Методология проектной работы Metabot

Основная логика ведения проектов и создания ботов “Метабот-стайл”

Ключевая идея процесса

Создание бота или цифрового ассистента в Metabot — это **не “построить кнопки” и “написать пару текстов”**.
Это полноценный **инженерно-производственный процесс**, который соединяет:

- аналитику,
- проектирование,
- разработку,
- тестирование,
- запуск,
- поддержку и развитие.

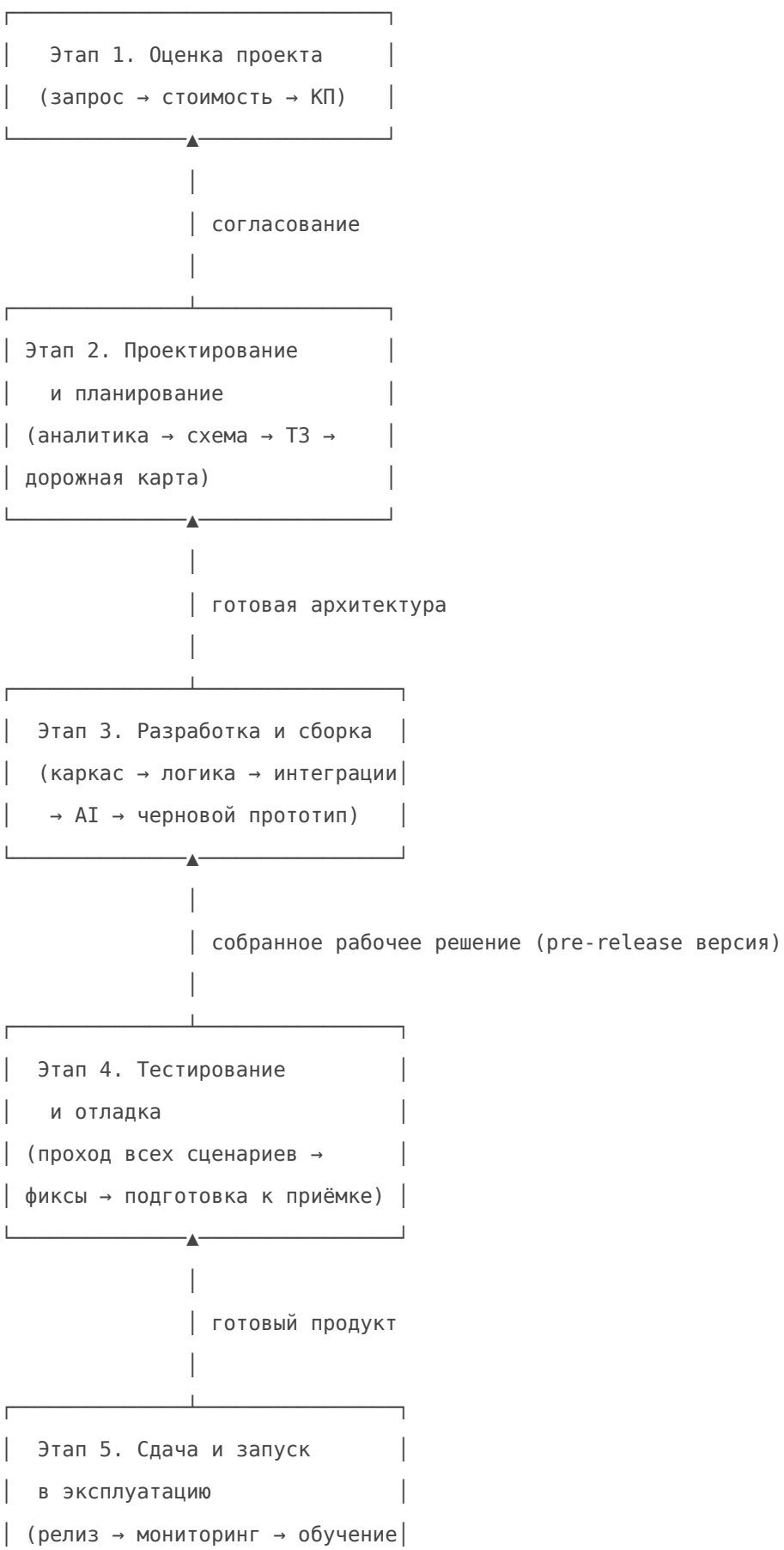
Мы работаем как команда, создающая **цифровой продукт**, который должен:

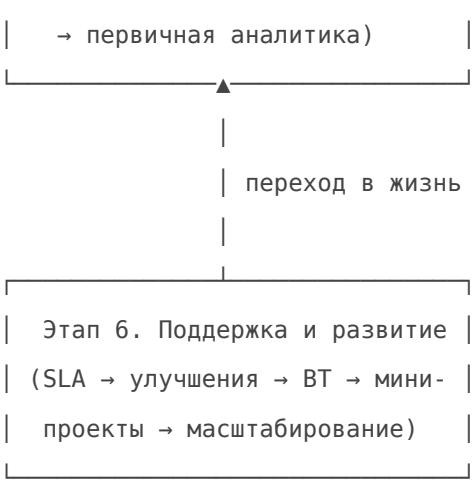
- решать реальную задачу бизнеса,
- иметь понятную архитектуру,
- быть устойчивым под нагрузкой,
- легко развиваться,
- быть прозрачным для других специалистов команды,
- обладать предсказуемым жизненным циклом.

Чтобы это обеспечить, мы используем **шестиступенчатую методологию Metabot Project Delivery**.

Схема процесса (общая логика)

Ниже — чистая, визуально читаемая схема процесса в стиле “одно окно понимания”.





Ниже подробное описание каждого этапа.

Этап 1 — Оценка проекта

Этот этап отвечает на базовый вопрос: **«Что именно делаем, сколько это стоит и в каком формате работаем?»**

1. Цели этапа

- Зафиксировать **исходный запрос клиента** и перевести его с «хочу бота» на язык задач и ограничений.
- Прикинуть **трудозатраты и стоимость** проекта с разумной точностью.
- Определиться с **моделью работы**: фиксированная цена или time & material.
- Подготовить для клиента **коммерческое предложение** (КП) и при необходимости — **концепцию решения**.

---

2. Входы этапа

Что нам нужно собрать, прежде чем считать деньги:

- Описание задачи от клиента (часто — «как есть», голосом/текстом).
- Базовая информация:
  - какие каналы нужны (Telegram, VK, WhatsApp, сайт, виджет и т.п.);
  - есть ли уже CRM, 1С, телефония, склад и т.д.;
  - какие процессы хотят трогать (лиды, поддержка, продажи, склад, сервис).
- Ограничения:
  - сроки («к вчерашнему дню» / к мероприятию / «спокойно за 2 месяца»);
  - бюджетные рамки, если клиент ими делится;
  - инфраструктурные ограничения (облако/он-прем, безопасность, политика ИБ).

- Примеры и референсы (если есть):
  - существующие скрипты операторов, регламенты;
  - примеры чужих ботов, которые нравятся;
  - уже написанные ТЗ/идеи, даже сырые.

---

3. Аналитика на этапе оценки

Важно: **аналитика — это уже работа**. Но на этапе оценки мы делаем её в гибком режиме:

- **Минимальная аналитика** (для типовых решений):
  - уточняем целевую аудиторию и ключевые сценарии (1–3 основные воронки);
  - фиксируем базовую структуру бота и данных;
  - проверяем, нет ли «красных флажков» по интеграциям.
- **Расширенная аналитика** (если проект нестандартный или большой):
  - короткие интервью с заказчиком/ключевыми пользователями;
  - разбор текущего процесса «как есть»;
  - набросок целевой схемы «как должно быть»;
  - первичный перечень рисков и допущений.

Часть этой аналитики мы часто **делаем “в подарок”**, чтобы клиенту было легче принять решение. Но честно понимаем, что глубокая аналитика — это уже отдельный оплачиваемый этап/спринт.

---

4. Концептуальное описание проекта

На этом же этапе мы часто готовим **концепцию** — человеческий текст + упрощённую структуру решения:

- Формулируем **цель проекта** с нашей, профессиональной точки зрения.
- Переводим запрос клиента в **структурированное описание**:
  - роли пользователей (клиент, оператор, менеджер, монтажник, и т.п.);
  - основные сценарии и ветки (воронки, сервисные цепочки);
  - ключевые точки интеграции (CRM, склад, телефония, 1С и т.д.).
- Подсвечиваем то, что клиент **не учёл**, но критично:
  - где нужны доп. статусы, справочники, таблицы;
  - где важно сразу продумать аналитику, логи, права доступа и т.п.

Пример такого концептуального уровня — структурное ТЗ на бота для сервиса автозапчастей: там уже есть роли, сценарии, статусы, требования к базам данных и интеграциям, но это ещё не детальная разработка всех фраз и экранов.

Такую концепцию мы иногда делаем **как бонус** к КП, иногда — как отдельный документ (особенно в сложных проектах и тендерах).

---

5. Модели ценообразования

На этом этапе мы выбираем **одну из двух моделей работы**:

1. **Фиксированная цена (fixed price)**
   - Подходит для:
     - более-менее типовых проектов;
     - понятного объёма работ;
     - клиентов, которые хотят «конечную сумму» и понятный набор результатов.
   - Что делаем:
     - оцениваем трудозатраты с запасом по рискам;
     - включаем в цену возможные изменения в разумных пределах;
     - фиксируем границы проекта: что входит, а что явно не входит.
2. **Time & Material (T&M)**
   - Подходит для:
     - сложных, эволюционирующих проектов;
     - ситуаций, когда сам клиент до конца не понимает конечный объём;
     - долгосрочных историй (цифровая трансформация, постоянное развитие).
   - Что делаем:
     - согласовываем ежемесячный бюджет (N часов/человек-часов в месяц);
     - описываем формат работы (спринты, приоритизация задач, отчётность);
     - оцениваем примерную «длину пути» и первые шаги.

Обе модели мы прямо **прописываем в КП**, чтобы клиент понимал, почему фикс дороже и когда выгоднее работать по T&M.

---

6. Как мы готовим коммерческое предложение

У нас фактически **два способа** подготовить КП:

6.1. Ручное КП (под нестандартные проекты)

Используем, когда:

- проект сильно кастомный;
- стандартных форм недостаточно, нужно больше текста, аргументации, схем;
- у клиента важен **нарратив**: описание решения, обоснование, риски, дорожная карта.

В ручном КП обычно есть:

- краткое описание заказчика и контекста;
- цели и задачи проекта;
- предлагаемое решение (уровень концепции);

- этапы работ и сроки;
- модель ценообразования (fixed или T&M, либо комбинация);
- стоимость по этапам;
- условия поддержки/сопровождения.

6.2. КП на базе стандартной спецификации (Sheets-шаблон)

Для **типовых и ходовых решений** мы используем наш **автоматизированный шаблон спецификации**:

- Внутри шаблона:
  - листы с входными параметрами (каналы, интеграции, нужные модули, типы ботов, хостинг, поддержка и т.д.);
  - листы с трудозатратами по ролям (аналитик, архитектор, разработчик, интегратор, тестировщик, проджект);
  - прайсы и формулы, которые автоматически считают стоимость;
  - итоговый лист/листы, которые печатаются как готовая **КП-ка**.
- Мы заполняем только **ключевые листы**, а остальное генерится автоматически:
  - это **экономит нам кучу времени**;
  - уменьшает вероятность ошибок в расчётах;
  - стандартизирует подход к оценке.

В итоговом документе учитывается:

- оценка работ (по часам и ставкам);
- стоимость лицензий/коробки (если нужно);
- стоимость хостинга/облака (по выбранному тарифу);
- наличие/отсутствие сопровождения и его цена;
- дополнительные работы (интеграции, аналитика, обучение и т.п.).

Пример.

---

7. Формат выдачи КП клиенту

На выходе этапа клиент получает один или несколько артефактов:

1. **Коммерческое предложение** (в PDF/Excel/документе):
   - стоимость и структура работ;
   - модель оплаты (фикса или T&M);
   - базовые сроки и этапы;
   - условия запуска и сопровождения.
2. **Концептуальное описание проекта** (по ситуации):
   - текстовое описание решения + простая структура сценариев;
   - иногда — базовая схема (CJM/диаграмма, структура бота);
   - фиксация ключевых допущений и ограничений.

3. **Дополнительные приложения (по необходимости)**:
   - официальное КП на бланке компании (для тендеров, закупок, крупных корпоратов);
   - черновое структурное ТЗ (как с ботом для разборки автозапчастей) — если нужно показать глубину понимания задачи.

---

8. Результат этапа и переход дальше

Этап считается завершённым, когда:

1. **Клиент понимает предлагаемое решение**, его функциональность и границы проекта.
2. **Согласовано коммерческое предложение**, включая:
   - стоимость работ (поэтапно или общая),
   - модель оплаты (фиксированная цена или T&M),
   - сроки и структура работ,
   - условия поддержки/хостинга/подписки (если применимо).
3. **Согласована модель юридического оформления**:
   - обычное КП (в PDF/Excel);
   - официальное коммерческое предложение по требованию клиента;
   - либо стандартный пакет документов (счёт, договор, спецификация/приложение).

После согласования стоимости и условий — наступает финансово-юридическая часть, которая у разных клиентов работает по-разному.

Стандартная модель запуска (наша базовая)

Для большинства проектов действует привычная, предсказуемая схема:

1. **Подписывается договор / оферта** (если требуется юридически).
2. Клиент вносит **предоплату** — обычно **50% стоимости проекта**.
   Это покрывает:
   - старт аналитики,
   - планирование,
   - подготовку схемы,
   - начало разработческих работ.
3. Мы начинаем этап 2 — **аналитика, планирование, требования, ТЗ и схема**.
4. Далее проект разбивается на этапы/спринты, и **оставшаяся часть стоимости оплачивается по факту выполнения этапов**:
   - каждый крупный блок работ закрывается **актом**;
   - клиент оплачивает соответствующую часть после приёмки этапа.

Этот процесс удобен и прозрачный — он привязан к понятным артефактам: схема, требования, каркас, интеграции, тесты, сдача.

Корпоративная модель (для крупных компаний)

Некоторые корпоративные заказчики работают по "постоплате":

- сначала подписывается договор, ТЗ или спецификация;
- мы **начинаем работы**, потому что у них внутренние регламенты финансовых служб;
- оплата производится **после закрытия этапов или даже всего проекта**;
- иногда требуется выставление счёта **только после факта** выполнения работ.

Мы просто подстраиваемся под эту модель, если это укладывается в рамки проекта и позволяет кэшфлоу.
Структура этапов при этом остаётся прежней — меняется только порядок финансовых документов.

Переход к этапу 2

После закрытия этапа 1 и юридически-финансового старта начинается **Этап 2: Проектирование и планирование**.

Этап 2 — Проектирование и планирование

После того как согласована юридически-финансовая рамка, в которой работаем, начинается **реальная производственная работа** — инженерный этап, на котором формируется архитектура будущего решения и готовится вся база для разработки.

Это фундамент всего проекта: именно здесь создаётся понимание **что**, **зачем** и **как** будет реализовано.

Этот этап объединяет то, что обычно в компаниях разделено: аналитика, проектирование, архитектура, детализация сценариев, технические требования и планирование задач. Фактически это — центр принятия решений и закладка всей логики будущего продукта.

Цели этапа

- Извлечь максимум информации от клиента и стейкхолдеров.
- Превратить сырые пожелания и хаотичные мысли клиента в структурированные требования.
- Построить **архитектуру бота / ассистента / системы**.

- Создать **схему (CJM / логическая диаграмма)** как основу всех сценариев.
- Сформировать **техническое задание (ТЗ)** / **спецификацию** — документ, в котором зафиксирована реальная логика продукта.
- Определить **объём работ**, разбить его на задачи, понять роли, распределить нагрузку.
- Подготовить команду: аналитика, архитектура, разработка, интеграции, маркетинг (если нужно).

Подэтапы Этапа 2

□□ 2.1. Интервьюирование и анализ контекста

Как бы ни выглядел исходный запрос клиента, мы всё равно **извлекаем смысл “из первых уст”**.

Что делаем:

- Проводим серию интервью с:
  - заказчиком,
  - ключевыми сотрудниками,
  - операторами/менеджерами,
  - пользователями (если возможно).
- Выясняем:
  - цели проекта,
  - реальные рабочие процессы,
  - pain points в организации,
  - бизнес-правила,
  - зоны риска,
  - где живут данные,
  - какие системы участвуют.
- Параллельно мы **обучаем клиента**:
  - объясняем, что бот — это не только кнопки и тексты;
  - нужен объясняющий контент;
  - нужны входные и выходные сценарии;
  - нужны fallback-ветки, правильная навигация;
  - важно думать не только о функционале, но и о коммуникации.

Золотое правило:

> *Бот всегда должен содержать объясняющий контент, а не только функциональные переходы.*

Артефакты:

- записи созвонов,
- транскрибации,
- голосовые заметки,
- первичные черновики структуры.

Это — сырьё, которое затем превращается в архитектуру бота.

□□ 2.2. Сбор технической информации

Здесь мы начинаем техническое “раскопывание”.

Мы собираем:

- API-документацию клиента;
- доступы к тестовым средам (если есть);
- модели данных (CRM, ERP, СУЗ, 1С);
- перечень необходимых интеграций;
- сведения о правах доступа и ролях;
- требования по безопасности;
- перечень каналов и ограничения по ним;
- существующие регламенты операторов, скрипты, бизнес-процессы.

Цель:

Сформировать **техническую карту проекта** — где будут интеграции, какая будет логика, какие данные нужно передавать, где система должна принимать решения автоматически.

□□ 2.3. Формирование понимания сценариев и логики

Этот подэтап — сердце инженерии.

Здесь мы делаем:

- сценарии входа (как пользователь впервые попадает в бота);
- коммуникационные ветки;
- сервисные ветки;
- разделение интерфейса на логические зоны;
- определение тона общения;
- правила fallback (что происходит, если пользователь ошибся);
- базовая структура меню;

- типы рассылок и триггеров;
- точки, где нужны интеграции;
- точки, где нужна помощь разработчика.

Это важно: **все эти вещи появляются не из ТЗ клиента, а из проектной экспертизы Metabot**.

□□ 2.4. Создание схемы (CJM / диаграммы логики)

(Ключевой артефакт — критически важный этап)

Золотое правило:

> **Схема — это одновременно документация, проектная канва и средство коммуникации.**

Что мы делаем:

- Строим схему в Sboard / Miro и т.п.
- Наносим всю логику:
  - меню,
  - триггеры,
  - интеграции,
  - сервисные ветки,
  - маршруты,
  - роли пользователей.
- Обозначаем там зоны для разработчиков (интеграционная логика).
- Прописываем переходы и состояния.

Почему это важно:

- Клиент видит проект целиком (даже если не всё понимает).
- Схема — это то, что можно согласовать.
- Схема — это то, что можно актировать.
- Схема экономит месяцы времени на разработке и десятки часов на согласованиях.
- На схеме уже заранее видно архитектуру — где каркас, где модули, где интеграции.

□□ 2.5. Формирование технической спецификации / ТЗ

После интервью, анализа и схемы — формируется **структурированное ТЗ**, уже не сырое.

Документ включает:

- описание целей проекта,
- набор каналов,
- структуру бота,
- все ключевые сценарии,
- логику переходов,
- перечень интеграций,
- параметры данных,
- что должен делать каждый модуль,
- зоны ответственности (аналитик, разработчик, интегратор),
- план работ по этапам.

Это уже **рабочий инженерный документ**, в отличие от концепции, которая была в Этапе 1.

□□ 2.6. Планирование работ, ролей и загрузки

На основе схемы и спецификации:

- распределяем задачи по этапам;
- создаём бэклог проекта;
- определяем, где нужны разработчики;
- оцениваем, сколько займёт каждая часть;
- формируем дорожную карту;
- уточняем риски;
- фиксируем актируемые точки.

Планирование делается в рамках бюджета, согласованного на Этапе 1.

Артефакты Этапа 2

К концу этапа у нас есть:

1. **Схема бота / ассистента** — согласованная и принятая.
2. **Техническое задание (ТЗ)** — структурированное, финальное.
3. **Спецификация / архитектурный документ** — детальная логика.
4. **API-карта точек интеграций** — набор методов, полей, сценариев.
5. **Медиаматериалы интервью** — записи, транскрипции.
6. **План работ / дорожная карта** — этапы, сроки, задачи, роли.

И главное — **понимание всей архитектуры будущего решения и видение конечного результата**.

Результат и переход к Этапу 3

Этап 2 считается завершённым, когда:

- Клиент согласовал схему.
- Клиент согласовал ТЗ / спецификацию.
- Определены все интеграции.
- Понятен объём и план работ.
- Команда знает, что выполнять и в каком порядке.
- Известны сроки.

Далее начинается **Этап 3: Разработка и сборка решения** — то есть изготовление каркаса бота, написание интеграций, сборка модулей и техническая реализация логики.

Этап 3 — Разработка и сборка решения

После того как согласованы схема, ТЗ и спецификация, начинается **производственный цикл** — реализация сценариев, логики, интеграций и функционала согласно проектной архитектуре. На этом этапе схема превращается в рабочий бот, ассистента или комплексную систему на базе Metabot.

Здесь работа переходит от аналитики и проектирования — к **реальному строительству**.

Цели этапа

- Перенести архитектуру и схемы из проектной среды в Metabot.
- Собрать каркас бота: скрипты, команды, ветки, меню, сервисные точки.
- Реализовать интеграции: API, плагины, вебхуки, базы знаний, слоты, контекст.
- Подготовить логику ИИ: ассистентов, промпты, модели, векторные базы.
- Настроить триггеры, события, условия и системные механики.
- Обеспечить тесную работу всех членов команды (аналитиков, архитекторов, разработчиков, интеграторов, AI-специалистов).
- Провести первичное тестирование и отладку.

Ключевой принцип

> **Разработка — это всегда работа на основе согласованной схемы, но допускается небольшое отклонение от схемы, если в процессе реализации найдено более оптимальное решение.**

Схема — это проектная канва, но реальная жизнь иногда требует скорректировать детали.

Подэтапы Этапа 3

□□ 3.1. Подготовка к сборке и распределение задач

Перед началом сборки архитектора бота / ведущий специалист, ответственный за проект:

- открывает финальную схему;
- делает ревью ТЗ, интеграционных точек, логики;
- распределяет задачи по членам команды:
  - архитекторы/сборщики Metabot
  - разработчики (JS / плагины / интеграции)
  - специалисты по ИИ
  - контент-специалисты
  - QA
- создаёт задания в системе (Redmine, ClickUp, Jira и т.д. — в зависимости от проекта);
- привязывает каждую задачу к конкретным частям схемы.

Важное правило:

> **Архитектор бота всегда задаёт контекст разработчику — потому что только архитектор знает, как вся система должна работать в целом.**

□□ 3.2. Перенос схемы в конструктор Metabot

Это основной блок работы специалиста Metabot.

Что делаем:

- Создаём **каркас секций, скриптов и узлов** по схеме.
- Переносим все узлы **в один-в-один структуре**, включая:
  - меню,
  - сценарии входа/выхода,
  - развилки,
  - сервисные цепочки,
  - fallback-ветки,
  - обработчики ошибок,
  - состояния,
  - технические точки.
- Задаём **уникальные коды**, понятные связи, аккуратную структуру.

Двунаправленные ссылки

Это сильная сторона подхода:

- В схеме (Sboard/Miro) в каждом узле ставим ссылку на соответствующий скрипт Metabot.
- В каждом скрипте Metabot добавляем ссылку обратно на узел схемы.

Это создаёт **идеальную навигацию и позволяет любому члену команды быстро найти контекст**.

> **Важно:** схема почти никогда не совпадает 1 в 1 с итоговым ботом — это нормально.

□□ 3.3. Встраивание интеграций

Когда в схеме есть точки, требующие программирования, делается следующее:

- В скриптах Metabot заранее создаются **“заглушки”**:
  - блоки с комментариями
  - отключённый JS-код
  - инструкции разработчику
  - список данных, которые нужно обработать
- Комментарии оформлены так, чтобы разработчик точно понимал:
  - что здесь будет происходить;
  - какие параметры нужно отправить/получить;
  - какие проверки сделать;
  - как обработать ошибку.

Задача архитектора — **передать смысл и контекст**, а не только техническое задание.

Затем:

- разработчик пишет код (плагины, API-вызовы, обработчики, триггеры);
- интеграции добавляются ровно в те точки, которые отмечены на схеме;
- логика связывается с общей архитектурой.

□□ 3.4. Реализация AI-части (если есть)

Если проект включает ИИ, подключаются AI-инженеры.

Возможные задачи:

- создание промптов и ассистентов;
- настройка истории, памяти, контекста;
- подготовка векторных баз знаний;
- настройка RAG-архитектуры;
- классификация интентов/ответов;
- подготовка fallback-поведения моделей;
- интеграция ИИ в общий пайплайн бота.

Эта часть идёт параллельно со сборкой каркаса.

□□ 3.5. Сборка контента и коммуникации

Для многих ботов важна не только логика, но и:

- тон общения;
- текстовые блоки;
- объясняющие сообщения;
- микро-копирайтинг;
- пользовательские инструкции;
- контентная часть ассистента;
- FAQ / статьи / базы знаний.

Это подэтап, который архитекторы и контент-специалисты делают совместно.

Золотой правило:

> **бот должен объяснять, вести, подсказывать — а не просто “давать функции”.**

□□ 3.6. Первичное тестирование и отладка в процессе разработки

Методология Metabot предполагает **не ждать финальной сборки**, а тестировать **по ходу**.

Что делается:

- Специалист проходит пути вручную на своём тестовом аккаунте.
- Проверяет каждый переход, развилку, действие.
- Тестирует интеграции, получая ошибки в чат через режим разработчика.
- Смотрит технические лог-сообщения.
- Исправляет мелкие недочёты на лету.

Этот подход позволяет:

- не копить ошибки;
- не превращать тестирование в катастрофу;
- ускорить этап QA в разы.

Артефакты Этапа 3

1. Полностью собранный каркас бота.
2. Реализованные интеграции (плагины, API, триггеры, JS).
3. Настроенные ассистенты, промпты, AI-слои.
4. Служебные скрипты, обработчики, fallback-логика.
5. Двунаправленные связи между схемой и Metabot.
6. Частично протестированный рабочий прототип.
7. Подготовленный набор задач на финальное тестирование.

Результат Этапа 3 и переход к Этапу 4

Этап 3 завершён, когда:

- весь функционал реализован;
- интеграции работают и выдают данные;
- AI-часть функционирует;
- все основные сценарии собраны;
- каркас бота полностью соответствует схеме (с допустимыми инженерными оптимизациями);

- баги уровня “критично” устранены.

Далее начинается **Этап 4: Тестирование и отладка**, где происходит глубокое проходное тестирование, работа с ошибками, проверка на реальных данных и подготовка релиза.

Этап 4 — Тестирование и отладка

После сборки и реализации всей логики начинается этап, который определяет успех проекта: **комплексное тестирование и отладка**.
Это критически важная часть, поскольку Metabot — платформа гибкая, а проекты часто содержат множество сценариев, условий, интеграций и ИИ-логики.

Тестирование проводится командой до передачи заказчику, чтобы заказчик видел уже **стабильный, логичный и работающий продукт**.

Цели этапа

- Проверить работоспособность всех сценариев, описанных в схеме и ТЗ.
- Убедиться, что логика соответствует архитектуре.
- Проверить корректность интеграций (API, базы данных, CRM, телефония, AI-блоков).
- Найти и устранить ошибки.
- Протестировать работу каждого узла, ветки, триггера, сообщения и перехода.
- Подготовить продукт к финальной демонстрации и приёмке.

Ключевые принципы тестирования Metabot

Алена очень точно подметила:

> **Чтобы качественно протестировать бота — нужно “прожить” его как реальный пользователь.**

Поэтому тестирование всегда включает:

- проверку разных типов пользователей,
- прохождение сценариев с ошибками,
- попытки “сломать” бота,
- тестирование необычных или редких переходов,
- проверку всех “если…” и “вдруг…”.

Подэтапы тестирования

□□ 4.1. Проход всех сценариев вручную

Команда проходит каждый путь:

- от входа до выхода,
- все ветки меню,
- все развилки,
- все обработчики ошибок,
- все сервисные логики,
- все скрытые сценарии,
- все fallback-ситуации.

Используются:

- тестовые лиды,
- тестовые пользователи,
- тестовые учётные записи CRM/ERP,
- тестовые ключи интеграций.

Тестирование проводится **несколько раз**, на разных аккаунтах, чтобы поймать любые расхождения.

🔗 4.2. Проверка интеграций

Интеграционные точки — зона повышенного риска.

Проверяем:

- вызовы API
- отправку данных
- получение данных
- корректность обработки ошибок
- задержки
- отправку email/SMS/webhook уведомлений
- поведение триггеров и условий

Если интеграция нестабильна — проектная команда выясняет:

- проблема в коде?
- проблема в API клиента?
- проблема в данных?
- проблема в среде?

Алена подчёркивала:

> **Каждый интеграционный шаг должен иметь fallback или обработчик ошибки.**

🔗 4.3. Отладка через режим разработчика

Для разработчиков и интеграторов доступен специальный режим:

- включается режим отладки,
- все технические ошибки отправляются в мессенджер (TG/VK/WhatsApp),
- получаем trace:
  - какое действие вызвало ошибку,
  - какая переменная пустая,
  - какой API ответ пришёл,
  - какой параметр не найден.

Этот режим — основной инструмент при разработке и тестировании.

□□ 4.4. Работа с логами (если доступно)

Важно:
в **облачной версии Metabot логирование ограничено для обычных пользователей**.

Доступ к логам есть:

- у специалистов на **выделенном сервере**,
- на **коробочной версии**,
- или у команды Metabot (через поддержку).

Что можно делать на выделенном сервере:

- смотреть системные логи,
- видеть критические ошибки,
- видеть очередь задач,
- остановить или перезапустить сервисы,
- заморозить бота, чтобы избежать некорректной рассылки/циклов.

Алена говорила:

> **“ Умеешь пользоваться логами — решаешь 90% проблем за минуты.**

На общем облаке есть ограничения:

- нет прямого доступа к логам,
- нельзя остановить платформу вручную.

Обходные пути:

- временно сменить токен бота в Telegram/VK — бот перестанет отвечать;
- отключить интеграцию на панели;
- обратиться в поддержку Metabot;
- вынести рискованные узлы в закрытые секции.

□□ 4.5. Проверка AI-логики

Если проект включает ИИ, тестирование усложняется.

Проверяем:

- корректность ответа ассистента,
- устойчивость к ошибочным вводам,
- работу памяти/контекста,
- работу векторных поисков,
- определение интентов,
- fallback-механику AI,
- скорость ответа.

Проводим несколько “прохождений”:

1. идеально корректный пользователь,
2. “средний пользователь”,
3. “хаотичный пользователь, вводящий чушь”,
4. стресс-тесты — 10–20 запросов подряд.

□□ 4.6. Финальная полировка

Когда основные ошибки устранены:

- выравниваем тексты,
- проверяем UX-логику,
- обеспечиваем единообразный стиль,
- проверяем скорость прохождения сценариев,
- проверяем корректность меток, тегов, логирования, аналитики.

Это этап, который часто недооценивают, но он сильно влияет на впечатление заказчика.

Передача заказчику для приёмки

После внутреннего тестирования:

- заказчику даётся доступ к боту;
- он сам проходит ключевые сценарии;
- фиксирует пожелания, правки, уточнения;
- команда вносит разумные доработки;
- проводится финальный прогон.

Алена говорила:

> **“Это нормальная практика — заказчик увидит что-то новое, когда “подержит бота в руках”.**

Главное — чтобы правки не ломали архитектуру.

Артефакты Этапа 4

1. Полностью протестированный бот/ассистент.
2. Список найденных и устранённых ошибок.
3. Лог прохождения тестов.
4. Отчёт о тестировании (по необходимости).
5. Готовый к приёмке прототип без критических багов.

Результат Этапа 4 и переход к Этапу 5

Этап 4 завершён, когда:

- все сценарии работают стабильно;
- интеграции отрабатывают без ошибок;
- AI-логика корректна;
- критических и средних багов нет;
- проект согласован командой;
- заказчик прошёл бота и дал подтверждение.

После этого начинается:

Этап 5 — Сдача и ввод в эксплуатацию

Этап 5 — это момент, когда проект переходит из инженерной стадии в реальную работу. Здесь команда несёт максимальную ответственность: решение впервые сталкивается с живыми пользователями, реальными данными и настоящими сценариями бизнеса.

Цель этапа — **безопасно и предсказуемо запустить проект**, обеспечить его стабильность в первые дни и недели, и провести заказчика через процесс приёмки.

Цели этапа

- Подготовить проект к запуску — инфраструктурно, технически и организационно.
- Передать заказчику работающее решение в полной готовности.
- Настроить мониторинг и смотреть первые реакции пользователей.
- Обеспечить стабильную работу бота/ассистента при реальной нагрузке.
- Быстро реагировать на первые инциденты, править мелкие недочёты.
- Дать клиенту инструкции, обучение и понимание, как жить дальше с системой.

Подэтапы Этапа 5

□□ 5.1. Финальная проверка перед релизом

Перед передачей клиенту команда проводит:

- финальное тестирование всех ключевых сценариев;
- проверку интеграций;
- тест входных/выходных сценариев;
- проверку обработчиков ошибок;
- проверку правильности логирования и аналитики;
- выравнивание коммуникационного стиля;
- корректность fallback-веток.

Это тот самый «последний прогон», когда команда самолично убеждается, что решение стабильно.

□□ 5.2. Передача проекта заказчику

Заказчик получает:

- доступ к боту, ассистенту или системе;
- доступ к админ-панели (если предусмотрено);
- доступ к документации (схема, ТЗ, пояснения, инструкции);
- видео-/текстовый walkthrough по ключевым сценариям;
- пояснение архитектуры (если требуется).

Клиент самостоятельно проходит сценарии:

- кликает кнопки;
- проверяет ветки;
- тестирует интеграции;
- пытается воспроизвести реальные ситуации.

Нормально, что заказчик замечает новые нюансы.
Это естественный этап: когда продукт впервые попадает в руки клиента, появляются мелкие уточнения, которые невозможно увидеть на схеме.

Команда вносит правки в разумных пределах.

□□ 5.3. Подготовка к запуску и выбор стратегии релиза

Стратегия зависит от типа проекта.

1) **Пилотный запуск (ограниченный доступ)**

Используется, если:

- функционал новый,

- важно собрать UX-обратную связь,
- есть риски высоких нагрузок,
- нужен быстрый цикл корректировок.

Проводится:

- запуск на малую группу пользователей (служба поддержки, операторы, часть отдела);
- сбор обратной связи;
- исправление найденного;
- постепенное расширение охвата.

2) **Мягкий старт (soft launch)**

Постепенный запуск:

- канал за каналом,
- источник трафика за источником,
- аудитория за аудиторией.

Используется для маркетинговых и коммуникационных проектов.

3) **Полный запуск (full launch)**

Используется:

- если бот встроен в основной канал бизнеса (сайт, Telegram, WhatsApp, VK),
- если нет возможности запускать постепенно,
- если решение заменяет старую систему.

Запуск производится по согласованному времени, чаще — в рабочие часы с оперативной поддержкой.

□□ 5.4. Мониторинг первых дней (критическая фаза)

Первые дни после запуска требуют особого внимания.

Команда:

- активно мониторит диалоги,
- проверяет поведение интеграций,
- отслеживает ошибки,
- измеряет скорость ответов,
- контролирует нагрузку,

- оперативно вносит точечные правки.

Если используются AI-компоненты — наблюдение ещё важнее:

- проверяем корректность ответов;
- отслеживаем «дрейф» поведения моделей;
- корректируем промпты, инструкции и fallback.

Система “должна не просто работать, а вести себя предсказуемо”.

□□ 5.5. Сбор аналитики

Сразу после запуска начинается сбор фактических данных:

- количество входов,
- завершение сценариев,
- процент ошибок,
- точки отказа,
- время ответа,
- конверсия по веткам,
- тепловые карты переходов,
- активность пользователей.

> **При анализе всегда найдутся места для улучшений — это нормально.**

Эти данные переходят в дальнейшую поддержку и развитие.

□□ 5.6. Инструкции и обучение

Команда обучает заказчика:

- как пользоваться системой,
- как смотреть статистику,
- как управлять контентом,
- как включать/отключать каналы,
- как пользоваться админкой,
- как работать со сбором данных,
- как реагировать на инциденты.

Обучение может быть:

- видео,

- звонок,
- демонстрация,
- мини-мануал.

□□ 5.7. Финальная приёмка работ

Когда заказчик проходит сценарии, тестирует систему и подтверждает, что всё работает:

- оформляется акт выполненных работ;
- заказчик принимает проект;
- начинается следующая стадия — поддержка и развитие.

Артефакты Этапа 5

1. Рабочий, запущенный бот/ассистент.
2. Инструкции и документация.
3. Список правок, внесённых после приёмки.
4. Аналитика первых дней.
5. Подтверждение заказчика о готовности решения.

Результат Этапа 5 и переход к Этапу 6

Этап 5 завершён, когда:

- продукт запущен,
- система работает стабильно,
- заказчик принял решение,
- собрана первичная аналитика,
- выполнены корректировки,
- продукт перешёл в штатную эксплуатацию.

Далее начинается **Этап 6. Поддержка и развитие проекта** — долгосрочная работа, в которой продукт живёт, улучшается и адаптируется под новые задачи клиента.

Этап 6 — Поддержка и развитие проекта

Когда проект дошёл до этого этапа, это означает, что основная задача, ради которой он запускался, **выполнена**.
Решение создано, протестировано, принято заказчиком и запущено в эксплуатацию.
Дальше начинается фаза, в которой бот или ассистент становится частью живой экосистемы бизнеса, а Metabot частью инфраструктуры заказчика: он работает, взаимодействует с пользователями, собирает данные, помогает процессам — и требует корректного сопровождения.

Поддержка и развитие — это **долгосрочный цикл работы**, в котором:

- устраняются ошибки и инциденты;
- вносятся мелкие правки;
- добавляются улучшения;
- обрабатываются новые бизнес-требования;
- строится roadmap развития;
- платформа может мигрировать в контур клиента;
- обеспечивается стабильная работа всей инфраструктуры.

Цели этапа

- Обеспечить стабильность и предсказуемость работы решения.
- Быстро реагировать на ошибки, инциденты и изменения на стороне клиента.
- Вносить мелкие правки и улучшения в рамках поддержки.
- Реализовывать новые бизнес-требования и развивать функционал.
- При необходимости переносить платформу или бота в инфраструктуру клиента.
- Управлять ресурсами команды и давать заказчику прозрачный SLA.

Подэтапы Этапа 6

□□ 6.1. Вход в режим поддержки

После запуска команда и заказчик определяют формат дальнейшей работы:

Вариант 1 — **Договор поддержки (SLA + развитие)**

Наиболее удобный вариант, включающий:

- резерв часов команды;
- фиксированную доступность специалистов;
- приоритет реакции;
- возможность оперативно исправлять ошибки;
- мелкие доработки без отдельного бюджетирования;
- доступ к менеджеру проекта или продуктовой группе;
- регулярные рабочие совещаний с заказчиком.

В этом режиме заказчику не нужно “перезаказывать” маленькие задачи — они выполняются автоматически в рамках пакета поддержки.

Вариант 2 — **Поддержка без резерва (on demand)**

Если заказчик не хочет резервировать время:

- задачи принимаются по запросу;
- работа оценивается отдельно;
- ставки выше, т.к. специалисты отвлекаются от других проектов;
- приоритет ниже, чем у проектов с SLA.

Работа без резерва — **самый тяжёлый режим для клиента**, т.к. ждать приходится дольше и стоит дороже.

□□ 6.2. Исправление ошибок и мелкие корректировки

В поддержку входят:

- исправление багов;
- корректировка текстов, веток, интерфейсов;
- адаптация логики после изменений у клиента;
- обновление интеграций после изменений API;
- поддержка AI-части: промпты, модели, fallback;
- мелкие улучшения функционала.

> **Поддержка — это не разработка, это сопровождение жизни продукта. Мелкие правки и шлифовка — обычный процесс.**

🔄 6.3. Реализация новых бизнес-требований (развитие)

Когда задачи выходят за рамки SLA или требуют серьёзного объёма, запускается процесс **Бизнес-Требований (BT/BRD)**:

1. Формируется бизнес-требование от заказчика.
2. Команда оценивает его:
   - объём работ,
   - сроки,
   - стоимость.
3. Заказчик одобряет бюджет.
4. Создаётся мини-этап проектирования:
   - схема,
   - ТЗ,
   - интеграции.
5. Требование входит в очередь развития и выполняется как мини-проект.

Таким образом:

> **Проект живёт циклами: BT → проектирование → реализация → тестирование → запуск.**

Это позволяет развивать продукт постепенно, прозрачными итерациями, без хаоса.

🔄 6.4. Управление ресурсами команды

В рамках SLA обычно резервируются:

- часы проектной группы;
- часы технической группы;
- интеграторы;
- разработчики;
- AI-специалисты.

Заказчик может использовать эти ресурсы гибко.

Если ресурсы не резервируются заранее — они тарифицируются выше.

□□ 6.5. Поддержка инфраструктуры и серверов

Многие заказчики работают:

- на общем облаке Metabot,
- на выделенных серверах,
- на коробочной версии внутри собственного контура.

На этом этапе может потребоваться:

- развернуть коробочную версию у заказчика (часто в медицине и крупных компаниях);
- перейти с облака Metabot на выделенный сервер;
- разделить каналы по бизнес-единицам;
- перенести базу данных или интеграционные модули.

Эти работы могут идти параллельно либо быть частью развития и оцениваются отдельно.

□□ 6.6. Мониторинг и реакция на инциденты (troubleshooting)

✔ Главное правило — **не паниковать**.

Любая ситуация решается, потому что:

- мессенджер всегда сохраняет связь с пользователем;
- можно отправить исправляющее сообщение;
- можно отозвать сообщение (Telegram позволяет);
- можно заморозить бота и прекратить отправку;
- можно дослать информацию позже;
- можно вручную повторно отправить уведомления.

Возможные ситуации:

- сломалась интеграция;
- зависла внешняя система;

- пришли неверные данные;
- переполнен лимит API;
- не отработала логика AI;
- “повело” текст ассистента.

Что делать:

1. **Остановить некорректное поведение**
   - отключить бота на платформе;
   - временно поменять токен в Telegram/VK;
   - выключить триггер;
   - на выделенном сервере — остановить сервис.
2. **Определить масштаб**
   - сколько пользователей затронуло;
   - в какой момент произошло;
   - какой модуль виноват.
3. **Собрать данные**
   - логи (на выделенном сервере);
   - техническое сообщение режима разработчика;
   - поведение интеграции.
4. **Выработать план действий**
   - переслать корректирующее сообщение;
   - отозвать ошибочное;
   - обработать негатив пользователей, например, извиниться и объяснить ситуацию;
   - починить код или интеграцию;
   - протестировать и вернуть в строй.

Email-оповещения и наблюдение

В некоторых проектах у вас настроены:

- email-уведомления на критических ошибках;
- уведомления о падении API;
- отчёты о сбоях интеграций;
- тревоги по работам cron/бота.

Это ускоряет реакцию и позволяет чинить за минуты.

□□ 6.7. Аналитика и рост продукта

После запуска и поддержки начинается “взрослая жизнь” бота:

- анализируются статистики;

- находятся слабые места;
- улучшаются воронки;
- строятся новые каналы;
- усиливаются AI-модули;
- подключаются новые интеграции;
- расширяется логика;
- накапливаются бизнес-требования;

Таким образом:

> **поддержка → аналитика → улучшения → развитие → новый цикл проекта.**

Артефакты Этапа 6

1. Договор SLA или on-demand формат.
2. План резервирования ресурсов.
3. Журнал выполненной поддержки.
4. Реализованные бизнес-требования.
5. Новые схемы, ТЗ, мини-проекты.
6. Инцидент-репорты и инструкции по устранению.
7. Dashboard аналитики/BI (по необходимости).
8. Итоговые версии систем и модулей.

Завершение Этапа 6

Этап 6 — это по сути **не конец проекта**, а переход в режим непрерывного развития. Продукт становится частью бизнеса, а бизнес — частью следующего витка улучшений.

Именно на этом этапе происходят:

- масштабирование,
- запуск новых сценариев,
- подключение AI-ассистентов,
- расширение на новые каналы,
- перенос в контур клиента,
- рост пользовательской базы
  — и формирование долгосрочного партнерства.

Версия #8
Artem Garashko создал 2025-11-28 12:30:05 UTC

Artem Garashko обновил 2025-11-29 10:37:45 UTC