J.A.R.V.I.S.

MEGA-AGENT: Hierarchický To-Do Roadmap

🟠 Fáze 1: Architektonická migrace a základní běh

- [ ] Aktualizace orchestrátorů a služeb na Mega-architecturu
  - [ ] Zkontrolovat README pro zastaralé názvy “Zen orchestrátor”
  - [ ] Upravit všechna docker-compose.yml na nové názvy (mega orchestrátor)
  - [ ] Projít startup skripty, aliasy, porty (přejmenovat z “zen” na “mega”)
  - [ ] Ověřit změny v dokumentaci, poznámkové soubory, .env
- [ ] Zajištění funkčnosti všech core MCP služeb
  - [ ] Ověřit běh každé hlavní MCP služby (docker-compose up, docker ps)
  - [ ] Pro každou MCP zkus GET/POST test endpoint (např. curl na “/health”, “/tools/list”)
  - [ ] Otestovat echo zprávu Rocket.Chat → Mega-Coordinator → retour
  - [ ] Logovat příchozí a odchozí zprávy pro debug
  - [ ] Poznačit chybové Docker logy, řešit postupně
- [ ] Záloha konfigurace a storage
  - [ ] Vygenerovat snapshot všech .env, config, docker-compose.yml
  - [ ] Export PostgreSQL a Redis databází se současným stavem
  - [ ] Kompresovat zálohy a uložit na Box/externí disk
  - [ ] Zapsat datum, umístění zálohy do logu/Notion page

🟡 Fáze 2: Paměťová vrstva a hybridní znalostní graph

- [ ] cldmemory-mcp rozběh, testy (semantic search)
  - [ ] Stáhnout a nainstalovat cldmemory-mcp (zkus docker pull/build)
  - [ ] Připrav Qdrant databázi: docker run nebo compose service
  - [ ] Nastavit .env (porty, API klíče modelu)
  - [ ] Propojit cldmemory-mcp s Mega-Coordinatorem (registry endpoint)
  - [ ] Otestovat semantic search na testovacím dotazu
  - [ ] Ověřit uložení znalostí z konverzace
- [ ] graphrag-mcp + Neo4j – hybridní vyhledávání
  - [ ] Deploy Neo4j přes docker-compose
  - [ ] Zapsat základní schéma (nodes/edges demo data)
  - [ ] Propojit graphrag-mcp s Neo4j a Qdrant
  - [ ] Otestovat hybridní dotaz (semantic + relationship)

- [ ] Spojení persistentní paměti s chatem
  - [ ] Otestovat propojení Rocket.Chat (nebo agent) s cldmemory-mcp
  - [ ] Nahrát historii zpráv do paměti, ověřit search minulých úkolů

🟢 Fáze 3: Kódová metadata – forai-mcp

- [ ] Metadata generování při nové úloze
  - [ ] Rozjet forai-mcp, nastavit ve filesystem-mcp
  - [ ] Ověřit generování metadat pro nové soubory
  - [ ] Integrace s git-mcp pro commit metadata
  - [ ] Export všech metadat do knowledge layer
- [ ] Automatizace search/trace nad projektem
  - [ ] Doplnit chybějící metadata ručně/příkazem
  - [ ] Načíst metadata do graphrag-mcp
  - [ ] Otestovat dotaz “kde se používá funkce X?”

🟣 Fáze 4: LLM server a těžká inference

- [ ] Test inference audio/překlad
  - [ ] Deploy Ollama/vLLM/Whisper
  - [ ] Připrav testovací audio file
  - [ ] Zpracuj request a loguj výstup
- [ ] Integrace s HAS (task orchestrace)
  - [ ] Vytvoř API pro tasky
  - [ ] Ověřit směrování: HAS → LLM server → zpět
  - [ ] Nastavit fallback pro chyby inference
- [ ] Přesměrování heavy task na final server
  - [ ] Nastavit Coordinator proxy routing
  - [ ] Uložit velké výsledky na Box (sdílení)

🟤 Fáze 5: UX, monitoring a bezpečnost

- [ ] Rocket.Chat / Langflow user test
  - [ ] Deploy Langflow, ověř funkce
  - [ ] Vytvoř workflow pro fiktivní úkol
  - [ ] Propojit s Mega-Coordinatorem

- [ ] Monitoring alerty/failover/security
  - [ ] Zkontroluj živost služeb v Prometheus
  - [ ] Simuluj výpadek, ověř alert
  - [ ] Zajisti pravidelné zálohování
  - [ ] Projdi logy, proveď security review