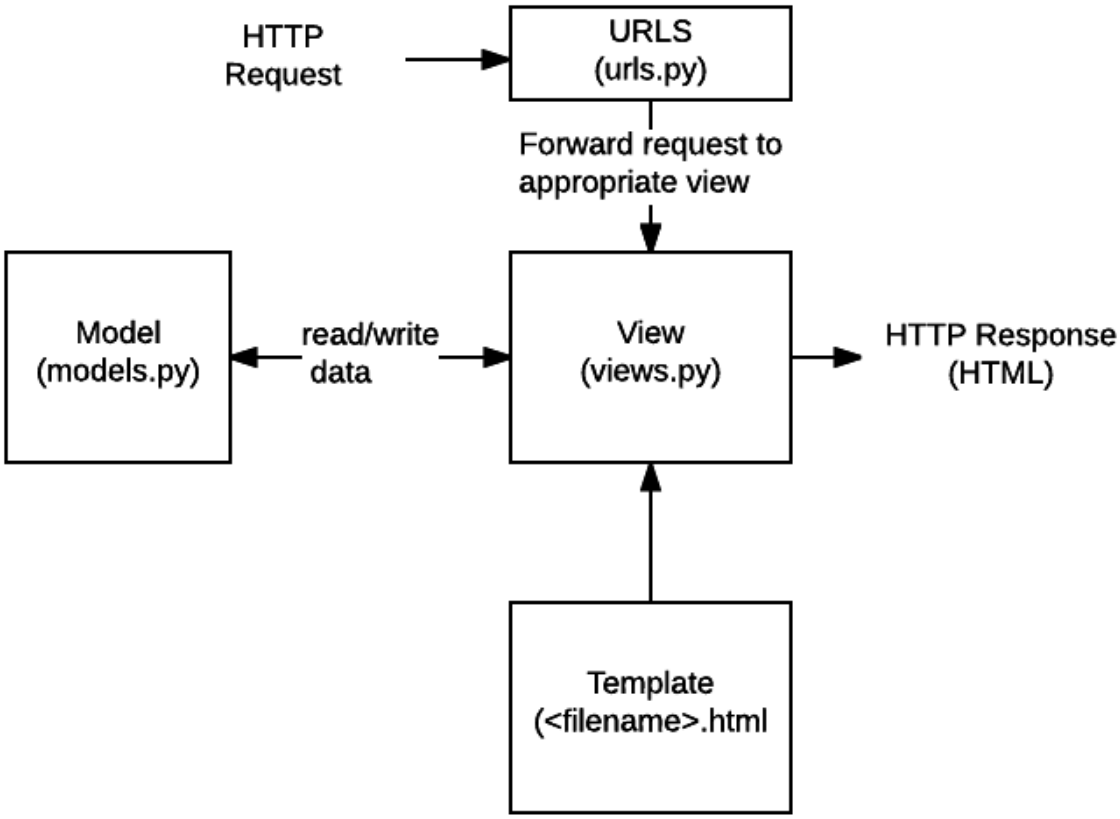

Рисунок 1 – Группировка кода

URLs: Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения (view) в виде аргументов.

View: View (англ. «отображение») — это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.

Models: Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для

управления (добавления, изменения, удаления) и выполнения запросов в базу данных.

Templates: Template (англ. «шаблон») — это текстовый файл, определяющий структуру или разметку страницы (например HTML-страницы), с полями для подстановки, которые используются для вывода актуального содержимого. View может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (model). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

Отображения (views) — это сердце веб-приложения, принимающего HTTP-запросы от веб-клиентов и возвращающего HTTP-ответы. Между этим они используют другие ресурсы фреймворка для доступа к базам данных, шаблонам визуализации и т. д.

В приведённом ниже примере показана минимальная функция представления index(), которая могла быть вызвана нашим сопоставителем URL-адресов в предыдущем разделе. Как и все функции отображения (view), она получает объект HttpRequest в качестве параметра (request) и возвращает объект HttpResponse. В этом случае мы ничего не делаем с запросом, и наш ответ просто возвращает жёстко запрограммированную строку. Мы покажем вам запрос, который делает что-то более интересное в следующем разделе.

```
## filename: views.py (Django view functions)
from django.http import HttpResponse
def index(request):
    # Получить HttpRequest — параметр запроса
    # Выполнить операции, используя информацию из запроса.
    # Вернуть HttpResponse
    return HttpResponse('Hello from Django!')
```

2 Архитектура разрабатываемого программного обеспечения

2.1 Модель предметной области

Реализация модели предметной области означает пополнение приложения целым слоем объектов, описывающих различные стороны определенной предметной области. Одни объекты призваны имитировать элементы данных, которыми оперируют в этой области, а другие должны формализовать те или иные бизнес-правила. Функции тесно сочетаются сданными, которыми они манипулируют.

Объектно-ориентированная модель предметной области часто напоминает схему соответствующей базы данных, хотя между ними все еще остается множество различий. В модели предметной области смешиваются данные и функции, допускаются многозначные атрибуты, создаются сложные сети ассоциаций и используются связи наследования.

В сфере корпоративных программных приложений можно выделить две разновидности моделей предметной области. "Простая" во многом походит на схему базы данных и содержит, как правило, по одному объекту домена в расчете на каждую таблицу. "Сложная" модель может отличаться от структуры базы данных и содержать иерархии наследования, стратегии и иные типовые решения, а также сложные сети мелких взаимосвязанных объектов. Сложная модель более адекватно представляет запутанную бизнес-логику, но труднее поддается отображению в реляционную схему базы данных. В простых моделях подчас достаточно применять варианты тривиального типового решения активная запись , в то время как в сложных без замысловатых преобразователей данных не обойтись.

Логика предметной обычно подвержена частым изменениям, поэтому весьма важна возможность простой модификации и тестирования этого слоя кода. Отсюда следует настоятельная необходимость снижать степень зависимости модели предметной области от других слоев системы. Более

того, именно это требование является основополагающим аспектом многих типовых решений, имеющих отношение к "расслоению" системы.

С моделью предметной области связано большое количество различных контекстов. Простейший вариант — однопользовательское приложение, где единый граф объектов считывается из дискового файла и располагается в оперативной памяти. Такой стиль работы присущ настольным программам, но менее характерен для многоуровневых приложений, поскольку в них намного больше объектов. Размещение каждого объекта в памяти сопряжено с чрезмерными затратами ресурсов памяти и времени. Достоинство объектно-ориентированных систем баз данных заключается в том, что они создают впечатление, будто объекты пребывают в памяти постоянно.

Обычно в ходе выполнения сеанса в память загружается полный граф объектов, хотя речь вовсе не идет обо всех объектах и, может быть, классах. Если, например, ведется поиск множества контрактов, достаточно считать информацию только о таких продуктах, которые упоминаются в этих контрактах. Если же в вычислениях участвуют объекты контрактов и зачтенных доходов, объекты продуктов, возможно, создавать вовсе не нужно.

Точный перечень данных, загружаемых в память, определяется параметрами объектно-реляционного отображения.

Если в продолжение процесса обработки нескольких вызовов необходим один и тот же граф объектов, состояние сервера следует сохранять.

Одна из типичных проблем логики предметной области связана с чрезмерным увеличением объектов. Чтобы избежать подобного, можно выделить общие характеристики предметов и сосредоточить их в одноименном классе, а все остальные функции вынести во вспомогательные классы сценариев транзакции или даже слоя представления.

При этом, однако, возникает опасность повторения фрагментов кода. Функции, не относящиеся к категории общих, отыскать довольно трудно, и

многие предпочитают этим просто не заниматься, соглашаясь с дублированием кода. Повторение часто приводит к усложнению и несогласованности, хотя, эффекты излишнего увеличения размеров классов наблюдаются значительно реже, чем можно было ожидать Если такое действительно происходит, результаты вполне очевидны и легко поправимы. Поэтому советую размещать весь родственный код в пределах одного класса и заниматься его разделением по нескольким классам только тогда, когда это в самом деле целесообразно

Для реализации решения моделей предметной области было создано пространство. В пространстве содержатся следующие классы:

- Класс Type
- Класс Member
- Класс Ypakovka
- Класс Rastenie
- Класс Prodaza
- Класс CartProduct
- Класс Cart
- Класс Order
- Класс Customer
- Класс NotificationManager
- Класс Notification
- I Класс mageGallary

2.2 Функциональные требования

В рамках процесса проектирования рассмотрим основные требования, предъявляемые к автоматизированной системе магазина. Функциональные требования исследуются и формулируются в виде модели вариантов использования.

Вариант использования – это независящее от реализации высокоуровневое представление конкретной функции разрабатываемой системы. Вариант использования представляет собой последовательность действий, выполняемых системой в ответ на событие, инициируемое внешним объектом (действующим лицом, актером) [12]. Другими словами, такое представление предназначено для упрощения взаимодействия с будущими пользователями системы, с клиентами, и особенно пригодятся для определения необходимых характеристик системы.

Составление вариантов использования производится на основе анализа требований к интернет-магазинам. Главными актерами в контексте программного средства являются администратор и пользователь этой системы.

Администратор в праве добавлять товары, устанавливать им характеристики, цену и категорию, а также удалить или изменить уже существующий товар. По мере осуществления продаж администратор может отслеживать статистику по продажам, посещению, активности и т.д.

Пользователи должны иметь возможность по просмотру, поиску, сортировки и категоризации товаров, перед непосредственной покупкой. Однако перед тем, чтобы пользователь в полной мере мог совершать какие-либо действия на сайте, ему необходимо авторизоваться или зарегистрироваться.

На рисунке 2 изображена диаграмма вариантов использования в нотации UML.

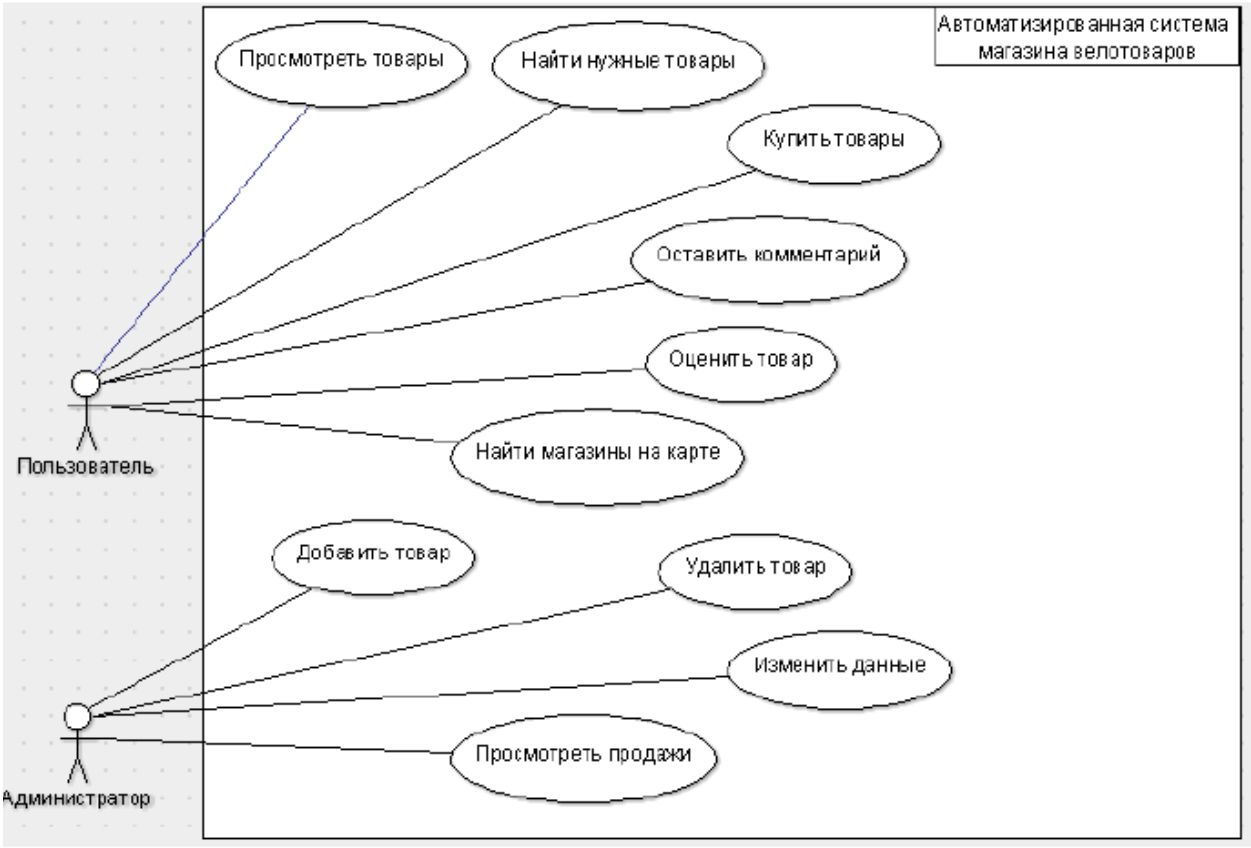

Рисунок 2 - Диаграмма вариантов использования в нотации UML

3 Структура программного обеспечения и еѐ реализации на языках программирования

Базовым шаблоном является base.html. Он содержит в себе шапку сайта, с логотипом, поисковым блогом, и навигацией. Доступ к этим элементам необходим из любой части сайта, поэтому базовый шаблон всегда будет использоваться при рендеринге страницы. Его наследуют остальные шаблоны

В проекте представлены следующие шаблоны:

– product_detail.html, страница с подробной информацией о конкретном товаре, характеристиками, рейтингом, отзывами и т.д.;

– cart.html, пользовательская корзина с возможностью изменять количество товаров, удалять или оформить покупку;

– wish.html, список желаемых товаров пользователя;

– login.html, страница авторизации пользователя;

– registration.html, страница регистрации нового пользователя;

Приняв во внимание требования, представленные в разделе проектирования, был разработан пользовательский интерфейс.

Рассмотрим основные страницы веб-приложения, такие как: главная страница интернет-магазина, где представлены акции и предложения, каталог товаров с возможностью выборки и сортировки по разным параметрам, страница с подробным описанием товара, его характеристиками и отзывами, карта магазинов в которых данный товар есть в наличии, а также пользовательская корзина, в которую попадают все товары, перед оформлением покупки.

Верхняя часть сайта является отдельным шаблонам, и несет в себе функции, которые необходимы из любой части интернет-магазина. Данный шаблон наследуют все остальные.

Интерфейс главной страницы представлен на рисунке 3.

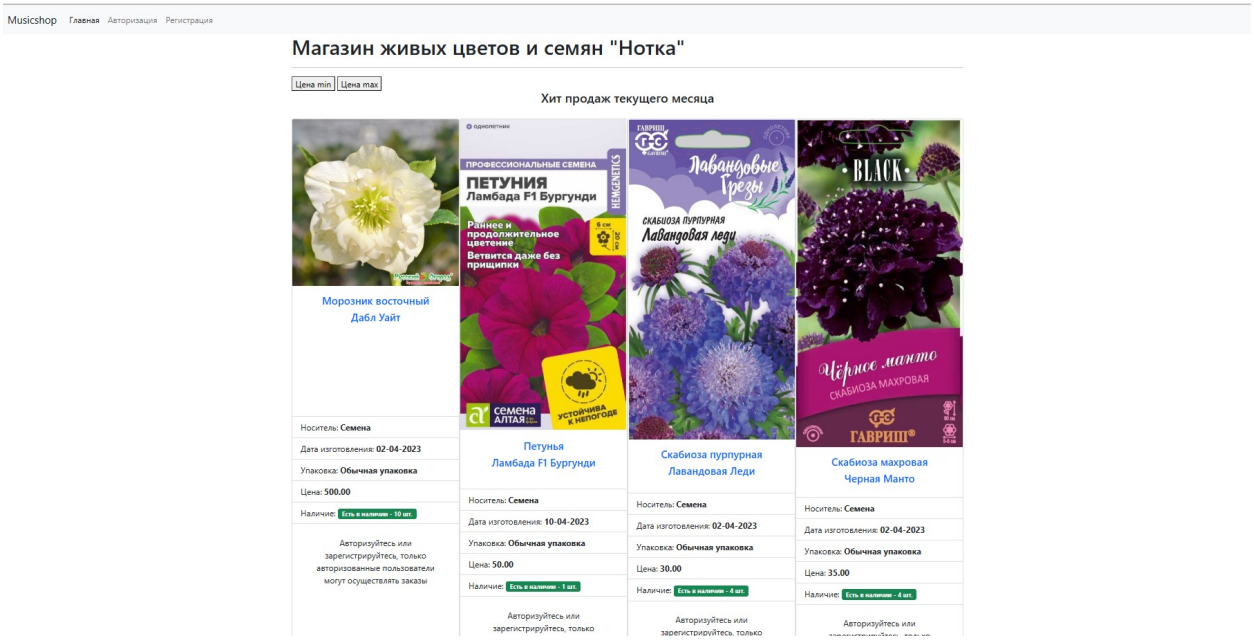

Рисунок 3 – Главная страница

На главной странице представлены продаваемые товаре. В верхней части кнопки «Авторизация» и «Регистрация». Есть возможность отсортировать цену по параметрам (min, max).

Для того чтобы пользователю совершать покупки необходимо пройти регистрацию «Рисунок 4».

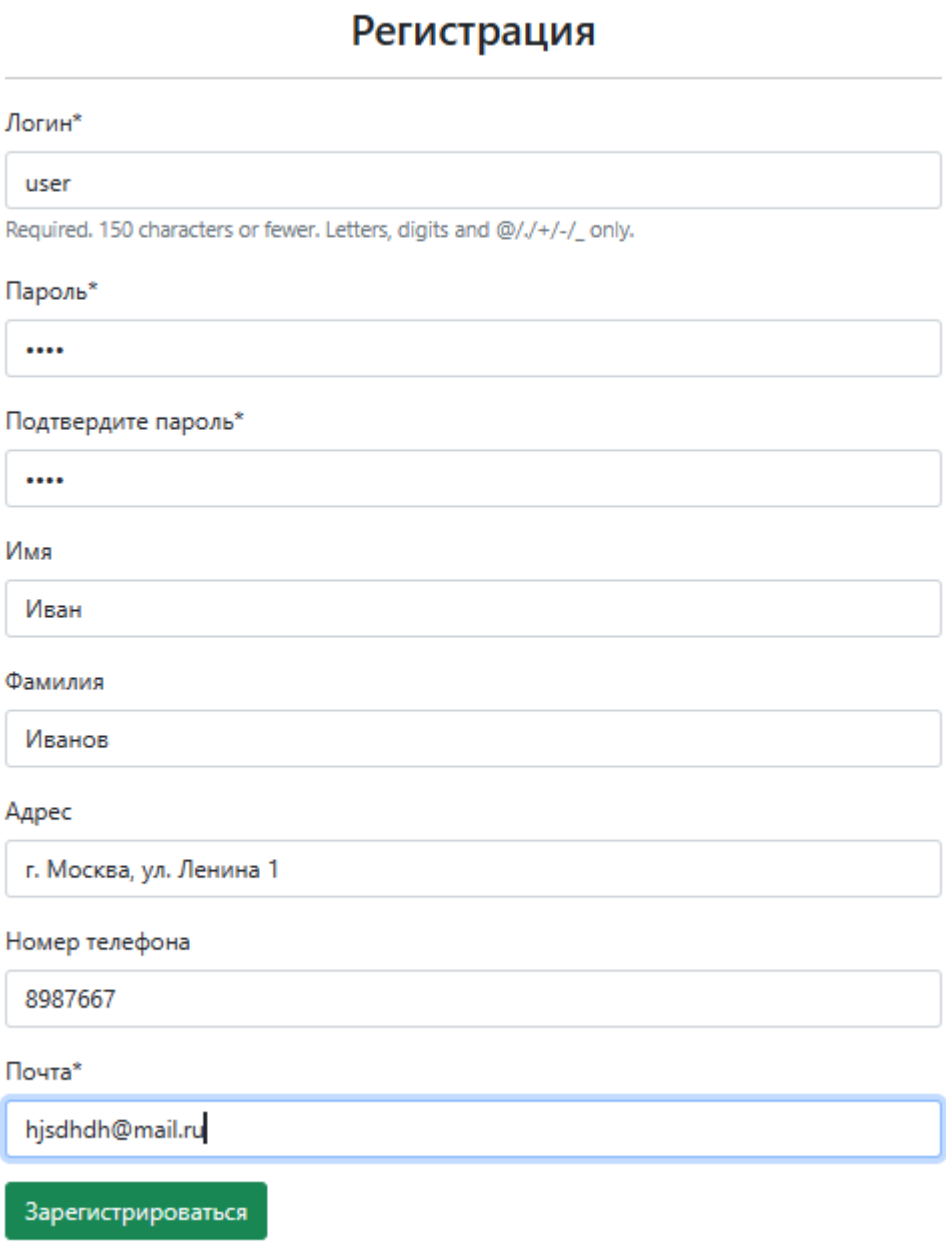

Рисунок 4 – Форма регистрации

Форма авторизации представлена на рисунке 5.

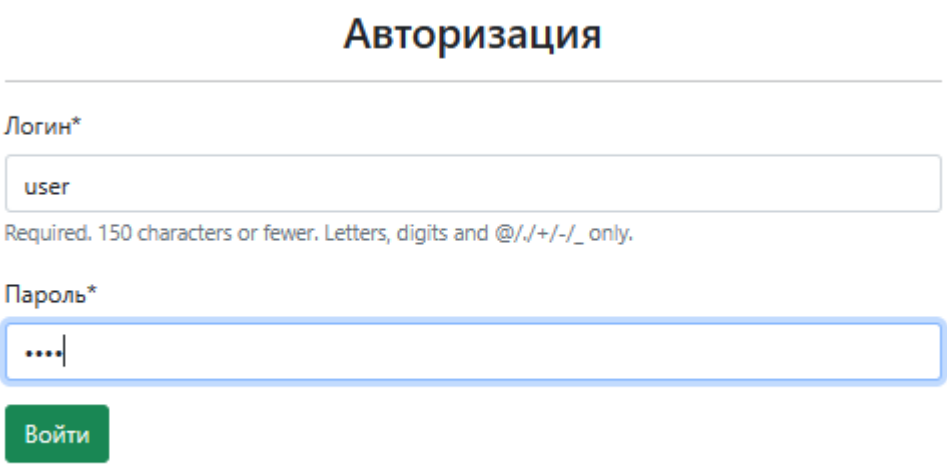

Рисунок 5 – Форма авторизации

Пользователь отдельно может просматривать галерею изображений выбранного товара (Рисунок 6).

Рисунок 6 – Галерея изображений

Также пользователь может просматривать информацию о дате изготовления, упаковки, описания и есть ли товар в наличии (Рисунок 7).

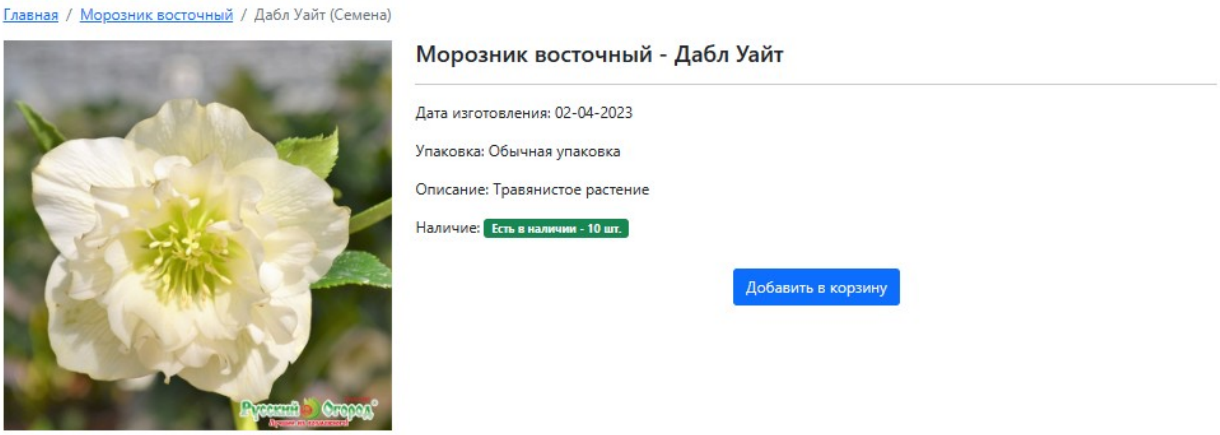

Рисунок 7 – Информации

После добавления в корзину в правом верхнем углу появляется информация о количестве добавленных товаров в корзину (Рисунок 8).

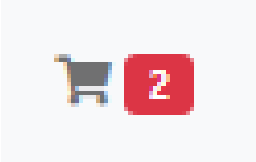

Рисунок 8 – Значок корзины

При переходе в саму корзину видим добавленные товары и общую сумму к оплате (Рисунок 9).

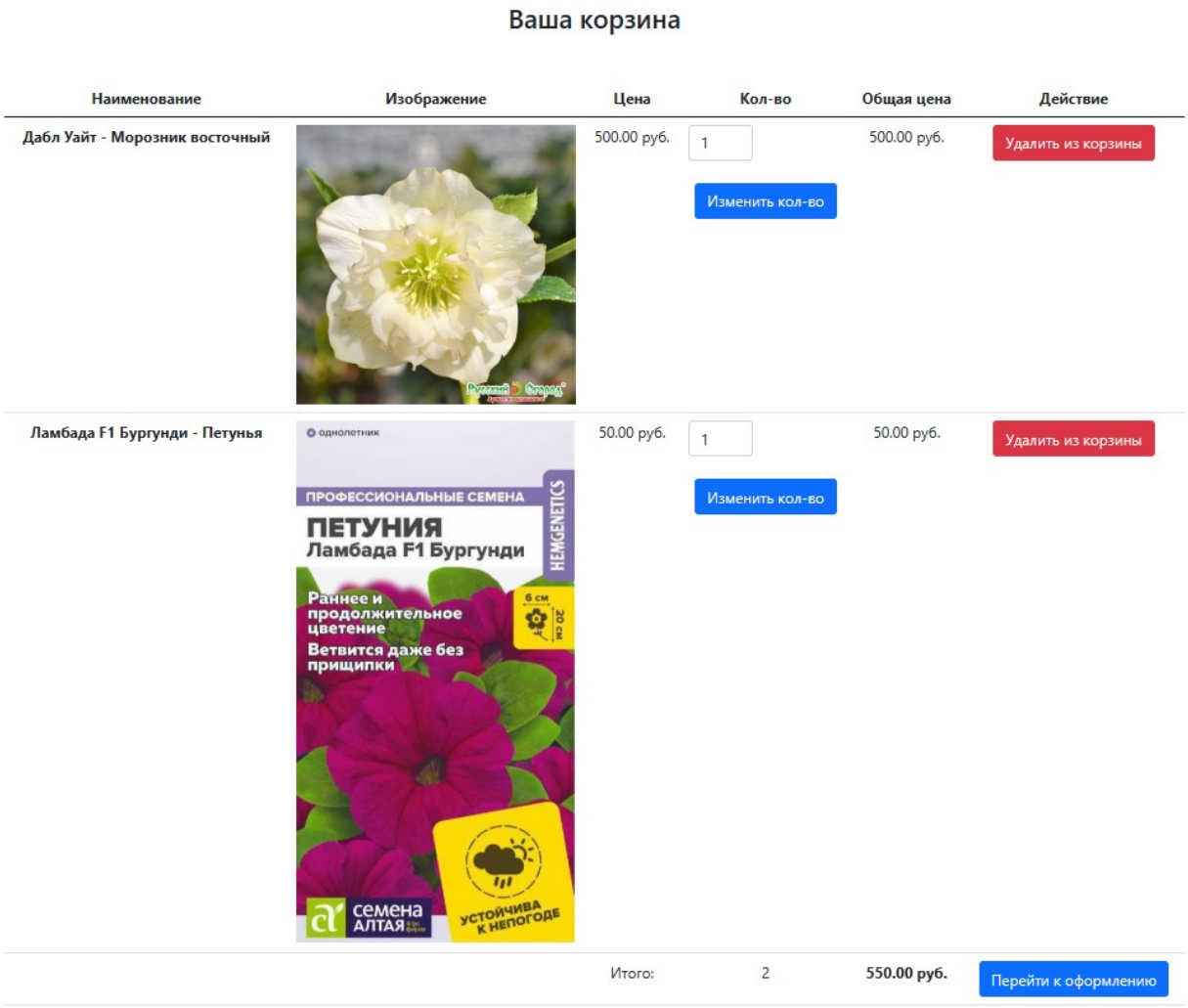

Рисунок 9 – Корзина

Форма заказа представлена на рисунке 10.

Рисунок 10 – Форма заказа

Можно выбрать доставку самовывозом или доставкой.

Если товар на складе закончился, около него появляется надпись «Нет в наличии» (Рисунок 11).

Рисунок 11 – Товара нет в наличии.

Админ панель представлена на рисунке 12.

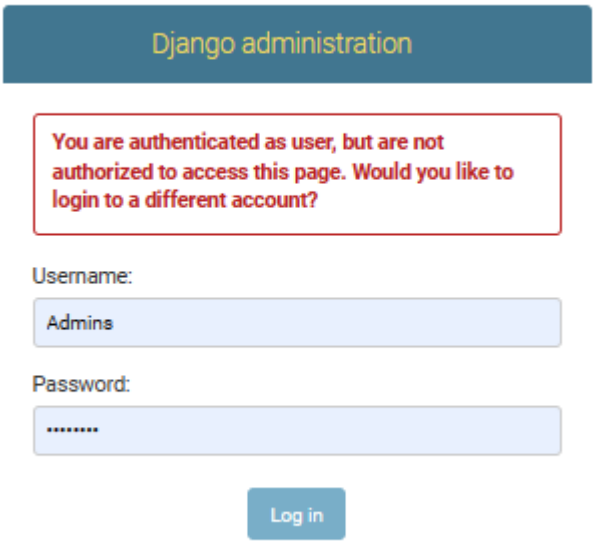

Рисунок 12 – Админ панель

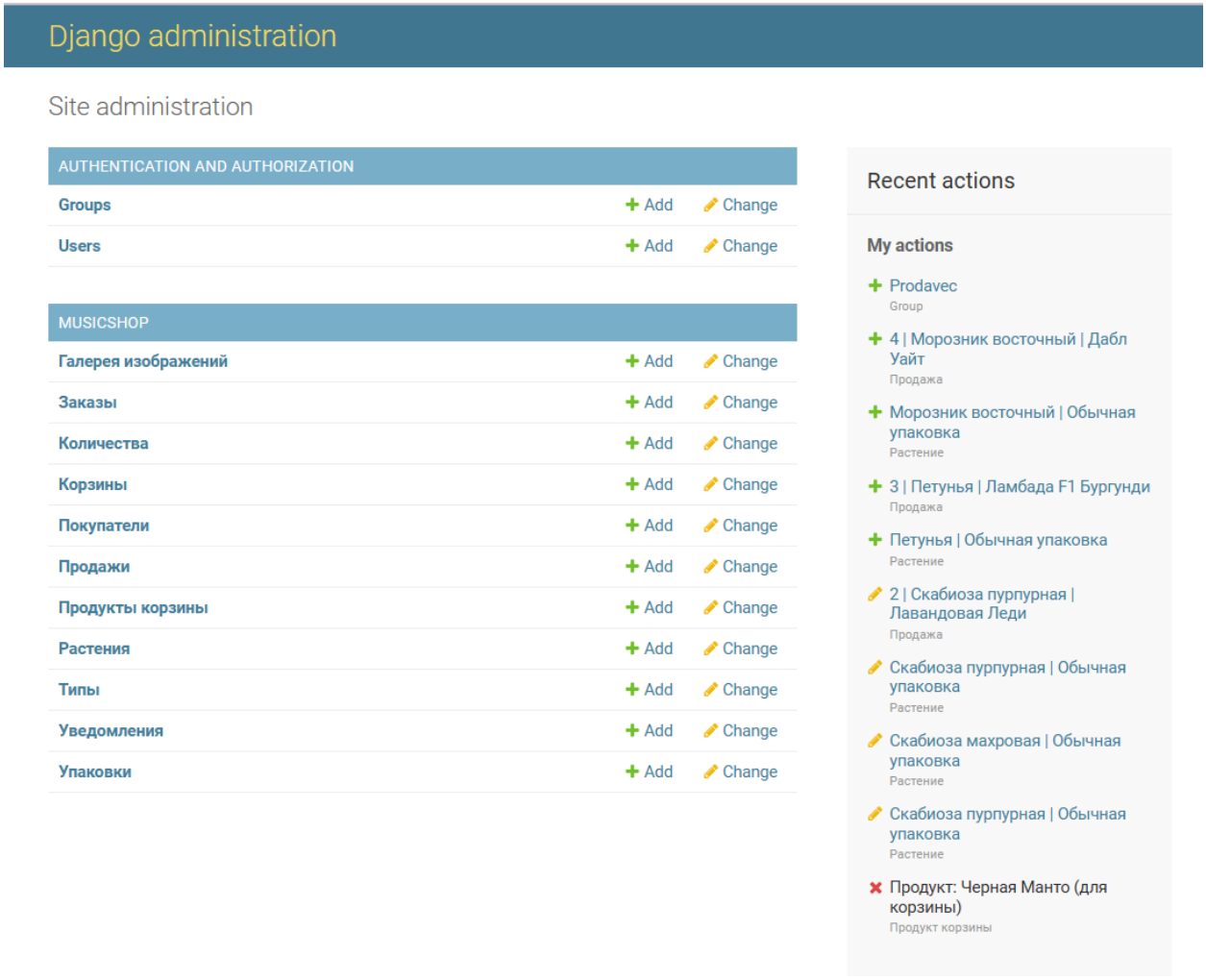

Рисунок 13 – Администрирование

В системе также существует отдельный пользователь «Продавец», которому доступны только таблицы, связанные с продажей и заказом.

ЗАКЛЮЧЕНИЕ

В результате была разработана автоматизированная система магазина живых цветов и семян, которая позволяет осуществлять продажи товаров в интернете, предоставляет удобный интерфейс для администрирования магазина, а интеграция с картами поможет клиенту найти ближайший магазин.

Тестирование программных средств заключалось в определении метода тестирования и выполнение тестов. Результаты функционального тестирования показали корректность работы системы.

Благодаря фреймворку Django удалось значительно ускорить разработку за счет реализации компонентного подхода и принципу DRY. Компонентная архитектура упростила определение и поддержание единого стиля приложения и правил поведения.

Разработанная система в дальнейшем может быть усовершенствован внедрением в приложение возможностей по анализу продаж, например с использованием кластеризации или ассоциативных связей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Элементы интернет-магазина [Электронный ресурс]. – URL: http://www.web.starport.ru/ecommerce/pg3_0.php (дата обращения: 23.04.23).
2. Фаулер, М. UML. Основы / М. Фаулер, К. Скотт. – СПб: Символ-Плюс, 2002. – 192 с.
3. Маклаков, С.В. BPwin и ERwin: CASE-средства для разработки информационных систем / С.В. Маклаков. – М.: Диалог-МИФИ, 2019. – 238 с.
4. Лутц, М. Изучаем Python, 4-е издание / М. Лутц. – СПб: Символ-Плюс, 2010. — 1280 с.
5. Model View Controller [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Model-View-Controller (дата обращения: 02.04.23).
6. Фреймворк Django [Электронный ресурс]. – URL: http://djbook.ru/rel1.8/intro/overview.html (дата обращения: 03.04.23).
7. Общий обзор архитектуры Django [Электронный ресурс]. – URL: http://kutaloweb.com/jeff_forcier_glava_3/obschiy_obzor_arhitektury_django/ (дата обращения: 03.04.23).
8. Разработка сайтов с использованием HTML5 [Электронный ресурс]. – URL: https://www.prcomm-spb.ru/html-5.html (дата обращения: 12.04.23).
9. CSS3 свойства [Электронный ресурс]. – URL: https://webformyself.com/sss3-svojstva/ (дата обращения: 12.04.23).
10. Современный учебник Javascript [Электронный ресурс]. – URL: https://learn.javascript.ru/ (дата обращения: 14.04.23.
11. Сравнение систем управления базами данных [Электронный ресурс]. – URL: http://devacademy.ru/posts/sqlite-vs-mysql-vs-postgresql/ (дата обращения: 14.04.23).

12. Диаграмма вариантов использования [Электронный ресурс]. – URL: http://www.intuit.ru/studies/courses/32/32/lecture/1004 (дата обращения 14.04.23).

13. Основы построения интерфейсов [Электронный ресурс]. – URL: http://www.info-system.ru/interface/begin_design_interface.html (дата обращения: 17.04.23).

14. Принципы разработки интерфейса [Электронный ресурс]. – URL: http://www.cmsmagazine.ru/library/items/usability/principles-of-user-interface-design/ (дата обращения: 17.04.23).

15. Функционально-стоимостной анализ [Электронный ресурс]. – URL: http://www.iso.ru/print/rus/document5954.phtml (дата обращения: 13.04.23).

16. Метрики кода программного обеспечения [Электронный ресурс]. – URL: https://www.viva64.com/ru/a/0045/ (дата обращения: 24.04.23).

17. Качественный анализ программного модуля на основе метрик кода [Электронный ресурс]. – URL: https://vunivere.ru/work18973 (дата обращения: 24.04.23).

18. Тестирование программного обеспечения [Электронный ресурс]. – URL: http://www.dpgrup.ru/testing.htm (дата обращения: 21.04.23).

ПРИЛОЖЕНИЯ

Листинг

```
{% extends 'base.html' %}

{% block content %}

    <h4 class="text-center">Личный кабинет</h4>
    <hr>
        <div class="row">
          <div class="col-4">
            <div class="list-group" id="list-tab"
role="tablist">
              <a class="list-group-item list-group-
item-action active" id="list-orders-list" data-bs-
toggle="list" href="#list-orders" role="tab" aria-
controls="list-orders">Заказы</a>
              <a class="list-group-item list-group-
item-action" id="list-wishlist-list" data-bs-
toggle="list" href="#list-wishlist" role="tab" aria-
controls="list-wishlist">Лист ожидания</a>
            </div>
          </div>
          <div class="col-8">
            <div class="tab-content" id="nav-
tabContent">
              <div class="tab-pane fade show active"
id="list-orders" role="tabpanel" aria-labelledby="list-
orders-list">


                  <table class="table">

                      <thead>

                          <tr>
                              <th scope="col">Номер
заказа</th>
                              <th
scope="col">Статус</th>
                              <th
```

```
scope="col">Сумма</th>
                                <th
scope="col">Детали</th>
                            </tr>

                        </thead>

                    <tbody>

                        {% for order in
customer.orders.all %}

                            <tr>
                                <th scope="row">{{
order.id }}</th>
                                <td>{{
order.get_status_display }}</td>
                                <td>{{
order.cart.final_price }} руб.</td>
                                <td><button
type="button" class="btn btn-primary" data-bs-
toggle="modal"
                                data-bs-
target="#orderDetails-{{ order.id }}">Детали
заказа</button></td>


                            <div class="modal fade"
id="orderDetails-{{ order.id }}" tabindex="-1"
                                aria-
labelledby="orderDetailsLabel-{{ order.id }}" aria-
hidden="true">

                                <div class="modal-
dialog modal-xl">

                                    <div class="modal-
content">

                                        <div
class="modal-header">

                                            <h5
class="modal-title text-center" id="orderDetailsLabel-
{{ order.id }}">

Информация о заказе #{{ order.id }}
```

```
                                                    </h5>
                                                    <button
type="button" class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
                                                </div>
                                                <div
class="modal-body">
                                                    <div
class="row">

                                                        <div
class="col-md-3 text-center mb-2">

<strong>Исполнитель/альбом</strong>
                                                        </div>
                                                        <div
class="col-md-3 text-center mb-2">

<strong>Изображение</strong>
                                                        </div>
                                                         <div
class="col-md-3 text-center mb-2">

<strong>Кол-во</strong>
                                                        </div>
                                                        <div
class="col-md-3 text-center mb-2">

<strong>Общая цена</strong>
                                                        </div>
                                                    <hr>
                                                    {% for item
in order.cart.products.all %}
                                                        <div
class="col-md-3 mb-3 text-center">

<strong>

<a href="{{ item.content_object.artist.get_absolute_url
}}"

class="text-decoration-none">

{{ item.content_object.artist.name }}

</a> - <a href="{{ item.content_object.get_absolute_url
```

```
}}"

class="text-decoration-none">

{{ item.content_object.name }}

</a>

</strong>

                                                    </div>
                                                    <div
class="col-md-3 mb-3 text-center">

<img src="{{ item.content_object.image.url }}"
class="img-fluid">
                                                    </div>
                                                    <div
class="col-md-3 mb-3 text-center">
                                                        {{
item.qty }} шт.
                                                    </div>
                                                    <div
class="col-md-3 mb-3 text-center">
                                                        {{
item.final_price }} руб.
                                                    </div>
                                                {% endfor
%}

                                            </div>
                                        </div>
                                        <div class="modal-
footer">
                                            <button
type="button" class="btn btn-secondary" data-bs-
dismiss="modal">
                                                Закрыть
                                            </button>
                                        </div>

                                    </div>

                                </div>
```

```
                            </tr>

                        {% endfor %}
                    </tbody>

                </table>



            </div>
            <div class="tab-pane fade" id="list-
wishlist" role="tabpanel" aria-labelledby="list-
wishlist-list">

                <div class="row">
                    {% for album in
customer.wishlist.all %}

                        <div class="card col-md-4 p-0
mb-3 mt-3">

                            <img src="{{
album.image.url }}" class="card-img-top">

                            <div class="card-body text-
center">

                                <h5 class="card-
title"><a href="{{ album.artist.get_absolute_url }}"
class="text-decoration-none">
                                    {{
album.artist.name }}
                                </a></h5>
                                <h5 class="card-
title"><a href="{{ album.get_absolute_url }}"
class="text-decoration-none">
                                    {{ album.name
}}
                                </a></h5>
                            </div>

                            <ul class="list-group list-
group-flush">

                                <li class="list-
```

```
group-item">
                                        Носитель:
<strong>{{ album.media_type.name }}</strong>
                                    </li>
                                    <li class="list-
group-item">
                                        Дата релиза:
<strong>{{ album.release_date|date:"d-m-Y" }}</strong>
                                    </li>
                                    <li class="list-
group-item">
                                        Жанр: <strong
class="badge bg-dark">{{ album.artist.genre.name
}}</strong>
                                    </li>
                                    <li class="list-
group-item">
                                        Наличие: {% if
album.stock %}<strong class="badge bg-success">
                                        Есть в
наличии</strong>{% else %}
                                        <strong
class="badge bg-danger">Нет в наличии</strong>{% endif
%}
                                    </li>
                                </ul>
                                <div class="card-body text-
center">
                                    <a href="{% url
'remove_from_wishlist' album_id=album.id %}"
                                       class="btn btn-
danger">Удалить из ожидаемого <i class="fas fa-
star"></i></a>
                                </div>

                            </div>

                        {% endfor %}
                    </div>

                </div>
            </div>
        </div>
{% endblock content %}
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/
all.css" integrity="sha384-
AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELG
roGkvsg+p" crossorigin="anonymous"/>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist
/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztc
QTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <link rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/
all.css" integrity="sha384-
AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELG
roGkvsg+p" crossorigin="anonymous"/>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-
light">
  <div class="container-fluid">
    <a class="navbar-brand" href="{% url 'base'
%}">Musicshop</a>
    <button class="navbar-toggler" type="button" data-
bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-
controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse"
id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-
current="page" href="{% url 'base' %}">Главная</a>
        </li>
          {% if not request.user.is_authenticated %}
```

```
          <li class="nav-item">
            <a class="nav-link" href="{% url 'login'
%}">Авторизация</a>
          </li>
            <li class="nav-item">
            <a class="nav-link" href="{% url
'registration' %}">Регистрация</a>
          </li>
          {% else %}
            <li class="nav-item">
            <a class="nav-link" href="{% url 'account'
%}">Личный кабинет</a>
          </li>
            <li class="nav-item">
            <a class="nav-link" href="{% url 'logout'
%}">Выйти</a>
          </li>
              <li class="nav-item dropdown">
                <a class="nav-link drowdown-toggle"
href="#" id="navbarDropdown" role="button" data-bs-
toggle="dropdown"
                   aria-expanded="false">

                    Уведомления <i class="fas fa-
bell"></i>
                    <span class="badge bg-{% if
notifications.count %}danger{% else %}secondary{% endif
%}">
                        {{ notifications.count }}
                    </span>
                </a>
                        <ul class="dropdown-menu" aria-
labelledby="navbarDropdown">

                            {% if notifications %}
                                {% for n in
notifications %}
                                    <li><span
class="dropdown-item">{{ n.text|safe }}</span></li>
                                {% endfor %}
                                    <li><hr
class="dropdown-divider"></li>
                                    <li><a href="{% url
'clear-notifications' %}"
```

```
class="drowpdown-item">Пометить все, как
прочитанное</a></li>
                                {% else %}
                                    <li><a class="dropdown-
item" href="#">Нет новых уведомлений</a></li>
                                {% endif %}
                            </ul>
                </li>
            {% endif %}
        </ul>
          {% if request.user.is_authenticated %}
          <ul class="navbar-nav">
              <li class="nav-item"><a href="{% url 'cart'
%}" class="nav-link"><i class="fas fa-shopping-
cart"></i> <span class="badge bg-danger">{{
cart.products.count }}</span></a></li>
          </ul>
          {% endif %}
      </div>
    </div>
</nav>

<div class="container">
    <div class="site-header"><div>
    <h1>Магазин живых цветов и семян "Нотка"</h1>
    <hr>

        <button id="sort-asc">Цена min</button>
        <button id="sort-desc">Цена max</button>
<div class="container">
    {% block content %}
        <div class="col-md-12">
                <div class="row">
                    <div class="col-md-4"></div>
                    <div class="col-md-4">
                        <h4 class="text-center">Хит
продаж текущего месяца</h4>
                        <a href="{{
month_bestseller.get_absolute_url }}">{{ month_bestsell
er.name }}</a>
                    </div>
                    <div class="col-md-4"></div>
                </div>
            <div class="row">
                {% for album in albums %}
```

```
                    <div class="card col-md-3 p-0 mb-3
mt-3">

                        <img src="{{ album.image.url
}}" class="card-img-top">
                        <div class="card-body text-
center">
                        <h5 class="card-title"><a
href="{{ album.artist.get_absolute_url }}" class="text-
decoration-none">
                            {{ album.artist.name }}
                        </a></h5>
                        <h5 class="card-title"><a
href="{{ album.get_absolute_url }}" class="text-
decoration-none">
                            {{ album.name }}
                        </a></h5>
                        </div>

                    <ul class="list-group list-group-
flush">

                        <li class="list-group-
item">Носитель: <strong>{{ album.media_type.name
}}</strong></li>
                        <li class="list-group-
item">Дата изготовления: <strong>{{ album.release_date|
date:"d-m-Y" }}</strong></li>
                        <li class="list-group-
item">Упаковка: <strong>{{ album.artist.genre.name
}}</strong></li>
                        <li class="list-group-
item">Цена: <strong>{{ album.price }}</strong></li>
                        <li class="list-group-item">

                            Наличие: {% if album.stock
%}<strong class="badge bg-success">
                            Есть в наличии - {{
album.stock }} шт.
                        </strong>{% else %}
                            <strong class="badge bg-
danger">Нет в наличии</strong>
                        {% endif %}

                        </li>
```

```
                    </ul>

                    <div class="card-body text-center">

                        {% if
request.user.is_authenticated %}

                            {% if album.stock %}
                                {% if album not in
cart.products_in_cart %}
                                    <a href="{% url
'add_to_cart' ct_model=album.ct_model slug=album.slug
%}">
                                        <button
class="btn btn-primary">
                                    Добавить в корзину
                                        </button>
                                    </a>
                                {% else %}
                                    <a href="#"
class="btn btn-default" disabled="">Добавлен в
корзину</a>
                                {% endif %}
                            {% else %}
                                {% if album not in
request.user.customer.wishlist.all %}
                                    <a href="{% url
'add_to_wishlist' album_id=album.id %}" class="btn btn-
warning">
                                    Добавить в
ожидаемое
                                    </a>
                                {% else %}
                                    <a href="#"
class="btn btn-default" disabled="">Добавлен в
ожидаемое</a>
                                {% endif %}
                            {% endif %}
                        {% else %}
                            <a href="#" class="btn btn-
default" disabled="">
                            Авторизуйтесь или
зарегистрируйтесь, только авторизованные пользователи
                            могут осуществлять заказы
                            </a>
```

```
                        {% endif %}
                    </div>

                    </div>

                {% endfor %}
            </div>

        </div>
    {% endblock content %}
</div>




</body>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/
dist/umd/popper.min.js" integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8
NT4GN1R8p" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/
js/bootstrap.min.js" integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0Q
KbrJ0EcQF" crossorigin="anonymous"></script>
</html>
```

```
{% extends 'base.html' %}


{% block content %}

    <h3 class="text-center mt-5 mb-5">Ваша корзина {%
if not cart.products.count %}пуста{% endif %}</h3>

    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-success alert-
dismissible fade show" role="alert">
                <strong>{{ message }}</strong>
                <button type="button" class="btn-close"
data-bs-dismiss="alert" aria-lable="Close"></button>
            </div>
```

```
        {% endfor %}
    {% endif %}

    {% if cart.products.count %}

        <table class="table text-center">
            <thead>
                <tr>
                    <th scope="col">Наименование</th>
                    <th scope="col">Изображение</th>
                    <th scope="col">Цена</th>
                    <th scope="col">Кол-во</th>
                    <th scope="col">Общая цена</th>
                    <th scope="col">Действие</th>
                </tr>
            </thead>
            <tbody>

                {% for item in cart.products.all %}
                    <tr>
                        <th scope="row">{{
item.display_name }}</th>
                        <td class="w-25"><img src="{{
item.content_object.image.url }}" class="img-
fluid"></td>
                        <td>{{
item.content_object.price }} руб.</td>
                        <td>

                            <form action="{% url
'change_qty' ct_model=item.content_object.ct_model
slug=item.content_object.slug %}"
                                  method="POST">
                                {% csrf_token %}
                                <input type="number"
class="form-control" style="width: 70px;" name="qty"
min="1" value="{{ item.qty }}">
                                <br>
                                <input type="submit"
class="btn btn-primary" value="Изменить кол-во">
                            </form>

                        </td>
                        <td>{{ item.final_price }}
руб.</td>
```

```
                        <td>
                            <a href="{% url
'delete_from_cart'
ct_model=item.content_object.ct_model
slug=item.content_object.slug %}">
                                <button class="btn btn-
danger">Удалить из корзины</button>
                            </a>
                        </td>
                    </tr>
                {% endfor %}
                <tr>
                    <td colspan="2"></td>
                    <td>Итого:</td>
                    <td>{{ cart.total_products }}</td>
                    <td><strong>{{ cart.final_price }}
руб.</strong></td>
                    <td><a href="{% url 'checkout'
%}"><button class="btn btn-primary">Перейти к
оформлению</button></a></td>
                </tr>
            </tbody>
        </table>

    {% endif %}


{% endblock content %}
```

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}

{% block content %}

    <h5 class="text-center mt-5 mb-5">Оформление
заказа</h5>
    {% if messages %}
        {% for message in messages %}

            <div class="alert alert-success fade show"
role="alert">
                <strong>{{ message }}</strong>
```

```
                <button type="button" class="btn-close"
data-bs-dismiss="alert" aria-label="Close"></button>
            </div>

        {% endfor %}

    {% endif %}
        <table class="table text-center">
            <thead>
                <tr>
                    <th scope="col">Наименование</th>
                    <th scope="col">Изображение</th>
                    <th scope="col">Цена</th>
                    <th scope="col">Кол-во</th>
                    <th scope="col">Общая цена</th>
                </tr>
            </thead>
            <tbody>

                {% for item in cart.products.all %}
                    <tr>
                        <th scope="row">{{
item.display_name }}</th>
                        <td class="w-25"><img src="{{
item.content_object.image.url }}" class="img-
fluid"></td>
                        <td>{{
item.content_object.price }} руб.</td>
                        <td>{{ item.qty }} шт.</td>
                        <td>{{ item.final_price }}
руб.</td>
                    </tr>
                {% endfor %}
                <tr>
                    <td colspan="2"></td>
                    <td>Итого:</td>
                    <td>{{ cart.total_products }}</td>
                    <td><strong>{{ cart.final_price }}
руб.</strong></td>
                </tr>
            </tbody>
        </table>

        <h3 class="text-center mt-5 mb-5">Форма
заказа</h3>
```

```
        <form action="{% url 'make-order' %}"
method="POST">
            {% csrf_token %}
            {{ form|crispy }}
            <input type="submit" class="btn btn-success
btn-block mb-3" value="Оформить заказ">
        </form>

{% endblock content %}
```

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}

{% block content %}

    <div class="row" style="margin-bottom: 200px;
margin-top: 160px;">
        <div class="col-md-6 offset-md-3">
            <h3 class="text-center">Авторизация</h3>
            <hr>
            <form action="{% url 'login' %}"
method="POST">
                {% csrf_token %}
                {{ form|crispy }}
                <input type="submit" class="btn btn-
success btn-block" value="Войти">
            </form>
        </div>

    </div>

{% endblock content %}
```

```
{% extends 'base.html' %}
{% load crispy_forms_tags %}

{% block content %}

    <div class="row" style="margin-bottom: 200px;
margin-top: 160px;">
```

```
        <div class="col-md-6 offset-md-3">
            <h3 class="text-center">Регистрация</h3>
            <hr>
            <form action="{% url 'registration' %}"
method="POST">
                {% csrf_token %}
                {{ form|crispy }}
                <input type="submit" class="btn btn-
success btn-block" value="Зарегистрироваться">
            </form>
        </div>

    </div>

{% endblock content %}
```

```
from django.shortcuts import render
from django import views
from django.contrib import messages
from django.http import HttpResponseRedirect
from django.contrib.contenttypes.models import
ContentType
from django.contrib.auth import authenticate, login
from django.db import transaction

from .models import Artist, Album, Customer,
CartProduct, Notification
from .forms import LoginForm, RegistrationForm,
OrderForm
from .mixins import CartMixin, NotificationsMixin
from utils import recalc_cart


class BaseView(CartMixin, NotificationsMixin,
views.View):

    def get(self, request, *args, **kwargs):
        albums = Album.objects.all().order_by('-id')
[:5]
        month_bestseller, month_bestseller_qty =
Album.objects.get_month_bestseller()
        context = {
            'albums': albums,
            'cart': self.cart,
```

```
            'notifications':
self.notifications(request.user)
        }
        if month_bestseller:
            context.update({"month_bestseller":
month_bestseller, "month_bestseller_qty":
month_bestseller_qty})
        return render(request, 'base.html', context)


class ArtistDetailView(NotificationsMixin,
views.generic.DetailView):

    model = Artist
    template_name = 'artist/artist_detail.html'
    slug_url_kwarg = 'artist_slug'
    context_object_name = 'artist'


class AlbumDetailView(CartMixin, NotificationsMixin,
views.generic.DetailView):
    model = Album
    template_name = 'album/album_detail.html'
    slug_url_kwarg = 'album_slug'
    context_object_name = 'album'


class LoginView(views.View):

    def get(self, request, *args, **kwargs):
        form = LoginForm(request.POST or None)
        context = {
            'form': form
        }
        return render(request, 'login.html', context)

    def post(self, request, *args, **kwargs):
        form = LoginForm(request.POST or None)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(username=username,
password=password)
            if user:
                login(request, user)
```

```
                return HttpResponseRedirect('/')
        context = {
            'form': form
        }
        return render(request, 'login.html', context)


class RegistrationView(views.View):

    def get(self, request, *args, **kwargs):
        form = RegistrationForm(request.POST or None)
        context = {
            'form': form
        }
        return render(request, 'registration.html',
context)

    def post(self, request, *args, **kwargs):
        form = RegistrationForm(request.POST or None)
        if form.is_valid():
            new_user = form.save(commit=False)
            new_user.username =
form.cleaned_data['username']
            new_user.email = form.cleaned_data['email']
            new_user.first_name =
form.cleaned_data['first_name']
            new_user.last_name =
form.cleaned_data['last_name']
            new_user.save()

new_user.set_password(form.cleaned_data['password'])
            new_user.save()
            Customer.objects.create(
                user=new_user,
                phone=form.cleaned_data['phone'],
                address=form.cleaned_data['address']
            )
            user =
authenticate(username=form.cleaned_data['username'],
password=form.cleaned_data['password'])
            login(request, user)
            return HttpResponseRedirect('/')
        context = {
            'form': form
        }
        return render(request, 'registration.html',
```

```
context)


class AccountView(CartMixin, NotificationsMixin,
views.View):

    def get(self, request, *args, **kwargs):
        customer =
Customer.objects.get(user=request.user)
        context = {
            'customer': customer,
            'cart': self.cart,
            'notifications':
self.notifications(request.user)
        }
        return render(request, 'account.html', context)


class CartView(CartMixin, NotificationsMixin,
views.View):

    def get(self, request, *args, **kwargs):
        return render(request, 'cart.html', {"cart":
self.cart, 'notifications':
self.notifications(request.user)})


class AddToCartView(CartMixin, views.View):

    def get(self, request, *args, **kwargs):
        ct_model, product_slug =
kwargs.get('ct_model'), kwargs.get('slug')
        content_type =
ContentType.objects.get(model=ct_model)
        product =
content_type.model_class().objects.get(slug=product_slu
g)
        cart_product, created =
CartProduct.objects.get_or_create(
            user=self.cart.owner, cart=self.cart,
content_type=content_type, object_id=product.id
        )
        if created:
            self.cart.products.add(cart_product)
        recalc_cart(self.cart)
```

```
        messages.add_message(request, messages.INFO,
"Товар успешно добавлен")
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class DeleteFromCartView(CartMixin, views.View):

    def get(self, request, *args, **kwargs):
        ct_model, product_slug =
kwargs.get('ct_model'), kwargs.get('slug')
        content_type =
ContentType.objects.get(model=ct_model)
        product =
content_type.model_class().objects.get(slug=product_slu
g)
        cart_product = CartProduct.objects.get(
            user=self.cart.owner, cart=self.cart,
content_type=content_type, object_id=product.id
        )
        self.cart.products.remove(cart_product)
        cart_product.delete()
        recalc_cart(self.cart)
        messages.add_message(request, messages.INFO,
"Товар успешно удален")
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class ChangeQTYView(CartMixin, views.View):

    def post(self, request, *args, **kwargs):
        ct_model, product_slug =
kwargs.get('ct_model'), kwargs.get('slug')
        content_type =
ContentType.objects.get(model=ct_model)
        product =
content_type.model_class().objects.get(slug=product_slu
g)
        cart_product = CartProduct.objects.get(
            user=self.cart.owner, cart=self.cart,
content_type=content_type, object_id=product.id
        )
        qty = int(request.POST.get('qty'))
        cart_product.qty = qty
```

```
        cart_product.save()
        recalc_cart(self.cart)
        messages.add_message(request, messages.INFO,
"Кол-во успешно изменено")
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class AddToWishlist(views.View):

    @staticmethod
    def get(request, *args, **kwargs):
        album =
Album.objects.get(id=kwargs['album_id'])
        customer =
Customer.objects.get(user=request.user)
        customer.wishlist.add(album)
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class ClearNotificationsView(views.View):

    @staticmethod
    def get(request, *args, **kwargs):

Notification.objects.make_all_read(request.user.custome
r)
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class RemoveFromWishListView(views.View):

    @staticmethod
    def get(request, *args, **kwargs):
        album =
Album.objects.get(id=kwargs['album_id'])
        customer =
Customer.objects.get(user=request.user)
        customer.wishlist.remove(album)
        return
HttpResponseRedirect(request.META['HTTP_REFERER'])


class CheckoutView(CartMixin, NotificationsMixin,
```

```
views.View):

    def get(self, request, *args, **kwargs):
        form = OrderForm(request.POST or None)
        context = {
            'cart': self.cart,
            'form': form,
            'notifications':
self.notifications(request.user)
        }
        return render(request, 'checkout.html',
context)


class MakeOrderView(CartMixin, views.View):

    @transaction.atomic
    def post(self, request, *args, **kwargs):

        form = OrderForm(request.POST or None)
        customer =
Customer.objects.get(user=request.user)
        if form.is_valid():
            out_of_stock = []
            more_than_on_stock = []
            out_of_stock_message = ""
            more_than_on_stock_message = ""
            for item in self.cart.products.all():
                if not item.content_object.stock:
                    out_of_stock.append(' - '.join([

item.content_object.artist.name,
item.content_object.name
                    ]))
                if item.content_object.stock and
item.content_object.stock < item.qty:
                    more_than_on_stock.append(
                        {'product': ' -
'.join([item.content_object.artist.name,
item.content_object.name]),
                         'stock':
item.content_object.stock, 'qty': item.qty}
                    )
            if out_of_stock:
                out_of_stock_products = ',
'.join(out_of_stock)
```

```
                out_of_stock_message = f'Товара уже нет
в наличии: {out_of_stock_products}. \n'

            if more_than_on_stock:
                for item in more_than_on_stock:
                    more_than_on_stock_message +=
f'Товар: {item["product"]}. ' \
                                                f'В
наличии: {item["stock"]}. ' \

f'Заказано: {item["qty"]}\n'
            error_message_for_customer = ""
            if out_of_stock:
                error_message_for_customer =
out_of_stock_message + '\n'
            if more_than_on_stock_message:
                error_message_for_customer +=
more_than_on_stock_message + '\n'

            if error_message_for_customer:
                messages.add_message(request,
messages.INFO, error_message_for_customer)
                return
HttpResponseRedirect('/checkout/')

            new_order = form.save(commit=False)
            new_order.customer = customer
            new_order.first_name =
form.cleaned_data['first_name']
            new_order.last_name =
form.cleaned_data['last_name']
            new_order.phone =
form.cleaned_data['phone']
            new_order.address =
form.cleaned_data['address']
            new_order.buying_type =
form.cleaned_data['buying_type']
            new_order.order_date =
form.cleaned_data['order_date']
            new_order.comment =
form.cleaned_data['comment']
            new_order.save()

            self.cart.in_order = True
            self.cart.save()
```

```
            new_order.cart = self.cart
            new_order.save()
            customer.orders.add(new_order)

            for item in self.cart.products.all():
                item.content_object.stock -= item.qty
                item.content_object.save()

            messages.add_message(request,
messages.INFO, 'Спасибо за заказ! Менеджер с Вами
свяжется')
            return HttpResponseRedirect('/')
        return HttpResponseRedirect('/checkout/')
```