

## Relatório 11 - Prática: Criando agentes com CrewAI e Streamlit (IV)

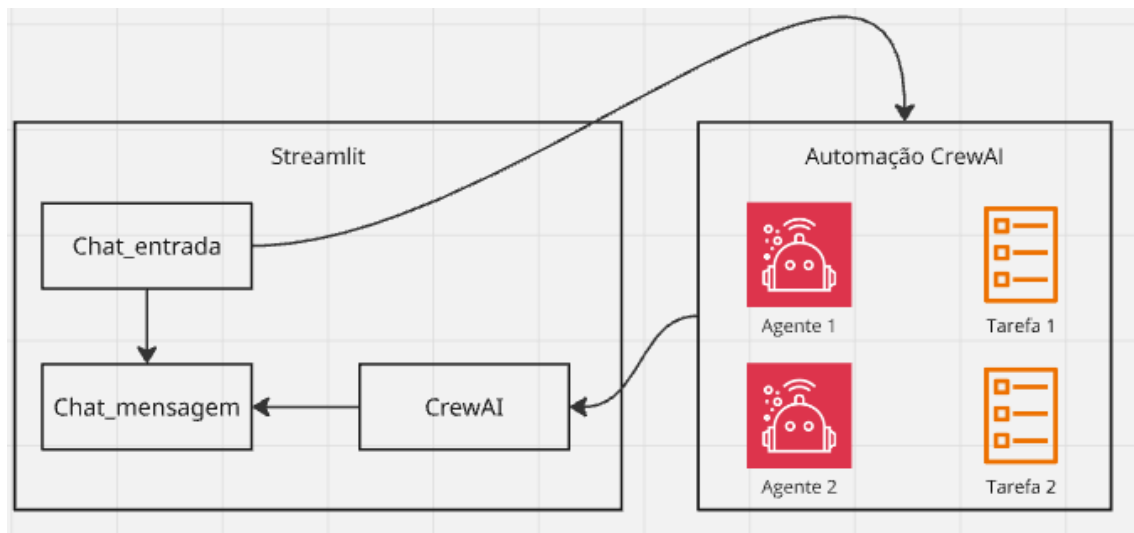
<Paulo Victor Dos Santos>

### Descrição da atividade

A atividade do card 11 é uma atividade prática voltada a usar agentes com a ferramenta streamlit.

O streamlit facilita na criação de aplicativos web através da linguagem python, é possível criar protótipos web programando em python, sem a necessidade de técnicas avançadas de frontend.

O primeiro vídeo demonstra de forma simples como utilizar o streamlit com agentes inteligentes, trata-se de uma aula prática com um caso de uso chamado "CrewAI Writing Studio". Neste caso de uso são utilizados dois agentes, um é escrito de texto para blog e o outro agente é o revisor do texto escrito.



O fluxo acima exibe a interação do aplicativo. A parte de automação consta com dois agentes e a resposta final é exibida de forma sequencial na interface criada pelo streamlit.

O segundo vídeo do card 11, exibe um aplicativo de planejamento de férias, chamado VacAgent. Neste aplicativo o usuário seleciona a cidade em que se encontra, a cidade de destino no qual tem interesse em viajar, as datas de interesse e um campo textual no qual pode inserir mais detalhes sobre as supostas férias.

Trata-se de um código que utiliza agentes de forma hierárquico, isso facilita a definição do que os agentes precisam de fato executar, trazendo velocidade nas respostas. Ou seja, o usuário pré selecionar alguns dados cruciais, pode fazer com que a aplicação seja mais rápida, uma vez que os agentes já possuem bastante informação, facilitando na tomada de decisão.

O último vídeo é uma solução interessante e bastante útil. Esta solução tem como escopo a pesquisa por profissionais a partir da descrição de uma vaga ou da capacitação do suporte profissional da área de tecnologia no qual deseja contratar.

O sistema pode fazer buscas no linkedin ou na internet para procurar o profissional mais adequado.

O último vídeo do card 11 apresenta conceitos de utilização de containers com agentes. Além da introdução aos containers, a aula demonstra como utilizar a biblioteca FASTAPI, o que facilita a integração e interoperabilidade com outros sistemas através de REST.

A prática aplicada no card 11 toma como base o sistema inicial apresentado que foi personalizado para atuar como um sistema de diagnóstico e laudos médicos.

A ideia é que o usuário possa descrever alguns sintomas e o sistema possa dar possíveis indicações de diagnóstico ou laudos, como se trata de um experimento a ideia é observar o comportamento dos agentes e suas interações na área médica.

localhost:8501

EGREDO DAS CO... Deploying the Huggin... amazon-sagemaker-e... Untitled Deploying the Huggin... Building GenAI Chatb... Introducing the Huggi... Deploy FLAI



## Diagnóstico e Laudos médicos

 Quais os sintomas você está sentindo?

 Sinto dor de cabeça após comer doce, o que pode ser?



### Aqui está o possível diagnóstico

Conforme revisado cuidadosamente o diagnóstico fornecido, sugiro algumas melhorias para facilitar a interpretação pelos usuários:

1. Especificar claramente os tipos de dores de cabeça mencionados (migraine e tension-type headaches) logo no início do diagnóstico para uma compreensão imediata do problema principal.
2. Detalhar os sintomas associados a cada tipo de dor de cabeça para diferenciar melhor entre eles e auxiliar no diagnóstico correto.
3. Incluir informações sobre os possíveis gatilhos alimentares que desencadeiam as dores de cabeça, como alimentos açucarados e certos alimentos, para orientar o paciente sobre possíveis mudanças na dieta.
4. Recomendar uma avaliação mais detalhada para confirmar o diagnóstico e desenvolver um plano de tratamento personalizado para o paciente, levando em consideração os gatilhos identificados.

Essas alterações ajudarão a tornar o diagnóstico mais claro, preciso e informativo, facilitando a interpretação e o desenvolvimento de um plano de tratamento eficaz para o paciente.



O intuito do teste realizado é a preparação para o trabalho final do Fastcamp, com a experiência adquirida com sistemas distribuídos, containers, FastApi e Streamlit, eu possa utilizar todas essas tecnologias para criar uma ferramenta útil com toda abrangência possível.

## Dificuldades

O código do segundo vídeo está disponível em um repositório na descrição do vídeo. Fiz o clone do repositório e segui as recomendações do arquivo README, conforme a imagem abaixo.

## CrewAI Framework

CrewAI simplifies the orchestration of role-playing AI agents. In VacAgent, these agents collaboratively decide on cities and craft a complete itinerary for your trip based on specified preferences, all accessible through a streamlined Streamlit user interface.

## Streamlit Interface

The introduction of [Streamlit](#) transforms this application into an interactive web app, allowing users to easily input their preferences and receive tailored travel plans.

## Running the Application

To experience the VacAgent app:

- **Configure Environment:** Set up the environment variables for [Browseless](#), [Serper](#), and [OpenAI](#). Use the `secrets.example` as a guide to add your keys then move that file ( `secrets.toml` ) to `.streamlit/secrets.toml`.
- **Install Dependencies:** Execute `pip install -r requirements.txt` in your terminal.
- **Launch the App:** Run `streamlit run streamlit_app.py` to start the Streamlit interface.

★ **Disclaimer:** The application uses GPT-4 by default. Ensure you have access to OpenAI's API and be aware of the associated costs.

## Details & Explanation

- **Streamlit UI:** The Streamlit interface is implemented in `streamlit_app.py`, where users can input their trip details.
- **Components:**
  - `./trip_tasks.py`: Contains task prompts for the agents.
  - `./trip_agents.py`: Manages the creation of agents.
  - `./tools directory`: Houses tool classes used by agents.
  - `./streamlit_app.py`: The heart of the Streamlit app.

## Using GPT 3.5

To switch from GPT-4 to GPT-3.5, pass the `llm` argument in the agent constructor.

As orientações sugerem a instalação das bibliotecas através do arquivo de requirements, a edição do arquivo de chaves e finalmente a execução padrão do streamlit.

```
line 1)) (3.5.4)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in d:\users\paulo\pycharmprojects\trip_planner_agent\venv\lib\site-packages (0.6.1)

[notice] A new release of pip is available: 23.2.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS D:\Users\paulo\PycharmProjects\trip_planner_agent> streamlit run .\streamlit_app.py
```

Porém mesmo seguindo as orientações e ainda trocando as versões do python o sistema quebra e não executa, muito provavelmente por incompatibilidade de alguma biblioteca, dado o tempo que o código foi publicado.

**ImportError: cannot import name 'Self' from 'typing' (C:\Python310\lib\typing.py)**

Traceback:

```
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
result = func()
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
exec(code, module.__dict__)
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\streamlit_app.py", lin
from crewai import Crew
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from crewai.agent import Agent
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from crewai.agents.crew_agent_executor import CrewAgentExecutor
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from crewai.agents.agent_builder.base_agent_executor_mixin import CrewAgen
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from crewai.memory.entity.entity_memory_item import EntityMemoryItem
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from .entity.entity_memory import EntityMemory
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from crewai.memory.memory import Memory
File "D:\Users\paulo\PycharmProjects\trip_planner_agent\venv\lib\site-packages
from typing import Any, Dict, List, Optional, Self
```

[Ask Google](#) [Ask ChatGPT](#)

Por se tratar de uma solução mais organizada e mais complexa, não achei viável consertar o código ou buscar uma solução. Preferi focar nos outros entregáveis, dado o prazo das atividades.

## Conclusões

Nota-se que através da utilização do streamlit é possível criar provas de conceito elaboradas para utilização de agentes. No último vídeo fica claro como é fácil e eficiente criar container com agentes, isso facilita a criação e utilização de agentes em arquiteturas modernas e distribuídas.

Além de facilitar na disponibilidade, a utilização de containers facilita a escalabilidade e a disponibilidade de soluções mais rebuscadas.

## Referencias

<https://www.youtube.com/watch?v=I-n9iTb6nis>

[https://www.youtube.com/watch?v=nKG\\_kbQUUDE](https://www.youtube.com/watch?v=nKG_kbQUUDE)

<https://www.youtube.com/watch?v=I-n9iTb6nis>

[https://github.com/yeyu2/Youtube\\_demos](https://github.com/yeyu2/Youtube_demos)

