**Веб-приложение для управления заказами на производстве: путь от задачи до реализации на примере ООО “KЗ ПРОМЭКО”**

***Чубатов Д.В.***

*студент*

*Сибирский государственный университет геосистем и технологий, институт геодезии и менеджмента, Новосибирск, Россия*

*E-mail:* [*dima.don84@gmail.com*](mailto:dima.don84@gmail.com)

Конечным продуктом любого производства является заказ, который должен быть изготовлен и доставлен клиенту в срок и с высоким качеством. Для управления этим процессом необходима эффективная система управления заказами, которая позволит оптимизировать процесс, ускорить его и обеспечить высокое качество продукта.

Мы рассмотрим путь от задачи до реализации веб-приложения для управления движением заказов на производстве на примере ООО “KЗ ПРОМЭКО”. Разработка такой системы управления позволяет значительно сократить использование бумажных носителей и перенести процесс управления движения заказов из ручного режима на электронный.

ООО “KЗ ПРОМЭКО” - динамично развивающееся предприятие, специализирующееся на производстве кабельной продукции. Бизнес-процесс планирования и управления производством в компании был частично автоматизирован, что приводило к задержкам и ошибкам

Область управления заказами на производстве является важной для эффективной работы любой компании. В настоящее время многие компании переходят на использование систем для управления заказами вместо использования бумажных носителей и ручного управления.

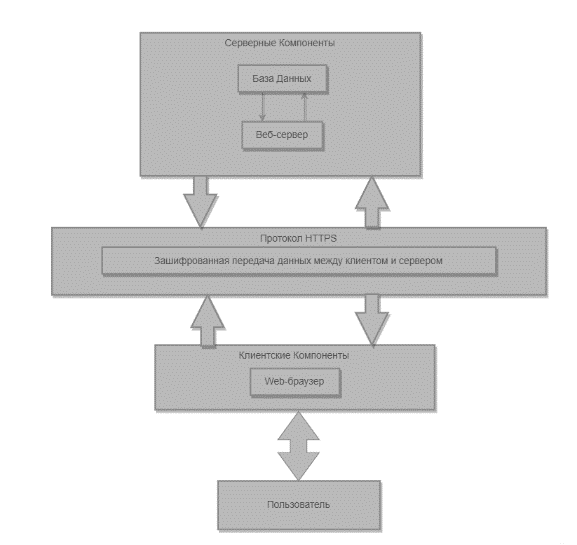
Веб-приложение является программным обеспечением, которое работает на сервере и обеспечивает доступ к данным через интернет, используя протоколы HTTP/HTTPS [1].

Рис. 1 - Диаграмма взаимодействия между пользователями, клиентскими и серверными компонентами.

Требования к приложению включали: возможность создания, редактирования и удаления заказов в ручном режиме и из файла, планирование производства, контроль сроков и этапов выполнения заказов, хранение истории заказов и возможность отчетности.

Для разработки веб-приложения был выбран фреймворк Django [2] на языке программирования Python. Django — это свободный фреймворк высокого уровня для веб-разработки, который позволяет разрабатывать быстрые и масштабируемые веб-приложения. Django также обеспечивает высокий уровень безопасности приложения и удобный интерфейс для работы с базами данных.

Для хранения данных была выбрана реляционная база данных PostgreSQL [3]. PostgreSQL — это мощная, надежная и расширяемая СУБД, которая поддерживает множество функций, таких как хранение и обработка геоданных и работа с JSON.

Для упрощения развертывания и управления веб-приложением был выбран Docker [4] — это платформа для разработки, доставки и запуска приложений в контейнерах. Docker позволяет создавать изолированные среды для приложений, что облегчает их развертывание и управление.

Проект будет разделен на модули.

Модули являются основными компонентами проекта, которые позволяют организовать код и логику приложения в более структурированном и управляемом виде.

В рамках нашего проекта были созданы следующие модули:

Модуль "users" - отвечает за управление пользователями системы, включая регистрацию, авторизацию и профили пользователей.

Модуль "orders" - отвечает за управление заказами на производстве, включая создание, редактирование и отслеживание статуса заказов.

Модуль "shelf" - отвечает за временное хранение заказов, отмеченных для отправки в производство на определенную операцию, и формированию новых заказов в производство.

Модуль "actions" - отвечает за сохранение и отображение действий пользователей в системе, такими как добавление, изменение и удаление данных.

Модуль "search" - отвечает за реализацию поиска по базе данных и предоставление результатов пользователю, и выгрузку по поисковому запросу в файл Excel.

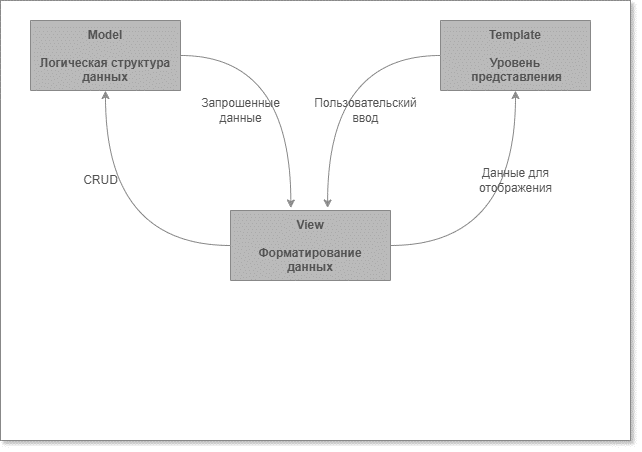
При разработке каждого модуля руководствовались принципам модульной архитектуры MVT (рис. 2) и разделили логику на несколько уровней: Модели (Models), Представления (Views) и Шаблоны (Templates).

Рис. 2 – Диаграмма архитектуры MVT.

Модели отвечают за работу с данными приложения и обеспечивают доступ к информации, хранящейся в базе данных. Представления обрабатывают запросы от пользователя и управляют тем, как информация будет представлена на экране. Шаблоны отвечают за визуальное представление данных и содержат HTML-код [5], определяющий, как информация будет отображаться на странице.

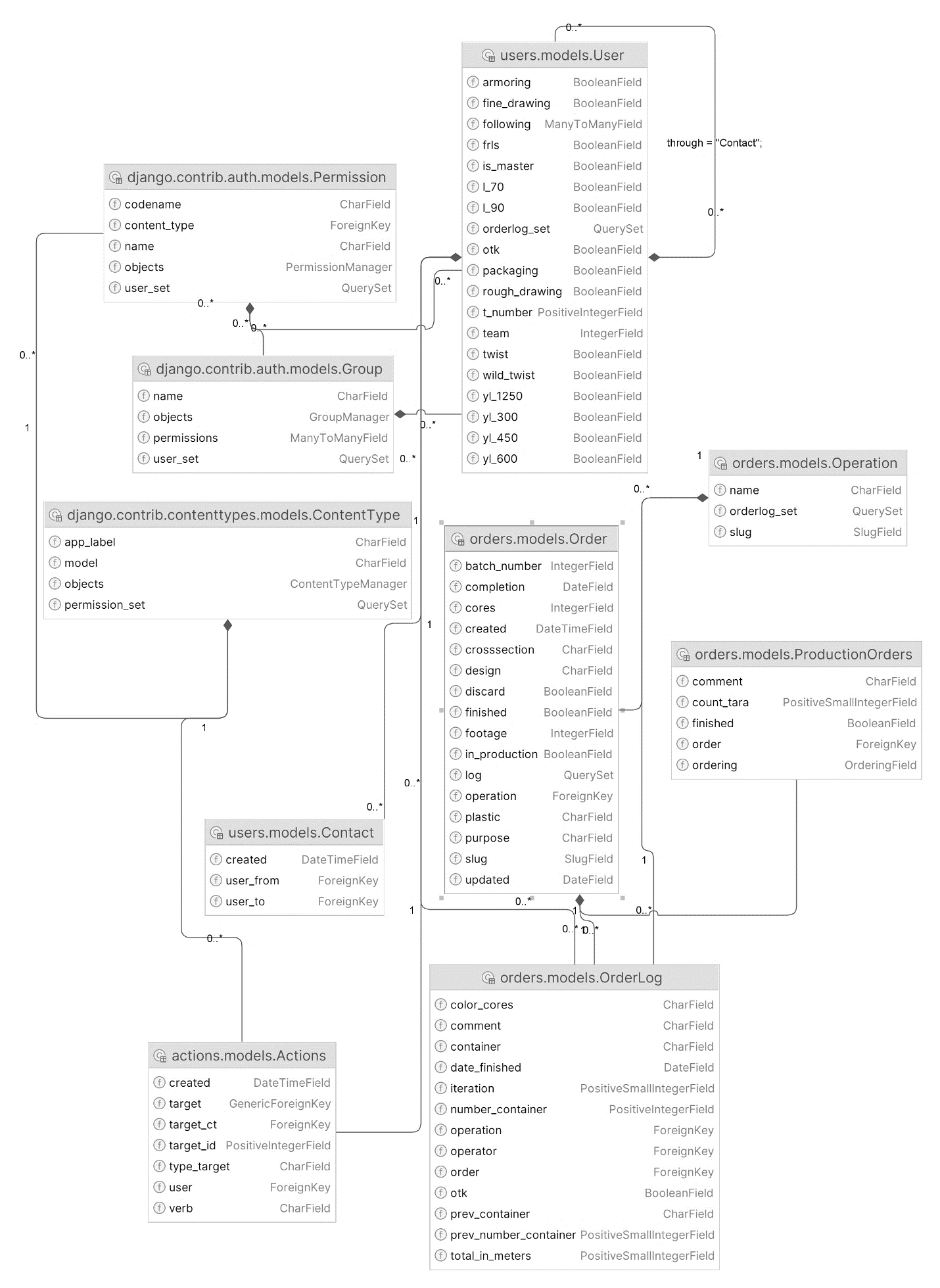
В процессе разработки веб-приложения для управления заказами на производстве была проведена работа по проектированию базы данных. Для этого были определены сущности, связи между ними, а также атрибуты каждой сущности (рис. 3).

Рис. 3 – диаграмма сущностей базы данных и их связей.

Первым шагом была разработана модель для пользователей (User). Данная модель содержит информацию о пользователях системы: их идентификатор, имя пользователя, адрес электронной почты и пароль.

Затем была создана модель заказов (Order), которая содержит информацию о заказах, поступающих на производство: их идентификатор, статус, дата создания и дата завершения, информацию о изготавливаемом кабеле.

Для управления производственным процессом была разработана модель ProductionOrder, которая содержит информацию о заказе на производственной операции: его идентификатор, статус, дата создания и дата завершения, а также информация о заказе, связанном с производственным заказом.

Модель Operation была создана для хранения информации о конкретных операциях, необходимых для выполнения производственного заказа. Она содержит идентификатор операции, ее наименование.

Для ведения журнала заказов была создана модель OrderLog. Она содержит информацию о действиях, выполняемых над заказами при прохождении производственных операций.

Для реализации хранения действий операторов с заказами была разработана модель Actions. Она содержит информацию о действиях, которые были выполнены над заказами: их наименование, описание и код.

Наконец, была разработана модель Contact для реализации подписки пользователям друг на друга и получать уведомления о действиях других пользователей, на которых они подписаны.

Все вышеперечисленные модели были разработаны с использованием ORM Django и хранятся в PostgreSQL базе данных.

После того, как была разработана модель базы данных, началась разработка бекенда приложения.

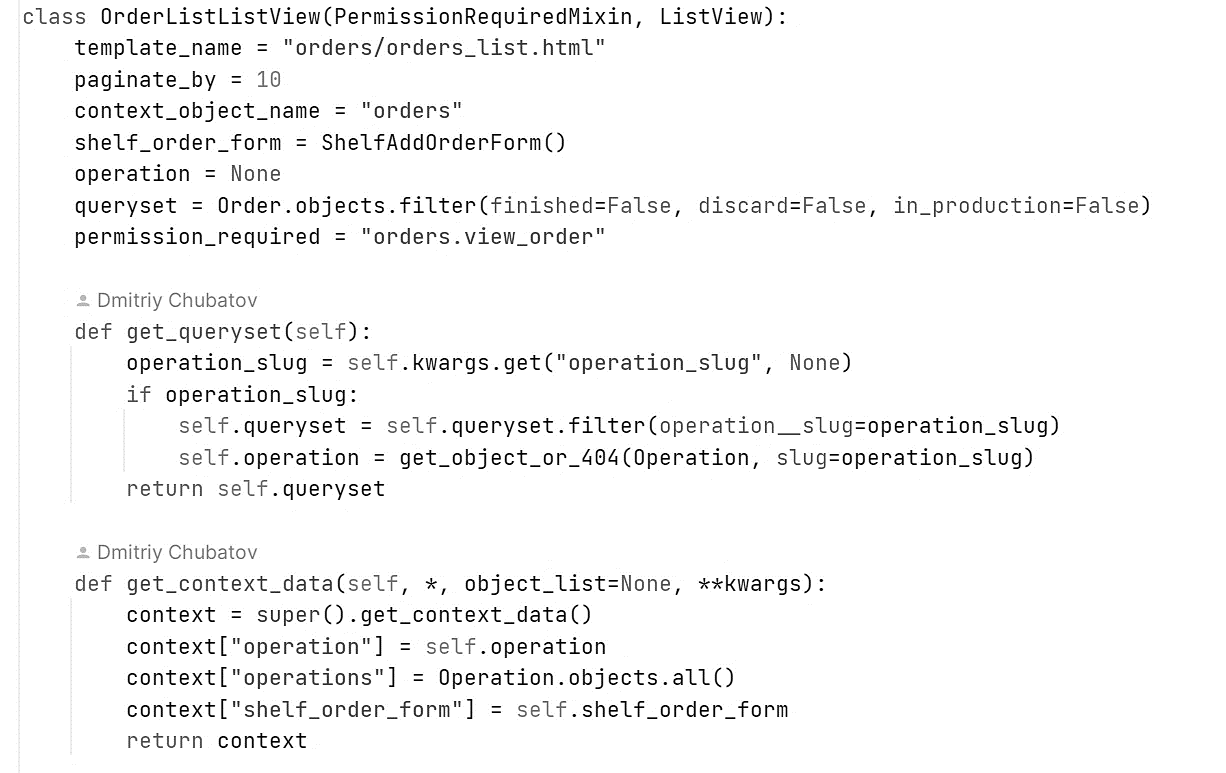
Основным компонентом бекенда является views.py - модуль, который обрабатывает запросы от фронтэнда и возвращает соответствующий ответ. В процессе разработки было создано несколько классов-обработчиков (рис. 4) для каждого модуля приложения.

Рис. 4 – Класс-обработчик представления, отвечает за обработку запроса по отображению заказов, всех или находящихся на определенной операции.

Кроме того, в рамках разработки бекенда были созданы шаблоны фронтэнда, которые отображают данные, полученные от сервера. Django использует шаблонизатор, который позволяет создавать шаблоны (рис. 5) с использованием специальных тегов и фильтров.

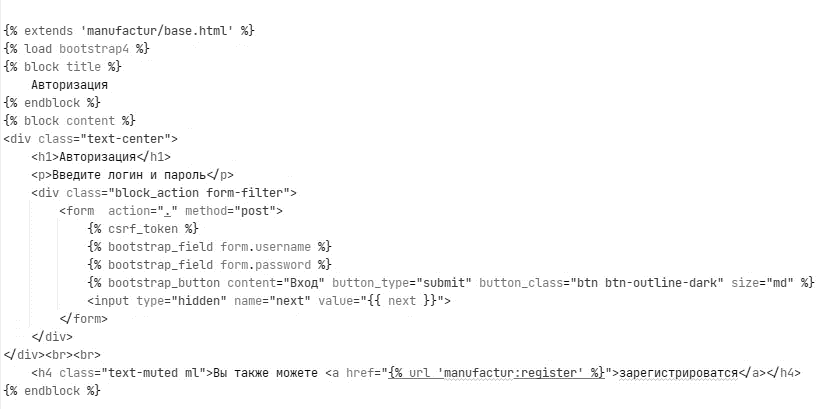
Для улучшения пользовательского интерфейса и удобства работы с веб-приложением использовались JavaScript библиотеки и фреймворки, такие как jQuery и Bootstrap.

Рис. 5 – Шаблон фронтэнда авторизации пользователей.

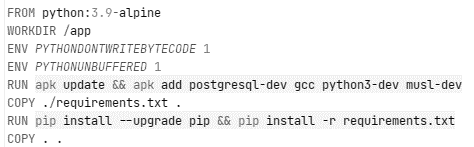
В нашем проекте мы использовали Docker для развертывания двух контейнеров: контейнера с базой данных и контейнера с веб-приложением. Для этого мы создали файл Dockerfile (рис. 6), в котором описали необходимые компоненты и настройки для каждого контейнера.

Рис. 6 – Dockerfile

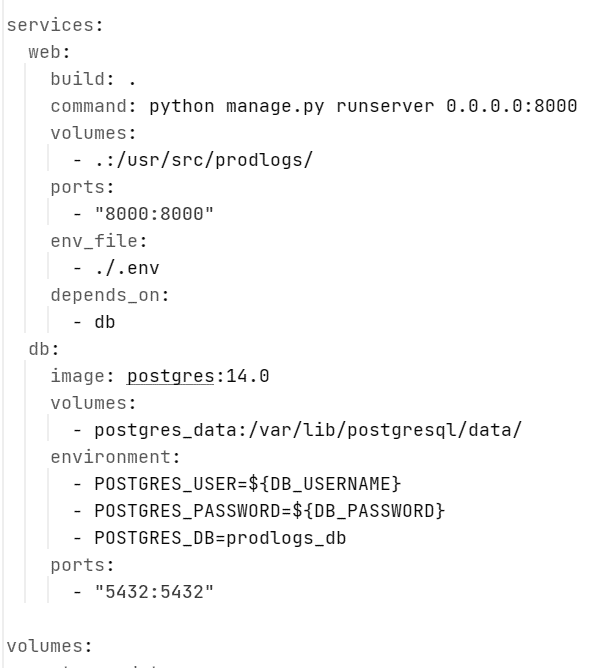
Для создания контейнеров и их запуска мы использовали инструмент docker-compose. В файле docker-compose.yml (рис. 7) мы описали конфигурацию для каждого контейнера, а также их взаимодействие. Контейнер с базой данных использовал образ PostgreSQL, а контейнер с веб-приложением - образ, созданный из Dockerfile.

Рис. 7 – файл docker-compose.yml

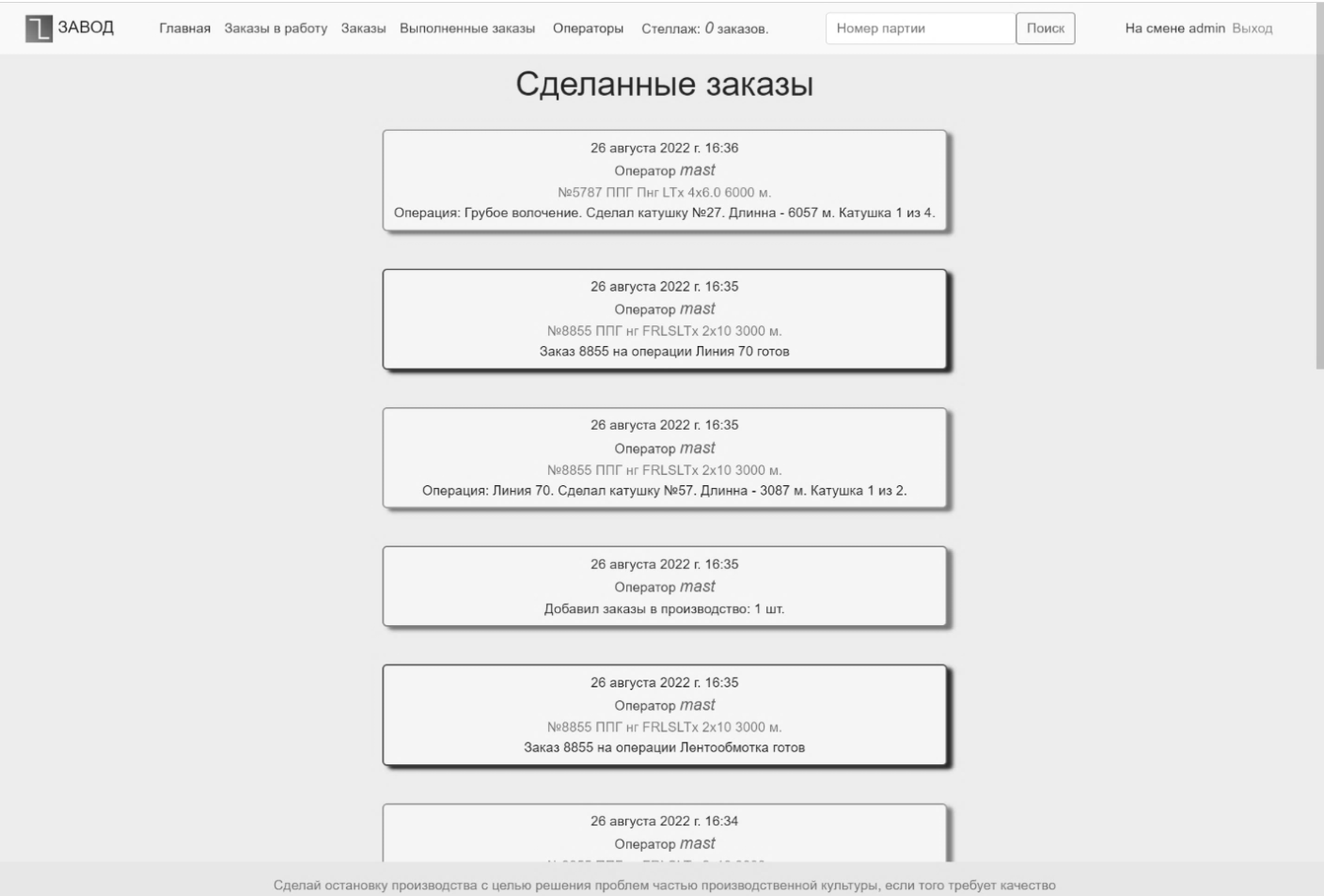
В проекте тестирование кода осуществлялось с помощью библиотеки unittest [6], которая входит в состав стандартной библиотеки Python.

Для каждой модели были написаны соответствующие тесты, которые проверяли правильность работы методов модели, а также их взаимодействие с базой данных. Такой подход позволяет обнаруживать и устранять ошибки еще до того, как они могут повлиять на работу приложения в целом.

Проверка корректной работы интерфейса приложения. Для этого была использована библиотека Selenium, которая позволяет автоматизировать тестирование веб-интерфейса. Selenium позволяет записывать и воспроизводить различные действия пользователя, такие как клики по кнопкам, заполнение форм и т.д.

Итоговое приложение представляет собой полноценную систему управления заказами на производстве, со следующими возможностями:

* Разделенный доступ для пользователей в зависимости от их роли в системе;
* Создание новых заказов, просмотр списка всех заказов, а также возможность просмотра отдельных заказов;
* Добавление заказов в производство и просмотр списка заказов, находящихся в производстве;
* Расширенным поиском по всем заказам в системе;
* Загрузка и выгрузка данных в формате Excel.

Рис. 8 – Главная страница

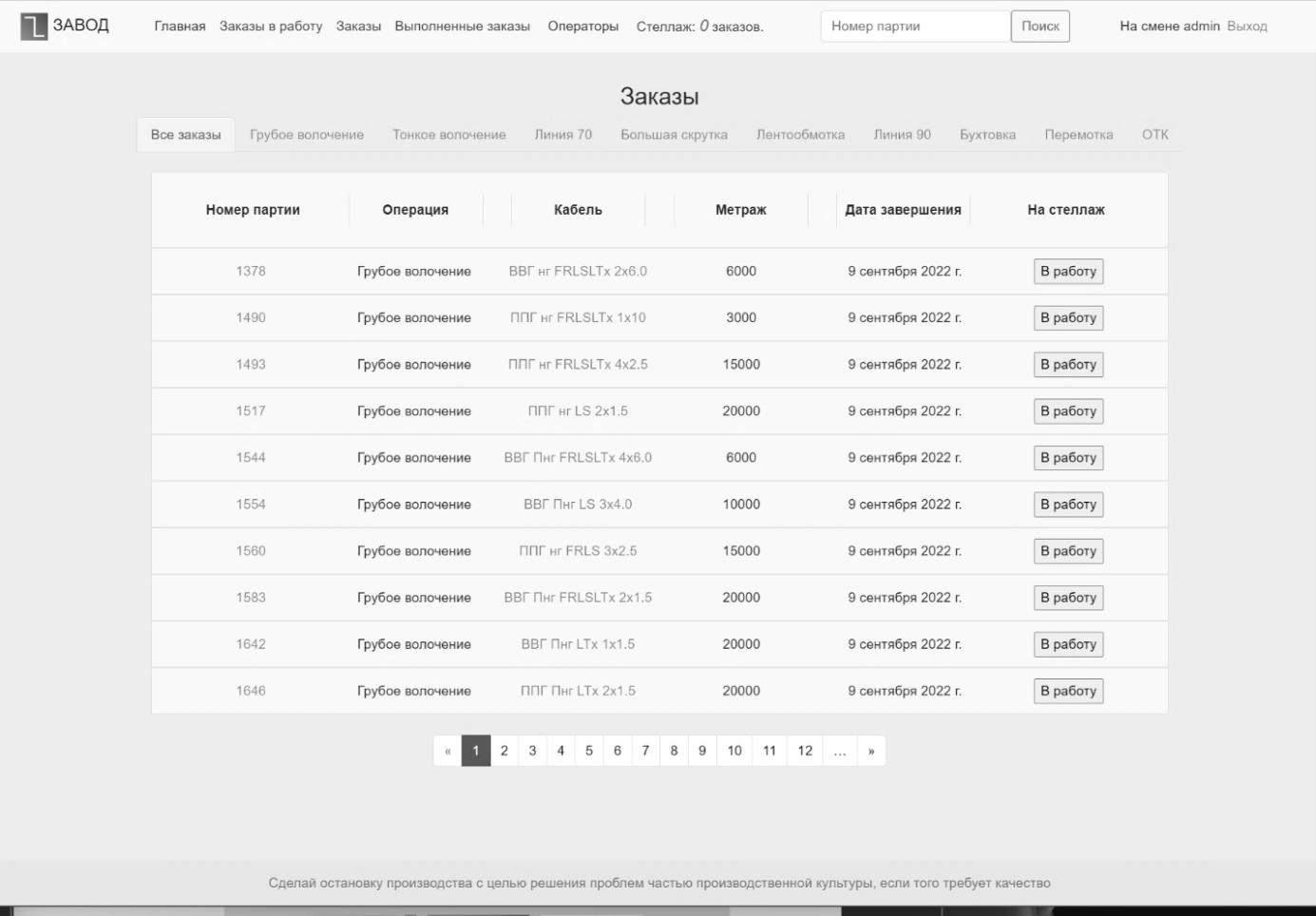
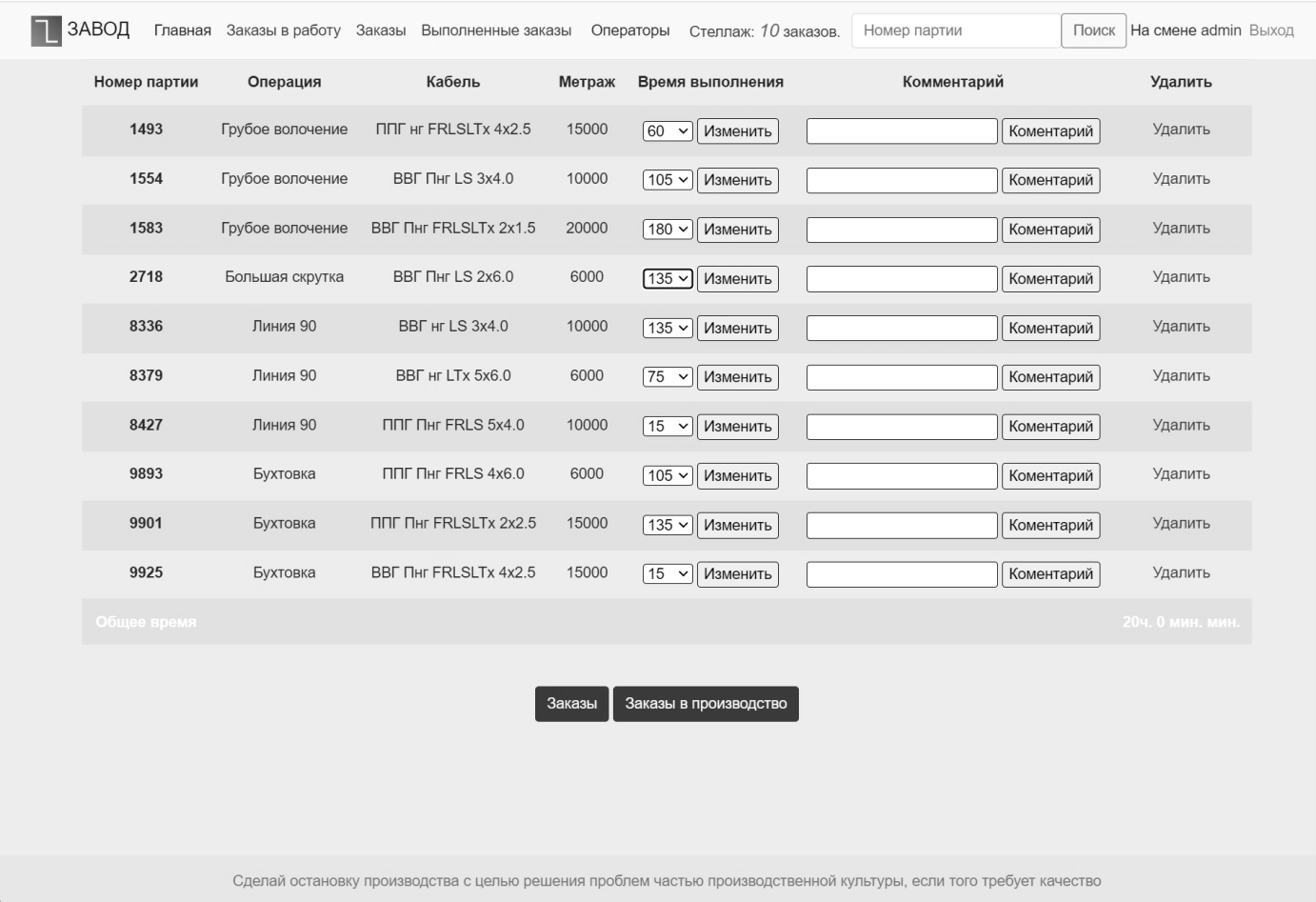


Рис. 9 – Страница со списком заказов



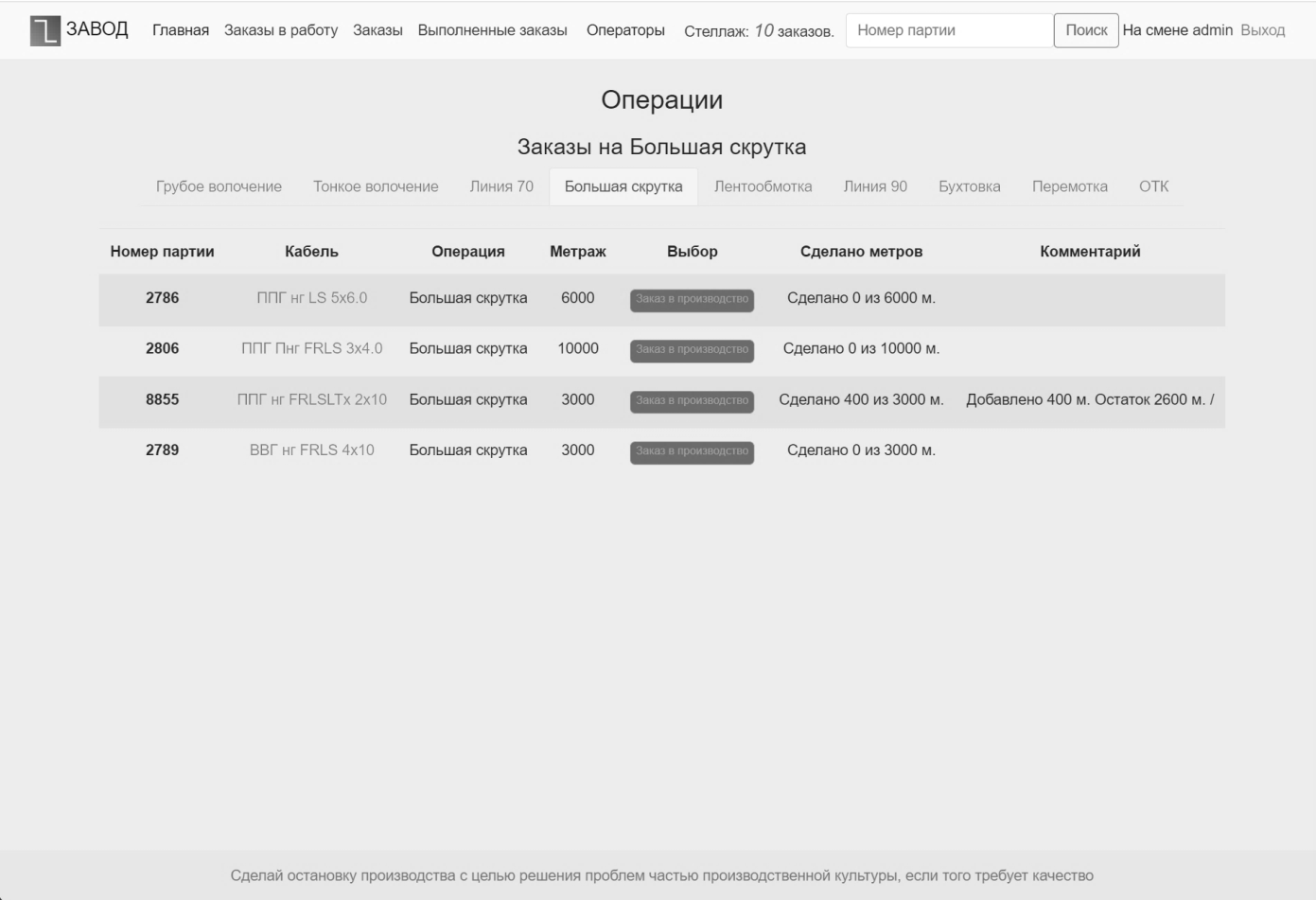
Рис. 10 – Страница с заказами отмеченных для отправки в производство.

Рис. 11 – Страница с заказами отправленных в производство на определённую операцию.

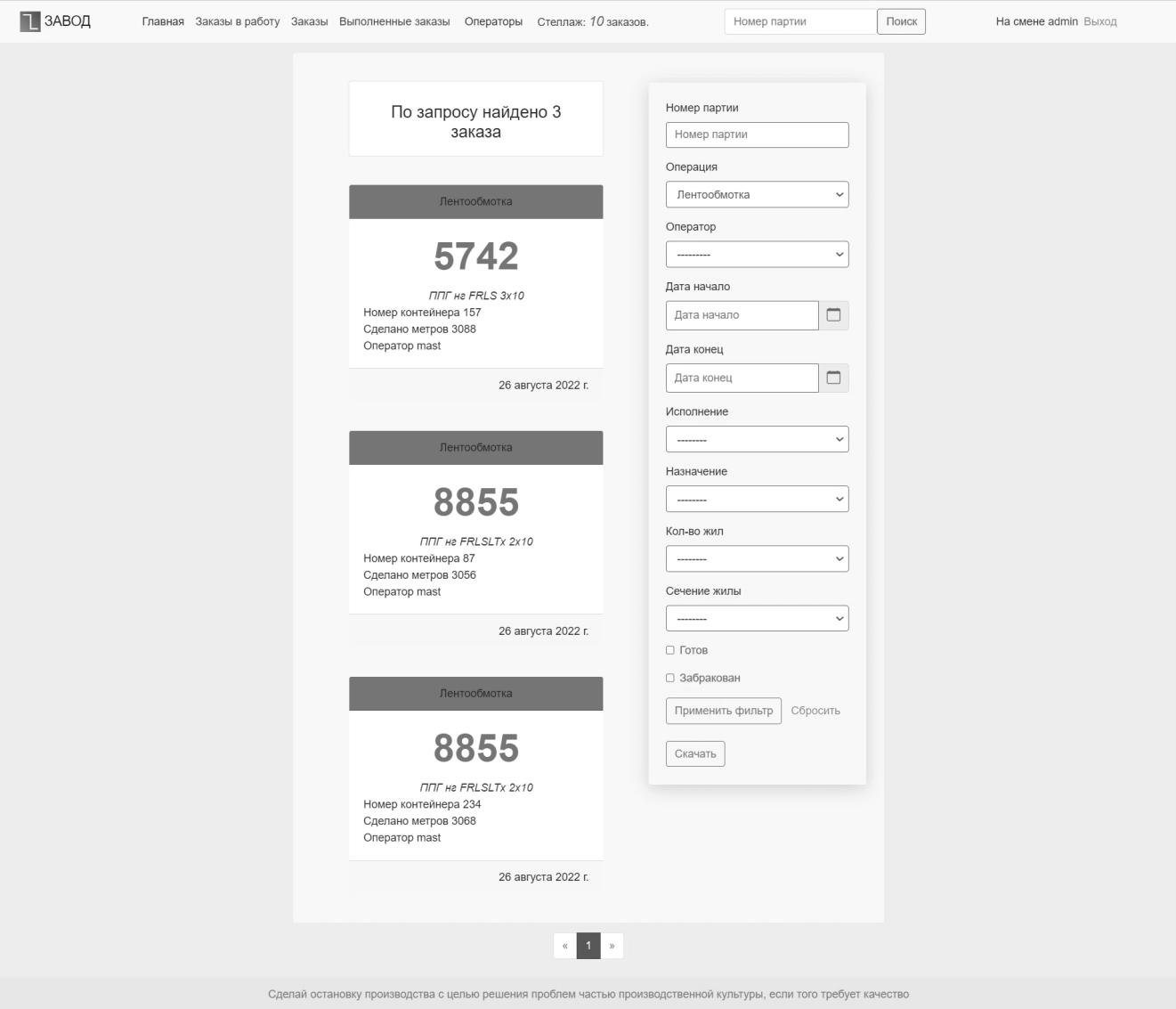


Рис. 12 – Страница расширенного поиска

**Литература**

1. Мартин Роберт. Чистая архитектура. Искусство разработки программного обеспечения.: Издательский дом «Питер», 2022.
2. Дронов В.А. Django 3.0. Практика создания веб-сайтов на Python.: БХВ-Петербург, 2021.
3. Моргунов Е. П. PostgreSQL. Основы языка SQL: учеб. пособие; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018.
4. Сейерс Э. Х., Милл А. Docker на практике.: ДМК Пресс, 2020.
5. Джон Дакет. HTML и CSS. Разработка и дизайн веб-сайтов.: Эксмо, 2011.
6. Абдрахманов М.И. Python unittest, 1-е издание. devpractice.ru, 2017.
7. Unmesh Gundecha. Selenium Testing Tools Cookbook, Eng.: Packt Publishing, 2015.