

# ICT INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

---

MODULO: Progettazione  
UNITÀ: Progettazione 2

Ing. Delisio Roberto



## API REST

# Fondamenti delle Applicazioni Web

Le applicazioni distribuite sono applicazioni in cui i vari componenti che compongono il sistema si trovano su macchine distinte.

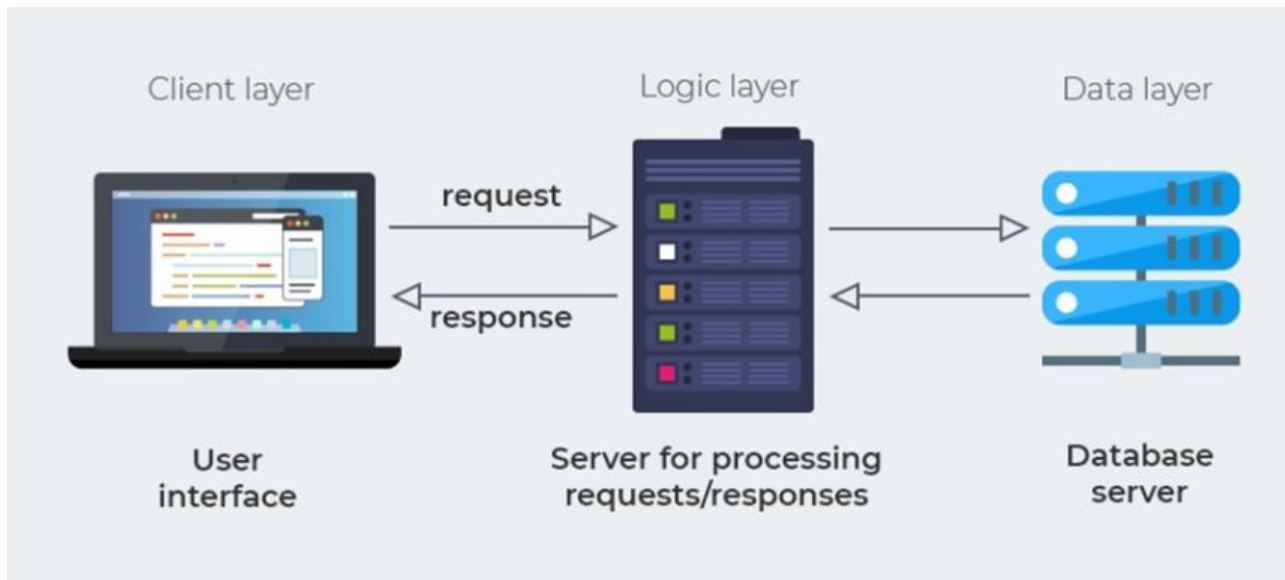
- I vari processi cooperano scambiandosi dati e messaggi

Un'applicazione distribuita è strutturata solitamente con un'architettura multi layer

- Ogni layer è costituito da un set di componenti
- Ogni set di componenti realizza un set di funzionalità

L'architettura multilivello più diffusa è l'architettura three-tier.

- Essa prevede:
- Client Layer
- Logic Layer
- Data Layer



# Client tier -> livello presentazione

**Scopo principale** è visualizzare le informazioni e raccogliere dati dall'utente e dall'interazione con l'utente

Questo tier di livello superiore, ad esempio, può essere eseguito su un browser web, come applicazione desktop o come GUI (finestra grafica).

Generalmente, i tier presentazione web sono sviluppati utilizzando HTML, CSS e JavaScript.

**Scopo principale** è recuperare informazioni dal client tier, elaborarle e metterle in relazione con il data tier.

Questo tier di livello intermedio **contiene la logica di business**, un insieme specifico di regole e comportamenti che il sistema fornisce.

Il tier applicativo può anche **aggiungere, eliminare o modificare i dati nel data tier** .

Il tier applicativo è tipicamente sviluppato utilizzando Python, Java, Perl, PHP o Ruby e comunica con il data tier tramite librerie(API)

**Scopo principale** è archiviare e gestire i dati persistenti dell'applicazione, solitamente attraverso un database

Questo tier è il livello più basso e interagisce solo con il livello applicativo.

Questo livello si realizza con database relazionali come PostgreSQL, MySQL, MariaDB, Oracle, DB2, Informix o Microsoft SQL Server o database NoSQL come Cassandra, CouchDB o MongoDB.

L'architettura Client-Server è il modello fondamentale su cui si basano quasi tutte le applicazioni web. Descrive una relazione in cui un programma, il Client, richiede servizi o risorse a un altro programma, il Server.

Pensa a un ristorante:

- **Il Cliente (Client):** Sei tu, seduto al tavolo. Hai bisogno di qualcosa (cibo, bevande). Per ottenerlo, fai una richiesta. Nel mondo web, il client è tipicamente il browser sul tuo computer o smartphone.
- **Il Server (Server):** È la cucina del ristorante. È sempre attiva, in attesa di ricevere ordini. Quando arriva un ordine (una richiesta), la cucina lo elabora (prepara il piatto) e restituisce il risultato (il piatto pronto). Nel mondo web, il server è un potente computer sempre connesso a Internet, in attesa di ricevere richieste dai client.

Il flusso di comunicazione è sempre lo stesso:

- **Richiesta (Request):** Il client invia una richiesta al server (es. "dammi la pagina principale di questo sito").
- **Elaborazione (Processing):** Il server riceve la richiesta, la elabora (cerca i dati in un database, esegue un calcolo, ecc.).
- **Risposta (Response):** Il server invia una risposta al client (es. il codice HTML della pagina richiesta, dei dati, o un messaggio di errore).

Questa architettura permette di centralizzare la logica complessa e i dati sensibili sul server, mantenendo il client leggero e focalizzato sulla presentazione delle informazioni.



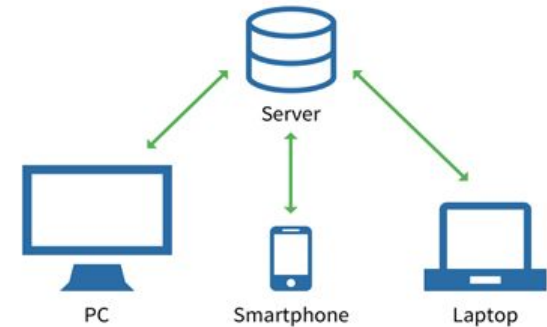
Un sistema client-server quindi è un contesto applicativo di rete dove un programma detto CLIENT richiede servizi ad un altro programma detto SERVER

Il programma cliente e il programma server girano tipicamente su macchine distinte collegate in rete.

Il Client potrebbe usare differenti dispositivi per collegarsi al Server

Regole generali:

1. Il server eroga servizi (su richiesta) per i client
2. I client si collegano ai server e richiedono servizi
3. Client e server devono aver concordato un protocollo di comunicazione e scambio dati



Il modello ha due attori, il Client (chi chiede) e il Server (chi risponde).

## Come il Client Trova il Server: Domini e DNS

- **Il Problema:** I server su Internet sono identificati da indirizzi numerici chiamati indirizzi IP (es. 142.250.184.142), che sono difficili da ricordare per gli esseri umani.
- **La Soluzione:** Usiamo i nomi di dominio (es. [www.google.com](http://www.google.com)), che sono etichette facili da memorizzare.
- **Il Traduttore (DNS):** Il DNS (Domain Name System) agisce come una "rubrica telefonica di Internet". Quando un utente digita un nome di dominio nel browser, il sistema DNS ha il compito di tradurre quel nome nell'indirizzo IP numerico corrispondente, permettendo al client di localizzare e contattare il server corretto.

Flusso di una Richiesta (con DNS):

1. L'utente scrive `www.google.com` nel browser (Client).
2. Il browser chiede al DNS: "Qual è l'IP di `www.google.com`?"
3. Il DNS risponde: "L'IP è `142.250.184.142`".
4. Il browser invia la sua richiesta HTTPS all'indirizzo `142.250.184.142` (Server).
5. Il server risponde.

Un indirizzo IP (Internet Protocol) è un'etichetta numerica che identifica in modo univoco un dispositivo (come un computer o uno smartphone) all'interno di una rete. La sua struttura dipende dalla versione del protocollo: IPv4 o IPv6.

## IPv4 (Internet Protocol version 4)

Questo è il formato più comune e conosciuto. Un indirizzo IPv4 è un numero a 32 bit, rappresentato per comodità come quattro blocchi di numeri decimali separati da un punto (notazione "dotted-quad").

- Formato: X.X.X.X (es. 192.168.1.10)
- Struttura di ogni blocco: Ogni blocco (chiamato ottetto) rappresenta 8 bit e può avere un valore compreso tra 0 e 255.

### Un indirizzo IPv4 è diviso in due parti logiche:

1. Network ID (Identificativo di Rete): La prima parte dell'indirizzo che identifica la rete specifica a cui il dispositivo è collegato. Tutti i dispositivi sulla stessa rete locale condividono lo stesso Network ID.
2. Host ID (Identificativo di Host): La seconda parte dell'indirizzo che identifica in modo univoco il dispositivo specifico all'interno di quella rete.

La separazione tra queste due parti è definita dalla Subnet Mask. Ad esempio, con l'indirizzo 192.168.1.10 e la subnet mask 255.255.255.0, significa che:

- 192.168.1 è il Network ID.
- 10 è l'Host ID.

## IPv6 (Internet Protocol version 6)

Introdotta per superare l'esaurimento degli indirizzi IPv4, IPv6 usa un numero a 128 bit, offrendo uno spazio di indirizzamento quasi illimitato.

- Formato: È rappresentato come otto gruppi di quattro cifre esadecimali, separati da due punti.
  1. Esempio: 2001:0db8:85a3:08d3:1319:8a2e:0370:7344
- Regole di Compressione: Per semplificarne la lettura, si possono abbreviare:
  1. I zeri iniziali di ogni blocco possono essere omessi (es. 0db8 diventa db8).
  2. Una sequenza contigua di blocchi composti solo da zeri può essere sostituita una sola volta da due punti (::).
    - Esempio: 2001:0db8:0000:0000:0000:8a2e:0370:7344 diventa 2001:0db8::8a2e:0370:7344.

La sua struttura è più complessa e generalmente divisa in un prefisso di routing globale (per la rete), un ID di sottorete e un ID di interfaccia (per il dispositivo).

## IP Pubblici vs. IP Privati

Indipendentemente dalla versione, gli indirizzi IP si dividono in due categorie:

- IP Pubblico: È l'indirizzo univoco che il tuo Internet Service Provider (ISP) ti assegna per essere identificato su Internet.
- IP Privato: È un indirizzo usato all'interno di una rete locale (come la rete Wi-Fi di casa tua). Esistono intervalli specifici riservati per questo uso (es. 192.168.x.x, 10.x.x.x). Questi indirizzi non sono visibili su Internet e permettono a più dispositivi di condividere un unico IP pubblico tramite un processo chiamato NAT (Network Address Translation) gestito dal router.

Un nome di dominio è un'etichetta facile da ricordare che identifica una risorsa su Internet, come un sito web. Il DNS (Domain Name System) è il sistema che traduce questi nomi leggibili in indirizzi IP numerici, necessari ai computer per comunicare tra loro.

## Com'è Fatto un Dominio: Un'Architettura Gerarchica

Un nome di dominio ha una struttura gerarchica che si legge da destra verso sinistra. Ogni "punto" separa un livello diverso della gerarchia.

Prendiamo come esempio l'indirizzo mail.google.com:

1. **Top-Level Domain (TLD) o Dominio di Primo Livello:** È la parte più a destra (.com). Definisce la categoria più ampia del dominio.
  - TLD generici (gTLD): .com (commerciale), .org (organizzazione), .net (network).
  - TLD nazionali (ccTLD): .it (Italia), .de (Germania), .uk (Regno Unito).
2. **Second-Level Domain (SLD) o Dominio di Secondo Livello:** È la parte centrale (google). Questo è il nome che viene registrato ed è il cuore dell'identità del brand. È univoco all'interno del suo TLD (esiste una sola google.com).
3. **Subdomain (Sottodominio) o Dominio di Terzo Livello:** È la parte più a sinistra (mail). I sottodomini vengono creati dal proprietario del dominio di secondo livello per organizzare e separare diverse sezioni o servizi del proprio sito.
  - www: Convenzionalmente usato per il sito web principale.
  - api: Spesso usato per esporre le API.
  - blog: Per una sezione dedicata al blog.

Se il dominio è l'indirizzo, il DNS è la "rubrica telefonica" che lo rende utilizzabile. Questa gestione si basa su due componenti chiave: i Nameserver e i Record DNS.

## **Nameserver (NS)**

Un Nameserver è un server specializzato che contiene tutti i record DNS per un determinato dominio. Quando acquisti un dominio (es. miosito.it), devi specificare quali sono i suoi "Nameserver autoritativi". Solitamente sono forniti dal tuo provider di hosting (es. ns1.tuohosting.com e ns2.tuohost-ing.com).

Questi server diventano la fonte ufficiale di verità per il tuo dominio. Qualsiasi computer nel mondo che voglia sapere l'indirizzo IP di miosito.it finirà per interrogare uno di questi nameserver.

I **Record DNS** sono le singole "voci" all'interno della rubrica gestita dal nameserver. Ogni record ha un tipo specifico che definisce il tipo di informazione che fornisce. I più importanti sono:

- **Record A (Address):** È il record più comune. Associa un nome di dominio a un **indirizzo IPv4**.
  - miosito.it. A 88.99.100.123
  - (Significa: "chi cerca miosito.it deve andare all'indirizzo IP 88.99.100.123").
- **Record AAAA (Quad A):** Simile al record A, ma associa un dominio a un **indirizzo IPv6**.
  - miosito.it. AAAA 2001:0db8:85a3:0000:0000:8a2e:0370:7334
- **Record CNAME (Canonical Name):** È un alias. Permette a un dominio di puntare a un altro dominio, invece che a un indirizzo IP.
  - www.miosito.it. CNAME miosito.it.
  - (Significa: "www.miosito.it è solo un altro nome per miosito.it. Trovate l'IP di quest'ultimo"). È utile per avere più nomi che puntano allo stesso server.
- **Record MX (Mail Exchange):** Specifica quali sono i server di posta elettronica responsabili di ricevere le email per quel dominio.
  - miosito.it. MX 10 mail.provider.com.
  - (Significa: "tutte le email indirizzate a @miosito.it devono essere gestite dal server mail.provider.com". Il numero 10 indica la priorità).
- **Record TXT (Text):** Permette di associare al dominio una stringa di testo arbitrario. È comunemente usato per scopi di verifica (es. per dimostrare a Google o Microsoft di essere il proprietario del dominio) o per configurazioni di sicurezza email come SPF e DKIM.
- **Record NS (Name Server):** Indica quali sono i nameserver autoritativi per un dominio (o sottodominio), delegando di fatto la gestione.



## Il Processo di Risoluzione DNS in Breve

Quando digiti `www.miosito.it` nel browser:

1. Il tuo computer chiede al suo risolutore DNS (solitamente fornito dal tuo provider Internet).
2. Se il risolutore non conosce la risposta, inizia un processo a catena: interroga i Root Server mondiali, che lo indirizzano ai server del TLD `.it`.
3. I server del `.it` dicono al risolutore: "Per `miosito.it`, devi chiedere ai suoi nameserver autoritativi, `ns1.tuohosting.com`".
4. Il risolutore interroga `ns1.tuohosting.com`, che finalmente legge i suoi record.
5. Trova prima il record CNAME per `www.miosito.it` che punta a `miosito.it`, e poi il record A per `miosito.it` che punta a `88.99.100.123`.
6. Restituisce l'IP `88.99.100.123` al tuo computer.
7. Il tuo browser contatta il server a quell'indirizzo IP.

# Cos'è un'applicazione web?

Un'applicazione web (web app) è un programma software a cui si accede tramite un browser web (come Chrome, Firefox, Safari) attraverso una rete come Internet o una intranet.

A differenza di un'applicazione desktop tradizionale che viene installata ed eseguita interamente sul computer dell'utente, un'applicazione web ha una parte del suo codice (il backend) che risiede su un computer remoto, il server, e una parte (il frontend) che viene eseguita nel browser dell'utente.

La caratteristica distintiva è l'interattività. Mentre un sito web può essere una semplice pagina di sola lettura, un'applicazione web permette all'utente di compiere azioni e manipolare dati.

**Esempi di applicazioni web:** Gmail, Google Docs, Facebook, Amazon, Trello.

Caratteristiche principali:

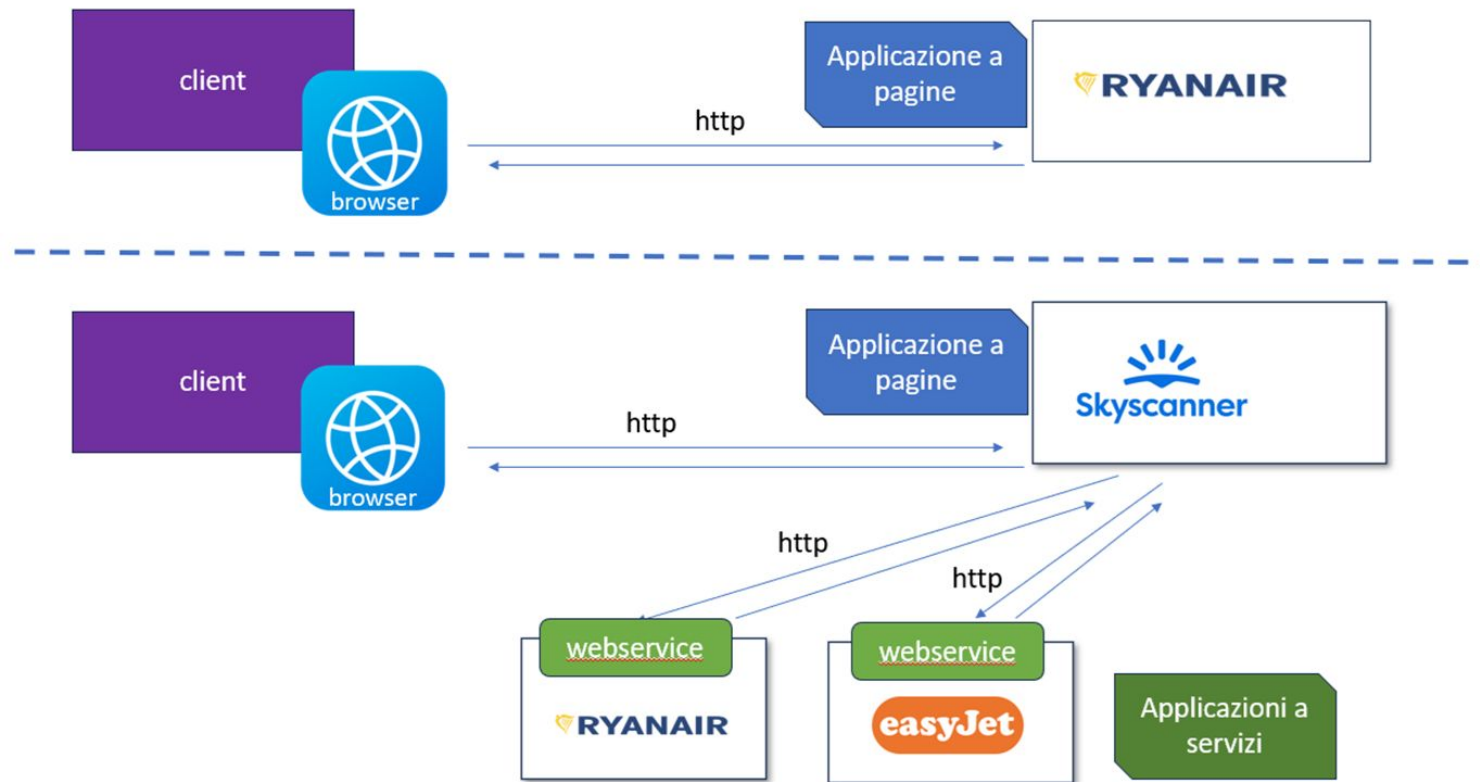
- Nessuna installazione richiesta da parte dell'utente.
- Accessibilità da qualsiasi dispositivo con un browser e una connessione a Internet.
- Centralizzazione degli aggiornamenti (si aggiorna l'applicazione sul server e tutti gli utenti vedono subito la nuova versione).

Le applicazioni web quindi sono particolari applicazioni distribuite che usano l'architettura client-server

## Caratteristiche

- Utilizzano il protocollo TCP/IP
- Si dividono in 2 famiglie principali:
  - Applicazioni a pagine: il client è l'utente che, attraverso il browser, chiede un servizio al server e riceve in risposta una pagina web
  - Applicazioni a servizi: il client è un programma che chiede un servizio al server e riceve in risposta dati

# Applicazioni a confronto



Il client è un programma (come il server)

- Applicazione Android
- Applicazione IOS
- Browser che esegue codice Javascript in una pagina web

Client e server si scambiano SOLO dati

Il sistema client-server è interoperabile

- Client e Server possono essere scritti con infrastruttura tecnologica e linguaggi indipendenti

La realizzazione segue uno di questi 2 stili principali:

- SOAP (stile classico)
- REST (nuovo stile)

Se l'architettura è Client-Server, Frontend e Backend sono i due lati dello sviluppo che costruiscono le due parti.

## Frontend (Lato Client)

Il Frontend è tutto ciò che l'utente vede e con cui interagisce nel browser. È la parte "visibile" dell'iceberg.

- **Ruolo:** Creare l'interfaccia utente (UI) e garantire una buona esperienza utente (UX).
- **Responsabilità:**
  - Struttura: Definire la struttura dei contenuti di una pagina.
  - Stile: Curare l'aspetto grafico, i colori, i layout e l'adattabilità ai diversi schermi (responsive design).
  - Interattività: Gestire click, animazioni, compilazione di form e aggiornamenti dinamici della pagina senza doverla ricaricare completamente.
- **Tecnologie principali:**
  - HTML (HyperText Markup Language): Per la struttura dei contenuti.
  - CSS (Cascading Style Sheets): Per lo stile e l'aspetto grafico.
  - JavaScript: Per l'interattività. Framework moderni come React, Angular e Vue.js sono basati su JavaScript e permettono di creare interfacce complesse e reattive.

# Frontend vs Backend: Ruoli e Responsabilità

## Backend (Lato Server)

Il Backend è il "cervello" dell'applicazione, la parte che l'utente non vede. È la parte "sommersa" dell'iceberg.

- **Ruolo:** Gestire la logica di business, i dati e la sicurezza.
- **Responsabilità:**
  - Logica applicativa: Eseguire le operazioni principali dell'applicazione (es. elaborare un pagamento, registrare un nuovo utente).
  - Gestione del database: Salvare, recuperare, modificare e cancellare dati da un database (es. lista di prodotti, post di un blog, informazioni degli utenti).
  - Autenticazione e autorizzazione: Verificare chi è l'utente (login) e cosa ha il permesso di fare.
  - Esporre le API: Creare i punti di contatto (endpoint) che il frontend può interrogare per ottenere o inviare dati.
- **Tecnologie principali:**
  - Linguaggi di programmazione: Python (con framework come Django, Flask), JavaScript (con Node.js), Java (Spring), PHP (Laravel), C# (.NET).
  - Database: SQL (PostgreSQL, MySQL) e NoSQL (MongoDB, Redis).
  - Web Server: Apache, Nginx.

Il ponte tra Frontend e Backend è l'API (Application Programming Interface). Il frontend non accede mai direttamente al database; chiede i dati al backend tramite una chiamata API.

Questi sono i due attori software principali nell'architettura Client-Server.

## **Browser Web (il Client Software):**

- Il suo compito principale è inviare richieste HTTP al server quando l'utente digita un indirizzo o clicca un link.
- Una volta ricevuta la risposta (tipicamente codice HTML, CSS, JavaScript), il suo secondo compito è interpretare (renderizzare) questo codice per mostrare all'utente una pagina web visivamente completa e interattiva.
- Esegue il codice JavaScript del frontend per gestire l'interattività.

## **Server Web (il Backend Software):**

- È un software specifico (es. Apache, Nginx) in esecuzione sulla macchina server.
- Il suo compito è ascoltare le richieste HTTP in arrivo dai browser.
- Se la richiesta è per un file statico (un'immagine, un file CSS), il server web lo trova e lo restituisce direttamente.
- Se la richiesta è per una risorsa dinamica (es. `api/products/123`), il server web la passa all'applicazione backend (scritta in Python, Java, etc.). L'applicazione elabora la richiesta, genera una risposta (spesso dati in formato JSON), e la restituisce al server web, che a sua volta la inoltra al browser del client.



# Differenza tra Siti Web Statici e Applicazioni Dinamiche

Questa distinzione è cruciale per capire perché abbiamo bisogno di un backend complesso e di API.

## Sito Web Statico

Un sito web statico è composto da un insieme di file (HTML, CSS, JS) pre-costruiti.

- Quando un utente richiede una pagina, il server web si limita a trovare il file HTML corrispondente e a inviarlo al browser così com'è.
- Il contenuto non cambia in base all'utente o alle sue interazioni. Ogni visitatore vede esattamente la stessa cosa.
- Analogia: Una brochure o un volantino digitale.
- Uso ideale: Siti di presentazione, portfolio, documentazione, landing page. Non richiedono un backend complesso o un database.

## Applicazione Web Dinamica

Un'applicazione web dinamica genera le sue pagine (o i suoi dati) "al volo" in risposta a una richiesta.

- Quando un utente richiede una pagina, la richiesta viene elaborata dalla logica del backend. Il backend può interrogare un database, personalizzare il contenuto in base all'utente (es. "Ciao, Marco!"), e costruire la risposta (HTML o dati JSON) al momento.
- Il contenuto è personalizzato e interattivo.
- Analogia: Una conversazione. La risposta dipende da chi sei e da cosa chiedi.
- Uso ideale: Social network, e-commerce, piattaforme di online banking, dashboard.

Nel contesto del tuo corso, le API REST sono il meccanismo che permette alle applicazioni dinamiche moderne di funzionare, consentendo al frontend di richiedere e manipolare dati in modo strutturato e prevedibile dal backend.

# Il protocollo HTTP

Un protocollo è un insieme di regole per dialogare.

Il protocollo TCP/IP è il protocollo utilizzato sul web

Le figure base sono Client e Server

Principio base: Il client invoca il server (mai il contrario)



Il protocollo TCP/IP è una famiglia di protocolli composta da 4 livelli

- Ogni livello si occupa di un aspetto e prevede un set di regole
- I livelli sono: applicazione, trasporto, rete, fisico**

# Livelli del protocollo

Il dialogo avviene correttamente solo se per ciascuno dei 4 livelli viene utilizzato lo stesso sotto protocollo.

Significa che affinché la comunicazione tra due sistemi (mittente e destinatario) avvenga con successo, il protocollo utilizzato da un determinato livello sul sistema mittente deve essere compreso e utilizzato anche dal livello corrispondente sul sistema destinatario. Sono come due persone che parlano la stessa lingua per capirsi.

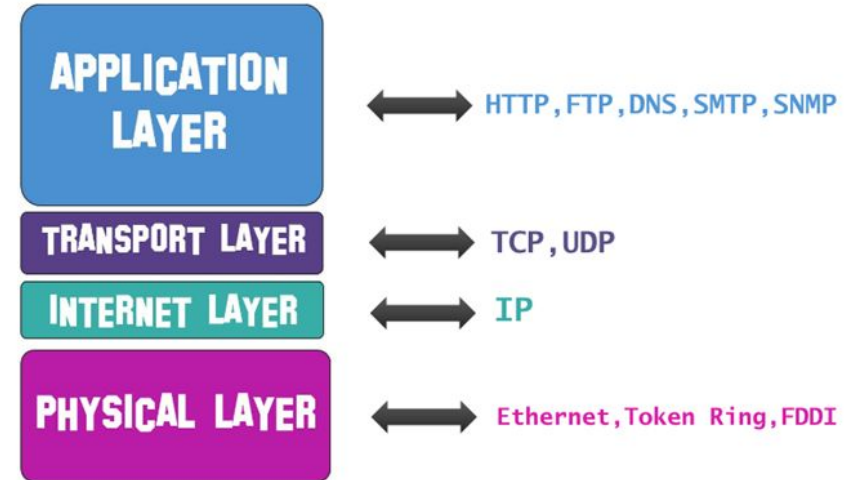
Livello fisico -> protocollo nativo

Livello rete -> IP

Livello trasporto -> TCP e UDP

Livello applicazione -> vari protocolli specifici per varie esigenze

- HTTP -> hyper text
- FTP -> file di grandi dimensioni
- SMTP -> mail
- ecc



Il livello di Applicazione è il più vicino all'utente finale. È responsabile di fornire i servizi di rete direttamente alle applicazioni software, come i browser web, i client di posta elettronica e i software per il trasferimento di file. In pratica, questo livello gestisce la rappresentazione dei dati, la codifica e il controllo del dialogo.

Funzioni principali:

- **Interfaccia con l'utente:** Fornisce i protocolli che le applicazioni utilizzano per scambiarsi dati sulla rete.
- **Gestione della sessione e presentazione:** A differenza del modello OSI, il TCP/IP raggruppa in questo unico livello anche le funzioni di gestione della sessione (apertura, mantenimento e chiusura di una connessione tra due host) e di presentazione (formattazione e traduzione dei dati, come la crittografia e la compressione).

Protocolli comuni:

- HTTP (HyperText Transfer Protocol): Utilizzato per la navigazione web.
- HTTPS (HTTP Secure): La versione sicura di HTTP, che utilizza la crittografia.
- FTP (File Transfer Protocol): Per il trasferimento di file.
- SMTP (Simple Mail Transfer Protocol): Per l'invio di posta elettronica.
- POP3 (Post Office Protocol 3) e IMAP (Internet Message Access Protocol): Per la ricezione della posta elettronica.
- DNS (Domain Name System): Per tradurre i nomi di dominio (es. [www.google.com](http://www.google.com)) in indirizzi IP.

Il livello di Trasporto ha il compito fondamentale di gestire la comunicazione tra due processi (applicazioni) su host diversi. Fornisce un canale logico per il flusso dei dati, garantendo che arrivino a destinazione in modo affidabile e ordinato, se richiesto.

Funzioni principali:

- **Segmentazione dei dati:** Suddivide i dati provenienti dal livello di Applicazione in segmenti più piccoli e gestibili.
- **Controllo di flusso:** Regola la quantità di dati che possono essere inviati per evitare di sovraccaricare il destinatario.
- **Controllo di errore e affidabilità:** Si assicura che tutti i segmenti arrivino a destinazione senza errori, in ordine e senza duplicati.

Protocolli principali:

- **TCP (Transmission Control Protocol):** È un protocollo orientato alla connessione e affidabile. Stabilisce una connessione (tramite un processo chiamato "three-way handshake") prima di inviare i dati e garantisce che ogni pacchetto inviato venga ricevuto correttamente, richiedendo la ritrasmissione in caso di perdita. È utilizzato dalla maggior parte delle applicazioni comuni come web e email.
- **UDP (User Datagram Protocol):** È un protocollo "connectionless" (senza connessione) e non affidabile. Invia i dati (datagrammi) senza stabilire una connessione preliminare e senza garantire la consegna. È più veloce del TCP ed è utilizzato per applicazioni dove la velocità è critica e la perdita di qualche pacchetto è tollerabile, come lo streaming video, i giochi online e le chiamate VoIP.

Questo livello è il cuore del modello TCP/IP. La sua responsabilità principale è l'instradamento dei pacchetti di dati attraverso la rete, dall'host di origine a quello di destinazione. Si occupa dell'indirizzamento logico e della determinazione del percorso migliore per i dati.

Funzioni principali:

- **Indirizzamento logico:** Assegna un indirizzo univoco (indirizzo IP) a ogni dispositivo sulla rete per identificarlo.
- **Instradamento (Routing):** Determina il percorso migliore che i pacchetti devono seguire per raggiungere la loro destinazione, anche attraverso reti diverse.
- **Frammentazione e riassemblaggio dei pacchetti:** Se un pacchetto è troppo grande per essere gestito da una rete intermedia, viene suddiviso in frammenti più piccoli, che verranno poi riassemblati a destinazione.

Protocolli principali:

- **IP (Internet Protocol):** È il protocollo fondamentale di questo livello. Si occupa di incapsulare i segmenti del livello di trasporto in pacchetti (o datagrammi) e di aggiungere le informazioni di indirizzamento (IP di origine e destinazione). Esistono due versioni principali: IPv4 e IPv6.
- **ICMP (Internet Control Message Protocol):** Utilizzato per inviare messaggi di errore e di controllo, come quelli generati dal comando ping per verificare la raggiungibilità di un host.
- **ARP (Address Resolution Protocol):** Utilizzato per risolvere (tradurre) un indirizzo IP in un indirizzo fisico (MAC address) all'interno di una rete locale.

# Livello di Accesso alla Rete (Network Access Layer)

Questo è il livello più basso del modello TCP/IP e si occupa di tutti gli aspetti legati alla trasmissione fisica dei dati sulla rete locale. Corrisponde ai livelli Fisico e di Collegamento Dati (Data Link) del modello OSI.

Funzioni principali:

- **Interfacciamento con l'hardware:** Gestisce la comunicazione diretta con la scheda di rete (NIC) e il mezzo fisico (cavo Ethernet, onde radio per il Wi-Fi, ecc.).
- **Indirizzamento fisico (MAC):** Utilizza gli indirizzi MAC (Media Access Control) per identificare i dispositivi all'interno della stessa rete locale.
- **Framing:** Organizza i bit provenienti dal livello fisico in unità logiche chiamate "frame".
- **Trasmissione fisica:** Converte i frame in segnali elettrici, luminosi o radio per la trasmissione sul mezzo fisico.

Protocolli e standard comuni:

- **Ethernet:** Lo standard più diffuso per le reti locali cablate (LAN).
- **Wi-Fi (IEEE 802.11):** Lo standard per le reti locali senza fili (WLAN).
- **PPP (Point-to-Point Protocol):** Utilizzato per le connessioni dirette tra due nodi.



Oltre al modello TCP/IP, che è il modello operativo su cui si basa Internet, esiste un altro importante framework concettuale per comprendere le reti: il modello ISO/OSI.

## **Cos'è il Modello ISO/OSI?**

Il modello OSI (Open Systems Interconnection) è stato sviluppato dall'ISO (International Organization for Standardization). A differenza del TCP/IP, che è nato da esigenze pratiche ed è diventato uno standard de facto, il modello OSI è un modello di riferimento teorico. Il suo scopo principale era quello di creare uno standard universale per la progettazione di protocolli di rete, garantendo che sistemi di produttori diversi potessero comunicare tra loro (interoperabilità).

La sua caratteristica fondamentale è la suddivisione delle funzioni di rete in sette livelli (layer) astratti. Ogni livello ha un compito specifico e comunica solo con il livello immediatamente superiore e inferiore.

I livelli sono numerati dal basso verso l'alto. Un modo comune per ricordarli è la frase in inglese: "Please Do Not Throw Sausage Pizza Away".

## 7 - Livello di Applicazione (Application Layer)

- Scopo: Fornisce l'interfaccia diretta per le applicazioni utente. È il livello con cui l'utente finale interagisce.
- Funzioni: Protocolli per servizi specifici come la navigazione web, l'invio di email o il trasferimento di file. Esempi: HTTP, FTP, SMTP, DNS.

## 6 - Livello di Presentazione (Presentation Layer)

- Scopo: Assicurare che i dati inviati dal livello Applicazione di un sistema siano comprensibili dal livello Applicazione di un altro sistema.
- Funzioni:
  - Traduzione dati: Converte i dati in un formato intermedio standard (es. da ASCII a EBCDIC).
  - Crittografia/Decrittografia: Gestisce la cifratura dei dati per garantire la privacy.
  - Compressione: Comprime i dati per ridurre la quantità di bit da trasmettere.

## 5 - Livello di Sessione (Session Layer)

- Scopo: Stabilire, gestire e terminare le connessioni (chiamate "sessioni") tra applicazioni.
- Funzioni:
  - Controllo del dialogo: Decide chi può trasmettere e per quanto tempo.
  - Sincronizzazione: Inserisce dei "checkpoint" nel flusso dati, in modo che in caso di errore la trasmissione possa riprendere dall'ultimo checkpoint, invece che dall'inizio.

## 4 - Livello di Trasporto (Transport Layer)

- Scopo: Fornisce un trasporto dati affidabile e trasparente end-to-end, da un processo su un host a un processo su un altro host.
- Funzioni: Segmentazione dei dati, controllo di flusso, controllo degli errori. Protocolli principali: TCP (affidabile, orientato alla connessione) e UDP (veloce, non affidabile).

## 3 - Livello di Rete (Network Layer)

- Scopo: Gestire l'indirizzamento logico dei dispositivi e determinare il percorso migliore (routing) per i dati attraverso la rete.
- Funzioni: Indirizzamento logico (indirizzi IP), instradamento dei pacchetti tra reti diverse. Protocollo principale: IP (Internet Protocol).

## 2 - Livello di Collegamento Dati (Data Link Layer)

- Scopo: Fornire un transito affidabile dei dati attraverso un singolo collegamento fisico (locale).
- Funzioni:
  - Framing: Organizza i bit in unità logiche chiamate "frame".
  - Indirizzamento fisico: Utilizza gli indirizzi MAC per identificare i dispositivi sulla rete locale.
  - Controllo degli errori: Rileva (e a volte corregge) gli errori avvenuti nel livello Fisico.

## 1 - Livello Fisico (Physical Layer)

- Scopo: Gestire la trasmissione e la ricezione dei bit grezzi non strutturati su un mezzo fisico.
- Funzioni: Definisce le specifiche elettriche, meccaniche e funzionali del mezzo trasmissivo (cavi, tensioni, frequenze radio, connettori)

Nel modello ISO/OSI, ogni livello comunica principalmente con i livelli immediatamente adiacenti a sé (quello superiore e quello inferiore) sullo stesso sistema, fornendo servizi al livello superiore e utilizzando i servizi del livello inferiore.

Inoltre, un concetto fondamentale del modello OSI è la **comunicazione logica "peer-to-peer" (tra pari)**. Questo significa che ogni livello su un sistema (mittente) comunica concettualmente con il livello corrispondente sull'altro sistema (destinatario).

## Comunicazione verticale (tra livelli adiacenti)

Quando i dati vengono inviati, passano attraverso i livelli del mittente dall'alto verso il basso. Ogni **livello aggiunge le proprie informazioni** di controllo (intestazioni e/o trailer) ai dati ricevuti dal livello superiore, un **processo chiamato incapsulamento**. Poi passa il pacchetto risultante al livello sottostante.

Quando i dati vengono ricevuti, il processo è inverso: i dati risalgono i livelli del destinatario dal basso verso l'alto. Ogni livello rimuove le informazioni di controllo aggiunte dal livello corrispondente sul mittente, decapsulando i dati, e poi li passa al livello superiore.

- Il livello N fornisce servizi al livello N+1. Ad esempio, il livello di trasporto fornisce servizi al livello di sessione.
- Il livello N utilizza i servizi del livello N-1. Ad esempio, il livello di trasporto utilizza i servizi del livello di rete

Quando i dati vengono inviati dal mittente, il flusso è dall'alto verso il basso nei livelli del modello ISO/OSI (e anche nel TCP/IP).

Riprendiamo il processo corretto:

- **Livello Applicazione (7):** L'applicazione genera i dati (es. un'email).
- **Livello Presentazione (6):** I dati vengono formattati e, se necessario, compressi o crittografati.
- **Livello Sessione (5):** Viene stabilita, gestita e terminata la sessione di comunicazione.
- **Livello Trasporto (4):** I dati vengono segmentati (divisi in parti più piccole) e vengono aggiunte le informazioni per il controllo di flusso e degli errori (es. numeri di sequenza). Questo è il primo livello che aggiunge un'intestazione significativa al "payload" (i dati dei livelli superiori).
- **Livello Rete (3):** Ogni segmento viene incapsulato in un pacchetto (o datagramma) e vengono aggiunte le informazioni di indirizzamento logico (es. indirizzi IP) per l'instradamento attraverso la rete.
- **Livello Collegamento Dati (2):** Il pacchetto viene incapsulato in un frame e vengono aggiunte le informazioni per l'indirizzamento fisico (es. indirizzi MAC) e il controllo degli errori a livello locale. Qui possono essere aggiunti sia un'intestazione che un trailer.
- **Livello Fisico (1):** Il frame viene convertito in un flusso di bit e trasmesso sul mezzo fisico (cavi, onde radio, ecc.).

Questo processo di aggiungere informazioni di controllo a ogni livello mentre i dati scendono, abbiamo detto, è chiamato incapsulamento. Quando i dati arrivano al destinatario, il processo è inverso: i dati risalgono i livelli dal basso verso l'alto, e ogni livello rimuove le intestazioni (e i trailer, se presenti) aggiunte dal livello corrispondente del mittente, un processo chiamato decapsulamento, fino a quando i dati originali non raggiungono l'applicazione

## Comunicazione orizzontale (logica "peer-to-peer")

Sebbene la comunicazione fisica avvenga solo verticalmente attraverso lo stack di protocolli, ogni livello opera come se stesse comunicando direttamente con il suo "pari" sul sistema remoto. Questo avviene attraverso l'uso di protocolli. Ogni protocollo definisce le regole e il formato dei dati per la comunicazione tra due entità dello stesso livello su sistemi diversi.

Ad esempio:

- Il livello di rete su un computer comunica logicamente con il livello di rete sull'altro computer per instradare i pacchetti.
- Il livello di trasporto su un computer comunica logicamente con il livello di trasporto sull'altro computer per garantire l'affidabilità della consegna dei dati.

Questo approccio a strati rende il modello OSI molto utile per la comprensione e la risoluzione dei problemi delle reti, poiché le responsabilità sono chiaramente separate e definite per ogni livello.

Sebbene il mondo operi sul modello TCP/IP, il modello OSI è estremamente utile per capire le reti in modo più granulare. La mappatura tra i due modelli può essere descritta come segue:

- **Livello Applicazione (TCP/IP):** Questo livello raggruppa le funzioni dei tre livelli superiori del modello OSI:
  - Livello 7 - Applicazione (OSI): Fornisce l'interfaccia e i servizi di rete per le applicazioni utente (es. HTTP, SMTP).
  - Livello 6 - Presentazione (OSI): Gestisce la formattazione, la crittografia e la compressione dei dati.
  - Livello 5 - Sessione (OSI): Stabilisce, gestisce e termina il dialogo tra gli host.
- **Livello Trasporto (TCP/IP):** Corrisponde direttamente al Livello 4 - Trasporto (OSI). Entrambi si occupano della comunicazione end-to-end affidabile e del controllo di flusso, utilizzando protocolli come TCP e UDP.
- **Livello Internet (TCP/IP):** Coincide con il Livello 3 - Rete (OSI). Entrambi gestiscono l'indirizzamento logico (IP), l'instradamento e la determinazione del percorso dei pacchetti attraverso le reti.
- **Livello di Accesso alla Rete (TCP/IP):** Questo livello unisce le responsabilità dei due livelli inferiori del modello OSI:
  - Livello 2 - Collegamento Dati (OSI): Gestisce l'indirizzamento fisico sulla rete locale (MAC address) e il framing.
  - Livello 1 - Fisico (OSI): Si occupa della trasmissione fisica dei bit attraverso il mezzo trasmissivo (cavi, Wi-Fi).

## Differenze principali:

- Modello Teorico vs. Pratico: OSI è un modello di riferimento generico, mentre TCP/IP è il modello su cui Internet è stato effettivamente costruito.
- Numero di Livelli: OSI ha 7 livelli, offrendo una separazione più netta delle funzioni, mentre TCP/IP ne raggruppa diverse in 4 livelli.

Il principio di comunicazione tra i livelli nel modello TCP/IP è molto simile a quello del modello OSI, sebbene ci siano alcune differenze nella suddivisione e nel numero dei livelli.

## Nel modello TCP/IP, come nell'OSI:

- Comunicazione verticale (tra livelli adiacenti): I dati, quando vengono inviati, passano attraverso i livelli del mittente dall'alto verso il basso. Ogni livello aggiunge le proprie informazioni di controllo (intestazioni) ai dati ricevuti dal livello superiore (processo di incapsulamento) e poi li passa al livello sottostante. Al contrario, quando i dati vengono ricevuti, risalgono i livelli del destinatario dal basso verso l'alto, e ogni livello rimuove le intestazioni pertinenti (processo di decapsulamento) prima di passare i dati al livello superiore.
- Comunicazione orizzontale (logica "peer-to-peer"): Concettualmente, ogni livello su un sistema comunica con il livello corrispondente sull'altro sistema. Questa comunicazione "tra pari" è definita dai protocolli specifici di quel livello. Ad esempio, il protocollo TCP (Transmission Control Protocol) opera a livello di trasporto e gestisce la comunicazione affidabile tra i processi di trasporto sui due sistemi.

## Differenze principali tra TCP/IP e OSI:

- Numero di livelli: Il modello TCP/IP è generalmente rappresentato con 4 o 5 livelli, mentre il modello OSI ne ha 7.
  - TCP/IP (4 livelli): Applicazione, Trasporto, Internet, Accesso alla Rete.
  - TCP/IP (5 livelli): Applicazione, Trasporto, Rete (Internet), Collegamento Dati, Fisico. (Questa è una suddivisione più dettagliata che separa il livello di accesso alla rete in collegamento dati e fisico, rendendolo più simile all'OSI per i livelli inferiori).
  - OSI (7 livelli): Applicazione, Presentazione, Sessione, Trasporto, Rete, Collegamento Dati, Fisico.
- Combinazione di livelli: Il modello TCP/IP combina alcune funzionalità che nell'OSI sono separate:
  - Il livello di Applicazione del TCP/IP include le funzionalità dei livelli di Applicazione, Presentazione e Sessione dell'OSI.
  - Il livello di Accesso alla Rete (o Collegamento Dati + Fisico) del TCP/IP include le funzionalità dei livelli di Collegamento Dati e Fisico dell'OSI.

In sintesi, anche se i nomi e il numero dei livelli cambiano, il principio fondamentale di una comunicazione stratificata, con interazioni verticali tra livelli adiacenti e comunicazioni logiche orizzontali tra livelli corrispondenti (peer-to-peer), rimane valido in entrambi i modelli.




La realizzazione di un'applicazione web coinvolge solamente il livello più alto del protocollo (application)

Per le applicazioni a pagine è previsto il protocollo HTTP(S)

Per le applicazioni a servizi non esisteva un protocollo specifico e si è deciso di utilizzare ancora HTTP(S)



HTTPS sta per HyperText Transfer Protocol Secure. Non è un protocollo separato, ma è il risultato della sovrapposizione del protocollo HTTP a un livello di sicurezza aggiuntivo chiamato SSL/TLS. In pratica, è lo stesso protocollo HTTP, ma con tutte le comunicazioni tra client e server protette da crittografia.

Visualmente, un sito che usa HTTPS è immediatamente riconoscibile dall'icona a forma di lucchetto  nella barra degli indirizzi del browser e dal prefisso https:// nell'URL.

Il livello di sicurezza è garantito da due protocolli storicamente successivi:

- SSL (Secure Sockets Layer): La versione originale, oggi considerata obsoleta e insicura.
- TLS (Transport Layer Security): Il successore di SSL. È lo standard attuale, più robusto e sicuro. Sebbene oggi si usi quasi esclusivamente TLS, è comune usare ancora il termine "SSL" per indicare il certificato di sicurezza.

HTTPS fornisce tre garanzie di sicurezza fondamentali:

1. **Crittografia (Encryption):** Rende i dati illeggibili a chiunque cerchi di intercettarli. Se un malintenzionato "ascolta" la comunicazione tra il tuo browser e il server, vedrà solo una sequenza di caratteri senza senso. Questo si ottiene tramite due tipi di crittografia che lavorano insieme:
  - Crittografia Asimmetrica (a chiave pubblica/privata): Usata all'inizio della comunicazione per scambiare in modo sicuro la chiave di sessione. Il server ha una chiave pubblica (che tutti possono vedere) e una chiave privata (segreta). I dati crittografati con la chiave pubblica possono essere decifrati solo con la chiave privata corrispondente.
  - Crittografia Simmetrica (a chiave condivisa): Molto più veloce di quella asimmetrica. Una volta che client e server hanno stabilito una chiave segreta condivisa (usando la crittografia asimmetrica), la usano per crittografare tutti i dati scambiati per il resto della sessione.
2. **Autenticazione (Authentication):** Garantisce che stai comunicando esattamente con il server a cui intendi connetterti (es. google.com) e non con un impostore. Questo processo si basa sul Certificato SSL/TLS.
3. **Integrità (Integrity) :** Assicura che i dati inviati non siano stati alterati o manomessi durante il transito. Ogni messaggio viene accompagnato da un "Message Authentication Code" (MAC), una sorta di firma digitale che permette al destinatario di verificare che il messaggio sia arrivato intatto.

Il processo che stabilisce una connessione sicura è chiamato TLS Handshake. Ecco una versione semplificata dei passaggi:

1. **Client Hello:** Il tuo browser (client) contatta il server e dice: "Ciao, vorrei stabilire una connessione sicura. Queste sono le versioni di TLS e gli algoritmi di crittografia che supporto".
2. **Server Hello:** Il server risponde: "Ciao. D'accordo, usiamo questa versione di TLS e questo algoritmo. Ecco il mio Certificato SSL/TLS e la mia chiave pubblica".
3. **Verifica del Certificato:** Il browser controlla il certificato del server. Un certificato SSL/TLS è come una carta d'identità digitale. Contiene informazioni sul proprietario del dominio ed è stato firmato digitalmente da un'entità fidata chiamata Certificate Authority (CA) (es. Let's Encrypt, DigiCert). Il browser verifica che:
  - Il certificato non sia scaduto.
  - Sia stato emesso per il dominio corretto.
  - La firma della CA sia valida (il browser ha un elenco preinstallato di CA attendibili).
4. **Scambio della Chiave di Sessione:** Se il certificato è valido, il browser si fida del server. A questo punto, il browser genera una chiave per la crittografia simmetrica (la "chiave di sessione"), la crittografa usando la chiave pubblica del server e la invia.
5. **Inizio della Sessione Sicura:** Solo il server, con la sua chiave privata, può decifrare il messaggio e ottenere la chiave di sessione. Ora, sia il client che il server possiedono la stessa chiave segreta. Da questo momento in poi, tutta la comunicazione avviene in modo rapido e sicuro tramite crittografia simmetrica.

Ottenere un certificato TLS/SSL è il passo fondamentale per abilitare HTTPS. Storicamente, questo processo era manuale e costoso. L'arrivo di Let's Encrypt ha rivoluzionato questo panorama.

## Cos'è Let's Encrypt?

Let's Encrypt è una Certificate Authority (CA) globale, non-profit, la cui missione è rendere il web un luogo più sicuro per tutti. Per raggiungere questo obiettivo, fornisce certificati TLS in modo gratuito e automatizzato. Il progetto è sostenuto da importanti organizzazioni come Electronic Frontier Foundation (EFF), Mozilla, Cisco, Google Chrome e altre.

Le caratteristiche chiave di Let's Encrypt sono:

- **Gratuito:** Chiunque possieda un nome di dominio può ottenere un certificato affidabile senza alcun costo.
- **Automatizzato:** L'intero processo di richiesta, convalida e rinnovo del certificato è pensato per essere gestito da software, senza intervento umano. Questo avviene tramite il protocollo ACME (Automated Certificate Management Environment).
- **Sicuro:** I certificati Let's Encrypt usano gli stessi standard crittografici dei certificati a pagamento, offrendo un livello di sicurezza identico per la cifratura dei dati.
- **Durata Breve:** I certificati hanno una validità di soli 90 giorni. Questo non è un difetto, ma una scelta di sicurezza: incoraggia l'automazione del rinnovo e riduce la finestra di danno in caso di compromissione di una chiave.

I certificati tradizionali sono emessi da **Certificate Authority (CA)** commerciali come DigiCert, Sectigo (ex Comodo), GlobalSign, ecc. Questi servizi offrono una gamma più ampia di prodotti e funzionalità rispetto a Let's Encrypt, solitamente dietro il pagamento di un canone annuale.

È fondamentale chiarire che, contrariamente a quanto si potrebbe pensare, i certificati SSL/TLS sono legati quasi esclusivamente ai nomi di dominio, non agli indirizzi IP. Lo scopo di un certificato è autenticare l'identità di un sito, e sul web l'identità è rappresentata dal nome di dominio (es. [www.miosito.it](http://www.miosito.it)). Il browser verifica che il nome nel certificato corrisponda a quello del sito visitato; in caso contrario, segnala un errore di sicurezza.

Legare i certificati agli IP sarebbe impraticabile a causa dell'hosting condiviso, dove un singolo IP ospita centinaia di siti. La tecnologia che rende possibile l'HTTPS in questi scenari è la Server Name Indication (SNI), un'estensione di TLS che permette al browser di comunicare al server il dominio a cui si vuole connettere, consentendo al server di presentare il certificato corretto.

Quando una Certificate Authority (CA) emette un certificato, esegue un processo di verifica per assicurarsi che il richiedente sia chi dice di essere. Il livello di approfondimento di questa verifica determina il tipo di certificato.

## 1. Domain Validation (DV)

È il livello di validazione più basilare e comune. Il processo è completamente automatizzato e verifica solo che il richiedente abbia il controllo amministrativo del nome di dominio (ad esempio, dimostrando di poter ricevere email a admin@miodominio.it o di poter modificare i record DNS del dominio).

**Perché si usa:** È estremamente rapido (richiede pochi minuti), facile da ottenere e ideale per proteggere la comunicazione con la crittografia. Non fornisce alcuna informazione sull'identità del proprietario del sito.

**Chi lo usa:** È il tipo di certificato emesso da Let's Encrypt. È perfetto per blog, siti personali, piccole imprese e qualsiasi sito in cui non sia critica la verifica dell'identità legale dell'organizzazione.

## 2. Organization Validation (OV)

Questo livello include la verifica del dominio (DV), ma aggiunge un passo di verifica manuale da parte della CA. Un operatore umano controlla l'esistenza e la legittimità dell'organizzazione che richiede il certificato, consultando database aziendali e registri pubblici.

**Cosa mostra:** Il certificato OV contiene il nome verificato dell'organizzazione e la sua località (città, paese). Gli utenti possono visualizzare queste informazioni cliccando sul lucchetto nella barra degli indirizzi del browser.

**Perché si usa:** Per aumentare la fiducia degli utenti. Dimostra che il sito non è gestito da un'entità anonima, ma da un'organizzazione legale e registrata.

**Chi lo usa:** Aziende, siti di e-commerce e organizzazioni che vogliono un livello di fiducia superiore rispetto al semplice DV.



## 3. Extended Validation (EV)

È il livello di validazione più rigoroso e standardizzato. La CA esegue un processo di verifica molto approfondito, definito da linee guida severe, che include il controllo dello stato legale, operativo e fisico dell'organizzazione, nonché la verifica che la richiesta del certificato sia stata autorizzata dall'azienda stessa.

**Cosa mostra:** In passato, i browser premiavano i siti con certificati EV mostrando una vistosa "barra verde" con il nome dell'azienda direttamente nella barra degli indirizzi. Oggi, questa interfaccia è stata per lo più rimossa, ma le informazioni sull'identità legale dell'azienda rimangono le più dettagliate e facilmente accessibili cliccando sul lucchetto.

**Perché si usa:** Per offrire il massimo livello di fiducia e garanzia sull'identità del proprietario del sito. È un potente strumento contro il phishing, poiché un malintenzionato non potrebbe mai superare un processo di verifica così stringente.

**Chi lo usa:** Banche, istituzioni finanziarie, grandi piattaforme di e-commerce e agenzie governative, dove la fiducia dell'utente è assolutamente critica.

## Costo

- Let's Encrypt: Totalmente gratuito.
- Tradizionali: A pagamento, con costi variabili in base al tipo di certificato e al fornitore.

## Processo di Ottenimento e Automazione

- Let's Encrypt: Progettato per essere completamente automatico tramite il protocollo ACME. Un client software gestisce richiesta, convalida e rinnovo.
- Tradizionali: Processo spesso manuale o semi-manuale che richiede la generazione di un CSR, l'invio alla CA e l'installazione manuale.

## Periodo di Validità

- Let's Encrypt: 90 giorni, una scelta che incentiva l'automazione e aumenta la sicurezza.
- Tradizionali: Tipicamente 1 anno, riducendo la frequenza dei rinnovi manuali.

## Livelli di Validazione

- Let's Encrypt: Offre solo la Domain Validation (DV), una verifica automatica che attesta unicamente il controllo del dominio.
- Tradizionali: Offrono anche la Organization Validation (OV), che verifica l'identità legale dell'azienda, e la Extended Validation (EV), un processo di verifica ancora più rigoroso.

## Garanzia e Supporto Tecnico

- Let's Encrypt: Non offre garanzie finanziarie e il supporto è gestito dalla comunità tramite forum e documentazione.
- Tradizionali: Offrono garanzie monetarie in caso di emissione errata e forniscono supporto tecnico dedicato (telefono, chat, email).

## **Scegli Let's Encrypt se:**

- Gestisci un sito personale, un blog, un portfolio o il sito di una piccola impresa.
- Devi proteggere API, servizi backend o ambienti di sviluppo/staging.
- Apprezzi l'automazione e vuoi eliminare la gestione manuale dei certificati.
- Il tuo budget è una considerazione prioritaria.

## **Considera un Certificato Tradizionale (OV/EV) se:**

- Gestisci un grande sito di e-commerce, una piattaforma di pagamenti, un sito bancario o di un'istituzione governativa.
- La fiducia dell'utente, basata sulla verifica dell'identità legale della tua azienda, è un fattore critico di business.
- Hai bisogno di una garanzia finanziaria che protegga te e i tuoi clienti.
- Necessiti di accesso a un supporto tecnico dedicato.

# Perché HTTPS è Fondamentale per le API REST?

Nel contesto delle API REST, usare HTTP semplice è un errore di sicurezza gravissimo. L'uso di HTTPS non è opzionale, è un requisito indispensabile.

- **Protezione dei Dati Sensibili:** Le API trasportano dati. Che si tratti di dati personali degli utenti, credenziali di accesso, informazioni di pagamento o segreti aziendali, questi viaggiano nel body o nei parametri della richiesta/risposta. Senza HTTPS, sarebbero in chiaro, facilmente intercettabili.
- **Sicurezza dei Token di Autenticazione:** Molte API REST usano token (es. Bearer Token, JWT) inviati nell'header Authorization per autenticare le richieste. Se un token venisse intercettato su una connessione HTTP, un malintenzionato potrebbe usarlo per impersonare l'utente e accedere senza restrizioni alle sue risorse.
- **Prevenzione di Attacchi "Man-in-the-Middle" (MITM):** Con HTTP, un attaccante potrebbe posizionarsi tra il client e il server, non solo per leggere ma anche per modificare le richieste e le risposte. Potrebbe, ad esempio, cambiare l'importo di una transazione finanziaria o iniettare dati malevoli nella risposta del server. HTTPS rende questo tipo di attacco quasi impossibile.
- **Affidabilità e Professionalità:** Esporre un'API su HTTP denota una mancanza di attenzione alla sicurezza. Molti client (specialmente le applicazioni mobile e i browser moderni) bloccano di default le richieste a risorse non sicure (chiamate mixed content), rendendo di fatto l'API inutilizzabile.

Il server offre servizi ed è in attesa del client

Il client chiede un servizio indicando:

- **URL** -> protocollo, ip, porta e path verso la risorsa da agganciare

http://IP:port/path/resource

- **VERB** -> sono parole codificate dal protocollo che servono a indicare il tipo di azione che si vuole compiere sulla risorsa indicata dalla URL

In ambito REST si considerano solo le seguenti:

- GET -> lettura
- POST -> inserimento
- PUT -> modifica totale
- PATCH -> modifica parziale
- DELETE -> cancellazione