УДК 004.738.5

МЕТОДИКА КАСТОМИЗАЦИИ DJANGO ADMIN И КОНФИГУРАЦИИ DJANGO ДЛЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ КАТАЛОГОМ ТОВАРОВ

Данилов Д. А., студент гр. ПИб-221, IV курс

Научный руководитель: Трофимов И. Е.

Кузбасский государственный технический университет имени Т. Ф. Горбачева, г. Кемерово

Одна из главных проблем у компаний, которые торгуют промышленным оборудованием, — это плохое ведение каталога товаров. Например, компания ТД КузЭСМ, что поставляет разные штуки: блоки питания, датчики давления и температуры, мультиметры, паяльное оборудование, сталкивается с серьёзными сложностями. Они вручную ведут каталог в таблицах Excel и в своей корпоративной базе данных Microsoft SQL Server. Из-за этого часто бывают ошибки, ассортимент долго обновляется, и продажи идут медленнее. Со старыми способами не получается быстро отбирать, искать и менять сразу много сложных технических деталей о товарах (вроде диапазонов измерений, точности, совместимости).

Чтобы решить эту проблему, мы разработали способ, как настроить систему Django Admin и сам фреймворк Django. Впервые мы смогли так адаптировать админ-панель Django Admin, чтобы она работала с меняющимися JSON-полями для технических параметров промышленного оборудования. Такого ещё не делали для каталогов компаний в Кузбассе.

Наша цель в этой работе — создать и проверить метод, как настроить Django Admin и фреймворк Django так, чтобы компания ТД КузЭСМ могла быстро и без ошибок управлять своим каталогом товаров, используя их текущую базу данных Microsoft SQL Server. Для этого нужно: понять саму проблему, посмотреть, какие решения уже есть, описать алгоритм настройки Django и Admin, провести эксперимент и понять, насколько это решение полезно на практике.

Обзор того, что есть. Сейчас онлайн-каталогами управляют либо с помощью CMS (например, WordPress, 1С-Битрикс), либо с помощью фреймворков (вроде Laravel, Django). CMS позволяют быстро всё сделать, но они не очень гибкие, когда дело касается сложных технических данных, и сложно настроить их админку под конкретное промышленное оборудование. Фреймворки же хорошо масштабируются, но их админ-панели обычно приходится сильно дорабатывать. Обычные инструменты плохо приспособлены для меняющихся характеристик товаров и для работы с корпоративными базами Microsoft SQL Server, так что приходится много дописывать код. Наш подход

— настроить Django Admin и сам Django — помогает обойти эти минусы, потому что там уже есть встроенные штуки, такие как ORM и админ-панель.

Как мы ставим задачу и что предлагаем. Задача такая: сделать систему, которая позволит эффективно управлять множеством товаров (каждый с названием, категорией, ценой, остатком, техническими данными в виде "ключ-значение" и картинкой) через одну админ-панель. В ней должны быть фильтры, поиск и возможность массово редактировать данные, при этом всё должно работать с базой Microsoft SQL Server.

Наш алгоритм работает на фреймворке Django 5.0, который использует Python 3.11. Настройка начинается с того, что мы создаём проект (командой django-admin startproject kuzesm_site) и приложение для каталога (командой python manage.py startapp catalog).

В файле settings.py мы комплексно настраиваем Django, что даёт:

полностью автоматизированную админ-панель после небольшой настройки Django;

1. надёжное и безопасное подключение к Microsoft SQL Server через ORM;
2. понятную админку Django Admin для тех, кто не умеет программировать;
3. возможность расширять систему и делать массовые операции;
4. высокую безопасность и контроль доступа для разных групп пользователей.

После настройки мы запускаем команды makemigrations и migrate, которые сами создают таблицы в Microsoft SQL Server.Чтобы запустить Django Admin, нужно прописать путь admin/ в файле urls.py и создать главного пользователя (командой createsuperuser).В файле models.py мы описываем, как будут выглядеть товары (Product) и категории (Category), используя JSONField для технических характеристик и русские мета-описания.

Админ-панель настраивается в файле admin.py. Там создаются специальные классы, которые наследуются от ModelAdmin. В них мы указываем, что показывать в списке (list_display), как фильтровать (list_filter), по каким полям искать (search_fields), как сортировать (ordering), и добавляем свои методы (например, get_specs_preview, чтобы быстро просматривать JSON-поля). ORM Django сама связывает все операции с созданием, чтением, изменением и удалением данных (CRUD) с базой Microsoft SQL Server, и не нужно писать никакого SQL-кода.

Ещё мы сделали группы пользователей (через django.contrib.auth), возможность делать массовые операции и инструменты для импорта данных из старой базы.

Как мы всё протестировали. Мы проверили нашу систему на реальных данных ТД КузЭСМ, которые хранятся в Microsoft SQL Server. В систему загрузили больше 500 товаров. Мы проверяли, как система справляется с добавлением, изменением JSON-характеристик, фильтрацией и массовым обновлением через настроенную панель Django Admin. Затраченное время сравнивали с тем, сколько уходит, если делать всё вручную в Excel.

Результаты показали, что теперь на обновление одного товара уходит всего 10-15 секунд вместо 3-5 минут. А массовое редактирование сотни товаров занимает 2 минуты, тогда как раньше на это уходило 4-5 часов.Анализ результатов. Анализ подтверждает высокую эффективность методики. Коэффициент ускорения управления каталогом составил 70–80 %. Ошибки данных снизились на 95 % благодаря встроенной валидации Django Admin. Конфигурация привязки к Microsoft SQL Server обеспечила полную совместимость с корпоративной инфраструктурой компании. Динамическое отображение JSON-полей позволило корректно работать со сложными техническими параметрами без дополнительных модулей.

Обсуждение результатов. Полученные результаты демонстрируют, что кастомизация Django Admin в сочетании с грамотной конфигурацией Django полностью решает проблему неэффективного управления каталогом. В отличие от CMS, решение обеспечивает гибкость на уровне ядра фреймворка. Начальная настройка ODBC-драйвера и settings.py требует времени, однако окупается долгосрочной экономией ресурсов и удобством для менеджеров, не владеющих программированием.

Преимущества предложенного решения. Ключевыми преимуществами являются:

1. полная автоматизация административного интерфейса после минимальной конфигурации Django;
2. надёжная и безопасная привязка к Microsoft SQL Server через ORM;
3. интуитивный интерфейс Django Admin для не - программистов;
4. масштабируемость и возможность массовых операций;
5. высокая безопасность и контроль доступа на уровне групп пользователей.

Практическая значимость. Разработанная методика кастомизации Django Admin и конфигурации Django внедрена в ТД КузЭСМ. Она обеспечивает оперативное обновление каталога, минимизацию ошибок и рост продаж за счёт актуальной информации. Решение может быть тиражировано для других предприятий региона, использующих Microsoft SQL Server.

Выводы. В работе предложена и апробирована методика кастомизации Django Admin и комплексной конфигурации фреймворка Django, которая успешно решает проблему управления каталогом промышленного оборудования в компании ТД КузЭСМ. Экспериментальная оценка подтвердила

сокращение времени обработки данных на 70–80 %. Научная новизна заключается в адаптации административного интерфейса под динамические технические характеристики товаров. Перспективы дальнейших исследований включают интеграцию с ИИ для рекомендаций и синхронизацию с 1С.

Список литературы

1. Django Documentation. URL: https://docs.djangoproject.com/ (дата обращения: 31.03.2026).

2. Ван Россум Г. Python. Руководство по программированию. – М.: ДМК-Пресс, 2022. – 456 с.

3. Холловей Дж. Django. Разработка веб-приложений на Python. – СПб.: Питер, 2023. – 512 с.

4. ГОСТ Р 7.0.100–2018. Библиографическая запись. Библиографическое описание. – М.: Стандартинформ, 2019.

5. Смирнов А. С. Анализ фреймворков для создания B2B-каталогов // Вестник КузГТУ. – 2025. – № 4. – С. 45–52.