УДК 004.42

СРАВНИТЕЛЬНЫЙ АНАЛИЗ FASTAPI, DJANGO И FLASK: ПРОИЗВОДИТЕЛЬНОСТЬ, УДОБСТВО РАЗРАБОТКИ И ВЫБОР ОПТИМАЛЬНОГО ФРЕЙМВОРКА ДЛЯ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ

Худницкий А.В., Шауро Е.А., студенты гр. 353502

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Жвакина А.В. – канд. техн. наук, доцент

Аннотация. В статье представлен сравнительный анализ трёх популярных Python-фреймворков — FastAPI, Django и Flask — с акцентом на их производительность, удобство разработки и применимость в современных веб-проектах. Рассматриваются ключевые особенности: асинхронная обработка запросов в FastAPI, «полнота» и структурированность Django, а также минимализм и гибкость Flask. На основе оценки сильных и слабых сторон каждого инструмента даются рекомендации по выбору оптимального решения для разных сценариев использования.

Ключевые слова. FastAPI, Django, Flask, Сравнительный анализ, Производительность, Удобство разработки, Асинхронная обработка, Минимализм, Веб-приложения, Высоконагруженные API, Бизнес-логика, Оптимизация выбора, Современные технологии.

Язык программирования Python широко используется в современных технологиях во множестве отраслей начиная с web-разработки и заканчивая машинным обучением. Для упрощения создания сайтов и приложений со сложной структурой и поддержкой баз данных в 2005 году компанией Lawrence Journal-World был создан full-stack фреймворк Django [1]. Уже на своих ранних версиях фреймворк поддерживал ORM, предлагал автоматически генерируемую административную страницу, маршрутизацию посредством URL, что помогало программистам сфокусироваться на логике разработки веб-приложения без необходимости контролировать низкоуровневые детали, такие как обработка HTTP-запросов, управление сессиями или взаимодействие с базой данных напрямую. За 20 лет развития вышло пять поколений фреймворка, новый релиз выходит в среднем каждые 8 месяцев. Проект развивается как open-source, его код можно найти на github.com/django/django. Сегодня Django остается самым популярным фреймворком для web-разработки на Python с самой большой клиентской базой.

Через пять лет после релиза Django австрийским программистом Арминой Ронахером был разработан микрофреймворк Flask – своего рода конструктор, содержащий только базовые возможности для создания проектов. В ходе разработки на Flask программист сам определяет технологии, которые ему необходимы, начиная с типа базы данных и заканчивая паттернами проектирования, то есть фреймворк придерживается простого, но расширяемого ядра. Минимализм и возможность расширения привлекла множество программистов по всему миру. Теперь Flask обладает крупным комьюнити пользователей, которые создают расширения и библиотеки для фреймворка на протяжении уже 15 лет. Слоганом Flask считается фраза “меньше – значит больше”.

Самым молодым, но от этого не менее популярным фреймворком для разработки на Python является асинхронный FastAPI, разработанный Себастьяном Рамиресом. Проект, подобно Django, развивается в формате open-source и доступен по адресу github.com/tiangolo/fastapi. Название для фреймворка подобрано не случайно: во-первых, благодаря библиотекам Starlette и Pydantic, которые отвечают за работу с web и для описания схем данных соответственно, достигается максимальная производительность и высокая скорость обработки запросов; во-вторых, FastAPI интуитивно понятен, что значительно влияет на количество ошибок и время, затраченное на разработку. Фреймворк обладает стремительно развивающимся сообществом, стандарты OpenAPI и асинхронности поддерживаются современными библиотеками и инструментами.

Приведенные выше фреймворки можно сравнить по следующим критериям: производительность, гибкость, количество модулей, размер сообщества, сложность изучения, вакансии.

Производительность.

При данной оценке важно понимать, что само по себе слово “производительность” не содержит в себе ни одной метрики. Иногда улучшения в одной области ведут к ухудшениям в другой, поэтому требуется четко понимать, что исследуется. Производительность любого приложения напрямую зависит от аппаратных ресурсов, например, от процессора, оперативной памяти, скорости интернет-соединения и передачи данных с твердых носителей. Однако, рассматривать будем только те характеристики, которые разнятся от фреймворка к фреймворку, если взять как факт одинаковое аппаратное обеспечение.

Фреймворк Django легко масштабируемый и способен обрабатывать в секунду большое количество запросов. Среди способов оптимизации кода, предлагаемых Django, можно выделить следующие: кэширование, оптимизация запросов к базе данных и ее масштабирование. Однако это тяжеловесный фреймворк, который опирается на множество компонентов и зависимостей, потому его часто относят к категории производительности от средней до высокой. В рабочих проектах совместно с Django используют Celery для фоновых задач.

Рассмотрим производительность Flask. Фреймворк не навязывает никакой структуры или дополнительных технологий разработчику, что и делает его быстрым для небольших приложений и API. По умолчанию он использует Werkzeug (комплексная библиотека веб-приложений WSGI) и Jinja2 (библиотека-стандарт для рендеринга шаблонов), которые эффективны, но в однопоточном режиме могут не справляться с высокой нагрузкой, поскольку обе библиотеки работают синхронно. Если приложение выполняет много I/O-операций (запросы к базе данных, API), это может стать “бутылочным горлышком”.

Компания TechEmpower в феврале 2025 года проводила независимые тесты производительности приложений [2], результат которых показал, что FastAPI по своим производительным характеристикам уступает лишь Starlette, потому как использует его внутри себя. Асинхронный протокол ASGI внутри Starlette позволяет обрабатывать множество запросов одновременно без блокировки ввода-вывода. К тому же, FastAPI не добавляет лишних слоев поверх Starlette и Uvicorn, что минимизирует задержки. Результаты тестирования приведены на рисунке 1.

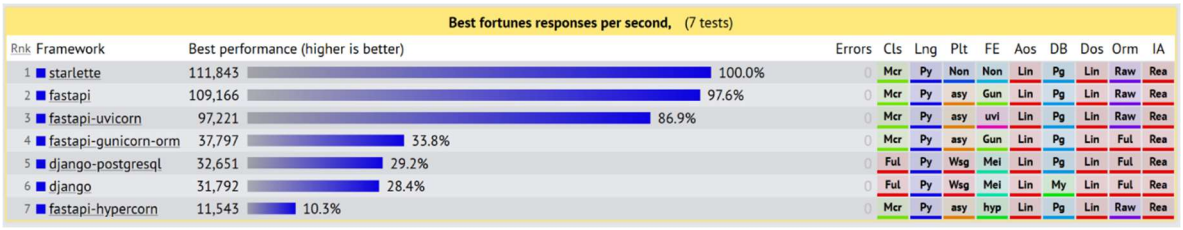

Рисунок 1 – Результаты тестирования производительности приложений

Таким образом, по результатам исследования наиболее высокой производительностью обладает FastAPI ввиду своей асинхронности и разбиения задач без блокирования основного потока. Flask изначально не поддерживает асинхронные операции, однако из-за простоты своей кастомизации демонстрирует стабильную и быструю работу. Фреймворк Django использует протокол WSGI, однако в рабочих проектах часть его задач выполняет Celery и т.п.

Гибкость.

Django – это высокоуровневый веб-фреймворк, который сочетает в себе готовые решения и гибкость для адаптации под различные задачи. Он предоставляет множество встроенных инструментов, но при этом позволяет глубоко кастомизировать почти каждый аспект программы.

Django построен на концепции “приложений” – переиспользуемых компонентов, которые можно комбинировать и настраивать, что свидетельствует о том, что можно написать свои модули либо собрать проект из уже готовых. Одним из самых значительных преимуществ данного фреймворка является встроенная кастомизируемая административная панель. Django работает с любым фронтендом через API и предлагает тысячи библиотек, к примеру, Django Rest Framework.

Однако Django формально реализует паттерн MTV (Model-Template-View), который является адаптацией классического MVC (Model-View-Controller). Попытки реализовать чистый MVC, MVP или MVVM требуют обходных путей. Например, для MVVM с фронтендом на React/Vue придется полностью отказаться от Django Templates и строить взаимодействие через API, что не всегда удобно.

Flask представляет собой микрофреймворк и оставляет программисту максимальную гибкость в выборе дополнительных компонентов. Flask не навязывает жесткую структуру проекта, что позволяет адаптировать его под любые нужды, использовать любые сторонние библиотеки и архитектуру приложения.

Подобно Flask FastAPI также не имеет какой-либо утвержденной архитектуры. FastAPI поддерживает интеграцию с любыми WSGI-приложениями, такими как Dash, Flask, Django, позволяя монтировать их внутри FastAPI. Фреймворк запросто работает с пользовательскими библиотеками, поддерживает JWT-аутентификацию.

Количество модулей.

Django имеет огромное количество пакетов - более 5000 пакетов на PyPI [3]. Популярные сторонние пакеты: Django REST Framework (API), Django Debug Toolbar (отладка), а модули: ORM, Админ-панель, аутентификация. Качество самих пакетов находится на высоком уровне.

Flask имеет около 1800 пакетов на PyPI, однако многие из них устарели или плохо поддерживаются. Популярными расширениями являются: Flask-SQLAlchemy (интеграция с ORM), Flask-Login (аутентификация), Flask-CORS (CORS-заголовки).

FastAPI имеет всего 500 пакетов на PyPI, многие из которых находятся в стадии активной разработки. Главной особенностью является то, что многие модули - адаптации из Starlette и Pydantic. Встроенные модули: валидация данных (Pydantic), асинхронная обработка данных, автодокументация (Swagger, OpenAPI), Dependency Injection.

Размер сообщества.

Django является самый старым фреймворков, поэтому имеет самое крупное сообщество среди 3 фреймворков: много учебников, книг, конференций (DjangoCon). Поддержка версий составляет 3 года, а развитая экосистема (готовые решения для любых задач: аутентификация, админка, ORM) позволяет создавать меньше “велосипедов” – большинство проблем уже решено.

Flask имеет много туториалов для новичков, хотя их количество меньше, чем у Django. Сообщество фреймворка ценит минимализм и кастомизацию.

FastAPI является самым молодым фреймворком среди троих, однако его сообщество очень активное. Сама документация имеет хорошие переводы на 10+ языков. Главная особенность - фокус на modern tech: async/await, OpenAPI, Pydantic - что привлекает молодых разработчиков. Готовых решений достаточно мало, но многие библиотеки Django/Flask адаптируются под FastAPI. Он является самым быстрорастущим фреймворком по данным Jetbrains Developer Survey.

Сложность изучения.

Django предоставляет все “из коробки”: (ORM, админ-панель, аутентификацию). Однако для их использования необходимо понимание архитектуры фреймворка. Тем не менее, Django обладает хорошей документацией и больших количеством туториалов. Он нацелен на разработчиков, которые готовы потратить достаточно времени на его изучение.

В отличие от Django, Flask дает лишь базовые инструменты, т.е. ORM, аутентификацию добавляет сам разработчик. К строгой структуре он не обязывает, что может быть проблемой при нехватке опыта или бонусом для большего контроля над каждым слоем приложения. Flask очень хорош для понимания основ HTTP, REST, работы с запросами, ответами.

FastAPI требует понимания async/await в Python, а также работы с моделями через Pydantic, что требует привычки к аннотациям типов. Стоит отметить автоматическую документацию (Swagger/OpenAPI), что упрощает тестирование и понимание работы API. Для обучения имеется отличная лаконичная документация.

Вакансии.

Для получения информации были изучены данные из следующих источников: rabota.by, belmeta.com, habr.com/vacancies (таблица 1).

Таблица 1 - Количество вакансий для Django, FastAPI, Flask

	rabota.by	belmeta.com	habr.com/vacancies
Django	31	22	13
FastAPI	31	16	10
Flask	17	10	2

В веб-экосистеме Python большая часть вакансий приходится именно на Django. Далее немного отстает FastAPI. Стоит добавить, что количество вакансий на данный фреймворк растет с каждым годом. Flask постепенно уступает позиции в индустрии – сегодня спрос на разработчиков со знанием Flask ниже по сравнению с более современными фреймворками.

После детального изучения трех Python-фреймворков можно сделать вывод, что выбор между ними должен основываться на конкретных требованиях проекта. Django идеально подходит для крупных проектов с четкой структурой, а благодаря “батарейкам” он сможет сэкономить десятки часов разработки. Flask, наоборот, предлагает программисту максимальную свободу. Вместе с тем это требует от разработчика большой ответственности: часть функционала, которая у Django идет “из коробки” требует самостоятельной реализации в Flask. FastAPI, являясь в некотором смысле компромиссом между Django и Flask, сочетает в себе высокую производительность асинхронного кода с удобством разработки. Автоматическая документация и встроенная валидация данных делают FastAPI отличным выбором для микросервисной архитектуры. В конечном итоге, нет “лучшего” фреймворка – есть наиболее подходящий для конкретной задачи. Также важно помнить, что эффективность технологий зависит от того, насколько правильно они применены.

Список использованных источников:

1. History of Django Framework [Электронный ресурс] : Zavod-IT. – Режим доступа: https://zavod-it.ru/blog/useful/history-of-django-framework. – Дата доступа: 09.04.2025.

2. Web Framework Benchmarks [Electronic resource] : TechEmpower. – Mode of access: https://www.techempower.com/benchmarks/#section=data-r23. – Date of access: 10.04.2025.

3. PyPI - хранилище программного опеспечения для языка программирования Python [Электронный ресурс]. - Режим доступа: https://pypi.org. – Дата доступа: 10.04.2025.

UDC 004.42

COMPARATIVE ANALYSIS OF FASTAPI, DJANGO, AND FLASK: PERFORMANCE, DEVELOPMENT CONVENIENCE, AND CHOOSING THE OPTIMAL FRAMEWORK FOR MODERN WEB APPLICATIONS

Khudnitskii A.V., student of group 353502, Shauro E.A., student of group 353502

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Zhvakina A.V. – Candidate of Technical Sciences, Associate Professor

Annotation. The article presents a comparative analysis of three popular Python frameworks - FastAPI, Django, and Flask - focusing on their performance, development convenience, and applicability in modern web projects. Key features are examined: asynchronous request processing in FastAPI, the "batteries-included" approach and structured architecture of Django, as well as the minimalism and flexibility of Flask. Based on an evaluation of each framework's strengths and weaknesses, recommendations are provided for selecting optimal solutions for different use cases.

Keywords. FastAPI, Django, Flask, Comparative analysis, Performance, Development convenience, Asynchronous processing, Minimalism, Web applications, High-load APIs, Business logic, Optimization of choice, Modern technologies.