



Ottimizzazione Thumbnail con 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



Perché il Pattern Proxy?

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

1 Controllo degli Accessi

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



2 Lazy Loading

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

3 Caching

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

Componenti Principali:

1.  **ImageService** (Interface)
 - Definisce il contratto base per tutti i servizi
2.  **RealImageService**
 - Gestisce l'accesso diretto allo storage

- Recupera le immagini originali

3. **CacheImageProxy**

- Implementa la cache temporanea
- Gestisce la scadenza delle immagini
- Pulisce automaticamente la cache

4. **ThumbnailProxy**

- Gestisce la creazione delle thumbnail
- Memorizza le versioni ridotte
- Ottimizza il trasferimento

Vantaggi della Soluzione

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

Come Utilizzare il pattern

```
# Inizializzazione
real_service = RealImageService(storage_path)
cache_proxy = CacheImageProxy(real_service)
thumbnail_proxy = ThumbnailProxy(cache_proxy)

# Utilizzo
image_id = "user_123_profile.jpg"
thumbnail = thumbnail_proxy.get_image(image_id)
```

Flusso di Esecuzione

1. Il client richiede un'immagine
2. ThumbnailProxy verifica se esiste una thumbnail
3. Se non esiste, richiede l'immagine originale tramite CacheProxy





4. CacheProxy verifica la cache
5. Se non in cache, recupera da ReallImageService
6. L'immagine viene processata e restituita al client

Test e Verifica

Per testare il sistema:

```
python main.py
```

Il test verifica:

-  Creazione iniziale thumbnail
-  Funzionamento della cache
-  Gestione multiple richieste
-  Corretto ridimensionamento

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