Microsoft Agent Framework: dai i superpoteri alla tua app!

Alessio Bianchini e Moreno Gentili
DoIT

aspitalia.com

Lodestar
IT'S TIME TO INNOVATE

.NET Conference Italia 2026

My travel application

Hotels | Flights

CHECK-IN – CHECK-OUT
12 Dec – 19 Dec 2026

Indonesia

6 stays you might like



Amanjiwo Resort

Pool Villa · Ocean view

★ 4.9 **$420** / night



Alila Uluwatu

Deluxe Suite · Cliff side

★ 4.8 **$310** / night



The Mulia Bali

Junior Suite · Garden

★ 4.7 **$275** / night



Ubud Hanging Gardens

Jungle Room · Terrace

★ 4.6 **$190** / night

Demo

Diamo i superpoteri a questa app!

.NET Conference
Italia 2026

✨ Lasciati ispirare

Voglio visitare un Paese esotico. Prenotami un viaggio di 4 notti tra l'1 e il 20 dicembre.



Partenza il 01/12/2026 — 4 notti

VOLO **Zephyr Jet**
✈ ZJ7108
Partenza: Rome il 01/12/2026 12:40
Arrivo: Bangkok il 02/12/2026 00:10

HOTEL **The Ivory Gate**
Executive Room

~~Bangkok~~ ~~EUR 776.00~~



Partenza il 02/12/2026 — 4 notti

VOLO **Tramontana Air**
✈ TX3260
Partenza: Rome il 02/12/2026 20:00
Arrivo: Bora Bora il 02/12/2026 12:20

HOTEL **The Crimson Oak**
Azure Room

Hai scelto Bora Bora EUR 1818.00



Partenza il 12/12/2026 — 4 notti

VOLO **Pelican Pacific**
✈ PQ7154
Partenza: Rome il 12/12/2026 19:40
Arrivo: Zanzibar il 13/12/2026 03:40

HOTEL **Villa Serenita**
Balcony Room

~~Zanzibar~~ ~~EUR 998.00~~

Checkout: viaggio a Bora Bora

- Paypal
- Carta di credito

Annulla | Paga EUR 1818.00

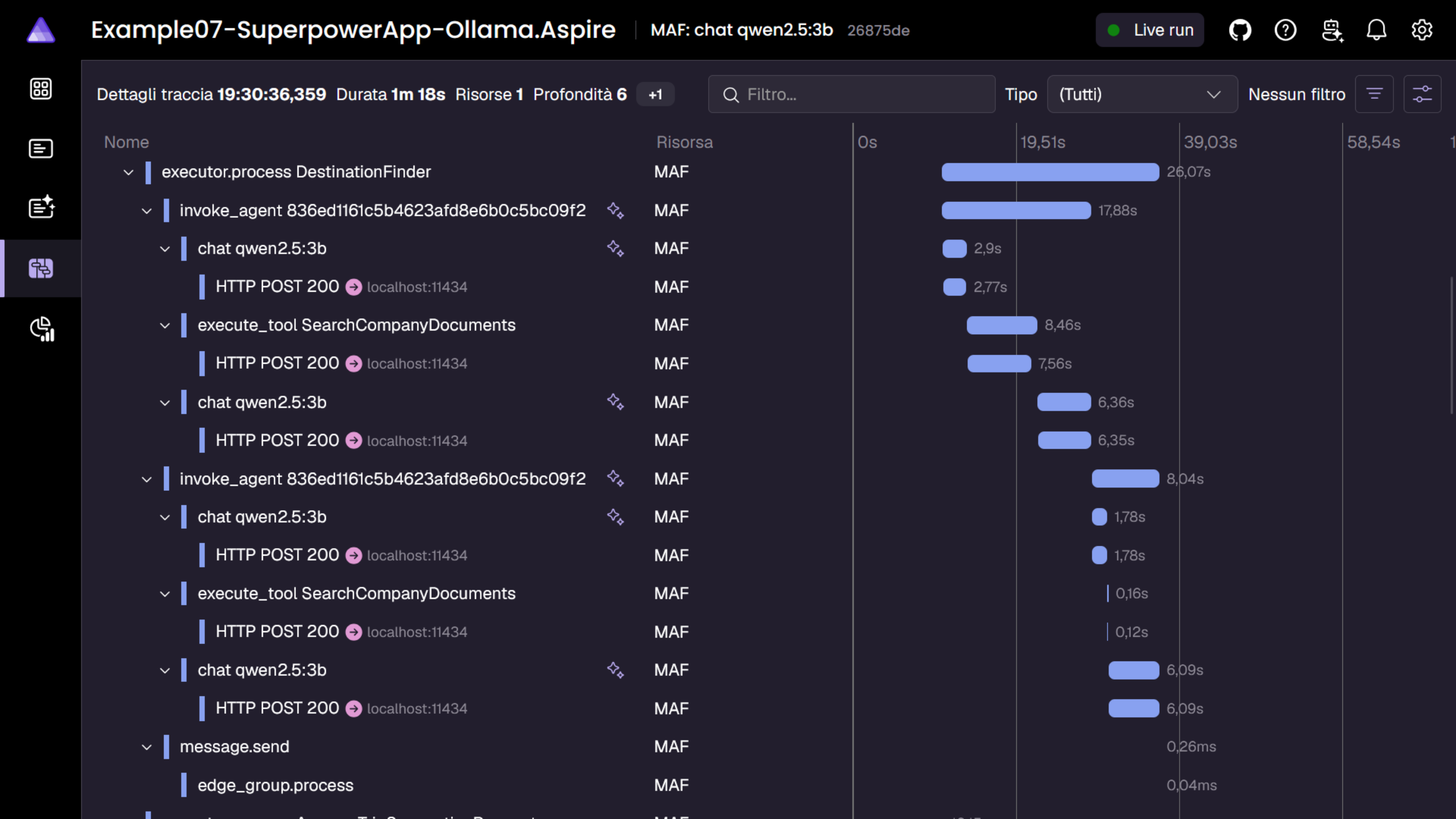
Si va a Bora Bora!

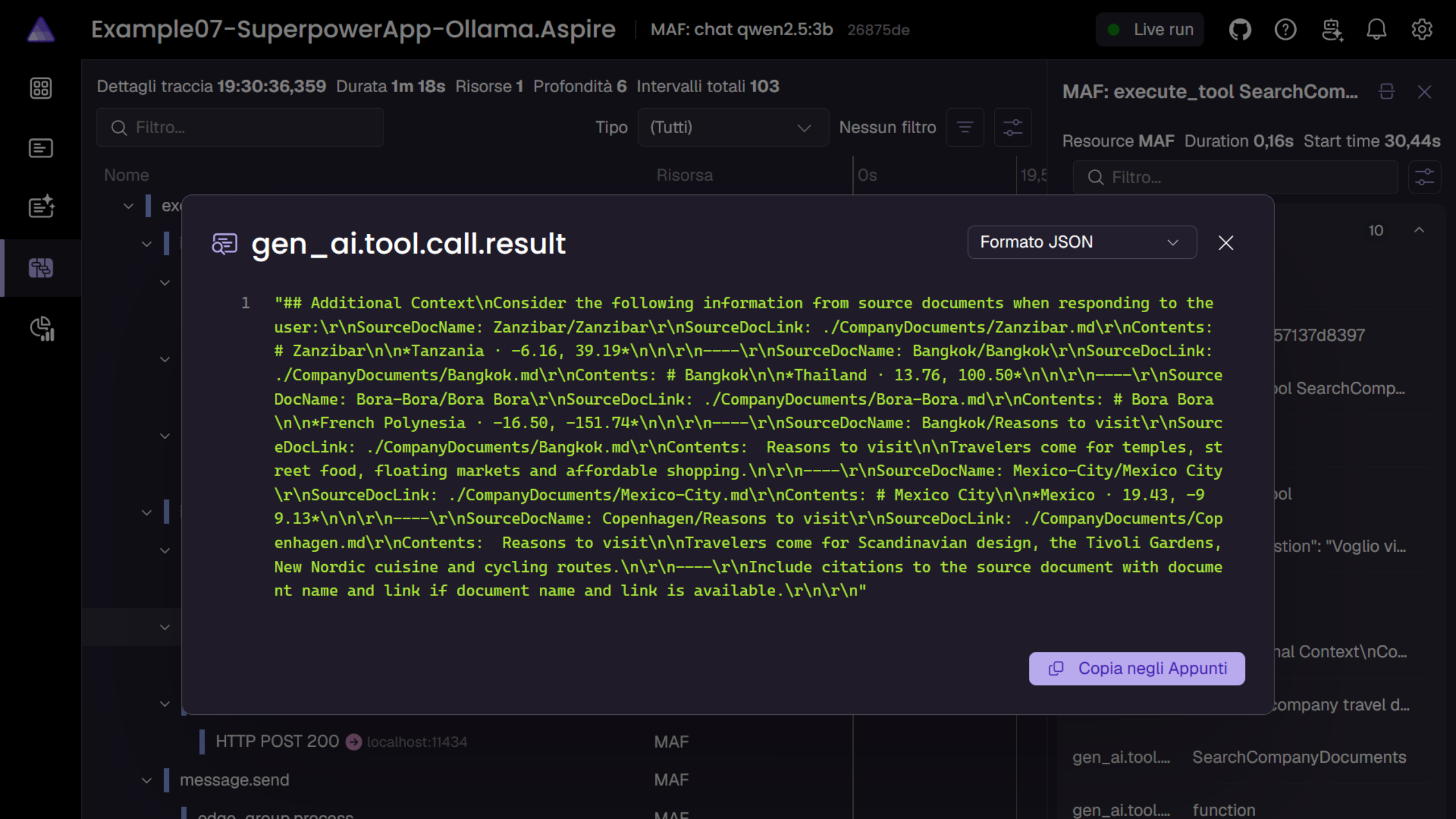
PNR: RJ97EM **Transaction ID:** TXN-20261118-1A02B1F3517C

Demo

Osservare la “traiettoria” del flusso agentico con Aspire.

.NET Conference Italia 2026


Example07-SuperpowerApp-Ollama.Aspire
MAF: chat qwen2.5:3b 26875de
Live run
Dettagli traccia 19:30:36,359 Durata 1m 18s Risorse 1 Profondità 6 +1
Filtro...
Tipo (Tutti)
Nessun filtro
Nome
Risorsa
0s
19,51s
39,03s
58,54s
executor.process DestinationFinder MAF 26,07s
invoke_agent 836ed1161c5b4623afd8e6b0c5bc09f2 MAF 17,88s
chat qwen2.5:3b MAF 2,9s
HTTP POST 200 localhost:11434 MAF 2,77s
execute_tool SearchCompanyDocuments MAF 8,46s
HTTP POST 200 localhost:11434 MAF 7,56s
chat qwen2.5:3b MAF 6,36s
HTTP POST 200 localhost:11434 MAF 6,35s
invoke_agent 836ed1161c5b4623afd8e6b0c5bc09f2 MAF 8,04s
chat qwen2.5:3b MAF 1,78s
HTTP POST 200 localhost:11434 MAF 1,78s
execute_tool SearchCompanyDocuments MAF 0,16s
HTTP POST 200 localhost:11434 MAF 0,12s
chat qwen2.5:3b MAF 6,09s
HTTP POST 200 localhost:11434 MAF 6,09s
message.send MAF 0,26ms
edge_group.process MAF 0,04ms


Example07-SuperpowerApp-Ollama.Aspire
MAF: chat qwen2.5:3b 26875de
Live run
Dettagli traccia 19:30:36,359 Durata 1m 18s Risorse 1 Profondità 6 Intervalli totali 103
Filtro...
Tipo (Tutti)
Nessun filtro
Nome
Risorsa
0s
MAF: execute_tool SearchCom...
Resource MAF Duration 0,16s Start time 30,44s
Filtro...
gen_ai.tool.call.result
Formato JSON
1 "## Additional Context\nConsider the following information from source documents when responding to the user:\r\nSourceDocName: Zanzibar/Zanzibar\r\nSourceDocLink: ./CompanyDocuments/Zanzibar.md\r\nContents: # Zanzibar\n\n*Tanzania · -6.16, 39.19*\n\n\r\n----\r\nSourceDocName: Bangkok/Bangkok\r\nSourceDocLink: ./CompanyDocuments/Bangkok.md\r\nContents: # Bangkok\n\n*Thailand · 13.76, 100.50*\n\n\r\n----\r\nSourceDocName: Bora-Bora/Bora Bora\r\nSourceDocLink: ./CompanyDocuments/Bora-Bora.md\r\nContents: # Bora Bora\n\n*French Polynesia · -16.50, -151.74*\n\n\r\n----\r\nSourceDocName: Bangkok/Reasons to visit\r\nSourceDocLink: ./CompanyDocuments/Bangkok.md\r\nContents: Reasons to visit\n\nTravelers come for temples, street food, floating markets and affordable shopping.\n\r\n----\r\nSourceDocName: Mexico-City/Mexico City\r\nSourceDocLink: ./CompanyDocuments/Mexico-City.md\r\nContents: # Mexico City\n\n*Mexico · 19.43, -99.13*\n\n\r\n----\r\nSourceDocName: Copenhagen/Reasons to visit\r\nSourceDocLink: ./CompanyDocuments/Copenhagen.md\r\nContents: Reasons to visit\n\nTravelers come for Scandinavian design, the Tivoli Gardens, New Nordic cuisine and cycling routes.\n\r\n----\r\nInclude citations to the source document with document name and link if document name and link is available.\r\n\r\n"
Copia negli Appunti
HTTP POST 200 localhost:11434
MAF
message.send
MAF
gen_ai.tool....
SearchCompanyDocuments
gen_ai.tool....
function

Risorsa MAF

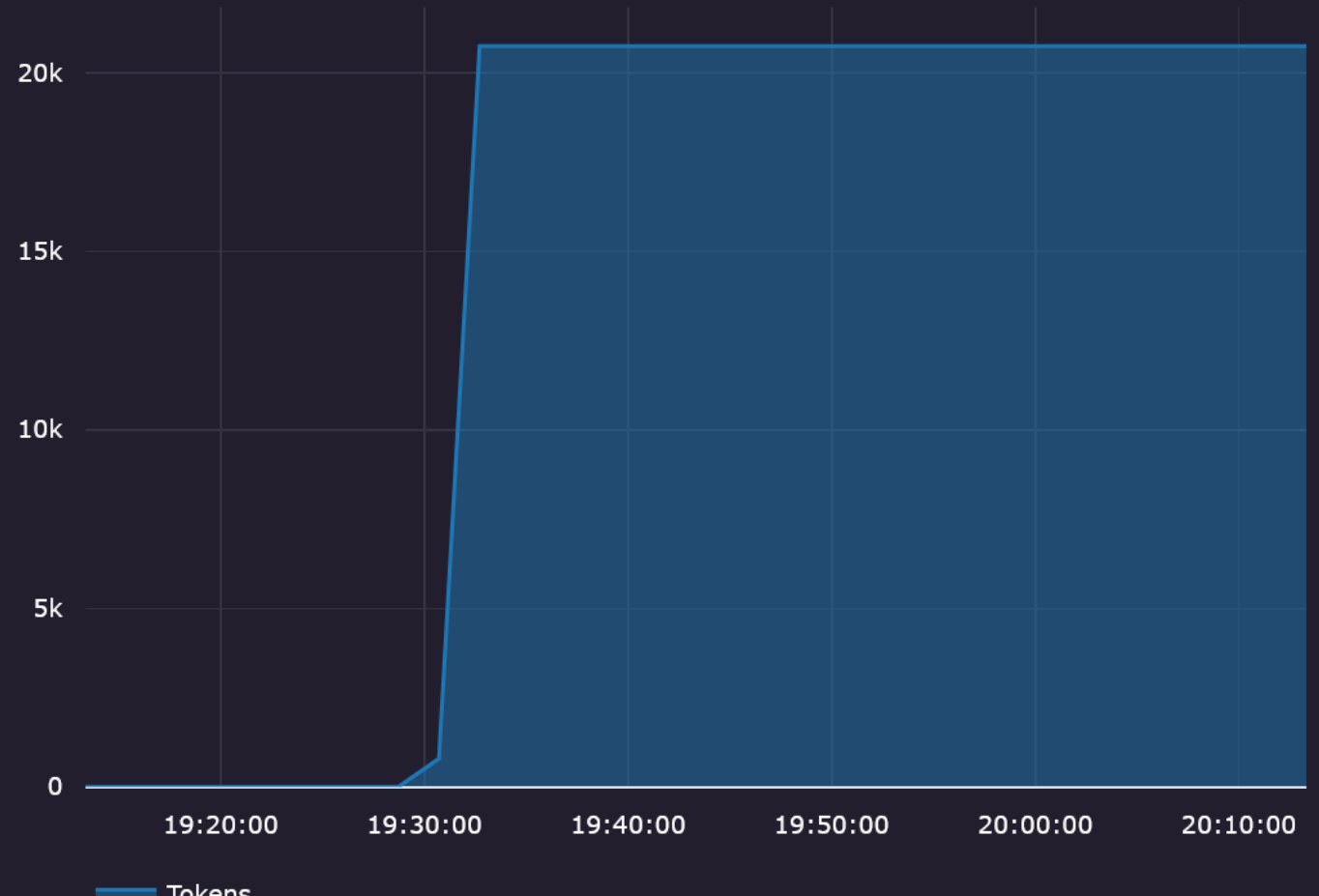
Durata Ultima ora

- MafDemo
  - gen_ai.client.operation.duration
  - gen_ai.client.token.usage
  - mafdemo.ai.tokens.total
- System.Net.Http
  - http.client.active_requests
  - http.client.connection.duration
  - http.client.open_connections
  - http.client.request.duration
  - http.client.request.time_in_qu...
- System.Net.NameResolution
  - dns.lookup.duration

mafdemo.ai.tokens.total

Cumulative input and output tokens used

Grafo | Risorse



Cosa abbiamo usato?

Microsoft Agent Framework

Architettura estendibile

È una SDK unificata



Open source e multipiattaforma

Collaborazione multi-agente

Capacità di ragionamento

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

...ma solo per destinazioni note

Sì, ma da dove vuole partire?

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

Dovrà collegarsi al DB o ad API

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

Dovrà chiedere conferma all'utente

Voglio visitare un Paese esotico.
Prenotami un viaggio di 4 notti
tra l'1 e il 20 dicembre.

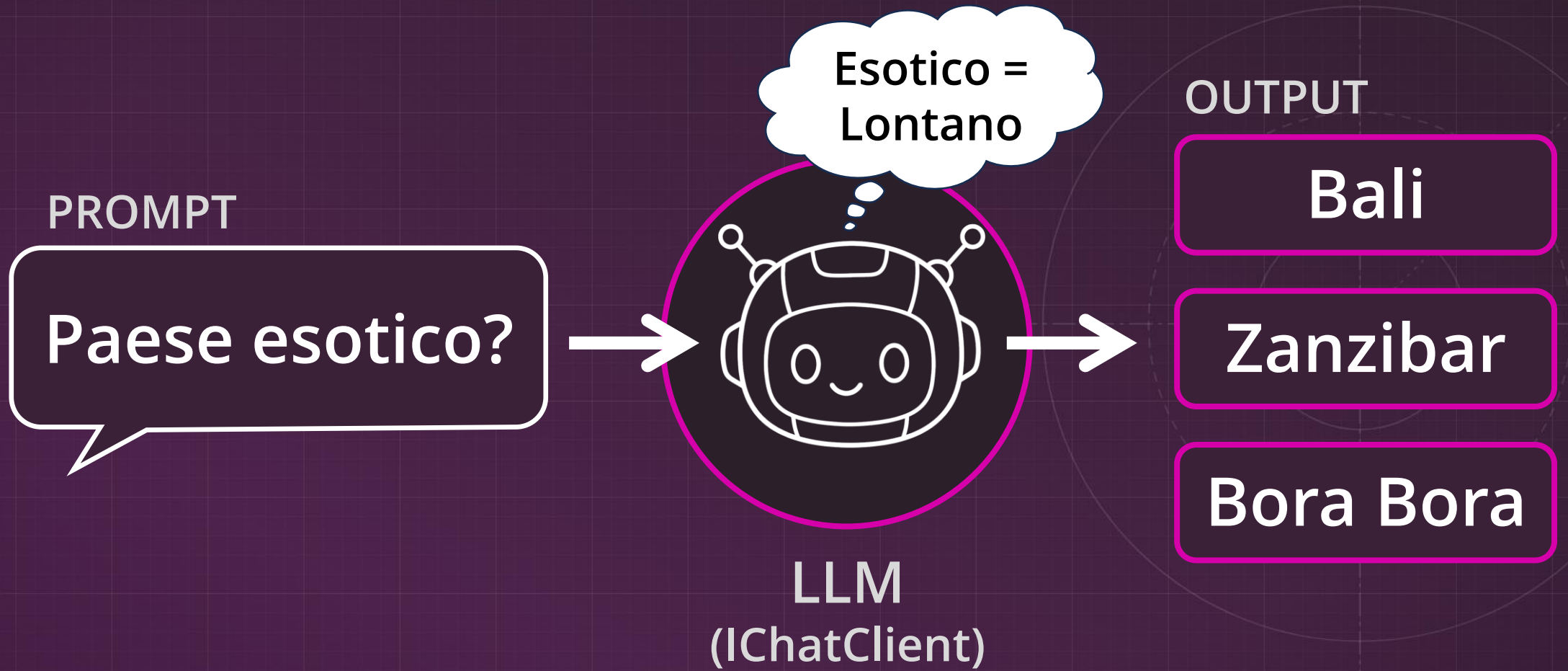
Dovremo proteggerci da usi impropri

Requisiti non funzionali

1. Guardrail: regolare il comportamento
2. Trajectory: integrazione con OpenTelemetry
3. Protocol: accesso dal frontend con AG-UI

OBIETTIVO 1

Aggiungere capacità di ragionamento



```
using Microsoft.Extensions.AI;
using OllamaSharp;

IChatClient chatClient = new OllamaApiClient("http://...", "qwen2.5:3b");

string prompt = "Suggerisci 3 mete di viaggio esotiche indicando luogo e Paese.";
ChatResponse response =
    await chatClient.GetResponseAsync(prompt);

response.Text       // Ti propongo le seguenti 3 mete: Bali in Indonesia,
                    // Bora Bora, isola tropicale nella Polinesia Francese e
                    // Svalbard, nell'estrema parte nord della Norvegia.
```

```
using Microsoft.Extensions.AI;
using OllamaSharp;

IChatClient chatClient = new OllamaApiClient("http://...", "qwen2.5:3b");

string prompt = "Suggerisci 3 mete di viaggio esotiche indicando città e Paese.";
ChatResponse<List<Destination>> response =
    await chatClient.GetResponseAsync<List<Destination>>(prompt);

response.Result    // [
                   //   { City: "Bali", Country: "Indonesia")
                   //   { City: "Bora Bora", Country: "Polinesia francese")
                   //   { City: "Svalbard", Country: "Norvegia")
                   // ]
```

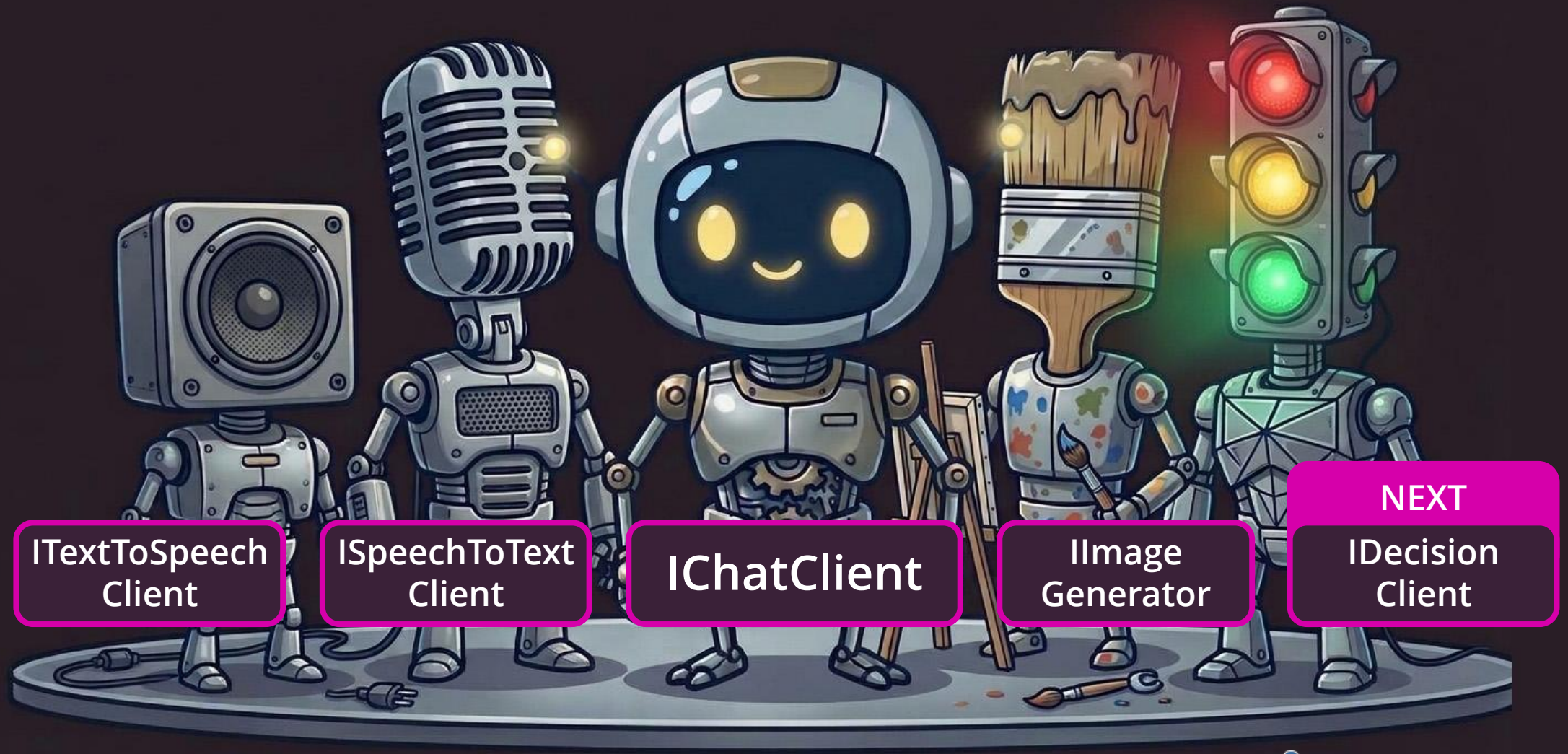
```
using Microsoft.Extensions.AI;
using OllamaSharp;

IChatClient chatClient = new OllamaApiClient("http://...", "qwen2.5:3b");

string prompt = "Suggerisci 3 mete di viaggio esotiche indicando città e Paese.";
ChatResponse response =
    await chatClient.GetResponseAsync(prompt);

response.Messages  // - ChatMessage
                   //     Contents:
                   //     - TextContent
                   //     - DataContent
                   //     - FunctionCall
```

Astrazioni di Microsoft.Extensions.AI



OBIETTIVO 2

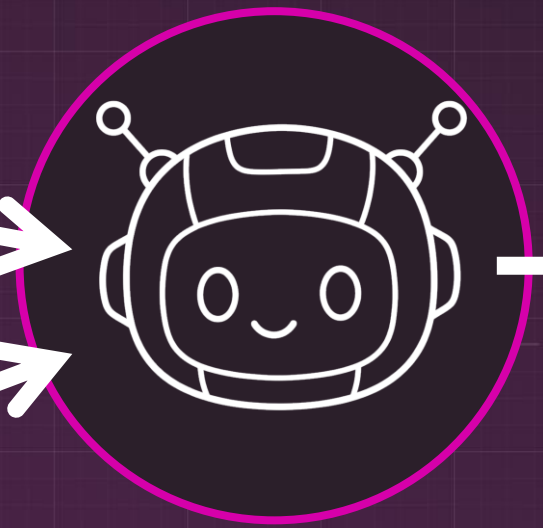
Integrare il prompt con info di contesto

PROMPT

Paese esotico?

CONTEXT PROVIDER

L’utente è di Bali



LLM
(AIAgent)

OUTPUT

- Lapponia
- Tokyo
- Namibia

```
using Microsoft.Agents.AI;
using Microsoft.Extensions.AI;
using OllamaSharp;

IChatClient chatClient = new OllamaApiClient("http://...", "qwen2.5:3b");

AIAgent agent = chatClient.AsAIAgent(
  new ChatClientAgentOptions {
    ChatOptions = new ChatOptions {
      Instructions = "Sei un agente di viaggio. Dai consigli in base al contesto."
    },
    AIContextProviders = [new UserProfileContextProvider()]
  });

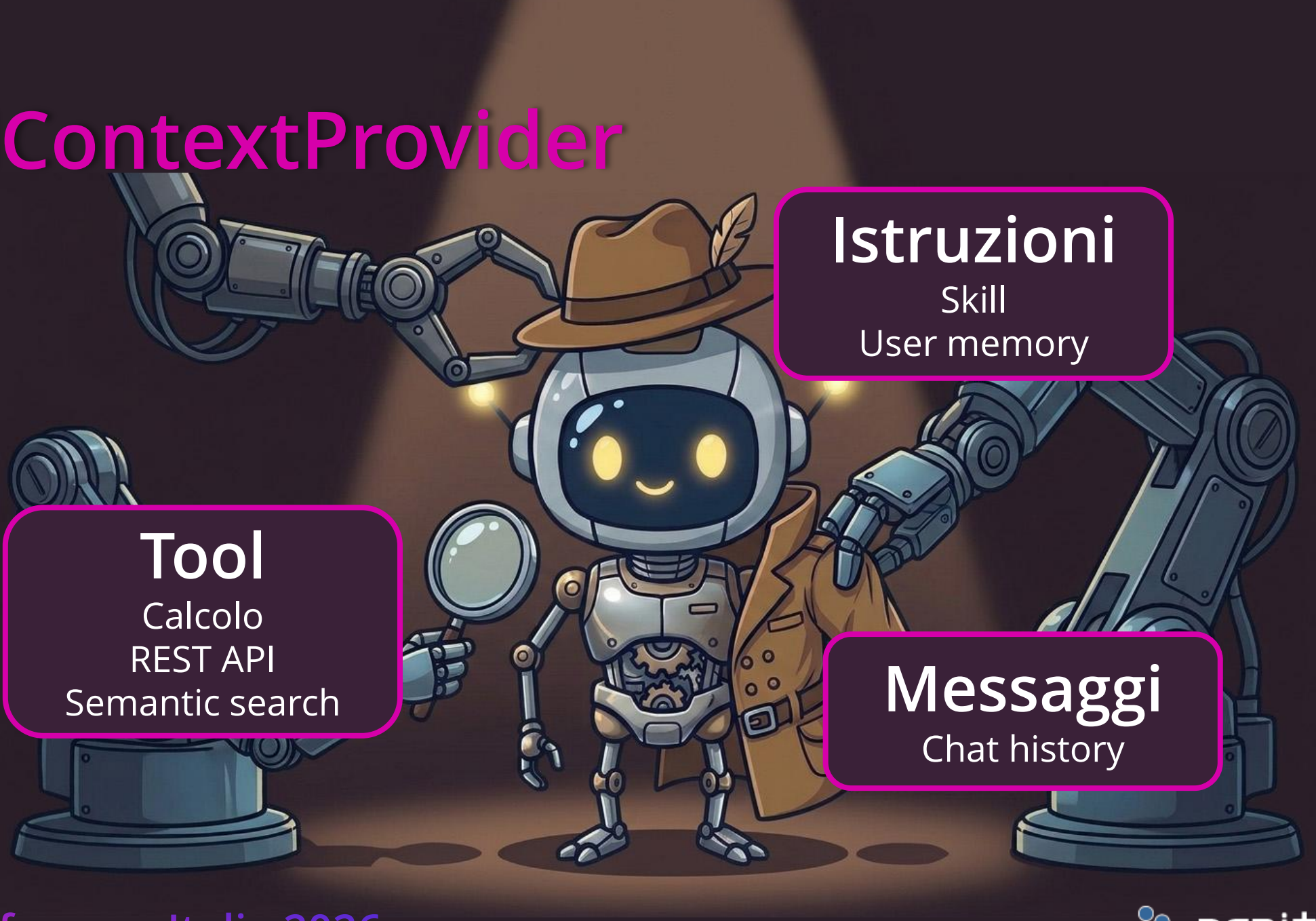
AgentResponse response = await agent.RunAsync("Consigliami mete esotiche");
```

```
using Microsoft.Agents.AI;

class UserProfileContextProvider : AIContextProvider
{
  protected override async ValueTask<AIContext> ProvideAIContextAsync(
    InvokingContext context, CancellationToken cancellationToken) =>
      new AIContext
      {
        Instructions = "L'utente abita a Bali", // TODO: recuperale dal DB

        // Altre possibilità
        Messages = [new ChatMessage(ChatRole.User, "Fammi lo sconto, eh")],
        Tools = [getWeatherTool]
      };
}
```

AIContextProvider



Istruzioni
Skill
User memory

Tool
Calcolo
REST API
Semantic search

Messaggi
Chat history

AIAgent



OBIETTIVO 3

Ancorare le risposte ai dati aziendali

PROMPT

Paese esotico?

CONTEXT PROVIDER



LLM
(AIAgent)

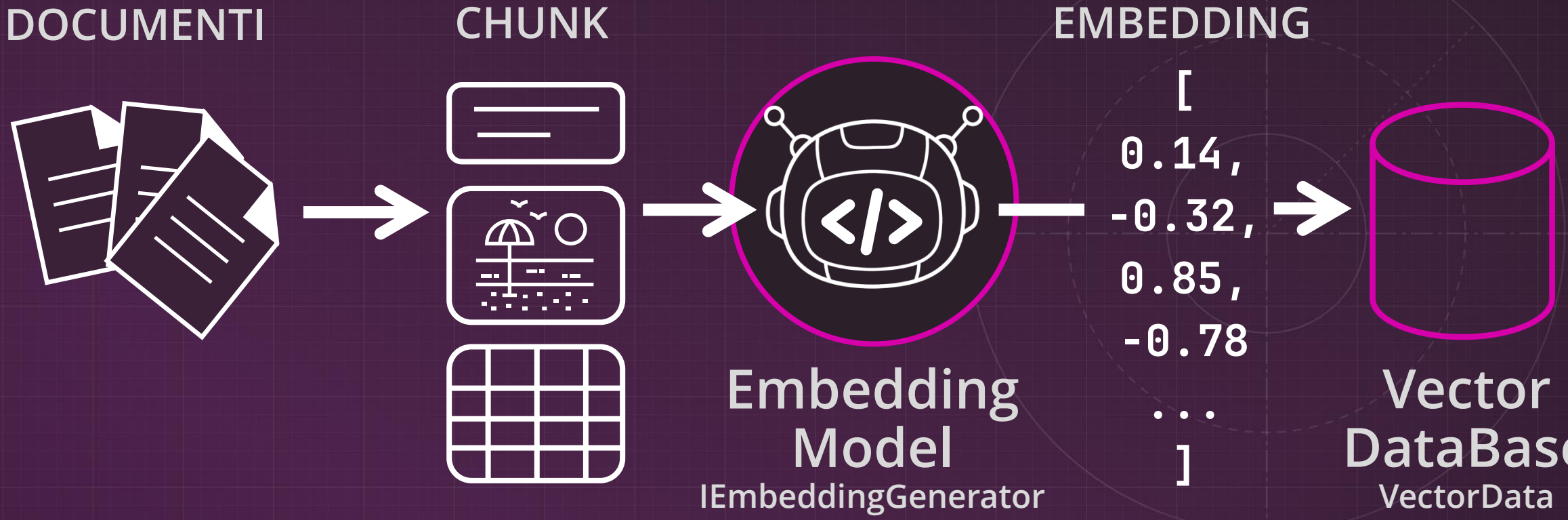
OUTPUT

Maldive

Namibia

OBIETTIVO 3 - PREPARAZIONE

Ingestione dei documenti aziendali



```
using Microsoft.Extensions.DataIngestion;

// Creazione e avvio della pipeline ETL
using IngestionPipeline<string> pipeline = new(reader, chunker, writer);

IngestionResult[] results = await pipeline
    .ProcessAsync(new DirectoryInfo("./docs", searchPattern: "*.md"))
    .ToArrayAsync();
```

```
using Microsoft.Extensions.DataIngestion;
using Microsoft.ML.Tokenizers;

// Reader
IngestionDocumentReader reader = new MarkdownReader();

// Chunker
Tokenizer tokenizer = TiktokenTokenizer.CreateForEncoding("cl100k_base");
IngestionChunkerOptions options = new(tokenizer)
{
  MaxTokensPerChunk = 1000,
  OverlapTokens = 500
};

IngestionChunker<string> chunker = new HeaderChunker(options);
```

```
using Microsoft.Extensions.AI;
using OllamaSharp;
using CommunityToolkit.VectorData.SqliteVec;

// Embedding generator
IEmbeddingGenerator<string, Embedding<float>> embeddingGenerator =
            new OllamaApiClient("http://...", "nomic-embed-text");

// Writer
SqliteVectorStore vectorStore = new(
  "Data Source=./vectors.db",
   new() { EmbeddingGenerator = embeddingGenerator });

VectorStoreWriter<string> writer = new(vectorStore, dimensionCount: 768);
```

OBIETTIVO 3

Ancorare le risposte ai dati aziendali

PROMPT

Paese esotico?

CONTEXT PROVIDER



LLM
(AIAgent)

OUTPUT

Maldive

Namibia

```
using Microsoft.Agents.AI;

AIContextProvider textSearchProvider = new TextSearchProvider(
  async (string prompt, CancellationToken ct) => {
    ReadOnlyMemory<float> e = await embeddingGenerator.GenerateVectorAsync(prompt);
    return await collection
                  .SearchAsync(e, top: 3, cancellationToken: ct)
                  .Select(e => e.Record.ToSearchResult())
                  .ToListAsync(ct);
  },
  new TextSearchProviderOptions
  {
    SearchTime = TextSearchBehavior.BeforeAIInvoke
  });
```

```
using Microsoft.Agents.AI;

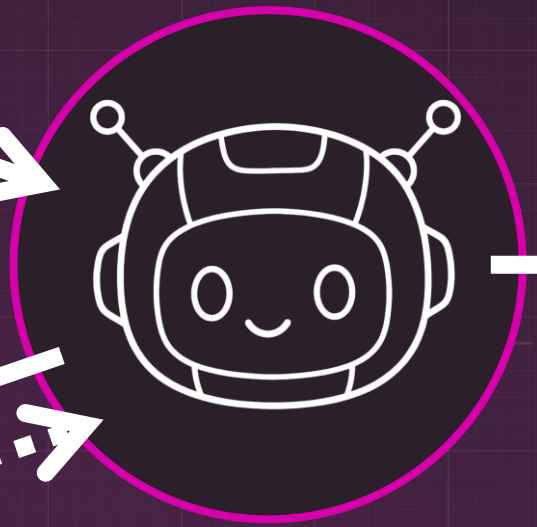
AIContextProvider textSearchProvider = new TextSearchProvider(
  async (string prompt, CancellationToken ct) => {
    ReadOnlyMemory<float> e = await embeddingGenerator.GenerateVectorAsync(prompt);
    return await collection
                  .SearchAsync(e, top: 3, cancellationToken: ct)
                  .Select(e => e.Record.ToSearchResult())
                  .ToListAsync(ct);
  },
  new TextSearchProviderOptions
  {
    SearchTime = TextSearchBehavior.OnDemandFunctionCalling
  });
```

OBIETTIVO 4

Consentire al modello di agire

PROMPT

Volo per Bali?



OUTPUT

Il 5/12 h9.00

TOOL

- REST API
- MCP Server
- Codice C#

LLM
(AIAgent con
function calling)

```
using Microsoft.Agents.AI;

decimal CalculateTotal(decimal flightCost, decimal roomCost) =>
    flightCost + roomCost + 25.00m; // Commission fee

AIFunction calculateTotalTool = AIFunctionFactory.Create(CalculateTotal);

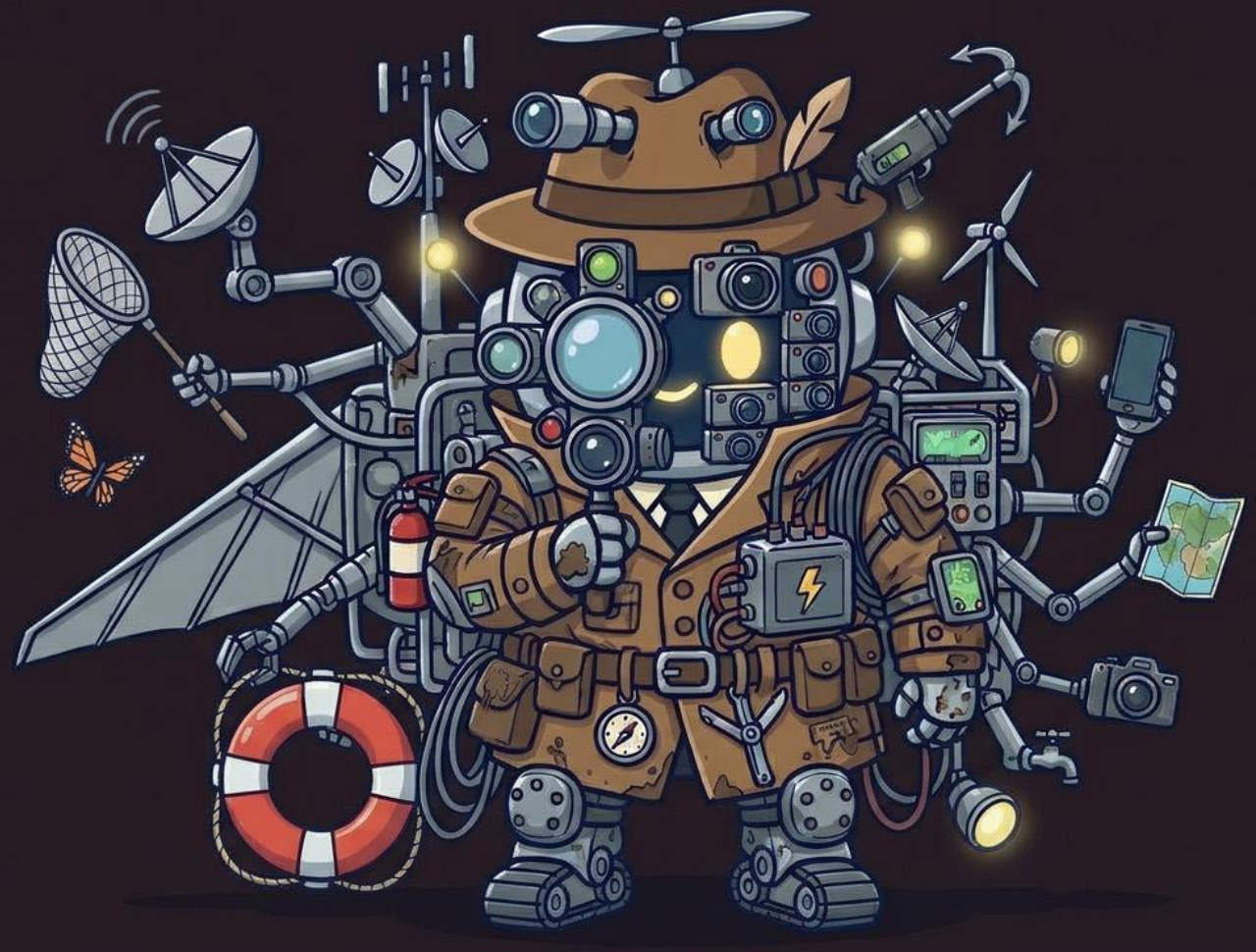
AIAgent agent = chatClient.AsAIAgent(
    new ChatClientAgentOptions {
        ChatOptions = new ChatOptions {
            Instructions = "Calcola il totale con il tool `CalculateTotal`",
            Tools = [calculateTotalTool],
            ToolMode = ChatToolMode.RequireAny
        }
    });
```

```
using Microsoft.Agents.AI;
using ModelContextProtocol.Client;

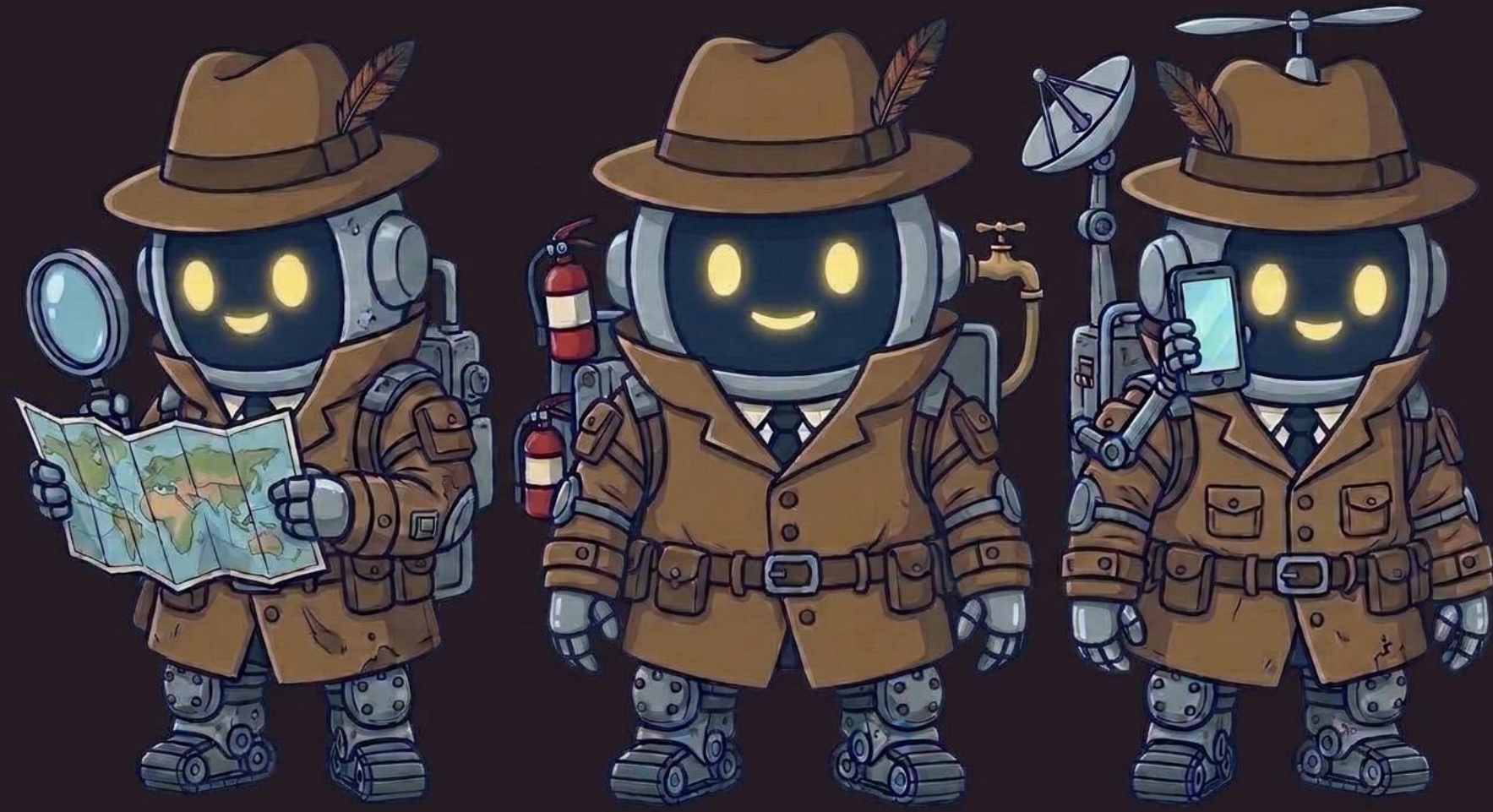
McpClient mcpClient = await McpClient.CreateAsync(new HttpClientTransport(new() {
  Endpoint = new Uri("https://mcp.hotelscasa.com/mcp")
}));

AIAgent agent = chatClient.AsAIAgent(
    new ChatClientAgentOptions {
        ChatOptions = new ChatOptions {
            Instructions = "Cerca disponibilità usando i tool",
            Tools = await mcpClient.ListToolsAsync(),
            ToolMode = ChatToolMode.RequireAny
        }
    });
```

Tool ne abbiamo?



Specializzare gli agenti



OBIETTIVO 5

Chiedere conferma all'utente



```
using Microsoft.Agents.AI;

decimal Pay(decimal amount, string currency) => // Qualcosa

AIFunction payTool = AIFunctionFactory.Create(Pay);
ApprovalRequiredAIFunction payToolWithApproval = new(payTool);

AIAgent agent = chatClient.AsAIAgent(
    new ChatClientAgentOptions {
        ChatOptions = new ChatOptions {
            Instructions = "Paga usando il tool",
            Tools = [payToolWithApproval],
            ToolMode = ChatToolMode.RequireAny
        }
    });
```

```
using Microsoft.Agents.AI;

AgentResponse response = await agent.RunAsync("Paga 100 euro", session);

ToolApprovalRequestContent approvalRequest = response.Messages
    .SelectMany(m => m.Contents)
    .OfType<ToolApprovalRequestContent>()
    .First();

bool approved = true; // Qualche tempo dopo, dall'utente...

ToolApprovalResponseContent response = approvalRequest.CreateResponse(approved);
ChatMessage decision = new(ChatRole.User, [response]);

AgentResponse finalResponse = await agent.RunAsync(decision, session);
```

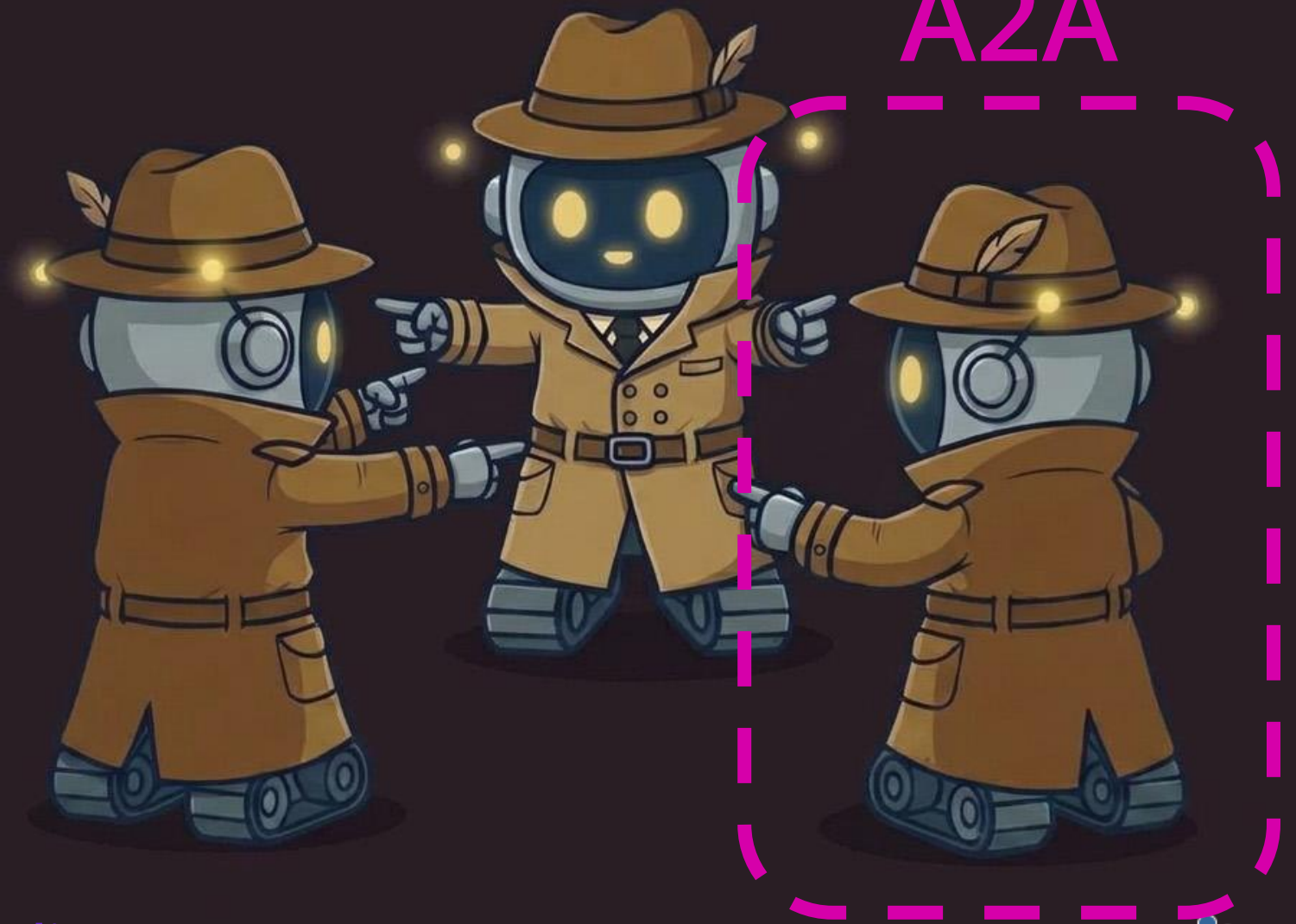
```
using Microsoft.Agents.AI;
using Microsoft.Agents.AI.Hosting;

// Ottenere la sessione dal session store
InMemoryAgentSessionStore sessionStore = new ();
AgentSessionStoreKey key = new AgentSessionStoreKey("UserId-12345");
AgentSession session = await sessionStore.GetOrCreateSessionAsync(agent, key);

// Persistere una sessione dopo averla usata con AIAgent
await sessionStore.SaveSessionAsync(agent, key, session);

// Ripristinare una sessione prima di usarla con un altro AIAgent
AgentSession resumedSession = await sessionStore.GetSessionAsync(agent, key);
```

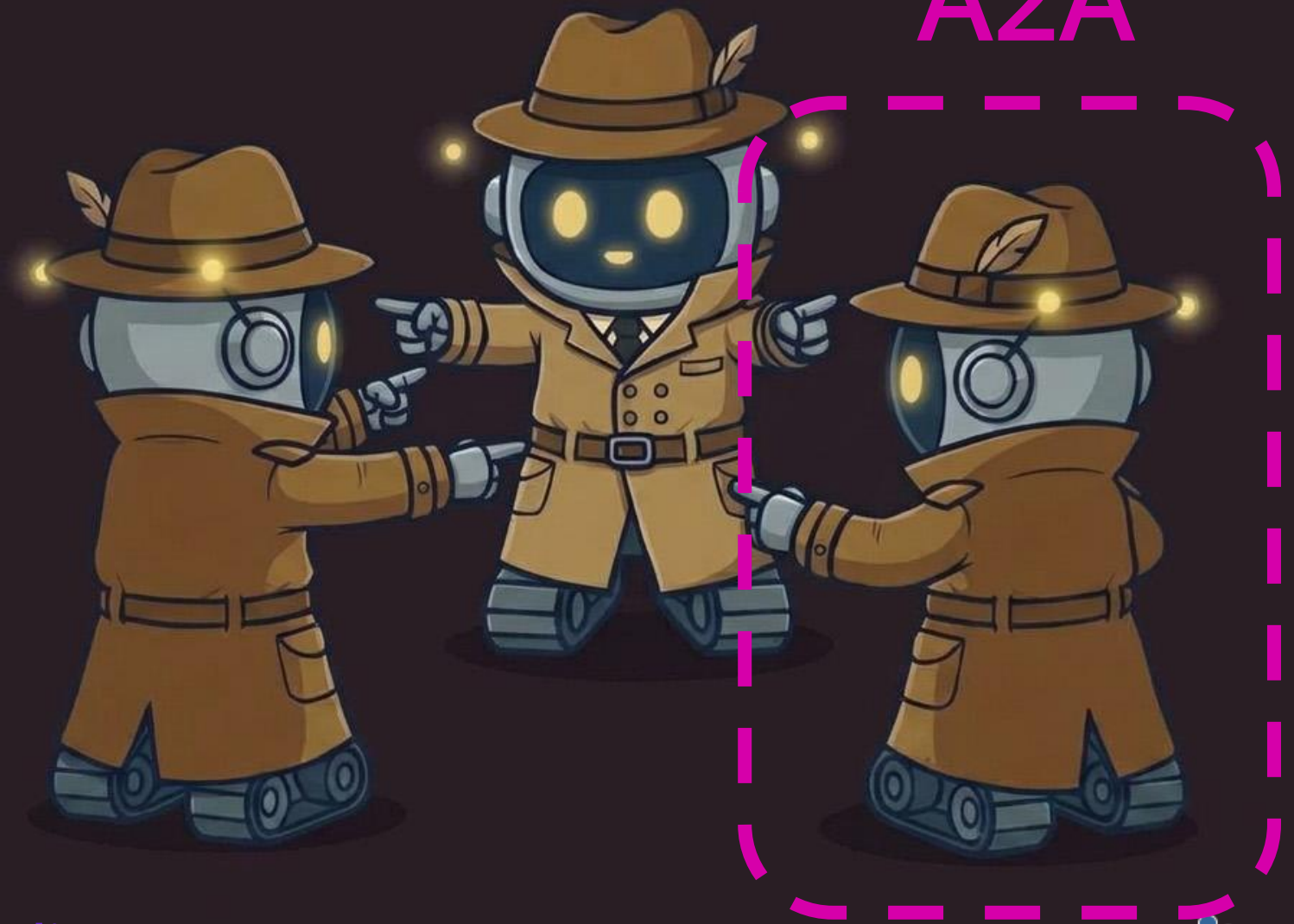
Workflow



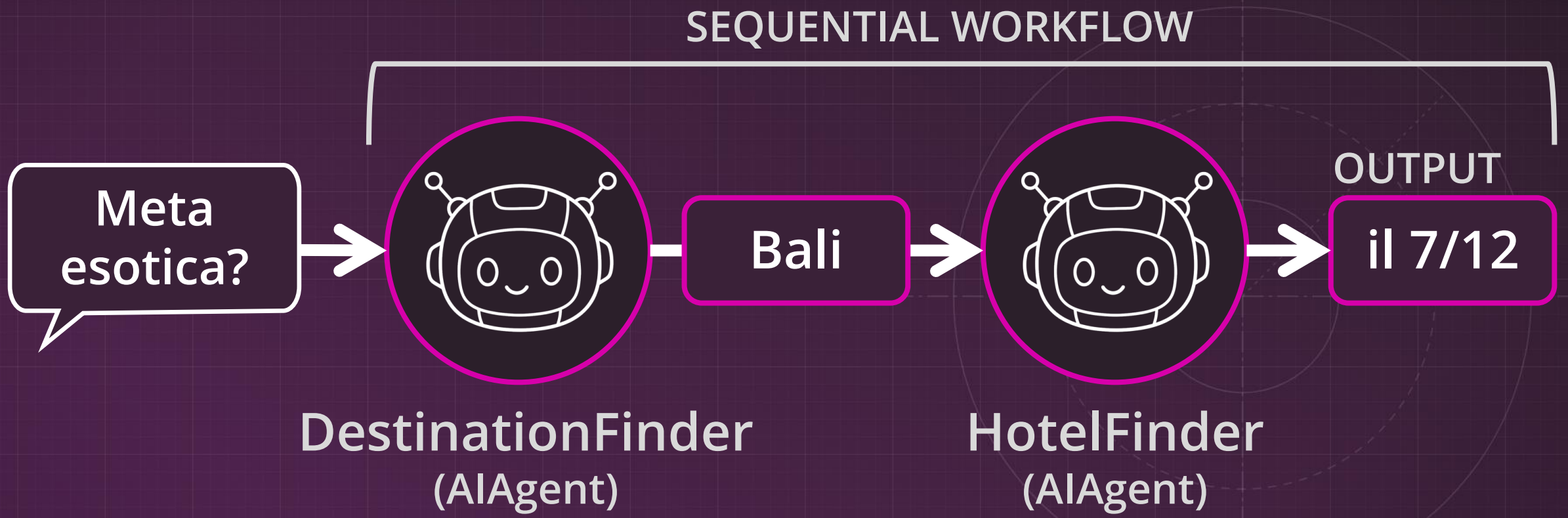
Workflow

Agente remoto
esposto col protocollo

A2A



OBIETTIVO 6: WORKFLOW, PATTERN BUILT-IN
Agenti che collaborano per risolvere problemi complessi



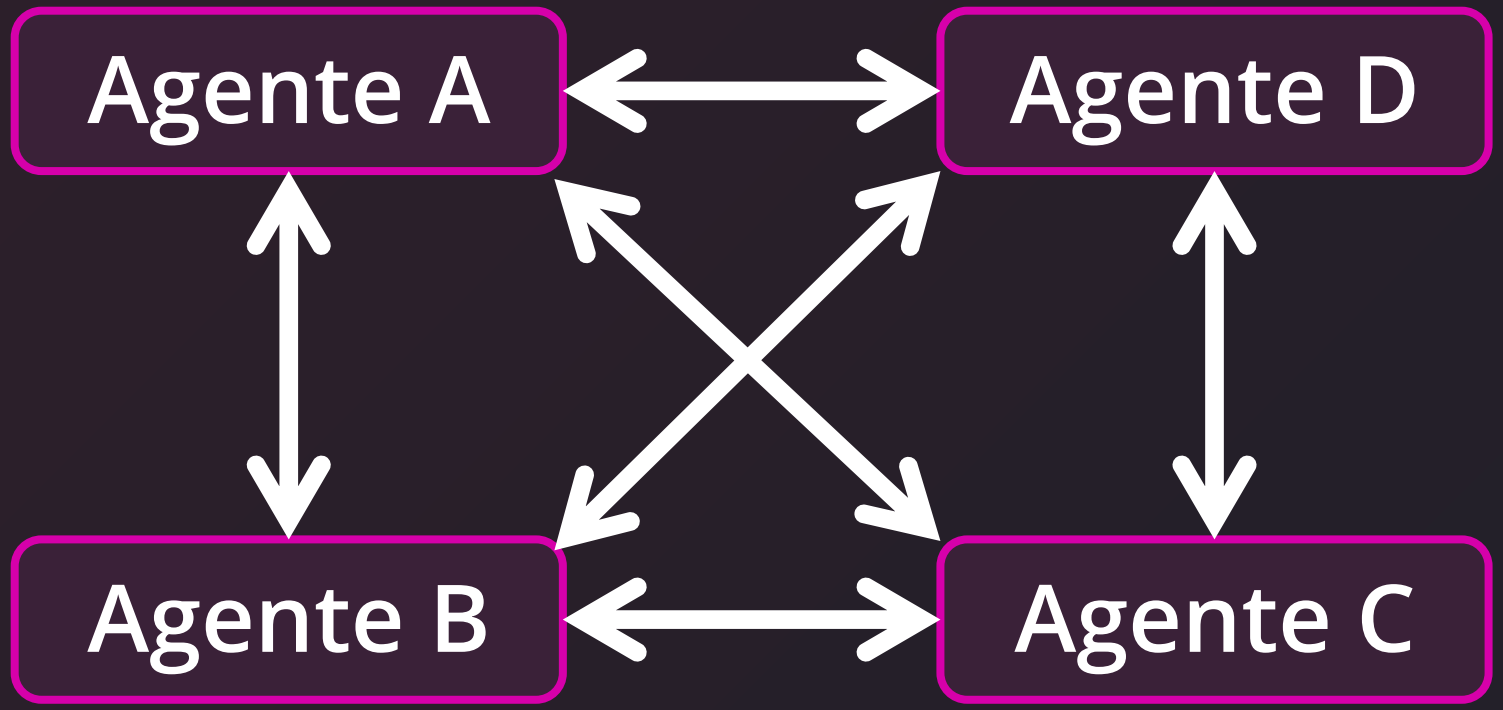
```
using Microsoft.Agents.AI;
using Microsoft.Agents.AI.Workflows;

AIAgent agent1 = chatClient.AsAIAgent(instructions: "You find destinations");
AIAgent agent2 = chatClient.AsAIAgent(instructions: "You find hotel rooms");

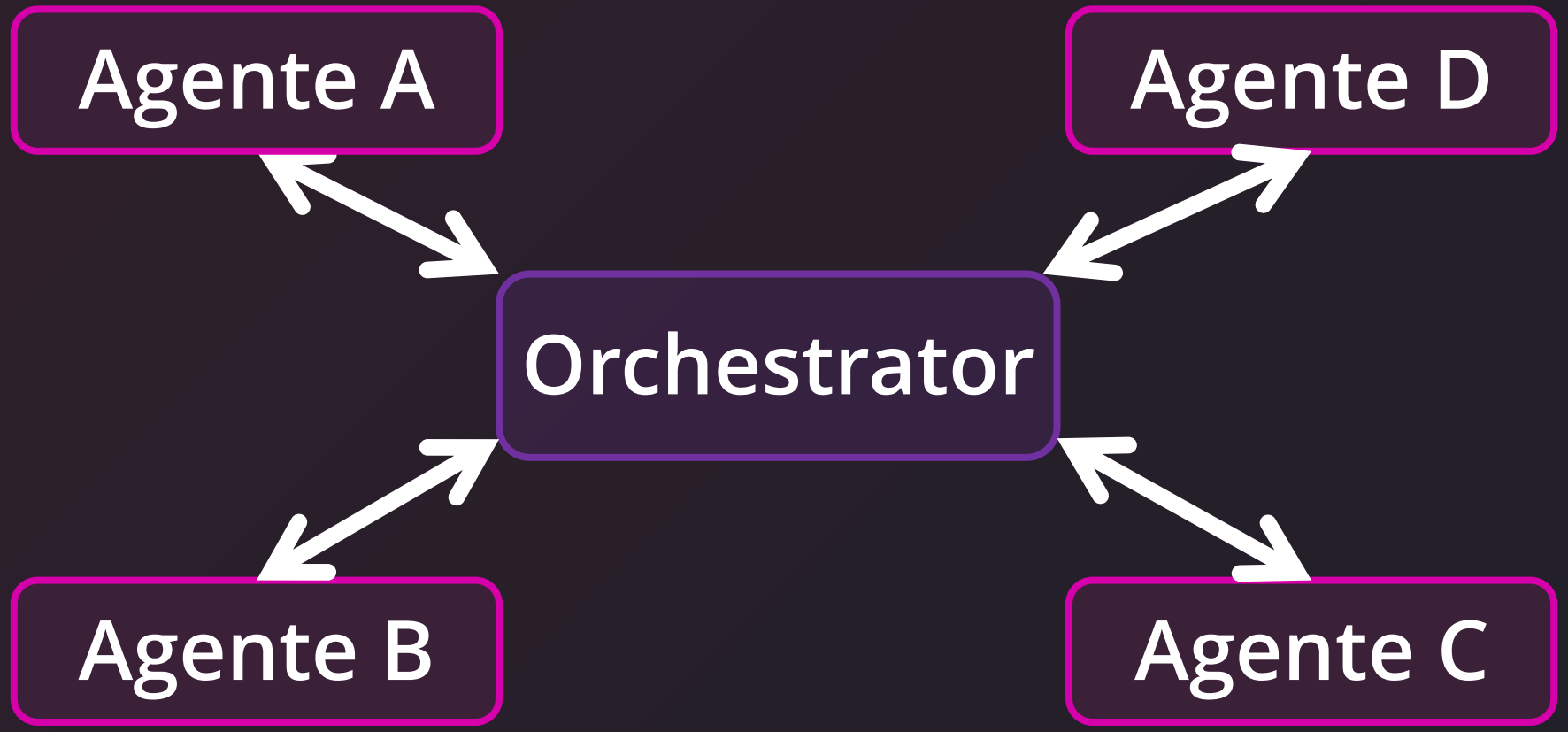
Workflow workflow = AgentWorkflowBuilder
    .CreateSequentialBuilderWith(agent1, agent2)
    .Build();

AIAgent agent = workflow.AsAIAgent();
AgentResponse response = await agent.RunAsync("Organizza un viaggio a Bali");
```

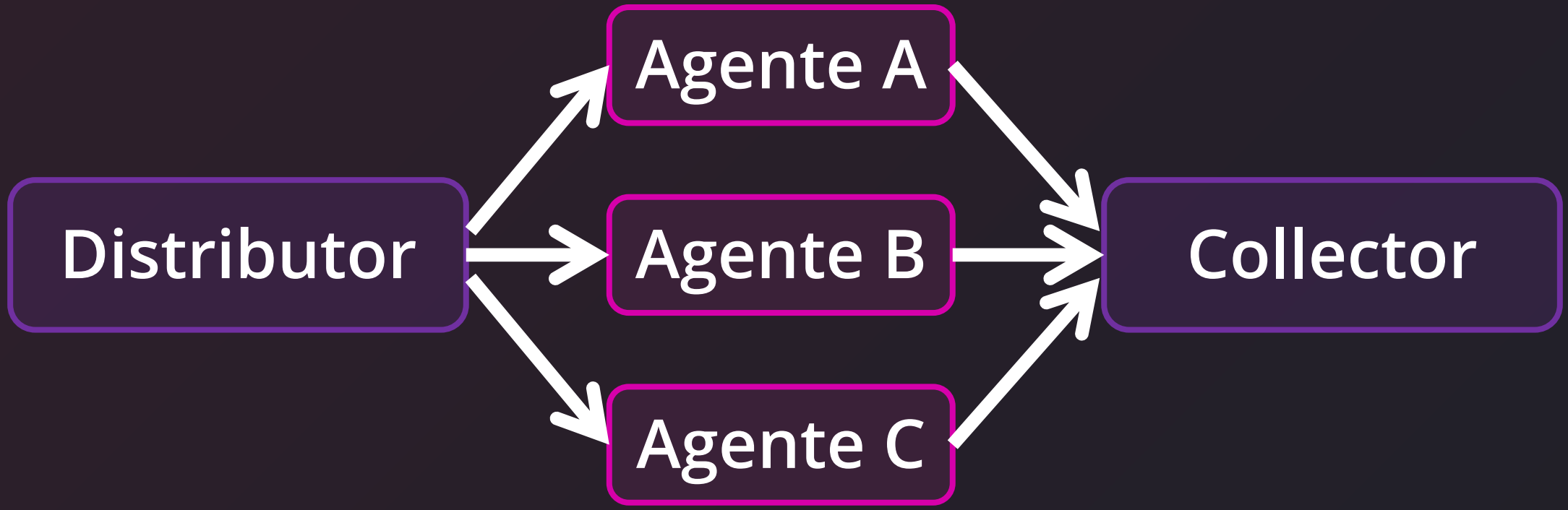
Workflow: Handoff



Workflow: Group Chat



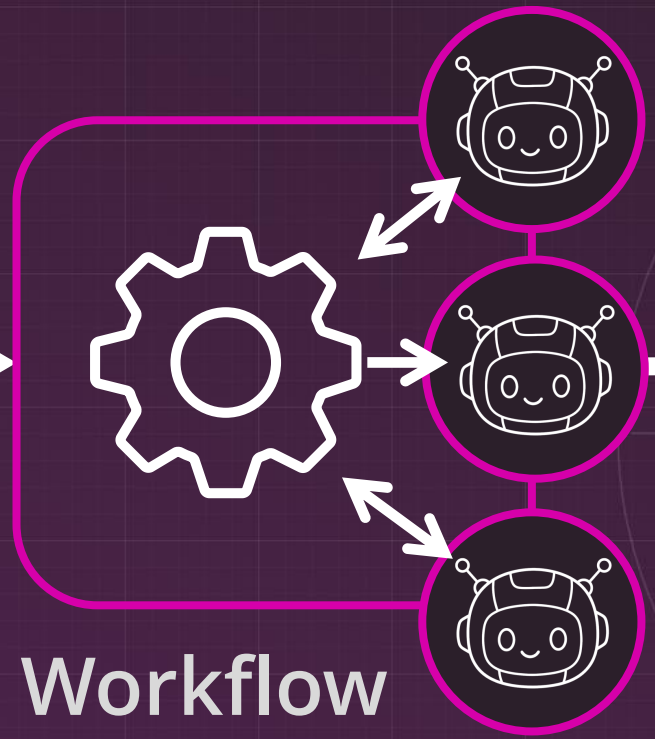
Workflow: Concurrent



OBIETTIVO 7: WORKFLOW CUSTOM
Orchestrare precisamente il flusso agentico

PROMPT

Pianifica viaggio a Tokyo



Workflow
(Multi agente)

OUTPUT

7 notti
Partenza: 8/12/26
Costo: EUR 2300

Executor



Executor



```
using Microsoft.Agents.AI;
using Microsoft.Agents.AI.Workflows;

[SendsMessage(typeof(MyResponse))]
internal partial class MyExecutor (IChatClient chatClient) : Executor ("MyExec")
{

  [MessageHandler]
  internal async ValueTask HandleAsync(
    MyRequest input, IWorkflowContext ctx, CancellationToken ct)
  {
      await ctx.SendMessageAsync(new MyResponse(), ct);
      await ctx.YieldOutputAsync(new MyResponse(), ct);
  }
}
```

```
using Microsoft.Agents.AI;
using Microsoft.Agents.AI.Workflows;

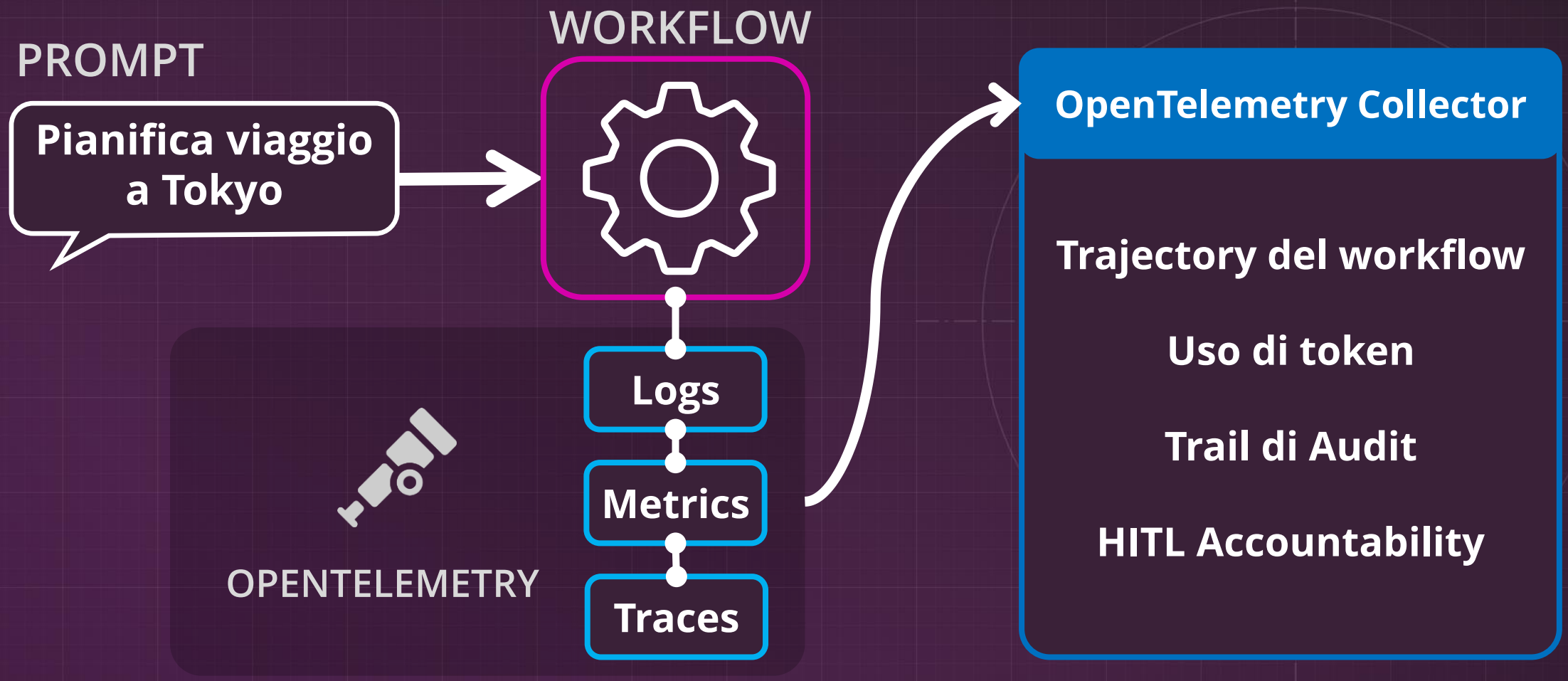
[SendsMessage(typeof(MyResponse))]
internal partial class MyExecutor (IChatClient chatClient) : Executor ("MyExec")
{

  [MessageHandler]
  internal async ValueTask HandleAsync(
    MyRequest input, IWorkflowContext ctx, CancellationToken ct)
  {
      await ctx.QueueStateUpdateAsync("userId", 123, ct);
      int userId = await ctx.ReadStateAsync<int>(new MyResponse(), ct);
  }
}
```

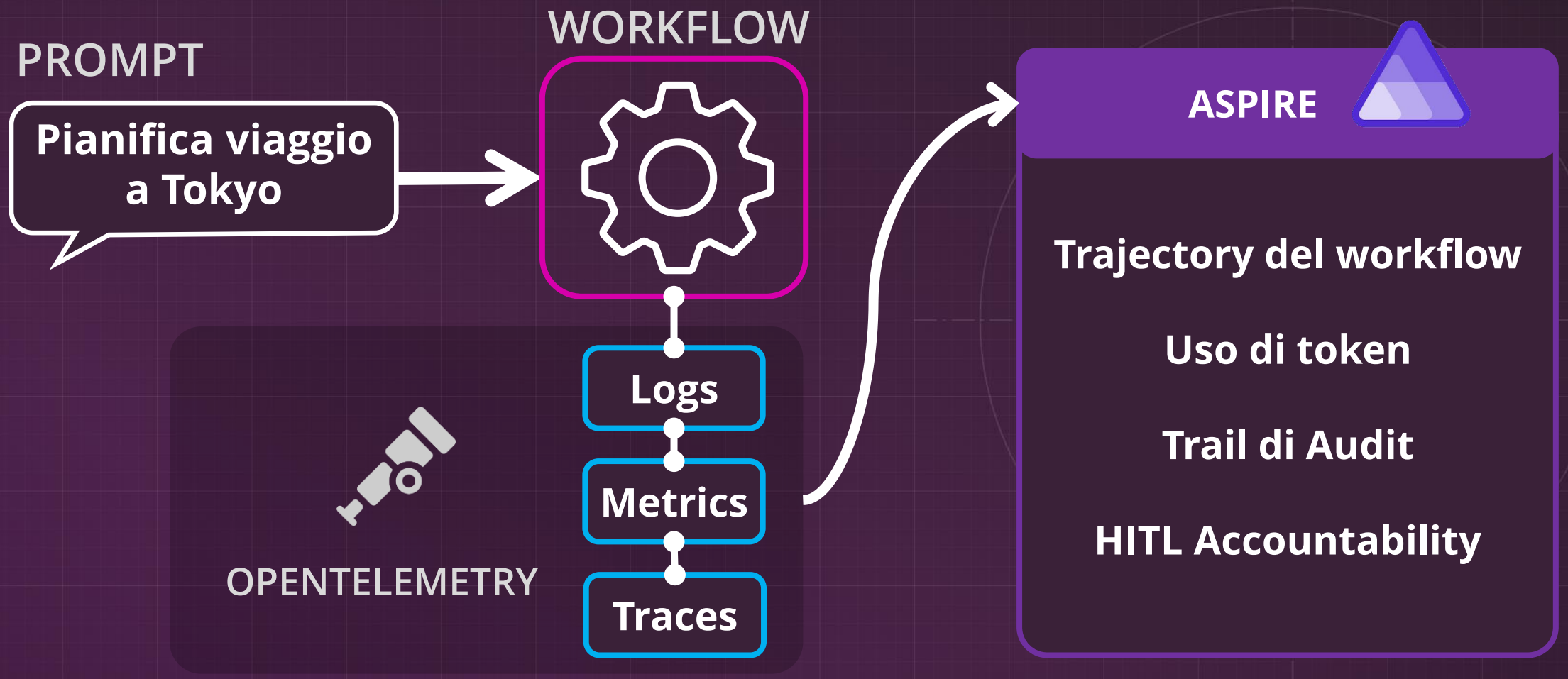
```
using Microsoft.Agents.AI;
using Microsoft.Agents.AI.Hosting;

Workflow workflow = new WorkflowBuilder(executor1)
            .AddEdge<MyRequest>(executor1, executor2, req => req.IsValid)
            .AddEdge<MyRequest>(executor1, executor3, req => !req.IsValid)
            .AddEdge<MyResponse>(executor2, executor4, null)
            .AddEdge<MyResponse>(executor3, executor4, null)
            .WithOutputFrom(executor4)
            .WithOpenTelemetry(activitySource: new ActivitySource("MyWorkflow"))
            .Build();
```

OBIETTIVO 8: OPENTELEMETRY
Osservabilità: tracce e consumo token



OBIETTIVO 8: OPENTELEMETRY
Osservabilità: tracce e consumo token



Demo

Osservabilità con OpenTelemetry e Aspire

.NET Conference
Italia 2026

OBIETTIVO 9: AG-UI E DEVUI
Dal backend alla UI moderna

PROMPT

Pianifica viaggio a Tokyo

IN SVILUPPO

DevUI

/devui

Zero codice front-end

Chat di test

IN PRODUZIONE

Protocollo AG-UI

Streaming SSE

Generative UI

Rich Cards e HITL

WORKFLOW E AGENTI



```
import { HttpAgent } from "@ag-ui/client";

const agent = new HttpAgent({url:'http://localhost:5050/aspNetSessionReactAgUi'});

agent.setMessages([{ role: "user", content: "Prenota un viaggio" }]);

const result = await agent.runAgent({}, {
  onTextMessageContentEvent,
  onTextMessageEndEvent,
  onToolCallStartEvent,
  onCustomEvent,
  onRunFinishedEvent,
  onRunFailed
});
```

Takeaways: L'AI Architecture in .NET

1. **Standardizzazione**
   Microsoft.Extensions.AI e AIAgent unificano cloud e SLM locali

2. **Disaccoppiamento**
   VectorData astrae lo storage vettoriale garantendo zero lock-in

3. **Governance**
   Checkpointing e HITL per processi transazionali sicuri

4. **Front-End**
   DevUI per il dev/test, protocollo AG-UI per la produzione


THE ATOLL
grazie

Alessio Bianchini e Moreno Gentili

Slide e materiale su
https://aspit.co/netconf-26





.NET Conference Italia 2026