

ИИ-агенты и Agent Skills

Экскурсия по AI: что есть что

Редакция	r0002
Дата сборки	2026-10-14
starter	0.17.2

Техническая редакция: пилот не проводился, живых свидетельств модели нет (education_status: pre-pilot, publication_ready: false).

Состав

Часть	Версия
Экскурсия по AI: что есть что	2.2

Оглавление

Экскурсия по AI: что есть что	3
С чего всё началось	3
После этой главы ты сможешь	4
Урок Э1. Модель и два входа к ней.....	4
Урок Э2. Агент и его упряжка	9
Зачем это тебе	15
Проверь себя.....	15
Домашняя работа — необязательно.....	16
Словарь главы.....	16
Возвращаемся к первоначальной сцене	17
Мост к главе 1 учебника.....	17

Экскурсия по AI: что есть что

Вводная глава

С чего всё началось

До школьного Фестиваля науки остаётся месяц. Для подготовки сформировали проектное бюро: несколько человек, которые следят, чтобы к сроку всё было готово. Ты в этом бюро тоже состоишь. Аня — координатор, Марк пришёл из лабораторной группы.

За первую же неделю в общей папке бюро скопилось двенадцать заметок с собраний: про афишу, про оборудование, про кабинеты и про многое другое. Отыскать в них нужное уже непросто, и Аня решает составить оглавление.

Она садится за школьный компьютер и открывает чат, которым в школе разрешено пользоваться: окно на сайте, куда вопрос пишут обычными словами. Буквы «ИИ» значат «искусственный интеллект»; пока будем так называть программу, которой можно задать вопрос на обычном языке. Аня просит ИИ собрать оглавление по заметкам из папки. Программа отвечает, что папки бюро она не видит, и предлагает на выбор: либо Аня пришлёт ей тексты заметок, либо составит оглавление сама, а программа подскажет как.

Марк тем временем сидит в кабинете информатики. Учитель показывает ему программу-помощника, запущенную прямо в папке бюро, и Марк диктует ей ту же просьбу — составить оглавление двенадцати заметок с собраний. Помощник сообщает, что нашёл двенадцать файлов, и спрашивает, можно ли создать в папке новый файл с оглавлением. Марк отвечает «да». Через минуту оглавление уже лежит в папке.

Вечером они сравнивают, что у кого вышло.

— У тебя ИИ сломан, — говорит Марк. — Мой сам всё сделал.

— Ничего он не сломан, — отвечает Аня. — Название системы на экране у нас было одно и то же. Только, похоже, под этим названием мы разговаривали с разными вещами. Вот ты мне советуешь: «спроси у ИИ». У какого? И через что?

Название и правда одно. А вели себя системы по-разному: Аня получила совет, Марк — готовый файл.

Вопрос-Прогноз. Название на экране одно, а поведение разное. Как ты думаешь, что у Ани и у Марка было одинаковым, а что разным? Запиши свою догадку и придумай, как её проверить.

Чтобы ответить, придётся обойти настоящие системы и посмотреть, из каких частей они состоят. Строить мы пока ничего не будем: только смотреть и называть увиденное. Потому глава и называется экскурсией.

Главный вопрос главы. Что есть что: модель, приложение вокруг неё, API, агент, инструменты, Agent Skill, MCP? И с чем из этого правильно иметь дело в каждой конкретной ситуации?

Незнакомых слов в этом вопросе много. Заучивать их сейчас не нужно: к концу экскурсии каждое займёт на карте своё место.

После этой главы ты сможешь

- различать модель и приложение, через которое с ней работают, — исходя из того, что делает приложение;
- объяснять, чем обращение через чат отличается от обращения через API: где хранится история разговора и кто пишет программу, которая с ней работает;
- сравнивать модели на своём наборе задач и объяснять, почему «лучшей модели вообще» не бывает;
- сравнивать три входа к модели — чат, агент для кода и программу через API — по трём вопросам: что система может изменить, кто подтверждает действие, где записаны шаги;
- показывать на схеме папки агента файл инструкций, файл с записанной процедурой и файл подключений, а также называть роль каждого;
- показывать на карте (про неё чуть ниже), где в показах были отдельные действие программы, подключённый внешний сервис и записанная процедура;
- аргументировать выбор входа — чат, агент или API — для твоей конкретной ситуации.

Урок Э1. Модель и два входа к ней

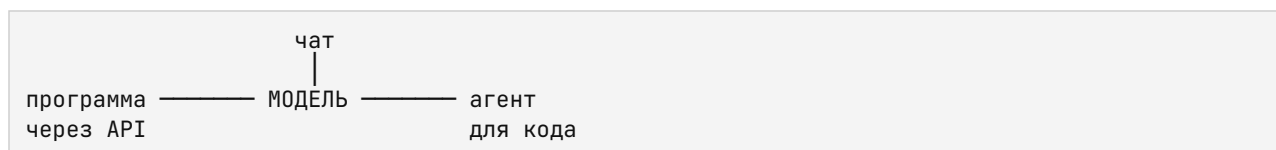
Как читать и анализировать показы

В этой главе восемь показов. **Показ — короткая демонстрация: на уроке её проводит учитель или включают видеозапись, а в книге она описана словами.** Перед показом запиши прогноз: что, по-твоему, сейчас произойдёт. После показа запиши, что случилось на самом деле, и вывод в одну фразу.

Но прежде три предупреждения.

1. Примеры разговоров, файлов и программ в этой главе составлены авторами. Блок, который начинается строкой `# пример`, — иллюстрация: так подобные системы обычно себя ведут, но это не запись настоящего разговора.
2. Показы проводит взрослый, под своими учётными записями (аккаунтами) и по правилам сервисов. Тебе не придётся нигде регистрироваться, ничего устанавливать и ни за что платить. Если у тебя есть аккаунт в ИИ-системе школы, показы 1–3 можешь повторить сам.
3. Системы в тексте мы называем по их роли: «чат», «агент для кода», «программа через API». Что скрывается за каждой ролью, ты увидишь в показах. Названия продуктов собраны в справке ниже и верны только на её дату: продукты меняются быстро, а роли остаются прежними.

Перенеси в тетрадь карту. В центре стоит модель — так называют часть, которая составляет ответы; подробнее о ней сказано в показе 2. Вокруг модели три пустые рамки. После каждого показа дописывай на карту то, что узнал.



Справка: названия на 4 октября 2026 года. Роль чата играют, например, чат DeepSeek и чат Claude; роль агента для кода — Claude Code; вход для программ — API Claude; роль внешнего сервиса — GitHub, сайт, где хранят проекты программ. Примеры взяты у двух

разработчиков, и это не список лучших: у других разработчиков названия свои, а роли те же. К тому времени, как ты читаешь эту книгу, названия и возможности вполне могли измениться.

Показ 1. Куда мы попали?

Вопрос-Прогноз. Учитель набирает в поисковике название известного чата и открывает первую же ссылку. Что окажется перед нами: «сам ИИ» или что-то другое?

В поисковой выдаче несколько строк с похожими названиями. Смотреть прежде всего надо на адрес сайта — строку в верхней части браузера. У чата есть официальный сайт, и его адрес называет сам разработчик. Рядом в выдаче нередко стоят сайты с похожими адресами: посредники, которые сами обращаются к чужой модели, или просто двойники. По названию на странице их не отличишь, отличают их по адресу. Похожий адрес ещё не значит обман, но это уже другой сайт, у которого другой владелец и другие правила.

Открываем официальный сайт и рассматриваем окно. В нём есть поле ввода и кнопка отправки. Слева тянется список прошлых разговоров. Возле поля стоят переключатели, у разных чатов свои: например, «включить поиск в интернете» или «выбрать модель». Внизу ссылка на условия использования; там написано, кому разрешено пользоваться сервисом, и в том числе с какого возраста.

Всё перечисленное в предыдущем абзаце делает самая обычная программа: рисует окно, хранит разговоры, помнит, кто вошёл. А на вопрос отвечает другая часть, спрятанная глубже.

Понятие. Приложение — программа, которой пользуется человек: принимает вопрос, обращается к модели и показывает ответ; чат в браузере — её пример. Строгое определение ты найдёшь в главе 1.

Вывод. Перед нами приложение — сайт-программа со своим адресом и своими правилами. «Самого ИИ» мы пока не видели.

На карте в рамку «чат» допиши: окно, список разговоров, переключатели, правила.

Показ 2. Что внутри

Вопрос-Прогноз. Чат называет сегодняшнюю погоду. Откуда он её знает?

Внутри приложения работает та часть, которая и составляет ответы, — языковая модель. Её получили, обучив на огромном количестве текстов, собранных до определённой даты; всё, что случилось позже, само в неё не попадает. Работает модель не на твоём компьютере и не в браузере, а на серверах разработчика. **Сервер** — компьютер, который по сети интернет отвечает на запросы других программ.

Понятие. Модель — часть системы, которая по переданным ей сведениям составляет ответ; сама она ничего не открывает — ни сайтов, ни файлов. Строгое определение ты найдёшь в главе 1.

Где же кончается модель и начинается приложение? Это видно на трёх опытах.

Опыт 1. Свежие сведения. Поиск в чате выключен. Спрашиваем: «Какая сегодня погода в нашем городе?» Сегодняшнего дня в текстах, на которых учили модель, конечно, нет, и честный ответ звучит примерно так: «У меня нет доступа к текущим данным».

Теперь включаем поиск и повторяем вопрос. В ответе появляются и погода, и ссылки на страницы. Модель при этом осталась прежней. Изменилось приложение: оно искало в интернете и добавило найденное к тому, что передало модели.

Опыт 2. Один вопрос дважды. В двух новых разговорах просим одно и то же: «Придумай название из двух слов для школьного кружка робототехники». Скорее всего, ответы выйдут разными. Готового ответа модель из хранилища не достает: она каждый раз составляет его заново, и выбор слов может получиться другим. Запомни это наблюдение; в конце главы оно объяснит одно неожиданное решение.

Опыт 3. Новый разговор. В одном разговоре называем кодовое слово «ласточка» и тут же спрашиваем его. Чат отвечает верно. Открываем новый разговор: «Какое кодовое слово я называл?» Если у приложения выключена память между разговорами, этого слова оно не знает.

Почему же в первом разговоре знало? Приложение каждый раз отправляет модели не один только новый вопрос, но и все прошлые реплики этого разговора. Модель «помнит» разговор ровно в той мере, в какой ей его прислали. Память между разговорами, если она есть, — тоже забота приложения: это оно сохранило запись и подложило её в новый разговор.

Понятие. Контекст — всё, что передано модели в данном конкретном обращении: вопрос и то, что программа добавила к нему, например прошлые реплики. Строгое определение ты найдёшь в главе 1.

Вывод. В наших показах модель получает текст и выдаёт текст. Поиск, история разговора и память между разговорами — работа приложения.

Проверь себя на ходу. Чат правильно назвал сегодняшнюю погоду. Что нужно выяснить, чтобы сказать, чья это заслуга, модели или приложения?

На карте рядом с моделью допиши: получает сведения — составляет ответ. В рамку «чат»: присылает модели историю разговора; умеет искать.

Показ 3. Модели разные

Вопрос-Прогноз. Одноклассник спрашивает: «Какая нейросеть самая лучшая?» Что ты ему ответишь и о чём спросишь сам?

В переключателе чата не одно название, а несколько: модель не одна на всех. У каждого разработчика своя **семья моделей** — так мы назовём модели одного разработчика, выпущенные под общим именем. В семье есть старшие и младшие, новые и прежние.

Понятие. Версия модели — имя и номер выпуска модели, по которым можно сказать, к какому именно варианту модели обращались. Но не путай её с **версией собственной процедуры**, о которой пойдёт речь в главе 13.

Чем же модели отличаются одна от другой? Разработчики описывают это так. Старшие модели семьи справляются с более трудными задачами, зато отвечают медленнее и обходятся дороже; младшие быстрее и дешевле.

Новые выпуски обучены на более свежих текстах. Модели различаются и тем, сколько текста могут учесть за одно обращение. Некоторые работают не только с текстом, но и с изображениями. У некоторых есть режим, в котором модель дольше готовит ответ. Но всё это лишь слова разработчика о собственной модели, и подойдёт ли она тебе, из них не узнаешь.

Проверить это можно самому.

1. Возьми шесть-восемь своих задач, ответы на которые тебе известны.
2. До запуска запиши критерии: что считать верным ответом, нужен ли ответ в заданном виде, важна ли скорость.
3. Задай каждой модели одни и те же задачи одними и теми же словами, каждую — в новом разговоре. Поиск и прочие возможности приложения выключи: иначе ты будешь сравнивать уже приложения.
4. Заполни таблицу.
5. Вывод делай только о том, что проверил.

Таблица может выглядеть, например, так. Результаты в ней придуманы для примера, настоящие модели тут ни при чём.

Задача	Модель А	Модель Б	Модель В
Перевести 72 км/ч в м/с	верно, 2 с	верно, 3 с	верно, 20 с
Перемножить два пятизначных числа	неверно, 2 с	неверно, 3 с	верно, 40 с
Сосчитать буквы «о» в длинной фразе	неверно, 2 с	верно, 3 с	верно, 35 с
Ответить одним числом, без пояснений	да	нет	да

Что можно прочесть по такой таблице? Для устного счёта хватит быстрой модели А. Где ошибка обходится дорого, нужна модель В, но её ответа приходится ждать: в нашей таблице от 20 до 40 секунд вместо 2–3. Модель Б не послушалась просьбы отвечать одним числом. Какая же из них лучше? Сам вопрос поставлен неверно. Лучше — для чего?

И ещё три оговорки, без которых вывод будет нечестным. Задач мало. Каждая задана лишь один раз, а по опыту 2 ты знаешь, что от раза к разу ответ меняется. И всё это верно только на одну дату: завтра выйдет новая версия. Как проверять всерьёз — на заранее выбранном наборе случаев и не по одному запуску, — разберём в главе 11. Рейтинги из интернета устроены точно так же, только задачи в них чужие.

Вывод. «Лучше» бывает только «лучше для моих задач на эту дату».

На карте рядом с моделью допиши: у модели есть имя и версия.

Показ 4. Второй вход: API

Вопрос-Прогноз. Программист хочет, чтобы его собственная программа получала ответы модели. Должна ли программа открывать браузер и печатать в окне чата?

Не должна. Окно чата сделано для человека, а для программ у разработчика модели есть другой вход.

Понятие. API — вход для программ: способ, которым одна программа обращается к другой по заранее описанным правилам и получает ответ в виде данных; здесь используется обращение к модели без окна чата.

В общих чертах этот вход устроен у разных разработчиков одинаково. Программа отправляет на сервер запрос из четырёх частей: адрес сервиса, ключ, название модели и список сообщений. Обратно приходят данные: текст ответа и служебные сведения — какая модель отвечала и сколько текста было на входе и на выходе. По этому объёму и считают плату. За чат обычно

платят подпиской или пользуются им бесплатно, с ограничениями; обращения же через API оплачиваются по объёму текста.

Понятие. Ключ API — секретная строка, по которой сервис узнаёт, чья программа к нему обращается. Строгое определение ты найдёшь в главе 10, где такие значения называются секретами.

Ключ похож на пароль: кто его знает, тот и тратит деньги владельца. Поэтому ключ не пишут в тексте программы и не кладут в общую папку; программа берёт его из настроек компьютера.

Ниже показано, как выглядит такое обращение на языке Python (на нём написаны и программы нашего курса) с библиотекой одного из разработчиков. **Библиотека** — готовый набор программ, который подключают к своей. Пример верен на октябрь 2026 года. Разбирать его строчку за строчкой не нужно: найди в нём только части запроса.

```
# пример
import anthropic

client = anthropic.Anthropic() # ключ берётся из настроек компьютера, в коде его нет

answer = client.messages.create(
    model="имя-модели",          # название и версия — из списка разработчика
    max_tokens=200,             # предел длины ответа
    messages=[
        {"role": "user", "content": "Какое кодовое слово я называл?"},
    ],
)
print(answer.content[0].text)
```

Название модели и список сообщений найти нетрудно. В фигурных скобках записано одно сообщение: кто говорит и что сказано. Ключа в коде нет, а адрес известен библиотеке.

Повторим теперь опыт с кодовым словом, только без чата. Первое обращение: «Кодовое слово нашей экскурсии — „ласточка“. Ответь одним словом: принято». Модель отвечает: «Принято». Второе обращение — та самая программа, что напечатана выше; в ней один лишь вопрос про кодовое слово. Модель ответит, что никакого слова не знает: в этом обращении о ласточке нет ни слова. Сервис из нашего примера разговора не хранит, каждое обращение для него отдельное. Чат, как мы помним, хранит все сообщения и ответы модели в каждом своём разговоре.

Чтобы модель назвала слово, программа должна прислать весь разговор сама, тремя сообщениями подряд: первую реплику человека, прошлый ответ модели и новый вопрос. Вспомни опыт 3: там историю присылало приложение-чат, и ты этого даже не замечал. Здесь её присылает программа. Или не присылает, если автор об этом не позаботился.

Понятие. Обязка — та часть программы, которая стоит между человеком и моделью: собирает то, что будет передано модели, хранит разговор, выполняет действия и следит за ограничениями. Строгое определение ты найдёшь в главе 3.

В чате обязку написал разработчик чата. При обращении через API готовой обязки нет, и всё, что чат делал вокруг модели, теперь пишет автор программы.

Замечание. Если ключ неверный, сервис вернёт сообщение об ошибке, и до модели запрос вовсе не дойдёт.

Проверь себя на ходу. Программа отправила модели только последний вопрос, без прошлых реплик. Что модель знает о начале разговора?

Вывод. Чат — готовая обвязка, которую написали другие. Через API обвязку пишешь сам, и даже историю разговора присылает твоя программа.

На карте в рамку «программа через API» допиши: ключ; запрос и ответ — данные; историю присылает программа.

Итог урока Э1

	Чат	Программа через API
Кто обращается к модели	приложение разработчика чата	твоя программа
Где история разговора	у приложения; оно само присылает её модели	там, где её сохранит программа; присылать должна она
Кто добавляет поиск и другие возможности	разработчик чата	автор программы

Мини-проверка. Разнеси шесть наблюдений по двум столбцам: «сделала модель» и «сделало приложение или программа».

1. Составлен текст ответа.
2. Сохранён список прошлых разговоров.
3. Найдена страница в интернете.
4. Показан переключатель режимов.
5. Модели присланы прошлые реплики.
6. Посчитан объём текста для оплаты.

Ответ. 1 — модель; 2–6 — приложение или программа. Допустимое уточнение к пункту 3: найденную страницу приложение передаёт модели, и ответ по ней составляет уже модель.

Практика и выходной билет урока Э1

Трек А. Заполни таблицу наблюдений для показов 1–4: твой прогноз; что произошло; чья это работа, модели или приложения; вывод в одну фразу. **Результат:** четыре строки таблицы и карта с двумя входами.

Трек В — необязательный. Вместе с учителем, в чате, которым школа разрешает пользоваться, повторите показы 1–3: рассмотрите окно, проведите три опыта, а если моделей в чате несколько, сравните две из них на трёх своих задачах. Личных данных не вводите. Результат запишите в ту же таблицу с пометкой «наш опыт» и сравните с книгой. Совпадёт, скорее всего, не всё, и разобраться, что именно разошлось и почему, тоже полезно.

Выходной билет урока Э1. Закончи фразу: «В чате историю разговора хранит и присылает модели..., а при обращении через API — ...».

Урок Э2. Агент и его упряжка

Вернись к прогнозу, который ты записал после первой сцены. У Ани был чат, то есть приложение, которое присылает модели разговор и показывает ответ. Что было у Марка, мы пока не знаем. С этого и начнём.

Показ 5. Чат советует — агент делает

Вопрос-Прогноз. Одно и то же задание получают чат и программа, запущенная прямо в папке с заметками. Что будет разным: ответ или то, что произойдёт вокруг ответа?

Задание такое: «В папке notes лежат 12 заметок с собраний групп. Собери по ним оглавление: по одной строке на заметку — имя файла и заголовок. Сохрани оглавление в файл».

Сначала чат. Его ответ выглядит примерно так:

```
# пример
У меня нет доступа к папкам на вашем компьютере. Пришлите тексты
заметок — я составлю оглавление. Или подскажу, как собрать его самостоятельно.
```

Именно такой ответ и получила Аня. Чат не сломан: у его приложения нет действия «прочитать файл на компьютере Ани», и модели нечем это сделать. В папке ничего не изменилось.

Теперь программа, которую показывали Марку. Её запускают в самой папке, через **командную строку** — окно, где команды программам набирают текстом. Такую программу называют агентом для кода: чаще всего ей поручают работу с файлами программ. Получив то же задание, она действует по шагам:

```
# пример
Просматриваю папку notes: найдено 12 файлов.
Читаю note01.md ... note12.md.
Хочу создать файл notes/INDEX.md. Разрешить? [да / нет]
> да
Файл notes/INDEX.md создан: 12 строк.
```

Посмотрим внимательно, что здесь произошло. Модель участвовала в выборе шагов: сначала посмотреть список файлов, потом прочитать их, потом создать новый. Но сама не выполнила ни одного шага. Папку открывала программа, окружающая модель; файлы читала тоже она и каждый раз показывала модели, что получилось. А перед тем как что-то изменить, программа остановилась и спросила человека.

Понятие. Агент — система, которая не только отвечает, но и действует: модель участвует в выборе шага, программа выполняет его и показывает модели, что получилось. Строгое определение ты найдёшь в главе 1; как шаги складываются в цикл — в главе 2.

Понятие. Инструмент — отдельное действие, которое программа умеет выполнить: прочитать файл, найти в интернете, создать файл. Строгое определение ты найдёшь в главах 1 и 4; как инструменты проектируют — в главах 6–8.

Понятие. Подтверждение человеком — вопрос программы человеку перед изменением; без ответа «да» изменение не выполняется. Строгое определение ты найдёшь в главе 14.

Спросит ли агент в следующий раз, зависит от режима, который выбрал человек. В нашем показе агент спрашивает перед каждым изменением. Бывают и режимы, в которых он меняет файлы без вопросов. Такой режим удобнее, и только; безопаснее от него действия ИИ не становятся. Да и вопрос «Разрешить?» ни от чего не защитит, если ты всегда будешь отвечать на него «да», не читая.

Заметь и ещё одно. Чат из показа 2 умел искать в интернете. Значит, инструмент есть и у него, а во многих чатах модель сама участвует в выборе, искать или нет. Легко решить, что помощник Марка «умнее» чата Ани. На деле они различаются тем, какие действия обвязка дала модели и до чего эти действия могут «дотянуться».

Проверь себя на ходу. Перед созданием файла агент спросил подтверждение. От чего зависит, спросит ли он в следующий раз?

Вывод. Агент — это модель вместе с программой, которая даёт ей инструменты, выполняет шаги и, если так настроено, спрашивает подтверждение у человека. Чат без доступа к данным может только советовать.

На карте в рамку «агент для кода» допиши: инструменты — читать, создавать, запускать; подтверждение человека.

Показ 6. Папка агента

Вопрос-Прогноз. Где агент хранит свои правила и настройки? И можно ли их прочитать?

Учитель открывает папку, в которой работал агент, обычным текстовым редактором. Рядом с заметками в ней лежат ещё четыре вещи.

- **Файл инструкций проекта.** Обычный текст с правилами для этой папки. Агент читает его в начале каждого *сеанса* — так мы назовём одну непрерывную работу с программой.
- **Папка настроек.** В ней, среди прочего, записано, какие действия агенту разрешены без вопроса, а какие запрещены. Модель тут ни о чём не просит: это настройка самой программы, и права доступа проверяет обвязка. Подробно о них рассказано в главах 3 и 14.
- **Папка с записанными процедурами.** О ней речь пойдёт в следующем показе.
- **Файл подключений.** В нём перечислены внешние сервисы, к которым подключён агент. О нём тоже в следующем показе.

Файл инструкций мог бы выглядеть так:

```
# пример
Правила проекта
- Работай только в этой папке.
- Не удаляй файлы.
- Оглавления сохраняй в виде «имя файла — заголовок».
```

Самое главное здесь то, что никакой тайны нет. Правила агента не запрятаны внутрь модели: они лежат в файлах, и, поправив их, можно проверить, изменилось ли поведение.

Справка: имена на октябрь 2026 года. У агента для кода Claude Code файл инструкций называется `CLAUDE.md`, папка настроек — `.claude`, процедуры лежат в папках внутри `.claude/skills`, а файл подключений — `.mcp.json`. У других агентов имена другие.

В учебном комплекте нашего курса есть такие же места: папка `skills/` с записанной процедурой и папка `workspace/my_agent/` для настроек твоего собственного помощника. Открывать их пока незачем: вторая понадобится в главе 1, первая — в главе 9.

Вывод. Настройка агента — это файлы: их можно открыть, прочитать и исправить. Такие же файлы ты будешь заполнять в курсе.

Показ 7. Skill и MCP

Вопрос-Прогноз. Сегодня агент собрал оглавление. Завтра появится другая папка с заметками, а бюро к тому же захочет ставить задачи в сервисе, где ведёт свой проект. Чего агенту не хватает для первого и чего для второго?

Записанная процедура. Учитель просит агента записать порядок, по которому тот собирал оглавление: когда его применять, какие шаги делать, чего делать нельзя и как понять, что работа готова. Агент спрашивает подтверждение и создаёт в папке процедур новую папку с одним-единственным файлом. Откроем его.

```
# пример
---
name: make-index
description: Собирает оглавление папки с заметками — по строке на заметку. Применять, когда просят оглавление или список заметок.
---

1. Просмотри папку и найди файлы заметок.
2. В каждом файле прочитай первую строку — заголовок.
3. Запиши строки «имя файла — заголовок» в файл INDEX.md в той же папке.

Не меняй сами заметки. Не удаляй файлы.
Готово, когда строк в оглавлении столько же, сколько заметок.
```

В файле три части: имя; описание, то есть когда процедуру применять; и шаги вместе с запретами и признаком «готово». На следующий день, получив просьбу собрать оглавление другой папки, агент находит подходящую процедуру по описанию и работает по ней. А может и человек сам назвать её по имени.

Понятие. Agent Skill («Навык Агента») — записанная процедура: папка с файлом SKILL.md, где сказано, когда её применять, по каким шагам работать и как понять, что готово. Строгое определение ты найдёшь в главе 4; как такой файл устроен и как его написать — в главе 9.

Agent Skills придумали не для этого учебника: это открытый формат, и им пользуются разные агенты. Сама процедура ничего не выполняет и новых прав не даёт. Действия по-прежнему совершают инструменты, а разрешает их обвязка.

Подключение. Бюро ведёт свой проект во внешнем сервисе — на сайте, где хранят файлы проекта и список задач к нему. Такое хранилище проекта называют **репозиторием**. Учитель подключает сервис к агенту. До подключения в списке инструментов агента были одни только действия с файлами и командами. После подключения появились новые: «создать задачу», «прочитать список задач». Теперь в ответ на просьбу «поставь задачу „Проверить оглавление“» агент, получив подтверждение, создаёт её в сервисе.

Понятие. MCP — общий способ подключить к программе внешние инструменты и данные. Строгое определение ты найдёшь в главе 4.

Почему способ назван «общим»? Его не придумывают заново для каждой пары «агент — сервис»: один и тот же способ годится для разных программ и разных сервисов. Однако подключить сервис ещё не значит ему довериться. Всё, что приходит из внешнего сервиса, — чужие данные, и верить им на слово нельзя. Этому посвящена глава 15.

Вывод. Инструмент — действие. MCP — способ подключить инструменты и данные. Agent Skill — записанная процедура.

На карте рядом с рамкой «агент для кода» допиши три строки: отдельное действие; подключённый сервис; записанная процедура.

Показ 8. Всё вместе

Вопрос-Прогноз. Нужна маленькая программа: она читает абзац из учебника и печатает три вопроса для самопроверки. Что из трёх — чат, агент, API — понадобится, чтобы её сделать, и для чего именно?

Назовём программу «Вопросник» и проследим её путь от замысла до запуска.

Шаг 1. Чат помогает обдумать. Учитель описывает в чате замысел и просит короткий план: какие нужны файлы и как проверить программу. Чат отвечает планом. Кроме текста на экране, ничего не появилось.

Шаг 2. Агент делает. План передают агенту для кода, запущенному в пустой папке. Агент создаёт файлы программы, проверяет её на ошибки и сохраняет проект в репозиторий. На каждое изменение он спрашивает подтверждение, а учитель читает и решает, что подтвердить.

Шаг 3. Программа работает сама. Готовый «Вопросник» запускает человек, и ни чат, ни агент ему больше не нужны. Внутри программы те же несколько строк, что в показе 4: она обращается к модели через API, передаёт ей абзац и просит три вопроса. Обязку для неё написал агент, а человек за ним присматривал.

Одна семья моделей побывала в трёх обязанностях; по именам они названы в справке в начале урока Э1. Сведём всё в таблицу.

	Чат	Агент для кода	Программа через API
Где работает	в браузере; модель — на сервере	на компьютере, в выбранной папке; модель — на сервере	внутри твоей программы; модель — на сервере
Что видит и что может изменить	то, что ты написал или приложил; в твоей папке не видит ничего	файлы папки: читает, создаёт, изменяет, запускает команды — в пределах настроек	только то, что прислала твоя программа; сама ничего не меняет
Где история и настройки	у приложения	в файлах папки и в настройках на компьютере	там, где их сохранит твоя программа
Кто отвечает за действие	действий в твоей папке нет	человек, который подтвердил действие или выбрал режим без вопросов	автор программы
Как обычно платят	подписка или бесплатно с ограничениями	подписка или оплата за объём	оплата за объём текста
Когда брать	обдумать, обсудить, объяснить	сделать работу в файлах под присмотром	встроить модель в свою программу

Вывод. Одна семья моделей — три обязанности. Выбирают по делу:

1. обсудить,
2. сделать в файлах,
3. встроить в программу.

Какие бывают агенты

Агент для кода — далеко не единственный вид агента. Различать их проще всего по тому, где они действуют.

- В папке с файлами и в командной строке — как в показах 5–8.
- В браузере: открывают страницы, нажимают кнопки, заполняют формы.

- В почте, календаре и документах.
- Внутри чата — когда чат сам ищет в интернете или что-то вычисляет.
- Собственные: их собирает программист под одну задачу, со своей обязанкой вокруг модели. Таким будет и помощник нашего бюро.

Любому из них стоит задать три вопроса. Что он может изменить? Кто подтверждает его действия? Где записаны его шаги? Чем уже ответ на первый вопрос и чем точнее ответы на второй и третий, тем спокойнее с таким агентом работать.

Мини-проверка, практика и выходной билет урока Э2

Мини-проверка. Для каждой ситуации выбери вход: «обсудить в чате», «поручить агенту для кода вместе с учителем» или «написать программу с обращением через API». Объясни выбор по трём вопросам.

1. Понять, с чего начать доклад о маятнике.
2. Разобрать папку с заметками кружка и собрать по ним сводку в файл.
3. Сделать сайт-викторину, где вопросы к каждому тексту составляются автоматически.
4. Попросить объяснить непонятное место в параграфе.
5. Разложить сорок файлов проекта по папкам, разобравшись, что в каком лежит.

Ответ. 1 и 4 — чат: нужно обсудить, менять нечего. 2 и 5 — агент вместе с учителем: работа идёт в файлах, и каждое изменение кто-то должен подтвердить. 3 — программа через API: модель должна отвечать внутри сайта, без человека у окна чата.

Практика, трек А. Дополни таблицу наблюдений строками показов 5–8 и закончи карту.

Результат: восемь строк таблицы и заполненная карта «что есть что».

Выходной билет урока Э2. Назови три вопроса, которые нужно задать любому агенту, прежде чем пустить его в папку.

Обязательные проверки

Проверка	ПРАВИЛЬНО	НЕПРАВИЛЬНО
Должен	в режиме с подтверждением агент создаёт файл только после ответа «да»	агент спрашивает всегда: на деле это зависит от режима
Не должен	чат без доступа к папке ничего в ней не меняет	чат «ничего не умеет»: у него могут быть свои инструменты, например поиск
Ошибка	обращение через API с неверным ключом возвращает сообщение об ошибке	модель «отказалась отвечать»: до неё запрос даже не дошёл
Граница	сравнение моделей — несколько задач, один запуск, одна дата	«модель А лучше модели Б» вообще

Безопасность

Безопасность. Личные данные, свои и чужие, в чат не вводят: всё, что ты написал, уходит на чужой сервер. Ключ — секрет, его не должно быть ни в тексте программы, ни в репозитории, ни в книге. Агента запускают в отдельной папке, а человек читает, что именно подтверждает; режим без вопросов безопасности не прибавляет, скорее наоборот. Адрес сайта проверяют прежде, чем что-нибудь на нём вводить. У сервисов есть возрастные правила: одни рассчитаны только на взрослых, другие допускают школьников лишь с

согласия родителей. Учётные записи заводят только с разрешения взрослых и по этим правилам; обходить ограничения сервиса или школы нельзя.

Зачем это тебе

1. **«ИИ сказал...»** Одноклассник ссылается на ответ ИИ. Задай ему три вопроса: через что он спрашивал; было ли программе откуда узнать то, о чём она говорит, — поиск, файл; на какую дату это верно.
2. **Доклад или реферат.** Чат не видит твой файл, пока ты его не передал, а всё переданное уходит на чужой сервер. Что можно передавать и чей это сайт, решают до первого сообщения.
3. **Проект кружка.** Замысел удобно обсудить в чате. Работа в папке — дело агента, и только вместе с учителем. Встроить модель в свою программу — это обращение через API; в нашем курсе оно остаётся необязательным продолжением.

Проверь себя

1. Чем приложение-чат отличается от модели?
2. Откуда модель в чате «знает» начало разговора?
3. На два одинаковых вопроса пришли два разных ответа. Что из этого следует для проверки ответов?
4. Что значит «эта модель лучше» и как проверить это для своей задачи?
5. Что программа должна прислать через API, чтобы модель учла прошлые реплики?
6. По каким трём вопросам сравнить чат без доступа к папке и агента для кода?
7. В папке агента лежат файл инструкций проекта, файл с записанной процедурой и файл подключений. За что отвечает каждый?
8. Найди ошибку: «Агент спросил подтверждение — значит, действие безопасно».
9. Что в этой главе — общий принцип, а что верно только на дату справки?

Ответы

1. Модель составляет ответ по переданным сведениям. Приложение принимает вопрос, передаёт сведения модели и показывает ответ.
2. Приложение каждый раз присылает ей прошлые реплики этого разговора.
3. По одному ответу судить нельзя: ответ проверяют, а сравнения повторяют.
4. «Лучше» бывает только для своих задач и критериев на определённую дату. Проверяют на наборе задач с известными ответами, в одинаковых условиях.
5. Весь прошлый разговор: иначе модель о прошлых репликах не узнает.
6. Что система может изменить; кто подтверждает действие; где записаны шаги.
7. Файл инструкций — правила для этой папки; файл процедуры — как выполнять повторяемую работу; файл подключений — список внешних сервисов.
8. Подтверждение защищает, только если человек читает, что подтверждает; и задаётся вопрос не во всяком режиме.
9. Принцип — роли и то, кто что делает. На дату справки верны названия продуктов, имена файлов настроек и возможности конкретных чатов.

Домашняя работа — обязательно

Выбери одну систему с ИИ, о которой ты слышал; пользоваться ею для этого не нужно. Запиши, что это: приложение-чат, агент или программа, которая сама обращается к модели? По каким признакам ты так решил и что нужно узнать, чтобы это проверить?

Словарь главы

Все формулировки здесь рабочие, такие же, как в рамках «Понятие». Точные определения даны в главах, названных в третьем столбце.

Слово в этой главе-экскурсии	Понятие	Строгое определение: статья словаря и глава
модель	часть системы, которая по переданным ей сведениям составляет ответ; сама она ничего не открывает — ни сайтов, ни файлов	Языковая модель / LLM — глава 1
приложение (чат)	программа, которой пользуется человек: принимает вопрос, обращается к модели и показывает ответ; чат в браузере — её пример	ИИ-приложение — глава 1
контекст	всё, что передано модели в данном конкретном обращении: вопрос и то, что программа добавила к нему, например прошлые реплики	Контекст — глава 1
версия модели	имя и номер выпуска модели, по которым можно сказать, к какому именно варианту модели обращались	слово вводной главы
API	вход для программ: способ, которым одна программа обращается к другой по заранее описанным правилам и получает ответ в виде данных	слово вводной главы
ключ API	секретная строка, по которой сервис узнаёт, чья программа к нему обращается	Secret / секрет — глава 10
обвязка	та часть программы, которая стоит между человеком и моделью: собирает то, что будет передано модели, хранит разговор, выполняет действия и следит за ограничениями	Harness / agent harness — глава 3
агент	система, которая не только отвечает, но и действует: модель участвует в выборе шага, программа выполняет его и показывает модели, что получилось	ИИ-агент — глава 1
инструмент	отдельное действие, которое программа умеет выполнить: прочитать файл, найти в интернете, создать файл	Tool / инструмент — главы 1 и 4
подтверждение человеком	вопрос программы человеку перед изменением; без ответа «да» изменение не выполняется	Human approval / подтверждение человеком — глава 14
Agent Skill	записанная процедура: папка с файлом <code>SKILL.md</code> , где сказано, когда её применять, по каким шагам работать и как понять, что готово	Agent Skill — глава 4; <code>SKILL.md</code> — глава 9
MCP	общий способ подключить к программе внешние инструменты и данные	MCP / Model Context Protocol — глава 4

Возвращаемся к первоначальной сцене

Теперь на вопрос Ани можно ответить. У неё был чат — приложение без доступа к папке бюро. Оно передало модели просьбу, модель составила ответ, и ответ этот оказался советом: действия «прочитать файлы Ани» у этого приложения нет. У Марка был агент, запущенный прямо в папке. Его обвязка дала модели инструменты — просмотреть папку, прочитать файлы, создать новый — и перед изменением спросила Марка.

Что же было одинаковым? Возможно, модель, а вернее, семья моделей; поэтому и название на экране совпало. Что было разным? Обвязка: что она передаёт модели, какие действия умеет выполнять и о чём спрашивает человека. ИИ Ани не был сломан, и ИИ Марка не был умнее.

Как это можно было проверить? Спрашивать «какой у тебя ИИ» почти бесполезно. Надо задать три вопроса:

1. Что этой программе доступно?
2. Какие действия она умеет выполнять?
3. О чём она спрашивает человека?

Аня и Марк решают собрать для бюро собственного помощника. С этого начинается глава 1.

Мост к главе 1 учебника

Напоследок два честных предупреждения.

Первое. На экскурсии работали живые системы. В практиках курса живой модели почти не будет: её место займёт учебная заглушка, а позже — ответы, подготовленные заранее. Одну причину ты уже видел в опыте 2: на один и тот же вопрос живая модель отвечает по-разному, и класс не смог бы сверить результаты. Об остальных причинах сказано во Введении, в разделе «Почему здесь не сразу живая модель».

Второе. Агент для кода на экскурсии создавал файлы и сохранял проект в репозиторий. Помощник, которого будешь строить ты, намеренно устроен скромнее: он читает учебные карточки задач и готовит черновик, а изменить что-либо может только после решения человека. Зато его можно разобрать целиком, до последней проверки, и доказать, что именно он сделал. О нём рассказано во Введении, в разделе «Что мы будем строить».

Эта глава-экскурсия показала, что есть что. Как всё это устроено внутри и как самому запрограммировать агента, она не объясняла. Начнём с вопроса, который с виду проще всех.

▮ **Следующий вопрос.** Кто отвечает, а кто действует?

Ответ ты найдёшь в главе 1, «Модель, приложение и агент».