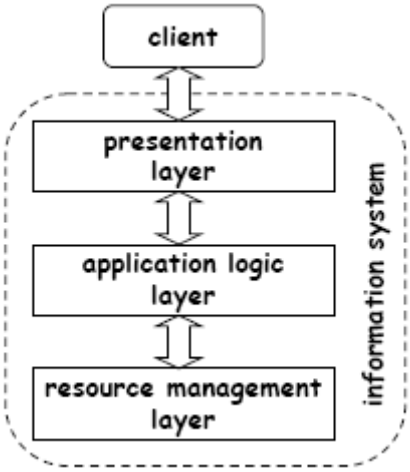
Layer di un Sistema Informativo



A **livello concettuale** i sistemi informativi sono progettati in termini di tre diverse componenti funzionali (**livelli**)

1. Presentazione
2. Logica dell'applicazione
3. Gestione delle risorse

Presentation Layer

- E' il livello del sistema che gestisce la **comunicazione** con le **entità esterne** al sistema stesso (***client***)
- Comprende le componenti che si occupano di presentare l'informazione verso i client, e che consentono ai client di interagire con il sistema per sottomettere operazioni ed ottenere risultati
- I client possono essere completamente esterni ed indipendenti dal presentation layer: nei sistemi accessibili tramite web browser che visualizzano pagine HTML, il presentation layer è costituito dal web server e dai moduli che concorrono a creare i documenti HTML (ad es. pagine attive lato server, Java Servlet), mentre il browser è il client
- In altri casi il client ed il presentation layer possono essere fusi insieme: nei sistemi ***client/server***, in genere, un programma assolve ad entrambi i compiti (ad es. la presentazione è lato client realizzando Applets o Applicazioni Java Swing)

Application Logic Layer

• E' il livello del sistema che si occupa del **processamento dei dati** necessario per produrre i risultati da inoltrare al livello di presentazione
• Un programma che implementa le operazioni legate ad un prelievo su un conto corrente bancario, o la sequenza di passi da compiere per effettuare un acquisto on-line sono esempi di logica applicativa di un sistema
• Il livello della logica applicativa è anche chiamato **processo di business**, insieme delle regole di business, o semplicemente ***server*** (in questi casi il sottostante livello è rispettivamente chiamato *persistence storage*, *business objects*, o **database**)

Resource Management Layer

• E' il livello che **gestisce i dati** che sono necessari al funzionamento dell'intero sistema
• I dati possono risiedere su una base dati, un file system, o altri contenitori di informazioni
• In linea di principio, il livello di gestione delle risorse gestisce le differenti sorgenti di dati che fanno parte del sistema informativo, indipendentemente dalla loro natura
• Nel caso in cui esso è implementato tramite un DBMS, è detto semplicemente data layer
• Secondo un'accezione più generale, questo può includere qualsiasi sistema in grado di fornire informazione

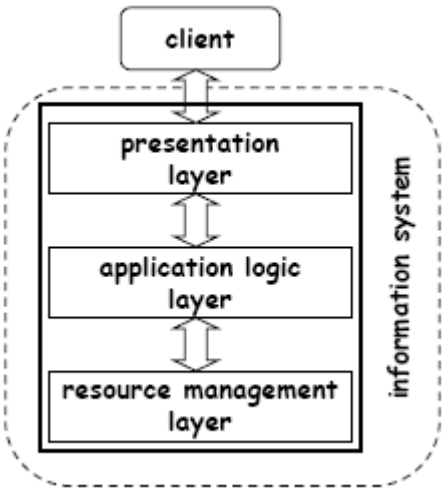
Architettura dei sistemi informativi

• Gli strati discussi finora sono costrutti concettuali che separano le funzionalità di un sistema informativo
• Nell'implementazione dei sistemi reali, questi strati possono essere combinati e distribuiti in diversi modi
• Quando consideriamo l'**implementazione** di un sistema informativo, facciamo riferimento ai livelli del sistema con il termine **tier**, piuttosto che conceptual layer

Architettura One-tier

L'architettura One-tier combina tutti i livelli concettuali in un unico tier
Rispecchia l'architettura hw basata su mainframe tipica dei primi calcolatori elettronici
Gli utenti accedevano tramite terminali non intelligenti (in genere schermo e tastiera)

Architettura One-tier – svantaggi

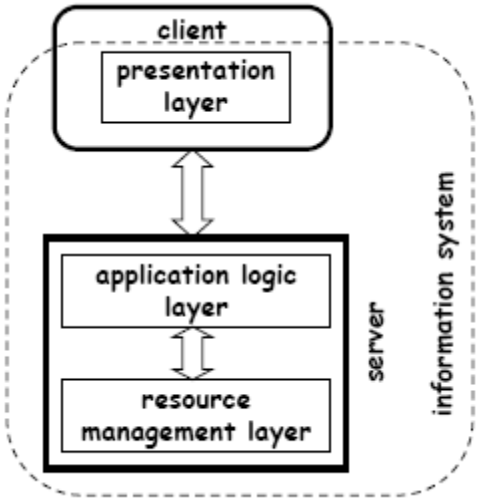


• L'intero livello di presentazione risiede sul mainframe, che controlla ogni aspetto dell'interazione con il client
• Il calcolo centralizzato delle visualizzazioni grafiche delle interfacce richiede molta potenza di calcolo. Di conseguenza, questo tipo di architettura supporta un numero non elevato di utenti
• Non vengono definite application program interface (API) che possano facilitare l'interazione con altri sistemi (poca portabilità del sistema)
• Sono difficili da manutenere e da aggiornare

Architettura One-tier – aspetti positivi

• Il progettista è libero di fondere i livelli concettuali per aumentare l'efficienza del sistema
• Non ci sono complessità dovute alla pubblicazione ed al mantenimento di interfacce, o a compatibilità fra diversi componenti
• Bassi costi di impiego, grazie alla bassa complessità dei client

Architettura two-tier



Il presentation layer migra verso il client

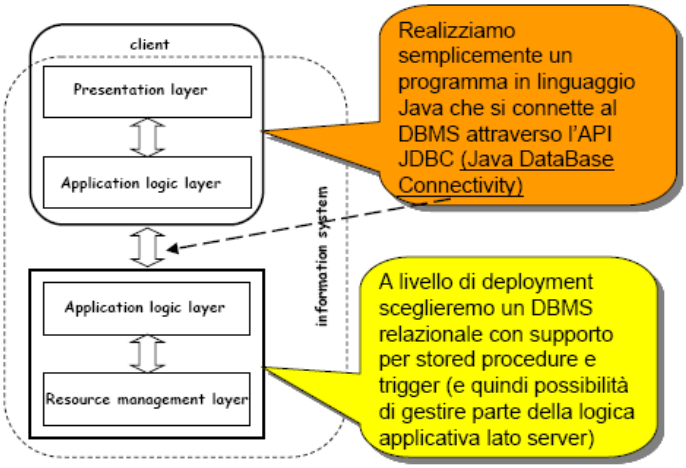
Si è affermata con il diffondersi di PCs e workstation e con lo sviluppo di nuove tecniche software per i sistemi distribuiti (ad es. RPC o *Remote Procedure Call* - chiamata di procedura remota)

Sistemi ***client/server***: i **client** hanno la possibilità di **processare ulteriormente** l'informazione.

I client si dividono in Thin client (con soli compiti di presentazione) e Thick client (inglobano parte della logica dell'applicazione)

Un esempio che, in genere, porta alla realizzazione di ***thick client***
è il caso in cui si vogliano realizzare semplici sistemi in cui i livelli concettuali di presentazione, logica dell'applicazione e gestione delle risorse siano ***disaccoppiati***:
• Con riferimento ad architetture client/server (anche quando il sistema risiede su una singola macchina), pur non volendo che la logica dell'applicazione risieda completamente nel DBMS per facilitare una eventuale migrazione verso architetture three tier
• Usando come interazione fra il livello della logica dell'applicazione ed il data layer ad esempio JDBC
• Realizzando un livello di presentazione concettualmente separato, anche senza necessariamente usare un middleware (in particolare senza ricorrere ad un application server).

Architettura client server con thick client



Architettura two-tier – vantaggi

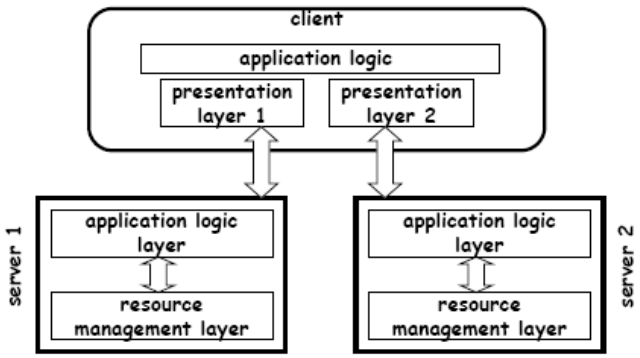
• I client sono indipendenti gli uni dagli altri: si possono avere diversi livelli di presentazione sulla base delle esigenze di diversi client
• La capacità di calcolo del client consente di avere interfacce grafiche sofisticate ed allo stesso tempo risparmio di risorse sul server
• Viene introdotto il concetto di API che facilita la comunicazione del server con i client (ad es. ODBC, JDBC)
• La possibilità di combinare sul server i livelli di logica dell'applicazione e di gestione delle risorse consente di mantenere una certa efficienza
• Al tempo stesso, le architetture two-tier garantiscono maggiore portabilità rispetto all'architettura one-tier

Architettura two-tier – svantaggi

• Supportano un limitato numero di client, a causa della necessità di mantenere informazioni sulla connessione e sull'autenticazione dei client
• La contemporanea presenza sul server della logica dell'applicazione e della gestione delle risorse richiede server dalle prestazioni abbastanza elevate
• I client si sono evoluti indipendentemente dai server ed hanno presentato nel tempo funzionalità sempre più avanzate. In particolare, hanno cercato di integrare più server, ma con l'architettura sbagliata!

Architettura two-tier ed Integrazione

L'interazione di uno stesso client con più server ha fatto nascere l'esigenza di uno strato di logica applicativa sul client stesso



Architettura three-tier

I livelli concettuali sono completamente disaccoppiati:
disaccoppiamento MVC : Model, View e Controller

Risponde all'esigenza di integrare più server

Si avvale della presenza di stabili interfacce sul lato server

Lo strato intermedio (middleware) implementa la logica dell'integrazione

Il middleware

Il middleware è l'insieme delle astrazioni di programmazione e delle infrastrutture che supportano lo sviluppo della logica dell'applicazione

• Come astrazione di programmazione:
- Nasconde i dettagli dell'hardware, della rete e della distribuzione della computazione
- Presenta sempre più potenti primitive che non cambiano i concetti di base di RPC ma aumentano la flessibilità nel loro uso
- La sua evoluzione è influenzata dalle tendenze nei linguaggi di programmazione (RPC e linguaggio C, CORBA e linguaggio C++, RMI e linguaggio Java, SOAP[1]-XML e Web services)

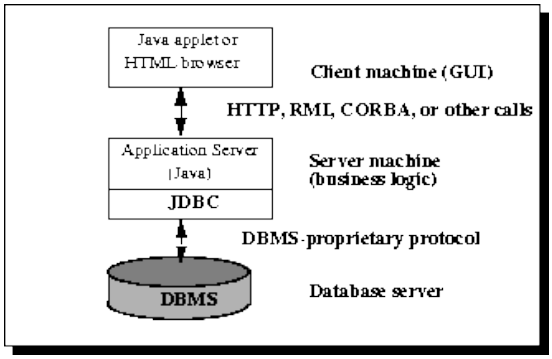
• Come infrastruttura:
- Fornisce una piattaforma per lo sviluppo e l'esecuzione di complesse applicazioni
- Presenta interfacce sempre più standardizzate
- Si evolve verso un'architettura orientata ai servizi
- L'obiettivo è l'integrazione delle piattaforme e la flessibilità nella configurazione

*La definizione di **middleware** consiste nel considerare come tale, tutto il software indipendente da una particolare applicazione, che utilizza i **servizi base di rete** come il TCP-IP. Quindi, nella gerarchia a strati che definisce le reti, il middleware si situa nello strato intermedio tra quello dei **servizi base di rete** e quello delle **applicazioni**.*

Quindi, col termine di **middleware** (software di mezzo) si intende il software che rende accessibili sul Web risorse hardware o software che prima erano disponibili solo localmente o su reti non Internet. Un esempio tipico di middleware è il software che rende accessibile dal Web un database.

Si tratta di un sistema a tre strati (in gergo **three-tiered**), dove ad esempio Java realizza l'interfaccia che sta nello strato di mezzo (perciò ***middleware***) tra il server del database e l'utente finale col browser. Si ricordi che le API JDBC supportano anche il modello *three-tier* nell'accesso a database.

Vanno considerati come middleware anche: emulatori di terminali, WWW, Corba, rpc, etc



***Three-tier** Architecture for Data Access*

[1] **SOAP** (inizialmente acronimo di **Simple Object Access Protocol**) è un protocollo leggero per lo scambio di messaggi tra componenti software. E' la struttura operativa (*framework*) estensibile e decentralizzata che può operare sopra varie pile protocollari per reti di computer. I richiami di procedure remote possono essere modellizzati come interazione di parecchi messaggi SOAP. SOAP dunque è uno dei protocolli che abilitano i servizi Web. Può muoversi sopra tutti i protocolli di Internet, ma HTTP è il più comunemente utilizzato e l'unico ad essere stato standardizzato dal W3C. SOAP si basa sul metalinguaggio XML e la sua struttura segue la configurazione Head-Body, analogamente ad HTML. Il segmento opzionale **Header** contiene *meta-informazioni* come quelle che riguardano il routing, la sicurezza e le transazioni. Il segmento obbligatorio **Body** trasporta il contenuto informativo e talora viene detto carico pagante, *payload*. Questo deve seguire uno schema definito dal linguaggio XML Schema.

Architettura three-tier

+ Consente l'integrazione di sistemi differenti all'interno di una LAN
+ Può essere usata negli stessi contesti dell'architettura two-tier: in questi casi tutta la logica dell'applicazione risiede nello strato intermedio, garantendo maggiore flessibilità e la scalabilità
+ Il livello della logica dell'applicazione può essere distribuito su diversi server

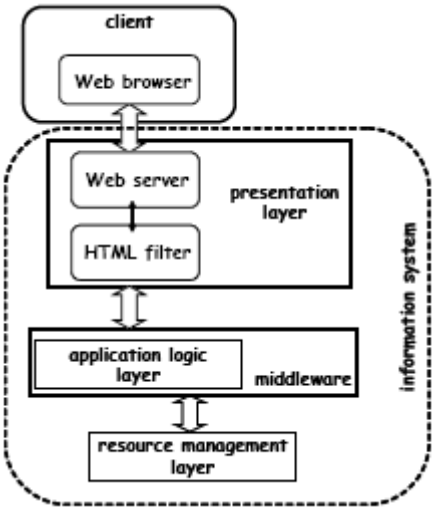
- Lo svantaggio principale è nel **decadimento delle performance** dovute ai problemi di comunicazione fra i diversi nodi del sistema
- Per integrare sistemi su internet devono fare uso di un componente addizionale: il **web server**

Tree-tier su internet: l'architettura N-tier

Si può considerare come la generalizzazione dell'architettura three-tier

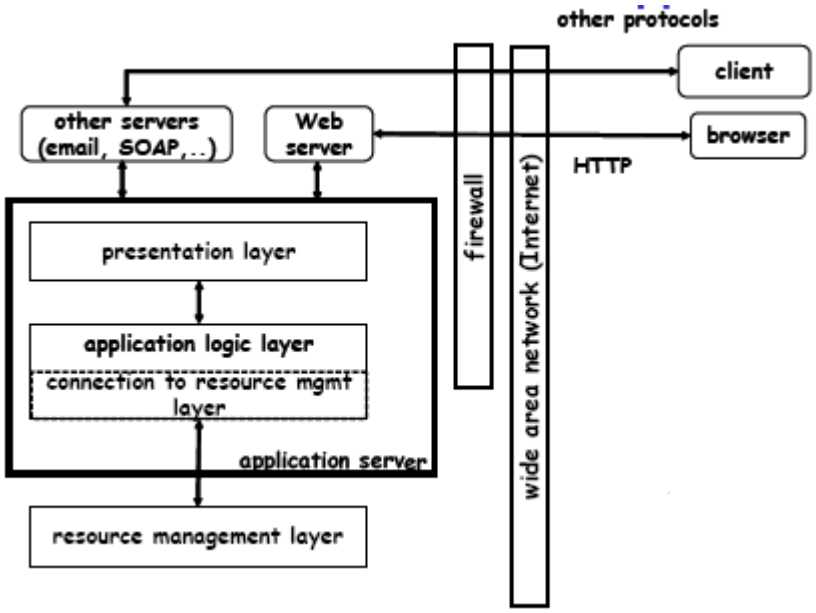
Il client è un browser web (è fuori dal presentation layer)

Il **web server** è un tier addizionale più complesso dei tradizionali tier di presentazione



Middleware for the web: Application Server

L'application server incorpora il presentation layer
Genera dinamicamente e gestisce i documenti che viaggiano sul protocollo http (in Java: Servlet, JSP,)
La connessione verso il Resource Management Layer è, in Java, tramite JDBC o ODBC

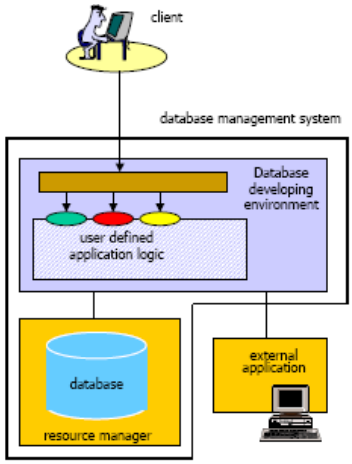


Application Server è in realtà un complesso sistema orientato allo sviluppo di applicazioni Web, ma non solo, infatti è progettato in generale per **separare la presentazione** (le pagine Web ne sono un esempio) **dalla logica interna** (cioè tutte quelle regole che determinano come devono essere elaborati i dati).

DBMS e l'approccio two-tier

• DBMS sono usati tradizionalmente per gestire i dati
• Gestire i dati non è l'unico scopo di un sistema: bisogna gestire la logica dell'applicazione
• Perchè non spingere la logica dell'applicazione dentro il DBMS?

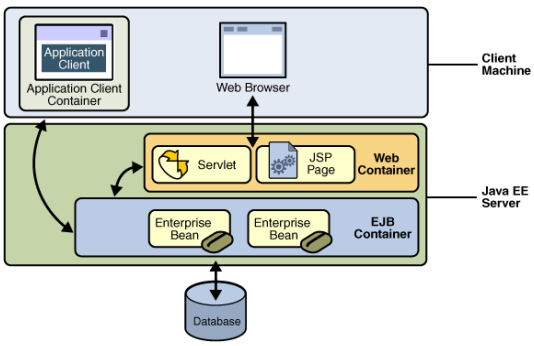
Questo è quello che molti produttori hanno fatto, proponendo per i DBMS un modello two-tier, in cui il DBMS fornisce molti **strumenti per la realizzazione della logica dell'applicazione** (ad es., triggers, replicazione, stored procedures, queuing systems, interfacce di accesso standard (ODBC, JDBC)).



DBMS e l'approccio three-tier

Secondo l'approccio two-tier, la logica dell'applicazione e la gestione dei dati convivono dentro il DBMS (che agisce da server nell'architettura)

• La comunicazione fra i due livelli è molto efficiente, ma il server necessario per eseguire i due livelli deve essere molto potente
• La scalabilità del sistema ne viene compromessa
• Un approccio three-tier aiuta a risolvere il problema, distribuendo la computazione
• Ovviamente la comunicazione fra i due livelli diventa più costosa



Si

EJB (Enterprise Java Beans): per realizzare la parte **Model** (detta anche **businness**)

parla esplicitamente di 2 sottolivelli del **livello di mezzo**:
il livello **server web** (tipo Tomcat)
e il livello **business** (contenitore di EJB)

Server J2EE: esempio di **Application Server**

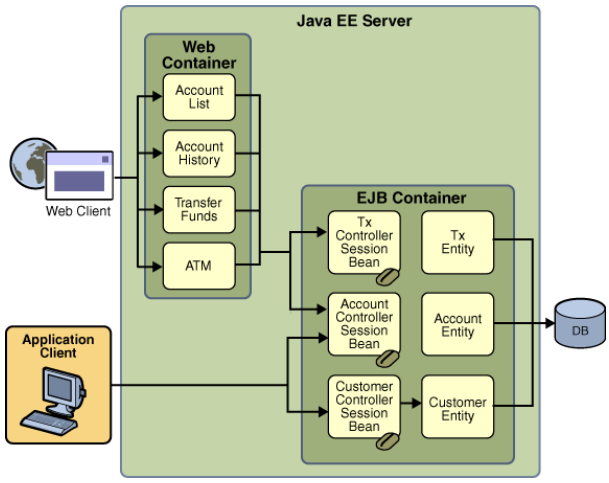


Fig: Duke's Bank **Application**