УДК 004.584

# ИСПОЛЬЗОВАНИЕ WEB-ПРИЛОЖЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ПРОЦЕССА УЧЕТА СЕРВИСНОГО ОБСЛУЖИВАНИЯ ГРУЗОВЫХ ВЕСОВ

Михайлик Д.А., студент гр. ИТб-221, IV курс Асанов С.А., старший преподаватель кафедры ИиАПС
Кузбасский государственный технический университет имени Т. Ф. Горбачева
г. Кемерово

Грузовые весы на дорогах используются для контроля массы и габаритов транспортных средств, чтобы предотвратить перегруз, сохранить дорожное полотно от разрушения и обеспечить безопасность движения. Они автоматически фиксируют общую массу, нагрузку на оси, скорость и номера машин, отправляя данные для оформления штрафов при нарушениях.

В настоящее время грузовые весы устанавливаются во всех регионах Российской Федерации, что влечет за собой высокий спрос на их производство, но для того, чтобы обеспечить правильное функционирование весового оборудование, надо проводить профессиональное обслуживание 2-4 раза в год.

Все компании, занимающиеся обслуживанием весового оборудования, ведут учёт, но не везде учёт данного процесса полностью автоматизирован.

Для автоматизации данного процесса разрабатывается система учета сервисного обслуживания весового оборудования. Система имеет название «УСО» - учет сервисного обслуживания, и будет включать в себя следующие механизмы:

- Просмотр транзакций деталей на производстве;
- Просмотр транзакций деталей на складе;
- Запрос складским работником деталей у производства;
- Оповещение мастером о завершении ремонта оборудования;
- Оповещение мастером о требуемых деталей для ремонта;
- Оповещение складским работником о поступлении деталей на склад;
- Оповещение об отправки деталей на склад с производства;
- Возможность запроса клиентом ремонта оборудования;
- Назначение исполнителей на ремонт весового оборудования;

В данной статье разберем, как можно внедрить в систему web-приложение для реализации учета сервисного обслуживания.

Web-приложение от предприятия не требует никакой определенный платформы, или установки, лишь доступ в интернет. Написано оно на языке Python с использованием Django фреймворка. Также в систему будет входить база данных PostgreSQL, к которой будет обращаться приложение, созданная в СУБД PgAdmin, и хранящая все необходимые данные.

Предполагается, что программа, отвечающая за работу web-приложения, будет работать на сервере непрерывно, чтобы обрабатывать запросы от пользователей. Программа будет обращаться к БД по требованию менеджеров, управляющих ей, для того чтобы вывести нужные данные, или отредактировать их. Диаграмма деятельности процесса назначения на ремонт исполнителя представлена на рисунке 1.

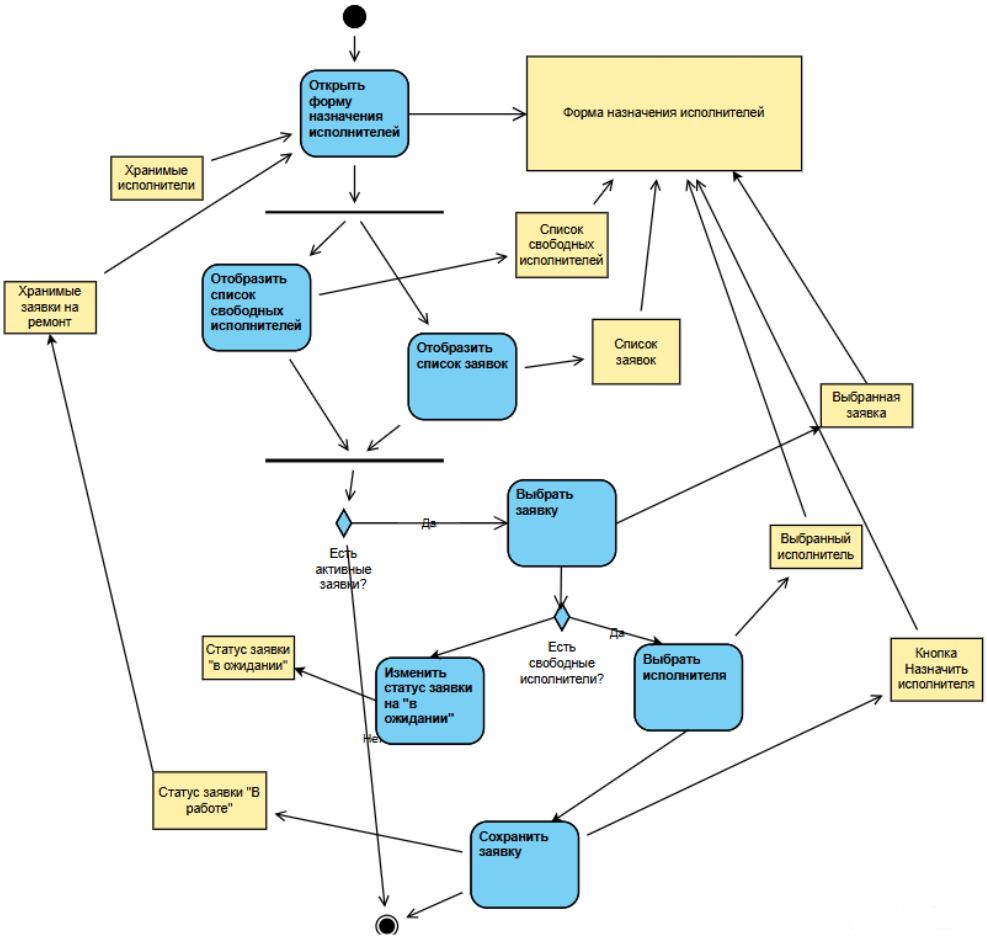


Рисунок 1 – Диаграмма деятельности процесса назначения на ремонт исполнителя.

Для того, чтобы отслеживать издержки поставок деталей на склад с производства, было введено решение просмотра транзакций, чтобы можно было создать оповещение о сомнительных транзакциях. Диаграмма деятельности для выполнения просмотра транзакций представлена на рисунке 2.

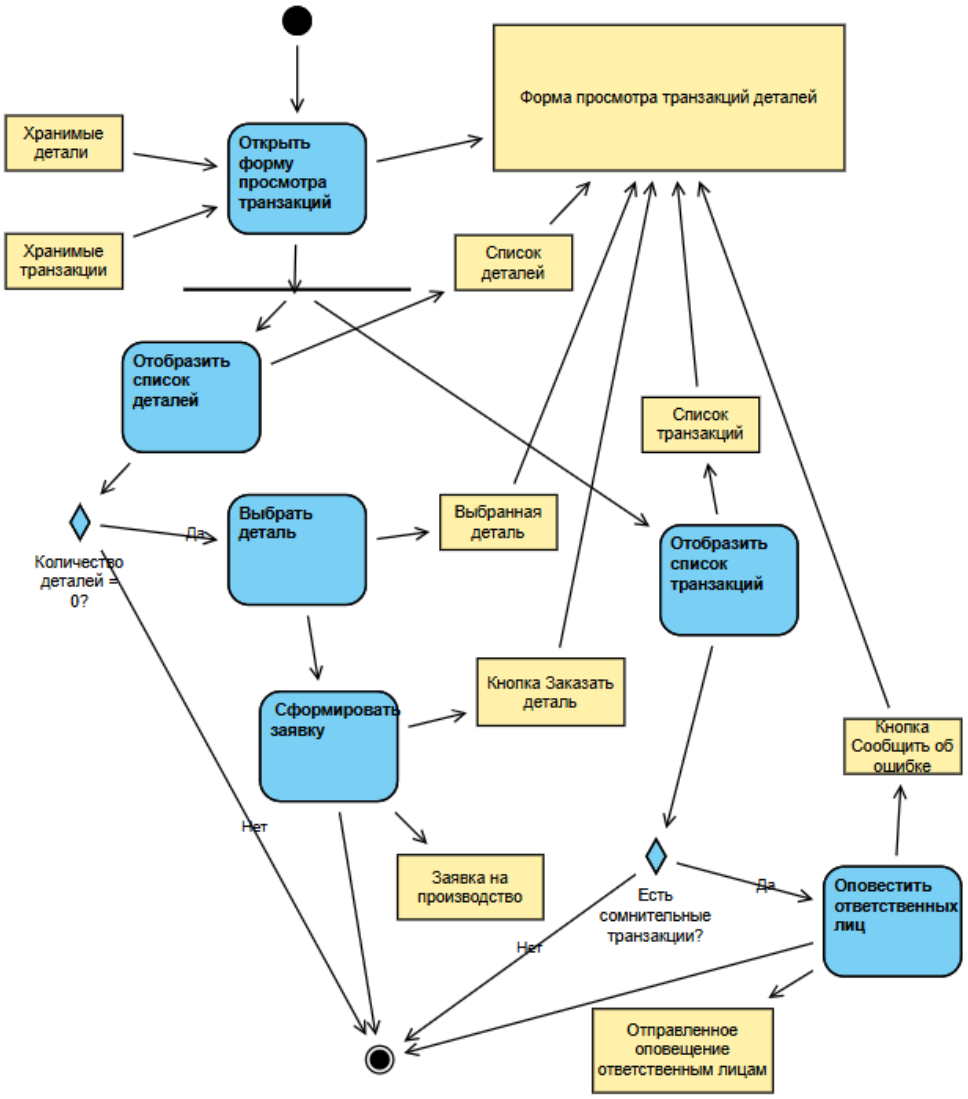


Рисунок 2 – Диаграмма деятельности процесса просмотра транзакций деталей на складе

Для создания web-приложения используем технологию Django и язык Python.

Данный фреймворк предоставляет встроенную ORM (Object-Relational Mapping), что позволяет описывать структур базы данных в виде Python-классов и минимизировать написание «сырых» SQL-запросов. Взаимодействие с PostgreSQL осуществляется через миграции, которые автоматически синхронизируют модели данных с таблицами в СУБД.

Архитектура приложения «УСО» строится на классическом паттерне MVC, который в Django реализован как MVT (Model-View-Template). Модели описывают сущности предметной области: «Оборудование», «Деталь», «Складская операция», «Заявка на ремонт», «Исполнитель». Представления (views) обрабатвыают HTTP-запросы от пользователей, выполняют бизнес-логику (проверка статуса деталей, назначения исполнителя) и возвращают ответ. Шаблоны (templates) генерируют HTML-страницы для взаимодействия с оператором через браузер.

Особое внимание при проектировании было уделено разграничению прав доступа. В системе предусмотрены четыре роли:

– Клиент – создает запрос на ремонт оборудования.

– Мастер – получает оповещения о новых заявках, запрашивает отсутствующие детали у складского работника, сообщает о завершении ремонта.
– Складской работник – обрабатывает запросы деталей, фиксирует поступление на склад, инициирует отправку.
– Администратор – назначает исполнителей на ремонт, просматривает транзакции и анализирует подозрительные операции;

Для реализации оповещений в реальном времени (мастер – склад, склад – производство) в web-приложении используются механизмы Django Channels и WebSockets. Это позволяет отправлять уведомления пользователям без необходимости перезагрузки страницы. Например, когда мастер оповещает о требуемых деталях, у складского работник мгновенно появляется всплывающее уведомление и обновляется список активных заявок.

Модель данных для учета транзакций деталей продемонстрирована на рисунке 3.

```
class Transaction(models.Model):
    TRANSACTION_TYPES = [
        ('IN', 'Поступление с производства'),
        ('OUT', 'Отправка мастеру'),
        ('WRITE_OFF', 'Списание'),
    ]
    part_name = models.CharField(max_length=200, verbose_name="Наименование детали")
    quantity = models.IntegerField(verbose_name="Количество")
    transaction_type = models.CharField(max_length=20, choices=TRANSACTION_TYPES)
    timestamp = models.DateTimeField(auto_now_add=True)
    initiated_by = models.CharField(max_length=100)  # кто выполнил операцию
    is_suspicious = models.BooleanField(default=False)  # флаг для оповещения

    def __str__(self):
        return f"{self.part_name} - {self.quantity} - {self.timestamp}"
```

Рисунок 3 – модель данных для учета транзакций деталей.

Данная модель используется представлением transaction_list_view для отображения всех перемещений деталей. При загрузке страницу выполняется запрос к БД, и если обнаруживаются транзакции с признаком is_suspicious=True (например, резкое списание большого количества деталей), система формирует оповещение. Это соответствует диаграмме деятельности, приведенной на рисунке 2.

Функционал назначения исполнителей на ремонт реализован через отдельное представление assign_executor_view. Менеджер видит список поступивших заявок, выбирает свободного мастера и назначает его. После этого мастер получает уведомление на своей рабочей панели. Для автоматического назначения (при равномерной загрузке мастеров) в Django-приложении может быть реализован дополнительный сервисный слой с простым алгоритмом Round-Robin.

Тестирование web-приложения проводилось в среде разработки с использованием встроенного сервера Django, а затем с использованием Gunicorn в качестве WSGI-сервера и Nginx для проксирования запросов. База данных PostgreSQL работала на отдельном хосте. Результаты тестирования

показали, что при одновременной работе до 50 пользователей (мастеров, складских работников, клиентов) среднее время отклика интерфейса не превышает 1,2 секунды, а уведомления доходят за 0,5-0,8 секунды.

Благодаря внедрению web-приложения «УСО» в компаниях, обслуживающих грузовые весы, достигается полная прозрачность учёта сервисного обслуживания. Исключаются задержки в оповещении о потребности в деталях, автоматизируется контроль издержек при поставках со склада и производства, сокращается время на назначение исполнителей. Использование Django и PostgreSQL обеспечивает масштабируемость: при росте числа обслуживаемых весов достаточно добавить вычислительные ресурсы серверу без переписывания логики приложения.

**Список литературы:**

1. Документация Django [Электронный ресурс]. – URL: https://docs.djangoproject.com/en/6.0/ (дата обращения: 29.03.2026)

2. Документация PosgtreSQL [Электронный ресурс] – URL: https://www.postgresql.org/docs/ (дата обращения: 29.03.2026)
3. Моделирование на языке UML в среде Visual Paradigm [Электронный ресурс] – URL: http://sp.cs.msu.ru/ooap/exer2017.html (дата обращения: 29.03.2026)