Introducció a la Programació Multiplataforma

Programació Multiplataforma i Distribuïda

Grau d'informàtica. EPSEVG

Setembre 2023

Jordi Esteve jesteve@cs.upc.edu

Programari Multiplataforma (cross-platform or multi-platform)

Aquell programari que, sigui un sistema operatiu, llenguatge de programació o programa, pot ésser executat en diverses plataformes.

Exemples:

- Una aplicació que sigui capaç d'executar-se a Windows, Linux i Mac OS X.
- O sobre diferents tecnologies mòbils: Android, iPhone.

Exemples: LibreOffice/OpenOffice, Mozilla Firefox, Mozilla Thunderbird, GIMP, Skype, ... (generalment de programari lliure).

Tipus de programari multiplataforma

- **Programari binari**: Requereix compilació individual per a cada plataforma que es dóna suport.

- **Programari fet amb llenguatges interpretats multiplataforma**: S'executa directament en qualsevol plataforma sense preparació especial, per exemple, el programari escrit en un llenguatge interpretat o bytecode portable pre-compilats que disposa d'intèrprets comuns o estàndards en vàries plataformes.

- **Programari client-servidor**: El client s'executa dins d'una aplicació multiplataforma, per exemple una aplicació web que fa ús com a clients els navegadors web estandaritzats.

A l'assignatura PMUD ens centrarem exclusivament en el programari multiplataforma basat en client-servidor web.

Una plataforma és una combinació de maquinari i programari que s'utilitza per executar aplicacions. Pot ser un sistema operatiu o una arquitectura d'ordinador, o la combinació d'ambdós.

Plataformes de maquinari

Una plataforma de maquinari es pot referir a l'arquitectura d'un ordinador o de l'arquitectura del processador.

Exemples: Les CPUs x86 i x86-64 (la més usual en PCs), PowerPC (antics Apple PCs), ARM (comuna en mòbils i tauletes).

Plataformes de programari

Les plataformes de programari pot ser un sistema operatiu o un entorn de programació, normalment es tracta d'una combinació de tots dos. Una excepció a això és Java, que utilitza una màquina virtual independent del sistema operatiu per al seu codi compilat (byte-code).

- Sistemes DOS al x86: MS-DOS, IBM PC DOS, DR-DOS, FreeDOS, ...
- Microsoft Windows (x86, x86-64, ARM)
- Linux (x86, x86-64, PowerPC, i altres arquitectures)
- BSD, sistema operatiu UNIX molt multiplataforma (NetBSD, per exemple)
- Mac OS X (x86, x86-64, PowerPC)
- Java+Màquina virtual
- Android i iPhone són dues plataformes diferents per a telèfons intel·ligents i tauletes

Programari binari

Tradicionalment el programari d'aplicació s'ha distribuït als usuaris finals com arxius binaris, especialment arxius executables. Els executables només suporten el sistema operatiu i l'arquitectura de computadors on van ser construïts o compilats.

El programador ha de construir/compilar el programari per a cada sistema operatiu i arquitectura diferent.

Exemple: Els navegadors Firefox o Chrome estan disponibles en Windows, OS X (PowerPC i x86) i Linux en múltiples arquitectures. Són distribucions executables diferents, tot i que vénen des del mateix codi font que s'ha adaptat per ser compilat en plataformes diferents.

Scripts i llenguatges interpretats

Una seqüència d'instruccions (script) pot ser considerat multiplataforma si el seu intèrpret està disponible en múltiples plataformes i l'script només utilitza les facilitats proporcionades pel llenguatge.

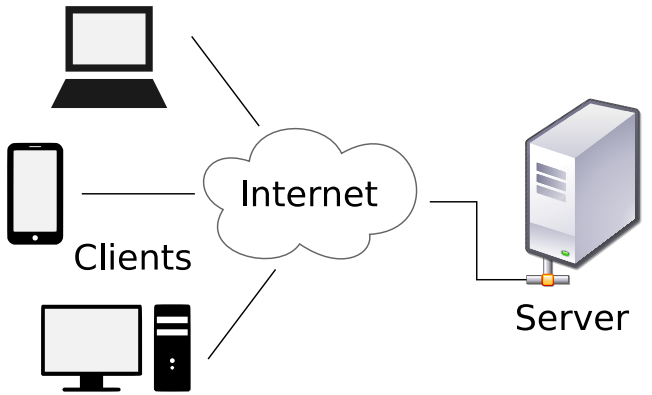
Exemple: Un script escrit en Python per a un sistema Unix-Linux és probable que s'executi amb poca o cap modificació en Windows, ja que Python té intèrprets per a moltes plataformes.

- **bash**: Shell d'Unix. Normalment s'executen a Linux i a altres sistemes Unix, així com en Windows a través de la capa de compatibilitat POSIX Cygwin.
- **Perl**: Llenguatge de scripting creat 1987. S'utilitza per a la programació CGI WWW, petites tasques d'administració del sistema i altres.
- **PHP**: Llenguatge de programació molt usat per aplicacions web, que també pot usar-se per realitzar scripts o aplicacions independents.
- **Python**: Llenguatge de scripting modern, centrat en el desenvolupament ràpid d'aplicacions i facilitat d'escriptura/lectura dels programes, en lloc de l'eficiència en temps d'execució.
- **JavaScript**: Llenguatge de scripting orientat a objectes que tots els navegadors web saben interpretar. També es pot utilitzar per crear aplicacions o servidors web a través de node.

Aplicacions Web

Les aplicacions web es consideren multiplataforma, ja que, idealment, són accessibles des de qualsevol dels diferents navegadors en diferents sistemes operatius.

Usen una arquitectura de sistema de tipus client-servidor i poden variar àmpliament en complexitat i funcionalitat.



Font: Wikipedia Llic: LGPL

Aquesta àmplia variabilitat complica l'objectiu de ser multiplataforma:

- No és fàcil fer aplicacions de funcionalitat avançada que siguin alhora multiplataforma.
- Fins i tot versions diferents del mateix navegador web (dins del mateix sistema operatiu) poden diferir considerablement entre si.
- Això està canviant amb l'adopció dels darrers estàndards (HTML5, CSS3, JavaScript) en els navegadors web més recents.

Aplicacions Web

Aplicacions web bàsiques

Realitzen tot o la major part de processament en un servidor sense estat, i passa el resultat al navegador web del client. Tota la interacció de l'usuari amb l'aplicació consisteix en simples intercanvis de sol·licituds de dades i les respostes del servidor. Segueixen un model simple de transaccions idèntic a la de servir pàgines web estàtiques.

Eren habituals en els primers temps del desenvolupament d'aplicacions web encara que avui en dia segueixen sent relativament comunes, especialment quan la compatibilitat entre plataformes i la simplicitat es considera més important que les funcionalitats avançades.

Aplicacions web avançades

Depenen de les funcions addicionals que es troben només en les versions més recents dels navegadors web: Ajax, JavaScript, HTML dinàmic, SVG, ...

Realitzen part del processament en el client gràcies a l'execució de codi JavaScript i interactuen amb el servidor quan és estrictament necessari, millorant notablement la interacció i disminuint l'intercanvi de dades entre client i servidor. Exemples: Gmail i Google docs.

Desafiaments pel desenvolupament multiplataforma

- Jocs de prova poden ser considerablement més complicats, ja que diferents plataformes poden exhibir comportaments lleugerament diferents o errors subtils.
- Els desenvolupadors sovint usen el subconjunt més baix del denominador comú de característiques que estan disponibles a totes les plataformes. Això pot afectar al rendiment de l'aplicació o prohibir l'ús de les funcions més avançades de cada plataforma.
- Les diferents plataformes sovint tenen diferents convencions d'interfície d'usuari i les aplicacions multiplataforma no sempre s'adapten a elles. Exemple: Les aplicacions desenvolupades per a Mac OS X i GNOME col·loquen el botó més important al costat dret d'una finestra o quadre de diàleg, mentre que Microsoft Windows i KDE ho fan a l'esquerra.
- Cada vegada que s'executa una aplicació escrita en llenguatges de script o sobre una màquina virtual ha de ser traduïda al codi executable natiu, imposant una penalització de rendiment.
- Diferents plataformes requereixen l'ús de formats de paquets nadius com RPM, DEB o MSI. Instal·ladors multiplataforma com InstallAnywhere, Jexpress o InstallBuilder solucionen aquesta necessitat.
- Entorns d'execució multiplataforma poden patir falles de seguretat multiplataforma, creant un entorn fèrtil per el malware multiplataforma.

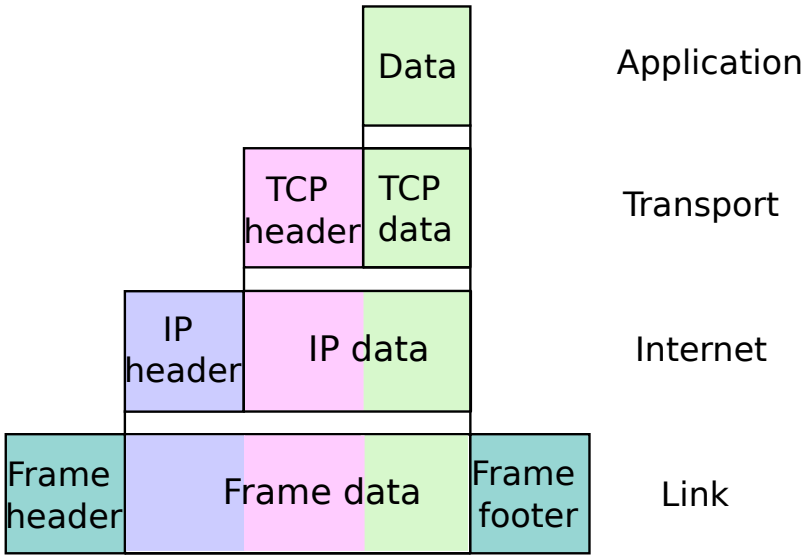
Internet

Internet (abreviació de **interconnected network**) és un sistema global d'interconnexió de xarxes d'ordinadors que usa la pila de protocols **TCP/IP**.

Les dades d'una aplicació s'encapsulen pel seu transport amb el **TCP** (Transmission Control Protocol) que assegura lliuraments fiables, ordenats i sense errors.

Al seu torn, els paquets TCP s'encapsulen dins paquets **IP** (Internet Protocol) especialitzats en enviar-los des d'un origen a un destí definit per una adreça IP.

En el darrer nivell els paquets IP són enviats entre dispositius físics amb enllaços físics (coure, fibra òptica, WIFI) usant xarxes físiques com per exemple Ethernet.

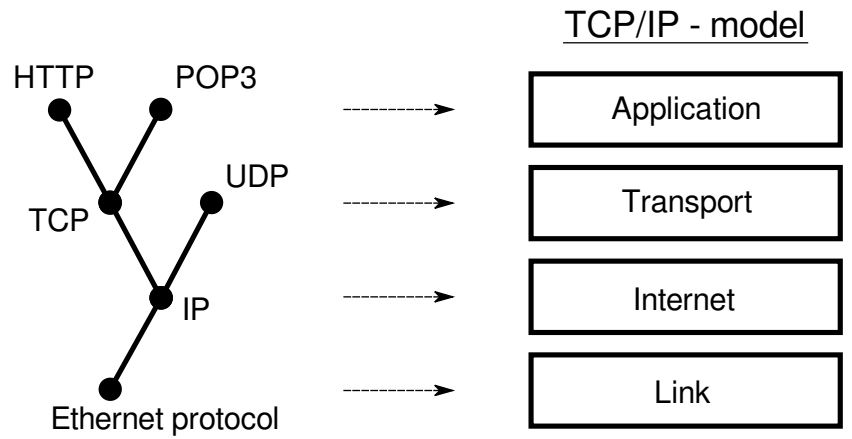


Font: https://en.wikipedia.org/wiki/File:UDP_encapsulation.svg. Llic: BY-SA3.0

Internet. Aplicacions

Segons el tipus d'aplicació a interconnectar s'usa un tipus de protocol a nivell d'aplicació diferent:

- Terminals virtuals: Telnet, SSH
- Transferència de fitxers: FTP, SCP
- Correu electrònic: SMTP, POP3, IMAP
- Web: HTTP, HTTPS



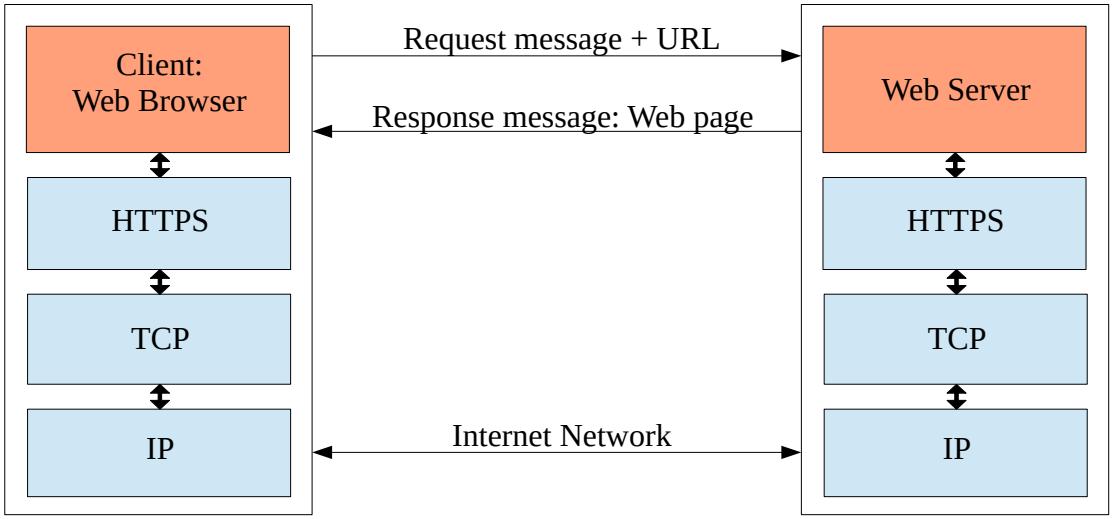
Font: https://en.wikipedia.org/wiki/File:Internet_layering.svg Llic: CC BY-SA3.0

WWW (la Web)

La **WWW** (World Wide Web) o la **Web** és un sistema d'informació accessible via internet a on els recursos o documents estan identificats per **URLs** (Uniform Resource Locators) i estan entrellaçats (**hypertext**) amb enllaços formant una teranyina mundial.

La Web, igual que la seva infraestructura de base internet, és descentralitzada i escalable, el que permet que creixi contínuament.

Els recursos/documents són oferts per servidors Web i consumits per clients anomenats navegadors web (**Web browsers**). Això permet fer aplicacions distribuïdes amb l'arquitectura client/servidor que es comuniquen amb els protocols HTTP/HTTPS sobre TCP/IP.



URL

URL (Uniform Resource Locator) o col·loquialment Web address.

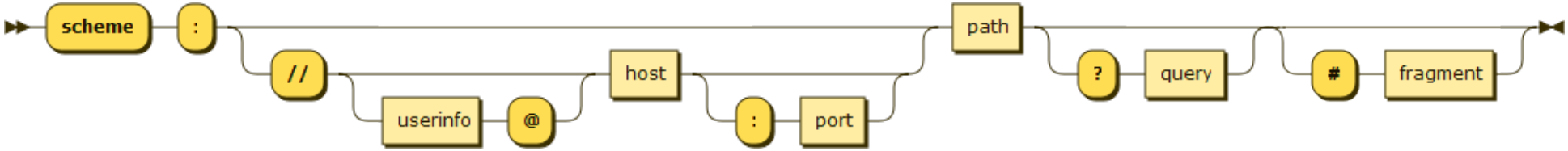
És una direcció única a un recurs o document en un servidor d'internet que indica la manera d'obtenir-lo.

Exemple: https://www.upc.edu/ca/graus/

Una URL (RFC1738) és un tipus de URI (Uniform Resource Identifier, RFC3986), un string que identifica unívocament un recurs en concret i que segueix aquestes regles:

scheme:[//[user:password@]host[:port]]path[?query][#fragment]

que gràficament equival a:



Font: https://en.wikipedia.org/wiki/File:URI_syntax_diagram.png. Llic: BY-SA4.0

El host (adreça IP) identifica l'ordinador a Internet i el port identifica l'aplicació dins de l'ordinador.

URL: scheme://user:password@host:port/path?query#fragment

https://upc.edu/dir/document.html

URL que identifica el recurs **document.html** de la carpeta **dir** del servidor **upc.edu**.

https://upc.edu:8000/dir/document.html

URL que identifica el recurs **document.html** de la carpeta **dir** del servidor **upc.edu** que escolta en el port **8000** enlloc del port 443 assignat per defecte als servidors web per escoltar el protocol https.

https://upc.edu/dir/document.html#seccio3

URL que identifica el fragment (o àncora) **seccio3** del recurs **document.html** de la carpeta **dir** del servidor **upc.edu**.

https://jordi:esteve@upc.edu/dir/document.html

URL igual a la primera però usant **jordi** com a identificador d'usuari i **esteve** com a contrasenya.

Es recomana enviar les contrasenyes en la URL només amb HTTPS. En HTTP és insegur.

https://upc.edu/dir/document.html?nom=Jordi&cognom=Esteve%20Cusin%C3%A9

URL que envia dos paràmetres (nom=''Jordi'' i cognom=''Esteve Cusiné'') en la query.

HTTP/HTTPS

The Web resources are accessed with the HTTP (Hypertext Transfer Protocol) or HTTPS (Hypertext Transfer Protocol Secure) protocols.

Both are application-level internet protocols over the TCP/IP protocols; they are designed to be simple and very scalable.

HTTPS is an extension of HTTP to offer a secure connection (a bidirectional end-to-end encryption). It allows the privacy and integrity of the data and it is essential when user authentication is required.

History:

HTTP/1.1 was used almost for 20 years, it was first documented in 1997.

HTTP/2 is a more efficient with a less latency version published in 2015. It is currently supported by most servers and web browsers.

HTTP/3 is the future successor of HTTP / 2, which uses UDP instead of TCP as a transport protocol.

HTTP messages

The **request message** consists:

- a request line (e.g., GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from the server using the GET method and the HTTP/1.1 protocol.)
- request header fields (e.g., Accept-Language: en).
- an empty line
- an optional message body

The **response message** consists:

- a status line which includes the status code and reason message (e.g., HTTP/1.1 200 OK, which indicates that the client's request succeeded.)
- response header fields (e.g., Content-Type: text/html)
- an empty line
- an optional message body

Font: Wikipedia

HTTP methods

GET: Requests a representation of the specified resource.

HEAD: Asks for a response identical to that of a GET request, but without the response body.

POST: Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI (e.g. an item to add to a database).

PUT: Requests that the enclosed entity be stored under the supplied URI. If the URI refers to an already existing resource, it is modified; if the URI does not point to an existing resource, then the server can create the resource with that URI.

DELETE: Deletes the specified resource.

TRACE: Echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers.

OPTIONS: Returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource.

PATCH: Applies partial modifications to a resource. Usually PATCH method is not available to avoid security problems.

Font: Wikipedia

HTTP. Exercises

1. Install the **curl** command line program on your laptop:

   ```
   sudo apt-get install curl
   ```

2. Open a web browser and visit https://ubiwan.epsevg.upc.edu

3. Open a terminal and make the same request using the curl program, analyzing the request and the response messages (if nothing is indicated, curl uses the GET method; the request header is show preceded by the > symbol, the response header is shown preceded by the < symbol).

   ```
   curl -v https://ubiwan.epsevg.upc.edu
   ```

4. Make similar requests using other methods and analyze the response messages:

   ```
   curl --request HEAD -v https://ubiwan.epsevg.upc.edu
   curl --request POST -v https://ubiwan.epsevg.upc.edu
   curl --request PUT -v https://ubiwan.epsevg.upc.edu
   curl --request DELETE -v https://ubiwan.epsevg.upc.edu
   ```

5. You can install a plugin for Chrome browser to make connection tests with a graphical interface, like Talend API Tester. Test it doing the same previous tests.

6. Or you can install an extension for Visual Studio Code editor like Thunder Client.

HTTP. Exercises

Test http://httpbin.org/ server, it echoes the data used in your request for any of these types:

- http://httpbin.org/ip Returns Origin IP.
- http://httpbin.org/user-agent Returns user-agent.
- http://httpbin.org/headers Returns header dict.
- http://httpbin.org/get Returns GET data.
- http://httpbin.org/post Returns POST data.
- http://httpbin.org/put Returns PUT data.
- http://httpbin.org/delete Returns DELETE data
- http://httpbin.org/gzip Returns gzip-encoded data.
- http://httpbin.org/status/:code Returns given HTTP Status code.
- http://httpbin.org/response-headers?key=val Returns given response headers.
- http://httpbin.org/cookies Returns cookie data.
- http://httpbin.org/cookies/set/:name/:value Sets a simple cookie.
- http://httpbin.org/basic-auth/:user/:passwd Challenges HTTPBasic Auth.
- http://httpbin.org/hidden-basic-auth/:user/:passwd 404'd BasicAuth.
- http://httpbin.org/digest-auth/:qop/:user/:passwd Challenges HTTP Digest Auth.
- http://httpbin.org/stream/:n Streams n lines.
- http://httpbin.org/delay/:n Delays responding for n seconds.

Note: You must change the parameters starting with : by a value, for ex http://httpbin.org/delay/2

HTML

HTML (Hypertext Markup Language) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

HTML is used to define the web pages with hyperlinks sent by the web servers that can be displayed in the web browsers.

In the early days of the Web, the web pages made with HTML were **static**: They were defined in text files that were directly sent by the web servers.

This has evolved in two ways:

- **Dynamic Web servers** use programming languages and databases to build versatile and changing HTML web pages.
- **Web clients** like the web browsers allow web pages to behave like **applications** thanks to the interaction offered by the JavaScript programming language and the refresh of the web page without the need to transfer it again by the Web server.

Web Server

Web servers can be physical machines or virtual machines in the cloud.

Most used web servers: Apache, Nginx, Microsoft-IIS, Node.js, ...

Dynamic web servers usually use frameworks to program web pages in several programming languages. Examples of most used frameworks for each language:

- Java: Spring MVC, Struts
- JavaScript: Node.js
- PHP: CakePHP, Symfony, Zend
- Python: Django, Flask
- Ruby: Ruby on Rails

Web servers can use several databases to store the information:

- Relational: PostgreSQL, MySQL, SQLserver, Oracle, ...
- Non-relational: MongoDB, Cassandra, ...

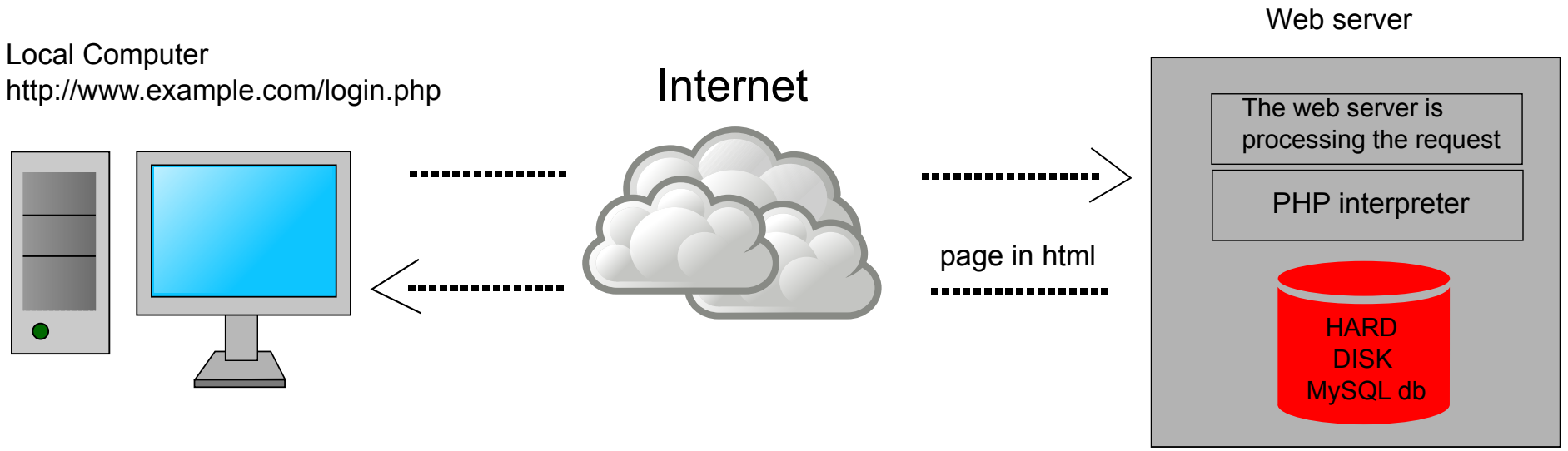
Web client

The web browser is the main client of access from a PC with Windows, GNU / Linux or MAC.

They are programmed in HTML, CSS and JavaScript and multiple JavaScript libraries.

Most used web browsers: Chrome, Firefox, Opera, Safari, Edge, ...

Many of the apps on mobile devices (Android, iPhone) also behave as web clients. They are currently the most used clients.



Font: https://en.wikipedia.org/wiki/File:Scheme_dynamic_page_en.svg Llic: CC-BY-SA3.0