

Microsoft Agent Framework

Как и зачем делать своего мини-агента?



ДИМА АФОНЧЕНКО
Dodo Engineering
.NET Tech Lead в Ai Hub



Тема доклада



Расскажу про процесс, который занимал 1,5 месяца

Разложим все на требования

Решим задачу с помощью MAF

Посмотрим пару задач поменьше



ДИМА АФОНЧЕНКО
.NET TECH LEAD

Как мы открываем страны?

Задачи на стыке техники и людей

Стек технологий

ЧАСТЬ 2

Infrastructure



Analytics



Backend



60k

Сотрудников

375

Команда IT

26

Стран



**Страны открываются
примерно раз в квартал**

Что такое открытие страны?

1

Деплой сервисов

4

Добавление языковых настроек в сервисы

2

Фича-флаги

5

Получение и формирование юридических документов

3

Конфиги окружений

6

Получение от разных людей необходимой для открытия информации

Что автоматизируется, а что нет

1

Деплой сервисов

2

Фича-флаги

3

Конфиги окружений

4

Добавление языковых настроек в сервисы

5

Получение и формирование юридических документов

6

Получение от разных людей необходимой для открытия информации

**Весь процесс занимает
1,5 месяца от старта до сдачи**

Одно открытие страны в цифрах

51

карточка —
шагов процесса

140

подшагов
в чек-листах

65%

может закрывать
агент — 91 подшаг

35%

упирается в людей —
49 подшагов

Из чего состоит ручное:

18 — спросить человека и дождаться ответа

16 — попросить чужую команду зарелизить

10 — работу делает другая команда

5 — очередь релиза и ручные проверки

А что если AI агент все сделает за нас?

Требования к системе

1

Собрать контекст

4

Легко локализовать ошибку в процессе

2

Сделать ПР по шаблону

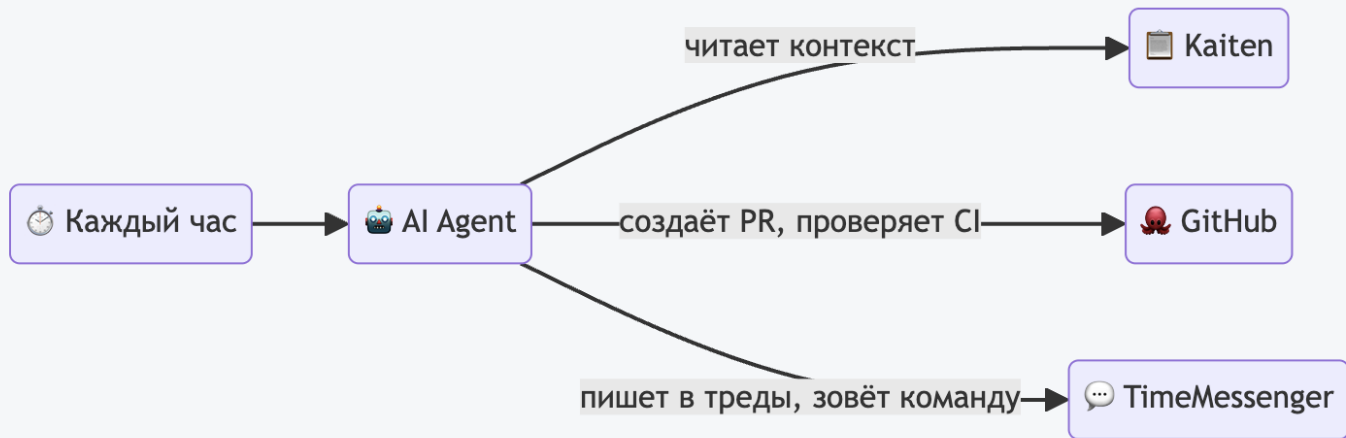
5

Гибкость — понимаем человеческие ответы

3

Соблюдать
последовательность

Как это представлялось



**Copilot, Cursor, Claude и т. п.
справились бы с такой
задачей, но...**

Выполнение требований агентами

1

Собрать контекст

4

Легко локализовать ошибку в процессе

2

Сделать ПР по шаблону

5

Гибкость — понимаем человеческие ответы

3

Соблюдать
последовательность

Выполнение требований жестким кодом

1

Собрать контекст

4

Легко локализовать ошибку в процессе

2

Сделать ПР по шаблону

5

Гибкость — понимаем человеческие ответы

3

Соблюдать
последовательность

Агенты + Код = 💕



**Что такое агент под
капотом?**

А что такое агент под капотом?

Агент = LLM + Инструменты + Цикл принятия решений

LLM

- Думает, рассуждает, решает что делать дальше

Инструменты (Tools)

- Вызовы: GitHub, Corporate Messenger, Task Tracker...

Agentic Loop

- Агент сам решает: думать ещё или действовать. Продолжает до финального ответа.

Кажется, что такое довольно просто написать

```
var response = await client.GetChatMessageContentAsync(history, settings, ct);

// Нужно вручную гонять цикл пока LLM хочет вызвать инструмент
while (response.FinishReason == "tool_calls") {
    var toolCall = response.Choices[0].Message.ToolCalls[0];

    // Нужно вручную диспетчеризовать по имени функции
    var result = await DispatchTool(toolCall.Function.Name, toolCall.Function.Arguments);

    // Нужно вручную добавить результат в историю
    chatHistory.Add(new ToolMessage(toolCall.Id, result));

    // Нужно снова отправить в LLM...
    response = await client.GetChatMessageContentAsync(history, settings, kernel, ct);
}
// + обработка ошибок, таймауты, лимит итераций, параллельные tool calls...
```

Но все не так просто

Написать цикл — 200–300 строк. Ладно, разово переживём.

А дальше:

- модель вызвала два тула сразу — параллельные вызовы, ещё код и ещё тесты
- цикл не остановился — лимиты и защита от закливания
- диалог перестал влезать — обрезка истории так, чтобы не потерять вызовы
- появилась вторая модель — у неё вызовы устроены иначе, слой переписываешь

И каждый пункт — это не только код. Это ещё время, чтобы понять, что ты в него упёрся.



Microsoft Agent Framework

Откуда взялся MAF

AutoGen

простые абстракции для агентов
и мультиагентных схем

Semantic Kernel

энтерпрайз: сессии и состояние,
типобезопасность, фильтры, телеметрия

Откуда взялся MAF

AutoGen

простые абстракции для агентов
и мультиагентных схем

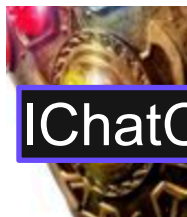
Semantic Kernel

энтерпрайз: сессии и состояние,
типобезопасность, фильтры, телеметрия

Код уже на Semantic Kernel? Переписывать не срочно —
у Microsoft есть Migration Guide, MAF это следующее поколение обоих

Ключевые абстракции

Ключевые абстракции MAF



IChatClient

```
// Создаем чат-клиента, который вызывает АПИ
var openAi = new OpenAIClient(new ApiKeyCredential(apiKey));
ChatClient chatClient = openAi.GetChatClient(gpt0Mini);
```

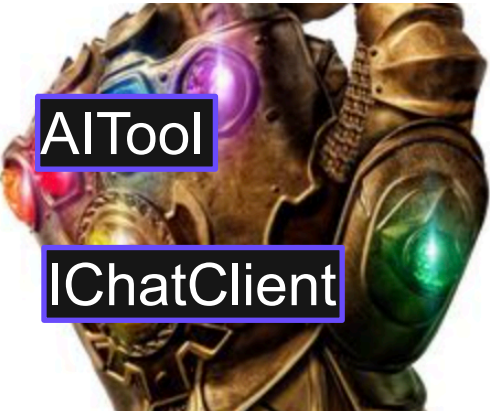
Ключевые абстракции MAF

AITool

- Вызов внешних инструментов

```
// Тул – обычный C#-метод; [Description] объясняет его LLM-у
public class GitHubPlugin(GitHubClient github)
{
    [Description("Create a pull request on GitHub")]
    public async Task<string> CreatePullRequestAsync(
        [Description("Branch with the changes")] string branch,
        [Description("PR title")] string title,
        CancellationToken ct = default)
    {
        var pr = await github.CreatePullRequestAsync(branch, title, ct);
        return $"PR #{pr.Number}: {pr.HtmlUrl}";
    }
}
```

Ключевые абстракции MAF



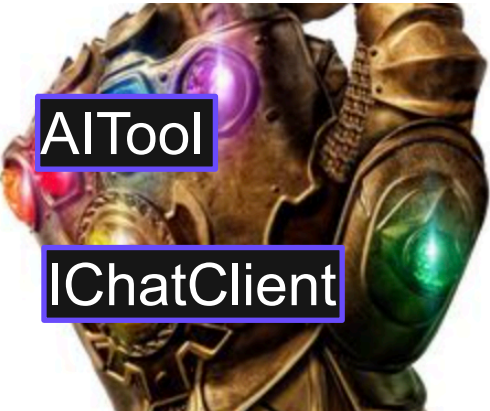
AITool

IChatClient

```
// Создаем чат-клиента, который вызывает АПИ
var openAi = new OpenAIClient(new ApiKeyCredential(apiKey));
ChatClient chatClient = openAi.GetChatClient(gptOMini);

// Создаем инструменты и превращаем в список AITool
var tools = new GitHubTools(github, owner, repo);
```

Ключевые абстракции MAF



AITool

IChatClient

```
// Создаем чат-клиента, который вызывает АПИ
var openAi = new OpenAIClient(new ApiKeyCredential(apiKey));
ChatClient chatClient = openAi.GetChatClient(gptOMini);

// Создаем инструменты и превращаем в список AITool
var tools = new GitHubTools(github, owner, repo);

IList<AITool> aiTools = [
    AIFunctionFactory.Create(tools.ReadFile),
];
```

Ключевые абстракции MAF

The image shows the Infinity Gauntlet, a golden gauntlet with six glowing stones. The stones are labeled with their respective powers: AI Tool (purple), IChatClient (green), and AI Agent (red).

AI Tool

IChatClient

AI Agent

```
// Создаем чат-клиента, который вызывает АПИ
var openAi = new OpenAIClient(new ApiKeyCredential(apiKey));
ChatClient chatClient = openAi.GetChatClient(gptOMini);

// Создаем инструменты и превращаем в список AITool
var tools = new GitHubTools(github, owner, repo);

IList<AITool> aiTools = [
    AIFunctionFactory.Create(tools.ReadFile),
];

// Соединяем все в агента = модель + инструменты
AIAgent agent = chatClient.AsAIAgent(
    instructions: "Ты помощник по репозиторию Dodo. Используй инструменты, " +
        "отвечай кратко.",
    name: "RepoAgent",
    tools: aiTools);
```

Ключевые абстракции MAF

The image shows the Infinity Gauntlet from Marvel, a golden gauntlet with six glowing stones. The stones are labeled with blue boxes containing white text: 'AI Tool' (red stone), 'IChatClient' (blue stone), and 'AI Agent' (purple stone). The text 'agent.RunAsync("prompt")' is overlaid on the gauntlet's hand.

agent.RunAsync("prompt")

AI Tool

IChatClient

AI Agent

```
// Создаем чат-клиента, который вызывает АПИ
var openAi = new OpenAIClient(new ApiKeyCredential(apiKey));
ChatClient chatClient = openAi.GetChatClient(gptOMini);

// Создаем инструменты и превращаем в список AITool
var tools = new GitHubTools(github, owner, repo);

IList<AITool> aiTools = [
    AIFunctionFactory.Create(tools.ReadFile),
];

// Соединяем все в агента = модель + инструменты
AIAgent agent = chatClient.AsAIAgent(
    instructions: "Ты помощник по репозиторию Dodo. Используй инструменты, " +
        "отвечай коротко.",
    name: "RepoAgent",
    tools: aiTools);

var response = await agent.RunAsync(prompt);
```

Демо: работа простого агента



Требования к системе: чем закрыли

Требование

Чем закрывается

1. Собрать контекст

агент + тулы к трекеру, мессенджеру и GitHub

Требования к системе: чем закрыли

Требование

Чем закрывается

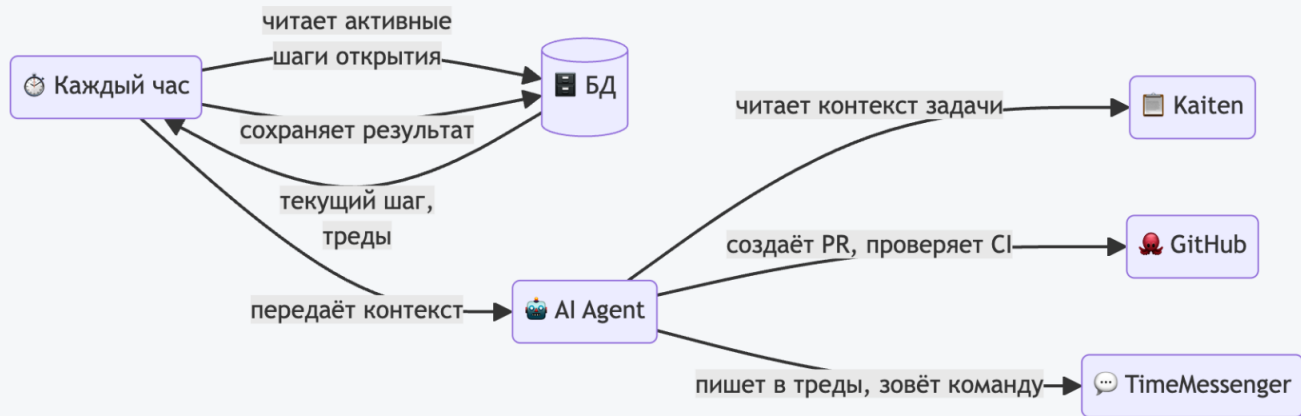
1. Собрать контекст

агент + тулы к трекеру, мессенджеру и GitHub

2. Сделать ПР по шаблону

тулы GitHub: ветка, коммит, PR

Детерминированные части нужно ВЫНОСИТЬ В КОД



Детерминированные части нужно ВЫНОСИТЬ В КОД

```
card:
  id: pr_to_pizza_site
  depends_on: [dev_deploy] # ← не начнём пока другой шаг не готов

checklists:
  - items:
    - id: change_code
      type: ai # ← агент делает сам
      instruction: "Добавь новую страну в appsettings.json..."

    - id: merge_pr
      type: merge_pr # ← ждём человека

      depends_on: [change_code]
```

В каком формате передавать?

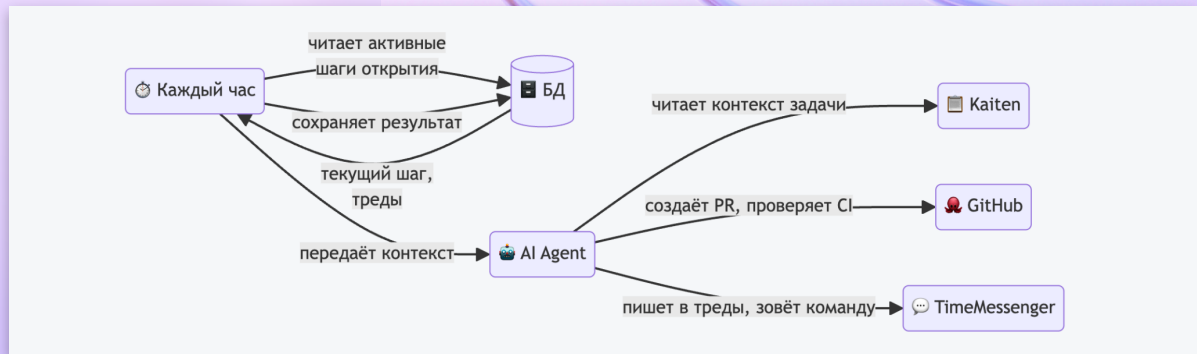
Ручное подтверждение

ApprovalRequiredAIFunction

- Если нужно подтверждение действия, то просто оборачиваем инструмент в `ApprovalRequiredAIFunction`.

```
// Регистрация – метод оборачивается в AITool,  
// MAF сам строит JSON-схему по [Description]  
var plugin = new GitHubPlugin(githubClient);  
  
var tools = new List<AITool>  
{  
    new ApprovalRequiredAIFunction(  
        AIFunctionFactory.Create(plugin.CreatePullRequestAsync()), // ← обернули  
    );  
};  
  
AIAgent agent = chatClient.AsAIAgent(  
    instructions: systemPrompt,  
    tools: tools);
```

Демо



Демо



Требования к системе: чем закрыли

Требование

Чем закрывается

1. Собрать контекст

агент + тулы к трекеру, мессенджеру и GitHub

2. Сделать ПР по шаблону

тулы GitHub: ветка, коммит, PR

Требования к системе: чем закрыли

Требование

Чем закрывается

1. Собрать контекст

агент + тулы к трекеру, мессенджеру и GitHub

2. Сделать ПР по шаблону

тулы GitHub: ветка, коммит, PR

3. Соблюдать последовательность

граф шагов в базе, а не в контексте модели

Требования к системе: чем закрыли

Требование

Чем закрывается

- | | |
|---------------------------------|--|
| 1. Собрать контекст | агент + тулы к трекеру, мессенджеру и GitHub |
| 2. Сделать ПР по шаблону | тулы GitHub: ветка, коммит, PR |
| 3. Соблюдать последовательность | граф шагов в базе, а не в контексте модели |
| 4. Легко локализовать ошибку | один запуск — один шаг со своим контекстом |

Требования к системе: чем закрыли

Требование

Чем закрывается

- | | |
|---------------------------------|--|
| 1. Собрать контекст | агент + тулы к трекеру, мессенджеру и GitHub |
| 2. Сделать ПР по шаблону | тулы GitHub: ветка, коммит, PR |
| 3. Соблюдать последовательность | граф шагов в базе, а не в контексте модели |
| 4. Легко локализовать ошибку | один запуск — один шаг со своим контекстом |
| 5. Гибкость | живой ответ человека разбирает модель |

**Как проследить, что ничего
не упало?**

Ключевые абстракции MAF

Middleware

- Для контроля либо за всем пайплайном (к примеру посмотреть цену всего запроса), либо за каждым вызовом инструмента (к примеру посмотреть цену конкретного шага)

```
// Уровень 1 – вокруг всего RunAsync: токены, latency, retry
AIAgent agent = chatClient
    .AsAIAgent(instructions: systemPrompt, tools: tools)
    .AsBuilder()
    .Use(runFunc: LogAndMeasureRun, runStreamingFunc: null)
    .Build();

async Task<AgentResponse> LogAndMeasureRun(
    IEnumerable<ChatMessage> messages, AgentSession? session,
    AgentRunOptions? options, AIAgent innerAgent, CancellationToken ct)
{
    var sw = Stopwatch.StartNew();
    var response = await innerAgent.RunAsync(messages, session, options, ct);
    log.LogInformation("Run · {Ms}ms · {Tokens} tokens",
        sw.ElapsedMilliseconds, response.Usage?.TotalTokenCount);
    return response;
}
```

Ключевые абстракции MAF

Middleware

- Для контроля либо за всем пайплайном (к примеру посмотреть цену всего запроса), либо за каждым вызовом инструмента (к примеру посмотреть цену конкретного шага)

```
// Уровень 2 – вокруг КАЖДОГО вызова инструмента
IChatClient chatClient = baseChatClient
    .AsBuilder()
    .UseFunctionInvocation(configure: fic =>
        fic.FunctionInvoker = LogToolInvocations)
    .Build();

async ValueTask<object?> LogToolInvocations(
    FunctionInvocationContext ctx, CancellationToken ct)
{
    log.LogInformation($"→ {Name}({Args})", ctx.Function.Name, ctx.Arguments);
    var result = await ctx.Function.InvokeAsync(ctx.Arguments, ct);
    log.LogInformation($"← {Name} → {Result}", ctx.Function.Name, result);
    return result;
}
```

Инструменты Observability

Observability

- Метрики из коробки по GenAI Semantic Conventions
- `gen_ai.client.token.usage` (input/output)
- `gen_ai.client.operation.duration`



```
// Одна строка в pipeline – и весь обмен с моделью трейсится
const string SourceName = "MyApp";

var chatClient = baseChatClient
    .AsBuilder()
    .UseOpenTelemetry(sourceName: SourceName,
        configure: cfg => cfg.EnableSensitiveData = isDev) // только dev/test!
    .Build();

AIAgent agent = chatClient.AsAIAgent(
    instructions: systemPrompt, tools: tools);
```

Что пошло не так

19

ворнингов за две
недели

- 10 — человек ответил неоднозначно
- 5 — агент неправильно вызвал инструмент
- 3 — не было доступа или инструмента
- 1 — кривое состояние карточки

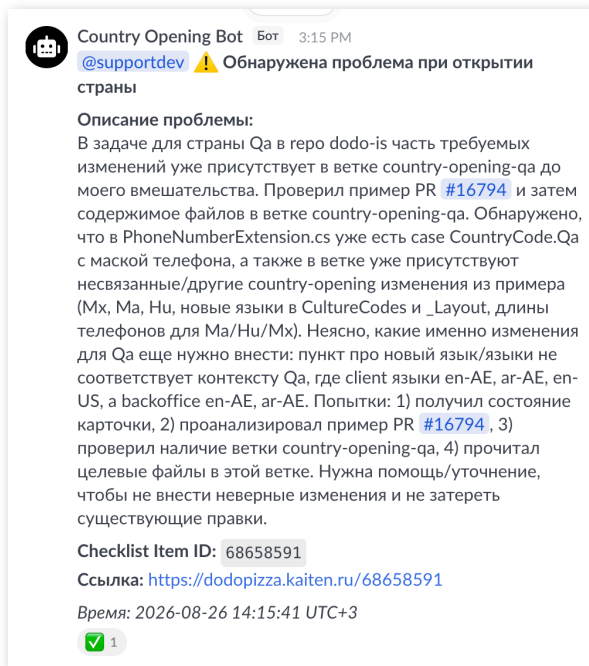
Как выглядит «неоднозначно»:

«Да, сегодня сделаю» — а срок прошёл

«напомни завтра утром»

«а кто вместо него? он в отпуске»

вопрос вместо ответа



Как мы узнаем, что всё сделано



- Ручные шаги: просим человека отметить пункт в карточке
- Часть карточек пока проверяем глазами
- Думаем: агент прикладывает ссылки на сделанное — опрашиваем их
- Думаем: второй агент проверяет за первым

Что если упало на середине

- PR создан, статус записать не успели. Через час будет второй PR? Нет
- Дёшево — попросить в инструкции перечитать карточку. Это надежда на модель
- Надёжно — идемпотентность в коде тула: «PR уже открыт, делать нечего»
- Агент — это пользователь вашего API: идемпотентность, ретрай, понятные ошибки



Итоги задачи

16

дней заняло
открытие
с агентом — было 45

~\$3

стоила работа агента
за всё открытие

140

подшагов ведёт агент —
все до одного

19

сообщений о проблемах
за две недели

Что изменилось:

процесс лежит шагами в базе, а не в голове
разработчик перестал быть диспетчером

91 подшаг агент закрывает сам
на остальных 49 идёт к людям и ждёт ответа

Календарь во многом задают чужие
команды и релизные окна — его агент не
сжал.

Первое открытие: Дринкит в Катаре,
и оно ещё не закончено.

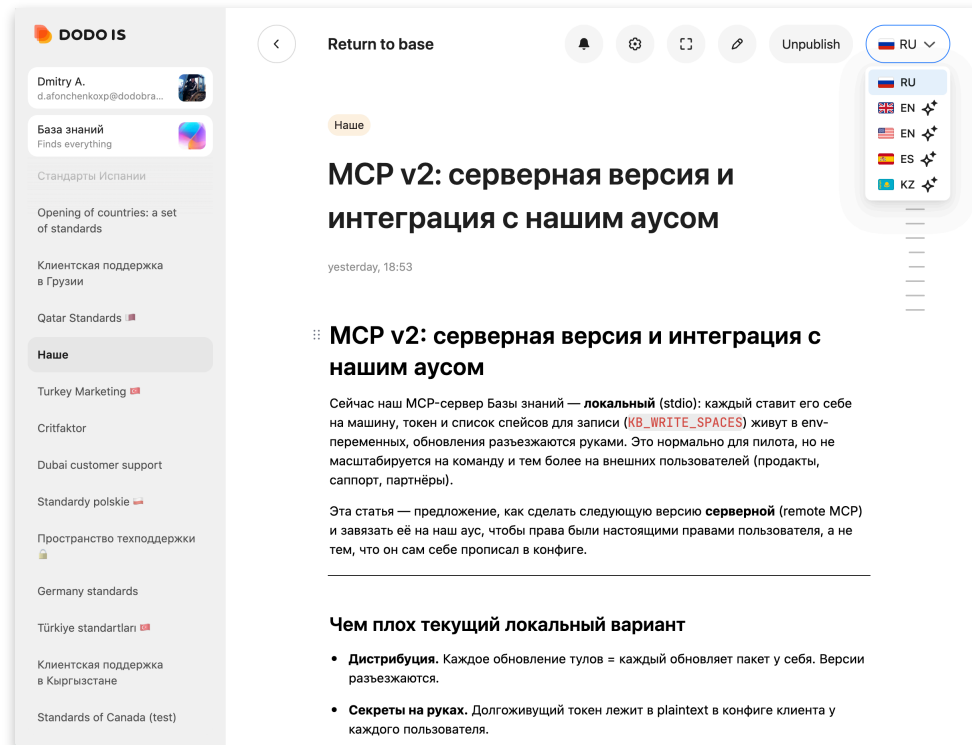
Наши рабочие задачи

Улучшаем качество AI-переводов

Улучшаем качество AI переводов

Проблема

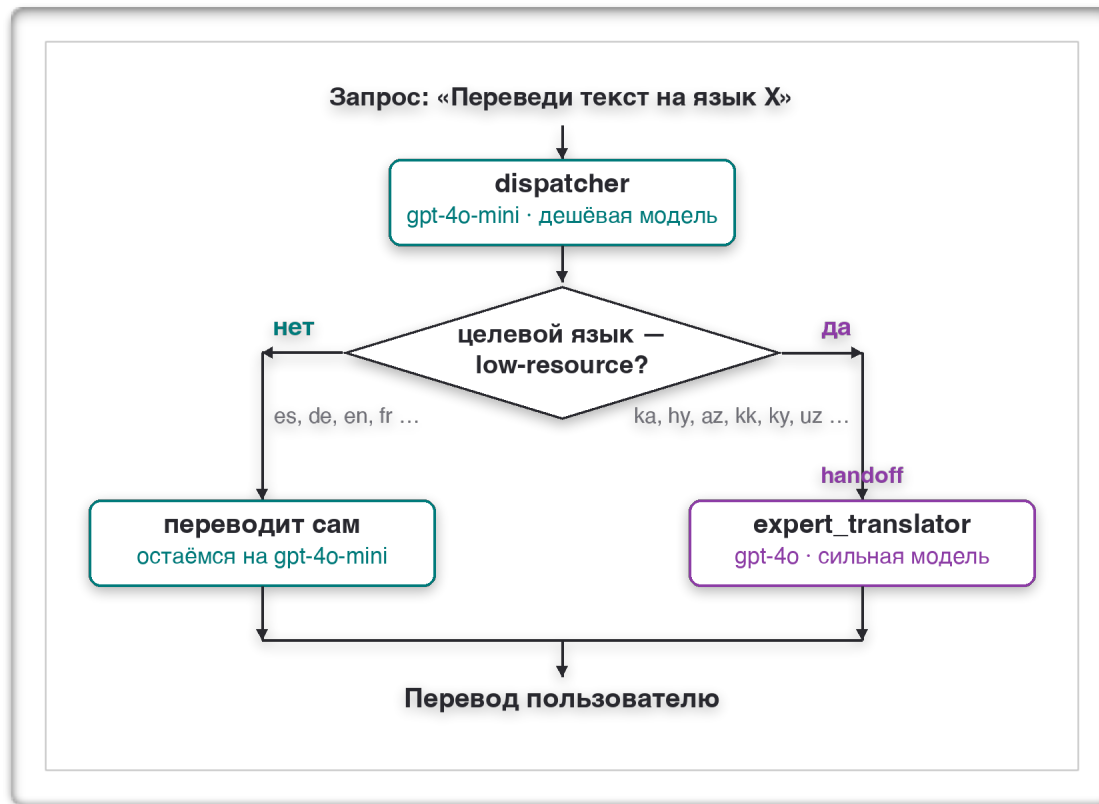
- Малые языки плохо переводятся дешевыми моделями



Демо: базовое качество переводов



Улучшаем качество AI переводов



Workflow для переводов

Workflow

- Автоматическое переключение между дешёвыми и дорогими моделями

```
AIAgent dispatcher = new ChatClientAgent(  
    mini, // gpt-4o-mini – дешёвая модель  
    name: "dispatcher",  
    instructions: "Translate the user's text into the requested " +  
                  "language. If the target language is low-resource " +  
                  "(ka, hy, az, ...) – hand off to expert_translator.");  
  
AIAgent expert = new ChatClientAgent(  
    strong, // gpt-4o – сильная модель  
    name: "expert_translator",  
    instructions: "You are an expert translator " +  
                  "for low-resource languages.");  
  
Workflow workflow = AgentWorkflowBuilder  
    .CreateHandoffBuilderWith(dispatcher)  
    .WithHandoff(dispatcher, expert, "Target language is low-resource")  
    .Build();
```

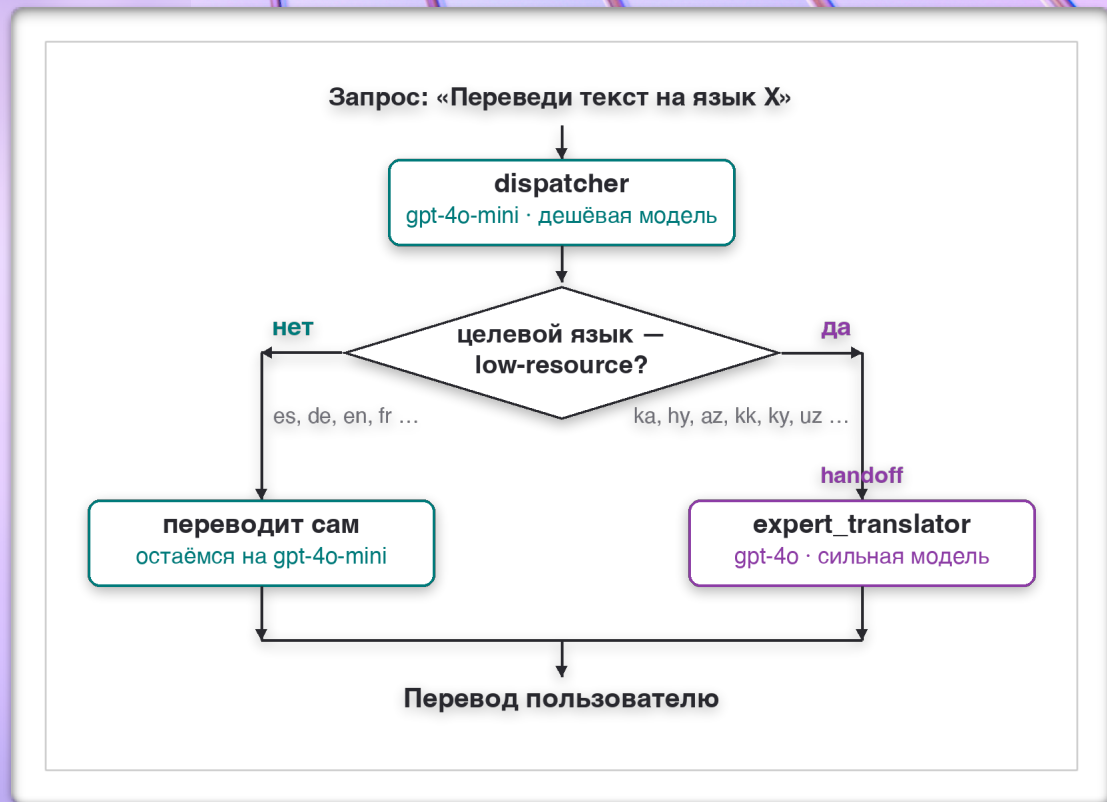
Workflow для переводов

Workflow

- Автоматическое переключение между дешевыми и дорогими моделями

```
AIAgent translator = workflow.AsAIAgent();  
  
var reply = await translator.RunAsync(  
    "Translate into Georgian (ka): Встречайте Додстер-комбо...");
```

Демо: диспетчер переводов



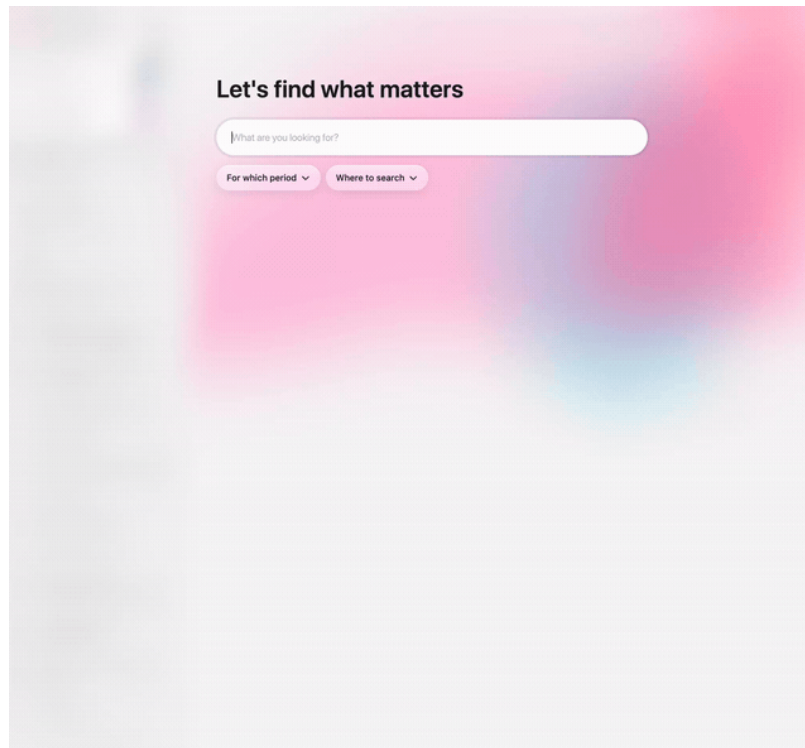
Демо: диспетчер переводов



Можно ли делать что-то сложнее?

Агенту нужен интерфейс

- 1 RunAsync отдаёт всё разом
- 2 А хочется видеть, что агент делает по дороге
- 3 И хочется видеть как он генерирует ответ



AG-UI: две строчки на сервере

AG-UI
поверх SSE

```
var builder = WebApplication.CreateBuilder();  
builder.Services.AddAGUIServer(); // ← раз  
  
var app = builder.Build();  
app.UseDefaultFiles();  
app.UseStaticFiles();  
app.MapAGUIServer("/agui", agent); // ← два  
  
await app.RunAsync("http://localhost:5199");
```

Фронтенд: руками и с библиотекой

```
// разбираем поток руками
const reader = res.body.getReader();
let buffer = '';

while (true) {
  const { value, done } = await reader.read();
  if (done) break;
  buffer += decoder.decode(value, { stream: true });

  const parts = buffer.split('\n\n');
  buffer = parts.pop();

  for (const part of parts) {
    const line = part.split('\n')
      .find(l => l.startsWith('data: '));
    if (!line) continue;
    const ev = JSON.parse(line.slice(6));

    switch (ev.type) {
      case 'TOOL_CALL_START': addChip(ev.toolCallName); break;
      case 'TOOL_CALL_RESULT': fillChip(ev.toolCallId, ev.content); break;
      case 'TEXT_MESSAGE_CONTENT': reply += ev.delta ?? ''; break;
      case 'RUN_ERROR': showError(ev.message); break;
    }
  }
}
```

```
// то же самое через @ag-ui/client
import { HttpAgent } from "../vendor/ag-ui-client.mjs";

const agent = new HttpAgent({ url: "/agui" });
agent.messages.push({ id: crypto.randomUUID(), role: "user", content: text });

// библиотека сама читает SSE и раскладывает поток на типизированные события
await agent.runAgent({}, {
  onToolCallStartEvent: ({ event }) => addChip(event.toolCallName),
  onToolCallResultEvent: ({ event }) => fillChip(event.toolCallId, event.content),
  onTextMessageContentEvent: ({ event }) => append(event.delta),
  onRunErrorEvent: ({ event }) => showError(event.message),
});
```

Демо: AG-UI интерфейс



Грабли и зрелость

- Постобработка потока — только через middleware
- SSE нельзя буферизовать — наш WAF буферизовал
- Весь интерфейс собрали за две недели
- Библиотеки сырые:
 - Microsoft.Agents.AI.Hosting.AGUI.AspNetCore 1.18.0-preview
 - AGUI.Abstractions / AGUI.Server 0.0.3
 - @ag-ui/client 0.0.59

Итого

- Детерминированное в код, недетерминированное в модель
- Давать модели маленькие шаги лучше чем давать процесс с кучей шагов
 - Делайте инструменты для агента так же как вы делаете их для пользователей: идемпотентность, понятные ошибки, ретраи работают тут так же хорошо
 - MAF — это не только про большие и сложные пайплайны и RAG. На нем легко собрать и небольших прикладных агентов

Когда что использовать

Инструмент	Когда брать
Coding agent + MCP	разовая или исследовательская задача, человек рядом и смотрит
MAF + свои тулы	повторяемый процесс, порядок шагов известен, нужны права и human-in-the-loop

Ссылки



Демки доклада



Документация
MAF



Исходник MAF

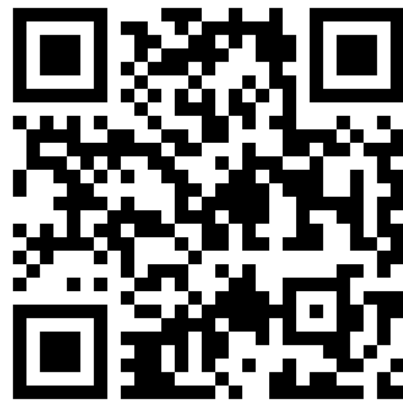


Доклад по
Semantic Kernel



**Спасибо
за внимание!**

Вопросы?



@dimasshortposts