🖼️ Ottimizzazione Thumbnail

🎯 Pattern Scelto: Proxy

📝 Descrizione del Problema

Il sistema deve gestire le immagini profilo degli utenti con le seguenti necessità:

- Ottimizzare il trasferimento delle immagini riducendone la risoluzione
- Implementare un sistema di cache temporanea (20 minuti)
- Generare thumbnail al primo accesso per utenti esistenti
- Generare thumbnail all'upload per nuovi utenti

🔧 Pattern utilizzato: Proxy

🧠 Ragionamento

Il Pattern Proxy è la scelta ideale per questo scenario per diversi motivi:

1️⃣ Controllo degli Accessi

- Gestisce l'accesso alle immagini originali
- Implementa la logica di caching
- Controlla la generazione delle thumbnail

2️⃣ Lazy Loading

- Genera le thumbnail solo quando necessario
- Ottimizza le risorse del sistema
- Riduce il carico iniziale

3️⃣ Caching

- Memorizza temporaneamente le immagini accedute di frequente
- Riduce il numero di accessi allo storage
- Migliora le performance del sistema

✅ Vantaggi

1. **Separazione delle Responsabilità**

- Ogni proxy ha un compito specifico
- Codice più organizzato e manutenibile

2. **Ottimizzazione delle Risorse**
   - Riduzione del traffico di rete
   - Minore utilizzo dello storage
   - Migliori performance
3. **Flessibilità**
   - Facile aggiungere nuove funzionalità
   - Semplice modifica dei comportamenti esistenti

❌ Svantaggi

- Maggiore complessità iniziale

🚀 Esempio di Utilizzo

```
# Esempio base di utilizzo
real_service = RealImageService(storage_path)
cache_proxy = CacheImageProxy(real_service)
thumbnail_proxy = ThumbnailProxy(cache_proxy)

# Richiesta immagine
image = thumbnail_proxy.get_image("user_123_profile.jpg")
```

📈 Conclusioni

Il Pattern Proxy fornisce una soluzione elegante e efficiente per:

- Ottimizzare le risorse
- Migliorare le performance
- Mantenere il codice organizzato
- Gestire in modo trasparente la complessità