Parte 3 — Ollama & Open WebUI

````
# Parte 3 — Ollama: instalação, modelos, Open WebUI e exemplos

## 8. Instalação & verificação
- Baixe: https://ollama.com/download
- Verifique:
```bash
ollama --version
ollama list
```

## 9. Comandos básicos
```bash
ollama pull llama3:8b
ollama run llama3:8b "Explique programação assíncrona em Python"
--temperature 0.2
ollama serve --port 11434 &
ollama list
```

## 10. Modelos recomendados (por caso de uso)
- **Chat**: `llama3:8b`, `llama3.1:8b`
- **Raciocínio**: `phi3`
- **Código/Dev**: `deepseek-coder-v2:16b`, `starcoder2:7b`,
`codellama-34b`
- **Visão**: `llava`, `qwen2-vl`
- **Embeddings**: `nomic-embed-text`, `text-embedding-3-large`

## 11. Exemplos práticos
**Terminal**
```bash
ollama run deepseek-coder-v2:16b "Refatore esta função Python para ser
mais eficiente: <código>"
```

**Python (cliente)**
```python
from ollama import Client
client = Client(host='http://localhost:11434')
resp = client.chat(model='llama3:8b',
messages=[{'role':'user','content':'Explique a regresão linear.'}])
print(resp['message']['content'])
```

## 12. Usando Ollama com Open WebUI (integração)
**Visão geral**: Ollama expõe API em `http://localhost:11434`. Open WebUI
é uma UI que pode consumir essa API.

**Pré-requisitos**
- Ollama + modelos instalados
- Docker ou instalação do Open WebUI
- Porta 11434 acessível

**Passos rápidos (Docker)**
```bash
ollama serve --port 11434 &
ollama list
````
````

```
docker run -d --name open-webui -p 3000:3000 open-webui/open-webui:ollama
```
Acesse `http://localhost:3000` e configure endpoint
`http://localhost:11434` se necessário.

Configurações & troubleshooting
- Verifique `ss -ltnp | grep 11434` se não conectar.
- Use proxy reverso (NGINX) para TLS em servidores públicos.
- Em Open WebUI, ajuste parâmetros do modelo (temperature, max_tokens, etc.) via Settings.

Exemplo prático (RAG)
1. Indexe documentos na Open WebUI (upload / apontar para indexador).
2. Selecione modelo Ollama e crie prompt que inclua contexto recuperado.

Mini■tutorial: emitir certificado TLS automaticamente (resumo rápido)

Eu gerei um conjunto de arquivos prontos para deploy com TLS automático por Let's Encrypt:

- `docker-compose-tls.yml` — compose com serviços: `open-webui`, `nginx` e `certbot`.
- `nginx-tls.conf` — configuração nginx preparada para ACME challenge. **Substitua** `SERVER_NAME_PLACEHOLDER` pelo seu domínio.
- `README_deploy_TLS.md` — instruções completas passo■a■passo.
- `issue-certs.sh` — script que automatiza a emissão inicial do certificado e recarrega o NGINX.

Passos rápidos (resumo):

1. Edite `nginx-tls.conf`, trocando `SERVER_NAME_PLACEHOLDER` pelo seu domínio (ex.: `minhadomain.com`).
2. Crie diretórios persistentes:
   ```bash
   mkdir -p ./letsencrypt ./html
   ```
3. Suba os serviços essenciais (Open WebUI + NGINX):
   ```bash
   docker compose -f docker-compose-tls.yml up -d open-webui nginx
   ```
4. Emita o certificado (use o script gerado):
   ```bash
   ./issue-certs.sh minhadomain.com seu@email.com
   ```
   O script irá:
   - substituir o placeholder no nginx.conf (local),
   - iniciar containers necessários,
   - rodar o `certbot` no container para gerar o certificado,
   - recarregar o NGINX para usar o certificado.
5. Teste `https://minhadomain.com` no navegador.

Observações:
- Se seu Docker não reconhecer `host.docker.internal`, ajuste `OLLAMA_API_URL` no `docker-compose-tls.yml` para apontar ao IP do host.
- Depois da emissão inicial, agende renovação periódica (ex.: `certbot renew` via cron). Veja `README_deploy_TLS.md` para detalhes.
