ChatGPT

**Manuale di Indicizzazione e Stima dei Costi per la Generazione e l'Aggiornamento del Manuale di Progetto**

---

# 1. Obiettivo del Documento

Questo manuale descrive:

- Processo di indicizzazione di un repository (C#, HTML, JS, CSS, configurazioni)
- Setup ambientale e script di esempio per chunking ed embeddings
- Opzioni di vector store con piani e costi
- Stima dettagliata dei costi API OpenAI
- Modalità di aggiornamento continuo (re-indicizzazione e fine-tuning)
- CI/CD e best practice di integrazione

# 2. Requisiti Tecnici

## 2.1 Ambiente di Sviluppo

- Python ≥ 3.8 con virtual environment ( `venv` o `conda` )
- Editor/IDE (VSCode, PyCharm)
- Connessione Internet per API OpenAI

## 2.2 Dipendenze (gratuiti)

```
pip install openai llama-index tqdm python-dotenv chromadb pinecone-client
```

- **openai**: client API
- **llama-index**: parsing, chunking, indicizzazione
- **tqdm**: barre di avanzamento
- **python-dotenv**: gestione `.env`
- **chromadb**: vector store locale
- **pinecone-client**: client Pinecone (opzionale)

## 2.3 Configurazione `.env`

```
# OpenAI API Key\OPENAI_API_KEY=sk-...

# Modelli OpenAI
OPENAI_EMBEDDING_MODEL=text-embedding-3-small
OPENAI_COMPLETION_MODEL=gpt-4o-mini

# Vector Store
VECTOR_STORE_TYPE=chroma          # o 'pinecone'
VECTOR_STORE_PATH=./repo_index.json
```

```
# Pinecone (se usato)
PINECONE_API_KEY=...
PINECONE_ENV=us-west1-gcp
PINECONE_INDEX=repo-index

# Chunking
CHUNK_SIZE=800
CHUNK_OVERLAP=100
```

### 2.4 Requisiti di Sistema

- RAM: 4–8 GB
- Spazio Disco: 100 MB–1 GB
- Monitoraggio dei rate limits API

## 3. Workflow di Indicizzazione

### 3.1 Preparazione del Codice

```
# Clonare e pulire
git clone git@github.com:utente/repo.git tuo-repo-pulito
cd tuo-repo-pulito
# Rimuovi build, dipendenze e metadati IDE/VCS
echo "node_modules bin obj dist .git .vs .vscode" | xargs rm -rf
```

- Includi: `*.cs`, `*.js`, `*.html`, `*.css`, `*.json`, `*.yml`, `*.yaml`
- Escludi file >5 MB e cartelle temporanee

### 3.2 Script di Indicizzazione (index_repo.py)

```
import os
glob
from dotenv import load_dotenv
from llama_index import SimpleDirectoryReader, GPTVectorStoreIndex
from llama_index.embeddings import OpenAIEmbedding

# Carica ambiente
load_dotenv()

reader = SimpleDirectoryReader(
    './tuo-repo-pulito',
    include=['*.cs','*.js','*.html','*.css','*.json','*.yml'],
    exclude_dirs=['node_modules','bin','obj','dist'],
    chunk_size=int(os.getenv('CHUNK_SIZE',800)),
    chunk_overlap=int(os.getenv('CHUNK_OVERLAP',100))
)
docs = reader.load_data()

index = GPTVectorStoreIndex.from_documents(
```

```
    docs,
    embedding=OpenAIEmbedding(model=os.getenv('OPENAI_EMBEDDING_MODEL'))
)
index.save_to_disk(os.getenv('VECTOR_STORE_PATH'))
print(f"Indice creato con {len(docs)} chunk.")
```

### 3.3 Querying e Generazione RAG

```
def genera_paragrafo(query, top_k=5):
    res = index.query(query, similarity_top_k=top_k)
    return openai.ChatCompletion.create(
        model=os.getenv('OPENAI_COMPLETION_MODEL'),
        messages=[{'role':'user', 'content':f"Usa questi chunk: {res} per
generare un paragrafo su: {query}"}]
    )
```

## 4. Opzioni di Vector Store e Prezzi

Per questo progetto, supponiamo:

- uso locale durante la prototipazione con **ChromaDB** (gratuito)
- passaggio a **Pinecone Starter** in produzione (1 vCPU, 2 GB RAM)

| Store | Piano scelto | Costo base/ mese | Costo ingest/ stor. | Esempio | Note |
|---|---|---|---|---|---|
| **ChromaDB** | — | \$0 | \$0 | — | Vector store locale gratuito |
| **Pinecone** | Starter (1 pod) | \$5.00 | \$0.25 per GB-mese | 2 GB = \$0.50 | Totale stimato: \$5.50/mese |
| Weaviate | Cloud Starter | \~\$5–10 | Variabile | — | Cloud con opzione self-hosted |
| Milvus | Open Source | \$0 (hosting escluso) | Variabile | VPS ≈ \$4–10/mese | Richiede installazione manuale |

> Esempio Pinecone: un progetto medio con 1–2 GB di embedding (3 000–5 000 chunk) e 720 ore di uptime costa circa **\$5.50/mese**. Nota: i costi aumentano linearmente con file, aggiornamenti e richieste.

## 5. Costi API OpenAI

| Servizio | Prezzo unitario | Consumo test | Costo stimato (USD) |
|---|---|---|---|
| Embeddings | \$0.0001 per 1 000 token | 1 000 richieste ×800 tok | \$0.08 |
| Completion GPT-3.5 | \$0.002 per 1 000 token | 50×500 tok | \$0.05 |

| Servizio | Prezzo unitario | Consumo test | Costo stimato (USD) |
|---|---|---|---|
| Completion GPT-4 | \$0.03 per 1 000 token | 50×500 tok | \$0.75 |
| Fine-tuning (training) | \$0.03 per 1 000 token | 10 000 tok | \$0.30 |

**Totale stimato test**: **\~\$1.18**

## 6. Aggiornamento Continuo e Fine-tuning

- **Re-indicizzazione**: su ogni push ricostruisci l'indice (cache chunk invariati)
- **Fine-tuning incrementale**:
- Raccogli correzioni in `train_data.jsonl`
- Esegui:

```
openai api fine_tunes.create \
  --training_file train_data.jsonl \
  --model gpt-3.5-turbo
```

- Modello custom per risposte migliori
- **Feedback loop**: form degli utenti integrato nel sito doc

## 7. CI/CD e Automazione

- **GitHub Actions** (2 000 min free/mese)
- Esempio workflow:

```
on: [push]
jobs:
  build-docs:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-python@v4
        with: python-version: '3.9'
      - run: pip install -r requirements.txt
      - run: python index_repo.py
      - run: mkdocs build
      - uses: peaceiris/actions-gh-pages@v3
        with:
          publish_dir: ./site
```

## 8. Best Practice e Ottimizzazioni

- Caching embeddings per chunk non modificati
- Monitoraggio costi OpenAI e Pinecone
- Documentazione versionata in branch dedicato
- Metriche di copertura: percentuale di file indicizzati
- Alert automatici su rate limit/API errors

---

*Fine del manuale approfondito e completo.*