Build a RESTful API that manages Trase Agents running tasks in production.

At Trase, an Agent is a runnable capability—code, tools, and prompts—designed to complete a class of jobs safely and reliably. A Task is a concrete job instance describing what to do, who can do it, and its lifecycle state. The Agents & Tasks API manages both and coordinates which agents can run which tasks.

- Using any programming language and framework of your choice, create a simple RESTful API that exposes two endpoints: /agents and /tasks.
- The /agents endpoint should support the following operations:
    - GET /agents: Return a list of all agents in JSON format, with each user having an id, a name, and a description attribute.
    - POST /agents: Create a new agent with the given name and description in the request body, and return the created agent in JSON format, with a unique id assigned by the server.
    - GET /agents/{id}: Return the agent with the given id in JSON format, or a 404 error if the agent does not exist.
    - PUT /agents/{id}: Update the agent with the given id with the new name and description in the request body, and return the updated agent in JSON format, or a 404 error if the agent does not exist.
    - DELETE /agents/{id}: Delete the agent with the given id, and return a 204 status code, or a 404 error if the agent does not exist.
- The /tasks endpoint should support the following operations:
    - GET /task: Return a list of all tasks in JSON format, with each post having an id, a title, a description, and a supported_agent_id attribute, which references the id of the agent who can complete the task. **Bonus:** Make it so tasks can support multiple agents running them.
    - POST /tasks: Create a new task with the given title, description, and supported_agent_id in the request body, and return the created task in JSON format, with a unique id assigned by the server. If the supported_agent_id does not match any existing agent, return a 400 error.
    - GET /tasks/{id}: Return the task with the given id in JSON format, or a 404 error if the task does not exist.
    - PUT /tasks/{id}: Update the task with the given id with the new title, description, and supported_agent_id in the request body, and return the updated task in JSON format, or a 404 error if the task does not exist. If the supported_agent_id does not match any existing agent, return a 400 error.

    - DELETE /tasks/{id}: Delete the task with the given id, and return a 204 status code, or a 404 error if the task does not exist.

- You can use any data storage method of your choice (in-memory, a file system, or a relational or non-relational database)
- You should provide clear and concise documentation for your API, including the expected request and response formats, the possible error codes and messages, and any assumptions or limitations you made.
- You should also write unit tests and integration tests for your API, using any testing framework of your choice, and provide instructions on how to run them.

Bonus (optional):

- Support soft deletion of agents and tasks
- Create endpoint(s) and data model(s) to enable users to run a task with a particular agent. The user should be able to start a task and get a list of currently running tasks.
- Include a README file explaining how to set up and run the API, and test everything
- Add rate limiting to your API