

# Сравнительный анализ инструментов разработки для проекта Pixo

## Сравнительный анализ №1: Backend-инструменты

Сравним три фреймворка для Python, которые помогают создавать серверную часть сайта: Flask, Django, FastAPI.

Инструмент	Простота использования	Функциональность	Документация и сообщество	Совместимость с другими технологиями	Производительность	Достоинства	Недостатки
Flask	Прост в использовании, легкий в обучении с нуля	Только базовые функции для web	Очень большое сообщество, множество туториалов	Легко интегрируется с фронтендом React и облачным хранилищем	Высокая, не нагружает систему лишними функциями	Легко начать, гибкий, не нагружает систему	Нужно самим придумывать структуру проекта и писать много кода
Django	Немного сложнее Flask, есть «магия», которая делает много сама	Полный фреймворк, ORM, админка, аутентификация	Большое сообщество, много готовых решений	Интеграция с React возможна через REST API	Ниже, чем у Flask из-за накладных расходов на встроенные компоненты. Все еще очень высока для большинства задач.	Много готовых технологий, меньше ручной работы	Перегружен для маленького проекта, сложнее учиться
FastAPI	Прост в базовом использовании, async может быть сложным	Современный фреймворк для API, поддержка async	Подробная документация растущее сообщество	Отлично подходит для фронтенда на React и работы с JSON	Очень высокая, благодаря архитектуре. Один из самых быстрых фреймворков на Python.	Быстро делать API, есть встроенная документация	Избыточный функционал, меньше примеров для обучения

### Вывод:

Мы решили использовать Flask, потому что с ним проще всего начать проект, он меньше нагружает систему, следовательно подходит для учебного проекта. Django и FastAPI — более продвинутые инструменты, слишком сложные и перегруженные для наших целей.

## Сравнительный анализ №2: Frontend-инструменты

Сравним три инструмента для разработки интерфейсов: React, Vue.js, Angular.

Инструмент	Простота использования	Функциональность	Документация и сообщество	Совместимость с другими технологиями	Производительность	Достиоинства	Недостатки
React	Средняя — нужно понять основы компонентов и состояния, но легко привыкнуть	Высокая — можно создавать любые интерфейсы, много библиотек	Очень большая база знаний, огромное сообщество	Отлично работает с разными технологиями (Node.js, Flask, Django и др.)	Высокая, интерфейс работает быстро	Простота создания интерфейсов, гибкость, большое сообщество	Требует дополнительных библиотек (например, для маршрутизации или работы с формами)
Vue.js	Очень простая для начала — понятный синтаксис, быстро осваивается	Хорошая — всё нужное уже встроено, но меньше сторонних библиотек	Хорошая документация, активное сообщество, особенно в Азии	Хорошо интегрируется с backend на Python.	Очень высокая, лёгкий и быстрый фреймворк	Простота, лёгкость, интуитивность, хороший выбор для новичков	Меньше примеров, чем у React или Angular
Angular	Сложнее в освоении — много терминов и строгих правил	Очень высокая — включает всё: маршрутизацию, формы, сервисы	Отличная официальная документация, большое сообщество	Хорошо работает с любым backend, но требует TypeScript.	Высокая, но может быть "тяжёлым" для мелких проектов	Полный набор инструментов из коробки, надёжность, структура	Сложный порог входа, больше кода, чем в React/Vue

### Вывод:

Для нашего проекта мы выбрали React+JavaScript, потому что он простой для новичка, гибкий, и можно легко соединять с сервером на Python. Vue.js тоже простой, но React более популярный, а Angular слишком сложный для учебного проекта.