Indice

|  |
| --- |
|  |

11 – Progettazione architetturale 2

11.1 – Viste architetturali 4

11.2 – Schemi architetturali 5

11.2.1 – Architettura a strati 6

11.2.2 – Architettura repository 8

11.2.3 – Architettura client-server 9

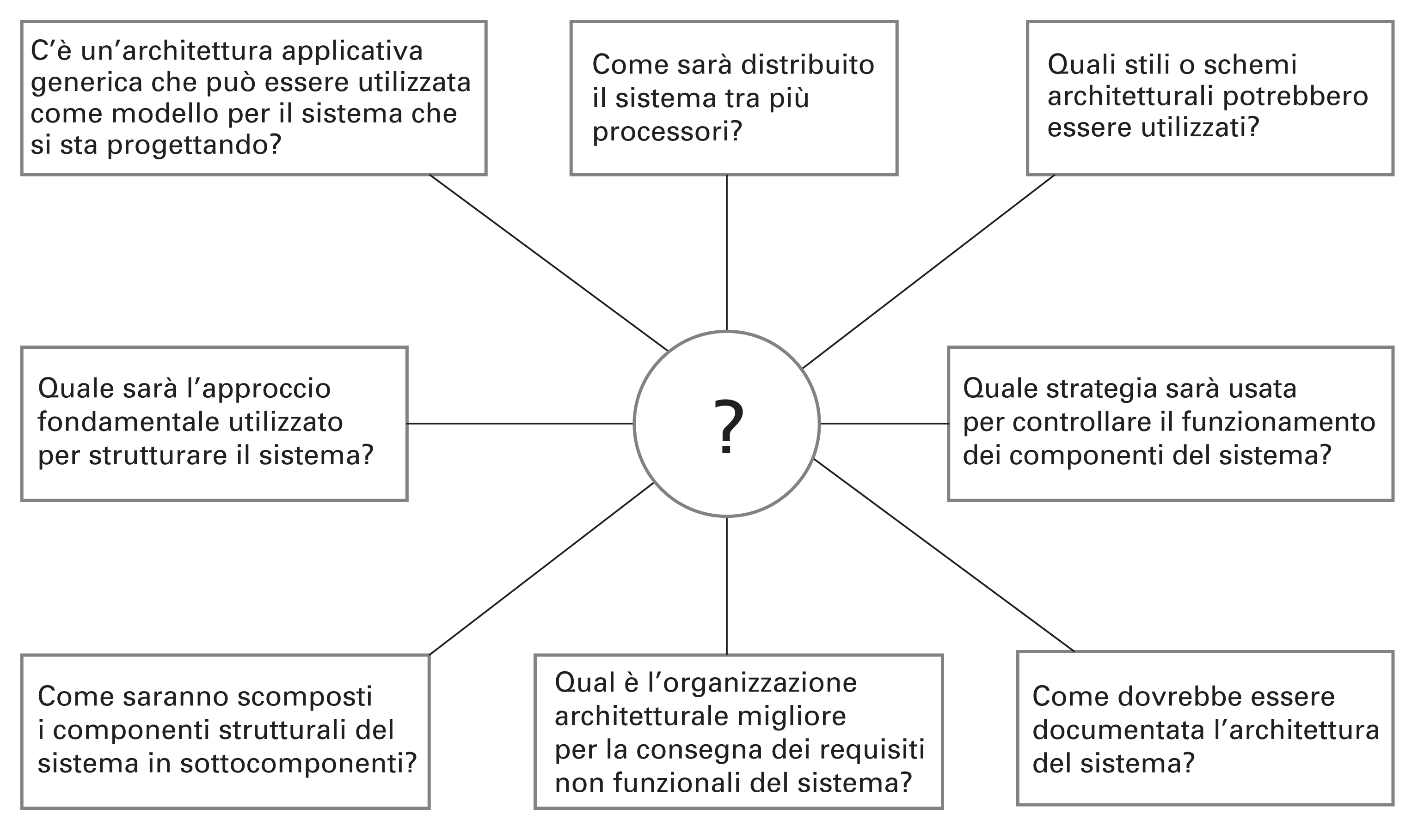
11.2.4 – Architettura pipe-and-filter 11

11 – Progettazione architetturale

|  |
| --- |
|  |

La **progettazione architetturale** è un processo creativo in cui si progetta un’organizzazione che soddisfa i requisiti funzionali e non funzionali di un sistema.

**Durante il processo di progettazione architetturale** gli architetti del sistema devono prendere una serie di decisioni fondamentali. Essi devono rispondere ad alcune domande fondamentali, riportate in figura.



L’architettura di un sistema software può basarsi su un particolare **schema** o **stile architetturale**. Uno **schema architetturale** è una descrizione dell’organizzazione del sistema, per esempio un’organizzazione client-server o un’architettura a strati. Gli schemi architetturali esprimono l’essenza dell’architettura che è stata utilizzata in diversi sistemi software.

A causa della stretta relazione tra le caratteristiche non funzionali del sistema e l’architettura del software, la scelta dello schema architetturale e della struttura dipende dai requisiti non funzionali del sistema:

1. **Prestazioni**: se le prestazioni sono un requisito critico, l’architettura dovrebbe essere progettata per localizzare le operazioni critiche all’interno di un piccolo numero di componenti, con questi componenti installati sullo stesso computer, anziché distribuiti nella rete.
2. **Protezione**: se la protezione è un requisito critico, si utilizza un’architettura strutturata a strati, con le risorse più critiche protette nello strato più interno, e con un altro livello di convalida della protezione a ogni livello.
3. **Sicurezza**: se la sicurezza è un requisito critico, l’architettura è progettata in modo che le operazioni relative alla sicurezza siano tutte collegate in un singolo componente o in un piccolo numero di componenti.
4. Disponibilità: se la disponibilità è un requisito critico, l’architettura è progettata per includere i componenti ridondanti, in modo che sia possibile sostituirli e aggiornarli senza fermare il sistema.
5. **Mantenibilità**: se la mantenibilità è un requisito critico, l’architettura del sistema dovrebbe essere progettata utilizzando componenti piccoli e autonomi che possono essere modificati velocemente.

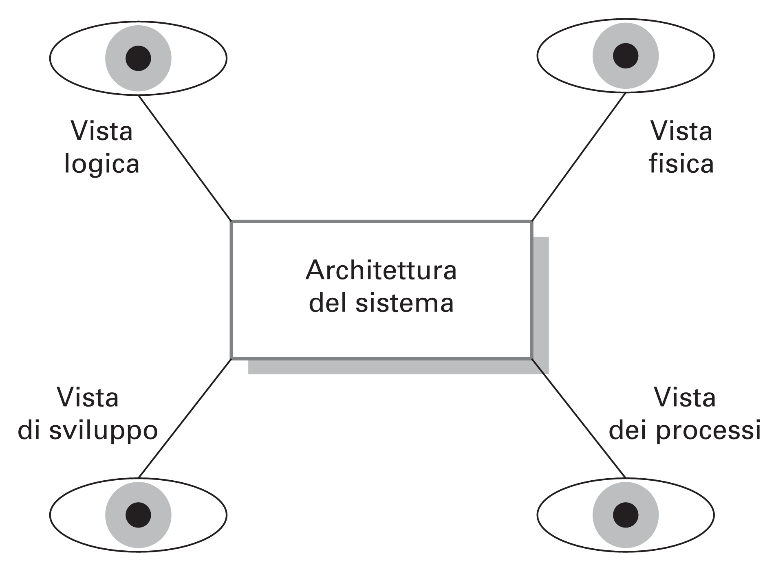
|  |
| --- |
|  |

11.1 – Viste architetturali

È impossibile rappresentare tutte le informazioni relative all’architettura di un sistema in un singolo diagramma, in quanto un modello grafico può mostrare soltanto una vista o prospettiva del sistema.

Nonostante esistano diverse opinioni su quali viste siano richieste, Krutchen nel suo famoso modello a 4+1 viste dell’architettura software, suggerisce quattro viste architetturali fondamentali che possono essere collegate tramite scenari o casi d’uso comuni (come in figura):

1. **Vista logica**: mostra le astrazioni chiave nel sistema come oggetti o classi di oggetti. Essa mette in relazione i requisiti del sistema con le entità.
2. **Vista dei processi**: mostra come, a runtime, il sistema è composto processi interattivi.
3. **Vista di sviluppo**: mostra come il software viene scomposto per lo sviluppo; ovvero mostra la suddivisione del software nei suoi componenti che sono implementati da un singolo sviluppatore o da un team di sviluppatori.
4. **Vista fisica**: mostra l’hardware del sistema e come i componenti del software sono distribuiti tra i processi nel sistema.

****

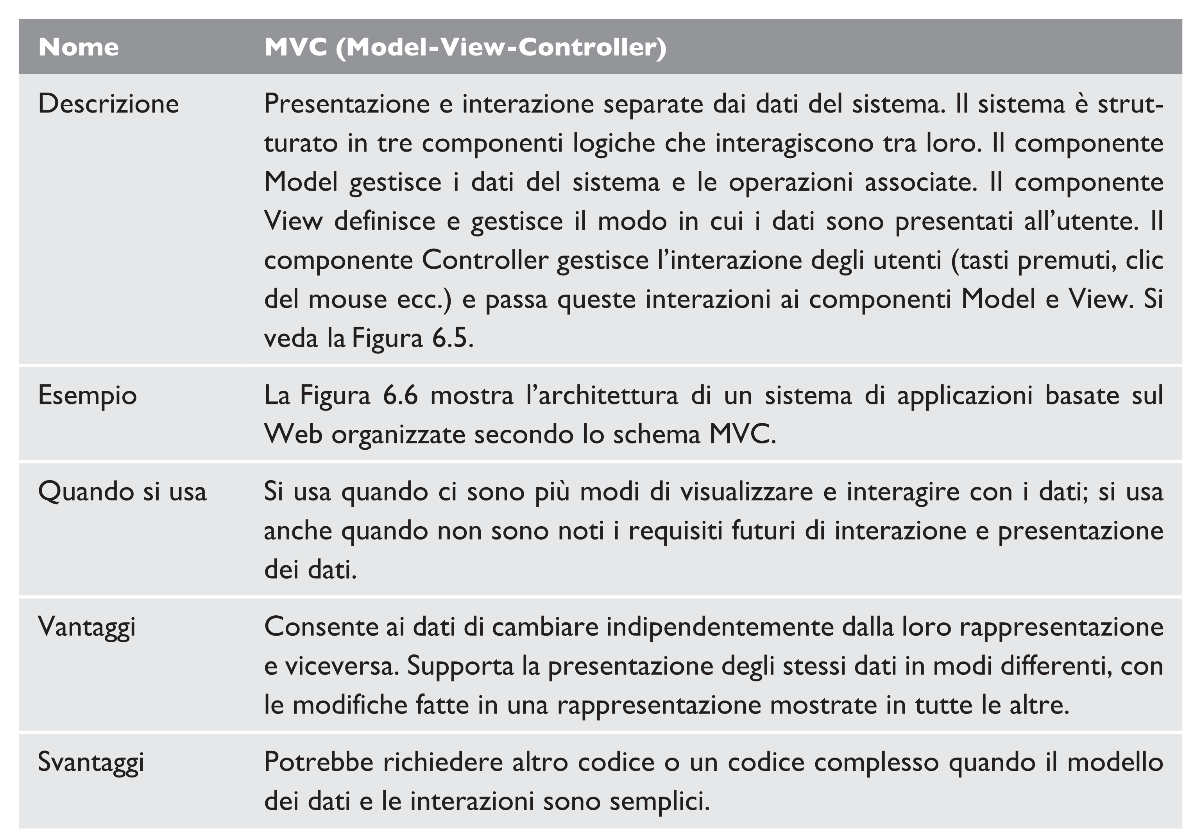
Quindi, le **viste concettuali dell’architettura di un sistema sono quasi sempre sviluppate durante la fase di progettazione**. Sono **utilizzate per spiegare l’architettura del sistema agli stakeholder e per rendere note le decisioni di progettazione architetturale**.

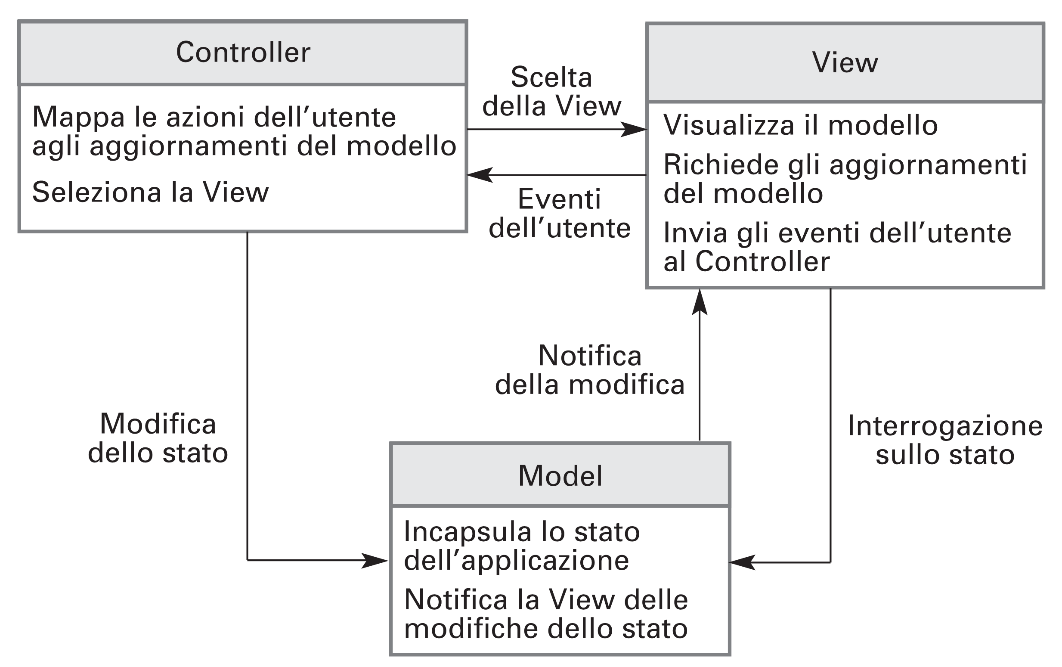
|  |
| --- |
|  |

11.2 – Schemi architetturali

In questo paragrafo si presentano gli schemi architetturali e se ne descrive brevemente alcuni di quelli più utilizzati.

Gli **schemi architetturali** possono essere **utilizzati** in un modo standard utilizzando un mix di testi e diagrammi come nelle seguenti figure:





Uno schema architetturale dovrebbe descrivere l’organizzazione di un sistema che ha avuto successo in altri sistemi preesistenti; dovrebbe includere le informazioni su quando il suo utilizzo è appropriato e i dettagli sui suoi vantaggi e svantaggi.

La tabella nella pagina precedente, descrive il noto schema **Model-View-Controller (MVC)**, che è la base per la gestione delle interazioni in molti sistemi web.

|  |
| --- |
|  |

11.2.1 – Architettura a strati

I concetti di separazione e indipendenza sono fondamentali nella progettazione architetturale, in quanto consentono di localizzare le modifiche.

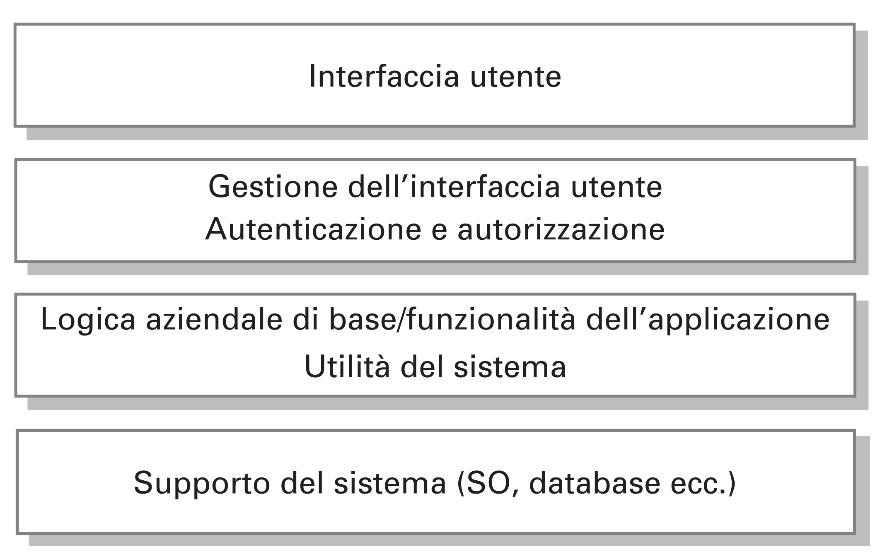
Lo schema di architettura a strati è un altro modo di **ottenere la separazione e l’indipendenza**. Questo schema è illustrato in figura, dove la funzionalità del sistema sono organizzate in strati separati, ciascuno dei quali si basa sulle funzioni e sui servizi offerti dallo strato immediatamente sottostante.

Immagine che contiene testo

Descrizione generata automaticamente

Questo approccio a strati **supporta** lo **sviluppo incrementale** dei sistemi. Infatti, quando viene sviluppato uno strato, alcuni dei servizi da esso forniti sono resi accessibili agli utenti. Quest’**architettura** è anche **modificabile** e **portabile**.

In figura un esempio di architettura a strati con quattro strati.



Lo strato più basso include il software di supporto del sistema.

Lo strato successivo è quello dell’applicazione.

Il terzo strato riguarda la gestione dell’interfaccia utente e le funzioni di autenticazioni e l’autorizzazione degli utenti.

Lo strato superiore fornisce informazioni per l’interfaccia utente.

|  |
| --- |
|  |

11.2.2 – Architettura repository

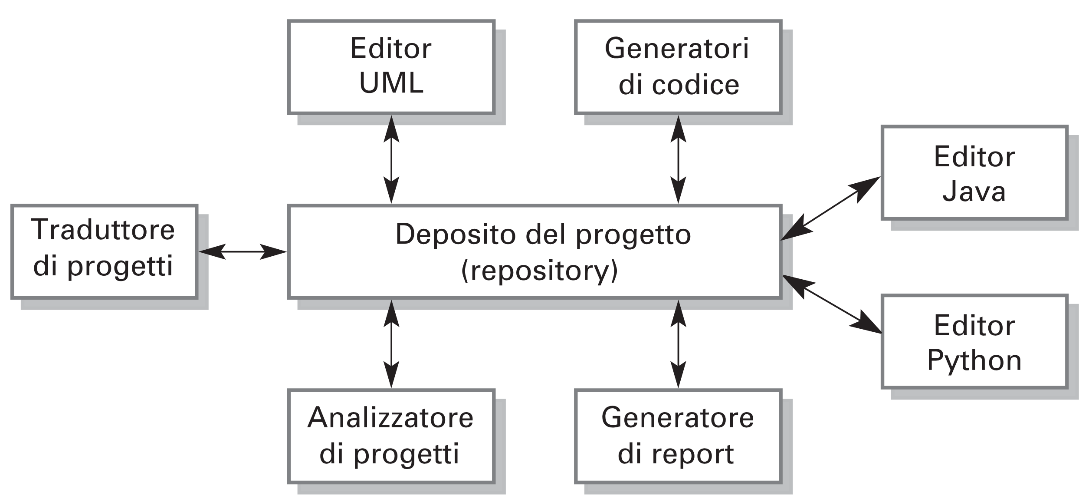
Lo **schema repository**, come in figura, spiega come una serie di componenti interattivi possono condividere i dati.

Immagine che contiene testo

Descrizione generata automaticamente

Questo modello è adatto alle applicazioni nelle quali i dati sono generati da un componente e utilizzati da un altro componente. Esempi di questo tipo di sistemi sono i sistemi di comando e controllo, i sistemi di gestione delle informazioni, i sistemi CAD (Computer Aided Design) e gli ambienti di sviluppo interattivo per il software.

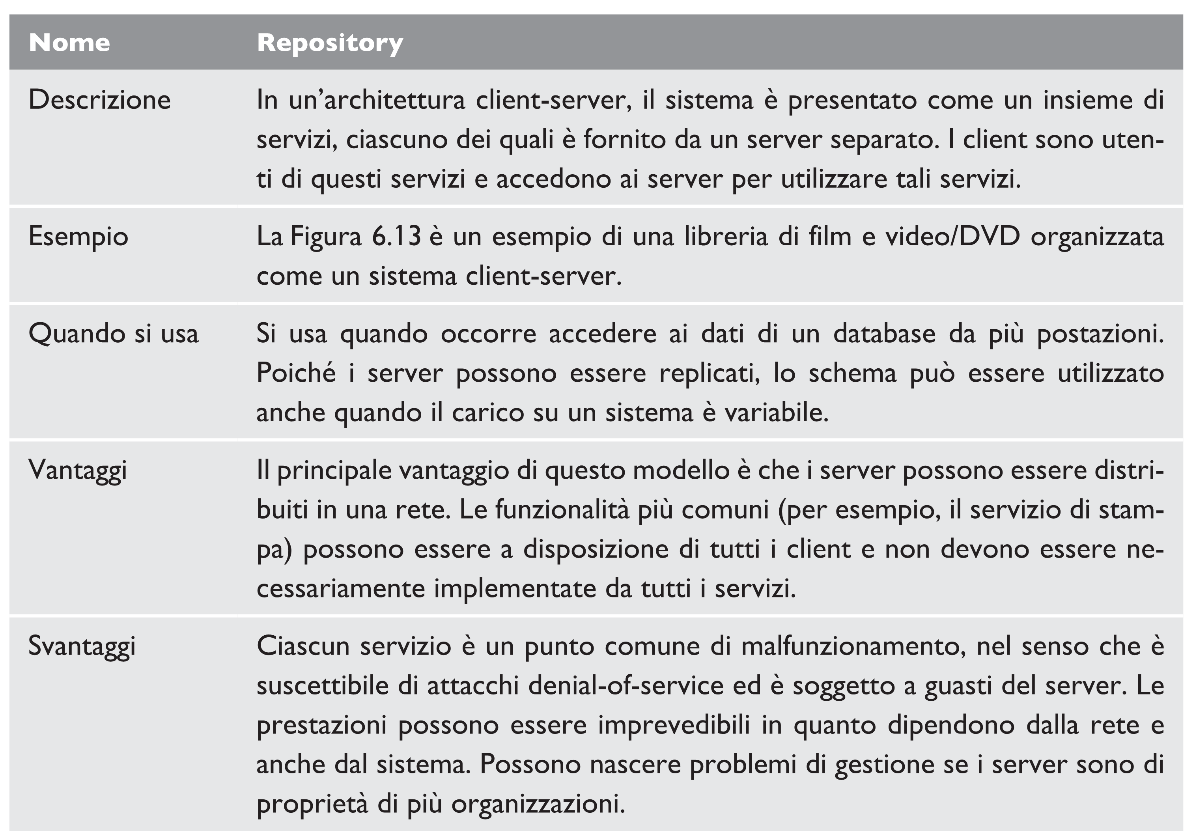
Invece, la figura sottostante mostra una situazione in cui può essere usato un repository.



|  |
| --- |
|  |

11.2.3 – Architettura client-server

Lo **schema client-server**, come in figura, illustra una tipica organizzazione a runtime di sistemi distribuiti.

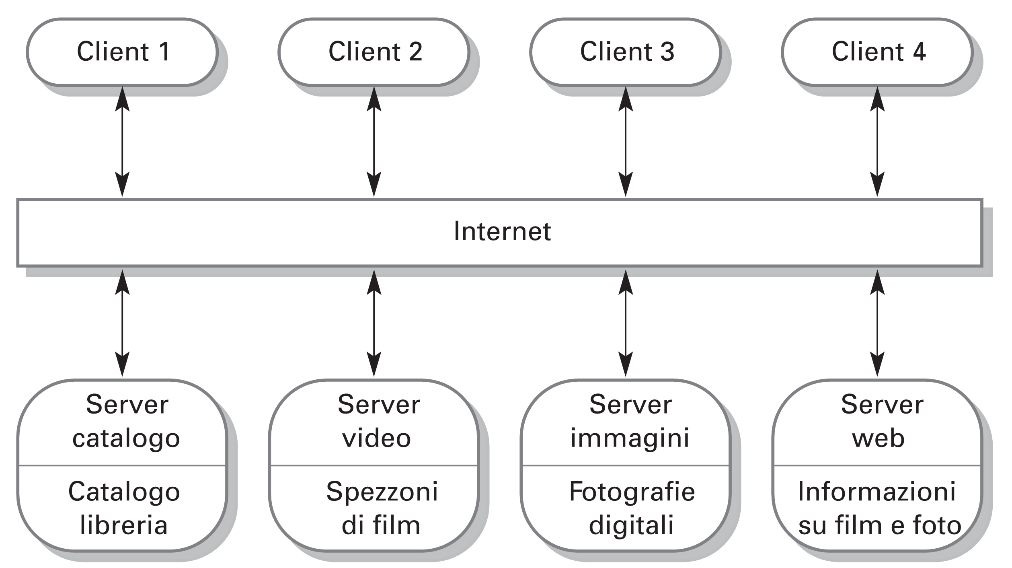


Un sistema conforme allo schema client-server è organizzato come un insieme di servizi e server associati e di client che accedono e usano tali servizi. I principali **componenti di questo modello** sono:

1. Un insieme di server che offrono servizi ad altri sottosistemi.
2. Un insieme di client che richiedono i servizi offerti dai server.
3. Una rete che permette ai client di accedere a questi servizi.

Ancora una volta, la separazione e l’indipendenza sono un importante vantaggio. I servizi e i server possono essere modificati senza influire su altre parti del sistema.

La seguente figura mostra l’esempio di un sistema basato sul modello client-server: si tratta di un sistema multiutente, basato sul Web, che fornisce una libreria di film e fotografie.



Il **vantaggio** più importante del modello client-server è che si tratta di un’architettura distribuita.

|  |
| --- |
|  |

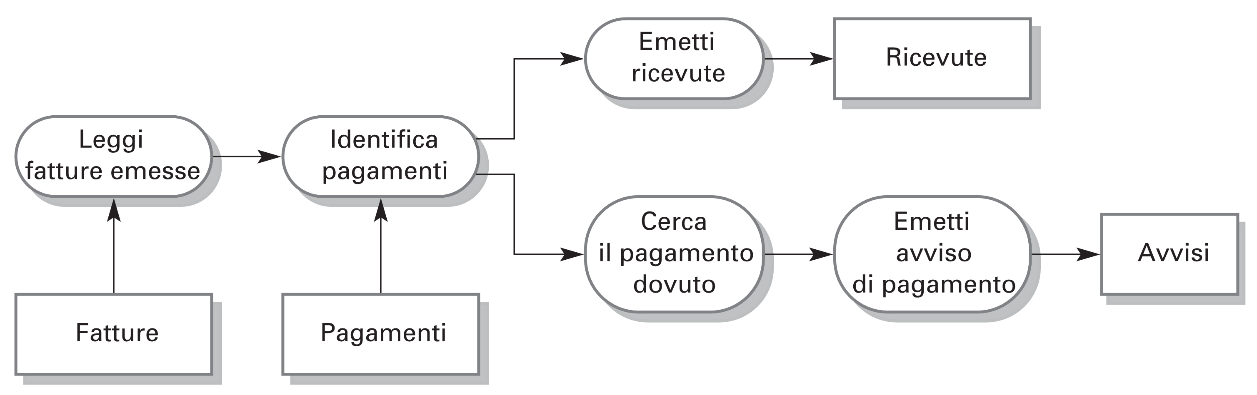
11.2.4 – Architettura pipe-and-filter

Lo **schema pipe-and-filter**, come in figura, rappresenta l’organizzazione a runtime di un sistema dove le trasformazioni funzionali elaborano i loro input e generano output.

Immagine che contiene testo

Descrizione generata automaticamente

Un esempio di questo tipo di architettura di sistema, usato in un’applicazione di elaborazione batch, è illustrato nella figura seguente:



Gli schemi pipe-and-filter sono **particolarmente adatti** ai sistemi di elaborazione batch e ai sistemi integrati, dove l’interazione con gli utenti è limitata.