Applicazioni Web

Generalità

Con il termine applicazione web viene definito il software sviluppato e utilizzato attraverso tecnologie web e linguaggi specifici.
Alla base delle tecnologie web ci sono due concetti fondamentali:

- Tecnologie client-side e server-side
- Linguaggi di programmazione e linguaggi di markup

Tecnologie del Web

Possiamo distinguere le tecnologie del Web in due gruppi, in base al posto in cui avvengono le elaborazioni.

- **Tecnologie client-side**: sono quelle che vengono eseguite sul client, ovvero sul computer dell'utente. Queste tecnologie sono eseguite dal browser e sono basate su linguaggi di markup e linguaggi di scripting. Se chiediamo al browser di visualizzare le sorgenti di una pagina web, vedremo proprio il codice HTML e JavaScript che il browser ha eseguito per visualizzare la pagina.
- **Tecnologie server-side**: sono quelle che vengono eseguite sul server, ovvero sul computer che ospita il sito web. Queste tecnologie sono eseguite da un server web e sono basate su linguaggi di programmazione come PHP, Python, Java, ecc. Il browser non ha alcun accesso ai sorgenti, ma solo al risultato.

Linguaggi di programmazione e linguaggi di markup

- **Linguaggi di programmazione**: sono linguaggi che permettono di scrivere programmi che vengono eseguiti da un computer. I linguaggi di programmazione sono utilizzati per scrivere le applicazioni server-side.
- **Linguaggi di markup**: sono linguaggi che permettono di descrivere la struttura di un documento. I linguaggi di markup sono utilizzati per scrivere le pagine web, come ad esempio HTML e XML.

Il modello client-server

Il modello client-server è costituito da un insieme di host che gestiscono una (o più) risorse, i server, e da un insieme di client che richiedono l'accesso ad alcune risorse distribuite gestite dai server. Inoltre ogni processo server può a sua volta diventare client per richiedere accesso ad altre risorse gestite da altri (processi) server.
Non sono gli host a essere server o client ma i processi che sono in esecuzione su di essi, dove come processo si intende un programma in esecuzione.

Distinzione tra server e client

Un programma chiamato client richiede dei servizi a un altro programma chiamato server. Quest'ultimo è ospitato su un computer chiamato host ed è in ascolto tramite un **socket** su una determinata porta, in attesa che un client richieda la connessione: il client invia la richiesta al server tentando la connessione proprio tramite tale porta, quella cioè su cui il server è in ascolto.

La distinzione tra client e server è puramente logica: un host può essere sia client che server, a seconda del ruolo che assume in una determinata comunicazione.

Comunicazione unicast e multicast

Ci sono due tipi di comunicazione tra client e server:

- **Unicast**: il client invia una richiesta al server e il server risponde al client, il server comunica con un solo client alla volta. Questo tipo di comunicazione è utilizzata per le comunicazioni punto-punto, ovvero tra un host e un altro host.
- **Multicast**: un server può rispondere a più client contemporaneamente. Se la richiesta di connessione tra client e server va a buon fine il server, prima di stabilire il canale di connessione con il client, sposta la richiesta dalla porta dalla quale è stata inviata a una porta diversa, in modo che possa accettare altre richieste di connessione.

Livelli e strati

Le comunicazioni tra client e server avvengono attraverso una serie di livelli o strati, ognuno dei quali svolge un compito specifico. Ciascun livello funziona da server per i suoi client nel livello precedente e da client per il livello successivo ed è organizzato in base al tipo di servizio che fornisce.

Spesso il modello client-server a livelli è combinato con quello a strati dove ogni strato viene definito dal punto di vista funzionale come un "livello di astrazione".

Livelli e strati

In generale nelle applicazioni possiamo individuare tre tipi principali di funzionalità che corrispondono a una struttura in tre strati o livelli (modello three-tier):

- **Front-end**: è il livello che si occupa di interagire con l'utente.
- **Logica applicativa**: è il livello che si occupa di gestire la logica dell'applicazione.
- **Back-end**: è il livello che si occupa di gestire i dati.

Livelli e strati

La nomenclatura indicata è quella tipica delle applicazioni Web, ma la suddivisione in tre strati può essere effettuata per ogni tipo di applicazione:

- **Presentational layer**: è il livello che si occupa di presentare i dati all'utente, ricevere i dati inseriti dall'utente e inviarli al livello successivo. Per esempio nel web è il livello che si occupa di generare le pagine HTML.
- **Resource management layer**: è il livello che si occupa di gestire i dati. Per esempio nel web è il livello che si occupa di gestire i dati del database.
- **Business layer**: è il livello che si occupa di gestire la logica dell'applicazione e le relazioni tra i dati.

Architettura a un livello

Nell'architettura a un livello tutte le funzionalità sono implementate in un unico programma. Questo tipo di architettura è stato utilizzato per i primi mainframe a cui si collegavano i terminali, il cui compito era quello di visualizzare i dati e inviare i comandi inseriti dall'utente al mainframe.

Questa architettura non fa parte del modello client-server.

Architettura a due livelli

Verso gli anni ottanta, con l'avvento dei personal computer, si è diffusa l'architettura a due livelli, in cui le funzionalità sono suddivise in due livelli: livello client e livello server.

Possiamo individuare due tipi di architettura a due livelli:

- **Thin client**: il client si occupa solo di presentare i dati all'utente e di inviare i dati inseriti dall'utente al server. Il server si occupa di gestire i dati e la logica dell'applicazione.
- **Thick client**: il client si occupa di gestire i dati e la logica dell'applicazione. Il server si occupa solo di gestire i dati.

Inizialmente si è diffusa l'architettura thin client, ma con l'avvento di Internet si è diffusa l'architettura thick client.

Architettura a tre livelli

A partire dagli anni Novanta l'architettura client-server è a tre livelli e a ogni livello corrisponde uno strato architetturale, come precedentemente descritto:

- **Front-end o presentation tier**: è l'interfaccia verso l'utente.
- **Logica applicativa** o **middle tier (business-tier)**.
- **Back-end** con l'accesso alle risorse/ai dati, anche detto **data tier (o resource-tier)**.

Architettura a tre livelli

I vantaggi dell'introduzione del middleware sono notevoli, soprattutto in termini di prestazioni e di scalabilità. Inoltre, grazie alla separazione tra logica applicativa e gestione dei dati, è possibile modificare la logica applicativa senza dover modificare la gestione dei dati e viceversa.
Anche in termini di sicurezza il modello a tre livelli porta notevoli vantaggi in quanto rende possibile l'introduzione di sicurezza a livello di servizio e quindi più facilmente gestibile.
Di contro, l'introduzione del middleware comporta un aumento della complessità dell'architettura, rendendo più difficile la progettazione e la manutenzione.

Architettura a n livelli

Le architetture client-server a *n* livelli sono una generalizzazione del modello client-server a tre livelli dove vengono scomposti e introdotti un numero qualunque di livelli e server intermedi.
Questa scomposizione viene effettuata per suddividere ulteriormente i compiti dei vari strati: prende anche il nome di applicazione multi-tier.