GigaChat или Claude — не тот вопрос. Реальный выбор B2B-команды РФ в 2026 — моноархитектура или роутер

2026-04-28

GigaChat или Claude — не тот вопрос. Реальный выбор B2B-команды РФ в 2026 — моноархитектура или роутер

TL;DR. Сравнения GigaChat, YandexGPT и Claude по цене токена и набору фич в 2026-м воспроизводят ту же ошибку, что обзоры BPM-систем десять лет назад: они отвечают на закупочный вопрос «какого вендора выбрать» вместо архитектурного — «как устроен ваш стек, чтобы вендор был заменим». Дешевизна моноархитектуры в первый год оплачивается стоимостью миграции на втором, и это уже измеримая величина — Anthropic, OpenAI, Google и AWS за один квартал 2026-го схлопнули agent harness в managed-продукт, а Сбер и Yandex Cloud отвечают тем же. Заметка для CTO, Head of Product и Tech Lead, которые в апреле 2026-го собирают LLM-стек и не хотят пересобирать его в апреле 2027-го.

Апрельский квартал, в котором провайдеры стали заменимы

8 апреля 2026 года Anthropic выпустил Managed Agents — managed execution loop, persistent memory через `/memories`, sandboxing и multi-agent orchestration. В тот же месяц OpenAI отгрузил AgentKit с визуальным Agent Builder, Connector Registry и встроенными Evals. Сбер на портале GigaChat API держит публичную тарификацию по пакетам токенов и три варианта развёртывания — облако, гибрид, on-prem. Yandex Cloud в Foundation Models собрал YandexGPT в одну линейку с Object Storage, Logging и managed-инфраструктурой.

Между этими четырьмя анонсами есть общая черта, которую обзоры обычно проходят мимо. Все четыре провайдера в 2026-м продают не модель, а стек: контракт API, managed orchestration, memory, observability. И как только стек становится частью продукта, цена ошибки выбора смещается с цены токена на стоимость переезда между стеками. Ровно поэтому сравнительные таблицы «GigaChat vs YandexGPT vs Claude» в 2026-м измеряют не то, что определяет экономику решения через год.

Дешевизна моноархитектуры — в долг

Базовый сценарий 2026-го у российской B2B-команды выглядит так. Команда выбирает одного провайдера — обычно по сочетанию цены, 152-ФЗ и удобства существующей инфраструктуры — и собирает на нём всё: парсинг входящих, классификацию, агентов с инструментами, summarisation. Это моноархитекту-

ра. Она экономит инженерные часы на старте: одна авторизация, один SDK, один формат tool calling, одна managed-память.

Контр-тезис: моноархитектура не дешевле — она дешевле в первый год. На втором году стоимость моноархитектуры — это стоимость миграции, и она конечна, измерима и обычно недооценена. Migration debt появляется не от плохого выбора, а от того, что любая моноархитектура жёстко связывает четыре независимых решения: формат структурированного вывода, протокол tool use, схему памяти и operations-контур. Когда хотя бы одно из четырёх перестаёт устраивать, переезжать приходится сразу всем стеком.

Альтернатива — роутер: внешний слой, который маршрутизирует запросы между провайдерами по типу нагрузки, чувствительности данных и стоимости. Роутер дороже в первый год — нужен evaluator-харнесс, единая schema domain-объектов, контракты между prompt-цепочкой и моделью. Но он линейно масштабирует решения второго года: смена одного провайдера меняет конфиг, а не архитектуру. В терминах предыдущей заметки про harness commodity, роутер — это та часть operating layer, которую provider-стеки в 2026-м начали поглощать, и которую командам выгодно не отдавать вверх по стеку без явного обмена.

Четыре стыка, на которых ломается моноархитектура

Чтобы увидеть, где конкретно моноархитектура накапливает долг, удобно смотреть не на «провайдеров вообще», а на стыки между подсистемами стека.

Structured output: формат входит в каждый prompt

OpenAI в Function Calling с 2023-го фиксирует контракт, при котором модель возвращает строго типизированный JSON по объявленной схеме. Anthropic в tool use реализует тот же контракт через `tool_choice` и явные input schemas. Это два формата, которые внешне делают одно и то же, но различаются в деталях: имена полей, форма передачи schema, поведение при несоответствии. На стороне Сбера механизм function calling в портале GigaChat API и сопровождающих технических разборах SberDevices на Хабре представлен своим интерфейсом; у Yandex — своим.

Команда, которая выбрала одного провайдера, кодирует именно его формат в каждый prompt и каждый валидатор. Через год, когда возникает сценарий с другим провайдером — например, reasoning-задача, где Claude даёт лучший результат на тех же примерах — миграция перетряхивает все места, где формат tool calling зашит. Это не «переписать промпты», это переписать тесты, евалуаторы, ретраи и логирование.

Команда с роутером изначально держит структурированный вывод как domain-объект, а формат провайдера — как адаптер; миграция меняет адаптер. Конкретно это значит, что в репозитории есть два уровня: типизированные классы доменных объектов (карточка клиента, событие, эскалация, статус) — независимые от провайдера, и тонкий слой провайдер-специфичного кода, ко-

торый только переводит между этими классами и форматом конкретного API. На длинных горизонтах поддержка двух адаптеров обходится дешевле одного жёстко прошитого формата, потому что оба адаптера обязаны проходить один и тот же evaluator-харнесс.

Tool use: orchestration провайдера против внешнего роутера

Anthropic в инженерной заметке про Managed Agents описывает execution loop, в котором агент «self-evaluates and iterates until it reaches a result». OpenAI в AgentKit даёт визуальный Agent Builder с Connector Registry и встроенными Evals. Это два разных способа упаковать одну и ту же задачу: длинная цепочка вызовов инструментов с возвратом контроля внешнему коду. Российские провайдеры в публичных материалах 2026-го тоже движутся в эту сторону, но с разной плотностью документации и разной зрелостью контракта на длинных горизонтах.

Команда, которая поверила, что «agent harness — это просто фича провайдера», и спроектировала flow внутри managed-конструкции, на втором году получает классическую vendor lock-in проблему. Смена провайдера здесь — это не «переключить ключ», это переписать саму форму orchestration: где живёт состояние, как оформлены retry, кто owns memory между шагами, как описан контракт инструмента. У каждого провайдера эти ответы разные, и в managed-продукте они зашиты в SDK, а не в коде команды.

Команда с внешним роутером — наоборот, держит orchestration снаружи, а провайдерский harness использует только там, где экономия на инфраструктуре оправдывает связанность. На практике это означает, что роутер берёт на себя три роли: классификатор запроса по типу нагрузки и чувствительности данных, владелец схемы domain-объектов и evaluator. Managed harness провайдера в такой архитектуре — один из исполнителей, не источник правды.

152-ФЗ как архитектурный, а не комплаенс-вопрос

Большинство обзоров обсуждают 152-ФЗ как фильтр между двумя множествами провайдеров: «российские можно, зарубежные нельзя для ПД». Это правда, но это не самый интересный уровень. Архитектурный уровень — что именно в стеке считается «обработкой ПД», и где проходит граница изоляции.

Документация GigaChat API предлагает три варианта развёртывания: облако Сбера, гибрид с данными на стороне клиента, on-prem. Yandex Cloud Foundation Models даёт data residency «в РФ» как умолчание. Команда, которая делает 152-ФЗ-чувствительный продукт на моноархитектуре одного из этих провайдеров, имплицитно принимает решение: весь продукт работает в одном комплаенс-периметре. На горизонте года это означает, что если в продукт добавляется задача, требующая reasoning-возможностей зарубежной модели — а такие задачи в 2026-м появляются регулярно — переезд требует разделить пайплайн на ПД-чувствительный и обезличенный, что архитектурно эквивалентно построению роутера задним числом, только в стрессе и под deadline.

Команда, которая с самого начала проектировала разделение по чувствительности данных как отдельную ось маршрутизации, ту же задачу решает добавлением ветки. 152-ФЗ перестаёт быть constraint на выбор провайдера и становится одним из measurable атрибутов запроса.

SLA и поведение под нагрузкой

У Anthropic и OpenAI публичные status-страницы и rate-limit-политики, описанные на уровне tier'ов. У GigaChat и Yandex Cloud публичная часть SLA на инференс LLM в 2026-м заметно беднее — корпоративные условия идут в индивидуальных договорах. Это не утверждение про качество, это утверждение про объём публичной информации, на которую может опираться внешний выбор.

Моноархитектура на любом из четырёх провайдеров делает SLA провайдера потолком SLA продукта. Если у провайдера падает регион или меняется rate-limit-политика, продукт деградирует синхронно, без степеней свободы. Роутер — наоборот, делает SLA продукта функцией политики маршрутизации: при деградации одного провайдера запросы уходят на резерв, latency p95 поднимается, но контур не останавливается. Цена этой устойчивости — поддерживать как минимум двух работающих провайдеров и единый evaluator-харнесс, на котором обе стороны проверяются на одинаковом наборе кейсов. Дешёвой эта устойчивость не бывает; вопрос в том, что обходится дороже — её отсутствие или её содержание.

Конкретные тесты для трёх ролей

Для CTO. Возьмите три задачи из текущего бэклога: одну со structured output, одну с tool use на 5+ шагов, одну с длинным контекстом. Зафиксируйте, в скольких местах кода и в скольких тестах прописан конкретный формат текущего провайдера — имена полей tool calling, формат сообщений, идентификаторы моделей. Если число таких мест превышает несколько десятков на одну задачу, у вас моноархитектура, и стоимость её замены через год — это столько же изменений, помноженное на число задач. Решение — не «срочно мигрировать», а вынести формат провайдера в адаптер и держать domain-объекты типизированными.

Для Head of Product. Запишите, какие фичи продукта зависят от поведенческих гарантий конкретного провайдера: структуры JSON, длины контекста, поведения tool calling на длинных горизонтах. Если таких фич больше трёх и все они сидят на одном провайдере, продукт зависит от его roadmap'а сильнее, чем от собственного. Тестовый сценарий: если завтра ваш текущий провайдер поднимет цену на 30% или удалит конкретную фичу — какие функции продукта перестают работать в течение недели. Это и есть продуктовая стоимость моноархитектуры, выраженная в feature-availability.

Для Tech Lead. Постройте один проект так, чтобы провайдер был заменим: единая schema domain-объектов, адаптеры под форматы провайдеров, evaluator-харнесс на 200–500 типичных кейсов с эталонными ответами. Это

инвестиция в недели, а не в дни. Признак, что харнесс реальный, а не на бумаге: вы можете прогнать новую модель на нём за ночь и получить численный ответ — лучше, хуже, на каких подмножествах. Без такого харнесса любая дискуссия о смене провайдера ведётся в терминах ощущений, что само по себе — диагностика степени lock-in.

Сигналы 2026 года

Тренд имеет несколько проверяемых сигналов. Первый — появление managed-роутеров в публичных продуктах: внешних сервисов, которые принимают единый API и сами решают, какому провайдеру отправить запрос. Если такие продукты выходят в GA, моноархитектура становится анахронизмом, а вопрос смещается на политику маршрутизации. Второй — публикация официальных кейсов перевода production-нагрузок между российскими и зарубежными провайдерами без потери качества. Такой кейс — публичное доказательство, что архитектурно стек уже переносим, и оценочная стоимость миграции из моноархитектуры в роутер становится прозрачной величиной. Третий, обратный, — появление команд, которые публично возвращаются с роутера на моноархитектуру, объясняя это операционной сложностью. Это будет означать, что граница применимости роутера выше, чем сейчас принято считать, и что для значительной части B2B-задач достаточно одного провайдера и грамотных адаптеров.

В любом сценарии вопрос «GigaChat или Claude» — не главный. Он выглядит как закупочный, но за ним стоит архитектурный, и именно архитектурный определяет, сколько будет стоить второй год.

Главное

- Сравнения LLM-провайдеров по цене и фичам в 2026-м воспроизводят ошибку обзоров BPM десять лет назад: они отвечают на закупочный вопрос вместо архитектурного.
- Реальная развилка — моноархитектура или роутер. Моноархитектура дешевле в первый год и дороже на втором за счёт стоимости миграции; роутер — наоборот.
- Стоимость моноархитектуры скрыта в четырёх стыках: формат structured output, протокол tool use на длинных цепочках, граница 152-ФЗ-изоляции, поведение под нагрузкой.
- Минимальная архитектура, которая делает провайдера заменимым: единая schema domain-объектов, адаптеры под форматы провайдеров, evaluator-харнесс на 200–500 кейсов.
- Сигналы 2026-го, на которые стоит смотреть: managed-роутеры в GA, публичные кейсы переноса production-нагрузок, обратные кейсы возврата на моноархитектуру.