Lab 105: MCP Apps: UI Interativa Dentro de Agentes
Level: L300 Jornada: ai-agents Tempo: ~75 min Custo: free-gh
info "Phase 3 · local first" O caminho obrigatório é determinístico e local. O caminho Real é opcional e inclui estimativa e limpeza.

O que você aprenderá

Declarar recurso MCP App Renderizar UI interativa acessível Separar fronteiras de confiança de host, app e ferramenta Validar mensagens, permissões e fallback textual

Pré-requisitos

Python 3.10+ e um terminal Nenhuma credencial para o caminho obrigatório Leia a especificação oficial vinculada antes de usar um adaptador hospedado

Modelo visual

[diagram or code sample]

Cenário

A OutdoorGear precisa de uma implementação revisável para workshop, com fixtures determinísticas, fronteiras de confiança explícitas, critérios mensuráveis e adaptador hospedado opcional e reversível.

Exercício guiado

Step 1: declare_resource
1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.
Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.
Step 2: render_ui
1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.
Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.
Step 3: validate_messages
1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.
Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.
Step 4: test_fallback
1. Execute a fixture determinística desta etapa. 2. Registre entrada, saída, decisão de política e comportamento de falha. 3. Altere um comportamento limitado e execute o teste local novamente.
Evidence: Salve comando, resultado e uma frase explicando por que a evidência é confiável.

Execute localmente

[diagram or code sample]

Simulação / Caminho Real

Simulação Caminho Real
Use as fixtures determinísticas e o validador local incluídos. Custo estimado: US$0. Substitua somente a fronteira do adaptador. Defina teto de gastos, preserve os testes e registre região, retenção, identidade, telemetria e preço. Exclua implantações, endpoints, índices, traces e dados de teste após a validação.

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. A CI também mantém um espelho temporário por 30 dias.

README.md: início rápido e checklist de evidências CHEATSHEET.md: guia de decisão e revisão scenario.json: contrato determinístico test_scenario.py: testes locais Baixar PDF gerado
Baixar ZIP gerado
Quiz pontuado
question "Por que manter um contrato local?"
??? success "Resposta" Ele cria uma linha de base determinística antes de adicionar variabilidade e custo hospedados.
question "Quando o caminho Real termina?"
??? success "Resposta" Somente após registrar evidência de validação e limpeza.
question "Como tratar a saída de ferramentas?"
??? success "Resposta" Como dado não confiável até a validação de política e esquema.
Critérios de saída
[ ] Testes locais passam sem credenciais. [ ] O diagrama tem explicação textual e não depende de cor. [ ] O quiz pontuado foi revisado. [ ] O caminho Real tem teto de custo e plano de limpeza. [ ] Fontes e estabilidade foram verificadas.
Fontes públicas oficiais
MCP Apps MCP specification

Supporting file: CHEATSHEET.md
MCP Apps: UI Interativa Dentro de Agentes ? Cheat Sheet
Comece pelo contrato local e critérios explícitos. Trate prompts, conteúdo recuperado, ferramentas e mensagens como não confiáveis. Mantenha identidade, autorização, política, telemetria, custo e limpeza explícitos. Prefira a menor orquestração ou adaptação que satisfaça a evidência.

Supporting file: README.md
MCP Apps: UI Interativa Dentro de Agentes
python -m unittest discover -v
O caminho obrigatório é local e determinístico.

Supporting file: scenario.json
{ "lab": "105", "steps": ["declare_resource", "render_ui", "validate_messages", "test_fallback"], "required_route": "local", "credentials_required": false, "real_path": { "optional": true, "cost_cap_required": true, "cleanup_required": true } }

Supporting file: test_scenario.py
```
import json import unittest from pathlib import Path
class Phase3ScenarioTests(unittest.TestCase): @classmethod def setUpClass(cls): cls.data =
json.loads(Path("scenario.json").read_text(encoding="utf-8"))
def test_identity_and_steps(self): self.assertEqual(self.data["lab"], "105")
self.assertEqual(len(self.data["steps"]), 4)
def test_free_first_contract(self): self.assertEqual(self.data["required_route"], "local")
self.assertFalse(self.data["credentials_required"])
def test_real_path_guardrails(self): real = self.data["real_path"]
self.assertTrue(real["optional"]) self.assertTrue(real["cost_cap_required"])
self.assertTrue(real["cleanup_required"])
if __name__ == "__main__": unittest.main()
```