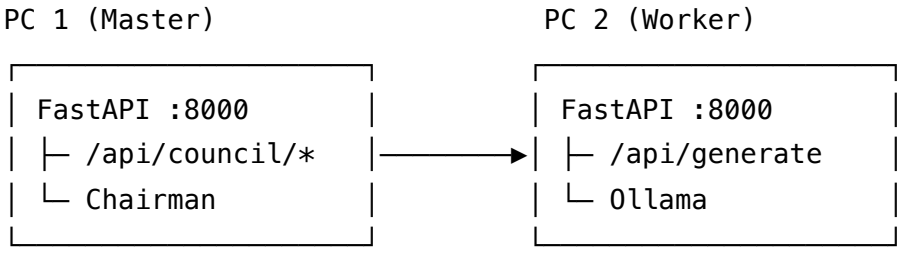
LLM Council Backend

Backend distribué pour le **LLM Council** - Un système de consensus local utilisant plusieurs LLMs.

🏗️ Architecture

Le système utilise une architecture **Master/Worker** distribuée :



📦 Installation

Prérequis

- Python 3.12+
- uv (gestionnaire de paquets)
- Ollama installé et en cours d'exécution

Installation des dépendances

```
cd backend
uv sync
```

Configuration Ollama (PC 2)

```
# Permettre le parallélisme
export OLLAMA_NUM_PARALLEL=5
export OLLAMA_MAX_LOADED_MODELS=5
ollama serve
```

Modèles recommandés

```
# Télécharger les modèles
ollama pull qwen2.5:0.5b    # Opinions rapides
ollama pull llama3.2:1b     # Review/notation
ollama pull phi3.5:latest     # Chairman
```

🚀 Démarrage

Mode Worker (PC 2 - Inference LLM)

```
cd backend
uv run python -m src.main --role worker
```

Mode Master (PC 1 - Orchestration)

```
cd backend
uv run python -m src.main --role master --worker-url http://PC2_IP:8000
```

Mode développement (tout-en-un)

```
# Terminal 1: Worker
uv run python -m src.main --role worker --port 8001

# Terminal 2: Master
uv run python -m src.main --role master --worker-url http://localhost:8001
```

📡 API Endpoints

Health Check

- `GET /health` - Statut du service
- `GET /health/ollama` - Connexion Ollama
- `GET /health/system` - CPU/RAM
- `GET /health/models` - Modèles disponibles

Worker (PC 2)

- `POST /api/generate` - Génération LLM simple
- `POST /api/generate/batch` - Génération parallèle

Master (PC 1)

- `POST /api/council/query` - Lancer une délibération
- `GET /api/council/session/{id}` - État d'une session
- `GET /api/council/models` - Modèles recommandés
- `WebSocket /api/council/ws/{id}` - Streaming temps réel

📝 Exemple de requête

```
curl -X POST http://localhost:8000/api/council/query \
  -H "Content-Type: application/json" \
  -d '{
    "query": "Quelle est la capitale de la France ?",
    "selected_agents": [
      {"name": "Expert_1", "model": "qwen2.5:0.5b"},
      {"name": "Expert_2", "model": "llama3.2:1b"},
      {"name": "Expert_3", "model": "gemma2:2b"}
    ],
    "chairman_model": "phi3.5:latest"
  }'
```

🔧 Variables d'environnement

Variable	Défaut	Description
`ROLE`	`worker`	`master` ou `worker`
`HOST`	`0.0.0.0`	Adresse de bind
`PORT`	`8000`	Port d'écoute
`OLLAMA_BASE_URL`	`http://localhost:11434`	URL Ollama
`WORKER_URL`	`http://localhost:8000`	URL du Worker (Master)
`CHAIRMAN_MODEL`	`phi3.5:latest`	Modèle Chairman

Variable	Défaut	Description
GENERATION_TIMEOUT	120	Timeout génération (s)

📂 Structure

```
src/
├── main.py           # Point d'entrée CLI
├── config.py         # Configuration Pydantic
├── models/           # Modèles de données
│   └── council.py    # AgentConfig, CouncilSession, etc.
├── services/         # Logique métier
│   ├── ollama_client.py  # Client HTTP Ollama
│   └── council.py    # Orchestration Stage 1-2-3
└── api/              # Routes FastAPI
    ├── council_routes.py  # Master endpoints
    ├── worker_routes.py   # Worker endpoints
    └── health_routes.py   # Monitoring
```

📚 Documentation API

Après démarrage, accédez à :

- Swagger UI: http://localhost:8000/docs
- ReDoc: http://localhost:8000/redoc