

ЧЕМУ НАУЧИЛИСЬ LLM ДЛЯ
РАЗРАБОТЧИКОВ ЗА ЭТИ 1.5 ГОДА
И ЧТО НАМ С ЭТОГО?

ОБО МНЕ

Родион Мостовой

- Более 10-ти лет опыта разработки (.NET)
- Делаю продукт с LLM под капотом
- Разработал .NET API клиент для популярных LLM
- Применяю LLM в своей работе с января 2023

GitHub: github.com/rodion-m

Telegram: @rodion_m_tg



Родион Мостовой

О ЧЕМ ПОЙДЕТ РЕЧЬ

- ▶ Рассмотрим продвинутые и неочевидные кейсы применения LLM в разработке
- ▶ Разберем основы промптинга (тезисно)
- ▶ Научимся выбирать лучшую для кодирования LLM
- ▶ Рассмотрим доступный инструментарий для разработчиков
- ▶ Конкретные инструменты и модели
- ▶ Отдельно посмотрим что работает из РФ
- ▶ Поднимем собственный "Copilot" и "ChatGPT"

О ЧЕМ НЕ ПОЙДЕТ РЕЧЬ

ЧЕГО НЕ БУДЕТ В ДОКЛАДЕ

- ▶ Базы ML, DL (AI, LLM, RAG)
- ▶ Деталей устройства нейросетей
- ▶ Доклад сугубо практический для разработчиков ПО (с примерами на .NET)

ОТ ПРАКТИКИ К ТЕОРИИ


ПРИМЕРЫ ЗАДАЧ, КОТОРЫЕ МОГУТ РЕШАТЬ НАИБОЛЕЕ СИЛЬНЫЕ LLM


КЕЙС: ОПТИМИЗАЦИЯ

- ▶ Дано: 100к иконок, зашитые в одном файле и код на C++/CLI, который их нарезает
- ▶ Задача: ускорить код и переписать алгоритм на C#

Chatbox



>≡ MetaExpert ↻ ⋮



 Почему этот код медленный? Ответь на русском.

<CPP> 

```
void ParseIcons() {
    int i = 0;
    for (int y = 0; y < lines_count; y++)
    {
        for (int x = 0; x < line_icons_count; x++)
        {
            String^ name = IconsNames[i];
            Bitmap^ icon = CutImage(x * icon_width, y * icon_height, icon_width, icon_height);
            Icons->TryAdd(name, icon);
            i++;
        }
    }
}

CutImage(int x, int y, int width, int height) {
    auto dst = gnew Bitmap(width, height, PixelFormat::Format32bppArgb);
    auto rect = Drawing::Rectangle(x, y, width, height);
    auto gfx = Graphics::FromImage(dst);
    gfx->DrawImage(MainImage, 0, 0, rect, GraphicsUnit::Pixel);
    delete gfx;
    return dst;
}
```

word count: 87, token count: 220  

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating



Этот код может быть медленным по нескольким причинам:

- Частое создание и удаление объектов:** В функции `CutImage` для каждой иконки создается новый объект `Bitmap` и объект `Graphics`. Это довольно ресурсоемкие операции, особенно если `lines_count` и `line_icons_count` имеют большие значения, что приводит к тысячам вызовов `CutImage`. Частое создание и удаление объектов увеличивает нагрузку на сборщик мусора (в контексте .NET), что может снизить производительность.
- Обращение к графическим ресурсам:** В каждом вызове `CutImage` происходит обращение к графическому ресурсу `MainImage` для создания нового `Bitmap`. Манипуляции с графическими ресурсами (особенно копирование частей изображений) занимают значительное время, особенно при работе с большими изображениями.
- Неэффективное использование памяти:** Создание отдельного объекта `Bitmap` для каждой иконки может привести к значительному расходу памяти, особенно если иконки имеют большой размер или их много. Это может привести к частым операциям сборки мусора, которые замедляют выполнение программы.
- Повторное использование объекта `Graphics`:** В текущей реализации для каждого копирования создается новый объект `Graphics`. Возможно, улучшение производительности можно было бы достичь путем повторного использования одного объекта `Graphics` вместо его пересоздания для каждой операции копирования.
- Отсутствие многопоточности:** Код выполняет все операции последовательно. Использование многопоточности для обработки различных строк или столбцов иконок могло бы значительно ускорить процесс, особенно на многоядерных процессорах.

Для оптимизации этого кода можно рассмотреть следующие шаги:

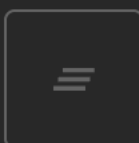
- Исследовать возможность минимизации создания объектов `Bitmap` и `Graphics`, возможно, путем их повторного использования.
- Попробовать оптимизировать обращение к `MainImage`, возможно, предварительно обработав изображение или изменив способ его обработки.
- Рассмотреть возможность использования многопоточности для параллельной обработки иконок.

word count: 240, token count: 735, tokens used: 1515, model: gpt-4-0125-preview



Ок, оптимизируй код по максимуму и перепиши его на C#

word count: 10, token count: 33



Chatbox

MetaExpert

Search...

```
{
    Parallel.For(0, lines_count, y =>
    {
        for (int x = 0; x < line_icons_count; x++)
        {
            int i = y * line_icons_count + x; // Calculate the index based on x and y
            string name = IconsNames[i];
            Bitmap icon = CutImage(x * icon_width, y * icon_height, icon_width, icon_height);
            Icons.TryAdd(name, icon);
        }
    });
}

private Bitmap CutImage(int x, int y, int width, int height)
{
    Rectangle rect = new Rectangle(x, y, width, height);
    Bitmap dst = new Bitmap(width, height, PixelFormat.Format32bppArgb);

    using (Graphics gfx = Graphics.FromImage(dst))
    {
        gfx.DrawImage(MainImage, new Rectangle(0, 0, width, height), rect, GraphicsUnit.Pixel);
    }

    return dst;
}
```

В этом коде:

- Использован `ConcurrentDictionary` для безопасной работы с коллекцией из многопоточного кода.
- `Parallel.For` используется для параллельной обработки строк. Это может значительно ускорить выполнение на многоядерных системах.

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Нейросеть предложила использовать `Parallel.For` и `ConcurrentDictionary`

Chatbox

MetaExpert

Search...

Можно ли еще больше оптимизировать?
word count: 5, token count: 23

Для достижения максимальной оптимизации обработки изображений в C#, особенно при работе с большим количеством мелких операций, как в случае с извлечением иконок, можно рассмотреть следующие дополнительные стратегии:

- Использование нативных библиотек:** Для обработки изображений .NET-библиотеки могут быть не самым быстрым решением из-за накладных расходов, связанных с управляемым кодом. Использование нативных библиотек через P/Invoke или C++/CLI может предложить значительные улучшения производительности. Например, библиотеки, такие как OpenCV или Direct2D, предлагают оптимизированные алгоритмы для работы с изображениями.
- Предварительная обработка изображений:** Вместо того чтобы обрабатывать каждую иконку отдельно, можно рассмотреть возможность предварительной обработки всего изображения, например, применяя фильтры или изменяя разрешение для уменьшения времени, необходимого для извлечения каждой иконки.
- Избегание многократного копирования пикселей:** В текущем подходе каждая иконка копируется в новый объект `Bitmap`, что может быть неэффективно. Возможно, будет более эффективным работать с одним большим изображением, а затем просто ссылаться на соответствующие области этого изображения при необходимости.
- Использование более эффективных структур данных:** Вместо `ConcurrentDictionary`, который обеспечивает потокобезопасный доступ за счет некоторых накладных расходов, можно рассмотреть использование обычного словаря с явной синхронизацией доступа в тех местах, где это необходимо, если порядок добавления элементов позволяет минимизировать конкуренцию за доступ.
- Параллельная обработка с ограничением:** При использовании `Parallel.For` или `Task.Run` важно учитывать ограничения системы, такие как количество ядер процессора, чтобы не создавать избыточное количество потоков, что может привести к контекстному переключению и ухудшению производительности. Использование `ParallelOptions` с установленным свойством `MaxDegreeOfParallelism` может помочь контролировать количество одновременно выпол...

word count: 238, token count: 767, tokens used: 4173, model: gpt-4-0125-preview


Попробуй нарезать иконки нативно
word count: 4, token count: 24

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Нейросеть предлагает перейти на нативные библиотеки

Chatbox

>≡ MetaExpert Search... ↺ ⋮

 Для нативной нарезки иконок в C#, мы можем использовать нативные API Windows, например, GDI+ через P/Invoke или другие низкоуровневые библиотеки для работы с изображениями. Однако, это требует глубокого понимания работы с нативным кодом и может быть не так просто в реализации и поддержке, особенно учитывая различия в платформах и версиях операционных систем.

Вместо детальной реализации через P/Invoke, я предлагаю рассмотреть использование библиотеки **ImageSharp**, которая предлагает высокую производительность и удобный API для работы с изображениями в .NET. ImageSharp оптимизирована для производительности и может использоваться как более простая альтернатива нативному подходу.

Пример использования ImageSharp для нарезки иконок:

- 1. Установка ImageSharp:** Вам нужно будет добавить библиотеку ImageSharp в ваш проект. Это можно сделать через NuGet Package Manager.

```
<SHELL>
Install-Package SixLabors.ImageSharp
```

- 2. Код нарезки иконок с использованием ImageSharp:**

```
<CSHARP>
using SixLabors.ImageSharp;
using SixLabors.ImageSharp.Processing;

public class IconParser
{
    private Image mainImage;
    private Dictionary<string, Image> icons = new Dictionary<string, Image>();
    private string[] iconsNames;
    private int linesCount;
    private int lineIconsCount;
    private int iconWidth;
    private int iconHeight;

    public IconParser(string imagePath, string[] iconsNames, int linesCount, int lineIconsCount, int iconWidth, int iconHeight)
    {

```

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Отказывается использовать наивный подход, предлагает воспользоваться любой ImageSharp

Chatbox


MetaExpert

```
public void ParseIcons()
{
    for (int y = 0; y < linesCount; y++)
    {
        for (int x = 0; x < lineIconsCount; x++)
        {
            int index = y * lineIconsCount + x;
            if (index < iconsNames.Length) // Проверка на выход за пределы массива
            {
                var iconName = iconsNames[index];
                var icon = mainImage.Clone(ctx => ctx.Crop(new Rectangle(x * iconWidth, y * iconHeight, iconWidth, iconHeight)));
                icons.Add(iconName, icon);
            }
        }
    }
}
```

Этот код использует ImageSharp для загрузки основного изображения, а затем нарезает иконки согласно заданным параметрам. Каждая иконка копируется в новый объект `Image`, что позволяет эффективно управлять памятью и производительностью.

Обратите внимание, что ImageSharp является кросс-платформенной библиотекой, что делает ваш код более универсальным и легким в поддержке по сравнению с нативными методами, которые могут потребовать разных реализаций для разных операционных систем.

word count: 316, token count: 847, tokens used: 4089, model: gpt-4-0125-preview

 Но мне нужен Bitmap. Используй низкоуровневый подход через нарезку по байтам.
word count: 11, token count: 41

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Настаиваем на Bitmap и даем подсказку - просим нарезать иконки по байтам

Chatbox

MetaExpert

Search...

```
BitmapData sourceData = sourceImage.LockBits(rect, ImageLockMode.ReadOnly, sourceImage.PixelFormat);

int bytesPerPixel = Image.GetPixelFormatSize(sourceImage.PixelFormat) / 8;
int byteCount = sourceData.Stride * sourceImage.Height;
byte[] pixels = new byte[byteCount];
System.Runtime.InteropServices.Marshal.Copy(sourceData.Scan0, pixels, 0, byteCount);

int iconStride = iconWidth * bytesPerPixel;
for (int y = 0; y < sourceImage.Height; y += iconHeight)
{
    for (int x = 0; x < sourceImage.Width; x += iconWidth)
    {
        byte[] iconPixels = new byte[iconStride * iconHeight];

        for (int row = 0; row < iconHeight; row++)
        {
            int sourceIndex = ((y + row) * sourceData.Stride) + (x * bytesPerPixel);
            int destinationIndex = row * iconStride;
            Array.Copy(pixels, sourceIndex, iconPixels, destinationIndex, iconStride);
        }

        Bitmap iconBitmap = new Bitmap(iconWidth, iconHeight, sourceImage.PixelFormat);
        BitmapData iconData = iconBitmap.LockBits(new Rectangle(0, 0, iconWidth, iconHeight), ImageLockMode.WriteOnly, iconBitmap.PixelFormat);
        System.Runtime.InteropServices.Marshal.Copy(iconPixels, 0, iconData.Scan0, iconPixels.Length);
        iconBitmap.UnlockBits(iconData);

        // Здесь вы можете сохранить iconBitmap в файл или использовать по назначению
    }
}


sourceImage.UnlockBits(sourceData);
```


[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Нейросеть справилась с запросом

Chatbox

>≡ MetaExpert ↻ ⋮

 Попробуй еще больше ускорить через unsafe
word count: 6, token count: 24

 Для дальнейшей оптимизации и ускорения процесса нарезки иконок из изображения в C# с использованием низкоуровневого подхода, мы можем применить `unsafe` код. Это позволит нам работать непосредственно с памятью, минуя некоторые проверки безопасности, что может привести к увеличению производительности.

Прежде чем приступить, убедитесь, что в вашем проекте включена поддержка `unsafe` кода. Это можно сделать в свойствах проекта в Visual Studio в разделе "Build" путем установки галочки "Allow unsafe code".

Вот пример кода, который использует `unsafe` для нарезки иконок:

```
<CSHARP>
using System.Drawing;
using System.Drawing.Imaging;

public class UnsafeIconExtractor
{
    public static void ExtractIconsUnsafe(string sourceImagePath, int iconWidth, int iconHeight)
    {
        using (Bitmap sourceImage = new Bitmap(sourceImagePath))
        {
            Rectangle rect = new Rectangle(0, 0, sourceImage.Width, sourceImage.Height);
            BitmapData sourceData = sourceImage.LockBits(rect, ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);

            try
            {
                byte bytesPerPixel = 4; // Так как используем PixelFormat.Format32bppArgb
                int iconStride = iconWidth * bytesPerPixel;
            }
        }
    }
}
```

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Но мы идем дальше и просим ее оптимизировать копирование байтов через unsafe код

Chatbox

MetaExpert

Search...

```
byte bytesPerPixel = 4; // Так как используем PixelFormat.Format32bppArgb
int iconStride = iconWidth * bytesPerPixel;

for (int y = 0; y < sourceImage.Height; y += iconHeight)
{
    for (int x = 0; x < sourceImage.Width; x += iconWidth)
    {
        Bitmap iconBitmap = new Bitmap(iconWidth, iconHeight, PixelFormat.Format32bppArgb);
        BitmapData iconData = iconBitmap.LockBits(new Rectangle(0, 0, iconWidth, iconHeight), ImageLockMode.WriteOnly, PixelFormat.Format32bppArgb);

        unsafe
        {
            byte* sourcePtr = (byte*)sourceData.Scan0.ToPointer();
            byte* destPtr = (byte*)iconData.Scan0.ToPointer();

            for (int row = 0; row < iconHeight; row++)
            {
                int sourceY = (y + row) * sourceData.Stride + (x * bytesPerPixel);
                int destY = row * iconData.Stride;

                for (int col = 0; col < iconStride; col++)
                {
                    destPtr[destY + col] = sourcePtr[sourceY + col];
                }
            }

            iconBitmap.UnlockBits(iconData);

            // Здесь вы можете сохранить iconBitmap в файл или использовать по назначению
        }
    }
}
```

[Enter] send, [Shift+Enter] line break, [Ctrl+Enter] send without generating

Задача решена с использованием unsafe

САМОЕ ВАЖНОЕ ПО ПРОМПТИНГУ

Sure, I will fix it
quickly *with ChatGPT*

...8 hours later...



САМОЕ ВАЖНОЕ ПО ПРОМПТИНГУ

☑ Пишите промпты на английском

▶ Большинство нейросетей лучше работают с английским

▶ Обычно на английский расходуется меньше токенов - значит, запросы можно делать больше, а денег расходовать меньше

☑ Используйте CoT (Chain of Thought или "думай шаг за шагом")

☑ Если надо, дайте модели больше контекста

☑ Используйте <теги> для разметки смысловых блоков

☑ По возможности используйте самые мощные LLM: Claude Sonnet 3.5, GPT-4o, Gemini Pro, LLAMA 3.1 405b

☑ **Не используйте модели из непонятных источников (телеграм боты)**

БЕНЧМАРКИ: А КТО ЛУЧШИЙ?

Бенчмарки LLM для кодирования

▶ LiveCodeBench

- Набор задач из LeetCode, AtCoder, Codeforces от Easy до Hard

▶ LiveBench

- "Крестный отец" бенчмарков, включающий LCB (секция Coding)
- Обновляется сильно чаще, чем LCB
- Создан совместно с Yann LeCun

LiveCodeBench (LCB)

Code Generation

Self Repair

Test Output Prediction

Code Execution

450 problems selected in the current time window. You can change start or end date to change the time window.

We estimate cutoff dates based on release date and performance variation. Feel free to adjust the slider to see the leaderboard at different time windows. Please offer feedback if you find any issues!



Rank	Model	Pass@1 ↓	Easy-Pass@1	Medium-Pass@1	Hard-Pass@1
	Claude-3.5-Sonnet	51.3	87.2	45.3	11
	GPT-40-2024-05-13	45.2	87.7	33.3	5.1
1	GPT-4-Turbo-2024-04-09	44.2	85	32.6	5.8
	DeepSeekCoder-V2	41.9	79.9	32	4.9
2	LLama3.1-405b-Ins-FP8	41.3	80.8	28.9	5.8
3	Mistral-Large	40.4	82.6	25.9	4.4
4	Gemini-Pro-1.5-August	40	81.9	23.4	7.5
	GPT-40-mini-2024-07-18	39.7	81.1	25.8	3.9
5	GPT-4-Turbo-1106	39.2	83.6	23.4	2.2

БЕНЧМАРКИ: А КТО ЛУЧШИЙ?

Мой сборник из 20+ разных бенчмарков для программирования: clck.ru/3Csova



ИНСТРУМЕНТАРИЙ

LLM ДЛЯ РАЗРАБОТКИ

~8 КАТЕГОРИЙ

- ▶ Чат (+ чат с расширенным контекстом текущего проекта)
- ▶ Auto-completion (автодополнение кода)
- ▶ Code Search: Поиск в коде по запросу на естественном языке ("где расположена регистрация?")
- ▶ Автогенераторы тестов
- ▶ Автогенераторы документации
- ▶ AI Ассистенты, IDE
- ▶ Интеграция с VCS: Code Review
- ▶ AI-разработчики: Создание программы целиком (в т.ч. итеративно)

ЧАТЫ

CLAUDE SONNET 3.5

ОТ ANTHROPIC

- ▶ Самая "умная" общедоступная LLM для программирования на начало сентября 2024
- ▶ Огромный размер контекстного окна (до 200.000 токенов)
- Из РФ официально ни чат, ни API недоступны
- Закрытая (проприетарная)

ПРОСТОЙ БЕСПЛАТНЫЙ ДОСТУП

- ▶ DeepSeek (deepseek.com)
 - ▶ Китайская модель для кодинга уровня gpt-4
- ▶ Devv.ai
 - ▶ Доступны для использования разные модели (Llama 3 405B, gemini-pro, и т. д.)
 - ▶ Умеет динамически догружать релевантный контекст из GitHub
- ▶ На сентябрь 2024 доступны из РФ

PLAYGROUND

Playground'ы, с кучей доступных моделей, в т. ч. опенсорсных

- ▶ <https://chat.lmsys.org> - огромный выбор моделей, в т. ч. проприетарных
- ▶ <https://labs.perplexity.ai>
- ▶ <https://groq.com> - сверхбыстрые версии Mixtral и LLAMA 3
- ▶ Более 20 публичных чатов и Playgrounds: <https://clck.ru/3CsnpD>



CHATBOX

ЧАТ ЧЕРЕЗ API

- ▶ Позволяет чатиться с разными LLM через API (в т. ч. локально)
 - ▶ Чат через API OpenAI
 - ▶ Чат через API OpenAI через прокси (об этом далее)
- ▶ Доступна веб-версия



OPENAI API ПРОКСИ

▶ <https://openrouter.ai>

▶ Доступно множество моделей (в т. ч. Claude и GPT-4o)

■ Нет оплаты картой российского банка, зато есть оплата криптой

▶ <https://proxuapi.ru> | <https://vsegpt.ru>

▶ Принимает карты российских банков

■ Цены выше, чем в OpenRouter

AI АССИСТЕНТЫ И AI-IDE

CURSOR

- ▶ "AI native" IDE
- ▶ Совместима с VS Code плагинами
- ▶ Разработчики ее очень хвалят
- ▶ Прекрасно подходит для разработки под JS/TS, Python
- ▶ Аналоги: Zed, Aide, Cousine Genie

AIDER

- ▶ AI-комбайн для разработки
- ▶ Автоматически применяет патчи к файлам и коммитит изменения
- ▶ Умеет учитывать фидбек от компилятора/линтера/тестов
- ▶ Работает из консоли
- ▶ Превращает любую IDE в Cursor
- ▶ *56-68% кода в обновлениях aider пишет сам aider*

AIDER

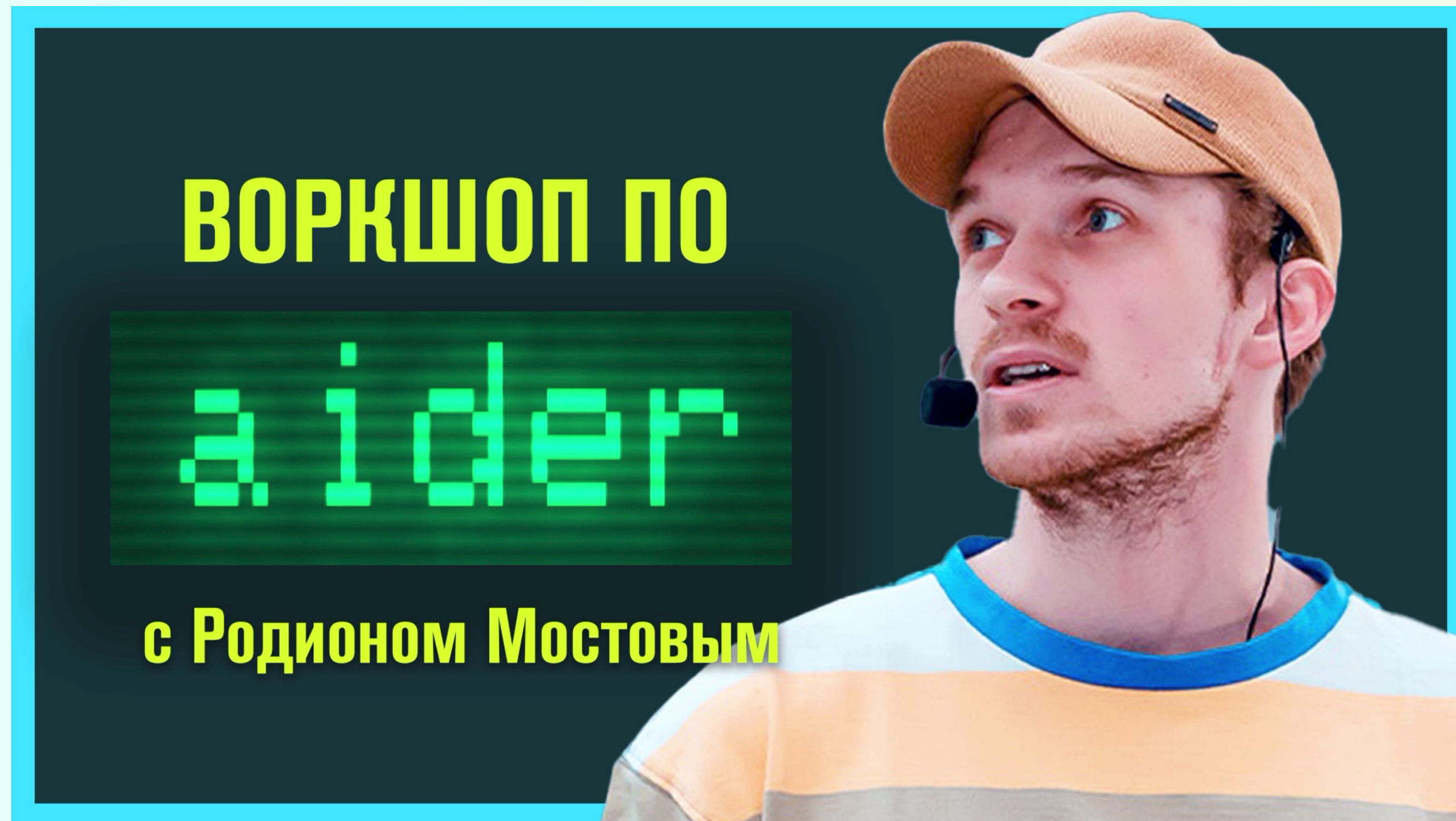
НЕДОСТАТКИ

- Требуется время на изучение



AIDER

РУССКОЯЗЫЧНЫЙ ВОРКШОП



AIDER

Оптимальные настройки с обратной связью для .NET (build, linter):

https://github.com/rodion-m/awesome_ai_for_programmers/blob/main/aider/dotnet/.aider.conf.yml



ПОИСК ПО КОДУ

AI-ЭКСПЕРТЫ ПО КОДОВОЙ БАЗЕ

▶ Greptile

- ▶ Поддерживает множество ЯП

▶ Опенсорсные

- ▶ repo2vec: <https://github.com/Storia-AI/repo2vec>

- ▶ bloop: <https://github.com/BloopAI/bloop>

▶ CodeAlive

- ▶ Полноценная поддержка .NET проектов

- ▶ Заявка на ранний доступ: <https://probelov.net.ru/codealive>

Can a paid order be cancelled?



1. The query asks if a paid order can be cancelled.
2. I first looked at the `SetCancelledStatus` method in the `Order` class [src/Ordering.Domain/AggregatesModel/OrderAggregate/Order.cs:142]. This method explicitly checks if the order status is "Paid" and throws an exception if it is, preventing cancellation.
3. The `CancelOrderAsync` method in the `OrderService` class [src/ClientApp/Services/Order/OrderService.cs:83] doesn't have any specific checks for paid orders, but it calls an API endpoint to cancel the order.
4. The `CancelOrderAsync` method in the `OrdersApi` class [src/Ordering.API/Apis/OrdersApi.cs:22] processes the cancellation request, but it doesn't have any specific logic for paid orders.
5. The `Handle` method in the `CancelOrderCommandHandler` class [src/Ordering.API/Application/Commands/CancelOrderCommandHandler.cs:19] calls the `SetCancelledStatus` method on the order, which we know prevents cancellation of paid orders.

Based on this analysis, my final answer to the user's query is:

No, a paid order cannot be cancelled. The system explicitly prevents the cancellation of orders that have already been paid.

Ask CodeAlive

Run ↩

Reset ↻

Пример поиска по коду dotnet/eShop через CodeAlive

CODE AUTO-COMPLETION

ЛОКАЛЬНЫЙ АВТОКОМПЛИШН

+ ЧАТ В IDE

▶ Tabby (★21K)

▶ Расширение для VS Code, JetBrains IDEs, Vim, NeoVIM

▶ Continue (★15K)

▶ Расширение для VS Code, JetBrains IDEs

▶ Refact.ai (★1.5K)

▶ Расширение для VS Code, JetBrains IDEs, Visual Studio, NeoVIM, Sublime

А ЧТО С PRIVACY?

GITHUB COPILOT

Policies

Suggestions matching public code Allowed ▾

GitHub Copilot can allow or block suggestions matching public code. See [the GitHub Copilot documentation](#) to learn more.

Allow GitHub to use my code snippets for product improvements *





Allow GitHub, its affiliates and third parties to use my code snippets to research and improve GitHub Copilot suggestions, related models and product features. More information in [Privacy FAQ](#).

<https://github.com/settings/copilot>

CHATGPT (WEB)

OPENAI PRIVACY REQUEST PORTAL

I would like to:

 Download my data Request a copy of your data	 Do not train on my content Ask us to stop training on your content
 Delete my ChatGPT account You can ask that we delete your personal data.	 ChatGPT Personal Data Removal Request Remove your personal data from ChatGPT model outputs.

<https://privacy.openai.com/policies?modal=take-control>



LLM ЛОКАЛЬНО: LM STUDIO

LMSTUDIO.AI

- ▶ Позволяет в пару кликов локально запустить чат с LLM
- ▶ А также OpenAI-compatible HTTP API сервер
- ▶ Можете начать с Yi-Coder-9B-Chat, Mistral NeMo (12B), Gemma 9B, Qwen2 7B, Phi-3.5-mini-instruct (4B)



ЕЩЕ КЕЙСЫ

КЕЙС: СОЗДАНИЕ ДОКУМЕНТАЦИИ

- ▶ Закидываем все тесты (или весь код) в LLM с большим контекстом
- ▶ Gemini Flash: 1M токенов (примерно 50.000 строк кода)
- ▶ Gemini Pro: 2M токенов (примерно 100.000 строк кода)
- ▶ Например, я так генерировал API Reference для `elasticsearch-net`

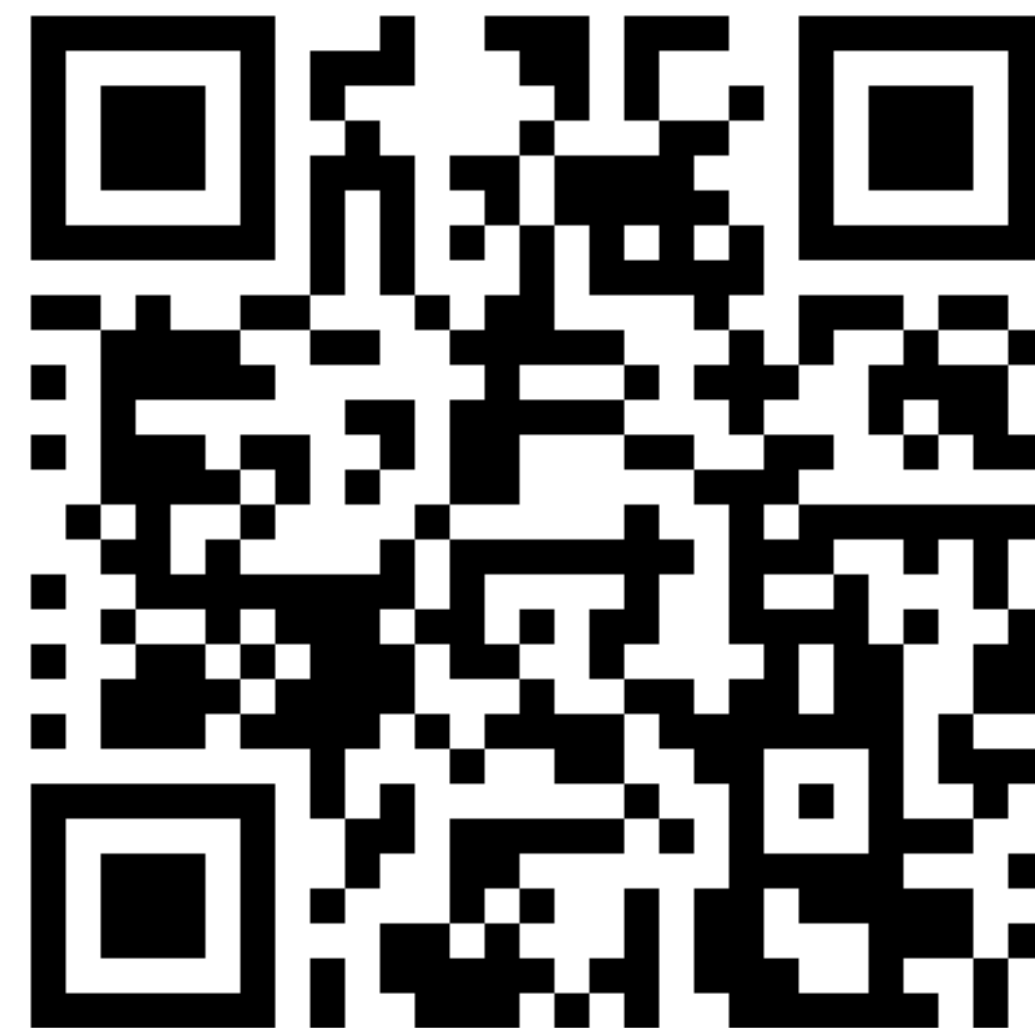
КЕЙС: УКРОЩЕНИЕ STATE MACHINE

- ▶ Дано: Машина состояний, сгенерированная компилятором
- ▶ Задача: Сделать код читаемым для человека
- ▶ Решение в моей статье "Еще раз про асинхронную машину состояний и где именно там аллокации"



КЕЙС: SOLUTION MERGER

- ▶ Пример программы, почти полностью сгенерированной Claude Sonnet 3.5
- ▶ Программа мерджит множество .NET солюшенов в один большой солюшн
- ▶ Readme так же сгенерирован



БОЛЬШЕ КЕЙСОВ

Создаем систему автоматизированной проверки домашних заданий с ChatGPT



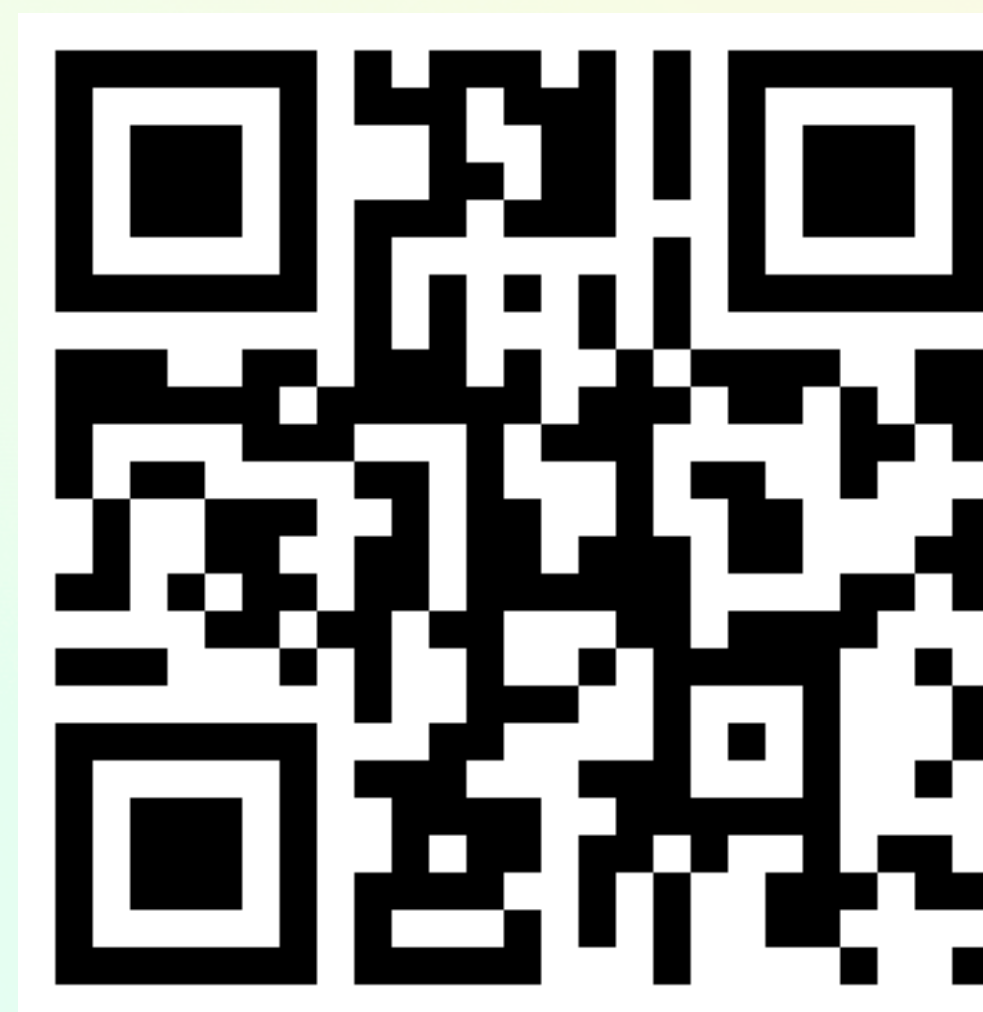
Доклад: "Выжимаем максимум из ChatGPT для программистов"

ИТОГИ

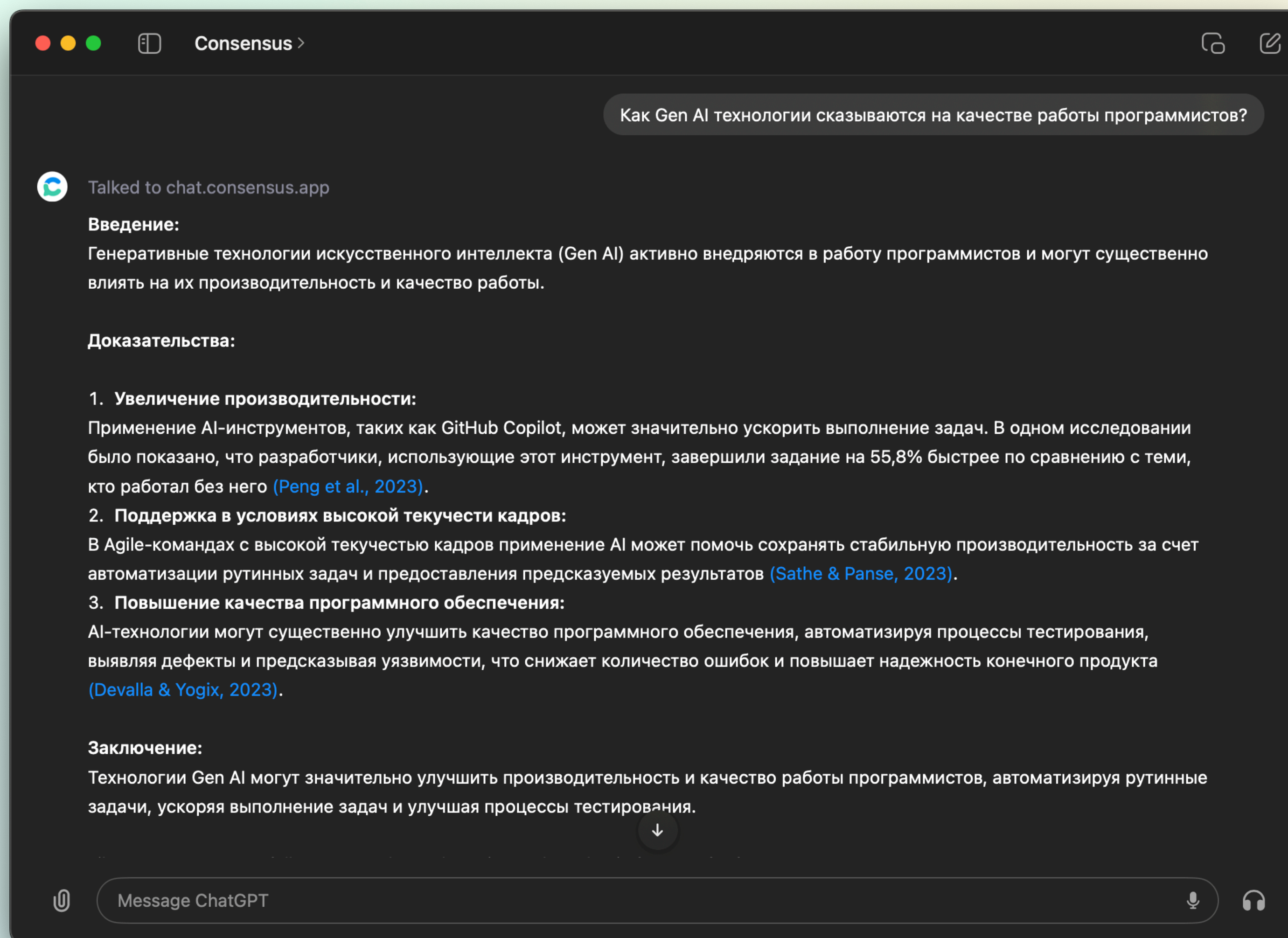
- ▶ Уже сейчас нейросети способны решать задачи сложности Junior+
- ▶ Правильно подобранный инструмент может сэкономить еще больше времени
- ▶ LLM могут быть прекрасным ассистентом, так что всегда проверяйте результат
- ▶ Четкая постановка задачи (ТЗ) сильно улучшает ответы нейросети
- ▶ За 1.5 года открытые модели уровня 8-12B уже вплотную достигают качества GPT-3.5
- ▶ А сильные открытые модели достигают качества GPT-4
- ▶ Более вдумчивые задачи (алгоритмы, многопоточность) трудны даже для сильных LLM

БОНУС: CONSENSUS

- ▶ Поиск исследований на естественном языке
 - по arXiv, ACM, PubMed, и еще 50+ источникам
- ▶ Standalone версия: <https://consensus.app/search/>
- ▶ Также, доступно в виде GTPs из ChatGPT



БОНУС: CONSENSUS



ПОЛЕЗНЫЕ МАТЕРИАЛЫ



СПАСИБО ЗА ВАШЕ ВРЕМЯ

БУДЬТЕ В КУРСЕ ОБ АКТУАЛЬНОМ В МИРЕ AI ДЛЯ РАЗРАБОТКИ

Мой канал: t.me/probelov_net

