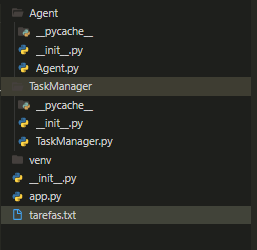
#### **1. Introdução**

Este documento descreve a estrutura e o funcionamento de um sistema de gestão de tarefas domésticas implementado em Python. O sistema consiste em um **gerenciador de tarefas** (TaskManager) e **agentes** (Agent) responsáveis por executar as tarefas. As tarefas são definidas em um arquivo externo .txt, e o sistema atribui as tarefas aos agentes com base nas condições e prioridades definidas.

#### **2. Estrutura do Código**

O sistema está organizado em três arquivos principais:

* app.py: O ponto de entrada do programa que inicializa o sistema e executa a lógica principal. localizado no root do projeto.
* TaskManager.py: Define a lógica de gerenciamento de tarefas, verificando condições e priorizando tarefas, localizado no diretório TaskManager
* Agent.py: Define os agentes responsáveis por realizar cada tarefa, localizado no diretório Agent



#### **3. Descrição dos Arquivos**

##### **3.1 app.py**

Este é o arquivo principal que executa o sistema.

**Função load\_tasks\_from\_file(file\_path)**

* Lê um arquivo de texto contendo as tarefas, suas condições e prioridades.
* O formato de cada linha no arquivo de tarefas (tarefas.txt) deve ser: nome\_tarefa=condição,prioridade. Por exemplo: wash\_dishes=True,1.
* A função retorna dois dicionários:
  + tasks: Um dicionário com o nome da tarefa como chave e um valor booleano que indica se a tarefa está disponível (True ou False).
  + priorities: Um dicionário com o nome da tarefa como chave e um valor inteiro que indica sua prioridade (quanto menor o número, maior a prioridade).

**Inicialização do TaskManager**

* O TaskManager é inicializado com as tarefas e suas prioridades lidas do arquivo.

**Criação dos Agentes**

* Um conjunto de agentes é criado. Cada agente é responsável por uma tarefa específica, como "lavar pratos" ou "varrer o chão". Os agentes são instâncias da classe Agent, e cada um possui um nome e uma tarefa associada.

**Execução da Simulação**

* O sistema entra em um loop onde o TaskManager atribui as tarefas aos agentes.
* A cada iteração, o TaskManager verifica qual tarefa está disponível e a atribui a um agente que possa realizá-la.
* Se não houver mais tarefas disponíveis, o loop é encerrado.

##### **3.2 TaskManager.py**

Este arquivo contém a classe TaskManager, que gerencia as tarefas e as atribui aos agentes.

**Atributos**

* tasks: Um dicionário que mapeia cada tarefa a uma condição booleana (disponível ou não).
* task\_priority: Um dicionário que associa cada tarefa a uma prioridade numérica.

**Métodos**

1. check\_conditions(): Verifica todas as tarefas disponíveis e retorna a primeira tarefa com condição True, respeitando a ordem de prioridade. A tarefa de maior prioridade (número mais baixo) é escolhida primeiro.
2. assign\_task(): Usa o método check\_conditions() para atribuir uma tarefa. Se uma tarefa estiver disponível, ela é retornada e exibida na tela. Caso contrário, uma mensagem indicando que não há tarefas disponíveis é exibida.
3. update\_conditions(task): Marca uma tarefa como concluída, alterando sua condição para False no dicionário de tarefas.

##### **3.3 Agent.py**

Este arquivo contém a classe Agent, que representa os agentes responsáveis por realizar as tarefas.

**Atributos**

* name: Nome do agente (por exemplo, "Agente 1").
* task: A tarefa que o agente está designado para realizar (por exemplo, "wash\_dishes").

**Métodos**

1. perform\_task(): Imprime na tela o nome do agente e a tarefa que ele está realizando.

#### **4. Exemplo de Arquivo de Tarefas (tarefas.txt)**

Aqui está um exemplo do conteúdo do arquivo tarefas.txt que define as condições e prioridades das tarefas:

wash\_dishes=True,1

sweep\_floor=False,2

water\_plants=True,3

take\_out\_trash=True,1

clean\_windows=False,4

Neste exemplo:

* As tarefas wash\_dishes, water\_plants e take\_out\_trash estão disponíveis (True).
* As tarefas sweep\_floor e clean\_windows não estão disponíveis (False).
* As prioridades indicam que as tarefas wash\_dishes e take\_out\_trash têm maior prioridade (1), seguidas por sweep\_floor (2) e water\_plants (3).

#### **5. Fluxo de Execução**

1. O arquivo tarefas.txt é carregado, e suas tarefas e prioridades são armazenadas.
2. O TaskManager verifica quais tarefas estão disponíveis e qual delas tem maior prioridade.
3. O sistema procura um agente que seja capaz de realizar a tarefa atribuída.
4. O agente executa a tarefa, e o estado do sistema é atualizado (a tarefa é marcada como concluída).
5. O processo continua até que todas as tarefas disponíveis sejam concluídas.

#### **6. Considerações Finais**

Este sistema simples de gerenciamento de tarefas domésticas pode ser facilmente adaptado para incluir mais tarefas ou agentes, além de condições mais complexas para a execução de cada tarefa.