MULTI-AGENT ARCHITECTURE

Di Maio, Montesi, Traversa

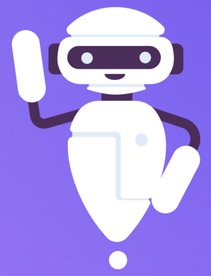
HOW WE CREATED OUR CREW

We used Crew AI

We found Crew AI to be the most intutitive when building:

- Agent definition
- Tools
- Workflow structure
- Minimal coding, Maxiaml prompting

OUR AGENTS



ANALYST

- Analyzes aggregated Italian public administration personnel data from the NoiPA portal, using an analysis tool to process queries and extract insights
- Produce both human-readable summaries and machine-readable CSV output specifically formatted for visualization.

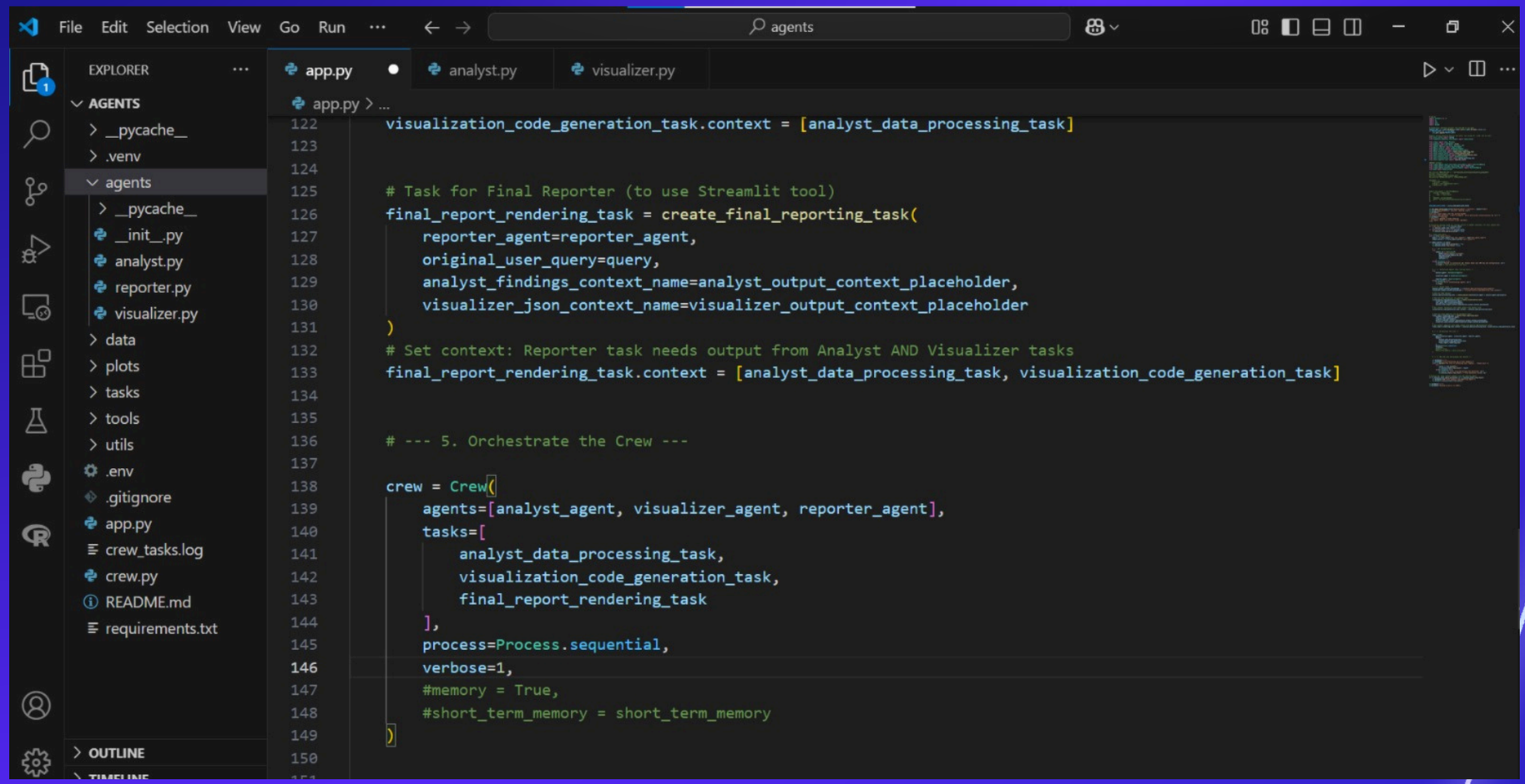
VISUALIZER

- Transforms the Analyst's CSV data into visualizations by generating the most appropriate chart type using Python
- Outputs a structured JSON containing visualization parameters, plot descriptions, and the file path to the saved image, adapting visualization approaches when necessary.

REPORTER

- Orchestrates the presentation layer by taking the Analyst's findings and Visualizer's JSON output and using a specialized Streamlit tool to render both text and visualization in the application.

REPOSITORY



- Agents (analyst, visualizer, reporter)
- Datasets
- Tasks
- Tools
- App.py

METRICS

We used a scale from 1-5

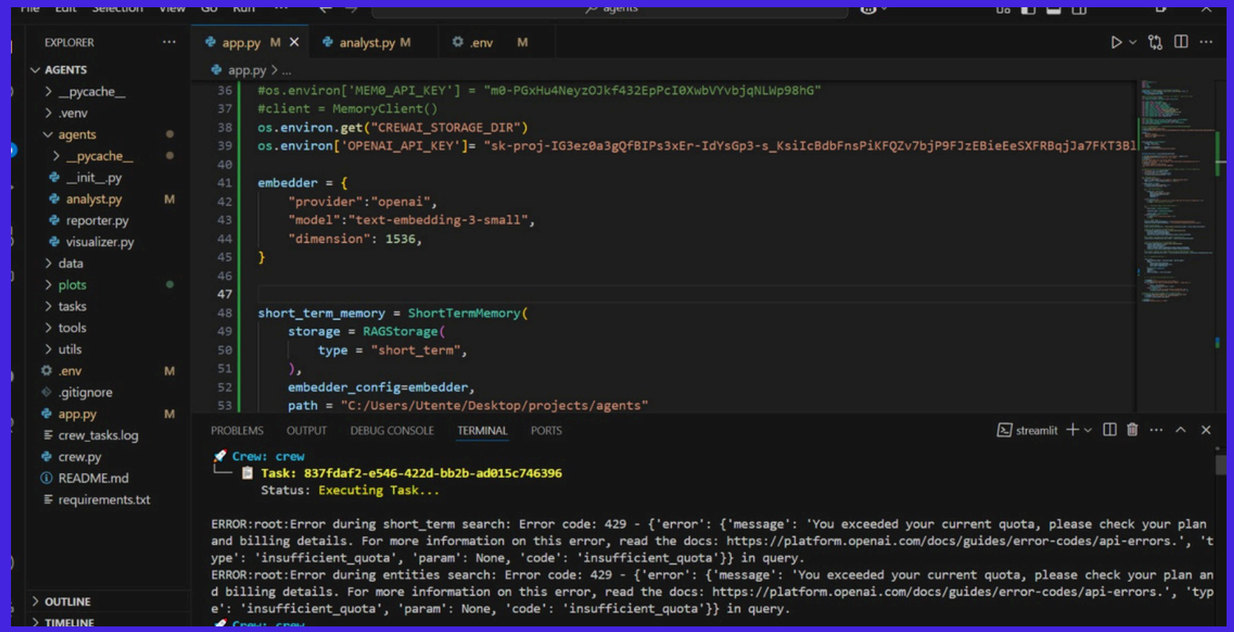
- **AVERAGE ACCURACY SCORE: 2,90**
- **AVERAGE SCORE ITALIAN QUERIES : 2,96**
- **AVERAGE SCORE ENGLISH QUERIES: 2,85**
- **AVERAGE TIME: 38 secs**

MEMORY

WHAT WE DISCOVERED:

- **No memory buffer supported**
- **Default configuration for storing memory in embeddings**
- **Memory intutive to construcut but only works with a paid text embedder model**

THE CODE



MAIN TAKE AWAYS

01

- A sequential architecture worked better than a hierarchical architecture for this task

02

- Queries in italian performed slightly better, this may be cause by the nature of the data

THANK YOU!