Modello Client/Server

Architetture C/S

I **paradigmi di comunicazione** sono le classi che identificano la natura dei diversi protocolli di comunicazione che nell'ambito del client/server vengono utilizzati.
Si parla di
3 tipologie di paradigmi:

- **RPC** (**Remote Procedure Call**).
  Paradigma comprendente tutti quei protocolli di comunicazione che si basano su una interazione per la quale
  il client fa una richiesta, questa attiva una procedura sul server e il server al termine di questa procedura invia al client i dati richiesti (HTTP si colloca all'interno di questo paradigma).
- **RDA** (**Remote Data Access**).
  Variante del modello RPC nel quale
  il tipo di interazione tra client e server è finalizzata all'accesso ad una struttura dati gestita da un DBMS server (che costituisce l'elemento server della coppia).
  In questa serie di protocolli
  il client effettua una richiesta di informazioni (tipicamente una query SQL)

gestita da un DBSM server il quale accede al DB e fornisce l'informazione richiesta.

- **QMP (Queued Message Processing**).
  I protocolli di comunicazione che attengono a questa categoria fondamentalmente
  si basano sul principio di accodamento della richiesta che il client fa al server all'interno di una coda nella quale, in un momento successivo a quello di accodamento, vengono attivati processi di elaborazione che producono dei risultati e di conseguenza generano un risultato che viene accodato a sua volta in una coda analoga a quella gestita per gli input dalla quale possono essere recuperati dai client che accedono al server.
  Rispetto alle altre due architetture,
  in questo caso viene introdotto il concetto di accodamento sia dei messaggi di input che di output che i client e i server si scambiano.

C/S e Internet

Che cos'è Internet? È una rete elettronica per il collegamento di calcolatori. Tecnicamente parlando è una rete di reti basata sul protocollo TCP/IP (Transmission Control Protocol/Internet Protocol).

Da un punto di vista strutturale Internet è l'insieme di 3 tipologie base di rete:

Internet = Public Internet + Intranets + Extranets

- **Public Internet**: parte di rete accessibile da tutti (non gratuitamente).
- **Intranets**: sotto-reti di Internet ad accesso controllato (accesso consentito soltanto ad un sottoinsieme ben definito di utenti); possono essere utilizzati soltanto da chi è connesso ad un punto di accesso *interno* a questa rete.
- **Extranets**: componenti di una Intranet rese pubbliche tramite un processo di autenticazione (esempio: la rete dell'home banking con cui io mi connetto alla banca).

Il modello C/S si istanzia attraverso una serie di protocolli applicativi, attraverso i quali possiamo attivare dei servizi sulla rete stessa:

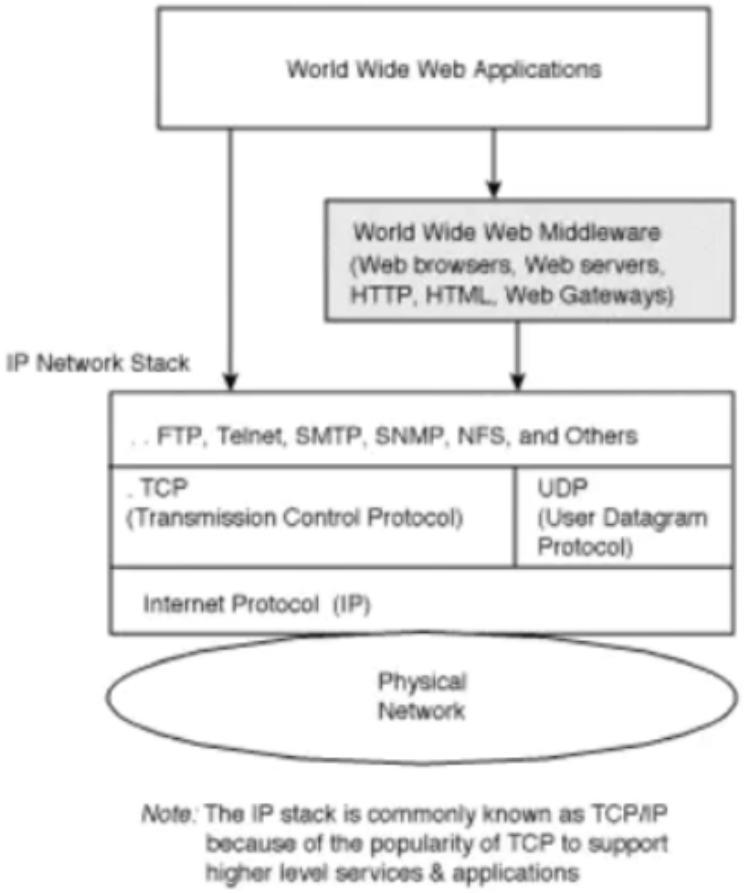
- Telnet: predecessore dell'HTTP. Attraverso un meccansimo di autenticazione (username e password) siamo in grado di collegarci in modalità terminale a una macchina diversa dalla nostra.

- FTP (File Transfer Protocol) : protocollo per lo scambio dei file tra due macchine.
- SMTP (Simple Mail Transfer Protocol) : protocollo dei server di posta.
- NFS (Network File System) : protocollo che consente di connettersi ad un disco remoto.
- DNS (Domain Name Services) : servizio che consente di associare a un indirizzo IP un nome logico (domain).
- SNMP (Simple Network Management Protocol)

C/S e W3

Il W3 può essere visto come un protocollo implementato da un insieme di middleware che operano su una rete IP.

Dato uno stack software (una rappresentazione gerarchica di un contesto applicativo), un **middleware** è quello strato che contiene quegli elementi del nostro stack software che si frappongono tra lo strato dell'interfaccia dell'applicazione stessa e lo strato di comunicazione dello stack sofware stesso. È quindi un insieme di servizi o ambienti di sviluppo che consentono a processi distribuiti di interagire tra di loro.



Un'applicazione di tipo Client (o Server) può essere schematizzata attraverso la struttura riportata sopra. Dal punto di vista software, è un software che comunica attraverso una rete fisica con altri software (client o server). Questa architettura software è costituita da:

- uno strato più esterno che è la vera e propria applicazione web che l'utente utilizza;
- uno strato intermedio costituito da tutte quelle tecnologie (ambiente di sviluppo, servizi) che consentono all'applicazione di operare secondo il modello del web (a questo strato appartengono : interpreti del linguaggio HTML, gestore dell'interazione attraverso il protocollo HTTP, gli elementi costituenti alla base dei browser e dei web service, ...)

Il W3 può essere visto quindi come un middleware, cioè come un insieme di componenti software che consentono ad un sistema client (o server) di interagire con la sua controparte secondo gli standard di comunicazione e le modalità di gestione dell'informazione e dei servizi definiti nell'ambito del W3.

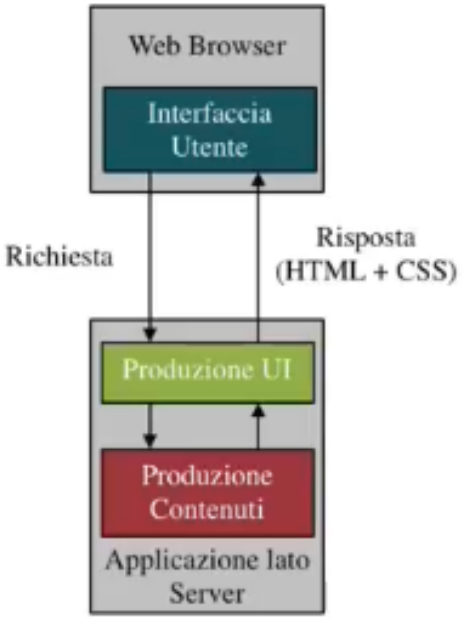
Il middleware W3 è una collezione di concetti e tecnologie quali:

- web browser: funzionalità core del browser usate per l’elaborazione.
- web server: sopra i web server vengono costruite le applicazioni.
- HTTP: protocollo di alto livello che usa il TCP/IP di livello più basso. Devo dare al web server il software per entrare nel mondo web (il server deve avere un componente capace di comprendere le richieste che gli arrivano via HTTP).
- URL: si intende il software che consente di generare un url a partire da un url parziale (tutto il software legato all’interpretazione degli url).
- HTML, XML ecc: troviamo i decodificatori da inserire nel web browser per la corretta decodifica delle pagine web.
- Strumenti di ricerca e navigazione.
- Gateways per risorse "non W3".

Il W3 introduce una nuova dimensione nell'architettura delle applicazioni C/S: le interfacce utente sono costituite dai Web Browsers (clients) che accedono direttamente ai documenti HTML gestiti dai Web Servers o ad altre applicazioni attraverso i Gateways.

I Server W3

Modello classico: un server W3 fornisce contenuti sotto forma di pagine HTML a un browser che ne fa richiesta, il quale visualizza queste pagine e gestisce l'interazione dell'utente con la pagina stessa.



Il server web si occupa sia della produzione dei contenuti, ma anche della produzione della User Interface, cioè della strutturazione di questi contenuti all'interno di un documento che poi determina le modalità di visualizzazione dei contenuti stessi lato browser.

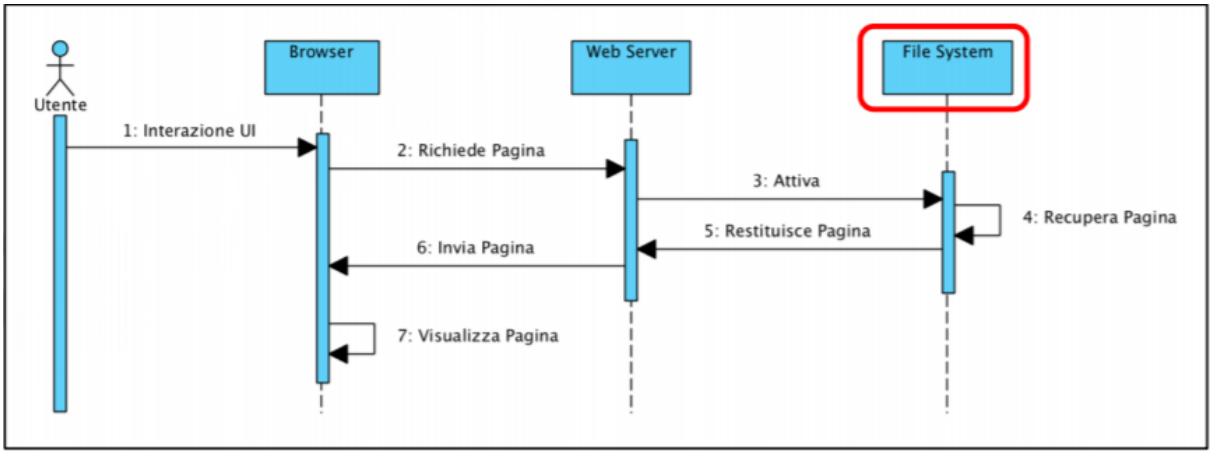
Il server (nel modello classico) non si occupa soltanto di fornire l'informazione, ma in qualche modo la struttura. Il browser quindi si comporta esclusivamente da visualizzatore dei contenuti e da gestore delle eventuali interazioni che l'utente del browser ha con la pagina stessa.

Contenuti statici

Il modello classico fornisce **contenuti statici**.

Alla richiesta di un client, che specifica tramite URL il documento a cui è interessato, il server recupera il file corrispondente e lo invia al richiedente.

Schema di interazione C/S:



Che cosa accade durante un'interazione C/S per la produzione di un contenuto statico? Mettiamo in gioco 4 attori:
- l'
utente, che attraverso il browser fa la richiesta;
- il
browser, che è l'applicazione lato client che determina l'interazione da gestire con il server web;
- il
server web;
- il
file system, contenuto all'interno del server web. Quello strato del SO che sta sulla macchina server che consente alla macchina server stessa di accedere ai file contenuti nel proprio file system.

1. Il processo si attiva con un'interazione che l'utente realizza con il suo browser (l'utente clicca ad esempio sull'ancora di una pagina per caricare il documento legato a quell'ancora, che nella nostra ipotesi è un contenuto statico, fondamentalmente un file che contiene del codice HTML).
   Interazione da parte dell'utente del browser con l'interfaccia utente.
2. Questa interazione determina nel browser l'attivazione di una richiesta, a partire dal link associato all'ancora dell'elemento su cui l'utente ha cliccato, di interazione con il server definito all'interno dell'URL per il recupero del contenuto specificato nell'URL stesso.
   Questa richiesta si realizza attraverso l'invio di un messaggio da parte del browser verso il web server, che è il depositario dei contenuti specificati nell'URL.

3-4-5. Questo messaggio verrà interpretato e decodificato dal web server e determinerà la necessità di andare a recuperare nel file system della macchina

server il contenuto statico (file) da inviare al client.
Il web server attiverà la procedura di accesso al file system per recuperare la pagina richiesta dal client a partire dal nome_file specificato nell'URL e restituirà questa pagina al web server. A questo punto il file system termina la sua operazione ed esce dal meccanismo di interazione.

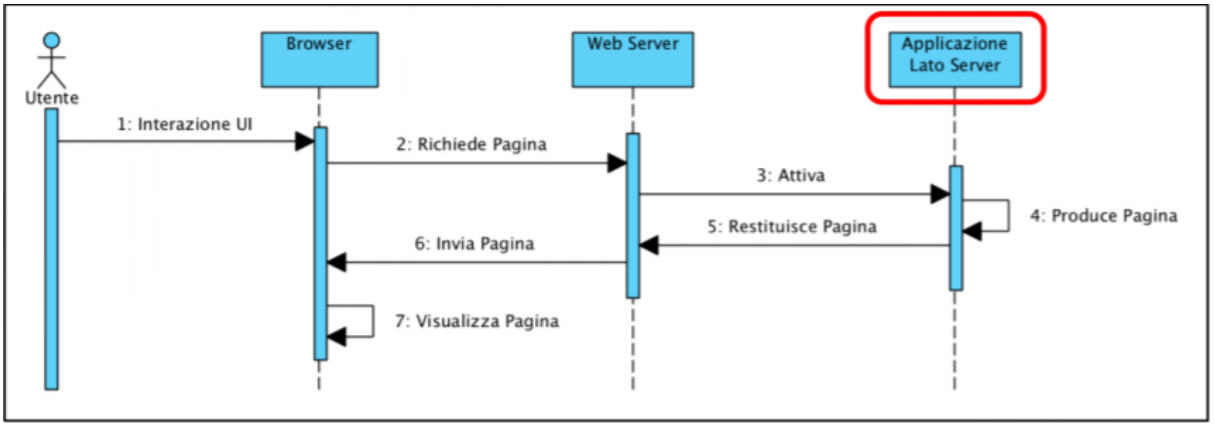
6-7. Il contenuto della pagina restituita dal file system al web server viene inserito all'interno di un messaggio HTTP (capsula) che il web server utilizza per interagire, rispondendo alla richiesta del client.
Il browser visualizza (decodifica) il contenuto della messaggio e "costruisce" il rendering grafico come specificato all'interno del file HTML.

Contenuti dinamici

Il modello classico fornisce **contenuti dinamici**.

In questo caso il server crea dinamicamente la pagina in base alle richieste del client.

Schema di interazione C/S:



Lo schema di interazione rimane articolato su 4 componenti, ma l'ultima non è il file system: il contenuto che il web server produce e rimanda al client è una pagina HTML che viene ri-generata all'atto di ogni richiesta da parte del client (perché il suo contenuto è tempo-dipendente).

La 4° componente dello schema è genericamente etichettata come *Applicazione Lato Server*: un programma che può essere eseguito dal server e che accede alla risorse del server e può essere attivato dal web server.

Tecnologia AJAX (Asynchronous Javascript And XML)

Il server non fornisce intere pagine, ma contenuti che vengono inseriti nella pagina corrente visualizzata dal browser.

AJAX è una tecnologia che ci consente di gestire un'interazione tra client e server che non determini esclusivamente da parte del server la produzione di contenuti costituiti da un'intera pagina, ma consenta al client di richiedere al server delle porzioni di pagina non formattati (dato grezzo), che vengono iniettate lato client e vanno a modificare la pagina esistente.

Il meccanismo di interazione non è più quello per cui il client, a seguito dell'interazione che l'utente fa sul contenuto della pagina, richiede al server una nuova pagina che viene integralmente caricata in sostituzione dell'esistente; ma determina un meccanismo per cui, sempre tramite l'interazione dell'utente, il client richiede al server un'informazione che poi viene renderizzata all'interno della pagina esistente.

L'enfasi lato server non è più sulla generazione dell'intera pagina, ma sulla generica generazione di contenuti parziali della pagina attualmente visualizzata dal web browser; contenuti che vengono generati a fronte di una richiesta che il web browser fa al server e che determinano come risultato non un codice HTML, ma un codice (quindi un contenuto informativo codificato) secondo standard che possono essere XML oppure JSON.

Nel dominio delle tecnologie AJAX JSON è il principale formato di scambio per questo tipo di interazione.

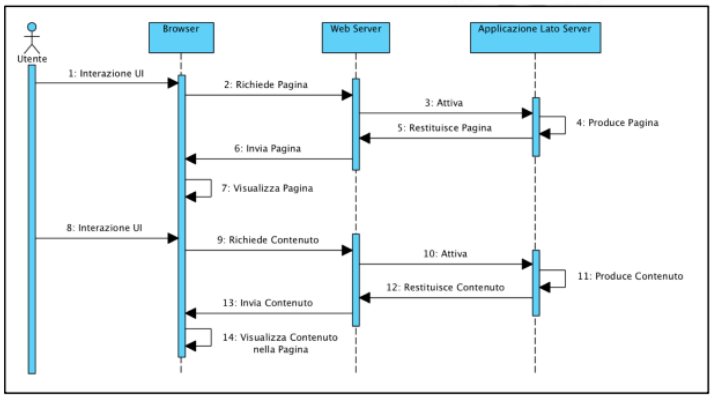
Il risultato generato dal server a fronte della richiesta del client viene passato non più direttamente all' interfaccia utente, ma piuttosto a un componente noto come **AJAX Engine** che consente di acquisire l'informazione e modificare dinamicamente la pagina visualizzata all'interno del browser stesso iniettando in essa i contenuti prodotti dal server

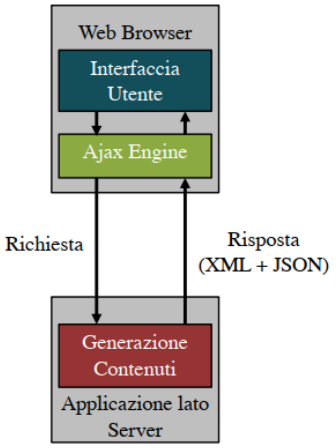
Nello schema classico il server produce la UI e il browser funge solo da interfaccia utente (trasforma il documento di testo in una pagina, fa il rendering).

Nello schema AJAX questo non avviene, l'applicazione server si occupa

soltanto di generare e fornire i contenuti; l'interfaccia utente inietta le porzioni dei contenuti all'interno della pagina corrente.

Gestione di contenuti di tipo AJAX:





Poiché stiamo parlando di iniezione di contenuti in una pagina esistente, l'interazione client/server, prima di passare per il canale AJAX, deve passare per il canale classico (quello di richiesta da parte del client di una pagina web e di risposta da parte del server con un documento che rappresenti la pagina stessa); da questo momento in poi i contenuti di questa pagina potranno essere modificati per iniezione.

- La 1° parte di questo schema quello che è lo schema che abbiamo già visto per i casi precedenti
  1. Interazione che l'utente fa sulla pagina attualmente visualizzata (per esempio cliccando su un'ancora).
  2. Il browser invia una richiesta di documenti in HTTP al server con l'URL del documento richiesto.
  3. Il web server attiva a questo punto una applicazione lato server (se il documento ha una struttura dinamica)

  4-5. Si ha la produzione del contenuto lato server, il quale viene restituito al web server

  6-7. Il web server impacchetta questo contenuto all'interno di un messaggio HTTP che viene poi ri-inviato al browser, il quale visualizza la pagina.
- 2° parte

8. L'utente vede un contenuto diverso rispetto a quello di partenza e può interagirci; se all'interno di questa pagina sono implementate le tecnologie AJAX (quindi dei meccanismi di interazione che passano attraverso questo processo), attraverso un'interazione l'utente può attivare un processo di caricamento di contenuti utilizzando la tecnologia Ajax.

9. Il browser richiede un contenuto al server, ma lo fa attraverso una procedura diversa rispetto a quella che abbiamo visto in precedenza per quello che riguarda il comportamento lato client.

10-11. La richiesta AJAX lato server determina l'attivazione di una applicazione lato server (sempre di una applicazione perché i contenuti AJAX non possono essere gestiti come contenuti statici, non sono mai fondamentalmente contenuti in un file che viene preso e riportato dal server) che produce il contenuto.

12-13. Questo contenuto viene restituito al web server che lo impacchetta in un messaggio HTTP e lo invia al browser.

14. Il browser però non visualizza il contenuto come nuova pagina web, ma inietta il contenuto nella pagina esistente (aggiunge a ciò che l'utente già vede nella stessa pagina HTML, che è quella che è già stata generata e visualizzata nella fase 7, un ulteriore contenuto che prima la pagina non aveva).
L'interazione avviene per porzioni di documento, e non per interi documenti. Non esiste quindi uno scambio classico in modalità richiesta-risposta di pagina, ma avviene attraverso un meccanismo che passa sempre per il protocollo HTTP che però trasferisce semplicemente porzioni di documento.

Single Page Applications

Il server non fornisce intere pagine, ma contenuti che il browser inserisce in pagine che genera automaticamente.

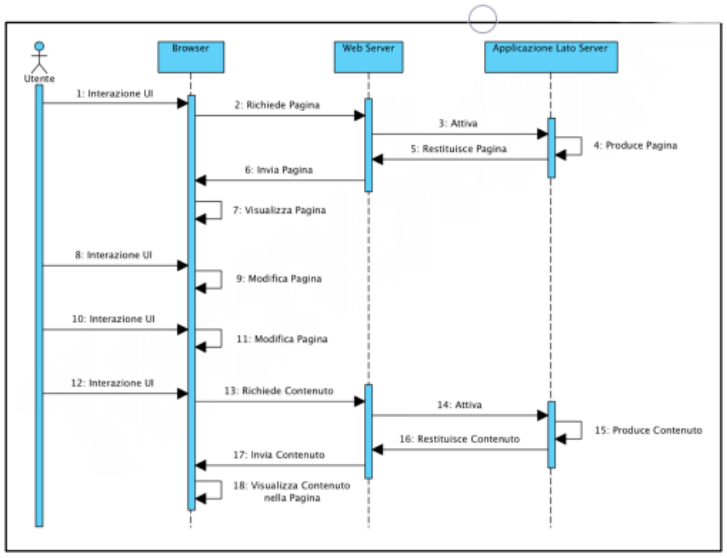
Utilizzata per la progettazione di applicazioni web che devono essere fruibili anche attraverso dispositivi mobili.

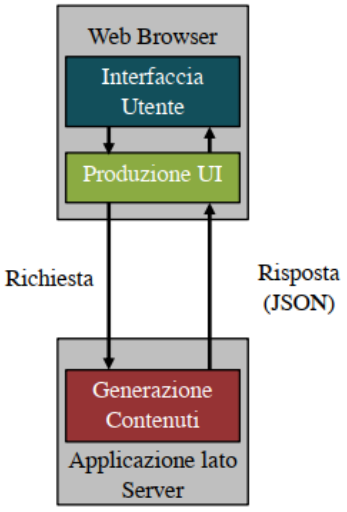
Utilizzando questo approccio il client carica la pagina iniziale della nostra app, gestisce l'interazione che l'utente fa con questa pagina in maniera locale (se i contenuti possono essere recuperati in maniera locale), ovvero accede al server che fornisce il contenuto laddove ci sia una richiesta di contenuto.

L'applicazione lato client genera anche il layout delle singole pagine (HTML), lasciando al server solamente il compito di fornire i dati necessari.

La produzione della UI è spostata direttamente sul client, mentre il server genera contenuti dinamici (non pagine) i quali vengono inviati al browser, codificati secondo lo standard JSON (standard che non prevede informazioni di tipo strutturale come l'HTML, ma fornisce solamente dati); è poi compito del web browser strutturare la visualizzazione delle diverse pagine e gestire l'interazione con l'utente.

Questo meccanismo di interazione è sostanzialmente diverso rispetto agli altri che abbiamo visto perchè fornisce lato client la possibilità di generare diverse pagine dall'applicazione web, senza bisogno di innescare per ogni pagina un meccanismo di comunicazione con il server.





- La 1° parte produce l'home page del sito e si articola nei soliti 7 punti che abbiamo già visto nei due schemi precedenti.

  1→7. L'utente interagisce con il browser (per esempio cliccando su un'ancora) che rimanda alla home page della nostra single page application. Il server a cui arriva la richiesta della home page attiva una applicaziione lato server che produce la pagina che viene restituita al web server, che poi la invia al browser attraverso un messaggio HTTP, che il browser visualizza.

- A questo punto l'utente si trova di fronte all'home page della nostra single page application, e comincia ad interagire con questa applicazione (come interagisce con una qualunque altra applicazioe web prodotto dalle

tecnologie viste in precedenza, cliccando su degli elementi che fanno riferimento ad altri contenuti).

8→11. Il passaggio da una pagina ad un'altra può essere gestito integralmente dal browser, perché al caricamento della prima fase, oltre ad aver caricato i contenuti specifici dell'home page, il browser ha caricato una serie di informazioni che consentono la strutturazione e la visualizzazione delle altre pagine del mio sito.
Questo passaggio da una pagina all'altra, senza interazioni con server ma semplicemente attraverso il recupero di informazione e di struttura che è stata caricata nella 1° fase dello sviluppo dell'applicazione, può avvenire per le diverse pagine dell'applicazione stessa.

Ci può essere una seconda interazione che l'utente fa con l'UI, la quale determina il caricamento di una seconda pagina (una modifica della pagina) direttamente gestita lato browser.
Più in generale può accadere che la richiesta che l'utente fa implichi un'interazione con un server, perché la richiesta fatta dall'utente implica la visualizzazione di contenuti che non sono disponibili stand-alone lato client.

12→17. L'interazione attiva un meccanismo di richiesta ad un qualche server (nel nostro caso è lo stesso server che ha fornito la pagina iniziale, ma in genere un qualunque server che fornisce informazione disponibile in rete) che determina la produzione di un contenuto lato server e l'invio di questo contenuto al browser, il quale però non visualizza il contenuto sotto forma di pagina web ma, con uno schema simile a quello dell'interazione AJAX, inietta questo contenuto all'interno della struttura della pagina che già possiede.
Questo meccanismo di interazione è sostanzialmente diverso rispetto agli altri che abbiamo visto, peché fornisce lato client la possibilità di generare diverse pagine dell'applicazione web stessa senza bisogno necessariamente, per ogni pagina, di innescare un processo di interazione con il server.

Protocollo HTTP

Il protocollo HTTP consente lo scambio di comandi che il client invia al server e che nel server determinano l'esecuzione di processi o di attività che determinano a loro volta la produzione del risultato che poi viene visualizzato dal client.

Nel caso dell'HTTP questi comandi viaggiano attraverso una connessione IP messa a disposizione dall'infrastruttura Internet sulla quale il web si basa.

Il meccanismo di interazione tra client e server è un meccanismo in cui il ruolo client e il ruolo server di due processi hanno una loro valenza soltanto nel contesto dell'interazione, cioè soltanto per il tempo necessario al client per inviare richieste al server e per ricevere da questo la risposta, dopodiché i due processi si disconnettono e perdono la loro identità di client/server, potendosi riconfigurare il client come server e il server come client di altre interazioni.

Questa caratteristica determina il fatto che il processo di interazione fra client e server è un processo **stateless**, cioè un client che si connette con un server, nel momento in cui termina l'interazione con quel server, perde memoria di questo fatto, ovvero non riconosce più quel server come qualcosa al quale ha già fatto accesso; nella prossima interazione che avrà con lo stesso server, lo tratterà come una macchina che vede per la prima volta. Significa che non esiste un meccanismo di memorizzazione permanente di informazioni o di dati relativi all'interconnessione fra due processi client e server.

Nel momento in cui le procedure che caratterizzano l'accesso all'informazione dei nostri server richiedono la memorizzazione dello stato pregresso dell'interazione fra client e server, si devono mettere in piedi alcuni meccanismi che affiancano l'interazione basata sul protocollo HTTP per poter mantenere memoria del passato dell'interazione fra una stessa coppia client-server.

Codifica MIME (Multipurpose Internet Mail Extension)

Ogni documento scambiato tra Client e Server è caratterizzato da un descrittore di tipo (fondamentalmente una stringa), il quale è codificato secondo lo standard **MIME** (*Multipurpose Internet Mail Extension*).

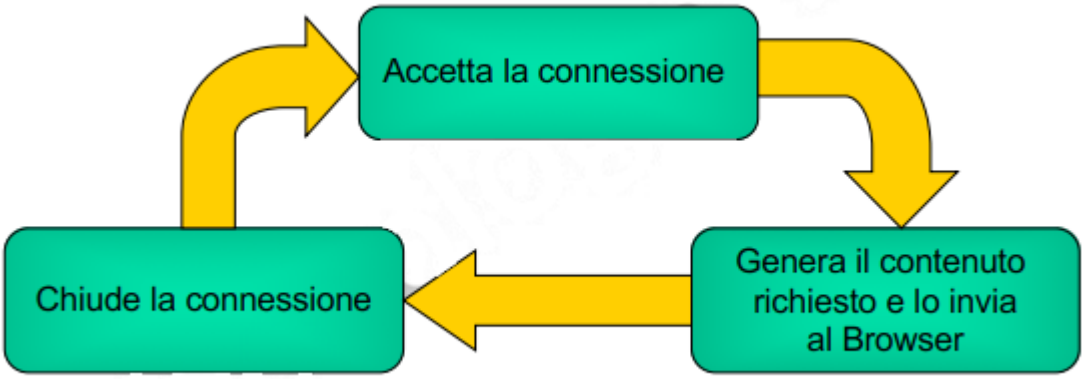
Un MIMEtype (associato a un documento) si rappresenta attraverso una codifica articolata in due parti: due sottostringhe divise da uno slash "/".

Il primo valore, a sinistra dello slash, rappresenta la tipologia macroscopicamente definita del contenuto (Text, Video ...). A ogni tipo primario è associato un tipo secondario che specifica lo standard di codifica utilizzato per quella tipologia di documento (Text/XML, Video/Mpeg). Queste codifiche vengono utilizzate nella negoziazione del formato.

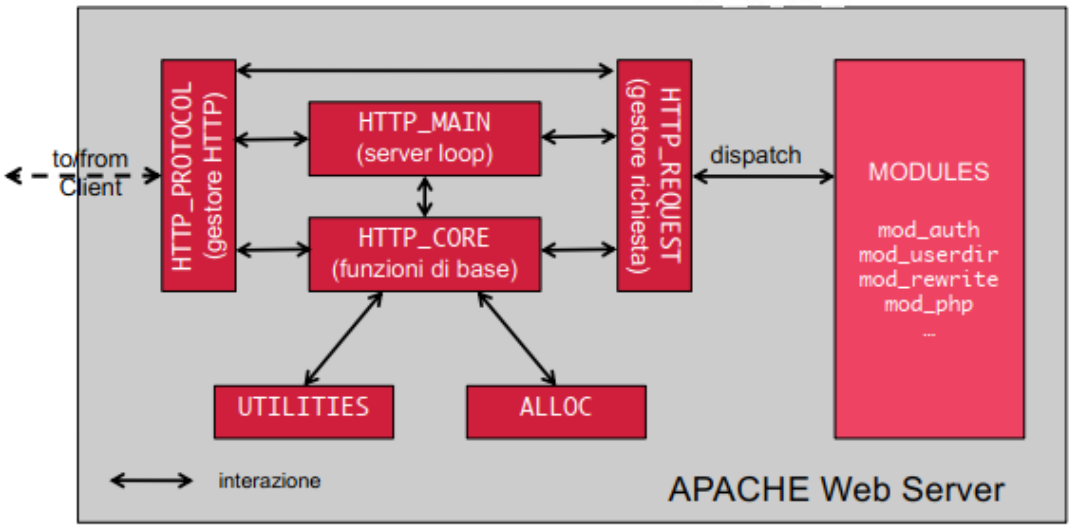
I dati dei tipi MIME viaggiano nei messaggi scambiati tra Client e Server all'interno delle intestazioni dei messaggi stessi.

La forma tipicamente associata ai messaggi che viaggiano nel protocollo HTTP è costituita da un'intestazione (una serie di informazioni sulla natura del contenuto) e da un corpo (il contenuto stesso del messaggio). Nell'intestazione del messaggio scambiato fra Client e Server nel protocollo HTTP viaggiano i tipi MIME.

Schema di funzionamento del Server



Architettura Web Server APACHE



Il web server Apache è il più diffuso tra i web server, poichè è open source ed è modulare.

Apache è un programma che gira su una macchina che ha un suo nome di dominio (e quindi è visibile all'interno della rete) e che può essere raggiunta da

richieste di altre macchine, che si configurano come client, che inviano alla nostra macchina delle richieste di acquisizione di risorse attraverso l'indicazione dell'URL della risorsa da acquisire. Tipicamente alla nostra macchina arriveranno tutte le richieste basate sull'URL in cui appare il nome di dominio che è assegnato alla macchina che stiamo considerando.

Nel momento in cui arriva una richiesta al web server Apache da parte di un browser, viene attivato il modulo *HTTP_PROTOCOL*, il quale gestisce il protcollo HTTP, cioè gestisce tutti i tipi di messaggi che possono essere inviati dai client.

L'**HTTP_PROTOCOL** si interfaccia con l'HTTP_MAIN, che implementa il server main loop (vedi *Schema di funzionamento del Server*).

L'**HTTP_MAIN** è quel blocco funzionale che gestisce l'accettazione della richiesta, l'attivazione del processo che genera i contenuti e l'inoltro della risposta al client. Si interfaccia con altri due protocolli, ovvero l'HTTP_CORE e l'HTTP_REQUEST.

L'**HTTP_CORE** è una sorta di libreria che contiene tutte le funzioni di base che vengono utilizzate dai moduli principali di Apache. È la parte del software di Apache che contiene le funzioni di servizio a tutte le altre componenti. Si possono identificare due sotto-componenti del CORE, ovvero UTILITIES (tutte le funzioni di utilità sulle quali si basa l'HTTP_CORE) e ALLOC (gestisce i meccanismi di gestione della memoria dinamica).
Si interfaccia anche col il modulo HTTP_REQUEST.

L'**HTTP_REQUEST** è costituito dalle componenti del modulo di Apache che acquisiscono la richiesta, la processano e producono il risultato.
(in
Schema di funzionamento del Server corrisponde al blocco "Genera il contenuto richiesto" senza la parte di invio al browser).

L'elemento che caratterizza l'architettura del web server Apache è l'articolazione attraverso una serie di moduli funzionali che sono definiti singolarmente dentro la sezione **MODULES**. Apache è cioè un'applicazione di tipo modulare (possiamo modulare l'efficienza dell'applicazione server in relazione alle reali esigenze della tipologia di server che vogliamo implementare).

Tra questi moduli alcuni identificano funzionalità fondamentali:

- **mod_auth** gestisce l'autorizzazione di un client ad accedere a particolari risorse gestite dal server.

- **mod_php** permette ad Apache di attivare su richiesta del server dei programmi scritti in PHP affinché producano il risultato richiesto dal client stesso.
- **mod_rewrite** consente di interpretare una richiesta fatta per una certa risorsa (ovvero un URL), rimappandola su un'altra risorsa interna la server stesso.

Browser

I browser sono client che utilizzano un'interfaccia grafica per visualizzare i documenti contenuti nei siti W3.

Sono costituiti da un nucleo software principale che implementa le funzioni di base.

Consentono l'accesso ai servizi basati su diversi protocolli W3 (HTTP, HTTPS, MAILTO, FTP, WebSocket, ...).

L'HTTP utilizza come strumento di base il protocollo Telnet di login remoto, nel quale i messaggi scambiati viaggiano in chiaro.
L'HTTPS utilizza invece l'SSH, che è la versione criptata del protocollo Telnet.

Il protocollo del WebSocket viene utilizzato in tutti i meccanismi di interazioni tra C e S nei quali occorre stabilire un collegamento permanente tra i due sistemi. Un'interazione di questo tipo serve quando il server fornisce al client un'informazione che è uno stream di dati, cioè un insieme di dati prodotti in tempo contuinuo e che devono essere mandati al client man mano che vengono prodotti (es. stream audio, video piuttosto che IoT come un sensore di temperatura ecc..).

Le funzionalità dei browser possono essere estese attraverso l'uso di *Plug-In* o *Helper*, software dedicati alla gestione di particolari formati di documenti.

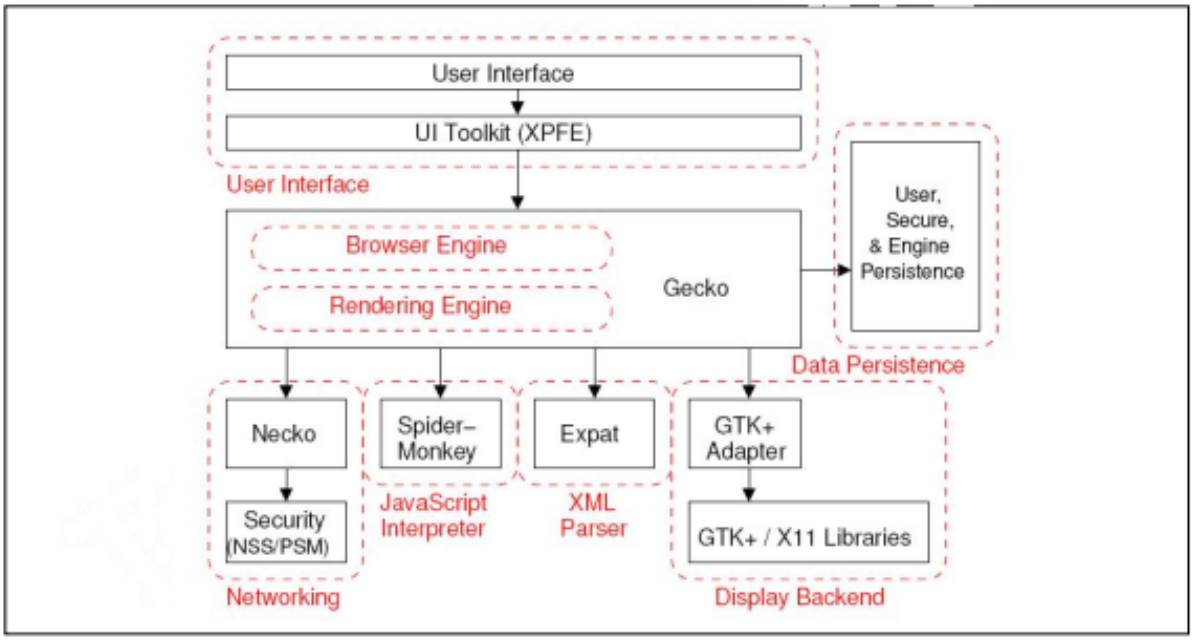
I **Plug-In** sono dei moduli software (che fondamentalmente implementano dei decodificatori) che possono essere inseriti all'interno del browser e che aggiungono una nuova funzionalità di decodifica senza modificare le componenti core del browser. Sono dei meccanismi di aggiornamento a runtime.

Gli **Helper** aiutano il browser nella visualizzazione di formati proprietari, i quali non richiedono soltanto la definizione di uno standard di decodifica, ma richiedono dei veri e propri programmi per la gestione dell'informazione.

(Esempio: file .docx. Quando clicchiamo su un'ancora contenente un riferimento ad un file di Word, si apre la finestra con scritto "Si sta cercando di scaricare un file di tipo .docx, cosa fare?" A questo punto si può indicare un programma che sia in grado di gestire quel documento oppure salvare il documento sotto forma di file).

Nel momento in cui si entra in modalità "*helper*" il client manda la richiesta al server, il server che non sa decodificare quel MIMEtype chiede aiuto all'utente per sapere come gestire quel "*problema*".

Architettura Browser Firefox



L'architettura di Firefox suddivide in macroblocchi tutte le principali componenti funzionali; è articolata in un core che realizza le funzionalità del browser stesso noto come *Gecko*, e che al suo interno contiene il motore del browser (*Browser Engine)* e il motore di visualizzazione dei contenuti (*Rendering Engine*).

Il motore del browser è costituito da un insieme di funzionalità che decodificano e gestiscono l'accesso all'informazione che il browser riceve, funzionalità che successivamente attivano il motore di rendering per la sua visualizzazione.

Il core di Firefox si interfaccia con tutti gli altri macroblocchi che costituiscono il browser.

Lo *User Interface* è il bloco che gestisce l'interfaccia utente, cioè quel blocco che visualizza il risultato del rendering prodotto dal motore di rendering per la visualizzazione di un certo documento e soprattutto gestisce le interazioni che l'utente attiva con il documento visualizzato. In questo caso l'UI utilizza una sua libreria di funzionalità di base (*toolkit, XPFE)*.

Un altro modulo che caratterizza questo browser è quello consente di gestire l'identità dell'utente che sta utilizzando il browser, tutte le problematiche legate alla sicurezza delle informazioni lato client, e il motore di persistenza (cioè l'insieme di quelle librerie che consentono di gestire la memoria locale del browser in un'ottica di persistenza dell'informazione tra una sessione e l'altra).

Il modulo *Networking* (Necko) gestisce i collegamenti di rete, quindi si occupa di gestire il meccanismo di scambio di informazioni fra il client e la rete Internet utilizzando i protocolli web analizzati. Utilizza per la sua funzionalità un insieme di librerie che si occupano della gestione dei protocolli sicuri.

Un altro componente che ogni browser ha è l'interprete JavaScript; JavaScript viene utilizzato lato server per introdurre una dinamicità nella gestione dei contenuti di una pagina e per implementare una serie di funzionalità lato client.

Il Parser XML non è direttamente integrato nell'engine del browser, perché sono delle componenti che possono essere presenti o meno nelle diverse versioni di Firefox stesso; nel browser engine è sempre integrato l'interprete HTML, perché è una compnente fondamentale per la rappresentazione dei contenuti delle pagine web.

Il *Display Backend* è la libreria per la visualizzazione dei contenuti che il browser presenta all'utente.