Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

ИИ-агент поддержки: закрываем 40-70% обращений без оператора

Как мы внедряем Claude-агента с tool use для «где заказ» и «сколько стоит»

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Малый бизнес на 5-50 сотрудников тонет в однотипных обращениях «где мой заказ» и «сколько стоит»: они не требуют участия человека, но держат клиента в очереди и съедают время оператора. Без автоматизации — либо второй оператор под пиковую нагрузку, либо падение конверсии из-за медленных ответов вечером и в выходные, когда на линии никого нет.

Почему это важно бизнесу

- Один оператор не отвечает мгновенно на 2-3 диалога параллельно — очередь растёт, клиент уходит к конкуренту с быстрым ответом.
- Найм второго оператора под пиковую нагрузку стоит дороже, чем токены агента на тот же объём типовых вопросов.
- Ошибка в озвученной цене или статусе заказа — репутационный и юридический риск, если менеджер отвечает «на глаз».
- Ночью и в выходные без агента бизнес теряет заявки: клиент без ответа за 5-10 минут чаще не возвращается.
- Эскалация «на всякий случай» без порога либо спамит операторов, либо агент молча тонет с недовольным клиентом.

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

40-70%

доля «где заказ»/«сколько стоит», которую агент закрывает без оператора — наш целевой KPI

наш стандарт

-0.5

порог сентимент-скора диалога для принудительной эскалации оператору (шкала -1…1)

наш стандарт

128K

потолок max_tokens при SSE-стриминге ответа — запас на tool use и адаптивное мышление модели

Claude API docs 2026

1 ч

TTL кэша статичного контекста (FAQ, прайс) вместо дефолтных 5 минут при частых обращениях

Prompt caching, Anthropic

~90%

экономия по цене токенов при чтении из кэша (cache_read) вместо полного прогона промпта

Prompt caching, Anthropic

-50%

стоимость ночного eval-регресса промпта на Message Batches API против обычных запросов

Batches API, Anthropic

РЕАЛИЗАЦИЯ 1 ИЗ 3

Статус заказа через tool use и CRM

Что настраиваем

Интернет-магазин/сервис с заказами в Bitrix24 или amoCRM, виджет на сайте и в Telegram

Как мы это делаем

1. Регистрируем клиентский tool get_order_status(order_id?, phone?) со strict:true — Claude не выдумывает поля, схема валидируется на входе.
2. Обработчик на сервере получает tool_use и дёргает Bitrix24 REST (crm.deal.list) или amoCRM API v4 по номеру/телефону через вебхук-токен, который не уходит в браузер.
3. При таймауте или 4xx/5xx возвращаем tool_result с is_error:true — агент честно говорит «сервис недоступен, передаю оператору», а не выдумывает статус.
4. Ответ форматируем через output_config.format (json_schema): статус, дата, трек-номер — рендерим карточкой в виджете, а не голым текстом.
5. Если заказ не найден по номеру — включаем уточняющий вопрос про телефон/email прежде чем эскалировать диалог.

РЕЗУЛЬТАТ

Клиент получает статус заказа за секунды без ожидания оператора, оператор не отвлекается на рутинный поиск в CRM, а закрытые тикеты не создают ложных обещаний — статус всегда идёт из CRM, а не из догадки модели.

КЛЮЧЕВОЙ НЮАНС

Права tool'а — read-only. Агенту нельзя давать API-ключ с правом менять сделку: разграничение прав на уровне серверного вебхука защищает от prompt injection, которая просит «отмени заказ» через чат.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Расчёт цены и коммерческого предложения

Что настраиваем

Товары/услуги с калькулируемой ценой (доставка, объём, тариф) — сайт и мессенджер

Как мы это делаем

1. Описываем tool get_price с enum-полями (тип услуги, объём, регион) — enum вместо свободного текста снижает разброс входных данных.
2. Модель Claude Sonnet 5 с thinking: {type:'adaptive'} сама решает, нужен ли уточняющий вопрос перед вызовом tool, и не считает цену по неполным данным.
3. Прайс-логика живёт не в промпте: tool дёргает наш прайс-калькулятор (Python/1C), возвращает число и условия действия цены.
4. Кэшируем статичный блок FAQ и тарифной сетки через cache_control: {type:'ephemeral', ttl:'1h'} — токены прайс-листа не пересчитываются на каждый диалог.
5. Финальный ответ проверяем output-guardrail на forbidden-фразы (обязательства, скидки не из прайса) перед отправкой клиенту.

РЕЗУЛЬТАТ

Цена всегда посчитана детерминированным калькулятором, а не сгенерирована моделью — снимаем разброс и юридические риски по озвученным обязательствам, а кэш на статичный прайс держит стоимость диалога низкой даже при росте трафика.

КЛЮЧЕВОЙ НЮАНС

Никогда не просим модель «посчитать» цену в уме — только через tool с реальной бизнес-логикой на бэкенде; в input_schema используем enum, а не строку, там где значения известны заранее.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Эскалация к оператору и защита от злоупотреблений

Что настраиваем

Виджет поддержки на сайте с очередью операторов в Bitrix24/amoCRM

Как мы это делаем

1. Считаем сентимент-скор диалога на каждом ответе; при падении ниже -0.5 или ключевых словах («менеджер», «жалоба», «возврат») — принудительная эскалация.
2. Эскалация создаёт задачу в Bitrix24 (task.item.add) или amoCRM (сделка/задача) через тот же серверный вебхук с полной историей диалога в описании.
3. В системном промпте держим только роль (identity), основной объём контекста — FAQ, примеры, guardrails — в первом user-сообщении, как рекомендует дока Anthropic.
4. Добавляем input/output-guardrail слой отдельно от модели: фильтр на кодировки (base64/leetspeak) и на упоминания конкурентов до и после ответа Claude.
5. Логи диалогов чистим от PII (телефон, адрес) перед сохранением в аналитику — оставляем только хэш или ID клиента.

РЕЗУЛЬТАТ

Оператор получает эскалацию с полным контекстом диалога вместо «здравствуйте, опишите проблему заново», а бизнес не зависит только от системного промпта как единственной линии защиты от злоупотреблений.

КЛЮЧЕВОЙ НЮАНС

Порог эскалации не выставляют один раз и забывают: гоняем его на 20-30 «злых» тестовых диалогах и подстраиваем — иначе либо заваливаем операторов, либо агент застревает с недовольным клиентом.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- **Цена «из головы» модели**
  Без tool модель иногда достраивает цену по паттерну из контекста — юридический риск. Цена должна идти только из калькулятора, не из текста промпта.

- **Нет кэша на статичный контекст**
  Каждый диалог заново прогоняет FAQ и прайс — дорого и медленно. Решение: cache_control ephemeral, ttl 1h на статичные блоки промпта.

- **Write-доступ агента к CRM**
  Если tool может менять сделку, а не только читать, инъекция в чате может отменить заказ. Решение: read-only токен на сервере, запись — только вручную.

- **Нет обработки is_error от CRM**
  При таймауте API агент может выдумать статус вместо «сервис недоступен». Решение: tool_result с is_error:true ведёт к фразе-заглушке и эскалации.

- **Sentiment-порог не откалиброван**
  Порог -0.5 без теста на «злых» диалогах либо заваливает операторов, либо агент долго терпит клиента. Тестируем на 20-30 кейсах перед запуском.

- **SSE без heartbeat и reconnect**
  На мобильном интернете соединение рвётся, диалог зависает без ответа. Решение: heartbeat каждые 15-20 секунд и reconnect клиента с sequence-id.

- **Весь контекст в system-промпте**
  Вся логика в system role вместо первого user-сообщения снижает управляемость (доки Anthropic). В system — только роль, остальное в первом user turn.

- **Нет eval перед релизом промпта**
  Правки промпта без регресс-теста на edge-кейсах (отмена, спор о сумме, грубость) незаметно роняют качество. Гоняем eval в Batches API перед релизом.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- 1 tool (get_order_status) + FAQ прямо в первом user-сообщении, без RAG
- Ручная эскалация по ключевым словам: менеджер, жалоба, возврат
- Одна модель на все задачи, без отдельного intent-роутера

НОРМАЛЬНО

- 2 tool'а (заказ + цена) и prompt caching FAQ с ttl 1h
- Сентимент-эскалация (-0.5) плюс вебхук в Bitrix24/amoCRM на задачу
- SSE-стриминг ответа в виджете и eval-регресс на 30-50 диалогов

ХОРОШО

- RAG по каталогу (эмбеддинги) плюс structured outputs для карточек цены
- Batches API для ночной прогонки регресс-тестов промпта — дешевле в 2 раза
- Haiku-классификатор намерений перед основной моделью-исполнителем
- Полный guardrail-стек: input/output фильтры, least-privilege tools, PII-стрип логов

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Есть отдельный read-only tool для статуса заказа без write-доступа к CRM?
- [ ] Задан порог сентимент-эскалации и протестирован на 20+ «злых» диалогах?
- [ ] Включён prompt caching для статичного FAQ и прайса (cache_control)?
- [ ] Обрабатывается is_error от CRM API при таймаутах и 4xx/5xx?
- [ ] Цена всегда идёт из tool-калькулятора, а не из текста модели?
- [ ] Настроены SSE-heartbeat и reconnect для мобильных клиентов?
- [ ] Есть eval-датасет с граничными кейсами: отмена, спор о сумме, грубость?
- [ ] Логи агента очищены от PII перед хранением и аналитикой?
- [ ] Определён протокол ретраев для 429/5xx Claude API в проде?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Проектируем и разворачиваем tool-схемы статуса заказа и расчёта цены под вашу CRM (Bitrix24/amoCRM/1C)
- Настраиваем эскалацию к операторам: сентимент-пороги, вебхуки задач, дежурные очереди
- Внедряем prompt caching и RAG по каталогу — снижаем стоимость и задержку каждого ответа
- Пишем eval-датасет и гоняем регресс перед каждым релизом промпта через Batches API
- Ставим SSE-виджет на сайт с heartbeat/reconnect и мониторингом time-to-first-token

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 Customer support agent — use-case guide (platform.claude.com — 2026)

02 Tool use overview (platform.claude.com — 2026)

03 Prompt caching guide (platform.claude.com — 2026)

04 Message Batches API (platform.claude.com — 2026)

05 Structured outputs (platform.claude.com — 2026)

06 REST API и вебхуки Bitrix24 (dev.1c-bitrix.ru — 2026)

07 amoCRM API v4 (сделки, вебхуки) (developers.amocrm.ru — 2026)

08 Наш шаблон эскалации и guardrails (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.