SmartInvention PowerAutomation Flask 服務器

🚀 概述

PowerAutomation Flask服務器是SmartInvention項目的核心後端服務，專門設計來支持VSCode插件和整個PowerAutomation生態系統的運行。

✨ 主要功能

- **VSCode插件API支持** - 為VSCode擴展提供完整的RESTful API接口
- **TRAE集成** - 無縫集成TRAE命令行工具，支持消息發送和數據同步
- **Manus自動化控制** - 提供Manus瀏覽器自動化的API接口
- **智能介入分析** - 對話內容分析和智能介入決策
- **對話歷史管理** - 完整的對話數據存儲和檢索
- **倉庫感知存儲** - 按Git倉庫組織和管理數據
- **跨域支持** - 完整的CORS配置，支持前端應用訪問

📁 項目結構

```
powerautomation_server/
├── venv/                          # Python虛擬環境
├── src/                           # 源代碼目錄
│   ├── main.py                    # Flask應用主入口
│   ├── models/                    # 數據模型
│   │   └── user.py                # 用戶模型
│   ├── routes/                    # API路由
│   │   ├── user.py                # 用戶相關API
│   │   └── powerautomation.py     # PowerAutomation核心API
│   ├── integrations/              # 集成模塊
│   │   └── trae_integration.py    # TRAE集成
│   ├── static/                    # 靜態文件
│   │   ├── index.html             # 默認首頁
│   │   └── favicon.ico            # 網站圖標
│   └── database/                  # 數據庫文件
│       └── app.db                 # SQLite數據庫
└── requirements.txt               # Python依賴
```

🔧 安裝和配置

環境要求

- Python 3.11+
- Flask 3.1+
- SQLite 3
- TRAE命令行工具（可選）

安裝步驟

1. **克隆倉庫**

```
git clone https://github.com/alexchuang650730/aicore0622.git
cd aicore0622/smartinvention/powerautomation_server
```

1. **激活虛擬環境**

```
source venv/bin/activate
```

1. **安裝依賴**

```
pip install -r requirements.txt
```

1. **啟動服務器**

```
python src/main.py
```

服務器將在 `http://0.0.0.0:5000` 啟動。

📋 API接口文檔

基礎信息

- **基礎URL**: `http://localhost:5000`
- **API前綴**: `/api/powerautomation`
- **內容類型**: `application/json`
- **CORS**: 已啟用，支持所有來源

核心端點

1. 健康檢查

```
GET /api/powerautomation/health
```

響應示例:

```
{
  "service": "PowerAutomation Server",
  "status": "healthy",
  "version": "1.0.0",
  "timestamp": "2025-06-22T14:30:00Z",
  "system_status": {
    "running": true,
    "trae_connected": false,
    "manus_connected": false
  }
}
```

2. 系統狀態

```
GET /api/powerautomation/status
```

響應示例:

```
{
  "success": true,
  "status": {
    "running": true,
    "trae_connected": false,
    "manus_connected": false,
    "stats": {
      "messages_processed": 0,
      "interventions_made": 0,
      "success_rate": 0.0
    }
  },
  "capabilities": [
    "trae_integration",
    "manus_automation",
    "intelligent_intervention"
  ]
}
```

3. 啟動系統

```
POST /api/powerautomation/start
```

響應示例:

```
{
  "success": true,
  "message": "PowerAutomation系統啟動成功",
  "status": {
    "running": true,
    "trae_connected": false,
    "manus_connected": false
  }
}
```

4. TRAE消息發送

```
POST /api/powerautomation/trae/send
```

請求體:

```
{
  "message": "要發送的消息內容",
  "repository": "smartinvention"
}
```

響應示例:

```
{
  "success": true,
  "message": "要發送的消息內容",
  "repository": "smartinvention",
  "stdout": "TRAE輸出",
  "timestamp": "2025-06-22T14:30:00Z"
}
```

5. TRAE數據同步

```
POST /api/powerautomation/trae/sync
```

請求體:

```
{
  "repository": "smartinvention",
  "force": true
}
```

6. Manus連接

```
POST /api/powerautomation/manus/connect
```

響應示例:

```
{
  "success": true,
  "message": "Manus服務連接成功",
  "manus_status": {
    "connected": true,
    "logged_in": true,
    "capabilities": [
      "task_list_traversal",
      "file_classification",
      "batch_download"
    ]
  }
}
```

7. 獲取Manus任務列表

```
GET /api/powerautomation/manus/tasks
```

8. 對話分析

```
POST /api/powerautomation/analyze
```

請求體:

```
{
  "messages": [
    {"role": "user", "content": "用戶消息"},
    {"role": "assistant", "content": "助手回復"}
  ],
  "repository": "smartinvention",
  "conversation_id": "conv_001"
}
```

9. 智能介入

```
POST /api/powerautomation/intervene
```

請求體:

```
{
  "type": "suggestion",
  "target": "trae",
  "message": "介入消息內容",
  "context": {
    "repository": "smartinvention",
    "conversation_id": "conv_001"
  }
}
```

10. 系統測試

```
POST /api/powerautomation/test
```

請求體:

```
{
  "type": "full"
}
```

🔌 VSCode插件集成

插件配置

VSCode插件可以通過以下方式連接到Flask服務器：

```
const API_BASE_URL = 'http://localhost:5000/api/powerautomation';

// 健康檢查
const healthCheck = async () => {
  const response = await fetch(`${API_BASE_URL}/health`);
  return response.json();
};

// 啟動系統
const startSystem = async () => {
  const response = await fetch(`${API_BASE_URL}/start`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    }
  });
  return response.json();
};

// TRAE發送消息
const sendTraeMessage = async (message: string, repository?: string) => {
  const response = await fetch(`${API_BASE_URL}/trae/send`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ message, repository })
  });
  return response.json();
};
```

狀態監控

插件可以定期檢查系統狀態：

```
const monitorStatus = async () => {
  try {
    const status = await fetch(`${API_BASE_URL}/status`);
    const data = await status.json();

    // 更新插件狀態顯示
    updateStatusBar(data.status);
  } catch (error) {
    console.error('狀態檢查失敗:', error);
  }
};

// 每30秒檢查一次狀態
setInterval(monitorStatus, 30000);
```

🧪 測試和驗證

基礎測試

```
# 測試服務器啟動
curl http://localhost:5000/api/info

# 測試健康檢查
curl http://localhost:5000/api/powerautomation/health

# 測試系統狀態
curl http://localhost:5000/api/powerautomation/status
```

功能測試

```
# 啟動系統
curl -X POST http://localhost:5000/api/powerautomation/start

# 測試TRAE發送（如果TRAE可用）
curl -X POST http://localhost:5000/api/powerautomation/trae/send \
  -H "Content-Type: application/json" \
  -d '{"message": "測試消息", "repository": "smartinvention"}'

# 執行系統測試
curl -X POST http://localhost:5000/api/powerautomation/test \
  -H "Content-Type: application/json" \
  -d '{"type": "full"}'
```

🔧 TRAE集成

TRAE可用性檢查

服務器會自動檢查TRAE命令行工具的可用性：

```
# 檢查TRAE是否安裝
which trae

# 如果TRAE可用，服務器將啟用完整的TRAE功能
# 如果TRAE不可用，相關API將返回錯誤信息
```

TRAE命令支持

- `trae -` - 通過stdin發送消息
- `trae sync` - 同步倉庫數據
- `trae sync --force` - 強制同步
- `trae --version` - 獲取版本信息

🚀 部署指南

開發環境

```
# 啟動開發服務器
python src/main.py
```

生產環境

建議使用Gunicorn或uWSGI部署：

```
# 安裝Gunicorn
pip install gunicorn

# 啟動生產服務器
gunicorn -w 4 -b 0.0.0.0:5000 src.main:app
```

Docker部署

```
FROM python:3.11-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY src/ ./src/
EXPOSE 5000

CMD ["python", "src/main.py"]
```

🔍 故障排除

常見問題

1. **TRAE不可用**
2. 確保TRAE命令行工具已安裝
3. 檢查PATH環境變量
4. 驗證TRAE命令權限
5. **端口衝突**
6. 檢查端口5000是否被占用
7. 修改main.py中的端口配置
8. **CORS錯誤**
9. 確保Flask-CORS已安裝
10. 檢查CORS配置
11. **數據庫錯誤**
12. 確保database目錄存在
13. 檢查SQLite文件權限

日誌查看

```
# 查看Flask日誌
tail -f /var/log/powerautomation.log

# 或在開發模式下直接查看控制台輸出
```

📈 性能優化

建議配置

- **生產環境**: 使用Gunicorn + Nginx
- **數據庫**: 考慮升級到PostgreSQL
- **緩存**: 添加Redis緩存層
- **監控**: 集成Prometheus + Grafana

擴展性

服務器設計為模塊化架構，可以輕鬆添加新的集成模塊：

1. 在`src/integrations/`目錄添加新模塊
2. 在`src/routes/powerautomation.py`中添加相應的API端點
3. 更新API文檔

🤝 貢獻指南

1. Fork項目
2. 創建功能分支
3. 提交更改
4. 創建Pull Request

📄 許可證

本項目採用MIT許可證。

📞 支持

如有問題或建議，請：

1. 查看本文檔的故障排除部分
2. 檢查GitHub Issues
3. 聯繫開發團隊

PowerAutomation Flask服務器 - 為智能自動化提供強大的後端支持 🚀