

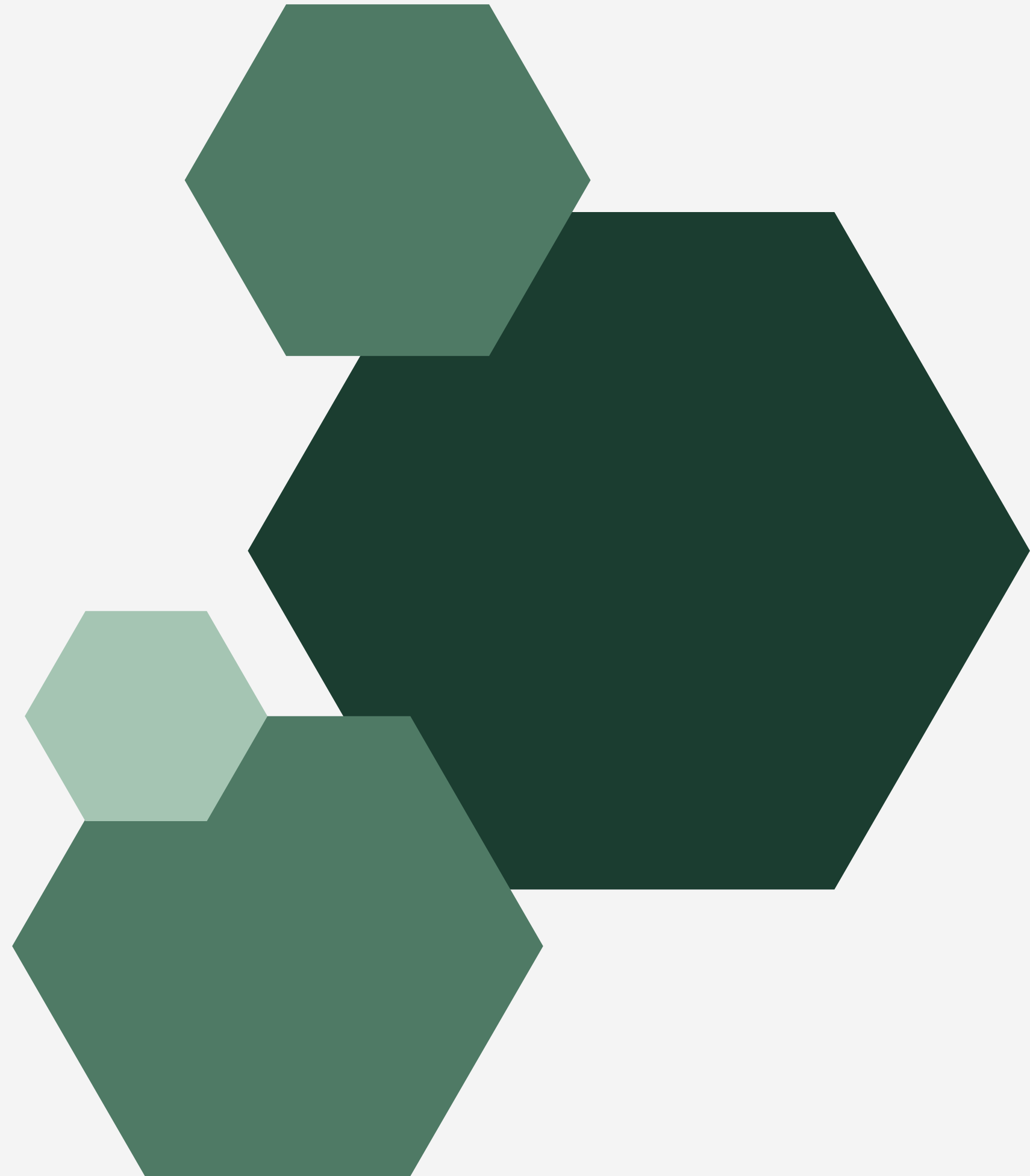


| <CodeHex16>

# Presentazione TB

13 Marzo 2025

[unipd.codehex16@gmail.com](mailto:unipd.codehex16@gmail.com)



# Tecnologie utilizzate

## Progetto

Realizzazione di una Webapp per interagire con un chatbot AI fornito da un'azienda ai propri clienti; il chatbot sfrutta come contesto i documenti e le FAQ aziendali.

⟨CodeHex16⟩

# Deploy e Containerizzazione

## Docker

Utilizzato per suddividere ed eseguire in container l'intera applicazione, rendendola facilmente distribuibile e scalabile in diversi ambienti.

Alternative considerate:

- LXC

# Frontend

## Svelte e Sveltekit

Framework reattivo e leggero, utilizzato per costruire l'interfaccia della webapp del chatbot in modo efficiente e performante.

Alternative considerate:

- Vue.js
- React
- Blazor

⟨CodeHex16⟩

# Backend e API

## Python

Linguaggio di programmazione principale, utilizzato per implementare la logica del chatbot e orchestrare l'interazione con l'AI.

## FastAPI

Framework backend in Python, scelto per la sua velocità e facilità nel gestire API REST, utilizzato per costruire la comunicazione tra le componenti software.

Alternative considerate:

- Flask

# Gestione dei dati

## MongoDB



Database NoSQL utilizzato per memorizzare documenti, cronologia delle conversazioni e utenti.

Alternative considerate:

- PostgreSQL

## ChromaDB



Database vettoriale per memorizzare e recuperare embedding dei documenti, consentendo ricerche semantiche rapide per fornire risposte contestuali.

Alternative considerate:

- Milvus

# Intelligenza Artificiale

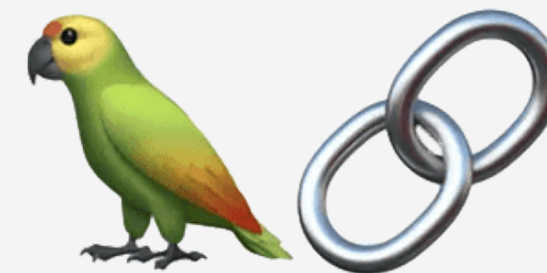
## OpenAI

Modelli di AI (in particolare 4o-mini) utilizzati per generare risposte basate sui documenti precedentemente caricati.

Alternative più rilevanti:

- Claude Haiku
- GPT o1-mini

## LangChain



Libreria che facilita l'integrazione tra modelli di AI e database, permettendo al chatbot di gestire query sui documenti e migliorare la qualità delle risposte.

⟨CodeHex16⟩

# Il Gruppo

MB

**Matteo Bazzan**  
matteo.bazzan.1@  
studenti.unipd.it

LR

**Luca Ribon**  
luca.ribon@  
studenti.unipd.it

FF

**Francesco Fragonas**  
francesco.fragonas@  
studenti.unipd.it

FS

**Filippo Sabbadin**  
filippo.sabbadin.4@  
studenti.unipd.it

GM

**Gabriele Isacco Magnelli**  
gabrieleisacco.magnelli@  
studenti.unipd.it

YZ

**Yi Hao Zhuo**  
yihao.zhuo@  
studenti.unipd.it

LR

**Luca Rossi**  
luca.rossi.24@  
studenti.unipd.it

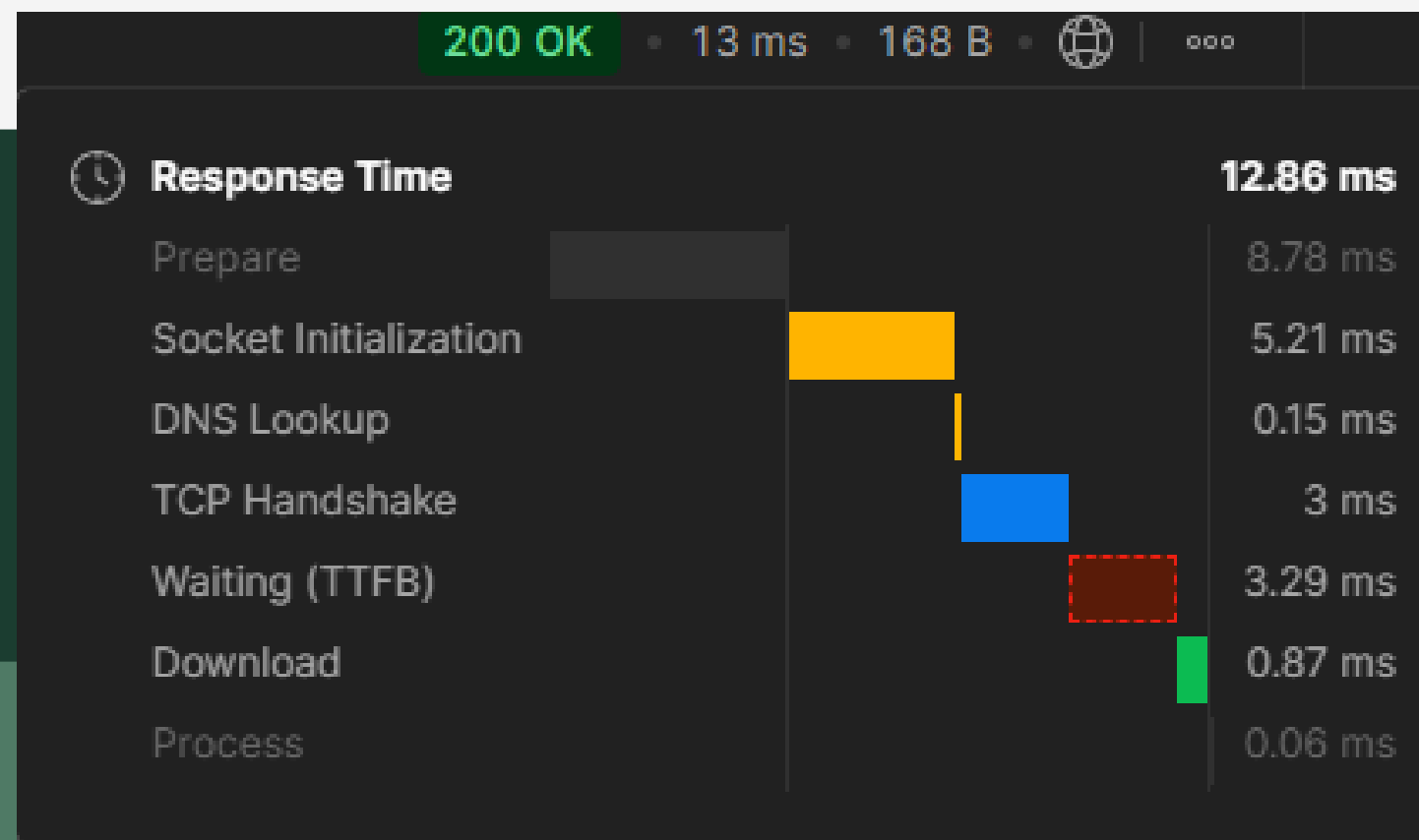
⟨CodeHex16⟩



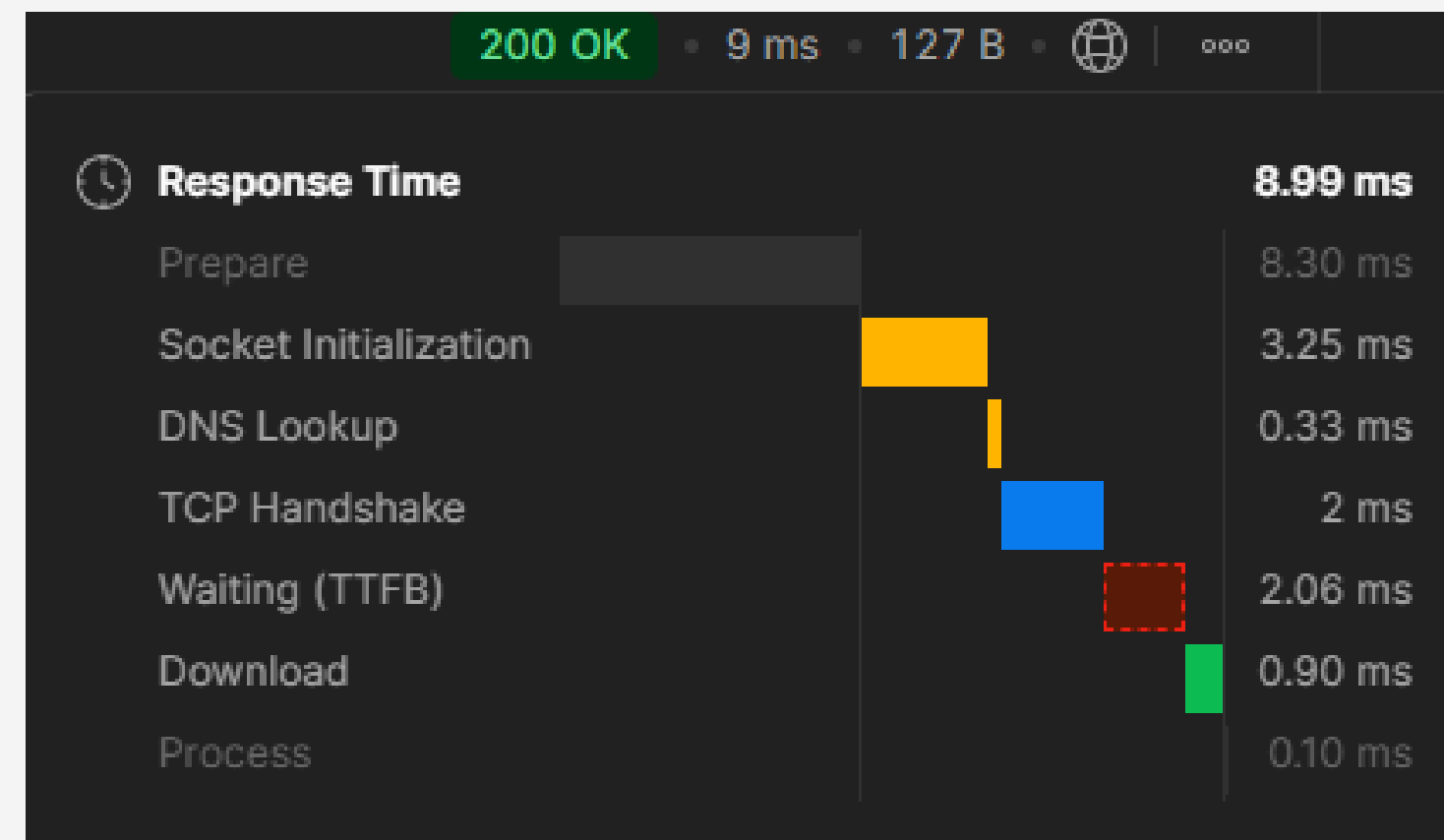
# ← Perché FastAPI?

## Confronto richiesta API base

Flask



FastAPI



⟨CodeHex16⟩

# ← Perché MongoDB?

```
{
  "_id": "67c220151f890f23980a91d0",
  "name": "Chat smartphone",
  "user_email": "mario.rossi@gmail.com",
  "created_at": "2025-03-01T10:15:22.123Z",
  "messages": [
    {
      "sender": "bot",
      "content": "Ciao, sono SupplAI, il tuo assis",
      "timestamp": "2025-03-01T10:15:22.123Z"
    },
    {
      "sender": "mario.rossi@gmail.com",
      "content": "Cerco un nuovo smartphone",
      "timestamp": "2025-03-01T10:15:45.556Z"
    },
    {
      "sender": "bot",
      "content": "Posso aiutarti a trovare lo smar",
      "timestamp": "2025-03-01T10:16:02.789Z"
    }
  ]
},
```

```
Database-API/Mongo_Postgres_PerformanceComparison on P main [!] via v3.13.2 (env) took 7s
> python3 main.py
MongoDB insert time: 0.01619272232055664 sec for 71 rows
PostgreSQL insert time: 0.048960120677948 sec for 71 rows

MongoDB fetchall time: 0.0013749837875366212 sec for 71 rows
PostgreSQL fetchall time: 0.0011650419235229492 sec for 71 rows

MongoDB fetchspecific time in json column: 0.0015075087547302246 sec for 71 rows
PostgreSQL fetchspecific time in json column: 0.0016937017440795898 sec for 71 rows

MongoDB fetchspecific time no json column: 0.0010768795013427735 sec for 8 rows
PostgreSQL fetchspecific time no json column: 0.0005383014678955079 sec for 8 rows

TIME DIFFERENCES IN PERCENTAGE
Mongodb insert time difference: -202.35879865483267 %
Mongodb fetchall time difference: 15.26867923219643 %
Mongodb fetchspecific time difference in json column: -12.351038676567107 %
Mongodb fetchspecific time difference no json column: 50.01284106306291 %
```

Dimensioni tabelle:

- MongoDB: 32.8 KB
- PostgreSQL: 48 KB

⟨CodeHex16⟩

# ← Perché ChromaDB?

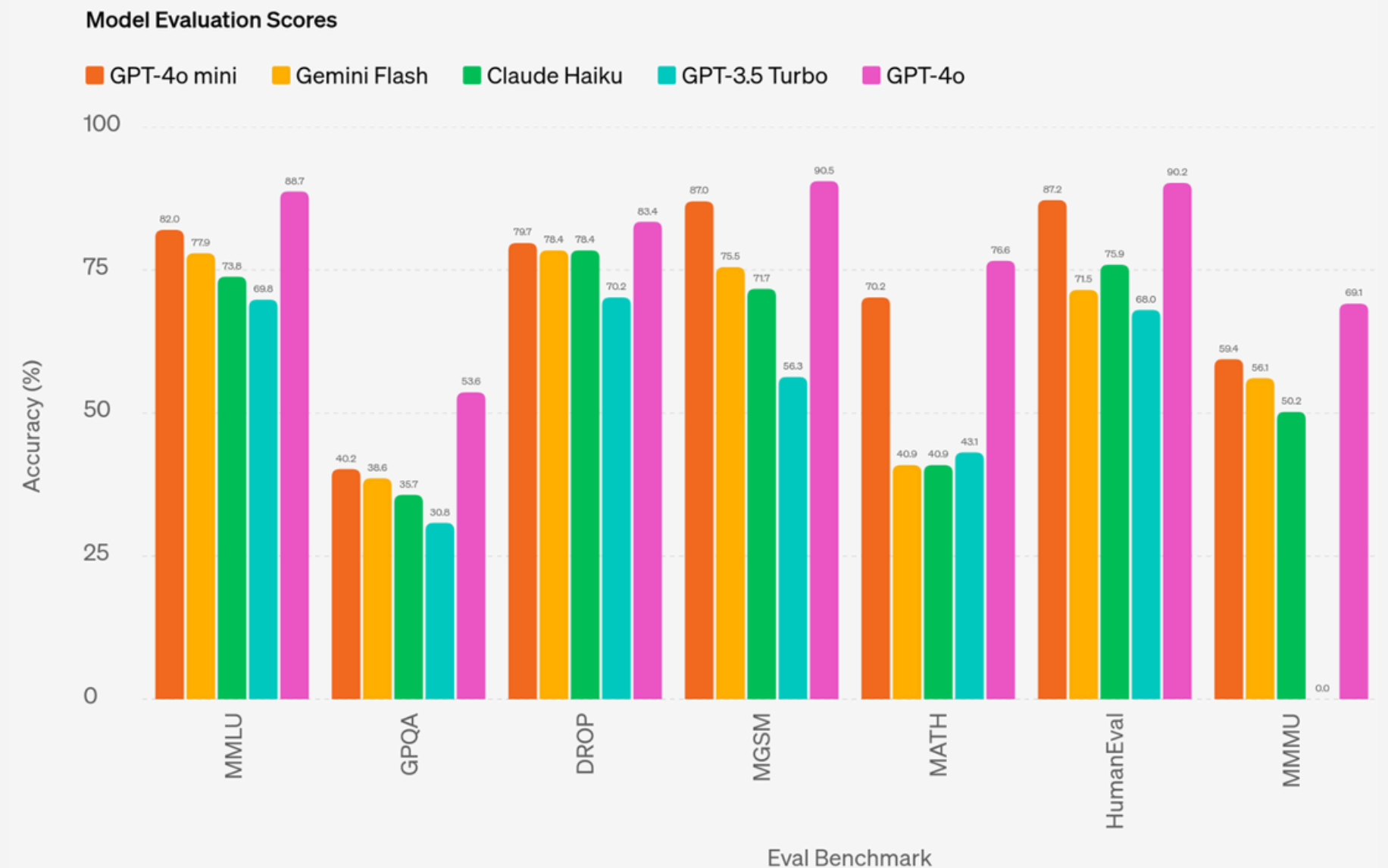
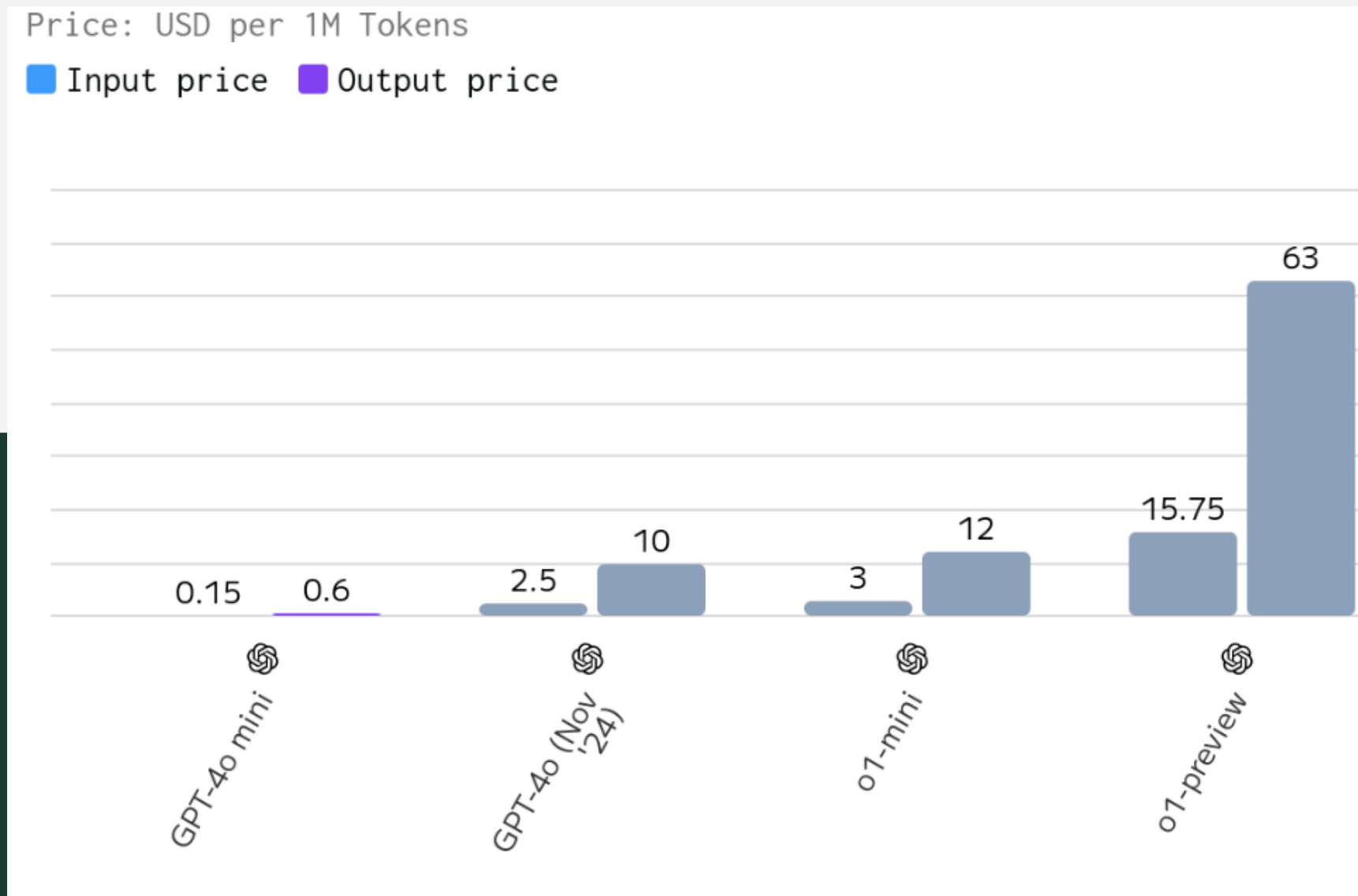
Vettorializzazione dei documenti del dataset	
Database	Tempo di risposta
Milvus	7.49 s
ChromaDB	7.30 s

⟨CodeHex16⟩

Ottenimento risposta domanda 1	
Database	Tempo di risposta
Milvus	8.54 s
ChromaDB	6.98 s
Ottenimento risposta domanda 2	
Database	Tempo di risposta
Milvus	2.56 s
ChromaDB	2.08 s

Revisione RTB - Confronto ChromaDB vs Milvus

# ← Perché GPT 4o-mini?



⟨CodeHex16⟩

Fonte: GPT 4o-mini advancing cost efficient intelligence

Revisione RTB - Confronto modelli LLM