# Breve documentazione sull'architettura del sistema

L'architettura scelta per la realizzazione del software rispetta il modello a strati, in particolare i livelli considerati sono tre: livello dati, implementato attraverso un database relazionale Postgres; livello logico, che comprende l'insieme delle classi e il relativo funzionamento e infine il livello utente, costituito da un'interfaccia user-friendly realizzata in Java attraverso l'utilizzo delle suite AWT e Swing.

Business logic trattata in modo esplicito:

– livello 1: gestione dei dati (DBMS, file XML, …..)

– livello 2: business logic (elaborazione dati, …)

– livello 3: interfaccia utente (presentazione dati, servizi)

Ogni livello ha obiettivi e vincoli di design propri. Nessun livello fa assunzioni sulla struttura o implementazione degli altri.

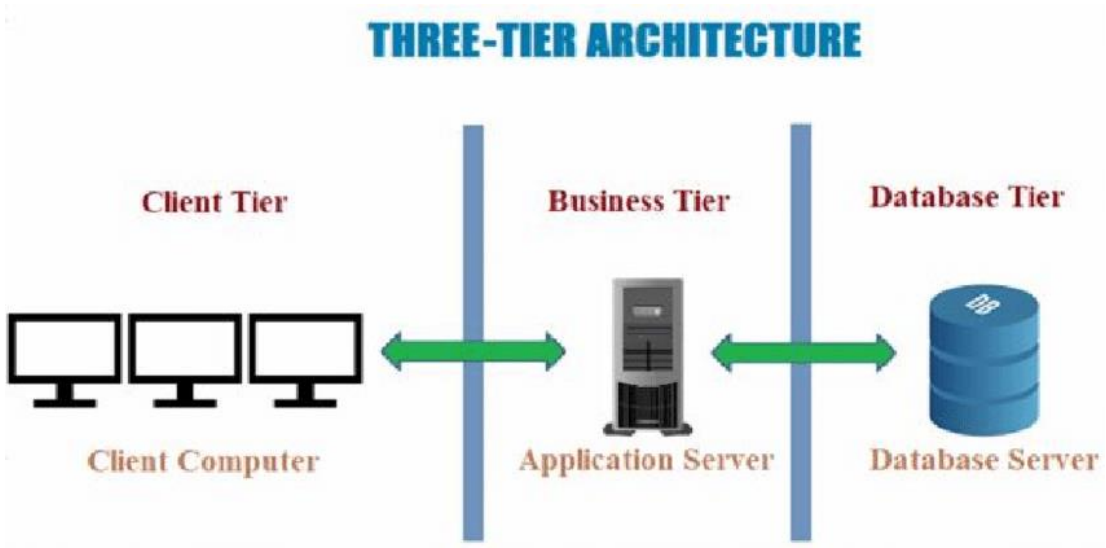
Non c'è comunicazione diretta tra livello 1 e livello 3:

– Interfaccia utente non riceve, né inserisce direttamente dati nel livello di data management

– Tutti i passaggi di informazione nei due sensi vengono filtrati dalla business logic

I livelli operano senza assumere di essere parte di una specifica applicazione

- applicazioni viste come collezioni di componenti cooperanti
- ogni componente può essere contemporaneamente parte di applicazioni diverse (e.g., database, o componente logica di configurazione di oggetti complessi)



## VANTAGGI

- Flessibilità e modificabilità di sistemi formati da componenti separate:
  - componenti utilizzabili in sistemi diversi
  - modifica di una componente non impatta sul resto del sistema (a meno di cambiamenti nelle API)
  - ricerca di bug più focalizzata (separazione ed isolamento delle funzionalità del sistema)
  - aggiunta di funzionalità all'applicazione implica estensione delle sole componenti coinvolte (o aggiunta di nuove componenti)
- Interconnettività
  - API delle componenti superano il problema degli adattatori del modello client server: N interfacce diverse possono essere connesse allo stesso servizio etc.
  - Facilitato l'accesso a dati comuni da parte di applicazioni diverse (uso dello stesso gestore dei dati da parte di business logics diverse)
- Gestione di sistemi distribuiti
  - Business logic di applicazioni distribuite (e.g., sistemi informativi con alcuni server replicati e client remoti) aggiornabile senza richiedere aggiornamento dei client

## SVANTAGGI

- Dimensioni delle applicazioni ed efficienza
  - Pesante uso della comunicazione in rete e latenza del servizio
  - Comunicazione tra componenti richiede uso di librerie SW per scambio di informazioni $\Rightarrow$ codice voluminoso