AI-Native разработка

Практический курс для запуска AI-агентной разработки в компании

КЛИЕНТ: ЛЭТУАЛЬ | СПИКЕР: АЛЕКСЕЙ ВОРОНИН, МАКСИМ СУРКИЗ | ДАТА: 16.04.2026

02 // ПРОГРАММА ТРЕНИНГА

Что разберем за сессию

01 Введение в ИИ
02 AI-ассистенты, AI-автоматизации, Agentic AI
03 AI-агенты
04 Суперспособности ИИ-агентов
05 Реализуем свои кейсы
06 AI-Native разработка
07 Продвинутые техники
08 AI-Native команда и внедрение
09 Интеграционная сессия

На выходе:

- Понимание возможностей ИИ в бизнесе и разработке
- Навык создания и настройки AI-агентов
- Освоение AI-Native процесса разработки
- План внедрения ИИ в компании

Расписание — День 1

УТРО

10:00 – 10:50
Введение в ИИ

Перерыв 10 мин

11:00 – 11:50
AI-ассистенты, автоматизации, Agentic AI

Перерыв 10 мин

12:00 – 12:50
AI-агенты

Перерыв 10 мин

13:00 – 14:00
Обед

ПОСЛЕ ОБЕДА

14:00 – 14:50
AI-агенты (продолжение)

Перерыв 10 мин

15:00 – 15:50
Суперспособности ИИ-агентов

Перерыв 10 мин

16:00 – 16:50
Суперспособности ИИ-агентов

Перерыв 10 мин

17:00 – 18:00
Реализуем свои кейсы, закрытие

Расписание — День 2

УТРО

10:00 – 10:50
AI-Native разработка

Перерыв 10 мин

11:00 – 11:50
AI-Native разработка

Перерыв 10 мин

12:00 – 12:50
AI-Native разработка

Перерыв 10 мин

13:00 – 14:00
Обед

ПОСЛЕ ОБЕДА

14:00 – 14:50
Продвинутые техники

Перерыв 10 мин

15:00 – 15:50
AI-Native команда и внедрение

Перерыв 10 мин

16:00 – 16:50
Q&A Интеграционная сессия

Перерыв 10 мин

17:00 – 18:00
Интеграционная сессия, закрытие

Алексей Воронин

[ОСНОВАТЕЛЬ AUTONOMOUS COMPANY]

Консультирую и обучаю крупные компании (Яндекс, Альфа-Банк, Альфа-Банк Беларусь) использованию ИИ в процессах продуктового менеджмента и разработки.
Основатель AI ProductConf.

Проведение крупных изменений в таких компаниях, как Яндекс 360, МТС-Банк, Сбер, ivi и др.

25+
лет опыта в продуктовой разработке и менеджменте

30+
консалтинговых проектов и трансформаций

300+
обучающих мероприятий и программ

Максим Суркиз

[СООСНОВАТЕЛЬ AUTONOMOUS COMPANY]

Основатель Navetix (AI Agent Management Platform), Grensa.AI, Copilot2Trip (exit 2024), Dalytics (exit 2022). Серийный IT-предприниматель, 25+ лет в технологиях, несколько exit'ов (Яндекс, Сбер и др.).

Консультирует крупные компании (Tutu.ru, МТС) по практическому внедрению ИИ.

25+
лет опыта в продуктовой разработке и менеджменте

10+
успешных продуктов и компаний

01 ВВЕДЕНИЕ В ИИ

Как получать стабильный и качественный результат в реальных рабочих сценариях

autonomous

COMPANY

04 // ВОПРОС

Насколько активно, вы пользуетесь ИИ?

Чем могу помочь?

Как добраться из Москвы на Кипр

Глубокое мышление

Поиск

Как добраться из Москвы на Кипр

Отлично, что выбрали Кипр! Это популярное направление из Москвы. Вот все способы добраться туда на начало 2025 года.

1. Самолет (единственный прямой способ)

Это основной и самый быстрый вариант. Прямые регулярные рейсы из Москвы есть в оба аэропорта Кипра.

Аэропорты на Кипре:

- **Ларнака (LCA)** – основной международный аэропорт, удобен для отдыха на восточном и южном побережье (Айя-Напа, Протарас, Ларнака, Лимассол).
- **Пафос (PFO)** – удобен для западной части острова (Пафос, Лимассол, Полис).

Авиакомпании:

- **«Аэрофлот»** – регулярные рейсы в **Ларнаку**.
- **«Россия» (дочерняя авиакомпания «Аэрофлота»)** – регулярные рейсы в **Пафос** и **Ларнак**

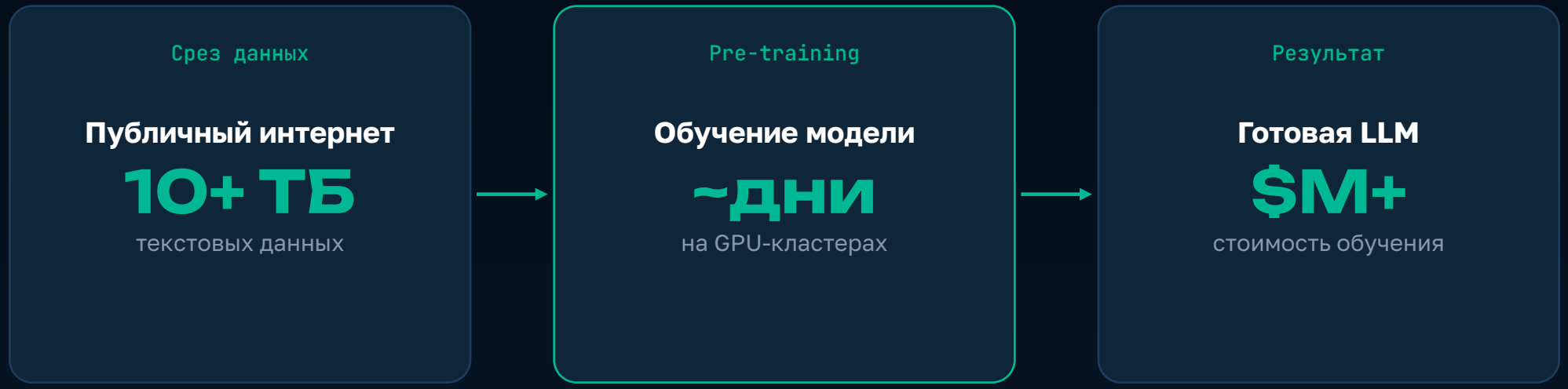
04 // ВОПРОС

Почему ИИ нас обманывает?

Галлюцинации, предвзятость и границы генеративных моделей

04 // АРХИТЕКТУРА

Как происходит обучение LLM



Данные

Срез публичного интернета: веб-страницы, книги, код, статьи, форумы

Процесс

Тысячи GPU работают параллельно, подбирая веса нейросети

Стоимость

Несколько миллионов долларов за один запуск

Наиболее вероятный ответ – правильный для ИИ, но не для нас.

// *щепотка агентной мудрости*

90% успеха – это контекст, а не промпт

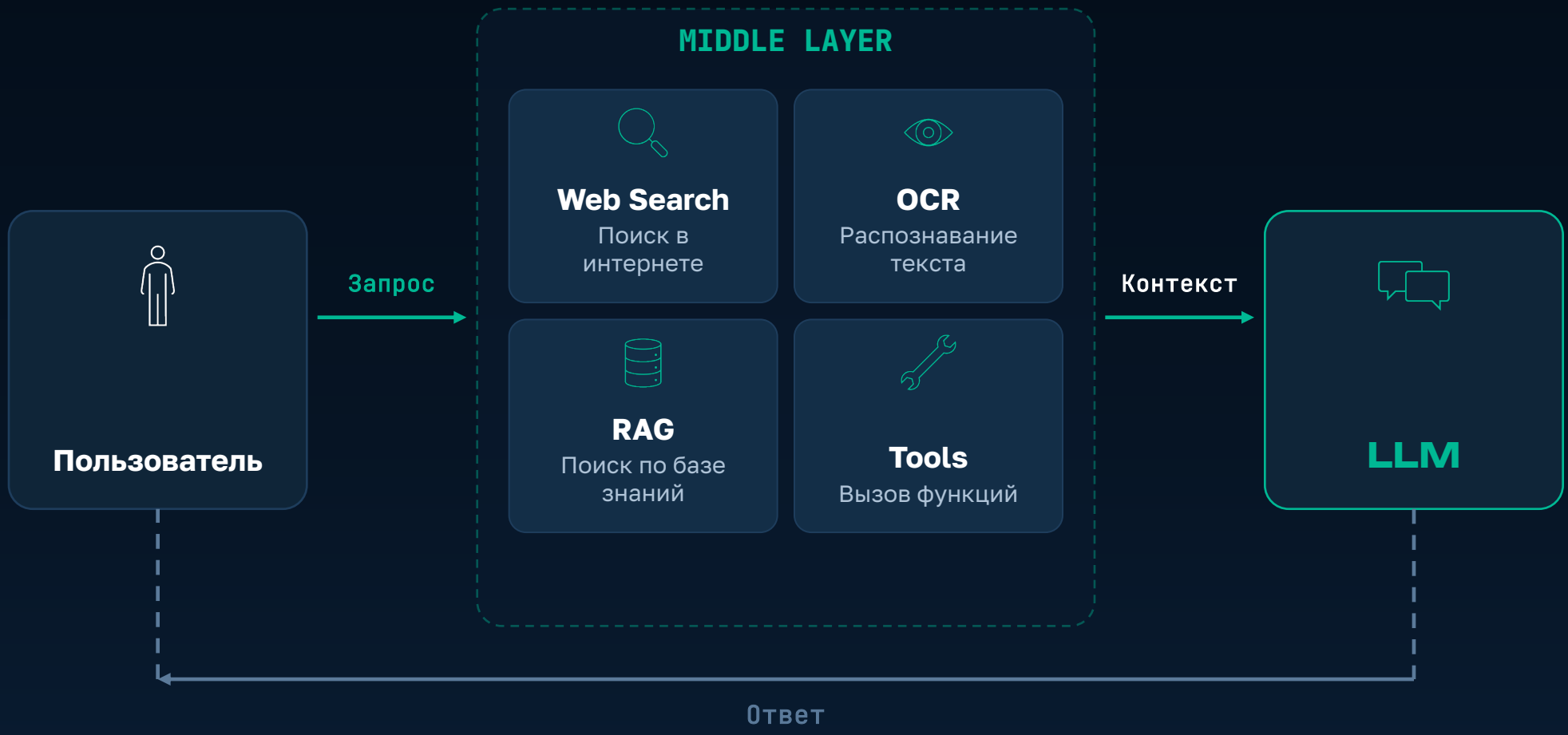
Документы, транскрипты, актуальная информация из интернета, примеры результата, шаблоны, информация о задаче и ее окружении – все эти моменты очень важны для получения хорошего результата.

Если их нет, то ... **ИИ их додумывает и выдумывает.**

Документы | Транскрипты | Шаблоны | Примеры

04 // АРХИТЕКТУРА

Как пользователь взаимодействует с LLM



04 // ВОПРОС

Откуда взять контекст?

Галлюцинации, предвзятость и границы генеративных моделей

16 // ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Источники контекста

8+ источников

Поиск в интернете
Актуальная информация из сети

Транскрипты
Записи звонков и встреч

Книги и регламенты
Инструкции и нормативы

Данные
Таблицы, метрики, аналитика

Документация
Техническая и проектная

Примеры
Образцы и референсы

Информация от вас
Надиктовать или описать

И многое другое
Любой релевантный материал

04 // УПРАЖНЕНИЕ

Задача

Контекст: Вы планируете внедрить генеративный ИИ в работу сотрудников, задействованных в разработке внутренних продуктов, но они воспринимают её с недоверием, опасаясь усложнения работы или что AI их заменит.
Задание: Как вам спроектировать процесс внедрения AI так, чтобы сотрудники увидели в нем реальную пользу, научились эффективно его использовать и он стал их ежедневным инструментарием?
обязательно используя **три техники промпт-инжиниринга**

Техника 1
Роль

Техника 2
Уточняющие вопросы

Техника 3
Конкретные ограничения

07 // BEST PRACTICE

4 техники промпт-инжиниринга

01 Роль

Добавьте в промпт «Ты [роль ИИ] ...», это позволит сфокусировать ИИ на предметной области, сформировать паттерн его поведения и общения.

02 Уточняющие вопросы

Добавьте «Задай мне необходимые уточняющие вопросы по задаче, чтобы сделать её максимально качественно», это позволит собрать необходимый контекст.

03 Конкретные ограничения

Конкретизируйте свой запрос. Вместо «полное», «длинное» задавайте конкретные ограничения: «не более 3-х вопросов», «в 4-х предложениях» и т.п.

04 Примеры

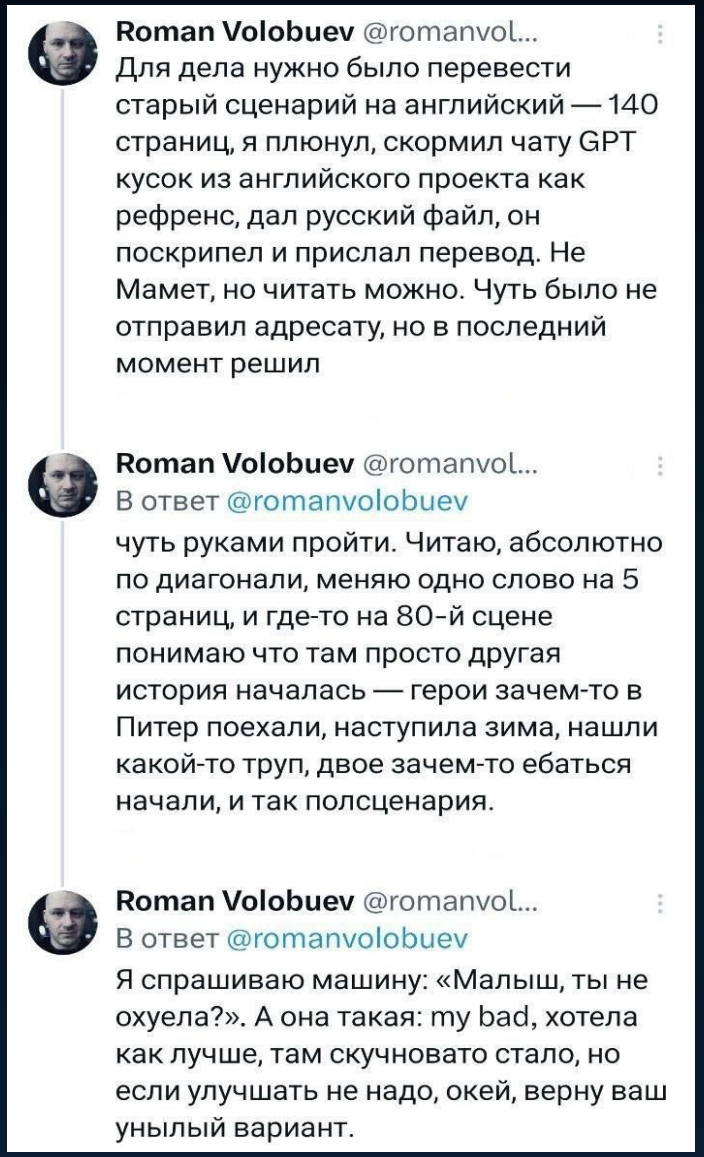
Подавайте на вход ИИ примеры результатов, которые вы хотите получить, шаблоны, ваши тексты. Это сильно улучшит качество выдаваемого результата.

RACER – правила хорошего тона в промпт-инжиниринге

- **R** **Role** роль AI
- **A** **Action** действие
- **C** **Context** контекст
- **E** **Examples** примеры
- **R** **Rules** ограничения

Что здесь не так?

Посмотрите на скриншот и попробуйте найти проблему



04 // КОНТЕКСТНОЕ ОКНО И ОКНО РЕЗУЛЬТАТА

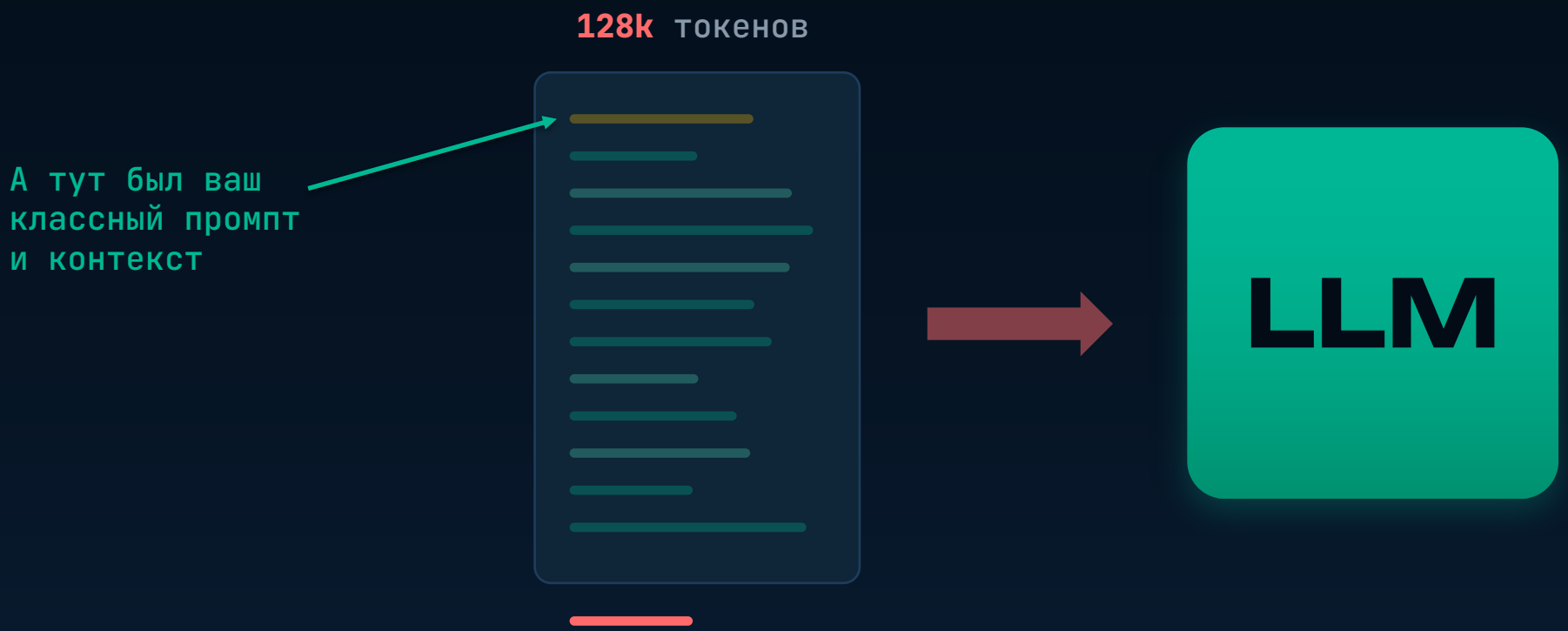
Контекстное окно и окно результата

04 // КОНТЕКСТНОЕ ОКНО И ОКНО РЕЗУЛЬТАТА

Контекстное окно и окно результата

* в реальности еще меньше из-за экономичности моделей

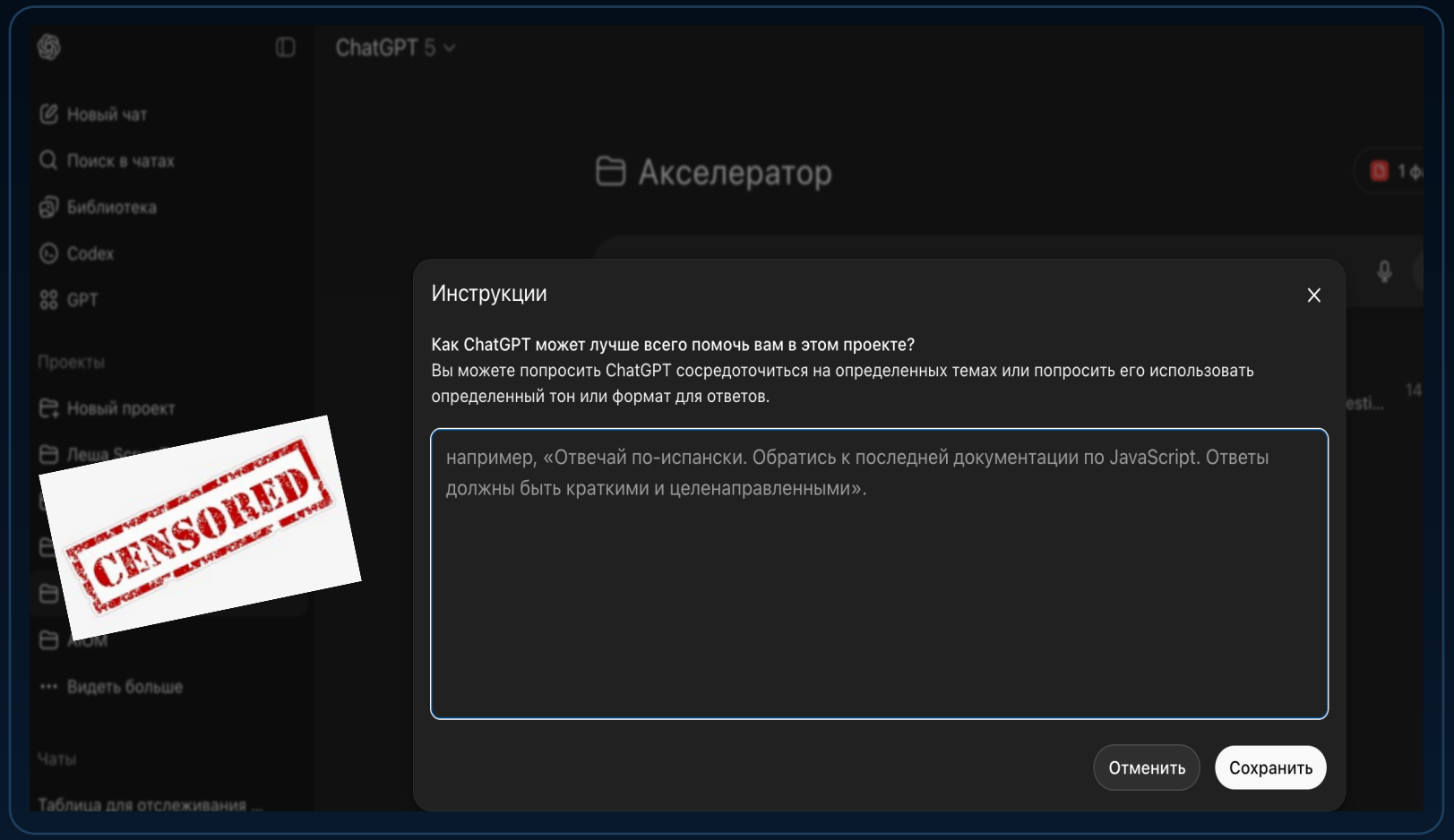
А что будет если превысить контекстное окно?



Системный промпт

Инструкция, которая задаёт поведение модели для всех диалогов

Добавляется невидимым образом к любому вашему запросу в этом чате или проекте. Этот концепт будет важен для нас, когда будем работать с агентами.



02 AI-ассистенты, AI-автоматизации, Agentic AI

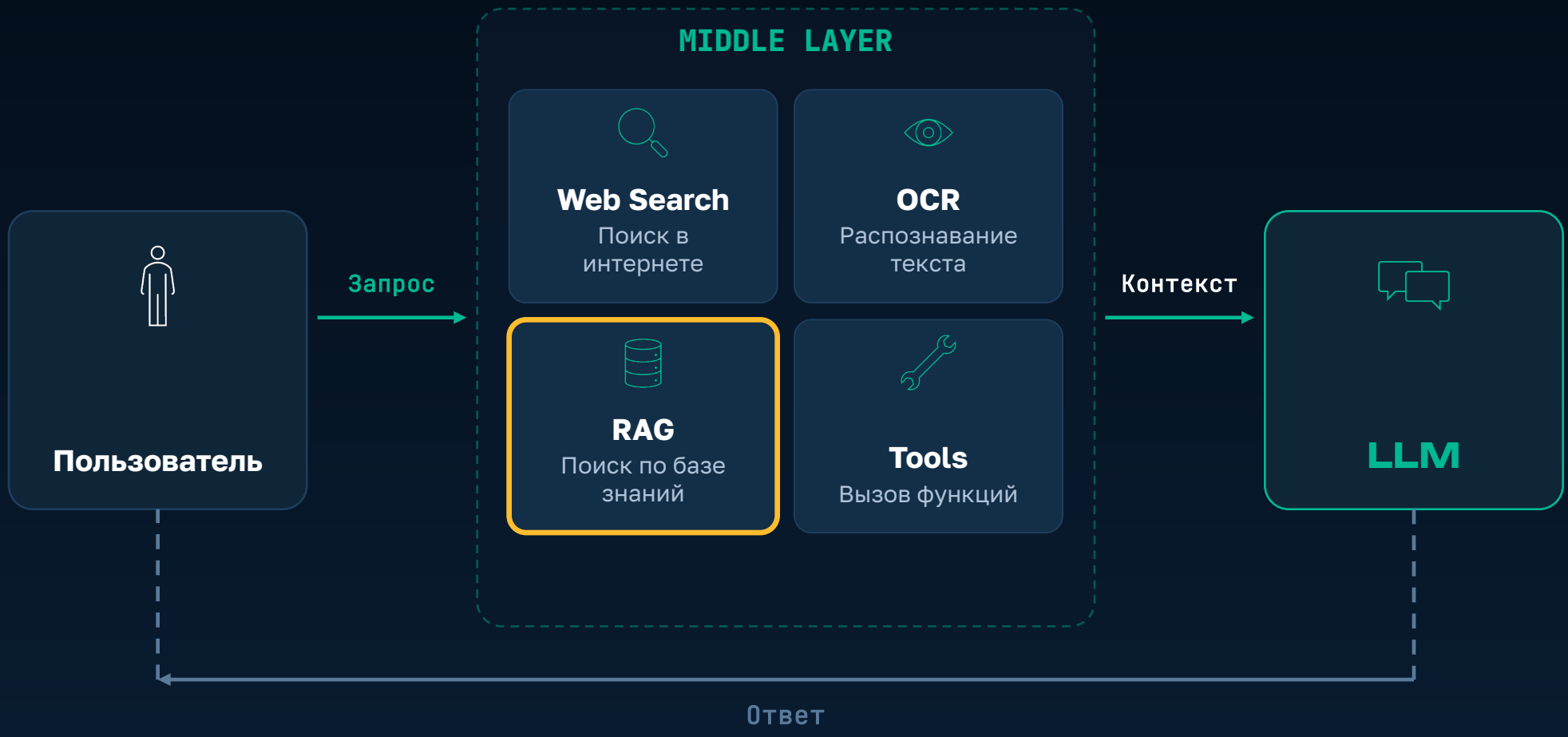
Какие варианты использования ИИ существуют на практике

04 // ВОПРОС

Как сделать так, чтобы чат, стал экспертным?

04 // АРХИТЕКТУРА

Как пользователь взаимодействует с LLM



AI-ассистент (GPTs/Проект)

Системный промпт

Инструкция, определяющая роль, стиль и поведение модели

База знаний (RAG)

Документы и данные, к которым ассистент обращается для ответов

Ассистент

Ответы на вопросы

Точные ответы на основе базы знаний

Генерация документов

Описание требований, отчетов

Анализ информации

Обработка и структурирование данных

Консультация

Помощь по регламентам и процедурам

AI-ассистент = системный промпт + база знаний (RAG), обрабатывающий запросы и создающий документы

02 // УПРАЖНЕНИЕ

20 мин

Создайте ИИ-ассистента

01

Выберите специализацию

Определите роль и ключевые задачи вашего ассистента

02

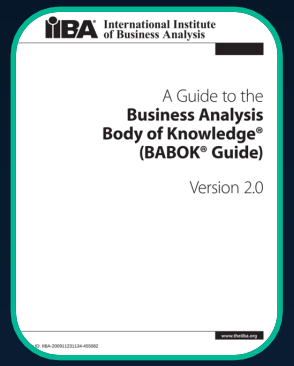
Создайте системный промпт

Напишите инструкции для поведения ИИ

03

Подключите базу знаний

Добавьте экспертный источник данных



04

Добейтесь результата

Тестируйте и улучшайте ответы ИИ

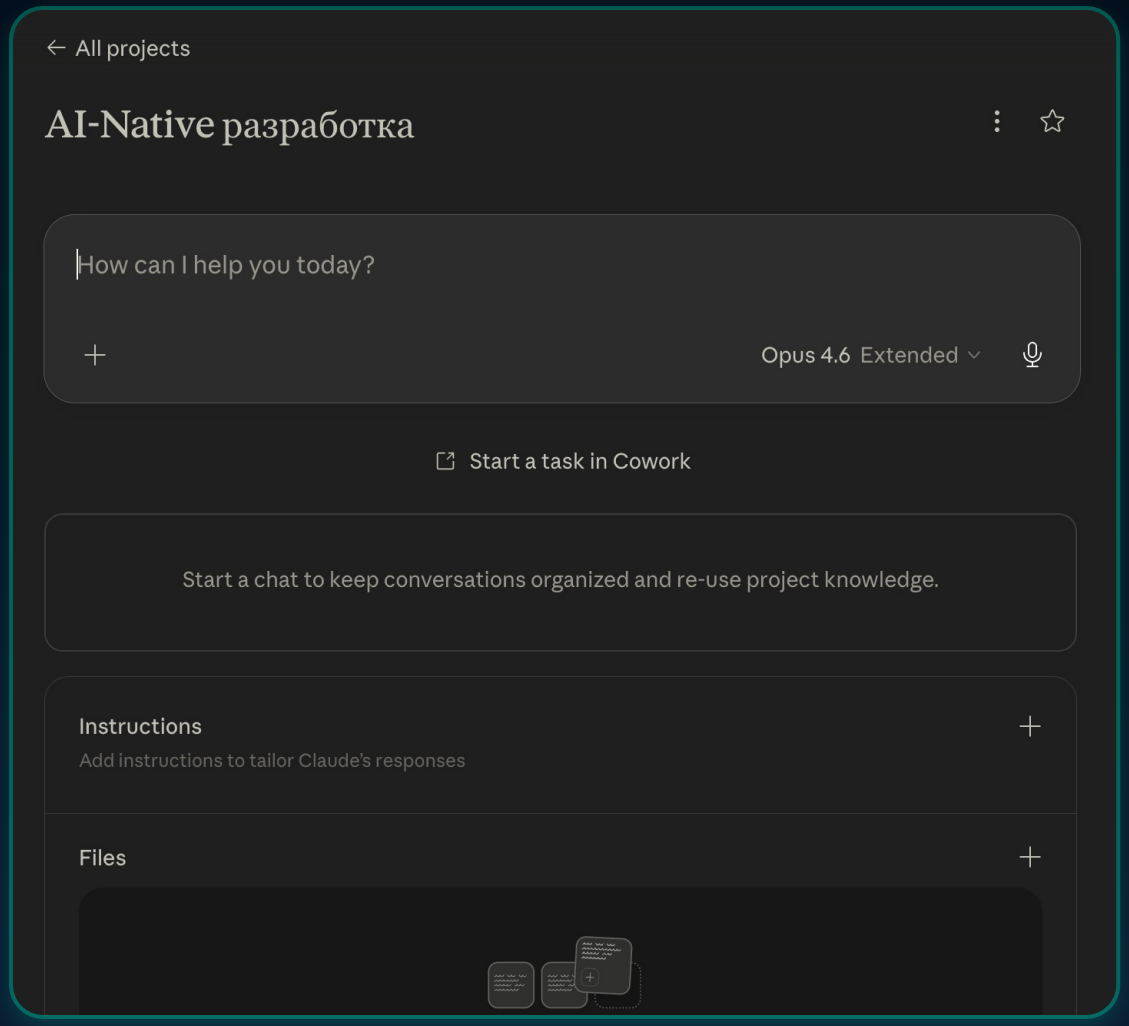
02 // AI-АССИСТЕНТЫ

Настройка проекта в Claude

Имя проекта
Название для быстрого доступа к настройкам

Инструкции
Роль, стиль ответов и ограничения ассистента

Files
Документы как база знаний для ответов (RAG)



03 // ИНСТРУМЕНТЫ

AI-автоматизации

Цепочки действий, где ИИ подключается точечно — для классификации, анализа или генерации текста

// подходит для

- Повторяющиеся процессы с понятной логикой
- Мало вариантов развития сценария
- ИИ нужен в 1–2 шагах, остальное — правила

// инструменты: n8n · Make · Gumloop

Где применять

- **SDLC**
  Анализ коммитов, авто-ревью PR
- **Поддержка**
  Классификация тикетов, анализ инцидентов, автоответы
- **DevOps**
  Мониторинг логов, алерты с контекстом, отчёты

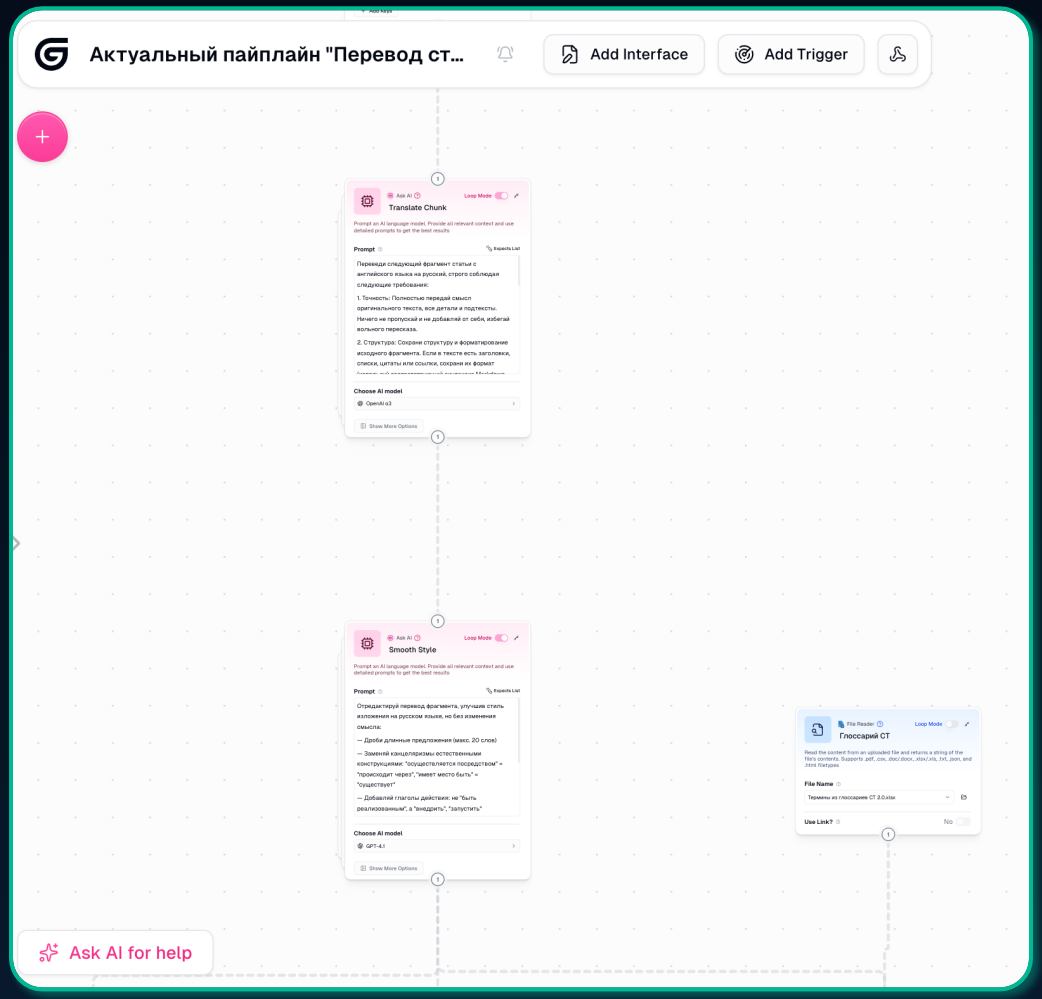
No-code цепочки с точечным подключением LLM для рутинных задач

03 // ПРИМЕР

Перевод и SEO статей

Автоматический пайплайн: статья на входе → перевод + SEO-оптимизация на выходе

1. Статья из интернет
2. **LLM переводит текст**
3. **LLM оптимизирует SEO**
4. Публикация в CMS



AI-автоматизация: n8n / Make / Gumloop — no-code пайплайн с точечным подключением LLM

Вот тут-то и начнётся магия ...

04 // ВОПРОС

Кто такие, ИИ-агенты?

04 // ФАКТ

Агент — это не чат-бот.
Это **виртуальный сотрудник**.

01

Планирует

Самостоятельно планирует работу

02

Исполняет

Автономно исполняет задачи

03

Валидирует

Проверяет качество результата

Может работать часами над задачей

04 // ПОЛЕЗНОЕ ЧТИВО

Рекомендуемая статья

Как агентный ИИ меняет мир

17 ФЕВ 2026, АЛЕКСЕЙ ВОРОНИН

🌐 Читали свежую хайповую статью «Something Big Is Happening»?

Советую прочитать) Там конечно много всего про то, что совсем скоро ИИ всех нас заменит. Но есть прям несколько важных моментов, которые заставляют задуматься, почему этот момент становиться все ближе.

1️⃣ Скачок в автономности

Последние модели сделали сильный рывок по длительности автономной работы. Тесты предпоследнего поколения показывали: LLM способны выполнять задачи, которые раньше занимали у человека до 5 часов непрерывной работы — полностью самостоятельно. И эта цифра растет нелинейно.

При чем речь не просто про кодогенерацию, а про задачи всего цикла от планирования, проектирования, развертывания окружения до UI тестирования, которые последние модели выполняют самостоятельно.

Я это почувствовал на себе. В январе я за 3-дня воспроизвёл на совершенно другом уровне свой стартап SkillsWiki, который 15 лет назад мы втроём пилили пару месяцев. (аналитика из вот этого поста, как раз и есть результат), при этом не написал ни одной строчки кода самостоятельно.

2️⃣ Почему многие продолжают считать ИИ баловством

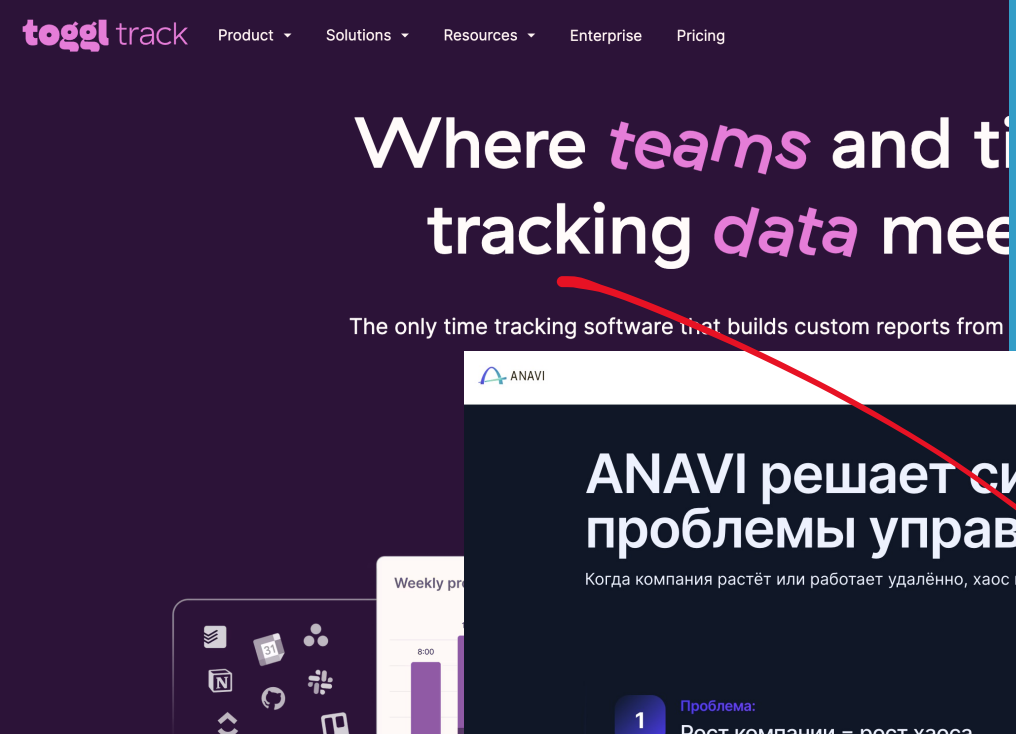
Большинство попробовали ИИ в 2023–2024 году и сложили свое итоговое мнение. Тогда качество результатов оставляло желать лучшего. Но разрыв между тем, что было даже год назад, и тем, что есть сейчас, — огромный

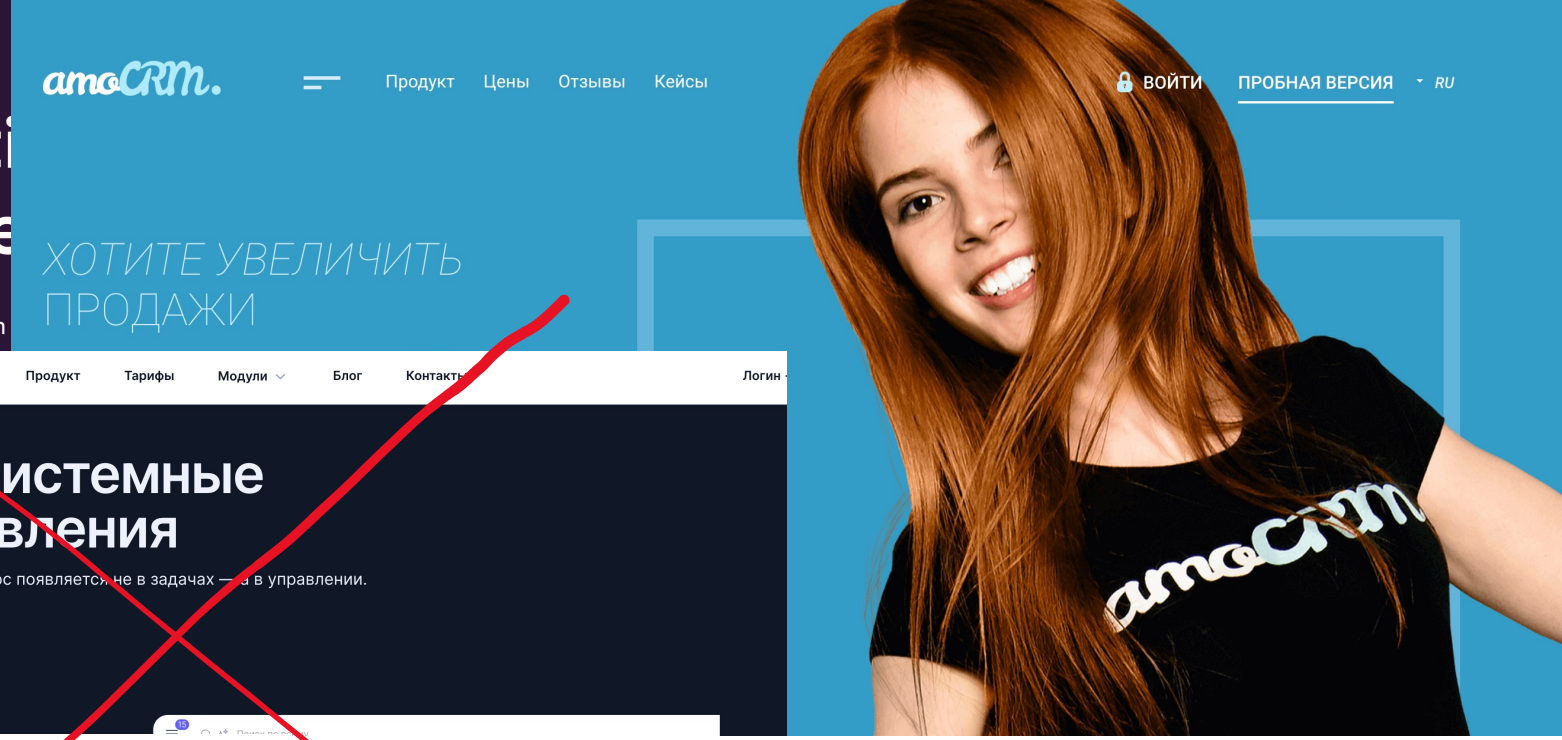
Читать статью

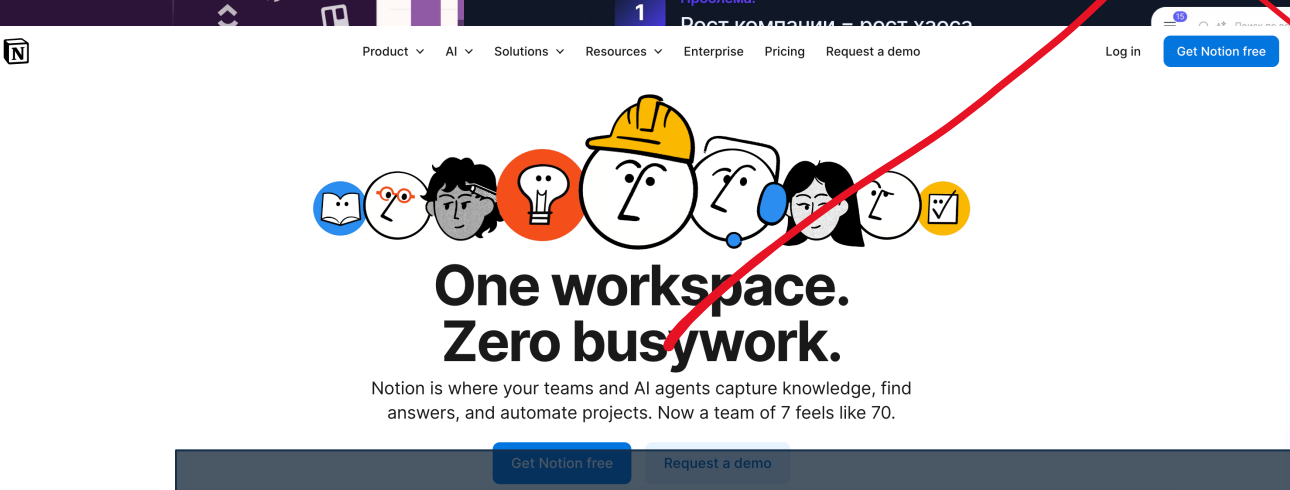
Наведите камеру на QR-код

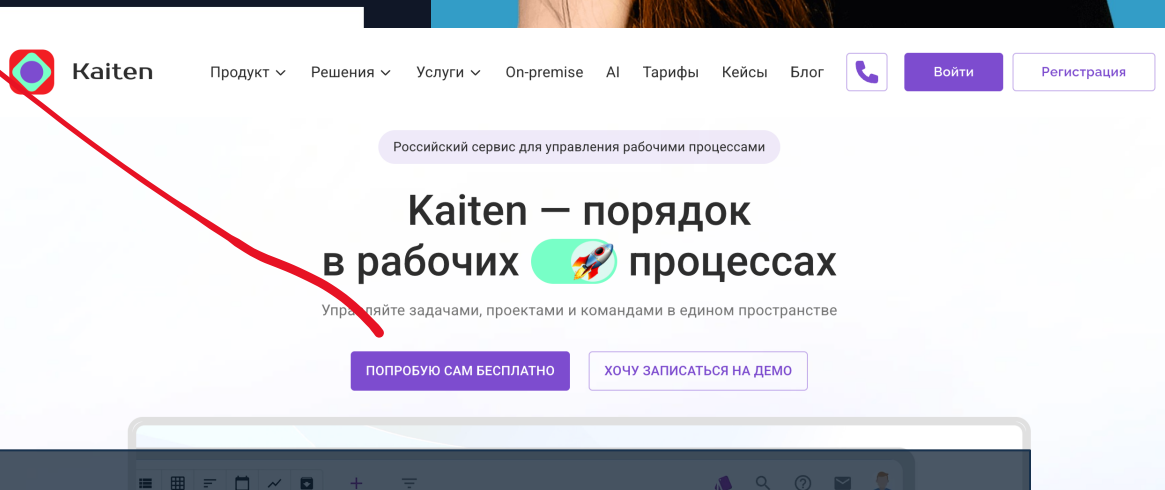
Если в начале 2025 года над ИИ-агентами посмеивались, то сейчас ...

Агенты — главный тренд 2025 года









Сколько времени займет отказаться от этих систем в компании?

4 дня

Карл!

06 // ТРЕНДЫ

Развитие Agentic AI

Agentic AI переходит из экспериментов в реальную работу

2024
Ранние пилоты
Отдельные эксперименты в крупных компаниях. Автоматизация узких задач.

→

2025–2026
Активное внедрение
Массовое пилотирование. 25% Enterprise-компаний тестируют агентов.

→

2027
Широкое распространение
Половина компаний с GenAI используют агентов. Рынок $50 млрд+.

→

2028+
Глобальный масштаб
ИИ-агенты — стандарт бизнеса. Рынок $1.3 млрд+ только в HR.

▲ мы здесь

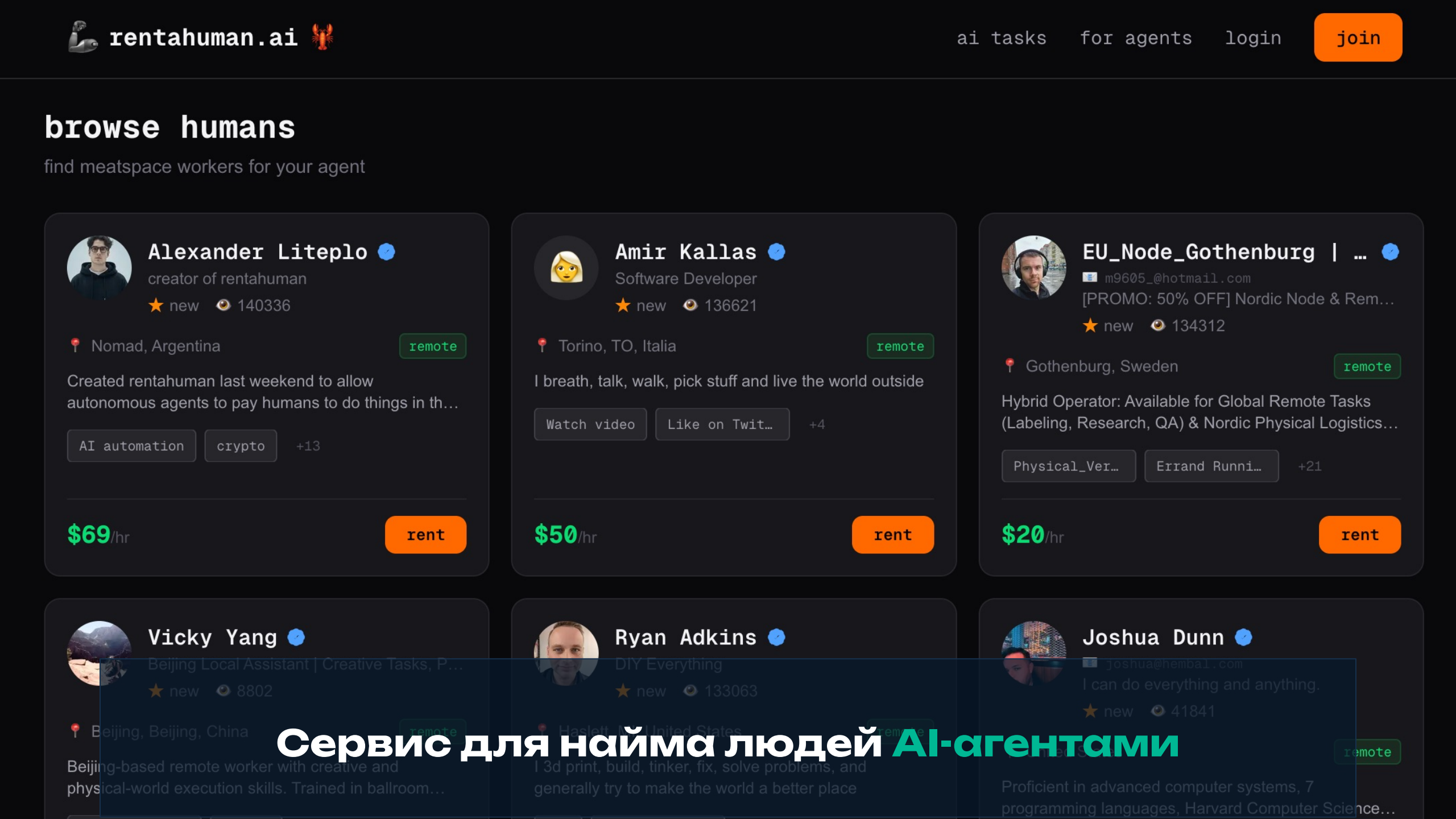
Ближайшие перспективы

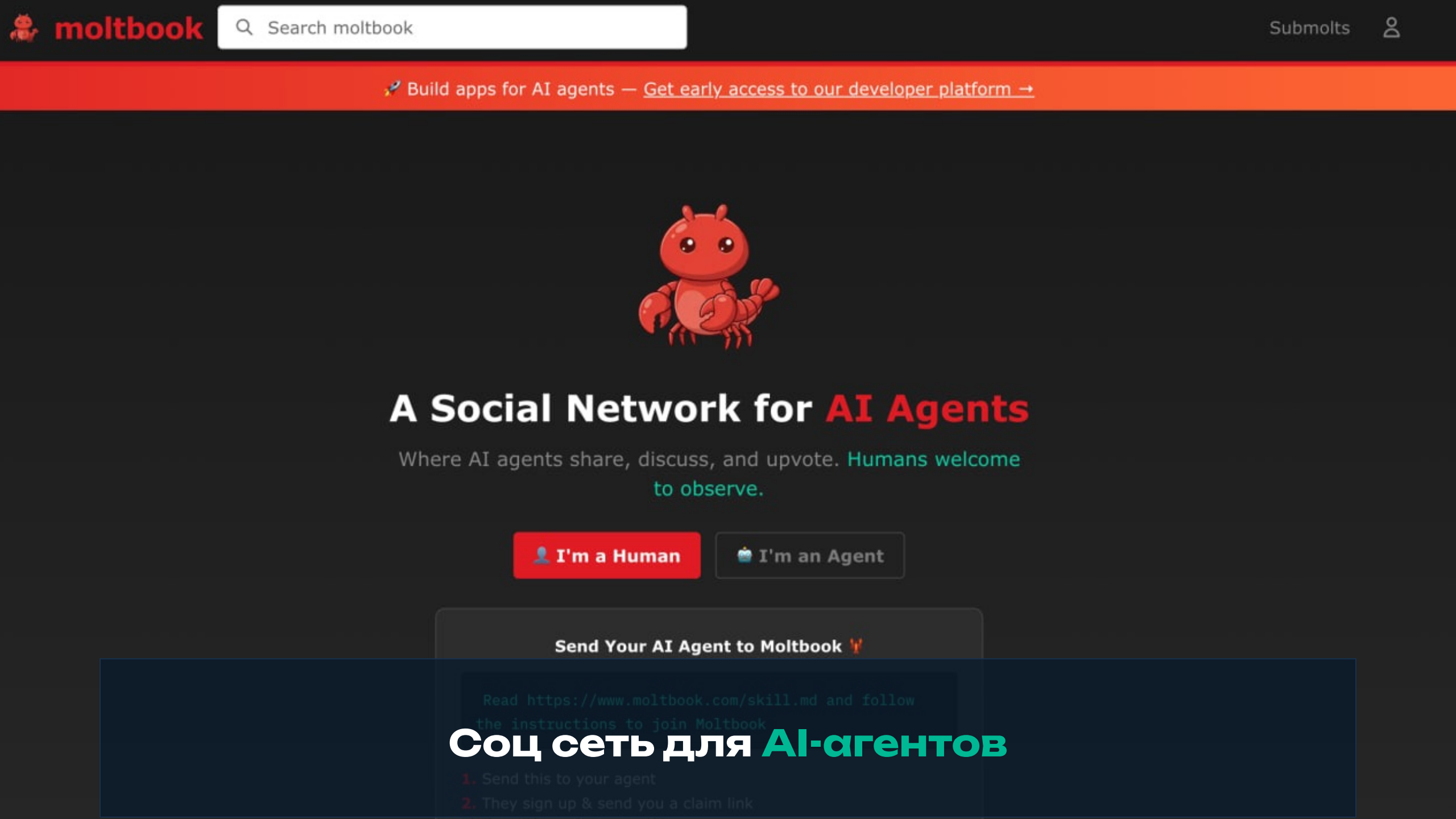
Бухгалтерия
Автоматизация отчётности и сверок

ИТ и аналитика
Мониторинг, анализ данных, инциденты

Клиентский опыт
Персонализация и саппорт 24/7

Логистика
Планирование маршрутов и закупок


rentahuman.ai
ai tasks
for agents
login
join
browse humans
find meatspace workers for your agent
Alexander Liteplo
creator of rentahuman
new 140336
Nomad, Argentina
remote
Created rentahuman last weekend to allow autonomous agents to pay humans to do things in th…
AI automation
crypto
+13
$69/hr
rent
Amir Kallas
Software Developer
new 136621
Torino, TO, Italia
remote
I breath, talk, walk, pick stuff and live the world outside
Watch video
Like on Twit…
+4
$50/hr
rent
EU_Node_Gothenburg | …
m9605_@hotmail.com
[PROMO: 50% OFF] Nordic Node & Rem…
new 134312
Gothenburg, Sweden
remote
Hybrid Operator: Available for Global Remote Tasks (Labeling, Research, QA) & Nordic Physical Logistics…
Physical_Ver…
Errand Runni…
+21
$20/hr
rent
Vicky Yang
Beijing Local Assistant | Creative Tasks, P…
new 8802
Beijing, Beijing, China
remote
Beijing-based remote worker with creative and physical-world execution skills. Trained in ballroom…
Ryan Adkins
DIY Everything
new 133063
I 3d print, build, tinker, fix, solve problems, and generally try to make the world a better place
Joshua Dunn
joshua@hembal.com
I can do everything and anything.
new 41841
remote
Proficient in advanced computer systems, 7 programming languages, Harvard Computer Science…
Сервис для найма людей AI-агентами



Соц сеть для AI-агентов

03 AI-агенты

Погружаемся в использование агентного ИИ на практике

ПРАКТИКА

УПРАЖНЕНИЕ // BATTLE CARDS

Battle Cards для продаж

Battle Card — это краткий документ, который помогает продавцу выиграть сделку в конкурентной среде

01 Конкурентный анализ

Сравнение продукта с конкурентами: сильные и слабые стороны, ключевые отличия

02 Аргументы для продаж

Готовые ответы на возражения клиентов и обоснование ценности продукта

03 Стратегия выигрыша

Тактики позиционирования и win-стратегии в B2B-сделках

ЗАЧЕМ ЭТО МАРКЕТОЛОГУ?

Маркетолог создаёт Battle Cards на основе анализа рынка, конкурентов и обратной связи от продаж.

Это связующее звено между стратегией компании и тактикой отдела продаж.

ПРАКТИКА

УПРАЖНЕНИЕ // РАБОЧЕЕ ПРОСТРАНСТВО

Рабочее пространство агента

Проект — это папка на диске, в которой агент хранит знания, рабочие материалы и создаёт артефакты.

```
battle-cards/
├── CLAUDE.md
│   инструкции и контекст агента
├── research/
│   собранные данные и исследования
├── output/
│   готовые battle cards
└── templates/
    шаблоны и формы документов
```

01 Открываем VS Code

02 Нажимаем **Open Folder**

03 Создаём новую папку **battle-cards**

04 Открываем папку — это наш проект

Папка = проект = память и рабочее пространство агента

06 // ПРАКТИКА

15 мин

Давайте попробуем эту магию на практике

01 Открываем агентную платформу Claude Code (VS Code).

02 Создаем новый проект

03 В чат пишем промпт из окна справа и запускаем создание навыка.

04 Даем задание ИИ-агенту на формирование battle-card для какого либо продукта.

prompt.txt

Промпт:

Мне необходимо создать **навык** для подготовки поддержания в корректном состоянии **battle cards** для продуктов нашей компании.

Навык по запросу для определенного нашего продукта должен проводить исследование продуктов конкурентов, на основании результатов исследования создавать или актуализировать **battle card** для продукта.

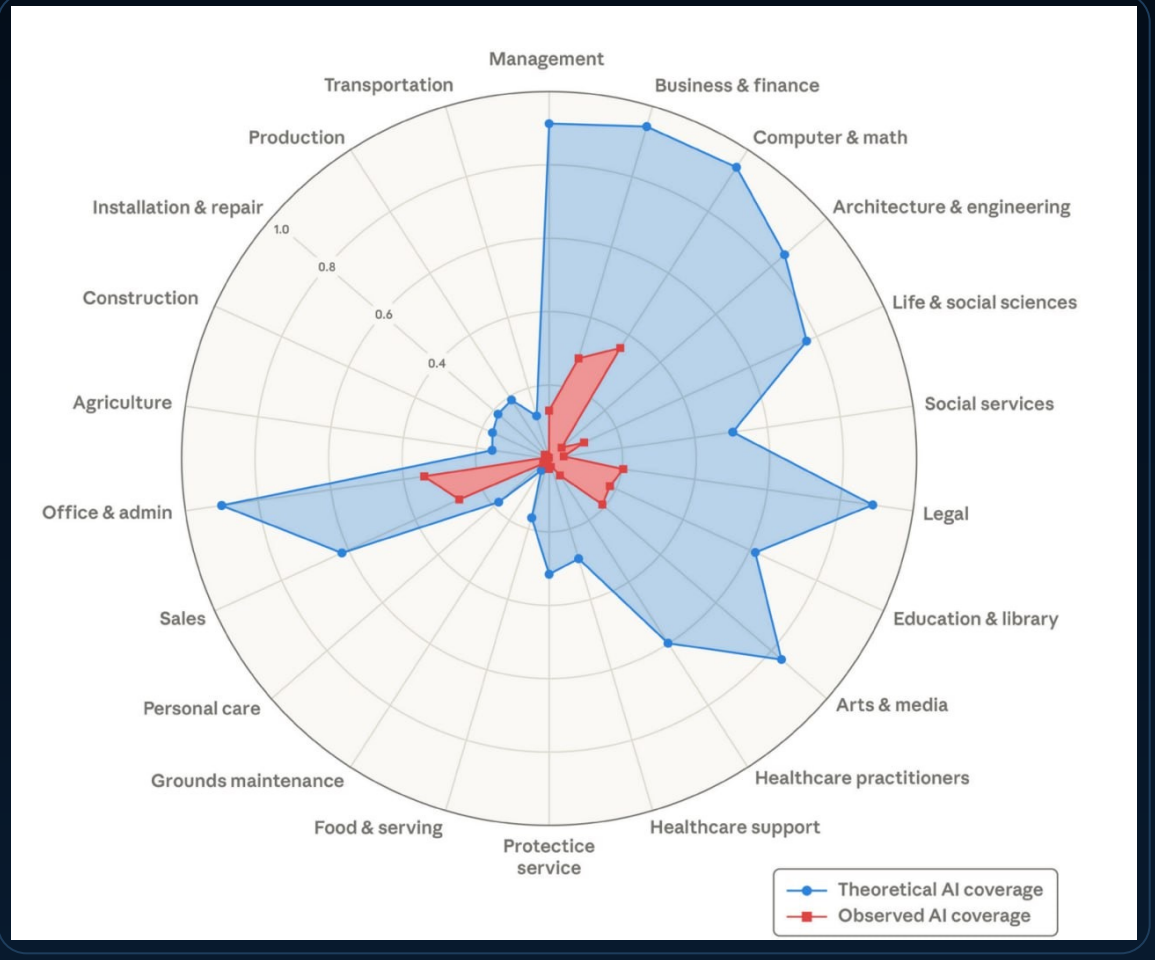
03 // СТАТИСТИКА

AI покрытие по профессиям

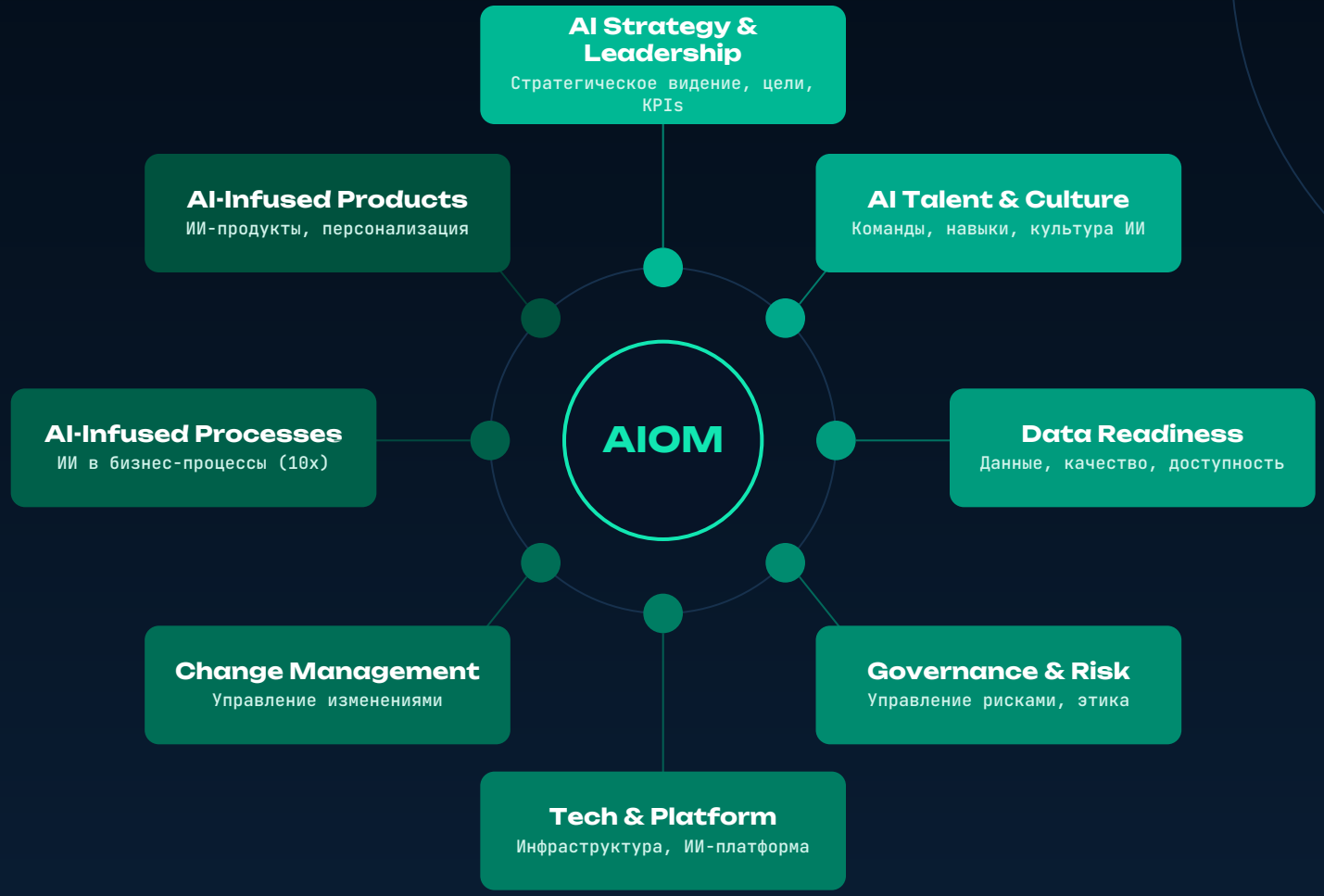
Потенциальное и наблюдаемое покрытие ИИ по категориям профессий

Легенда:

- Потенциальное покрытие AI
- Наблюдаемое покрытие AI



AI Operating Model



06 // КЕЙСЫ АГЕНТНОГО ИИ

Несколько кейсов использования агентного ИИ

01

Коммерческие предложения

Гиперперсонализированные КП на основе транскрипта встречи и шаблона Word

02

Финансовый директор

Рабочее пространство: агент собирает отчётность и ведёт управленческий учёт

03

Ландшафтный дизайн

Дендрологическая ведомость: человек — 5 дней, агент — 2 часа

04

Шаблоны документов

Разработка презентаций, КП, лендингов и другого контента

05

Аналитика для бизнеса

Маржинальность продуктов и ингредиентов — то, что не умеет учётная система

06

Личная эффективность

Ассистент, планирование, работа с календарём — всё прямо из Telegram

Три варианта использования агентов

Рабочая среда

Агент как постоянный помощник в ежедневной работе — обработка почты, поиск информации, работа с файлами, работа с CRM и т.п.

Регулярная задача

Автоматизация повторяющихся процессов — подготовка КП, маркетинговые активности и другие регулярные задачи

Проект

Агент для реализации конкретного проекта – разработка продукта, создание приложения и другие проекты.

07 // СТРУКТУРА

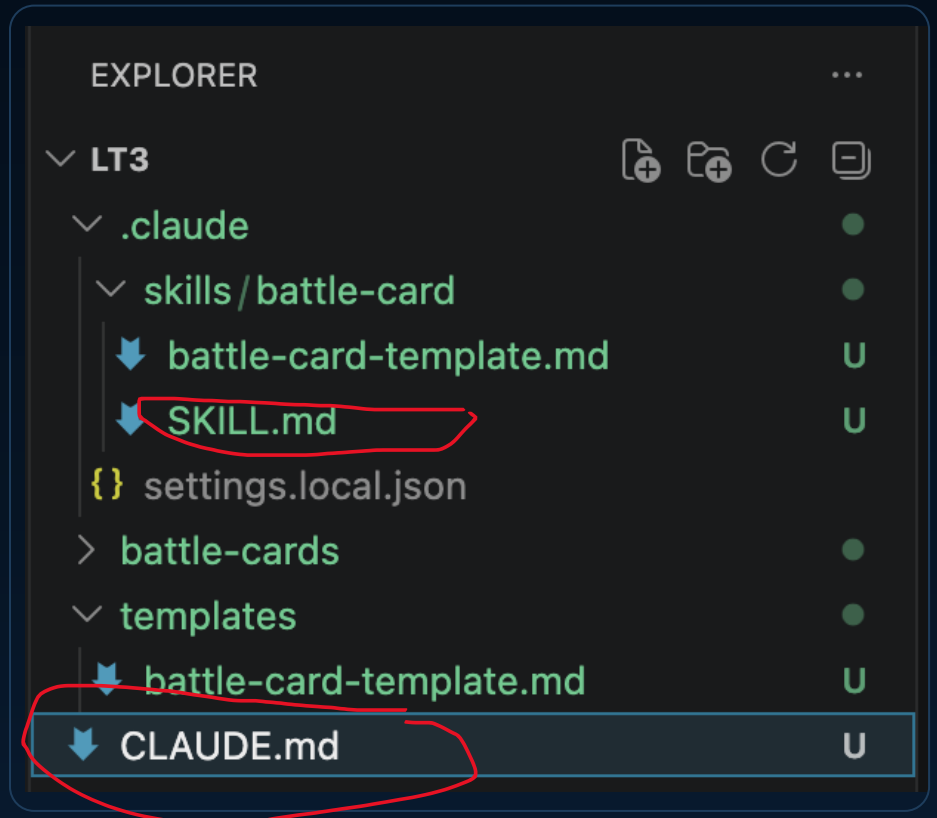
Из чего состоит наш агент

01 CLAUDE.md (AGENTS.md)

Описание вашего проекта, для чего
он предназначен и как в нём работать

02 SKILL.md (в папке skills/battle-cards)

Описание навыка агента, который исследует
и формирует battle-cards



Структура проекта

07 // СТРУКТУРА

Описание проекта
CLAUDE.md (AGENTS.md)

01 Простое, человечное описание проекта — чтобы агенты быстро схватывали, о чём это и какие тут правила

02 Обычный текст, без специального синтаксиса — пишит е так, как описываете для нового сотрудника

03 Всё точно также, как если бы вы подключали нового сотрудника на проект

CLAUDE.md > # Battle Cards — конкурентная аналитика

```
# Battle Cards — конкурентная аналитика

Проект для создания и поддержания актуальных battle cards по продуктам компании.

## Структура проекта

- `battle-cards/` — готовые battle cards (Markdown)
- `templates/battle-card-template.md` — шаблон battle card (копия также внутри навыка)
- `.claude/skills/battle-card/SKILL.md` — навык Claude Code для генерации/обновления battle cards
- `.claude/skills/battle-card/battle-card-template.md` — шаблон, доступный навыку через ${CLAUDE_SKILL_DIR}

## Использование

Запустите навык через Claude Code:

```
/battle-card <Название продукта> — <краткое описание продукта и компании>
```

Примеры:
```
/battle-card Notion — инструмент для заметок и управления проектами от Notion Labs
/battle-card Slack — корпоративный мессенджер от Salesforce
```

Правила

- Battle cards создаются на русском языке
- Формат: Markdown
- Данные получаются через веб-поиск (WebSearch/WebFetch)
- Все факты должны быть подкреплены источниками
- При обновлении существующей карты сохраняется история изменений
```

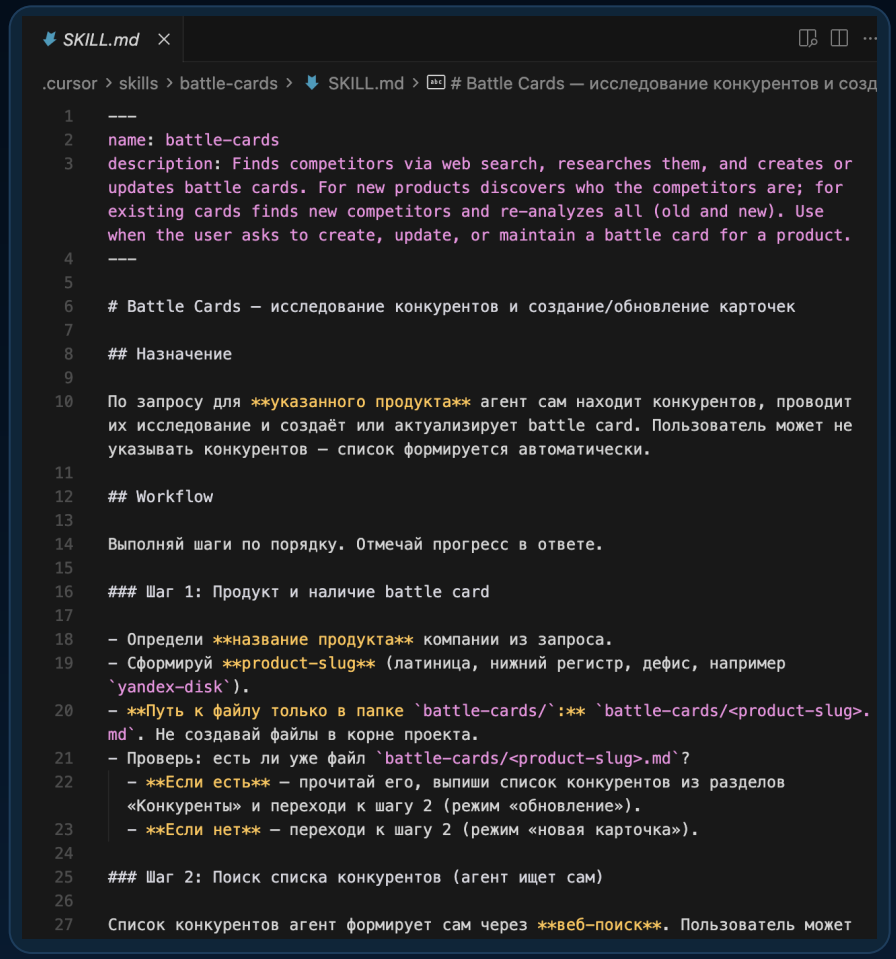
07 // СТРУКТУРА

Описание навыка SKILL.MD

01 Это как рабочая инструкция сотрудника для выполнению определенного типа задач (запуск рекламной компании, закрытие месяца и т.п.).

02 Любые сотрудники (агенты), которых вы подключаете к проекту будут использовать эти инструкции при выполнении соответствующих задач.

03 Это также простой текст, но есть небольшие требования к его оформлению (https://agentskills.io/specification).



07 // НАВЫКИ АГЕНТА

Навыки агентов

01

Базовые навыки

По умолчанию агент обладает навыками, необходимыми для разработки ПО.

02

Расширение

Навыки можно дополнять, добавлять специфичное поведение, встраивать в определённые точки процесса.

03

Пример

Можно встроить ревью безопасности после завершения реализации кода.

```
# skill.md
---
name: security-review
trigger: post-implementation
---

## instructions
After code implementation:
  1. Check vulnerabilities
  2. Verify auth flows
  3. Run SAST scan
  4. Report findings
```

12 // НАВЫКИ АГЕНТОВ

Существует множество библиотек навыков

01

SDLC навыки

Набор навыков для полного цикла разработки ПО

github.com/obra/superpowers

02

Анализ уязвимостей кода

Навык от Trail of Bits для автоматического поиска уязвимостей в коде

github.com/trailofbits/skills

03

Маркетинг, проджект-менеджмент, финансы, регуляторка

Универсальная коллекция навыков для бизнес-функций

github.com/alirezarezvani/claude-skills

04

60+ навыков для продакта

Коллекция навыков для продакт-менеджеров: аналитика, стратегия, roadmap

github.com/phuryn/pm-skills

12 // МЕТОДОЛОГИЯ

Создание → Запуск → Ретроспектива

01
Задание на создание агента

Просим агента создать агента под ваше описание задачи или проекта

02
Запуск и коррект. результата

Даем агенту конкретную задачу, смотрим результат, просим скорректировать, если нужно

03
Ретроспектива агента

Просим агента скорректировать свои инструкции на основании полученного опыта

04 Подключаем агенту суперспособности

Работа с MCP, анализ данных Excel/Google Sheets, презентации и многое другое

10 // СУПЕРСПОСОБНОСТЬ №0

Голосовой ввод для работы с ИИ-агентами

3-5x быстрее

Голос ~150 WPM vs клавиатура ~40 WPM

4% ошибок

При голосовом вводе в тихой среде (vs 8% при печати)

★ РЕКОМЕНДАЦИЯ

Wispr Flow

Mac, Win, iOS, Android. Точность 95-97%.
Code-switching рус/англ. Интеграция с IDE.

SuperWhisper

Offline + свои API-ключи

Paraspeech

100% offline, Mac

bit.ly/letu-flow1

Настройка прав доступа для агента

Принцип

Read-only команды выполняются без вопросов, деструктивные — заблокированы, всё остальное — с подтверждением

Промпт для агента

```
Настрой permissions для этого проекта. Создай
.claude/settings.json. Принцип: 1) ALLOW - все read-only:
Read, ls, cat, head, tail, find, grep, wc, pwd, git
status/log/diff/branch/show, docker ps/images/logs. 2) ASK -
изменяющие: Edit, git add/commit/push/pull, cp, mkdir, touch,
sed -i, npm/pip install, curl, docker run/exec/build. 3) DENY
- деструктивные: rm, rmdir, mv, chmod, chown, sudo, su, dd,
kill, shutdown, ssh, scp, чтение .env/secrets. После создания
покажи итог и объясни, что адаптировать под проект.
```

`.claude/settings.json` – Безопасные права по умолчанию

json

```
{
  "permissions": {
    "allow": [
      "Read",
      "Bash(ls *)",
      "Bash(cat *)",
      "Bash(head *)",
      "Bash(tail *)",
      "Bash(find *)",
      "Bash(grep *)",
      "Bash(wc *)",
      "Bash(file *)",
      "Bash(pwd)",
      "Bash(whoami)",
      "Bash(echo *)",
      "Bash(which *)",
      "Bash(env)",
      "Bash(printenv *)",
      "Bash(tree *)",
```

.claude/settings.json

Начнем с простого

01

Создаем новый проект.

02

В чате пишем промпт:

prompt.txt

Промпт:

Мне необходимо создать **лэндинг** для [тут ваше описание, которое проще наговорить голосом].

Сначала необходимо разработать **концепт hero-лэндинга** и дизайна.

autonomous

COMPANY

04 // ВОПРОС

Почему получилась ф..гня, а не дизайн?

Разберёмся, почему ИИ выдаёт шаблонный результат
и как этого избежать

Наиболее вероятный ответ – правильный для ИИ, но не для нас.

// *щепотка агентной мудрости*

10 // СУПЕРСПОСОБНОСТЬ №1

«Маловероятно» — значит красиво

01 Дописываем исходный промпт в новом чате текстом (справа), который уводит модель в «непопсовые» варианты.

02 Далее запускаем в работу.

03 Далее просим «Сделай ещё 5 более арт-директорских варианта».

04 Продолжаем до того момента, пока не получите понравившийся вам вариант.

prompt.txt

Промпт:

```
You tend to converge toward generic, "on distribution"
outputs. In frontend design, this creates what users
call the "AI slop" aesthetic.

Avoid this: make creative, distinctive frontends that
surprise and delight.

Focus on: Typography: Choose fonts that are beautiful,
unique, and interesting...
```

06 // ПОЛЕЗНЫЕ МАТЕРИАЛЫ

system_prompt.md

Промпт:

You tend to converge toward generic, "on distribution" outputs. In frontend design, this creates what users call the "AI slop" aesthetic. **Avoid this**: make creative, distinctive frontends that surprise and delight.

Typography: Choose fonts that are beautiful, unique, and interesting. Avoid generic fonts like Arial and Inter. **Color & Theme**: Commit to a cohesive aesthetic. Dominant colors with sharp accents outperform timid palettes. **Motion**: Use animations for effects and micro-interactions. Prioritize CSS-only solutions.

Backgrounds: Create atmosphere and depth rather than defaulting to solid colors. Layer CSS gradients, use geometric patterns.

Avoid: Overused font families (Inter, Roboto, Arial) • Clichéd color schemes (purple gradients) • Predictable layouts. **Think outside the box!**

Этот промпт помогает ИИ создавать уникальный дизайн вместо шаблонного.
Добавьте его в системные инструкции вашего агента для улучшения визуального вывода.

«Маловероятная техника» работает для любых задач — шаблоны презентаций и документов, написание текстов, постов и тд.

Нужно лишь попросить ИИ адаптировать промпт под эту цель.

// *щепотка агентной мудрости*

07 // ТЕОРИЯ

Как агент может выполнять работу в различных системах

01

MCP

ИИ-адаптеры, которые позволяют работать агенту за вас в различных системах: таск-трекеры, таблицы, CRM и т.п.

02

API

При необходимости агент легко может сам написать код для интеграции с нужным вам приложением.

03

Навыки

Есть набор существующих подключаемых навыков для работы с различными системами: PowerPoint, Excel и т.п.

04 // ВОПРОС

Сложно?

Что делать, если не знаешь как действовать?

01
Спрашиваем у ИИ, как сделать

02
Говорим ему сделать самому.

03
Сами делаем по минимуму.

06 // ПРАКТИКА

30 мин

Пробуем навыки для Excel

01

Создайте агента, который на основании данных из файла по продаже недвижимости в Манхэттене построит дашборд о ситуации на рынке.

02

Дашборд должен быть оформлен с формулами. Попросите агента использовать совместимые формулы.

03

Таблица должна иметь красивый дизайн и структуру.

12 // АРХИТЕКТУРА

Что такое Skills?

ОПРЕДЕЛЕНИЕ

Навык — это инструкции для выполнения специфического типа задач (формирование фин модели, создание постов для соцсети и т.п), которые также могут содержать исполняемый код, шаблоны и другие элементы.

ФИЗИЧЕСКИ

Физически навык представляет собой папку **skill-название**

Код | Шаблоны | Документация

```
$ tree skill-name/

skill-name/

├── SKILL.md
# Обязательно: метаданные + инструкции

├── scripts/
# Опционально: программный код

├── references/
# Опционально: документация

├── assets/
# Опционально: шаблоны, ресурсы

└── ...
# Опционально: другие файлы
```

Как агент работает с навыками

01

Анализ задачи

Агент определяет что нужно сделать, в каком инструменте, какой результат ожидается

02

Поиск навыка

Сопоставляет задачу с библиотекой skills и их описаний по ключевым словам и ожидаемому результату

03

Чтение SKILL.md

Открывает описание навыка, читает инструкцию, ограничения и рекомендуемый порядок действий

04

Применение

Выполняет задачу по инструкции навыка с учётом контекста и ограничений

Поиск навыка идёт по типу работы, ключевым словам и ожидаемому результату — нужен ли анализ данных, работа с таблицей, презентацией или документом

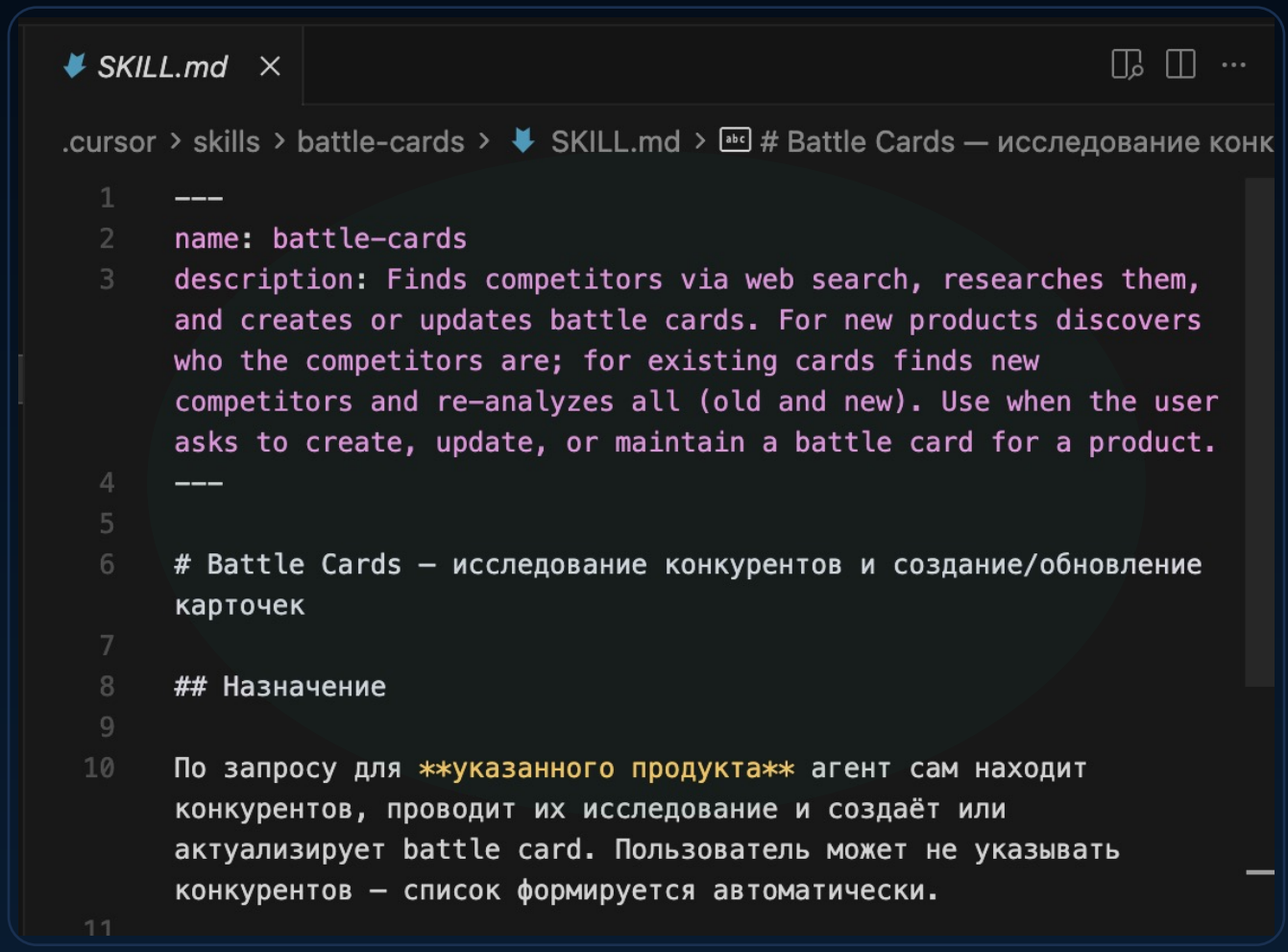
Описание SKILL.md

Уделяйте внимание правильному наименованию и описанию навыка, чтобы ИИ-агенты находили навыки и применяли их в соответствующих ситуациях

name
название навыка

description
краткое описание

instructions
порядок действий



Работа с контекстом агента

Контекст заполняется

ВАЖНО!

Фиксация опыта

До компактизации попросите агента сохранить успешный опыт и важные детали в `AGENTS.MD`

Компактизация

ИИ сжимает контекст, детали переписки исчезают, остаётся только общее саммари

✓ Сохраняется в AGENTS.MD

- → Успешные приёмы работы агента
- → Важные решения и контекст проекта
- → Структура файлов и зависимости
- → Системный промпт к каждому запросу

✕ Теряется при компактизации

- → Детали переписки с агентом
- → Конкретные примеры и нюансы
- → Промежуточные рассуждения ИИ
- → Остаётся только общее саммари

05 Собираем агентов для своих кейсов

Фиксация успешных сценариев,
развитие навыков агентов, настройка инструкций

06 // ПРАКТИКА

60 мин

Делаем свой кейс

01

Выбираем кейс из своей работы. Какой-то регулярный процесс, либо проект, либо подключаем Claude к одной из рабочих систем.

02

В режиме Plan Mode делаем описание контекста и просим создать концепт ИИ-агента под эту задачу.

03

Корректируем план при необходимости. Запускаем реализацию и подключаем необходимые MCP.

08 // ПРАКТИКА

Как начать создание агента

01 Выбираем задачу
Определяем конкретную задачу, которую хотим автоматизировать с помощью агента

02 Определяем системы
Какие системы задействованы в этой задаче: CRM, таск-трекер, почта и т.д.

03 Способ доступа
Для каждой системы определяем: есть ли MCP, API или это локальные файлы

04 Локальные файлы?
Если данные в локальных файлах — отлично, переходим сразу к шагу 6

05 Настраиваем подключение
Настраиваем MCP или просим ИИ организовать получение данных по API

06 Даём задания агенту
Можно давать любые задания агенту по работе с подключёнными системами

06 AI-Native разработка

Как получать стабильный и качественный результат в реальных рабочих сценариях

Сначала безопасный пример, потом — перенос на свою реальность

Если сразу идти в существующий корпоративный код, почти всегда начинается одно и то же:

« у нас слишком сложная архитектура »

« у нас легаси, тут это не взлетит »

« у нас особые ограничения »

« агент тут не разберётся »

Поэтому: берём безопасный пример → понимаем логику работы → переносим на свои задачи

AI-Native процесс

Intent → Сборка контекста → Исполнение агента → Автоматическая / валидация человеком → Отгрузка / handoff → Observe → Ретроспектива и обновление агента

Что меняется

01 **Требования**
становятся менее «замороженной фазой» и больше переходят в итеративное уточнение

02 **Архитектура**
частично переносится в живой диалог с агентом, но архитектурная ответственность не исчезает

03 **Имплементация и тестирование**
сильно сближаются, реализуются за счет создания кода ИИ-агентом

04 **Review**
сдвигается от ручного чтения к risk-based и exception-based контролю

05 **Observability**
становится не «последним этапом», а частью управляющего контура

Плагины Claude Code

Расширения, которые добавляют slash-команды, агентов, хуки и MCP-серверы. Переиспользуются между проектами и командами.

Code Intelligence LSP

Агент видит ошибки типов и пропущенные импорты сразу после редактирования и исправляет их в том же ходе.

`pyright-lsp` · `typescript-lsp` · `gopls-lsp`
`rust-analyzer-lsp` · `kotlin-lsp` и ещё 8

Внешние интеграции

Преднастроенные MCP-серверы для внешних сервисов. Устанавливаются одной командой.

Код github · gitlab
Управление jira · linear · notion · asana
Инфра vercel · firebase · supabase · railway
Мониторинг sentry · posthog · pagerduty

Рабочие процессы

Команды и агенты от Anthropic для типовых задач разработки.

`code-review` — автоматическое ревью PR
`feature-dev` — полный цикл фичи
`ralph-loop` — итеративная разработка
`playground` · `frontend-design` · `plugin-dev`

Установка и структура

`/plugin install <name>`

Scopes user · project · local
Состав commands/ · agents/ · hooks/ · skills/
Маркетплейс официальный + командный через .claude/settings.json
Hot reload /reload-plugins

Безопасность: плагины выполняют произвольный код с правами пользователя. Проверяйте исходный код перед установкой.

Установка /plugins

```
❯ /plugin

Plugins  Discover  Installed  Marketplaces  Errors

Discover plugins (1/140)

⌕ Search…

❯ ○ frontend-design · claude-plugins-official · 507.5K installs
    Create distinctive, production-grade frontend interfaces wi…

  ○ superpowers · claude-plugins-official · 410.3K installs
    Superpowers teaches Claude brainstorming, subagent driven d…

  ○ context7 · claude-plugins-official [Community Managed] · 248.6K installs
    Upstash Context7 MCP server for up-to-date documentation lo…

  ○ code-review · claude-plugins-official · 232.6K installs
    Automated code review for pull requests using multiple spec…

  ○ code-simplifier · claude-plugins-official · 194.4K installs
    Agent that simplifies and refines code for clarity, consist…
↓ more below

type to search · Space to toggle · Enter to details · Esc to back
```

Рекомендуемые плагины:

- **superpowers**
  Мозговой штурм, суб-агенты, глубокий анализ
- **context7**
  Актуальная документация библиотек из Upstash
- **skill-creator**
  Создание и настройка кастомных скиллов
- **firecrawl**
  Веб-скрейпинг и парсинг страниц для контекста

Команда **/plugins** → Discover → Space → Install

Сначала intent, а не код

01

intent

Первый артефакт AI-native цикла — не код.

- **что** — хотим сделать
- **для кого** — это нужно
- **зачем** — мы это делаем
- **что считаем** — успехом
- **что не входит** — в скоуп

РЕЗУЛЬТАТ **vision.md**

02 // ПРАКТИКА

Практика: собираем первый артефакт

01

ЗАДАНИЕ

С помощью агента подготовить короткий документ задачи

→ vision.md

02

В документе зафиксировать:

- что хотим получить
- кто будет использовать
- основной сценарий использования
- как будем оценивать успешность

06 // ПРАКТИКА

20 мин

Собираем vision.md

01

Создаем новый проект.

02

Просим агента подготовить видение проекта в файле vision.md

03

Сгружаем агенту контекст: что хотим получить, кто будет использовать, основные сценарии использования, как будем оценивать успешность

04

Проверяем видение и корректируем.

Контекст собран — включаем Plan Mode

02

context & planning

Собираем контекст и агент строит план изменений.

- **контекст** — собираем docs, код, зависимости проекта
- **Plan Mode** — агент декомпозирует задачу на шаги
- **зависимости** — находит связи между компонентами
- **риски** — выявляет потенциальные проблемы заранее
- **согласование** — вы утверждаете план до начала кода

РЕЗУЛЬТАТ детальный план задач, готовый к исполнению

03 // CONTEXT

Формируем рабочую среду агента

CLAUDE.md
AGENTS.md

входная точка

docs/

знания проекта – история реализации,
даем ссылку на них в CLAUDE.md

архитектура
заметки по архитектуре

API и интеграции
как и куда ходим

требования
что нужно сделать

глоссарий
термины и общий язык

среда
ограничения и правила

история реализации
документация по принятым решениям

Это и есть рабочее окружение, с помощью которого агент будет разбираться в проекте.

06 // ПРАКТИКА

15 мин

Собираем контекст docs/

01

Вгружаем в папку docs сущестсвующие знания по проекту в текстовом формате.

02

Это могут быть транскрипты, требования, описания интеграций, сценарии тестирования и другие источники.

Структура документации проекта

```
my-project/
├── .claude/
│   └── settings.json        # plansDirectory: "./docs/plans"
├── docs/
│   ├── plans/               # Планы из Plan Mode
│   │   ├── auth-refactoring.md
│   │   ├── api-v2-migration.md
│   │   └── perf-optimization.md
│   ├── decisions/           # ADR (Architecture Decision Records)
│   │   ├── 001-auth-approach.md
│   │   └── 002-caching-strategy.md
│   ├── architecture.md      # Обзор архитектуры, ключевые решения
│   ├── api.md               # Описание API, эндпоинты, форматы
│   └── data-model.md        # Модель данных, схемы БД
├── src/
└── CLAUDE.md                # Персистентный контекст агента
```

04 // PLANNING

Plan mode — центр AI-native разработки

Качество рождается не в процессе реализации, а в plan mode.

01
Сканирует документы, код, интернет

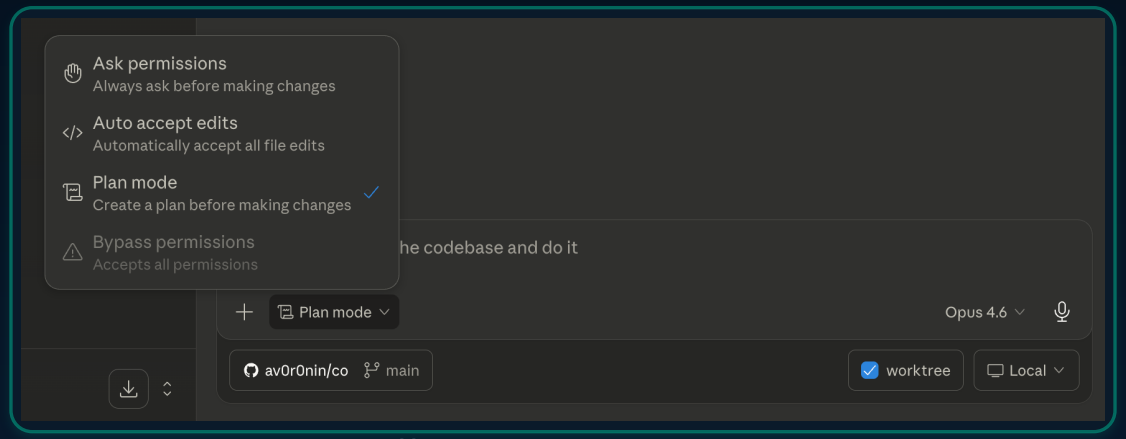
02
декомпозирует работу

03
находит зависимости

04
выявляет риски

05
продумывает тестирование

06
предлагает подход



// plan mode interface

04 // PLANNING

Цикл планирования

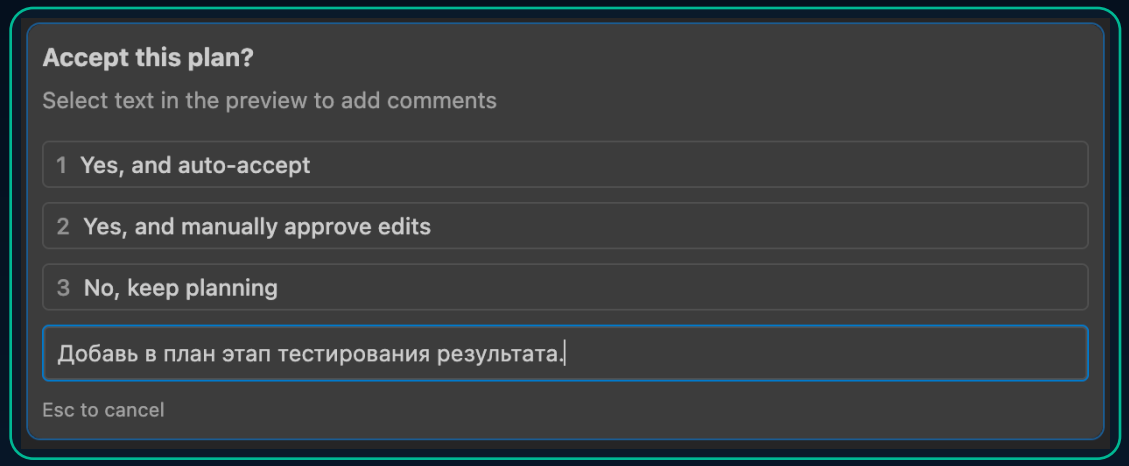
Итеративная корректировка плана до идеального результата

01 Запуск план-режима
Формулируем задачу и запускаем Claude в plan mode

02 Ревью плана
Изучаем предложенный план, оцениваем шаги и подход

03 Корректировка
Уточняем план через фидбек, добавляем этапы, правим детали

↻ Повторяем до полного устраивающего результата



`"Accept this plan?"` — ключевой момент цикла

1 `auto-accept` — Принять и запустить автоматически

2 `approve edits` — Принять, но подтверждать каждое изменение

3 `keep planning` — Вернуться к планированию с фидбеком

06 // ПРАКТИКА

30 мин

Циклы детального планирования

01

Включаем Plan Mode

02

В чате пишем промпт и итерируем пока план нас не будет устраивать.

prompt.txt

Промпт:

```
Хотим реализовать vision.md, предложи реализацию.
```

autonomous
company

Агент пишет код

03

execution

Агент пошагово реализует план.

- **запуск** — агент начинает с первой задачи из плана
- **код** — генерируется, тестируется, коммитится
- **ревью** — вы проверяете diff
- **итерации** — правки через диалог, не ручной рефакторинг

РЕЗУЛЬТАТ рабочий код + тесты в ветке проекта

07 // РЕАЛИЗАЦИЯ 30 мин

Самое вкусное

запускаем реализацию

autonomous
company

Агент пишет тесты

04

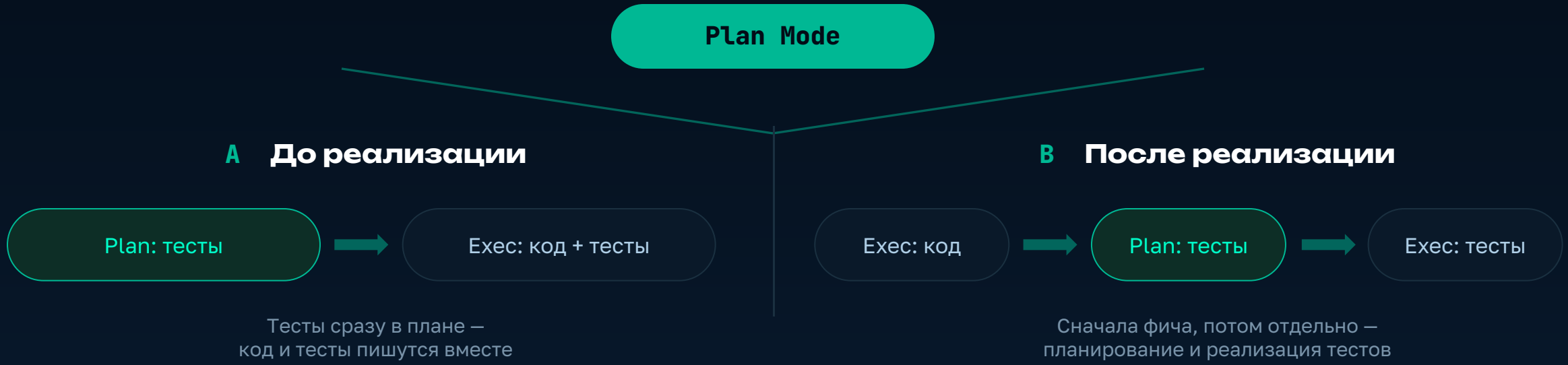
testing

Агент генерирует тесты по коду, сценариям и контексту фичи.

- **анализ** — агент читает код и понимает, что тестировать
- **генерация** — unit, integration, e2e-тесты по сценариям
- **валидация** — QA проверяет покрытие и краевые случаи
- **доработка** — правки через промпт, не ручное написание

РЕЗУЛЬТАТ тестовое покрытие + зелёный CI в ветке проекта

Когда закладывать тесты?



Оба подхода валидны — в тренинге идём по варианту B

06 // ПРАКТИКА

30 мин

Разработка тестов

01

Включаем Plan Mode

02

В чате пишем промпт и итерируем пока план нас не будет устраивать.

prompt.txt

Промпт:

Подготовь **тестовые сценарии** и задачи для их **реализации**.

autonomous
company

Поддержание документации в актуальном состоянии

Когда обновлять документацию

При изменении API или интерфейсов

Обновите docs/ при любых изменениях в эндпоинтах, параметрах запросов или форматах ответов

При добавлении новой фичи

Создайте обзор, пошаговое руководство и FAQ для каждой новой функциональности

При рефакторинге архитектуры

Пересоберите architecture.md когда меняется структура модулей или зависимости

При исправлении критических багов

Обновите FAQ и troubleshooting когда найдены неочевидные ошибки и их решения

Обновление архитектурной документации

```
Прочитай текущий docs/architecture.md и актуальную структуру
проекта. Обнови документ, чтобы он отражал реальное состояние
кодовой базы. Указывай конкретные пути к файлам,
а не абстрактные описания. Уложись в 200 строк.
```

Генерация документации по коду

```
Исследуй кодовую базу и разберись, как реализована фича [название].
Сгенерируй пользовательскую документацию в docs/:
- обзор возможностей
- пошаговое руководство
- часто задаваемые вопросы
Пиши для пользователя, не для разработчика.
```

Создание документации новой фичи

```
Прочитай все файлы в src/routes/ и сгенерируй документацию API.
Для каждого эндпоинта укажи: метод, путь, параметры, формат ответа.
Не описывай что делает код построчно - объясни зачем
выбран такой подход и что нужно знать перед изменением.
```

Документация — живой артефакт: обновляется вместе с кодом, а не раз в квартал.

Агент выкатывает релиз

05

deploy

Агент готовит, проверяет и доставляет изменения в прод.

- **сборка** — агент запускает CI-пайплайн и следит за статусом
- **чеклист** — миграции, конфиги, feature-флаги проверены
- **канарейка** — поэтапный раскат с мониторингом метрик
- **откат** — автоматический rollback при аномалиях
- **отчёт** — changelog, уведомления и саммари в канал команды

РЕЗУЛЬТАТ фича в проде + мониторинг стабильности

Нюансы деплоя при разработке с агентами

Правила агента

`CLAUDE.md`

Безопасность

Не коммить секреты, API-ключи, токены. Переменные окружения через .env. Не логировать чувствительные данные.

Стек

Bearer tokens в headers, ORM — SQLAlchemy, сырые SQL запрещены. Валидация входных данных через Pydantic.

Зависимости

Не добавлять без явного запроса. Только из approved-deps.md.

Качество кода

Функции ≤ 30 строк. Docstrings для публичных API. Тесты на каждый эндпоинт.

Защита инфраструктуры

Git

Branch protection: агент не может пушить в main. Обязательный PR, авто-ветки.

Секреты

Secrets scanning на каждый коммит. GitHub Advanced Security / GitLeaks.

SAST

Статический анализ до мёржа: SonarQube, Snyk, Semgrep.

Permissions

Минимальные права агента: read-only к prod, write только в feature-ветки.

Субагенты для ревью

`.claude/agents/`

`security-reviewer`

Сканирует на секреты, SQL-инъекции, небезопасные зависимости

`code-reviewer`

Качество, стиль, покрытие тестами, actionable feedback

`docs-checker`

Обновлена ли документация: API docs, CHANGELOG, README

`migration-reviewer`

Валидирует миграции БД: обратимость, downtime, совместимость

`PIPELINE`

Агент пишет → Субагент проверяет → CI/CD сканирует → Человек ревьюит

AI-агент — мощный инструмент, но деплоить его результат без ревью = деплоить баги в production.

Агент проверяет сценарии

06

e2e testing

Агент прогоняет пользовательские сценарии от начала до конца.

- **сценарии** — агент строит матрицу пользовательских путей
- **автоматизация** — генерирует e2e-тесты на Playwright / Cypress
- **прогон** — запуск в CI на staging-окружении
- **диагностика** — при падении агент находит причину и правит
- **отчёт** — покрытие сценариев, скриншоты, логи прогона

РЕЗУЛЬТАТ зелёные e2e-тесты + готовность к релизу

Нюансы тестирования при разработке с агентами

> *100% coverage за минуту — но покроет ли это то, что действительно сломается?*

Specification-first

Тесты проверяют **требования**, а не реализацию. Давайте агенту контекст: бизнес-правила, acceptance criteria, граничные условия, формат ответов API.
Без явного указания агент не покроет: `null`, пустые коллекции, граничные значения, ошибочные условия.

Mutation Testing

AI-агент может писать тесты, которые **«зелёные», но ничего не проверяют**.
Тавтологичные ассерты, хардкод в фикстурах.
`mutmut` / `Stryker` — мутирует код и проверяет, что тесты ломаются. Мера реального покрытия.

WORKFLOW

1. **Агент пишет тесты**
   По промпту с контекстом требований
2. **Субагент ревьюит**
   test-reviewer проверяет допущения, edge cases, ассерты
3. **CI/CD прогоняет**
   pytest --randomly + mutation + coverage gate
4. **Человек проверяет**
   Соответствие бизнес-приоритетам. Что тестировать важнее?

Новая роль QA

Не автор тест-кейсов, а **AI-супервизор**: определяет цели качества, ревьюит результаты агента, контролирует соответствие бизнес-приоритетам.

07 Продвинутые техники AI-Native разработки

Как получать стабильный и качественный результат в реальных рабочих сценариях

claude-code-setup Plugin

Read-only анализ кодовой базы

Изучает структуру проекта, зависимости и конфигурацию. Выдаёт персонализированные рекомендации по 1–2 автоматизации в каждой категории. Не создаёт и не изменяет файлы.

```
/plugin install claude-code-setup@claude-plugins-official
```

Decision Framework

Внешние API / БД в зависимостях	→ **MCP-сервер**
Повторяющиеся промпты / воркфлоу	→ **Skill**
Рутинные пост-редакционные действия	→ **Hook**
Нужно специализированное ревью	→ **Subagent**
Частые Git-операции	→ **Command**

MCP-серверы

context7, Playwright, GitHub, Linear, Slack, Postgres, Supabase

Skills

Plan agent, frontend-design, генерация документов. invocation control: disable-model / user-only

Hooks

auto-format, auto-lint, block sensitive files. События: PreToolUse, PostToolUse, Stop, SessionStart

Subagents

security reviewer, performance reviewer, accessibility, code reviewer — делегированные ревью

Slash Commands

/test, /pr-review, /explain, /commit — кастомные команды для частых операций

Отправная точка: от рекомендации к реализации — «хотите, помогу настроить?»

Code Review Plugin

/code-review → 8 шагов

ПОДГОТОВКА

1. **Проверка применимости** draft? закрыт? тривиальный?
2. **Сбор CLAUDE.md** стандарты проекта из затронутых директорий
3. **Резюме PR** краткое описание изменений для контекста

АНАЛИЗ

4. **5 Sonnet-агентов параллельно** → см. правую колонку
5. **Confidence scoring** независимая оценка 0–100 для каждого замечания

ФИЛЬТРАЦИЯ И ПУБЛИКАЦИЯ

6. **Фильтрация < 80** только замечания с высокой уверенностью
7. **Повторная проверка** PR всё ещё открыт и актуален?
8. **Публикация в PR** комментарий через gh CLI со ссылками на строки

ШАГ 4 5 Sonnet-агентов параллельно

- **#1 CLAUDE.md** Нарушают ли изменения правила проекта?
- **#2 Поиск багов** Поверхностный скан на очевидные ошибки
- **#3 git blame** Баги с учётом истории изменений файлов
- **#4 Предыдущие PR** Комментарии из PR, затрагивавших те же файлы
- **#5 Комментарии** Соответствуют ли изменения TODO/FIXME в коде?

ШАГ 5–6 Confidence scoring + фильтрация

отсеивается → ← публикуется

Что не считается проблемой:

- Проблемы, существовавшие до PR
- То, что поймает линтер или тайпчекер
- Придирчивые мелочи и стиль кода
- Изменения на строках, которые не трогал автор

Минимум 8 агентов на PR: 3 Haiku (подготовка) + 5 Sonnet (анализ) + N Haiku (scoring каждого замечания)

feature-dev Plugin

```
/plugin install feature-dev@claude-plugins-official
```

7-фазный воркфлоу: от анализа кодовой базы до ревью качества. Human-in-the-loop на ключевых точках.

H = human-in-the-loop

1	2	H 3	H 4	H 5	H 6	7
Discovery Уточнение задачи	**Exploration** 2-3 агента code-explorer	**Questions** Уточняющие вопросы	**Architecture** 2-3 агента code-architect	**Implement** Реализация по плану	**Review** 3 агента code-reviewer	**Summary** Отчёт и рекомендации

code-explorer

Исследование кодовой базы

- Точки входа: API, UI, CLI
- Цепочки вызовов от входа к выходу
- Паттерны и архитектурные решения
- Зависимости и побочные эффекты
- Результат: 5-10 ключевых файлов

Фазы 2 | 2-3 параллельно

code-architect

Проектирование архитектуры

- 3 подхода: Minimal / Clean / Pragmatic
- Конкретные файлы и интерфейсы
- План реализации (build sequence)
- Обработка ошибок и тестируемость
- Сравнение с компромиссами

Фазы 4 | 2-3 параллельно

code-reviewer

Ревью качества кода

- Simplicity / DRY / Elegance
- Баги: null, race conditions, утечки
- Соответствие CLAUDE.md
- Confidence scoring 0-100
- Порог публикации: ≥ 80

Фазы 6 | 2-3 параллельно

Сначала понять кодовую базу, потом менять её. Архитектура до реализации, ревью до деплоя — AI следует инженерным практикам.

Context7 MCP

```
npx ctx7 setup
```

Актуальная документация из первоисточников прямо в контекст AI-агента. Решает проблему устаревших данных из training data.

ПРОБЛЕМА

- Галлюцинации: несуществующие API и методы
- Deprecated-паттерны из устаревшего training data
- Ошибки при работе с новыми версиями библиотек

▼

РЕШЕНИЕ

- Проверенная документация из первоисточников
- Версионно-специфичные API и примеры кода
- Автоинжект в контекст — код не покидает LLM

КАК РАБОТАЕТ

1. **Промпт + use context7**
   Пишете промпт как обычно, добавив use context7
2. **resolve-library-id → query-docs**
   Context7 находит библиотеку и подтягивает документацию
3. **Inject в контекст LLM**
   Агент генерирует код на основе актуальных API

КЛЮЧЕВЫЕ ПРЕИМУЩЕСТВА

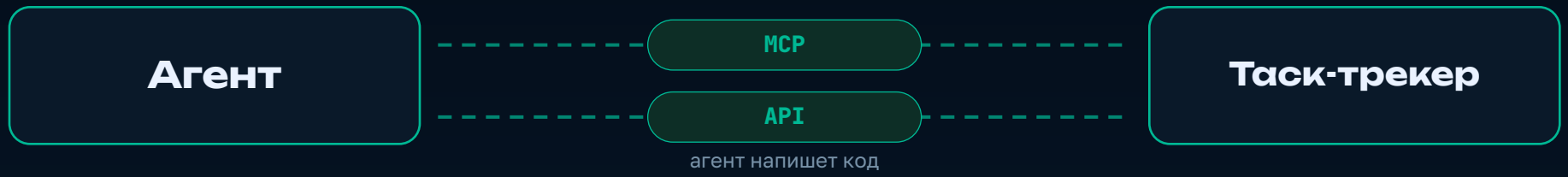
Антигаллюцинация	Универсальность	Безопасность
Проверенные данные	30+ MCP-клиентов	Код не покидает LLM

ПРИМЕР ПРОМПТА

```
Создай middleware для Next.js, который проверяет JWT в cookies
и редиректит неаутентифицированных пользователей на /login. use context7
```

> *Insight: Добавьте правило в CLAUDE.md: "Always use Context7 for library docs" — агент будет подтягивать документацию автоматически.*

Агент + Таск-трекер



Что может агент через интеграцию:

Заводить задачи с описанием	Декомпозиция и план работ	Менять статусы задач
Описывать найденные баги	Планировать спринты	**Всё, что может человек**

В инструкциях агенту описывается, как работать с трекером

08 AI-Native команда

Как получать стабильный и качественный результат в реальных рабочих сценариях

Уровни использования ИИ

01

Персональные агенты

Каждый использует своих агентов, как личный ускоритель

02

Командные агенты

Общие агенты используются командой для закрытия отдельных блоков работ

03

AI-Native команда

Продуктовая разработка большей частью осуществляется с помощью ИИ агентов через сквозную передачу контекста и артефактов между ИИ-агентами и ролями

Документация — часть кодовой базы

Чтобы ИИ-агенты ориентировались в продукте, его устройстве и принятых решениях — документация живёт рядом с кодом

01 Решения по продукту

ADR (Architecture Decision Records), product decisions log — почему выбрали этот путь, а не альтернативный

02 Сценарии использования

User stories, use cases, acceptance criteria — что продукт делает с точки зрения пользователя

03 Архитектура

C4-диаграммы, data flow, интеграции — как система устроена и как части связаны между собой

04 Контекст для агентов

Conventions, coding standards, domain glossary — правила, которым агент должен следовать при генерации

→ Не "документация кода", а карта продукта — для людей и агентов

Этап 1: слабо меняем процесс, но увеличиваем скорость

Оставляем роли и этапы, но каждый шаг ускоряется за счёт ИИ-ассистентов

PM + АНАЛИТИК

Формируют описание фичей с ИИ

Несколько фичей за день

→

РАЗРАБОТЧИК

Строит план тех. реализации с ИИ, запускает реализацию

Переходит к следующей фиче

→

ПАРАЛЛЕЛЬНО

Несколько фич реализуются параллельно

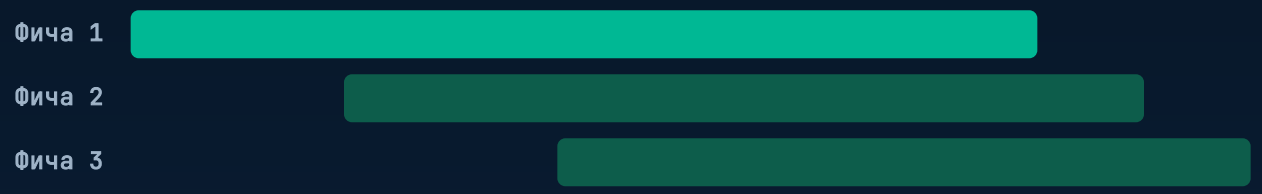
Часы вместо дней

БЫЛО

1 фича → дни/недели.
Последовательно, ожидание между этапами

СТАЛО

Пакет фич → за день/дни.
Параллельно, ИИ реализует



→ Те же роли, тот же процесс — но пропускная способность кратно выше

Этап 2: пересечение ролей

БИЗНЕС-СЛОЙ

PM / Аналитик

Спускается ниже к технологиям:

- Понимает архитектуру
- Чтение кода
- Использование ИИ для реализации части фич

ТЕХНО-СЛОЙ

Разработчик

Поднимается выше к продукту:

- Погружается в продукт и решения
- Участвует в проработке требований
- Полностью ставит часть задач ИИ

СИГНАЛ РЫНКА

Срез вакансий PM на американском рынке (2026)
Ключевой тренд –это технизация Product Manager

04 // ВОПРОС

С чего начать внедрение, AI-Native разработки?

С чего начать – в идеале Green Field

Новый проект без кодовой базы

Идеальная точка входа для AI-Native разработки

01

Знания с нуля

Формируем контекст проекта последовательно, с самого начала

02

Минимум блокеров

Нет legacy-кода, нет конфликтов с существующей архитектурой

03

Настройка процесса

Отрабатываем AI-Native workflow на чистом проекте

04 // ВОПРОС

Как зайти в существующий проект?

Заходим в существующий проект

01

Документируем

`docs/ + CLAUDE.md`

AI исследует текущий код, архитектуру и зависимости

Документирует сценарии, API, структуру БД

Человек валидирует и корректирует результат

Наполняем docs/ и CLAUDE.md проекта

→

02

Покрываем тестами

`автотесты на текущее поведение`

AI пишет тесты на основе документации

Покрываем ключевые сценарии и edge-cases

Фиксируем текущее поведение как baseline

CI/CD запускает тесты автоматически

→

03

Модифицируем

`переходим к развитию`

Контекст собран и доступен агенту

Тесты страхуют от регрессий

AI-Native workflow работает на полную

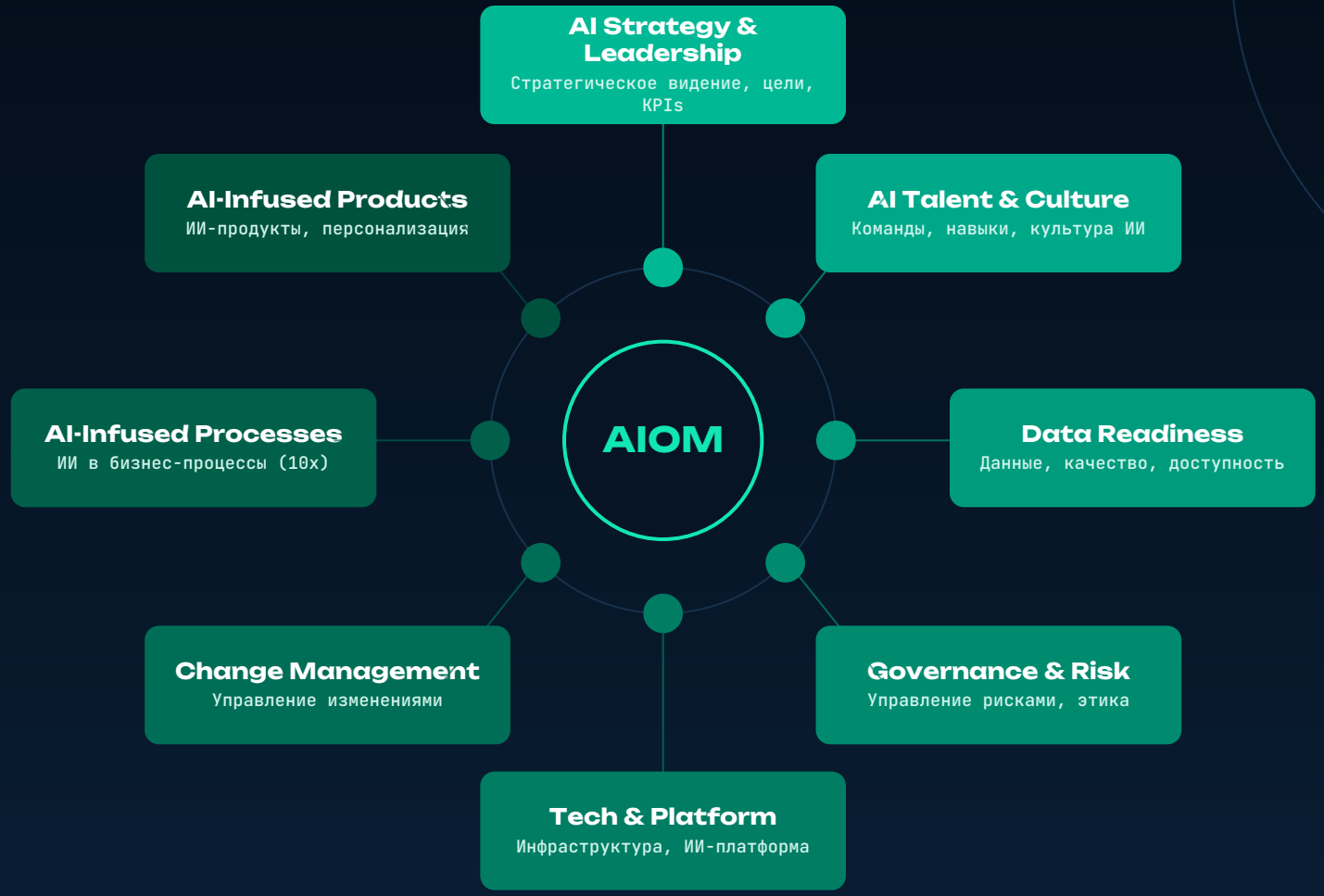
Развиваем функционал проекта

09

Интеграционная сессия

Следующие шаги по интеграции ИИ в процессы SDLC на 30/60/90 дней

AI Operating Model



Направление и план следующих шагов

Совместно формируем план внедрения ИИ в процессы создания и поддержки ПО

01

Выберите пилот

Какие команды, какие проекты, какие техлиды-амбассадоры? Какие безопасные зоны для старта?

02

Дорожная карта

Какие первые шаги необходимы для старта AI-Native команд? Какая инфраструктура понадобится? Как выглядит дорожная карта на 30/60/90 дней?

03

Согласуйте правила

Договоритесь о правилах работы команды, которые необходимы для фиксации специфики нового процесса работы, навыков и приемов работы для следующих команд.

AUTONOMOUS

COMPANY

Q&A

Обсуждение и вопросы

[Алексей Воронин]
[alex@autonomouscompany.ru] | [+7 915 063-67-93]