

AI-инструменты для разработчиков: от обзора к эффективному внедрению



Андрей Терещенко
Full-Stack разработчик

Обо мне



14 лет в разработке

Fullstack разработчик мобильных приложений и веб-сервисов.



Руководитель IT агентства

В команде 20+ человек.



Финалист и призер хакатонов

Проверяю новые технологии в условиях жестких временных рамок.



Категории AI инструментов



Chat-боты

Интерактивные помощники для диалога и решения задач через естественный язык

ChatGPT, Claude.ai, AI Studio, DeepSeek



No-code конструкторы

Создают интерфейсы и приложения по текстовому описанию без программирования

Bolt.new, Lovable.dev, V0.dev, Replit



IDE-ассистенты

Помогают писать, анализировать и оптимизировать код прямо в среде разработки

Cursor, Windsurf, Cline, GitHub Copilot, Claude Dev



AI-агенты

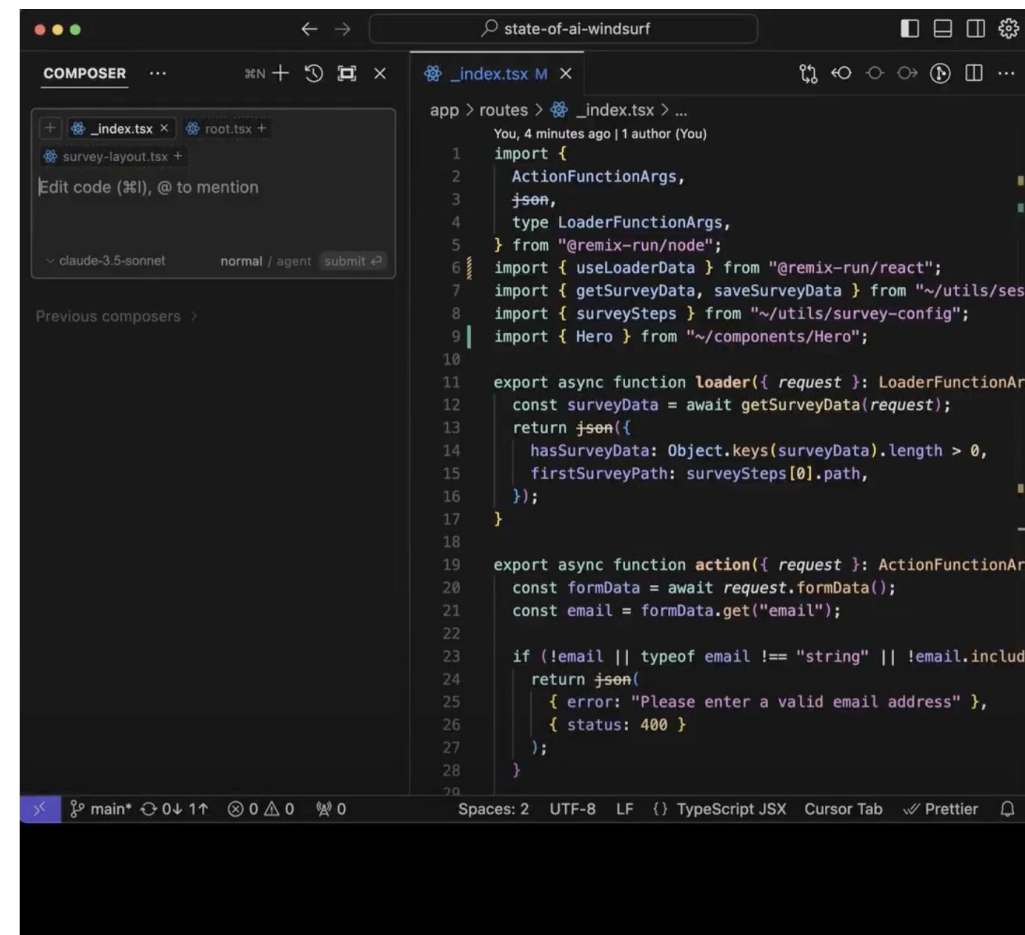
End-to-end решения, которые автоматически трансформируют задачи в готовый код

Codex, Remote Agents, Background Agents, Zen Agents

AI IDE Ассистенты

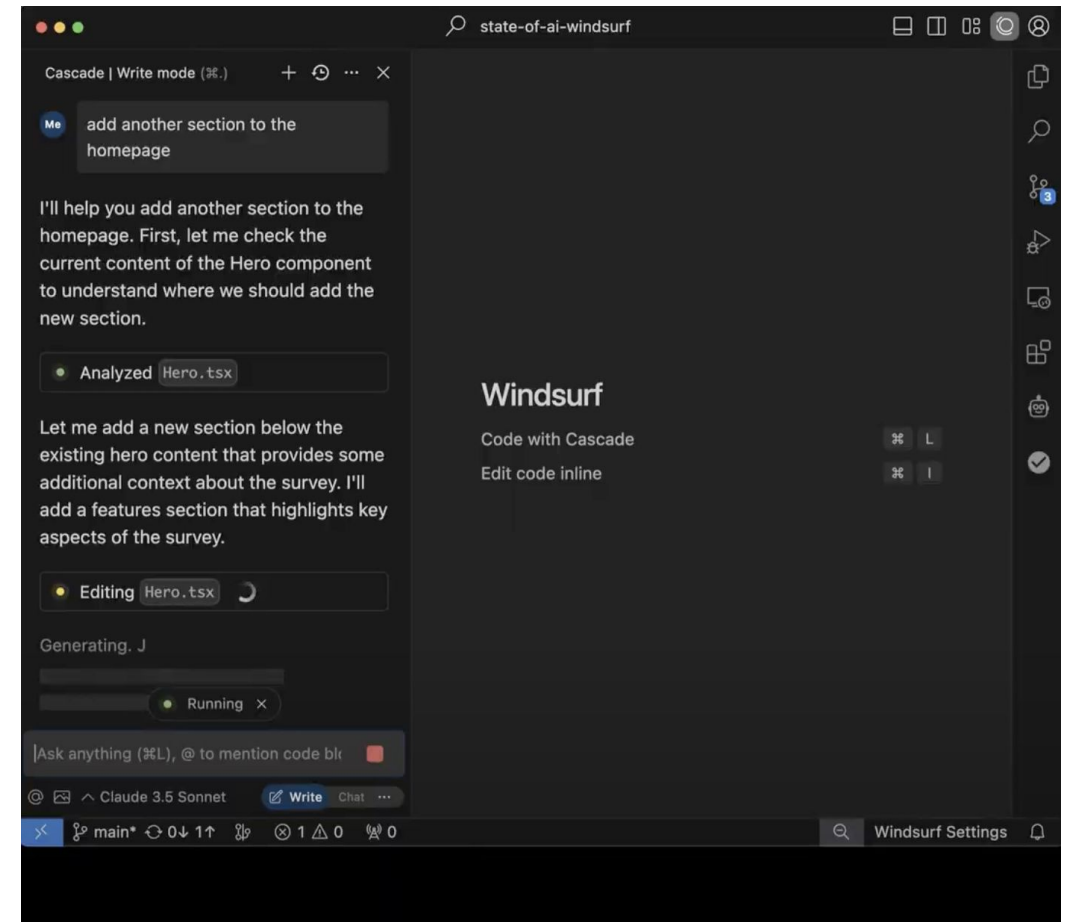


- Форк VS Code с глубокой ИИ-интеграцией.
- Автодополнение, генерация кода по описанию.
- Понимание структуры кодовой базы
- Встроенный ИИ-агент для автоматизации задач.



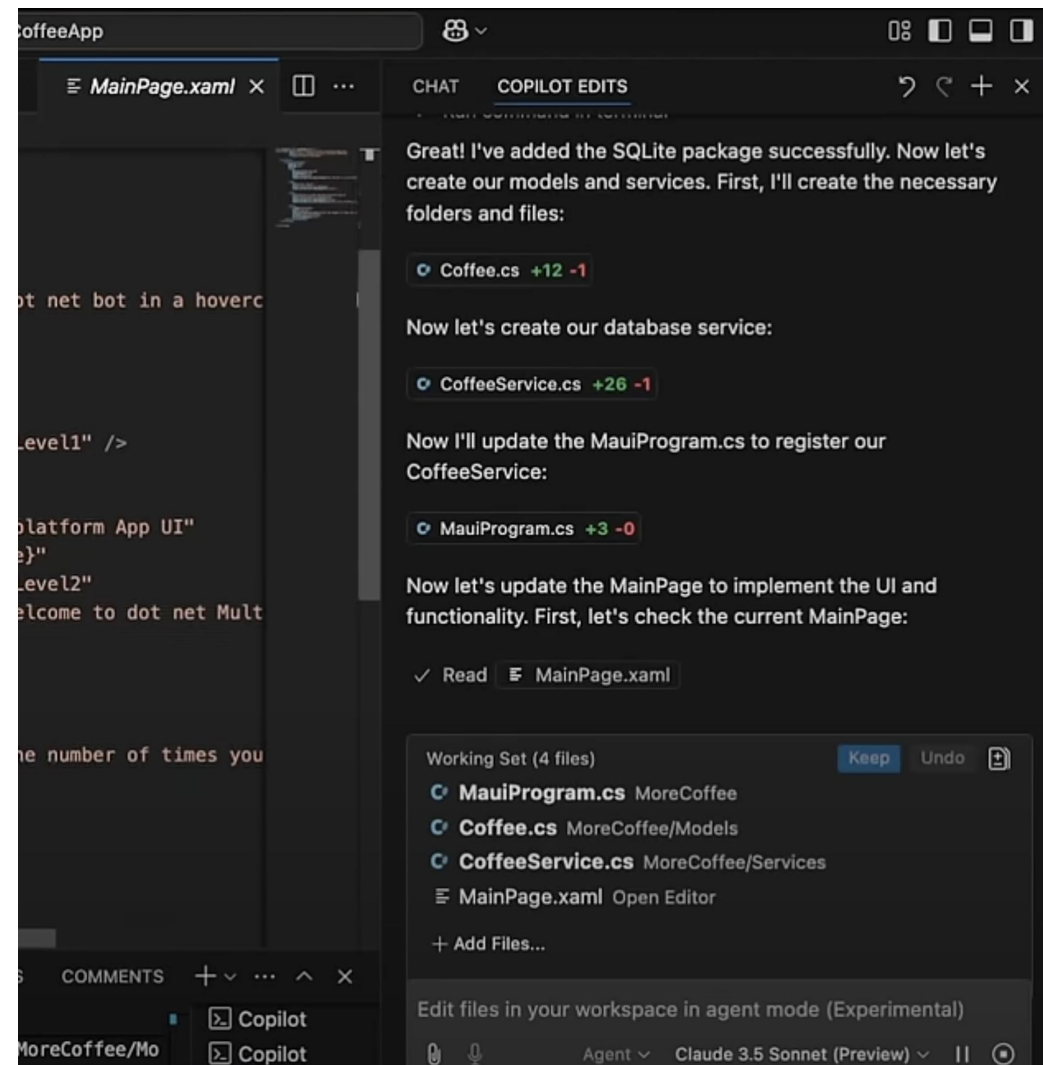


- Аналог Cursor, форк VS Code
- Есть урезанные расширения для VSCode и IntelliJ
- Cascade AI-агент
- Memories функция, сохраняющая контекст между сессиями



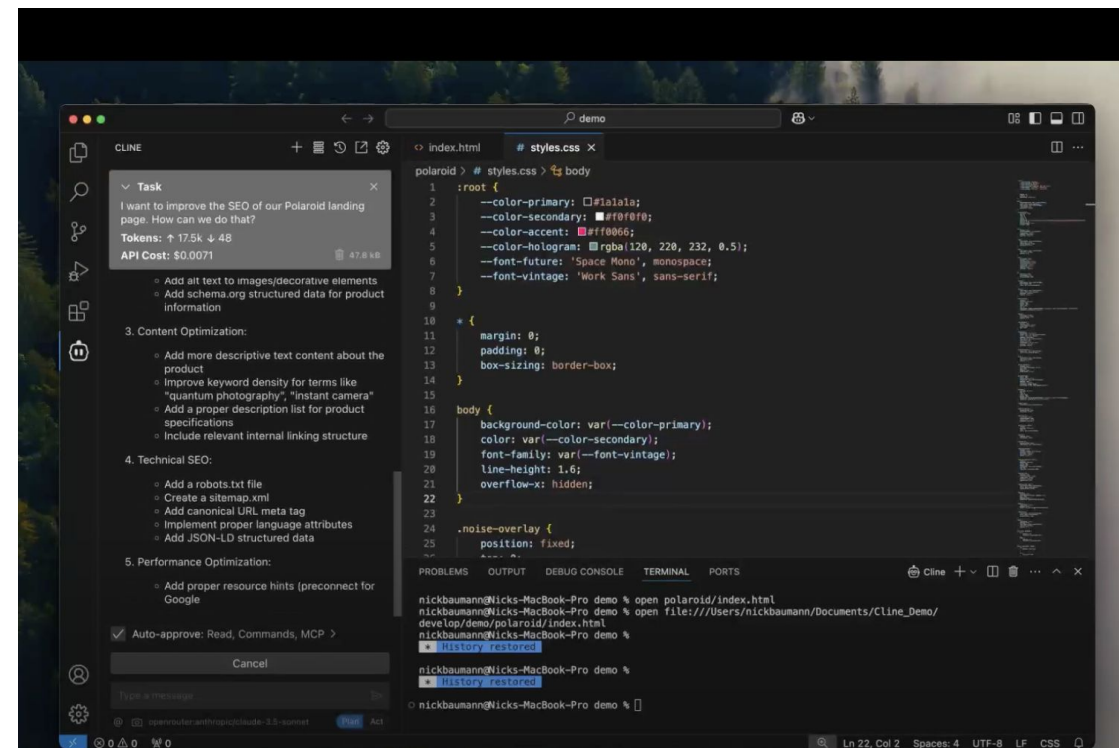
Github Copilot

- Разработан Microsoft, интегрируется во множество IDE (VS Code, IntelliJ, Neovim и др.).
- Автодополнение кода
- Чат для вопросов и AI-агент



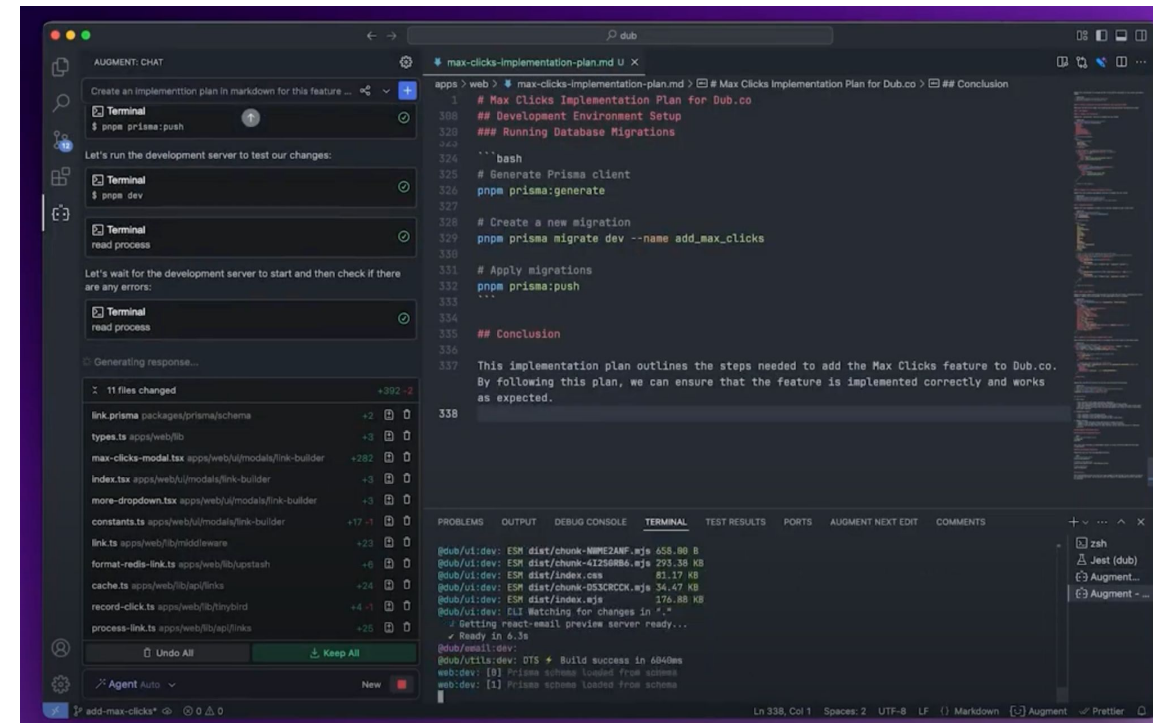
Cline/Roo Code

- Плагин для VS Code
- Автономный агент, работающий с вашими API-ключами к LLM (включая локальные).
- Фокус на безопасности и контроле затрат.
- Функционал: создание/редактирование файлов, выполнение команд, браузеринг.



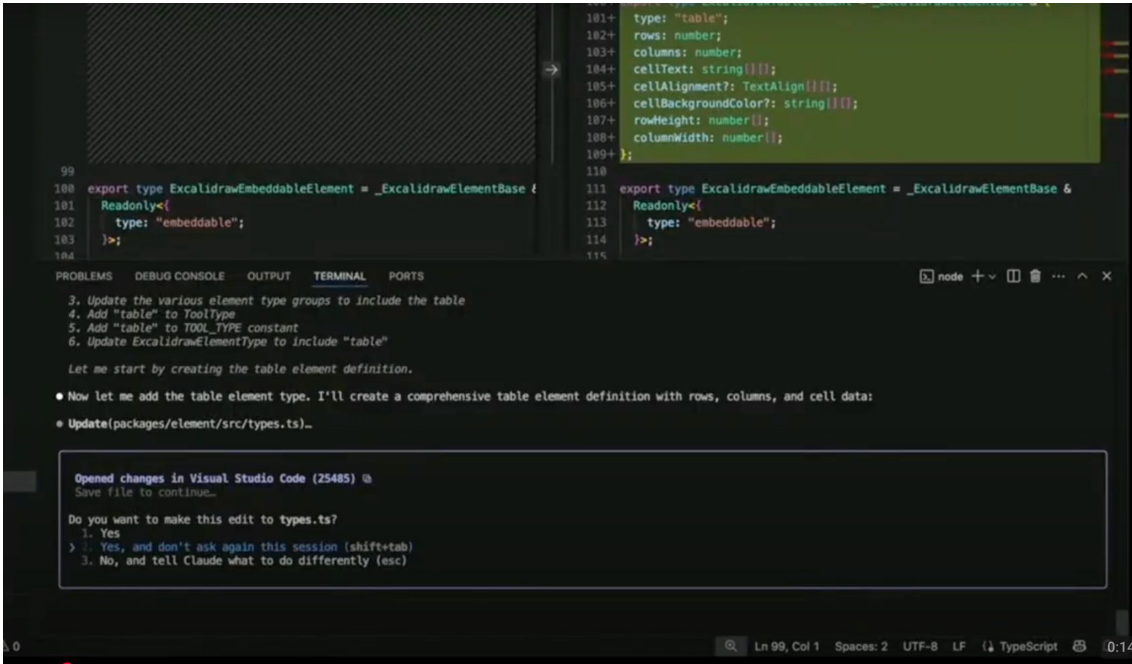
{."} Augmented Code

- Расширение для JetBrains IDE, VS Code, VIM.
- AI Agent с глубоким пониманием контекста
- Интеграция с системами управления проектами (Jira, Linear): комментирование, создание веток.



Claude code

- AI Agent для терминала, может работать часами без остановки, от задачи до коммита.
- Есть плагины для IDE, для просмотра изменений



```

101+ type: "table";
102+ rows: number;
103+ columns: number;
104+ cellText: string[][];
105+ cellAlignment?: TextAlign[];
106+ cellBackgroundColor?: string[];
107+ rowHeight: number[];
108+ columnWidth: number[];
109+ };
110
111 export type ExcalidrawEmbeddableElement = _ExcalidrawElementBase &
112   Readonly<{
113     type: "embeddable";
114   }>;
115
99
100 export type ExcalidrawEmbeddableElement = _ExcalidrawElementBase &
101   Readonly<{
102     type: "embeddable";
103   }>;
104

```

3. Update the various element type groups to include the table
4. Add "table" to ToolType
5. Add "table" to TOOL_TYPE constant
6. Update ExcalidrawElementType to include "table"

Let me start by creating the table element definition.

• Now let me add the table element type. I'll create a comprehensive table element definition with rows, columns, and cell data:

• Update(packages/element/src/types.ts)...

Opened changes in Visual Studio Code (25485) @
Save file to continue...

Do you want to make this edit to types.ts?

1. Yes
- > 2. Yes, and don't ask again this session (shift+tab)
3. No, and tell Claude what to do differently (esc)

Сравнение ИИ инструментов

Cursor

\$20/месяц Pro

Стоимость задачи: \$0.04 - \$0.06

- ✓ Автокомплит
- ✓ Режим агента
- ✗ Ограниченный контекст

Windsurf

\$15/месяц Pro

Стоимость задачи: \$0.03 - \$0.06

- ✓ Бесплатный автокомплит
- ✓ Плагины для IntelliJ
- ✗ Частые изменения цен

GitHub Copilot

\$10/месяц Pro

Стоимость задачи: \$0.066 - \$0.45

- ✓ Поддержка всех IDE
- ✓ PR review
- ✗ Медленные запросы

Cline/Roo Code

Бесплатно + cost API

Стоимость задачи: \$0.010 - \$0.45+

- ✓ Свои модели
- ✓ Контроль затрат
- ✗ Нет автокомплита

Augmented Code

\$50/месяц Dev

Стоимость задачи: \$0.08

- ✓ Большой контекст
- ✓ Умный анализ
- ✗ Передача кода на сервер

Claude Code

Max Plan \$100-\$200

\$3-15/1M токенов

- ✓ Opus/Sonnet 4.0
- ✓ Терминальный агент
- ✗ Нет автокомплита

Сравнение ИИ инструментов

Сравниваем инструменты

Параметр	Cursor	Windsurf	GitHub Copilot	Cline/Roo Code	Augmented Code	Claude code
Функциональность	- IDE на базе VSCode - Автокомплит - Режим Агента	- IDE на базе VSCode - Урезанные плагины для других IDE - Умный Автокомплит - Режим Агента - Функция для сохранения контекста	- Плагины для различных IDE - Автокомплит - Режим Агента - Github PR review	- Плагин для VS Code - Режим агента	- Плагины для различных IDE - Автокомплит - Режим Агента - Функция для сохранения контекста	- Плагины для JetBrains IDEs и VS Code - AI agent для терминала
Основные преимущества	- Удобное IDE, с различными режимами работы Agent/Auto Complete Chat	- Бесплатный автокомплит - Параллельное выполнение задач в режиме агента - Есть плагины для IntelliJ	- Плагины для большинства IDE - Удобная интеграция с github	- Возможность использовать свои модели для взаимодействия с агентом в том числе и локальные - Точное управление затратами на LLM	- Продвинутый Анализ контекста проекта - Большое контекстное окно	- Возможность использовать последние модели Opus/Sonnet 4.0 по фиксированной стоимости на Claude Max Plan.
Известные Недостатки	- ограничение контекстного окна - использование моделей с большим контекстным окном очень не выгодно - VSCode запретил использовать некоторые плагины в кастомных форках IDE	- неясный максимальный размер контекстного окна на подписке pro - частые изменения ценовой политики - иногда может выдавать результат хуже, чем при использовании cline, нужно тестировать на своих задачах.	- Функции агента только начали достигать до конкурентов - Медленная скорость выполнения запросов к LLM - В демо периоде были ограничения по частоте обращений к моделям - Списывается баланс за каждое обращение к LLM, в рамках одной задачи может быть происходить вплоть до 10 - 15 обращений	- Нет Автокомплита - Сложность интеграции и настройки - Любое действие - это дополнительный запрос к LLM, который может стоить денег	- Нет возможности выбирать модель, с помощью которой вы будете решать задачу - Исходный код проекта сливается на сервера Augmented для анализа и построения базы знаний	- Нет Автокомплита - Без Max Plan предлагается использовать Anthropic API, что довольно дорого. - Каждое действие это запрос к LLM за которое списывается баланс
Цены	Free: 2000 комп./мес; 50 медленных премиум запросов к Агенту Pro: \$20/мес (+ неограниченное число медленных запросов?) Стоимость одной задачи одной: 0.04\$(0.06\$ для бизнеса)	Free: полнофункциональный автокомплит, 25 (обращений в месяц к Агенту); Pro: \$15/мес, 500 обращений к топовым моделям Стоимость одной задачи: 0.03\$(0.06\$ для бизнеса)	Free: 2000 автокомплит действий в месяц, 50 (запросов к API агенту); Pro: 10\$/мес, 300 обращений в месяц Стоимость одного обращения: 0.033\$(0.066\$ для бизнеса) Стоимость одной задачи: ~0.066-0.24\$ (~0.12-0.45\$ для бизнеса)	Бесплатен (open-source). Затраты - только на API LLM (по потреблению) или на свои мощности. Стоимость одного обращения/решения задачи: зависит от модели, количества запросов и контекстного окна ~0.1-0.25\$	Free: 50 обращений, (разрешено обучение AI на кодовой базе) Developer: 50\$/мес, 600 обращений к AI (приватность кода) Стоимость одной задачи: 0.08\$	Token Based: \$3/M input tokens \$15/M output tokens Max Plan 100\$ = 5x запросов от Pro Plan = 25x запросов от бесплатного плана 😊 Max Plan 200\$ = 100x запросов от бесплатного плана Стоимость одной задачи: абсолютно не прозрачна (разработчики упираются в лимит после 10-120 минут активной работы)



<https://juicy-parsley-da8.notion.site/AI-204b2b7ab78d805caf14c625a2217b06>

Внедрение AI-агентов в разработку

Как решать задачи с AI-агентами

1

Описываем задачу

Формулируем максимально чётко и ясно, добавляем все релевантные детали: ссылки на файлы, фрагменты кода, контекст.

2

Адаптируем план с помощью ИИ

Просим ИИ на основе описания задачи сформировать детализированный план действий. Сохраняем результат в `implementation_plan.md`.

3

Реализуем план

Просим ИИ следовать составленному плану. При качественном описании и мощной модели можно получить рабочее решение на 90–95%.

4

Анализируем код

Проверяем сгенерированный код, правим проблемные участки вручную или с помощью агента.

5

Фиксируем изменения

Коммитим результат и переходим к следующей задаче.

Пример: генерируем план

CHAT ROO CODE + 🤖 📄 ⌛ ↺ ⚙️ ✕

Task ✕

Architect task for the following:

- Recording started new object added to the history list, with information: like state: recording, duration of recoding, name which contains time of the recording started. When recording finished, state change to processing, should display progress bar of processing, as well as update of the text with processed info in real time.
- For that we should create domain model, which will identify different statuses and different models
- 1. A domain model (`RecordingHistoryItem`) that tracks different recording states
- 2. A service (`RecordingHistoryService`) to manage active and completed recordings
- 3. A view component to display recording items with appropriate UI for each state
- 4. Integration with the existing ContentViewModel to update recording state and progress

Context Length: 10.3k 63.8k ✕

Tokens: ↑ 182.0k ↓ 13.2k 722 kB

Checkpoint 📄 ⌛

✓ API Request ✕

The updated plan, incorporating the `TranscriptionDisplayItem` enum and `TranscriptionDataProvider`, has been successfully saved to `recording_history_enhancement_plan.md`.

With this refined architectural plan in place, we are now well-prepared to move to the implementation phase.

Roo wants to switch to code mode because: To implement the updated architectural plan for enhancing the recording and transcription history feature, as detailed in `recording_history_enhancement_plan.md`, which now includes `TranscriptionDisplayItem` and `TranscriptionDataProvider`.

☒ Auto-approve: Read, Write, Execute, Browser, MCP, Subtasks >

Resume Task Terminate

Type a message...

(@ to add context, / to switch modes, hold shift to drag in files)

Architect ^ vccode ✨ 📷 ➤

Пример: реализация

Architectural Plan: Enhanced Recording and Transcripti... +

Checkpoint 1 ↩

May 24 10:04 PM

Architectural Plan: Enhanced Recording and Transcription History

This document outlines the plan to enhance the recording and transcription history feature in the Transcript application.

Expand

Augment May 24 10:04 PM

I can see you have the `recording_history_enhancement_plan.md` file open, which outlines a comprehensive architectural plan for enhancing the recording and transcription history feature in your Transcript application. This plan describes a three-phase approach with detailed models, services, and UI components.

How can I help you with this plan? Are you looking to:

1. **Implement** specific parts of this plan (like creating the models, services, or UI components)?
2. **Review or refine** the architectural approach?
3. **Get started** with Phase 1 implementation?
4. **Understand** how this integrates with your existing codebase?
5. **Something else** related to this enhancement plan?

Let me know what you'd like to work on, and I can help you move forward with the implementation or any other aspects of this plan.

Architectural Plan: Enhanced Recording and Transcripti... +

`iptApp.swift` and passes them down through the environment. I need to configure the `RecordingUpdateService` in the `AppRouterHost` where the managers are available. Let me update it:

> Edited file +1 -1 AppRouterHost.swift Transcript

Now I need to remove the old transcription subscription from `ContentViewModel` since the `RecordingUpdateService` will handle this:

> Edited file +6 -4 ContentViewModel.swift Transcript

Let me run a quick check to see if there are any compilation issues:

> ⚠️ Diagnostics

Perfect! Now let me create a simple test to verify the architecture is working. Let me check if there are any existing tests or if I should create a simple verification:

• Read directory Transcript

Excellent! The architecture has been successfully implemented. Let me provide a summary of the corrected implementation:

Summary of Corrected Implementation

12 files changed +1367 -178 Discard All Keep All

Пример: **профит**

Commit 93b3073

mr-procrastinator committed last week

maintaining recording progress in the recording dedicated item

1 parent 2339b4f commit 93b3073

Filter files...

16 files changed +1370 -185 lines changed

Search within code

Files changed:

- .ai-rules/agent-rules.md +2
- Packages/AudioTranscript/Sources/AudioTranscript.swift +52 -35
- Transcript/AppRouterHost.swift +2 -1

Diff for Transcript/AppRouterHost.swift:

```
@@ -12,14 +12,15 @@ struct AppRouterHost: View {
12     @StateObject private var appRouter = AppRouter()
13     @EnvironmentObject var transcriptionManager: TranscriptionManager
14     @EnvironmentObject var audioRecordingManager: AudioRecordingManager
15 +    @EnvironmentObject var transcriptionDataProvider: TranscriptionDataProvider
16
17     var body: some View {
18         Group {
19             switch appRouter.activeScreen {
20                 case .onboarding:
21                     OnboardingView()
22                 case .main:
23                     ContentView(audioRecordingManager: audioRecordingManager,
24                               transcriptionManager: transcriptionManager,
25                               transcriptionDataProvider: transcriptionDataProvider )
26             }
27         }
28     }
29 }
```

Извлекаем **максимум**

Описываем специфичные правила для проектов

Используйте Cursor Rules, `.cline/.cursor-rules` для описание правил работы/кодирования и структуры проекта

Формируем Memory Bank

Позволяем ИИ помнить предыдущие взаимодействия в рамках проекта или задачи, обеспечивая контекстную непрерывность работы.

Test-Driven Development (TDD)

Просим ИИ сначала написать тесты, что улучшает качество кода и помогает лучше структурировать разработку.

Используем полезные MCP и плагины

- Taskmaster для разбивки большой задачи на подзадачи и контроля их исполнения
- PlayWrite для доступа к браузеру
- Context 7 для актуальной документации и уменьшения галлюцинаций.

Проблемы и решения

Сложность ИИ-разработки

Происходит фундаментальный сдвиг от традиционного "написания" кода к необходимости тщательного анализа и ревью сгенерированного ИИ кода.

Решение: Внедрение обязательного код-ревью на всех этапах и систематическое обучение команды навыкам анализа вывода ИИ.

Галлюцинации и ошибки

ИИ может выдумывать несуществующие функции, методы или использовать устаревшие API, что приводит к неработающему коду

Решение: Использование специализированных плагинов с доступом к актуальной документации, тщательная перепроверка критических участков кода и формулирование максимально четких промптов.

Работа со старой кодовой базой

ИИ испытывает трудности с пониманием контекста и особенностей легаси-кода, что может привести к несовместимым решениям.

Решение: Формирование базы знаний по модулям для предоставления максимально полного контекста по существующей архитектуре системы.

Шаги к успешной ИИ-интеграции

01 Обучение и эксперименты

Организируйте воркшопы, где команда совместно решает задачи с ИИ. Пилотные группы для тестирования разных инструментов.

02 Разработка политик и баз знаний

Сформулируйте лучшие практики, рекомендации по безопасности, гайдлайны по промптингу. Создайте общедоступную базу знаний .

03 Поиск и поддержка "евангелистов"

Выявите энтузиастов в команде, которые будут делиться опытом, следить за новинками и мотивировать коллег.

04 Используем полезные MCP и плагины

Собирайте фидбэк, измеряйте метрики (если возможно: скорость разработки, качество кода). Адаптируйте подход на основе результатов.

ИИ в разработке: Это марафон, а не спринт

- Внедрение ИИ – это изменение парадигмы кодирования и взаимодействия с технологиями
- Процесс может быть непростым: разработчики могут столкнуться с трудностями и вернуться к привычным методам
- Ключ к успеху: дисциплинированный подход, регулярный сбор фидбэка, групповые обсуждения, коллективная ответственность и мотивация

Вопросы?



Андрей Терещенко
Full-Stack разработчик



https://t.me/mr_pro



<https://www.linkedin.com/in/a-tereshchenko/>



https://t.me/transform_the_energy