УДК 004.738.5+004.43

Преимущества использования шаблонизаторов Django для бизнеса: баланс между эффективностью и стоимостью разработки

Забродин Андрей Владимирович — кандидат ист. наук, доцент кафедры «Информационные и вычислительные системы». Научные интересы: информационные системы, аналитика данных, парадигмы программирования, облачные технологии. E-mail: zabrodin@pgups.ru

Михеев Андрей Дмитриевич — студент 2-го курса бакалавриата направления 09.03.01 «Информатика и вычислительная техника». Научные интересы: автоматизация и клиентская лояльность, веб-разработка. E-mail: miheev.31.andrey@gmail.com

Граур Максим Александрович — студент 2-го курса бакалавриата направления 09.03.01 «Информатика и вычислительная техника». Научные интересы: автоматизация и клиентская лояльность, веб-разработка. E-mail: maksimgraur22@mail.ru

Петербургский государственный университет путей сообщения Императора Александра I, Россия, 190031, Санкт-Петербург, Московский пр., 9

Для цитирования: Забродин А. В., Михеев А. Д., Граур М. А. Преимущества использования шаблонизаторов Django для бизнеса: баланс между эффективностью и стоимостью разработки // Интеллектуальные технологии на транспорте. 2026. № 2 (46). С. 36–45. DOI: 10.20295/2413-2527-2026-246-36-45

Аннотация. *Рассматривается проблема выбора технологического стека для веб-проектов малого и среднего бизнеса.* ***Цель:*** *на фоне распространения архитектур с разделением фронтенда и бэкенда проанализировать целесообразность использования встроенного шаблонизатора Django как более простого решения для проектов с ограниченным бюджетом, небольшой командой и умеренными требованиями к интерактивности страниц сайта.* ***Результаты:*** *показано, что серверный рендеринг в сочетании с запросами одностраничного приложения позволяет сократить архитектурную сложность, снизить затраты на разработку и поддержку приложения, сохранить преимущества SEO и использовать встроенные механизмы безопасности Django. Сделан вывод о том, что для ряда прикладных бизнес-задач Django-шаблоны могут выступать как рациональный архитектурный выбор.* ***Практическая значимость:*** *результаты исследования могут быть использованы при проектировании систем массового сбора данных и взаимодействия с ними, построении распределенных поисковых роботов и создании платформ анализа информации на основе многоагентных систем.* ***Обсуждение:*** *представленные наблюдения отражают практическую структуру функционирования распределенного поискового робота. В отличие от классических поисковиков предложенный агент-ориентированный подход позволяет перераспределять нагрузку, динамически изменять стратегии обхода при обнаружении блокировок.*

Ключевые слова: *Django, шаблонизатор, сравнительный анализ архитектур, веб-разработка, наблюдаемость, AJAX, малый бизнес*

2.3.5 — *математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей (технические науки)*

Введение

В последние годы в веб-разработке стала стандартом практика разделения клиентской и серверной частей приложения [1]. Фронтенд обычно создают с помощью JavaScript-фреймворков и библиотек React, Vue или Angular. Бэкенд реализуют на отдельных платформах и обеспечивают взаимодействие с клиентом через API. Этот подход оптимален для крупных нагруженных систем, где нужны сложная бизнес-логика и независимое масштабирование компонентов. Однако в проектах малого и среднего бизнеса такую архитектуру веб-приложения используют гораздо чаще, чем требуется [2]. Для типовых задач — корпоративных сайтов, каталогов, личных кабинетов, сервисов записи, небольших интернет-магазинов — важнее другие критерии: скорость разработки, ограниченный бюджет, небольшой размер команды, хорошая поисковая индексация и предсказуемость сопровождения. Разделение фронтенда и бэкенда в таких случаях может провоцировать избыточную сложность: нужно поддерживать две отдельные кодовые базы, требуется проектировать и поддерживать API, необходимо синхронизировать изменения между обеими частями приложения.

Более простой вариант — использовать встроенный шаблонизатор Django [3] для создания веб-приложения с единой серверной логикой. При этом, находясь на одной и той же странице, можно сохранить интерактивность интерфейса при помощи динамического изменения контента посредством AJAX-запросов. Такой подход позволяет балансировать между SPA (Single Page Application — одностраничное веб-приложение) и SSR (Server-Side Rendering — серверный рендеринг, позволяющий командам предоставлять динамический контент), сохраняя простоту и скорость разработки.

Цель статьи — показать, когда использование Django-шаблонов становится обоснованной альтернативой разделенной архитектуре. Основное внимание уделено проектам с ограниченными ресурсами, где критически важны простота реализации, быстрая скорость запуска и удобство поддержки проекта. В рамках исследования проведен сравнительный анализ двух архитектур [4] по ключевым критериям: совокупная стоимость владения (total cost of ownership, TCO), скорость поисковой индексации, уровень встроенной безопасности и затраты на сопровождение.

Новизна исследования заключается в переосмыслении роли встроенного шаблонизатора Django. Вместо того чтобы считать его устаревшим инструментом, он рассматривается как рациональное архитектурное решение для определенного класса бизнес-ориентированных веб-проектов. Ключевое отличие данного подхода заключается в том, что с «технологической современности» фокус смещается на соответствие реальным потребностям бизнеса, включая стоимость сопровождения, требования SEO, уровень безопасности и простоту дальнейшего развития проекта.

Методология исследования

Django традиционно воспринимается как серверный фреймворк для быстрой разработки [5, 6], а архитектурные исследования акцентируют внимание на масштабируемых системах с API [7]. В проектах с ограниченными ресурсами остаются недостаточно изученными такие аспекты, как совокупная стоимость владения [8], эффективность поисковой индексации [9], организационная сложность сопровождения [10]. Статья восполняет этот пробел, анализируя условия, при которых Django-шаблоны становятся рациональным выбором для создания веб-приложения.

Сравнительный анализ архитектурных подходов

В рамках проведенного сравнительного анализа архитектурных паттернов, результаты которого представлены в табл. 1, отражены ключевые различия между подходами на основе Django-шаблонов и архитектурой с разделением фронтенда и бэкенда на базе React + API. Особое внимание уделено техническим и организационным аспектам реализации.

Критерии сравнения были подобраны на основе анализа профессиональной литературы [5, 9, 10] и практического опыта работы с проектами разного

Таблица 1

Компаративный анализ двух архитектурных паттернов

Параметр сравнения	Архитектура А: Django-шаблоны (SSR)	Архитектура Б: React + API (SPA)
Тип рендеринга	Серверный (SSR)	Клиентский (CSR), возможен гибридный (SSR через Next.js)
Количество кодовых баз	1 (Python)	2 (JavaScript + Python/Java/Go)
Способ передачи данных	Непосредственно в контексте шаблона	Через REST/GraphQL API
Требования к специалистам	1 Middle Python-разработчик	1 Middle React + 1 Middle Backend
Инструментарий отладки	Django Debug Toolbar, логи на сервере	React DevTools + Postman + логи API
SEO из коробки	Да (полный HTML)	Нет (требуется SSR-надстройка)

масштаба. При формировании системы оценки были выделены ключевые факторы: экономическая эффективность (анализ стоимости разработки, внедрения и сопровождения), простота поддержки, поисковая оптимизация, безопасность, скорость вывода на рынок, масштабируемость, кадровые требования.

Гипотеза исследования заключается в том, что для веб-проектов малого и среднего масштаба использование Django-шаблонов может обеспечить более выгодное соотношение цены и качества по сравнению с разделенной архитектурой фронтенда и бэкенда.

Для проверки гипотезы проведем формальное сравнение с использованием метрики TCO за 24 месяца:

$$TCO = DevCost + (TeamSize \times AvgSalary \times 24) + InfraCost + SupportCost,$$

где DevCost — стоимость начальной разработки (человеко-месяцы × ставка);

TeamSize — число разработчиков в команде поддержки;

AvgSalary — средняя зарплата [11];

InfraCost — стоимость хостинга, CDN, мониторинга;

SupportCost — стоимость исправления ошибок и доработок.

Расчет носит ориентировочный характер и используется для сравнительной оценки рассматриваемых подходов разработки, все параметры занесены в табл. 2. Значения ставок и инфраструктурных расходов приведены как усредненные показатели и могут изменяться в зависимости от региона, квалификации специалистов и масштаба проекта.

Проведенное сравнение показывает, что для рассматриваемого класса проектов общая стоимость владения архитектурой React + API может значительно превышать аналогичный показатель для решения на Django-шаблонах. Полученный результат позволяет предварительно подтвердить выдвинутую гипотезу, однако его не следует распространять на иные типы веб-приложений без учета масштаба проекта и состава команды.

Таблица 2

Исходные данные на примере интернет-магазина

Параметр	Django + шаблоны	React + API
Команда разработки, чел.	1 Middle	1 Middle React + 1 Middle Backend
Ставка, тыс. руб./мес.	200	200
Время разработки, мес.	3	5 (из-за интеграции API)
DevCost, тыс. руб.	600	2000
TeamSize, чел.	0,5 (частичная)	1,5 (1 Front + 0,5 Back)
InfraCost, тыс. руб./мес.	10	25 (дополнительные сборки, CDN)
TCO за 24 мес., тыс. руб.	600 + (0,5 · 200 · 24) + (10 · 24) = **3240**	2000 + (1,5 · 200 · 24) + (25 · 24) = **9800**

Влияние на SEO: количественная оценка

Проведено экспериментальное сравнение индексации двух идентичных сайтов — тестовых интернет-магазинов на 500 товарных страниц, реализованных на Django-шаблонах и React. Оба сайта имели одинаковую структуру каталога, набор товарных страниц и текстовое наполнение. Различие заключалось в способе формирования HTML-страницы. Сайты были одновременно поданы в Google Search Console и «Яндекс Вебмастер». Результаты были сформированы через 14 дней после отправки сайтов на индексацию (табл. 3).

Способ рендеринга напрямую влияет на индексацию сайта поисковыми системами. При серверном рендеринге поисковый робот получает готовый HTML-документ, в то время как при клиентском рендеринге значительная часть содержимого формируется после выполнения JavaScript. Общая логика различия между серверным и клиентским рендерингом представлена на рис. 1.

Таблица 3

Сравнение индексации сайтов

Показатель	Django (SSR)	React (CSR)
Проиндексировано страниц, количество	487 (97,4 %)	112 (22,4 %)
Среднее время до индексации, дней	1,2	5,8
Позиции в топ-50 по 10 ключевым запросам	7/10	2/10
Обнаружение новых товаров роботом, ч	< 1	> 48

Рис. 1. Схема сравнения серверного рендеринга на Django и клиентского рендеринга на React

Полученные данные позволяют предположить, что сайт на Django-шаблонах может раньше начать получать активный трафик за счет более быстрой индексации страниц.

Безопасность и наблюдаемость архитектурных решений

Для оценки особенностей сопровождения и сложности конфигурации был проведен обзор данных CVE (Common Vulnerabilities and Exposures — список известных уязвимостей и дефектов информационной безопасности) за 2023–2025 годы для Django, а также типичных компонентов разделенной SPA-архитектуры: React, React DOM и серверных API-фреймворков. Анализ носит сравнительно-иллюстративный характер (табл. 4).

В разделенной архитектуре проекта уязвимости могут возникать на разных уровнях системы: в настройках CORS, механизмах авторизации API, обработке CSRF-защиты, зависимостях клиентской части или серверного фреймворка. В Django часть стандартных механизмов защиты берется «из коробки», что снижает число разделенных точек контроля для небольшой команды разработки.

Различия в наблюдаемости были также проверены на примере преднамеренно внесенной ошибки, связанной с замедленным запросом к базе данных (рис. 2).

Таблица 4

Обзор и сравнение данных CVE

Компонент	Количество CVE (критических и высоких)	Исправлено за < 7 дней, %	Требует обновления двух частей
Django	12	100	Нет
React + React DOM	8	75	Да
Express / Spring	27	60	Да
Связка React + Backend	35	0	Да

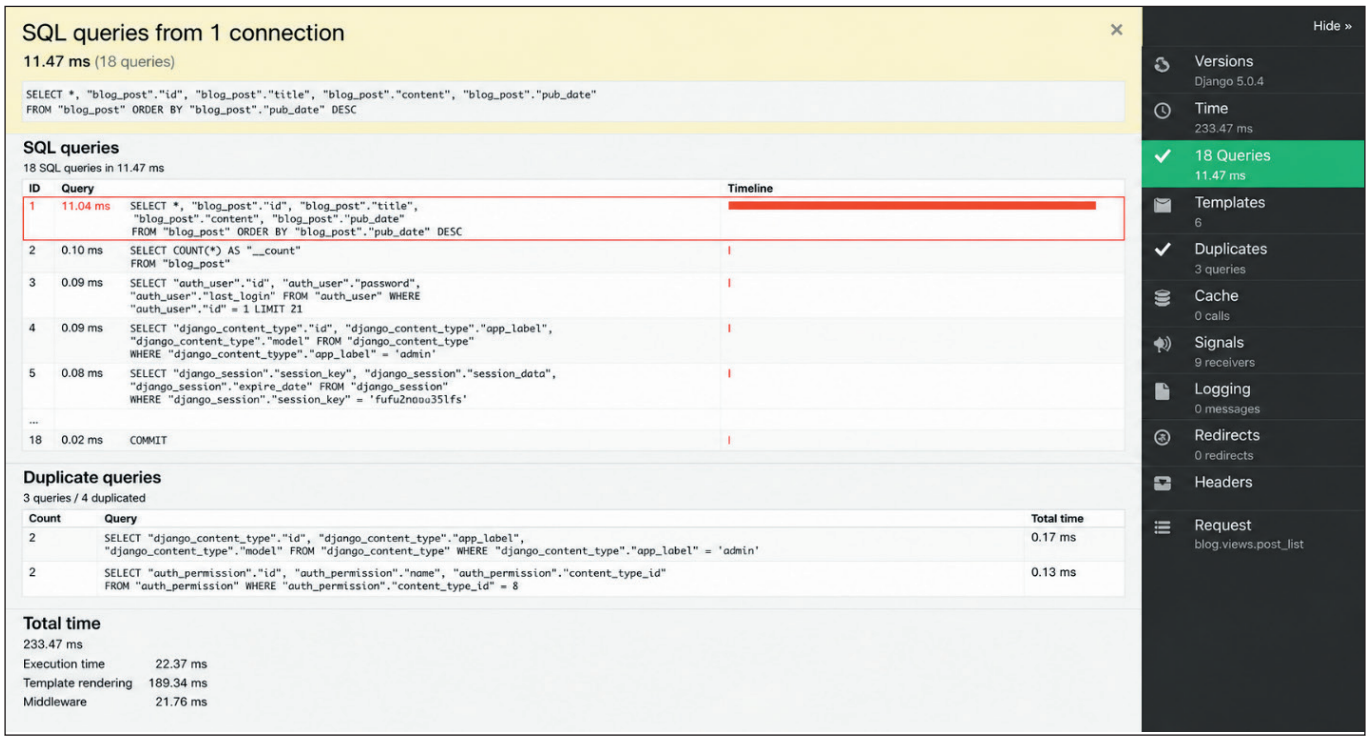

Рис. 2. Скриншот работы Django Debug Toolbar

В варианте Django + Debug Toolbar разработчик получает возможность непосредственно увидеть дублирующиеся SQL-запросы. В варианте React + API наблюдаемость аналогичной проблемы требует сопоставления данных клиентской части, сетевых запросов и серверных логов.

Реализация SPA-подобного взаимодействия средствами Django-шаблонов

Проведена экспериментальная реализация интернет-магазина с использованием Django 5.0 и встроенных шаблонов [12], асинхронных запросов, а также эндпоинтов (endpoint), возвращающих страницу с передачей только блока контента. Результаты нагрузочного тестирования показаны в табл. 5 и на рис. 3–5. Общие рекомендации по оптимизации производительности шаблонов и снижению нагрузки на сервер подробно разбираются в [5].

Ключевые результаты эксперимента позволяют сделать следующие выводы: плавность переходов между страницами в представленном примере оказалась сопоставимой с вариантом на React, при этом SEO-показатели сохраняются на высоком уровне благодаря выполнению загрузки страницы средствами серверного рендеринга. Кроме того, единая кодовая база снижает риск дублирования маршрутизации,

Таблица 5

Результаты нагрузочного тестирования

Метрика	Django (полная перезагрузка)	Django (AJAX/SPA-подход)	React (CSR, без Next.js)
First Contentful Paint (FCP), с	0,90	0,90	1,80 (ожидание JS)
Time to Interactive (TTI), с	1,20	1,10	2,40
Скорость перехода между страницами, с	0,80 (перезагрузка)	0,12 (подмена блока)	0,10
SEO-готовность (Lighthouse SEO)	100	100	78

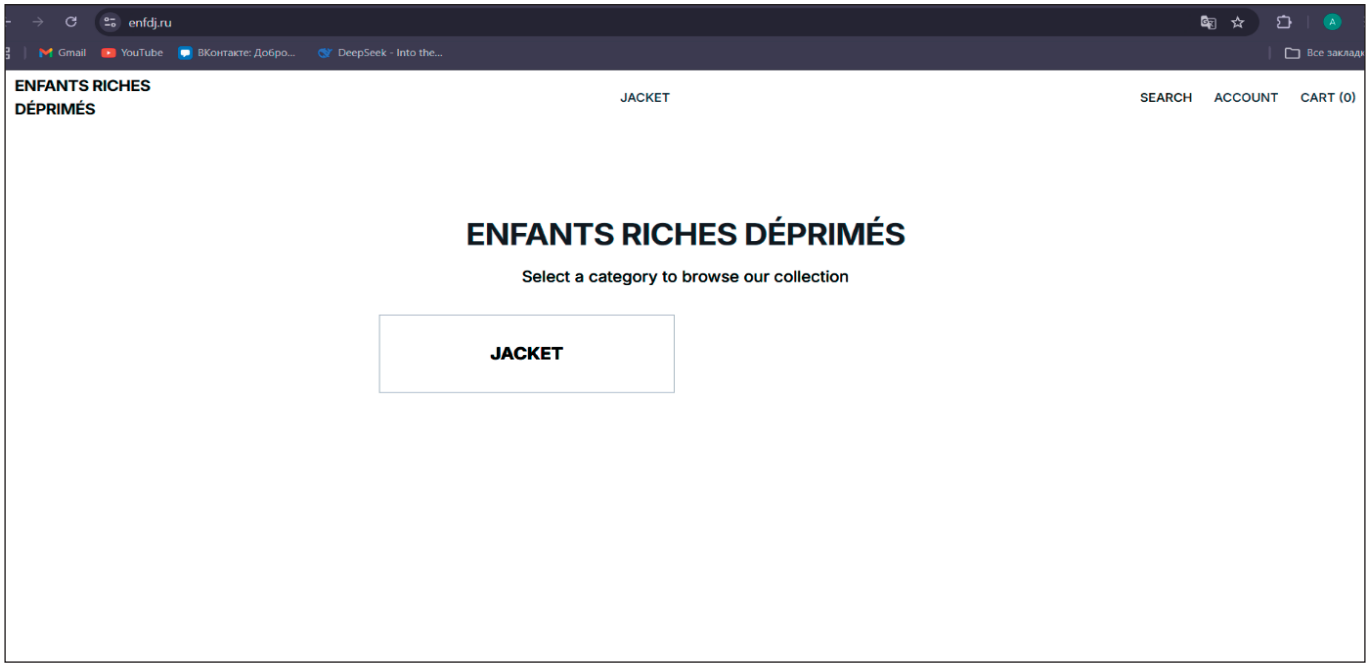

Рис. 3. Главная страница сайта — серверный рендеринг

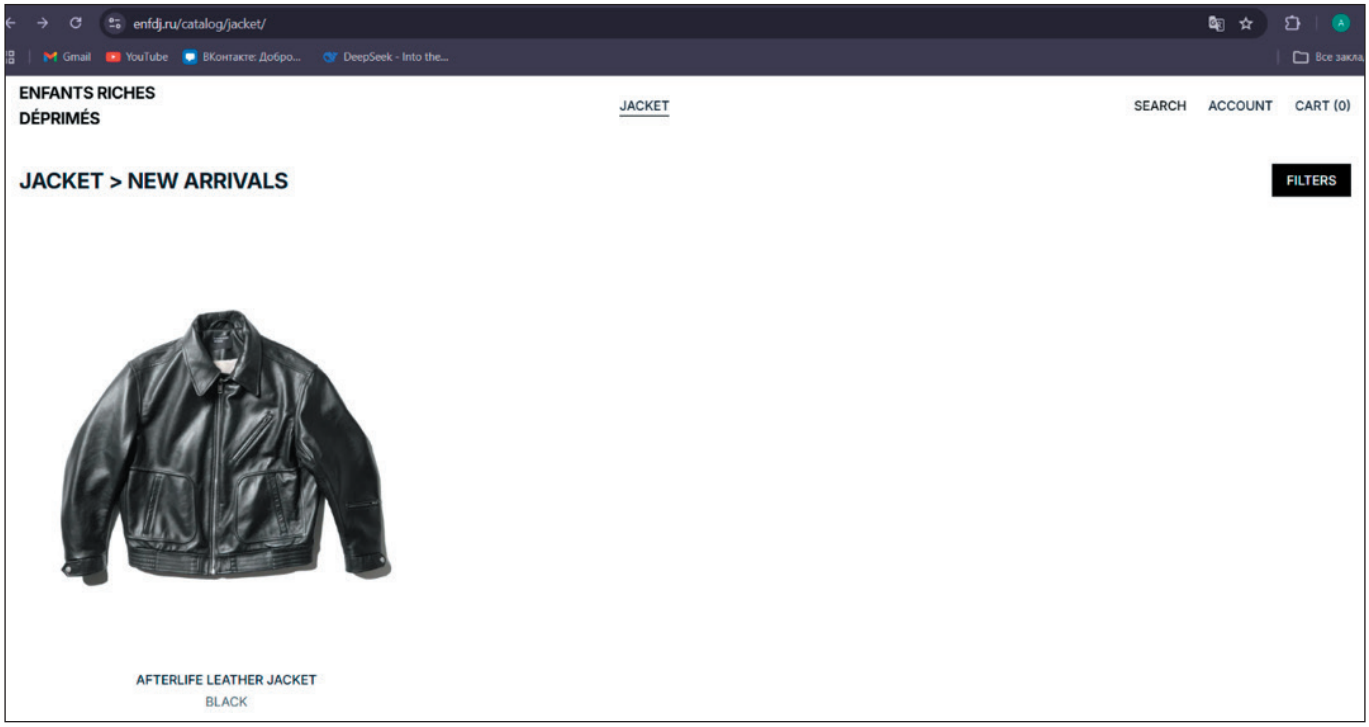

Рис. 4. Каталог — подгрузка через AJAX

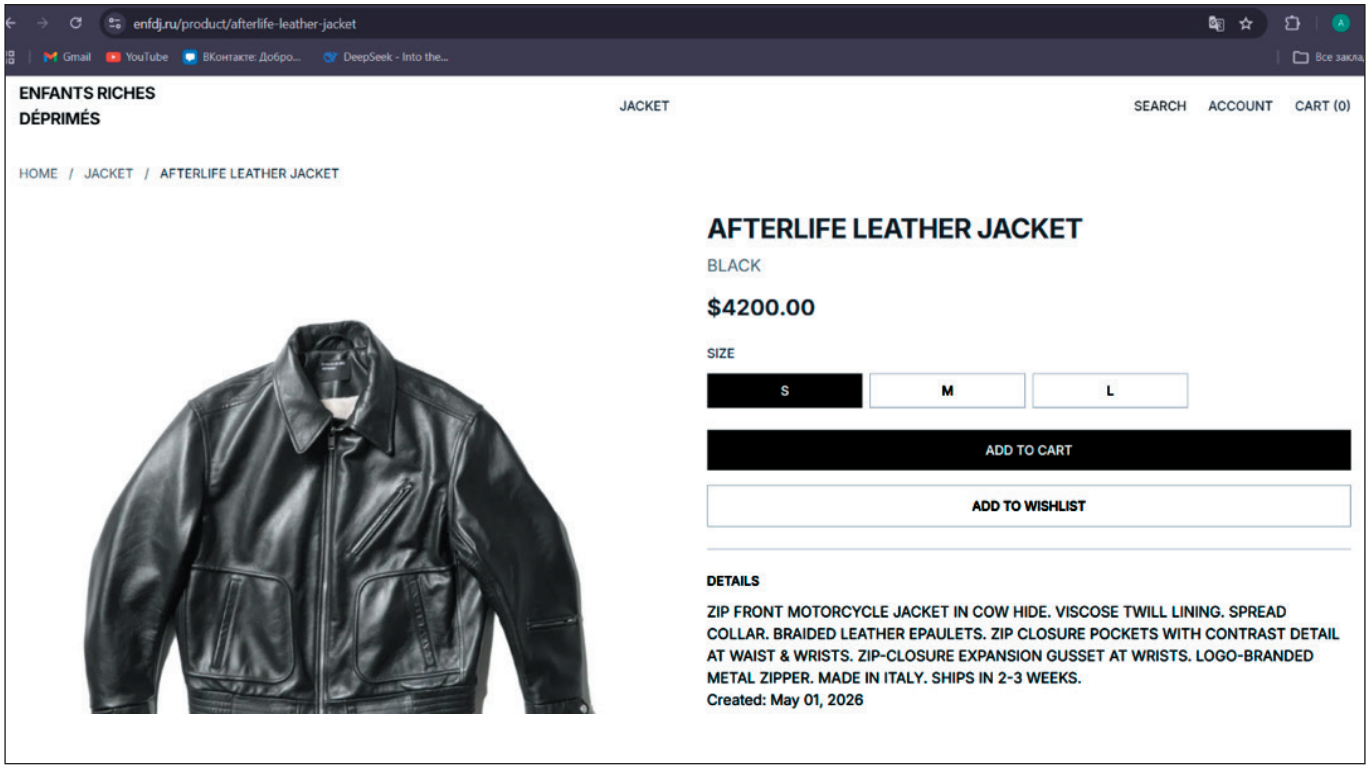

Рис. 5. Карточка товара — динамический контент с изменением URL

валидации и логики приложения, а реализация SPA-подобной навигации не потребовала выделения отдельного фронтенд-разработчика. На основе проведенного сравнительного анализа сформулированы рекомендации по выбору архитектуры (табл. 6).

Таблица 6

Рекомендации по выбору архитектуры зависимости от требований проекта

Условия проекта	Предпочтительный подход
Небольшая команда разработки и ограниченный бюджет	Django-шаблоны
Корпоративный сайт, каталог, сервис записи, MVP	Django-шаблоны
Высокие требования к SEO и скорости запуска	Django-шаблоны или гибридный SSR-подход
Сложная клиентская логика и большое число интерактивных состояний	React/Vue/Angular + API
Независимое масштабирование фронтенда и бэкенда	Разделенная архитектура

Анализ количественных и качественных характеристик — от совокупной стоимости владения и скорости индексации до безопасности и наблюдаемости — показывает, что выбор архитектуры не сводится к противопоставлению «современный vs устаревший».

Представленные в табл. 6 сценарии наглядно демонстрируют: для проектов с ограниченными ресурсами, типовыми бизнес-задачами и умеренными требованиями к интерактивности подход на основе Django-шаблонов обеспечивает лучший баланс между эффективностью разработки и долгосрочными затратами, тогда как разделенная архитектура оправдана лишь при наличии четких требований к сложной клиентской логике или необходимости независимого масштабирования компонентов.

Заключение

Проведенное исследование позволяет рассматривать встроенный шаблонизатор Django как практически значимый инструмент для веб-проектов малого и среднего бизнеса. При ограниченном бюджете, небольшой команде разработки и умеренных требованиях к интерактивности серверный рендеринг снижает архитектурную сложность, уменьшает объем работ и упрощает последующую поддержку приложения. Сопоставление с разделенной архитектурой React + API выявляет преимущества Django-шаблонов в тех случаях, когда проект не требует сложной клиентской логики и независимого масштабирования фронтенда. Наиболее существенными оказываются единая кодовая база, предсказуемая поисковая индексация, меньшее число точек взаимодействия уровней приложения и более простая экранизация ошибок. Вместе с тем Django-шаблоны имеют собственные ограничения. Они менее удобны при разработке интерфейсов с высокой степенью интерактивности, офлайн-режимом работы и большим количеством независимых frontend-компонентов. Следовательно, выбор между Django-шаблонами и SPA-архитектурой должен определяться не технологической новизной решения, а задачами проекта [13], ресурсами команды и требованиями к дальнейшему сопровождению проекта.

СПИСОК ИСТОЧНИКОВ

1. Фаулер М. Шаблоны корпоративных приложений = Patterns of Enterprise Application Architecture / пер. с англ. Испр. изд. СПб.: Диалектика, 2020. 544 с.

2. Nesher G. SPA Are Dead, Long Live SPA // Semaphore Blog. 23.08.2023. URL: http://semaphore.io/blog/single-page-applications (дата обращения: 14.05.2026).

3. Дронов В. А. Django 4. Практика создания веб-сайтов на Python. СПб.: БХВ-Петербург, 2023. 800 с.

4. Наумов Р. В. Программирование Питон. Выбор веб-фреймворка // Достижения науки и образования. 2016. № 12 (13). С. 25–26.

5. Django Documentation. Version 5.0. URL: http://docs.djangoproject.com/en/5.0 (дата обращения: 16.05.2026).

6. Лутц М. Изучаем Python: авторитетный курс объектно-ориентированного программирования. Пятое издание = Learning Python. Fifth Edition: в 2 т. / пер. с англ. и ред. Ю. Н. Артеменко. СПб.: Диалектика, 2020.

7. Jadhav S. B., Gholve S. S. Django vs. FastAPI: A Comparative Study for High-Performance Web Applications // International Journal of Advance and Applied Research. 2025. Vol. 6, no. 23. Pp. 47–51. DOI: 10.5281/zenodo.15119179

8. Software Engineering Cost Estimation using COCOMO II Model / K. Htay [et al.] // International Journal of Trend in Scientific Research and Development. 2019. Vol. 3, iss. 5. Pp. 2326–2329. DOI: 10.31142/ijtsrd28019

9. Основы поисковой оптимизации сайтов, использующих JavaScript // Google for Developers. URL: http://developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics (дата обращения: 17.05.2026).

10. Achille A. Server-Side Rendering vs. Client-Side Rendering: A Guide for Web Development // BairesDev Blog. 13.05.2026. URL: http://www.bairesdev.com/blog/server-side-client-rendering-web-development (дата обращения: 15.05.2026).

11. Зарплаты IT-специалистов в первой половине 2025: +2 %, рост замедлился // Хабр. 14.08.2025. URL: http://habr.com/ru/specials/936618 (дата обращения: 14.05.2026).

12. The Django Template Language: Django Documentation. URL: http://docs.djangoproject.com/en/5.1/ref/templates/language (дата обращения: 17.05.2026).

13. Verbina E. Which is the Best Python Web Framework: Django, Flask, or FastAPI? // PyCharm Blog. 18.02.2026. URL: http://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi (дата обращения: 13.05.2026).

Дата поступления: 19.05.2026
Решение о публикации: 22.05.2026

Advantages of Using Django Template Engines for Business: The Balance Between Efficiency and Cost of Development

Andrey V. Zabrodin — PhD in History, Associate Professor of the “Information and Computing Systems” Department. Research interests: information systems, data analytics, programming paradigms, cloud technologies. E-mail: zabrodin@pgups.ru

Andrey D. Mikheev — 2nd year Bachelor’s Degree Student in 09.03.01 Informatics and Computer Technology. Research interests: automation and customer loyalty, web development. E-mail: miheev.31.andrey@gmail.com

Maksim A. Graur — 2nd year Bachelor’s Degree Student in 09.03.01 Informatics and Computer Technology. Research interests: automation and customer loyalty, web development. E-mail: maksimgraur22@mail.ru

Emperor Alexander I St. Petersburg State Transport University, 9 Moskovsky ave., Saint Petersburg, 190031, Russia

For citation: Zabrodin A. V., Mikheev A. D. Graur M. A. Advantages of Using Django Template Engines for Business: The Balance Between Efficiency and Cost of Development, *Intellectual Technologies on Transport*, 2026, no. 2 (46), pp. 36–45. DOI: 10.20295/2413-2527-2026-246-36-45 (In Russian).

Abstract. *The problem of choosing a technology stack for small and medium-sized business web projects is considered.* ***Purpose:*** *against the background of the proliferation of architectures with separation of frontend and backend, to analyze the feasibility of using the built-in Django template engine as a simpler solution for projects with a limited budget, a small team and moderate requirements for the interactivity of site pages.* ***Results:*** *it is shown that server-side rendering combined with single-page application queries can reduce architectural complexity, reduce application development and support costs, preserve SEO benefits, and use Django’s built-in security mechanisms. It is concluded that Django templates can act as a rational architectural choice for a*

number of applied business tasks. ***Practical significance:*** *the research results can be used in the design of mass data collection systems and interaction with them, the construction of distributed search robots and the creation of information analysis platforms based on multi-agent systems.* ***Discussion:*** *the presented observations reflect the practical structure of the functioning of a distributed search robot. Unlike classical search engines, the proposed agent-based approach allows you to redistribute the load and dynamically change the bypass strategies when blocking is detected.*

Keywords: *Django, template engine, comparative architecture analysis, web development, observability, AJAX, small business*

REFERENCES

1. Fowler M. Shablony korporativnykh prilozheniy [Patterns of Enterprise Application Architecture]. Saint Petersburg, Dialektika Publishing House, 2020, 544 p. (In Russian)

2. Nesher G. SPA Are Dead, Long Live SPA, *Semaphore Blog*. Published online at August 23, 2023. Available at: http://semaphore.io/blog/single-page-applications (accessed: May 14, 2026).

3. Dronov V. A. Django 4. Praktika sozdaniya veb-saytov na Python [Creating Websites with Python]. Saint Petersburg, BHV-Peterburg Publishing House, 2023, 800 p. (In Russian)

4. Naumov R. V. Programmirovanie Piton. Vybor veb-freymvorka [Python Programming. Choosing a Web Framework], *Dostizheniya nauki i obrazovaniya*, 2016, no. 12 (13), pp. 25–26. (In Russian)

5. Django Documentation. Version 5.0. Available at: http://docs.djangoproject.com/en/5.0 (accessed: May 16, 2026).

6. Lutz M. Izuchaem Python: avtoritetnyy kurs obektno-orientirovannogo programmirovaniya. Pyatoe izdanie [Learning Python. Fifth Edition]. In 2 volumes. Saint Petersburg, Dialektika Publishing House, 2020. (In Russian)

7. Jadhav S. B., Gholve S. S. Django vs. FastAPI: A Comparative Study for High-Performance Web Applications, *International Journal of Advance and Applied Research*, 2025, vol. 6, no. 23, pp. 47–51. DOI: 10.5281/zenodo.15119179

8. Htay K., et al. Software Engineering Cost Estimation using COCOMO II Model, *International Journal of Trend in Scientific Research and Development*, 2019, vol. 3, iss. 5, pp. 2326–2329. DOI: 10.31142/ijtsrd28019

9. Understand the JavaScript SEO Basics, *Google for Developers*. Available at: http://developers.google.com/search/docs/crawling-indexing/javascript/javascript-seo-basics (accessed: May 17, 2026).

10. Achille A. Server-Side Rendering vs. Client-Side Rendering: A Guide for Web Development, *BairesDev Blog*. Published online at May 13, 2026. Available at: http://www.bairesdev.com/blog/server-side-client-rendering-web-development (accessed: May 15, 2026).

11. Zarplaty IT-spetsialistov v pervoy polovine 2025: +2%, rost zamedlilsya [Salaries of IT Specialists in the First Half of 2025: +2%, Growth Slowed Down], *Khabr [Habr]*. Publishe online at August 14, 2025. Available at: http://habr.com/ru/specials/936618 (accessed: May 14, 2026). (In Russian)

12. The Django Template Language: Django Documentation. Available at: http://docs.djangoproject.com/en/5.1/ref/templates/language (accessed: May 17, 2026).

13. Verbina E. Which is the Best Python Web Framework: Django, Flask, or FastAPI? *PyCharm Blog*. Published online at February 18, 2026. Available at: http://blog.jetbrains.com/pycharm/2025/02/django-flask-fastapi (accessed: May 13, 2026).

Received: May 19, 2026
Accepted: May 22, 2026