Новости ИИ: как выбрать фреймворк для ИИ-разработки и не переплатить

10.11.2026 · PDF-версия статьи с сайта cms.dolinais.ru

Почему стоимость ИИ-разработки стала новостью

В ноябре 2026 года разработчик Нолан Лоусон опубликовал сравнение, сколько денег и времени требуется нейросетям для создания одного веб-приложения на разных JavaScript-фреймворках. Результат оказался неожиданным для многих: самым дорогим вариантом стал Angular, а самым дешёвым — связка Hono и htmx. Разница в медианной стоимости — почти два раза. Это не просто любопытный факт: если вы планируете бюджет на ИИ-агентов или выбираете стек для проекта, где код будет генерироваться машиной, цифры напрямую влияют на экономику продукта.

В этой статье разберём, почему фреймворк влияет на расходы, как считать стоимость агентной разработки и какие практики из свежих материалов помогут не переплачивать. Речь пойдёт о 2026 годе — с учётом текущих инструментов и ограничений.

Что показало исследование

Нолан Лоусон сравнил несколько популярных JavaScript-фреймворков в задаче генерации одного и того же приложения с помощью ИИ. Метрики — деньги и время. Самый дорогой — Angular, самый дешёвый — Hono + htmx. Разница в медианной стоимости почти двукратная. Причины не в «плохости» Angular, а в том, как модели работают с контекстом.

Чем больше в фреймворке неявных соглашений, boilerplate и специфического синтаксиса, тем больше токенов агенту нужно прочитать и сгенерировать. Чем проще и предсказуемее API, тем короче ответы и меньше итераций. Hono — минималистичный веб-фреймворк для edge-сред, htmx — библиотека, которая позволяет делать динамические интерфейсы через HTML-атрибуты без тяжёлого клиентского JS. Оба дают модели короткий и однозначный контекст, что снижает расход токенов и количество правок.

> Стоимость ИИ-разработки — это не только цена токенов, но и число итераций, повторных чтений истории и исправлений после ошибок.

Почему фреймворк влияет на счёт

Агент работает итеративно: читает задачу, генерирует код, получает ошибку, исправляет. Каждый шаг — это токены. Если фреймворк требует много шаблонного кода или имеет неочевидные правила, модель чаще ошибается, и цикл удлиняется. В материале «AI-агент в проде для тех, кто вкатывается» приведён показательный пример: агент вместо 30 ходов сделал 100, а денег ушло в 5,3 раза больше, потому что на каждом ходу он заново читал всю историю. Если процесс упал на 25-м ходе и задачу запустили с нуля, она дорожает на 81%, а действия повторяются — например, письмо клиенту уходит дважды.

Отсюда практический вывод: выбирая стек, вы выбираете не только удобство для людей, но и «стоимость» для модели. Простые, явные API и минимум магии снижают расходы. Это не значит, что нужно отказываться от

Angular или других мощных фреймворков — но в проектах с жёстким бюджетом на ИИ-генерацию стоит оценить альтернативы.

Как считать бюджет агента

Чтобы не переплачивать, введите базовую единицу — стоимость одного «хода» агента. Она складывается из цены входных и выходных токенов, длины контекста и числа шагов. Практические шаги:

1. **Замерьте базовый сценарий.** Возьмите типовую задачу (например, создание CRUD-страницы) и прогоните её на выбранном стеке. Зафиксируйте число ходов, объём токенов и время.
2. **Ограничьте цикл и бюджет.** Задайте максимальное число ходов и потолок расходов. В том же материале рекомендуется ограничивать цикл, иначе агент может «застрять» и сделать 100 ходов вместо 30.
3. **Держите контекст коротким.** Не подавайте агенту всю историю на каждом шаге. Используйте суммаризацию, RAG или переранжирование. В статье приводятся данные: поиск по словам и переранжирование сокращают провалы с 5,7% до 1,9%.
4. **Делайте инструменты идемпотентными.** Повторный вызов не должен создавать дубли. Это защищает от двойных писем и повторных действий при перезапуске.
5. **Добавьте наблюдаемость.** Логируйте ходы, токены и ошибки. Без этого невозможно понять, где бюджет утекает.

Для оркестрации длительных процессов в проде упоминается Temporal — инструмент, который помогает управлять состоянием и повторными попытками. Но начинать стоит с простых ограничителей и логирования.

Практические рекомендации по выбору стека

Если проект предполагает активную ИИ-генерацию кода, оцените стек по трём критериям:

- **Явность API.** Чем меньше неявных соглашений, тем предсказуемее ответы модели.
- **Объём boilerplate.** Минимум шаблонного кода снижает число токенов и правок.
- **Распространённость примеров.** Модели лучше знают популярные библиотеки, но популярность не всегда означает дешевизну: Angular популярен, но в тесте оказался дорогим.

Не стоит делать вывод, что htmx подходит для всего. Для сложных интерфейсов может понадобиться React или Vue. Но если задача — небольшое приложение или внутренний инструмент, связка Hono + htmx может оказаться экономически выгодной. Главное — измерить на своём сценарии, а не полагаться на общие советы.

Что ещё важно учитывать в 2026 году

Стоимость — не единственный фактор. Надёжность и безопасность тоже влияют на бюджет: ошибки агента приводят к повторным запускам, а утечки данных — к штрафам. В материале про агента в проде подчёркивается важность подтверждения человеком для критичных действий. Это особенно актуально для финансовых операций и коммуникаций с клиентами.

Кроме того, не забывайте про RAG (Retrieval-Augmented Generation — генерация с дополненной выборкой): в некоторых случаях поиск по документации сокращает число ошибок и повторных попыток. Но RAG не нужен

вовсе, если задача укладывается в контекст модели. Решение принимается по метрикам, а не по моде.

Вывод

Новость о разнице в стоимости ИИ-разработки на разных фреймворках — повод пересмотреть бюджет и стек. В 2026 году экономика агентной разработки зависит от архитектуры, а не только от цены токенов. Ограничивайте циклы, держите контекст коротким, делайте инструменты идемпотентными и измеряйте стоимость на своём сценарии. Тогда выбор фреймворка станет инженерным решением, а не лотереей.

Источник: https://cms.dolinais.ru/novosti-ii-kak-vybrat-freymvork-dlya-ii-razrabotki-i-ne-pereplatit