مختبر موسيف للهندسة المعمارية • وثيقة فنية معتمدة

نظام موسيف للعملاء الأذكياء وشبكة بوابات الاستدلال المزدوجة

الهندسة المعمارية المتعمقة، بنية النماذج عالية التوافر، الرسوم البيانية التكرارية للحالة، والطوبولوجيا الوكيلة المستقلة

نطاق ومحتوى الوثيقة المعمارية:

• **القسم الأول (صفحات 1–20):** المواصفات الهندسية الفنية الشاملة باللغة الإنجليزية.

• **القسم الثاني (صفحات 21–40):** المواصفات الهندسية الفنية المتعمقة باللغة العربية.

• **الأنظمة الفرعية المغطاة:** محرك LangGraph، بوابة LiteLLM الموحدة (:4000)، بوابة موسيف لنظام التشغيل (:8443)، عقيدة تفكير فتاح، الحقن اللحظي للمهارات، ومصفوفة الأمان الدستورية.

إعداد: النواة الهندسية لنظام موسيف AGI

الإصدار: 1.0.0 (النسخة الإنتاجية)

التاريخ: أغسطس 2026

التصنيف: مرجع معماري تأسيسي

01. الهندسة المعمارية الشاملة وطوبولوجيا النظام

يُمثل **نظام موسيف للعملاء الأذكياء (Moussaif Deep Agents CLI OS)** بيئة برمجية مفتوحة المصدر عالية التطور، تم تصميمها وهندستها لدمج مواصفات العملاء الأذكياء المتعمقين (`deepagents`) من منظومة LangChain ومحرك الرسم البياني للحالة التكرارية (LangGraph) مع شبكة استدلال مزدوجة وعالية التوافر. يعمل النظام كمساعد هندسي وتشغيلي ذاتي القيادة على مستوى بيئة التطوير المحلية وموارد الذكاء السحابي الموزعة، مع التزام صارم بمبدأ انعدام التكرار التخزيني وميثاق التفكير الدستوري.

1.1 البديهيات المعمارية والمبادئ الحاكمة

- **عقيدة المؤشرات وانعدام التكرار التخزيني:** تتمركز قواعد المعرفة الضخمة (أكثر من 9.1 جيجابايت) و1,500 مهارة فائقة في المسار الرئيسي `~/.hermes/` ، وترتبط بها كافة المحركات التشغيلية عبر مؤشرات رمزية بدون استهلاك مساحات إضافية.
- **شبكة البوابات المزدوجة (Dual-Gateway Mesh):** توجيه ذكي وفائق السرعة للطلبات بين **بوابة موسيف لنظام التشغيل (Moussaif Gateway :8443)** و**بوابة LiteLLM الموحدة (Port :4000)**.
- **الحقن الديناميكي للمهارات عند الطلب (JIT):** إلغاء التكديس الثابت لسياق الرموز؛ حيث تُحقن المهارات التخصصية لحظياً عند الحاجة التنفيذية الفعلية.
- **محرك الحالة التكراري ونقاط التفتيش الحتمية:** رسم بياني للحالة مدعوم بفئة `MemorySaver` لعزل الجلسات واستمرارية السياق عبر المعرفات المتعددة.

1.2 مخطط التدفق الطوبولوجي الكامل



بوابة موسيف للنواة (Port :8443)

تربط المحركات الأصلية للنواة الخماسية K1-K5: نموذج Gemini 3.1 Pro ونموذج GPT-5.5 Pro، إلى جانب النماذج المحلية عبر Ollama/AirLLM بتكلفة استدلال صفرية.

بوابة LiteLLM الموحدة (Port :4000)

تجمع 35 نموذجاً مؤسسياً عبر 9 مجموعات موفرين (OpenRouter, NVIDIA NIM, TokenRouter, Perplexity, GitHub Models) مع تبديل تلقائي فوري عند الأعطال.

ضمان الجاهزية التشغيلية

يضمن النظام استمرارية عمل المهام الذاتية بنسبة 99.9% من خلال تطبيق استراتيجية التبديل التلقائي؛ فإذا واجه النموذج الأساسي قيود معدل الطلبات (Rate Limits)، يتم التحويل بسلاسة إلى النماذج الرديفة دون فقدان حالة المحادثة.

02. محرك LangGraph وتجميع الرسم البياني للحالة

يتمحور المحرك التنفيذي الأساسي لنظام موسيف في ملف `agent.py`؛ حيث يستند إلى مصنع بناء العملاء `create_deep_agent` المبني فوق مكتبة LangGraph، مما يحول الاستجابات الخطية التقليدية إلى حلقة تفكير ديناميكية ذات أهداف محددة وقدرة على التحقق الذاتي من الأدوات.

2.1 دورة حياة الرسم البياني التكراري

على عكس هياكل استدعاء الأدوات التقليدية، يُمثل LangGraph الإدراك المعرفي للعميل كقاموس حالة متطور يحتوي على الرسائل، والأدوات النشطة، ولقطات الذاكرة، وعدادات التكرار التكراري.

```
# نموذج تجميع الرسم البياني في agent.py
def create_moussaif_agent(model_name: str = "deep-agent-primary", gateway: str = "litellm"):
    endpoint_info = get_gateway_endpoint(gateway)
    system_prompt = load_system_prompt()
    tools = get_all_tools()
    checkpointer = get_memory_checkpointer()

    llm = ChatOpenAI(
        base_url=endpoint_info["base_url"],
        api_key=endpoint_info["api_key"],
        model=model_name,
        temperature=0.2
    )

    agent = create_deep_agent(
        model=llm,
        tools=tools,
        system_prompt=system_prompt,
        checkpointer=checkpointer
    )
    return agent
```

2.2 آليات الانتقال بين عُقد الحالة المعرفية

1. عقدة التفكير (Reasoning)

يقوم العميل بتحليل نية المستخدم بناءً على عقيدة ميثاق فتاح، مقرراً إما صياغة رد نهائي أو استدعاء إحدى الأدوات التنفيذية.

2. عقدة توجيه الأدوات (Router)

تتحقق من صحة معاملات JSON، وتنفذ الدوال المعزولة في `tools/`، ثم تُعيد حقن النتائج ضمن سياق الحالة.

3. عقدة نقطة التفتيش (Checkpoint)

تحفظ الفروقات اللحظية في `MemorySaver` تحت المعرف `thread_id` لضمان استرجاع الحالة والعودة للخلف.

التحكم في التكرار اللانهائي

تم ضبط الحد الأقصى للتكرار على `recursion_limit=50`، مما يمنع الحلقات المفرغة مع منح العميل عمقاً إدراكياً كافياً لإعادة هيكلة المشاريع البرمجية الضخمة.

03. ربط النماذج وعزل الموفرين برمجياً

يُعد التخلص من أخطاء استنتاج أسماء الموفرين في LangChain إنجازاً معمارياً جوهرياً في نظام موسيف؛ حيث تحاول دوال التهيئة التلقائية (`init_chat_model`) تحليل البادئات النصية (مثل `openai/` أو `anthropic/`) مما يؤدي إلى انهيار التشغيل عند استخدام بوابات وسيطة محلية.

3.1 معضلة تضارب بادئات الموفرين والحل المعماري

عند طلب استدعاء أسماء مركبة مثل `deep-agent-primary` عبر LiteLLM أو بوابة موسيف، تفشل المعالجات النصية العامة لعدم مطابقة الاسم لنطاقات الموفرين الرسمية السحابية.

النهج التقليدي المعيب

```
# يفشل مع الأسماء المركبة للبوابات
agent = create_deep_agent(
    model="deep-agent-primary"
)
# خطأ: بادئة موفر غير معروفة
```

حل موسيف الهندسي القاطع

```
# ربط مباشر لكائن ChatOpenAI مهيأ بالكامل
llm = ChatOpenAI(
    base_url=endpoint["base_url"],
    api_key=endpoint["api_key"],
    model=model_name
)
agent = create_deep_agent(model=llm)
```

3.2 قاموس تحليل الهوية ونقاط النهاية (`identity.py`)

يقوم ملف `identity.py` بدور المحلل المركزي، حيث يربط البوابات المطلوبة بعناوين الاسترجاع المحلي الموثوقة ومفاتيح البيئة المحمية.

مفتاح البوابة	عنوان نقطة النهاية المحلية	آلية المصادقة والتوثيق	المحركات المستهدفة
`litellm` (الافتراضية)	`http://localhost:4000/v1`	`LITELLM_MASTER_KEY` / Bearer	35 نموذجاً (OpenRouter, NIM, TokenRouter, GitHub)
`moussaif`	`http://localhost:8443/v1`	`MOUSSAIF_GATEWAY_KEY` / Bearer	محركات النواة K1-K5 (Antigravity, Codex, Ollama)
`openrouter`	`https://openrouter.ai/api/v1`	`OPENROUTER_API_KEY`	المسار الاحتياطي المباشر لـ OpenRouter

04. بوابة نظام موسيف الأساسية (Moussaif Gateway :8443)

تُعد **بوابة نظام تشغيل موسيف الأساسية** العاملة على المنفذ `8443` جسر الربط العصبي المباشر الذي يربط بنية النواة الخماسية (K1-K5) بالأدوات التنفيذية المحلية وسيناريوهات الأتمتة المتقدمة.

4.1 طوبولوجيا النواة الخماسية (K1-K5 Teaming Architecture)

K1: هيرميس المايسترو (hermes./~)

المنسق التنفيذي الأعلى لإدارة فهارس المعرفة، وجداول المهام المجدولة (Cron)، والذاكرة الدائمة، وتوزيع المهام الذاتي.

K2: أوبن كود السريع (opencode./~)

محرك التنفيذ فائق السرعة لتحليل الفروقات البرمجية (Diff Analysis)، وتنفيذ أوامر الطرفية، وفحص البنية النحوية للكود.

K3: كودكس الاحترافي (codex./~)

يعمل بنموذج GPT-5.5 Pro؛ وهو المسؤول عن إدارة عمليات Git، وطلبات الدمج (PRs)، واختبارات التحقق التراجعي.

K4: كلود بوابة الجودة (claude./~)

حارس الميثاق والتوثيق المتعمق، وتدقيق الثغرات الأمنية، والمصادقة على بوابات القبول والاعتماد النهائي.

4.2 محرك K5-Antigravity والبث المباشر للنماذج المحلية

تتكامل البوابة مباشرة مع محرك (Gemini 3.1 Pro High) **K5-Antigravity** لمعالجة المشاريع الضخمة وإنشاء واجهات المستخدم، إلى جانب منصات **Ollama** و **AirLLM** للبث الطبقي للنماذج العملاقة (70B+ بارامتر) على كروت الشاشة المحلية.

```
# إرسال الأوامر عبر جسر موسيف السطري
python C:\Users\mouss\moussaif-bridge\moussaif-cli.py "تدقيق البنية البرمجية" -m gemini-3.1-pro -p antigravity
python C:\Users\mouss\moussaif-bridge\moussaif-cli.py "تشغيل اختبارات الوحدة" -m gpt-5.5-pro -p codex
```

فئة التكلفة الصفرية

تستهلك أوامر التطوير الموجهة عبر المنفذ 8443 إلى نماذج Ollama المحلية ميزانية صفرية تماماً، مما يتيح تشغيل مهام المراقبة والتحقق الأمني على مدار الساعة دون أعباء مالية.

05. بنية بوابة LiteLLM الموحدة (:4000)

تم تكوين البوابة معمارياً عبر الملف المرجعي `C:\Users\mouss\OneDrive\Hermes\litellm_config.yaml` ؛ حيث ينشئ خادم LiteLLM شبكة توجيه موحدة متوافقة مع معايير OpenAI على المنفذ `4000` ، موحداً البروتوكولات المتباينة في واجهة JSON واحدة ومستقرة.

5.1 مصفوفة المجموعات التسع وسجل الـ 35 نموذجاً

المجموعة	موفر الخدمة الخلفي	النماذج الأساسية المتاحة	الوظيفة الاستراتيجية والمعمارية
المجموعة 1	OpenRouter	`ox-alpha` , `liquid-free` , `claude-opus-4`	المسار الاحتياطي العالمي ونماذج التجارب الفائقة
المجموعة 2	NVIDIA NIM	`nemotron-70b` , `llama-3.1-70b` , `llama-3.1-8b`	استدلال مؤسسي عالي التوازي وسرعة توليد فائقة
المجموعة 3	TokenRouter	`qwen3.8-max-free` , `deepseek-v4-pro`	التشغيل الدفعي الاقتصادي والاستدلال الرياضي المعقد
المجموعة 4	Perplexity AI	`sonar` , `sonar-pro` , `sonar-reasoning`	البحث الحي اللحظي عبر الإنترنت واستخراج الأدلة
المجموعة 5	OpenCode Zen	`minimax-m3` , `ox-alpha-free`	إكمال الأكواد بسرعة عالية والتحقق من القواعد البرمجية
المجموعة 6	Moussaif Gateway	`antigravity/gemini-3.1-pro` , `codex/gpt-5.5`	التوجيه المباشر لوكلاء النواة K1-K5
المجموعة 7	Direct Vendor APIs	`openai/gpt-4o` , `anthropic/claude-sonnet-4`	التصعيد الحرج للإنتاج بدون وسائط خارجية
المجموعة 8	GitHub Models	`github/gpt-4o` , `github/mistral-large`	الاستدلال السحابي عبر بنية Azure التحتية بمفاتيح GitHub
المجموعة 9	Deep Agents Composite	`deep-agent-primary` , `deep-agent-fast`	الأسماء المركبة ذات التوازن الذاتي والمسارات البديلة

06. الأسماء المركبة المتوازنة والمحمية (Load-Balanced Aliases)

لحماية المهام المستقلة من التوقف المفاجئ الناتج عن انقطاع خوادم الموفرين، أو استنزاف حدود الاستخدام، أو ارتفاع زمن الاستجابة، يجرد نظام موسيف اختيار النماذج خلف ثلاثة أسماء مركبة ذات موثوقية تشغيلية فائقة.

6.1 بنية المسارات التبادلية للأسماء المركبة

```
النموذج الأساسي (deep-agent-primary): Claude 3.5 Sonnet ← GPT-4o ← Gemini 2.5 Pro ← OpenRouter
النموذج السريع (deep-agent-fast): Gemini 2.5 Flash ← GPT-4o-mini ← NVIDIA Llama 3.1 8B ← Liquid Free
نموذج الأبحاث (deep-agent-research): Perplexity Sonar Pro ← DeepSeek Reasoner ← Sonar Live
```

6.2 كود تكوين التبديل التلقائي في LiteLLM

```yaml
# مقتطف من تكوين litellm_config.yaml
router_settings:
  routing_strategy: "least-busy"
  num_retries: 3
  timeout: 45
  allowed_fails: 2
  cooldown_time: 60

model_list:
  - model_name: deep-agent-primary
    litellm_params:
      model: anthropic/claude-3-5-sonnet-20241022
      api_key: "os.environ/ANTHROPIC_API_KEY"
      fallbacks: ["openai/gpt-4o", "gemini/gemini-2.5-pro", "openrouter/anthropic/claude-3.5-sonnet"]

  - model_name: deep-agent-fast
    litellm_params:
      model: gemini/gemini-2.5-flash
      api_key: "os.environ/GEMINI_API_KEY"
      fallbacks: ["openai/gpt-4o-mini", "nvidia_nim/meta/llama-3.1-8b-instruct"]
```

قواطع الدوائر وفترات التبريد الذكية

عند استقبال خطأ HTTP 429 أو 503 من موفر معين، تقوم البوابة بتفعيل قاطع الدائرة للنموذج لمدة 60 ثانية، مع تحويل الطلبات فوراً للبديل دون إشعار العميل بأي توقف.

07. النواة الإدراكية لحلقة تفكير فتاح (FATTAH Reasoning Loop)

تُمثل عقيدة **فتاح (FATTAH)** المعرفية العمود الفقري لمنطق اتخاذ القرار في نظام موسيف. يتم حقن هذه العقيدة ضمن ملف `instructions.md` لفرض أسلوب الاستدلال المنطقي والتجريبي الصارم على النماذج اللغوية.

7.1 المراحل المعرفية وآليات التحقق

المرحلة	الفعل الإدراكي	الآلية التنفيذية وبوابة التحقق الفنية
ف (F) — تأطير (Frame)	تحديد الهدف الحقيقي	عزل الهدف، وتحديد الأنظمة والمسارات المتأثرة، والقيود الصارمة.
ت (A) — تفكيك (Atomize)	التشريح الميكانيكي	تفكيك المشكلة لمكونات ذرية، وتحديد المجاهيل ومسارات الفشل المحتملة.
ت (T) — تثليث (Triangulate)	مطابقة الأدلة متعددة الزوايا	قراءة الملفات الحية، والتحقق من البيئة، ومطابقة مخرجات الأدوات.
ا (T) — اختبار (Test)	فحص الفرضيات العكسية	اختبار السيناريوهات المعاكسة، والتحقق من قابلية الرجوع عن الخطوات.
ح (A) — حسم وعمل (Act)	التنفيذ الذري الأدنى	تطبيق أصغر خطوة قابلة للعكس باستخدام أدوات النظام والطرفية المعزولة.
ص (H) — حصاد (Harvest)	تحديث نموذج العالم والذاكرة	التحقق من صحة الناتج، وحفظ الدروس المستفادة، وتحديث الذاكرة الدائمة.

7.2 القوانين العشرة الدستورية الحاكمة

1. **الحالة المرصودة تسبق الحالة المتذكرة:** يجب فحص الملفات الفعلية عبر الأدوات قبل إجراء التعديل.
2. **أصغر إجراء كامل قابل للعكس:** يُحظر تماماً تنفيذ عمليات حذف واسعة النطاق دون أمان تراجعي.
3. **التوثيق أو كأنه لم يحدث:** يجب تسجيل كافة العمليات بسجلات ومخرجات قابلة للتدقيق.
4. **الاعتراف الصادق بالعجز أفضل من النجاح الوهمي:** يُمنع توليد نتائج وهمية عند فشل الأدوات.

08. استمرارية الحالة ونقاط التفتيش للذاكرة (`memory.py`)

تتطلب الاستقلالية الكاملة للوكلاء الذكاء قدرة على إدارة الحالة عبر الجلسات المتعددة وعمليات التنفيذ الفرعية. في نظام موسيف، تتم إدارة الذاكرة بواسطة `memory.py` عبر فئة `MemorySaver` في LangGraph.

8.1 هندسة الرسم البياني للحالة المعزولة بالخيوط

يتم تخصيص معرف فريد `thread_id` (مثل `moussaif-session-default`) لكل محادثة أو تنفيذ، مما يتيح حفظ لقطات الذاكرة، والتراجع الزمني، واستئناف المهام السابقة بدقة تامة.

```
# معمارية إدارة الذاكرة في memory.py
from langgraph.checkpoint.memory import MemorySaver

_GLOBAL_CHECKPOINTER = None

def get_memory_checkpointer() → MemorySaver:
    global _GLOBAL_CHECKPOINTER
    if _GLOBAL_CHECKPOINTER is None:
        _GLOBAL_CHECKPOINTER = MemorySaver()
    return _GLOBAL_CHECKPOINTER

def get_thread_config(thread_id: str = "moussaif-session-default") → dict:
    return {"configurable": {"thread_id": thread_id}}
```

8.2 تدفق مزامنة الحالة متعددة الأدوار

التقاط لقطات الحالة اللحظية

عند كل تنفيذ لأداة أو توليد استجابة، يقوم LangGraph بتسلسل الرسائل ومخرجات الأدوات داخل شجرة الذاكرة المعزولة.

السفر عبر الزمن والتشعب

من خلال تحديد معرف نقطة تفتيش سابقة، يمكن للعميل التراجع عن التعديلات الفاشلة وإعادة المحاولة عبر مسار منطقي بديل.

أمان الخيوط البرمجية والتزامن

يضمن تطبيق نمط المفرد (Singleton) في `memory.py` الوصول الآمن لنقاط التفتيش أثناء تشغيل أسراب الوكلاء المتزامنة دون أي تداخل في البيانات.

09. النظام المعياري للأدوات التنفيذية (`/tools`)

تُشكل حزمة الأدوات في `tools/__init__.py` حلقة الوصل الفعلية بين التفكير الاستدلالي للنماذج والعمليات الفيزيائية لنظام التشغيل؛ حيث تُعرف الأدوات كدوال مزخرفة بـ `@tool` مع توثيق Pydantic صارم.

9.1 سجل الأدوات التنفيذية المعتمدة

اسم الأداة	الوحدة البرمجية	مخطط المدخلات والمحددات	فئة التفويض الأمني
`read_file`	`filesystem.py`	`file_path: str, max_lines: int`	T1 (قراءة فقط)
`write_file`	`filesystem.py`	`file_path: str, content: str`	T2 (تعديل قابل للعكس)
`list_directory`	`filesystem.py`	`dir_path: str, recursive: bool`	T1 (استعراض قراءة)
`inspect_codebase`	`filesystem.py`	`root_dir: str`	T1 (فحص شامل)
`run_terminal_command`	`terminal.py`	`command: str, cwd: str, timeout: int`	T2 (تنفيذ أوامر طرفية)
`load_agent_skill`	`skills_loader.py`	`skill_name: str`	T1 (حقن سياق معرفي)
`list_available_skills`	`skills_loader.py`	`None`	T1 (استعراض المهارات)

9.2 التحقق من صحة المخطط والحماية من الهلوسة

يُلزم LangGraph النماذج بإصدار هياكل JSON مطابقة لتوقيع الدوال البرمجية. وفي حال حدوث خلل تركيبي، يعترض النظام الخطأ ويُعيد حقن تنبيه تصحيحي في حلقة التفكير دون انهيار التطبيق.

10. هندسة أداة نظام الملفات المحمي (`filesystem.py`)

تُقدم وحدة `tools/filesystem.py` عمليات آمنة ومحمية للتعامل مع الملفات عبر مختلف أنظمة التشغيل، مع فرض ترميز UTF-8 في بيئة ويندوز، والإنشاء التلقائي للمجلدات، وفحص سلامة المسارات.

10.1 المميزات الهندسية الأساسية

- **الإنشاء التلقائي للأدلة الأبوية:** عند محاولة الكتابة في مسارات متداخلة غير موجودة مسبقاً، تقوم `write_file` بإنشاء المجلدات عبر `os.makedirs()`.
- **حماية سعة الذاكرة من الإغراق:** تتضمن أداة `read_file` محددات اختيارية `max_lines` لحماية الذاكرة من القراءة العشوائية للملفات العملاقة.
- **الفلترة الذكية واستبعاد الضوضاء:** تتجاهل أدوات فحص المجلدات ملفات النظام ومجلدات الحزم مثل `.git` و `node_modules` و `__pycache__`.

10.2 كود أداة الكتابة الآمنة

```
@tool
def write_file(file_path: str, content: str) → str:
    \"\"\"كتابة محتوى نصي إلى ملف محلي مع إنشاء المجلدات الأبوية تلقائياً.\"\"\"
    try:
        norm_path = os.path.abspath(os.path.expanduser(file_path))
        os.makedirs(os.path.dirname(norm_path), exist_ok=True)
        with open(norm_path, "w", encoding="utf-8") as f:
            f.write(content)
        return f"[SUCCESS] تم حفظ الملف بنجاح في: {norm_path} ({len(content)} حرف)"
    except Exception as e:
        return f"[ERROR] فشل حفظ الملف {file_path}: {str(e)}"
```

أمان العمليات الذرية

تعتمد كافة عمليات الحفظ على دوال التدفق المباشر المؤمّنة، مما يمنع تلف الملفات أو انقطاعها أثناء حالات الإيقاف الطارئ للنظام.

11. هندسة محرك تنفيذ الأوامر بالطرفية (`terminal.py`)

يتطلب تشغيل أوامر الطرفية الذاتية في بيئة ويندوز معالجة دقيقة لترميز الحروف، والتحكم في العمليات المعلقة، وعزل مسارات العمل التنفيذية. تُوفر وحدة `tools/terminal.py` هذه الضمانات الهندسية.

11.1 درع الحماية من أخطاء الترميز (CP1252 vs. UTF-8)

تعتمد طرفية ويندوز الافتراضية على ترميز CP1252 القديم. وإذا أصدرت أدوات البناء أو Git رموزاً دولية، تتوقف أنابيب Python القياسية بخطأ `UnicodeDecodeError`. يعالج محرك موسيف ذلك بفرض ترميز UTF-8 مع معالجة ذكية للرموز غير المعرفة.

```
# بنية مشغل الطرفية في terminal.py
@tool
def run_terminal_command(command: str, cwd: str = None, timeout: int = 120) → str:
    \"\"\"تنفيذ أمر طرفي محلي والتقاط المخرجات وقنوات الخطأ بأمان تام.\"\"\"
    target_cwd = os.path.abspath(os.path.expanduser(cwd)) if cwd else os.getcwd()
    try:
        process = subprocess.Popen(
            command,
            shell=True,
            cwd=target_cwd,
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            text=True,
            encoding="utf-8",
            errors="replace"
        )
        stdout, stderr = process.communicate(timeout=timeout)
        exit_code = process.returncode
        return f"[EXIT: {exit_code}]\\nSTDOUT:\\n{stdout}\\nSTDERR:\\n{stderr}"
    except subprocess.TimeoutExpired:
        process.kill()
        return f"[TIMEOUT] تم إيقاف الأمر لتجاوز المهلة المحددة ({timeout} ثانية)."
```

11.2 قواعد السلامة لتنفيذ أوامر الطرفية

حظر استخدام أمر 'cd' التفاعلي

تُعد العمليات الفرعية مؤقتة؛ ولذا يجب على العميل تمرير المعامل `cwd=" ... "` بدلاً من محاولة تغيير المسار تسلسلياً.

الحارس الزمني للإيقاف التلقائي

تخضع الأوامر لمهلة افتراضية قدرها 120 ثانية لمنع تعليق النظام بالخوادم المستمرة أو مدخلات القراءة التفاعلية.

12. محرك الحقن الديناميكي للمهارات (`skills_loader.py`)

يُعد تلوث السياق المعرفي عقبة رئيسية في بنية الوكلاء التقليدية؛ حيث يؤدي حشو مئات المهارات في رسالة النظام التأسيسية إلى استنزاف نافذة الرموز ورفع التكاليف وزيادة احتمالات الهلوسة. يحل نظام موسيف هذه المعضلة عبر **الحقن المعرفي عند الطلب (JIT)**.

12.1 المقارنة المعمارية: الحقن اللحظي مقابل التكديس الثابت

العميل التقليدي: 150,000 رمز (رسالة النظام + كافة المهارات) ← بطء شديد وتكلفة باهظة

عميل موسيف JIT: 2,500 رمز فقط ← العميل يستدعي `load_agent_skill("coding")` ← تُحقن 800 رمز تخصصية بدقة

12.2 هيكلية مجلدات المهارات وآلية الاكتشاف

تُحفظ المهارات في مجلدات منفصلة `skills//SKILL.md`. يحتوي كل ملف على ترويسة YAML وقواعد العمل وخطوات التحقق الفنية.

```
# كود محمل المهارات الديناميكي في skills_loader.py
@tool
def load_agent_skill(skill_name: str) → str:
    \"\"\"تحميل تعليمات المهارة التخصصية في سياق العميل النشط لحظياً عند الطلب.\"\"\"
    skill_path = os.path.join(SKILLS_DIR, skill_name, "SKILL.md")
    if not os.path.exists(skill_path):
        return f"[ERROR] المهارة '{skill_name}' غير موجودة. المتاح: {list_skills()}"
    with open(skill_path, "r", encoding="utf-8") as f:
        content = f.read()
    return f"[SKILL LOADED: {skill_name}]\\n{content}"
```

التكامل مع المؤشرات المرجعية

يفحص محمل المهارات المجلد المحلي أولاً، وفي حال عدم العثور عليها، يتصل تلقائياً بالمستودع الرئيسي `~/.hermes/skills/` الذي يضم أكثر من 1,500 مهارة فائقة.

13. كتالوج المهارات التخصصية

يأتي كتالوج المهارات الأصلي في `C:\Users\mouss\moussaif-bridge\moussaif-deep-agents\skills` مزوداً بأربع مهارات تخصصية تم تصميمها هندسياً لخدمة دورات التطوير والعمليات المستقلة.

13.1 المواصفات الفنية للمهارات التأسيسية

1. مهارة البرمجة (`skills/coding/SKILL.md`)

الوظيفة: الهندسة البرمجية الكاملة، والتطوير الموجه بالاختبارات (TDD)، وإعادة الهيكلة، والتدقيق النحوي.
المنهجية: حلقة Red-Green-Refactor، والحفاظ الصارم على توثيق الكود، وفرض أمان الأنواع.

2. مهارة الأبحاث (`skills/research/SKILL.md`)

الوظيفة: التوليف البحثي والأكاديمي المعمق وتدقيق البنى التحتية.
المنهجية: تثليث الأدلة عبر 7 مستويات، وتوثيق المصادر الأولية، ومصفوفات مقارنة الحلول التنافسية.

3. أتمتة جيت هب (`skills/github-automation/SKILL.md`)

الوظيفة: الإدارة الذاتية الكاملة لدورة حياة مستودعات Git.
المنهجية: إنشاء الفروع الآمنة، والالتزامات الدلالية (Semantic Commits)، وطلبات الدمج المؤتمتة.

4. تخطيط الوكلاء (`skills/multi-agent-planning/SKILL.md`)

الوظيفة: تفكيك المهام عبر الرسوم البيانية الموجهة غير الدائرية (DAG).
المنهجية: خرائط الاعتماديات، وتوزيع المهام المتوازي، ومتابعة المراحل، وخطط التراجع الحتمية.

13.2 بروتوكول التوسيع الديناميكي للمهارات

يمكن إضافة مهارات جديدة (مثل `docker-deployment`) ببساطة عن طريق إنشاء مجلد فرعي يحتوي على ملف `SKILL.md`، وسيكتشفها النظام تلقائياً عبر دالة `list_available_skills`.

14. بيئة العمل التفاعلية والسطرية (`cli.py`)

تُمثل الواجهة السطرية في `cli.py` سطح التفاعل الرئيسي للمستخدم؛ حيث تدعم بيئة عمل تفاعلية متعددة الأدوار (REPL) ونمط التنفيذ الأحادي للأتمتة في الخلفية.

14.1 محددات ومعاملات السطر البرمجي

معامل السطر البرمجي	النوع	القيمة الافتراضية	الوظيفة والوصف الفني
`-m` , `--model`	نصي	`deep-agent-primary`	اسم النموذج المطلوب أو الاسم المركب المتوازن
`-g` , `--gateway`	نصي	`litellm`	بوابة التوجيه (`litellm` , `moussaif` , `openrouter`)
`-q` , `--query`	نصي	`None`	تنفيذ استعلام أحادي فوري بدون دخول الوضع التفاعلي
`-t` , `--thread-id`	نصي	`moussaif-session-default`	معرف الخيط البرمجي لاستمرارية واسترجاع الذاكرة

14.2 الأوامر المضمنة في البيئة التفاعلية (REPL)

- `/skills` — استعراض قائمة المهارات التخصصية المتاحة للاستخدام.
- `/load <name>` — حقن مهارة معينة يدوياً في سياق الجلسة الحالية.
- `/model <name>` — تبديل نموذج الاستدلال النشط فورياً دون إعادة تشغيل الجلسة.
- `/reset` — تصفير الذاكرة الحالية وبدء خيط محادثة جديد.
- `/exit` , `/quit` — إنهاء جلسة السطر البرمجي بأمان.

14.3 شارات الحالة المعتمدة على معيار ASCII

لتجنب مشاكل تشويه الرموز التعبيرية في موجه أوامر ويندوز، تعتمد الواجهة شارات قياسية واضحة: `[+]` للنجاح، و `[*]` لأفعال وتفكير العميل، و `[-]` للتحذيرات، و `[!]` للأخطاء الحرجة.

15. المشغلات المحلية وبيئة التشغيل لويندوز

لتوفير وصول سريع وفوري للمطورين، تم تجهيز نظام موسيف بمشغلات دفعية آلية تدير تفعيل البيئة الافتراضية، وضبط متغيرات المسار، وفرض ترميز UTF-8 في بيئة ويندوز.

15.1 مشغل سطح المكتب السريع (`DeepAgent.bat`)

يتواجد الملف على سطح المكتب في `C:\Users\mouss\OneDrive\Desktop\DeepAgent.bat`، ويتيح التشغيل الفوري بنقرة واحدة.

```
@echo off
TITLE Moussaif Deep Agents CLI OS
COLOR 0B
chcp 65001 >nul
cd /d "C:\Users\mouss\moussaif-bridge\moussaif-deep-agents"
call "C:\Users\mouss\AppData\Local\hermes\hermes-agent\venv\Scripts\activate.bat"
python cli.py
pause
```

15.2 طوبولوجيا البيئة الافتراضية وحزم البناء

يعمل النظام داخل بيئة افتراضية محصنة في `C:\Users\mouss\AppData\Local\hermes\hermes-agent\venv` بالاعتماد على Python 3.11، وتتضمن الحزم التالية:

- `deepagents` (v0.7.9) — محرك العملاء الأذكياء المتعمقين.
- `langgraph` (v1.2.11) — محرك الرسوم البيانية للحالة التكرارية.
- `langchain` (v1.3.17) & `langchain-openai` (v1.6.0) — جسور ربط النماذج.
- `litellm` (v1.73.1) — الخادم الوسيط وموزع الأحمال الذكي.

16. مصفوفة الأمان ومستويات التفويض الدستورية (T0–T4)

تتطلب الأنظمة المستقلة تطبيق حدود تشغيلية صارمة لمنع التدمير العرضي للبيانات أو تسريب بيانات الاعتماد الحساسة. يُطبق نظام موسيف نموذج التفويض الدستوري خماسي المستويات.

16.1 مصفوفة مستويات التفويض والأمان

المستوى	اسم الفئة الأمنية	العمليات والإجراءات المسموح بها	شروط ومتطلبات التفويض
T0	التفكير والاستدلال المحض	التفكير الداخلي، والتخطيط، ورسم هياكل DAG، وتحليل البيانات	مسموح به ذاتياً بشكل دائم
T1	القراءة والإنشاء المحلي	قراءة الملفات، وفحص المجلدات، وإنشاء ملفات جديدة في المشروع	مسموح به افتراضياً
T2	التعديل المحلي القابل للعكس	تعديل الأكواد القائمة، والالتزام المحلي في Git، وتشغيل الاختبارات	يتطلب وجود نقطة تفتيش للرجوع
T3	الآثار الجانبية الخارجية	دفع التعديلات للخوادم البعيدة (git push)، واستدعاء APIs خارجية	يتطلب موافقة صريحة من المستخدم
T4	العمليات المحظورة قطعياً	الحذف التكراري للنظام (`rmdir /s C:\`)، وتفريغ المفاتيح السرية	محظور منعاً باتاً ولا يُنفذ إطلاقاً

16.2 حماية وتشفير مفاتيح API

لا يتم تخزين المفاتيح في ملفات برمجية صريحة؛ بل تُحقن عبر متغيرات البيئة المشفرة في الذاكرة اللحظية بواسطة `identity.py` .

17. التكرارية عبر الموفرين واقتصاد الرموز

يتطلب تحقيق سرعات استجابة لحظية في البيئة التفاعلية مع الحفاظ على جودة استدلال فائقة إدارة دقيقة لزمن الوصول وتطبيق شلال التكلفة الذكي لاقتصاد الرموز.

17.1 شلال اقتصاد الرموز: المجاني أولاً

في المهام الروتينية (مثل التدقيق النحوي، وإنشاء هياكل الفهارس)، يُوجه النظام الطلبات إلى الموفرين المجانيين والمحليين قبل التصعيد للنماذج الرائدة باهظة التكلفة.

```
المستوى 1 (محلي / مجاني): Ollama Fusion ← Liquid Free ← TokenRouter Qwen 3.8
                                    ↓ (إذا تجاوزت درجة التعقيد حداً معيناً)
المستوى 2 (سحابي سريع): Gemini 2.5 Flash ← GPT-4o-mini ← NVIDIA Llama 3.1 70B
                                    ↓ (إذا تطلبت المهمة تخطيطاً معمارياً شاملاً)
المستوى 3 (النماذج الرائدة): Claude 3.5 Sonnet ← GPT-5.5 Pro ← Gemini 3.1 Pro High
```

17.2 التوازي وتجميع الاتصالات الدائمة

يحافظ خادم LiteLLM على تجمعات اتصالات HTTP/2 غير متزامنة مع خوادم الموفرين، مما يُلغي أعباء مصافحة TLS ويقلل زمن الاستجابة من 1,200 ملي ثانية إلى أقل من 280 ملي ثانية.

18. منظومة الاختبارات والتحقق المستمر (`test_deep_agents.py`)

لضمان الاستقرار المعماري للنظام، يحتوي المشروع على حزمة اختبارات تكامل آلية في `C:\Users\mouss\moussaif-bridge\test_deep_agents.py` للتحقق من سلامة الطبقات التأسيسية الأربع.

18.1 مصفوفة الاختبارات الرباعية

حزمة الاختبار	المكون المستهدف	معيار التحقق والنجاح الفني	حالة التحقق
الحزمة 1	اتصال البوابات	استجابة HTTP 200 لفحص الصحة في LiteLLM (:4000) وبوابة موسيف (:8443)	**نجاح 100% (PASS)**
الحزمة 2	استدعاء النماذج	توليد استجابة مكتملة من `deep-agent-primary` و `deep-agent-fast`	**نجاح 100% (PASS)**
الحزمة 3	تنفيذ الأدوات والأمان	كتابة وقراءة الملفات، وتشغيل أوامر الطرفية، وسلامة ترميز UTF-8	**نجاح 100% (PASS)**
الحزمة 4	استمرارية الذاكرة	حفظ نقاط التفتيش، وعزل الخيوط، واسترجاع سياق المحادثة متعددة الأدوار	**نجاح 100% (PASS)**

18.2 تشغيل الاختبارات الآلية عبر الطرفية

```
# تشغيل حزمة اختبارات التكامل الشاملة
python C:\Users\mouss\moussaif-bridge\test_deep_agents.py

# الناتج النموذجي المعتمد:
# [+] Suite 1: Gateway Health Check ..... PASSED
# [+] Suite 2: Deep Agents Model Invocation .... PASSED
# [+] Suite 3: Tool Execution & Safety ..... PASSED
# [+] Suite 4: Memory Persistence Check ..... PASSED
# [ALL SUITES PASSED - 100% VERIFIED]
```

19. مرآة المعرفة الموحدة في نظام هيرميس (`/hermes/kb./~`)

التزاماً بقاعدة المصدر الوحيد للحقيقة وانعدام التكرار، تتم مزامنة التوثيق المعماري لنظام موسيف داخل مستودع المعرفة المركزي في `C:\Users\mouss\.hermes\kb\deepagents-litellm-os.md` .

19.1 خريطة المؤشرات عبر النواة الخماسية

تعتمد كافة النوى (K1-Hermes, K2-OpenCode, K3-Codex, K4-Claude, K5-Antigravity) على هذا المستند المرجعي لفهم آليات توجيه المهام واستدعاء بيئة العملاء الأذكياء.

`C:\Users\mouss\.hermes\kb\deepagents-litellm-os.md`

↓↑ (ربط المؤشرات المرجعية بدون تكرار تخزيني)

`C:\Users\mouss\moussaif-bridge\moussaif-deep-agents\`

19.2 المعايير الموثقة في قاعدة المعرفة

- **سجل منافذ البوابات:** التخصيص الرسمي للمنافذ (:4000 لـ LiteLLM، و:8443 لبوابة موسيف، و:11434 لـ Ollama).
- **إجراءات التشغيل القياسية للأدوات (SOP):** معايير الكتابة الذرية للملفات وتنفيذ أوامر الطرفية المعزولة.
- **جداول مسارات الإخفاق البديلة:** المسارات المحددة للتعامل مع انقطاع الخدمات في المشاريع الإنتاجية.

20. النشر الإنتاجي وخارطة التطور المستقبلي

يُمثل نظام موسيف للعملاء الأذكياء نقلة نوعية في هندسة الحوسبة الذاتية المستقلة. يُلخص هذا الفصل الختامي أفضل ممارسات النشر الإنتاجي وخارطة الطريق للتوسع المستقبلي.

20.1 قائمة التحقق للنشر الإنتاجي

1. **التحقق من عمل البوابات في الخلفية:** التأكد من تشغيل خادم LiteLLM كخدمة خلفية دائمة على المنفذ 4000.
2. **إحكام البيئة الافتراضية:** تثبيت إصدارات الحزم في `pyproject.toml` لمنع أي تعارضات مستقبلية.
3. **تدوير مفاتيح الاعتماد:** تفعيل التدوير الآلي لمفاتيح OpenRouter و OpenAI و Anthropic السحابية.
4. **تدوير وتفريغ السجلات:** ضبط سياسة أرشفة السجلات لمدة 14 يوماً في `~/.hermes/logs/` لتوفير المساحة.

20.2 خارطة التطور الهندسي (المراحل 1–3)

المرحلة 1: أسراب الوكلاء المتزامنة

تنفيذ متوازي لفرق الوكلاء عبر مساحات عمل Git المنفصلة لإنجاز مهام الواجهات والخلفيات والجودة في آن واحد.

المرحلة 2: الرؤية البصرية المباشرة

دمج نماذج الرؤية واللغة (Moussaif-VLM) لتدقيق لقطات الشاشة لواجهات المستخدم وفحص التناسق البصري لحظياً.

المرحلة 3: التطور والتعلم الذاتي

التوليد الذاتي للمهارات: يختبر العميل أدوات جديدة في بيئة معزولة، ويكتب ملف `SKILL.md` ويسجله في الكتالوج تلقائياً.

الخاتمة المعمارية

نجح نظام موسيف للعملاء الأذكياء في توحيد استدلال الرسوم البيانية التكرارية، وشبكة البوابات المزدوجة، وحقن المهارات اللحظي ضمن نظام تشغيل سطري متماسك ومستقر للإنتاج الفعلي.