```
from swarms.structs.base_swarm import BaseSwarm
from swarms.structs.omni_agent_types import OmniAgentTypes
from typing import Optional, Sequence, List
from swarms_memory import BaseVectorDatabase



class FederatedSwarm(BaseSwarm):
    def __init__(
        self,
        name: Optional[str] = "FederatedSwarm",
        description: Optional[str] = "A swarm of swarms",
        swarms: Optional[Sequence[BaseSwarm]] = None,
        memory_system: BaseVectorDatabase = None,
        max_loops: Optional[int] = 4,
        *args,
        **kwargs,
    ):
        super().__init__(
            name=name, description=description, *args, **kwargs
        )
        self.name = name
        self.description = description
        self.swarms = swarms
        self.memory_system = memory_system
        self.max_loops = max_loops
```

```
def add_swarm(self, swarm: BaseSwarm):
    self.swarms.append(swarm)

def remove_swarm(self, swarm: BaseSwarm):
    self.swarms.remove(swarm)

def get_swarm(self, name: str) -> BaseSwarm:
    for swarm in self.swarms:
        if swarm.name == name:
            return swarm
    return None

def get_swarm_agents(self) -> List[OmniAgentTypes]:
    agents = []
    for swarm in self.swarms:
        agents.extend(swarm.agents)
    return agents

def get_swarm_agent(self, name: str) -> OmniAgentTypes:
    for swarm in self.swarms:
        for agent in swarm.agents:
            if agent.name == name:
                return agent
    return None

def get_swarm_agent_by_id(self, agent_id: str) -> OmniAgentTypes:
```

```
    for swarm in self.swarms:
        for agent in swarm.agents:
            if agent.agent_id == agent_id:
                return agent
    return None


async def run_single_swarm(
    self, swarm: BaseSwarm, *args, **kwargs
):

    await swarm.run(*args, **kwargs)


async def run_multiple_swarms(self, *args, **kwargs):
    for swarm in self.swarms:
        await self.run_single_swarm(swarm, *args, **kwargs)
```