CREACIÓ D'UN LLOC WEB D'EMMAGATZEMATGE DE FITXERS AL NÚVOL

Doublescloud.com

David Esteba Fàbrega

2n BAT A

Tecnologia, Toni Serrano

Índex

Introducció

En els últims anys hem pogut veure com els sistemes d'emmagatzematge al núvol han aconseguit molta popularitat i han desplaçat als sistemes clàssics d'emmagatzematge físic que s'havien utilitzat fins avui com els discs durs externs, els DVDs o les mateixes memòries USB.

Davant el gran creixement que han tingut aquests serveis, em vaig preguntar si seria capaç de crear una aplicació web que resolgués les necessitats d'un sistema d'emmagatzematge de fitxers al núvol. Per aquesta raó, vaig decidir proposar com a finalitat principal d'aquest treball, la programació d'una aplicació que respongués aquestes necessitats.

Els objectius principals proposats per aquest treball són:

- Conèixer els diferents llenguatges de programació que hi ha i aprendre els més adequats per dur a terme aquest treball.
- Preparar el servidor on s'allotjarà la pàgina web.
- Comprendre els diferents patrons de disseny utilitzats en el desenvolupament dels llocs web i utilitzar el que més s'adeqüi a les necessitats de l'aplicació.
- Conèixer les diferents vulnerabilitats que es poden donar en els llocs web i aprendre a solucionar-les.
- Programació de la pròpia aplicació.

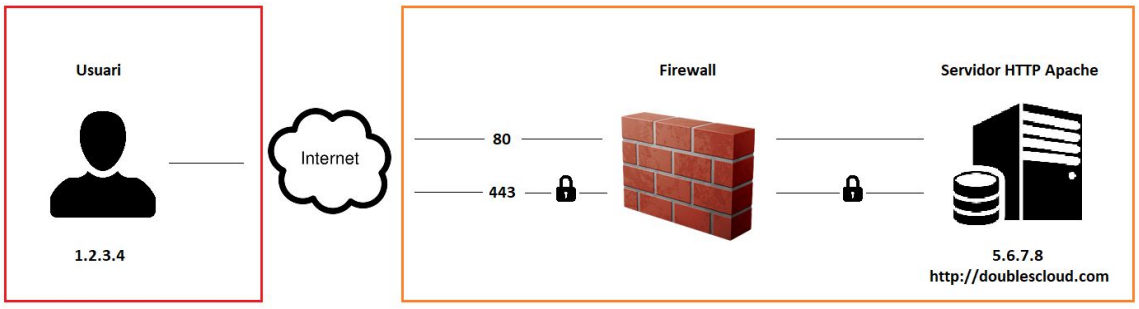
Una vegada em van donar llum verda per procedir amb el treball, vaig començar a pensar en els diferents aspectes citats, fins que vaig tenir una idea general i vaig poder passar a programar, per tal d'aconseguir crear un lloc d'emmagatzematge web que pugui servir com a alternativa a les diferents aplicacions que trobem al mercat actualment.

Funcionament extern de l'aplicació

En aquest apartat, ens centrarem en explicar els conceptes a tenir en compte a l'hora de preparar un servidor per tal que pugui allotjar una aplicació web. Hem decidit dividir-lo en tres parts:

- Una part dedicada a Internet, on s'expliquen conceptes bàsics per entendre com funciona la connexió entre l'usuari i el servidor.
- La part del servidor on s'expliquen elements necessaris pel correcte funcionament de l'aplicació.
- La part de l'usuari que explica com pot accedir a l'aplicació i interactuar-hi.

Per tal de dur a terme el següent apartat, ens basarem en l'esquema que tenim a continuació:



Imatge 1: Connexió de l'usuari al servidor a través d'Internet.

Internet

Introducció

A l'esquema anterior veiem dues parts ben diferenciades: una de color vermell que representa a l'usuari i l'altre de color taronja que representa la part del servidor. Totes dues parts es poden comunicar a través d'Internet.

Internet és una xarxa global on hi ha milions d'ordinadors connectats entre si, degut això, cada sistema connectat ha de tenir una adreça única. Les adreces d'Internet són del tipus: **nnn.nnn.nnn.nnn** on **nnn** ha de ser un número que vagi de **0 a 255**. Aquesta adreça es coneix amb el nom d'**adreça IP**.

Si s'observa la foto del principi, es pot veure com tant l'usuari com el servidor estan connectats a Internet i tots dos tenen la seva IP corresponent, en aquest cas l'usuari té la IP 1.2.3.4 i el servidor la IP 5.6.7.8.

Capes de protocol i paquets

Ja sabem com es connecten dos ordinadors a través d'Internet, però com ho fem per transmetre informació? Sabem que el nostre ordinador té la direcció IP 1.2.3.4 i volem enviar un missatge al servidor on l'aplicació web està allotjada. Primer de tot, si aquest missatge és llarg, s'haurà de dividir en diferents **paquets**, petites quantitats d'informació que permeten tractar el missatge amb més facilitat. Òbviament, el missatge s'ha de transmetre a través dels cables que es connecten amb Internet o bé a partir de tecnologies inalàmbriques com el Wi-Fi o 3G, per tant, si volem que el missatge es transmeti, s'ha de traduir de caràcters alfanumèrics a senyals electrònics, ser transmesos per Internet i tornar a ser traduït a caràcters alfanumèrics. Això es pot aconseguir gràcies a l'ús d'un conjunt de regles que utilitza Internet per tal de comunicar-se amb l'usuari i el servidor. Aquest conjunt de regles s'anomena **model TCP/IP**, que està estructurat de la següent manera:

Capes de protocol	Comentaris
Capa d'aplicació	Protocols específics per aplicacions com HTTP, FTP, e-mail, etc.
Capa de transmissió	TCP dirigeix els paquets a una aplicació determinada utilitzant un port determinat.
Capa d'Internet	La capa d'IP dirigeix els paquets al servidor fent ús del protocol IP.
Capa de hardware	Converteix paquets de dades alfanumèriques a senyals electrònics i a la inversa.

Protocols d'aplicació: HTTP (Hypertext Transfer Protocol)

Els **protocols d'aplicació** són els encarregats d'oferir un conjunt de regles per fer anar una aplicació determinada.

Un exemple d'aquests seria un dels serveis més utilitzats a Internet, el **World Wide Web** (WWW), que no és res més que la Web tradicional. El protocol d'aplicació que fa funcionar la Web és **HTTP**. Aquest protocol és l'utilitzat pels navegadors i els servidors web per tal de comunicar-se.

HTTP és un protocol orientat a la no connexió. Els **clients** (navegadors webs) envien sol·licituds als **servidors** per elements com pàgines webs o imatges. Un cop la sol·licitud ha obtingut resposta per part del servidor, la connexió entre els dos s'acaba. Això significa que cada cop que es vulgui realitzar una petició, s'haurà d'iniciar una nova connexió i així successivament.

Protocol TCP (Transmission Control Protocol)

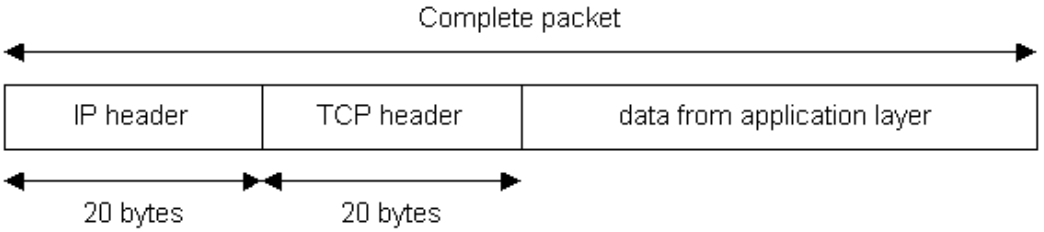
Sota la capa d'aplicació trobem la capa **TCP**. Quan les aplicacions comencen una connexió amb un altre ordinador, el missatge que vulguin enviar (fent servir un protocol de la capa d'aplicació) passa per la capa de **TCP**. **TCP** és el responsable de conduir els protocols de la capa d'aplicació a l'**aplicació correcta** a l'**ordinador de destinació**. Per tal de dur a terme això, es fan servir els **ports**. Podríem dir que els ports són com diferents canals que té un ordinador. Per exemple, podem navegar per Internet mentre mirem el correu. Això és degut a què les dues aplicacions fan servir **ports** diferents, si no només una aplicació podria fer ús de la connexió. Quan un paquet arriba a un ordinador i comença el seu camí a la capa de protocols, un cop arriba a la capa de **TCP**, aquest decideix quina aplicació ha de rebre el paquet.

Així doncs, **TCP** no és res més que el protocol que permet a dos amfitrions establir una connexió i intercanviar dades. **TCP** garantitza l'entrega de dades, és a dir, que les dades no es perdin durant la transmissió i també garanteix que els paquets siguin entregats en el mateix ordre en el que han estat enviats i a l'aplicació corresponent.

Protocol IP (Internet Protocol)

Com ja hem explicat al principi d'aquest apartat, les direccions **IPs** són com identificadors que tenim dins d'una xarxa. Per tant, el **protocol IP** és l'encarregat de guiar els paquets cap a la seva destinació, en aquest cas al servidor de l'aplicació web.

La imatge a continuació mostra com queda un paquet després d'haver passat per la capa d'**aplicació**, **TCP** i **IP**. Les dades es divideixen en paquets a la capa **TCP**, s'afegeix la capçalera, el paquet continua a la capa **IP**, s'afegeix la capçalera **IP** i després el paquet és transmès per Internet.



Imatge 2: Paquet un cop ha passat per totes les capes.

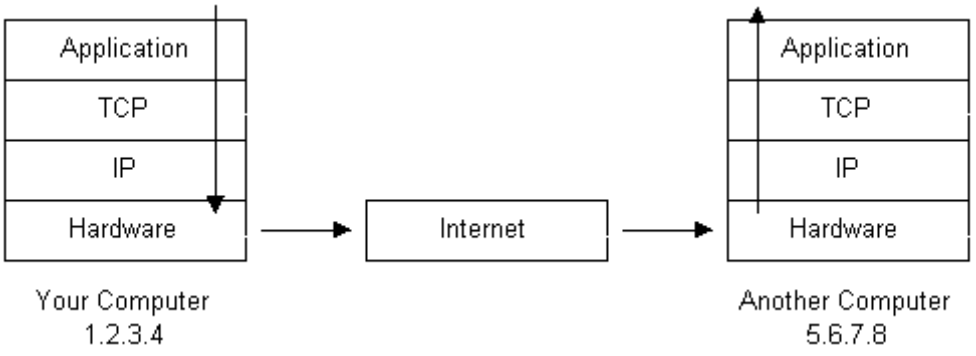
Capa de maquinari o hardware

La capa de **maquinari o hardware** és la capa més baixa del **model TCP/IP**. Aquesta és l'encarregada de convertir els paquets, els quals ja han passat per les capes d'aplicació, **TCP** i **IP**, de dades alfanumèriques a senyals electrònics i la l'inrevés.

Aquesta capa pot estar formada per diferents protocols, però el més usat, per no dir l'únic, en el món d'Internet és l'Ethernet. Aquest protocol és l'encarregat de definir les característiques elèctriques, la longitud i diàmetre dels cables, tots els elements dins d'una xarxa, com ha d'estar connectat, entre altres.

Posant-ho tot en funcionament

Ara que ja hem vist com funciona el **model TCP/IP**, anem a veure el camí que realitzaria el nostre missatge cap al servidor:



Imatge 3: Recorregut d'un paquet en el model TCP/IP.

1. El missatge començaria a la capa d'aplicació, on s'especificaria l'objectiu de la comunicació.
2. Un cop arriba a la capa **TCP**, aquesta comprova si el missatge a enviar és prou curt per enviar-lo en un sol paquet. Si no ho és, aquesta capa s'encarregaria de dividir la informació en diferents **paquets**.
3. Ja amb els diferents paquets creats, se li assignaria un port, que com ja hem explicat, dependria de l'aplicació que estiguéssim utilitzant.
4. Un cop hagi passat per la capa **TCP**, el paquets continuaran a la capa **IP**. Aquí és quan cada paquet rep l'adreça IP de la seva destinació.
5. Un cop els paquets tenen el **port** i l'adreça **IP**, estan preparats per ser enviats per Internet. La capa de **maquinari** serà l'encarregada de convertir els caràcters alfanumèrics que contenen els nostres paquets en senyals electròniques i transmetre'ls a través de la línia de telèfon.
6. Al final de la línia de telèfon, ens trobarem amb el router de l'**ISP (Internet Service Provider)** que examinarà la destinació de cada paquet i els enviarà a través d'Internet.
7. Al final, si tot ha anat bé, els paquets acabaran arribant a l'ordinador amb l'adreça **IP** 5.6.7.8. Allà, els paquets segueixen l'ordre invers del diagrama anterior i aniran pujant.
8. A mida que els paquets van pujant, totes les dades del **protocol IP** i els ports del **protocol TCP** es separen del paquet per tal de deixar el missatge del protocol d'aplicació.
9. Un cop les dades arriben a dalt de tot, els paquets han reconstruït el missatge original.

DNS (Domain Name System)

Què és?

El **DNS** és una tecnologia basada en una base de dades que serveix per trobar noms en les xarxes, és a dir, per conèixer la **direcció IP** de la màquina on està allotjat el **domini**.

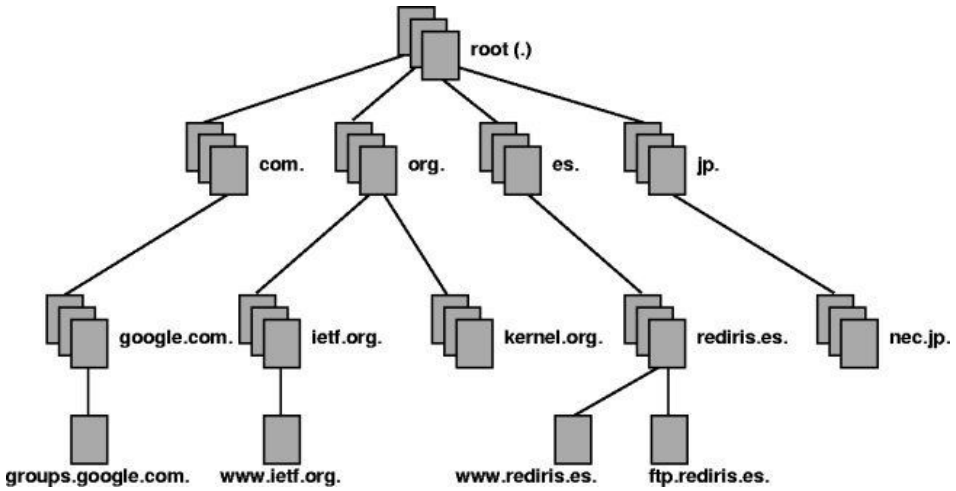
Un **domini** és un nom únic que identifica a un lloc web a Internet. Un exemple d'aquests, seria **doublescloud.com,** l'utilitzat en aquesta ocasió per l'aplicació web.

Com ja hem vist en apartats anteriors, quan un ordinador està connectat a una xarxa (ja sigui Internet o una xarxa domèstica) té assignada una direcció **IP**. Si estem en una xarxa amb pocs ordinadors, és fàcil tenir memoritzades les direccions **IP** de cadascun dels ordinadors i així accedir a ells, però què passa si hi ha milions de dispositius i cadascú té una **IP** diferent? Doncs que es faria impossible, per això existeixen els dominis i els **DNS** per traduir-los.

Per tant, el **DNS** és un sistema que serveix per traduir els noms de la xarxa, i està format per tres parts amb funcions ben diferenciades.

- **Client DNS:** està instal·lat en el client (nosaltres) i realitza peticions de resolució de noms als servidors DNS.
- **Servidor DNS:** són els encarregats de contestar les peticions i trobar els noms mitjançant un sistema estructurat en base d'arbre.
- **Zones d'autoritat:** són servidors o grups d'ells que tenen assignats trobar un conjunt de dominis determinats (com els *.es* o els *.org*).

Com funciona?



Imatge 4: Estructura d'arbre dels servidors DNS.

La resolució de noms utilitza una estructura d'arbre, mitjançant la qual els diferents servidors **DNS** de les zones d'autoritat s'encarreguen de trobar les direccions de la seva zona, i si no se sol·licita a un altre servidor que creguin que coneix la direcció. A continuació explicaré com es realitza una petició **DNS** senzilla.

- Introduïm en el nostre navegador el domini de l'aplicació: www.doublescloud.com
- El nostre **Sistema Operatiu** comprova la petició i veu que no està a la seva "memòria caché" la direcció del domini (perquè suposem que no l'hem visitat recentment), després realitza la petició al servidor **DNS**.
- El servidor **DNS** que tenim configurat tampoc té memoritzada la **direcció IP** d'aquest domini, per tant realitza una petició al servidor encarregat de la zona d'autoritat **.com**.
- El servidor encarregat de la zona d'autoritat **.com** té una taula de dades en la qual estan guardades les **direccions IPs** de les màquines i els seus **dominis**. Ho busca i el servidor **DNS** li respon que està emmgatzemada a la màquina amb la direcció 46.105.171.70.

- Finalment el servidor li torna la consulta a l'usuari (**navegador** en aquesta ocasió) i comença l'intercanvi de paquets.

Pot semblar un procés llarg i complex, però en realitat no ho és, ja que es tracta d'un procés repetitiu en què els servidors estan preparats per respondre en qüestió de microsegons. A més, en la majoria d'ocasions, no es produeix tota aquesta petició, ja que els servidors tenen zones de memòria "memòria caché" en les quals s'emmagatzemen les peticions més habituals per tal de respondre més ràpid i no saturar la xarxa amb multitud de peticions.

L'usuari

En aquest apartat parlarem de dos conceptes molt importants en el procés de comunicació web per part de l'usuari: el **navegador** i les **galetes**.

El navegador

El navegador web és el programa que **permet l'accés** al lloc web, interpretant la informació de diferents tipus d'arxius i llocs webs per tal que puguin ser visualitzats.

La comunicació entre el servidor web i el navegador es realitza mitjançant el protocol d'aplicació **HTTP o HTTPS** (versió segura de HTTP).

La funció principal dels navegadors és descarregar documents *HTML*, imatges o altres i mostrar-los en pantalla.

Galetes

Una **galeta** és una petita informació enviada per un lloc web i emmagatzemada en el **navegador** de l'usuari, de manera que el lloc web pugui consultar l'activitat prèvia de l'usuari. En una **galeta** podríem trobar informació com:

- Si l'usuari ha marcat el "check" per recordar l'usuari i la contrasenya.
- L'usuari i la contrasenya introduïdes per l'usuari, i per quan torni a accedir al lloc web, s'autentifiqui automàticament sense sol·licitar l'usuari i la contrasenya.
- L'idioma seleccionat per l'usuari l'última vegada que va accedir, per mostrar-li directament la pàgina amb aquest idioma i no solicitar-lo cada vegada que entri.
- L'últim producte que va comprar en un lloc web (si es tracta d'una botiga virtual) per mostrar-li ofertes de productes relacionats.

Aquesta **galeta** es guarda a l'ordinador de l'usuari i s'envia al servidor cada vegada que es fa una petició, d'aquesta manera el servidor pot accedir a les dades.

El servidor

En aquesta secció trobem els diferents elements que fan possible el funcionament de l'aplicació web per part del servidor.

Firewall

Un **firewall** és un dispositiu que permet gestionar i filtrar el trànsit entrant i sortint que hi ha entre dues xarxes.

Si el trànsit entrant o sortint compleix amb una sèrie de regles que es poden especificar, el trànsit circularà amb total normalitat. En cas de no complir-les, el trànsit serà bloquejat.

De tal manera, un **firewall** ens pot servir per evitar intrusions no desitjades a la nostra xarxa i també bloquejar cert trànsit provinent de la nostra xarxa.

Aquest element és prou important per tenir-lo configurat correctament, per aquest motiu, ha sigut un element que he tingut en compte a l'hora de dur a terme la configuració del servidor. Aquí és on entra en joc el **protocol TCP** i els seus **ports**, explicats en apartats anteriors.

Protocol	**Port**
HTTP	80
HTTPS	443

Aquests són alguns del ports que s'han hagut de configurar al servidor pel seu correcte funcionament. Tenir aquests ports oberts, ens permetrà que l'usuari sigui capaç d'intercanviar paquets a través d'aquests protocols, que en aquest cas es tracta dels protocols d'aplicació **HTTP** i **HTTPS**. Una altra funció que se li pot donar als **firewalls**, és per parar atacs **DoS** que veurem en els següents apartats.

Servidor web

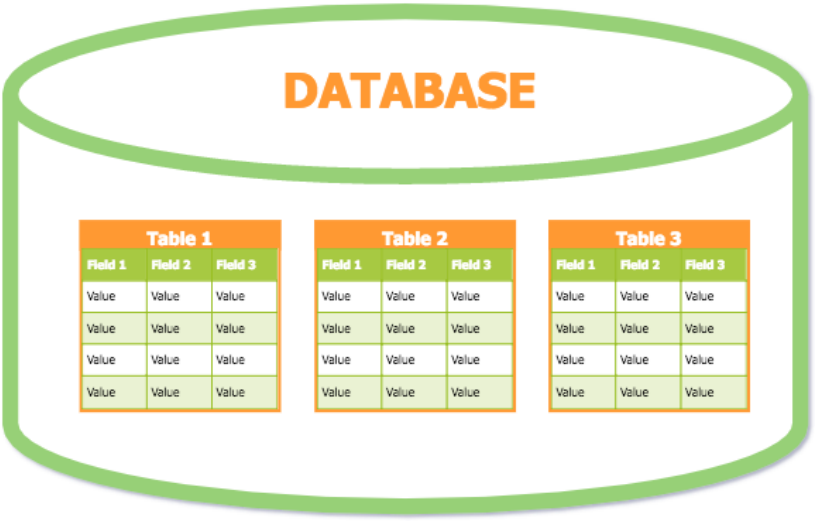
Un servidor web és el programa informàtic que processa una aplicació del costat de servidor. El codi rebut pel client és renderitzat pel navegador web. Per tal de transmetre aquesta informació s'utilitza el protocol **HTTP**, que pertany a la capa d'aplicació del **model TCP/IP**.

Com es pot veure, ja sabem que l'usuari utilitzarà el **navegador** per tal d'intercanviar **paquets HTTP** amb el **servidor**, aquest, al mateix temps, tindrà un programa que interpretarà el protocol **HTTP**, en aquest cas el **servidor HTTP**, per tal de poder intercanviar paquets.

En aquest cas, he fet servir el servidor **HTTP Apache** que és un servidor web HTTP de codi obert que permet configurar un munt d'opcions.

Base de dades

Una **base de dades** és un conjunt organitzat de dades segons una estructura coherent, i accessibles des d'un o més programes o aplicacions, de manera que qualsevol d'aquestes dades pot ésser extreta del conjunt i actualitzada, sense afectar ni l'estructura del conjunt ni les altres dades.



Imatge 5: Esquema d'una base de dades

A dalt podem veure una imatge que explica molt bé com s'estructura una base de dades, a continuació l'explicarem amb un exemple:

Suposem que tenim una empresa i volem guardar la informació dels nostres clients, dels productes que venem i un registre de les compres que s'han efectuat. En aquest cas, una base de dades és l'eina ideal. Primer de tot podríem crear una **base de dades** amb el **nom** de la **nostra empresa** i a continuació crearíem tres **taules** que es podrien dir **Clients**, **Productes** i **Compres**

Aquestes taules tindrien unes **columnes** que serien les diferents propietats i unes **files** que serien els diferents registres. Anem a fer la taula de **Clients**:

Id	**Nom**	**Cognoms**
1	Albert	Comes Comes
Correu	**País**	**Adreça**
albertcomes@gmail.com	Espanya	Carrer nou, num. 7, Girona

En el cas de la taula de **Clients,** tenim 6 columnes que ens indiquen diferents propietats dels nostres clients com el nom, els cognoms, el correu electrònic, etc. A més tenim 1 **registre** que és el nostre client amb la seva informació.

Com es pot observar, aquesta taula té una **propietat** que es diu **Id,** aquesta propietat serà utilitzada en altres taules per referir-se a aquest determinat usuari. Per ensenyar això, combinarem la taula de **Productes** i **Compres** amb la de **Clients**.

Suposem que la taula **Productes** té la següent estructura:

Id	**Nom**	**Preu**	**Quantitat**
1	Bolígraf BIC	1.20 €	22
2	Llapis	0.80 €	20
3	Goma	1.30 €	18

I la taula de **Compres** tindrà la següent estructura:

Id	**Id_Client**	**Id_Producte**	**Data**
1	1	2	20/09/2017
2	1	3	20/09/2017

D'aquesta manera, si un **Client** realitza una compra, aquesta serà introduïda dins de la taula de **Compres**. En aquest cas observem que aquesta taula té la **Id** de la compra per identificar-la, la **Id** del **Client** per saber de quin client es tracta i la **Id** del **Producte** per tal de saber quin producte va comprar. A més a més, té una columna per la **Data** per saber en quin moment es va efectuar la compra.
Ara que ja sabem com s'estructura una base de dades, anem a parlar de com podem extreure o introduir informació. Per fer-ho, es fan servir les anomenades **consultes**, que no són res més que una línia de codi indicant el que es vol.

Anem a posar uns quants exemples:

Si volguéssim obtenir la informació del Client amb *albertcomes@gmail.com* com a correu electrònic, hauríem de realitzar la següent consulta:

SELECT * FROM Clients **WHERE** Correu **=** "albertcomes@gmail.com"

Un cop realitzada aquesta consulta, obtindríem tota la informació de l'usuari.

Si volguéssim actualitzar el correu electrònic del client amb el número **1** com a **Id** a *albertcc@gmail.com,* s'hauria de realitzar la següent consulta:

SELECT * FROM Clients **WHERE** Id = "1";

Un cop realitzada aquesta consulta, l'usuari amb **Id = 1** tindria *albertcc@gmail.com* com a correu electrònic.

Com podem veure, una base de dades no és res més que un conjunt de dades estructurat que ens permet guardar diferent informació per tal de ser utilitzada en una aplicació. En

l'aplicació s'han hagut de crear diferents taules que corresponen a la d'usuaris, arxius, registres d'events, etc.

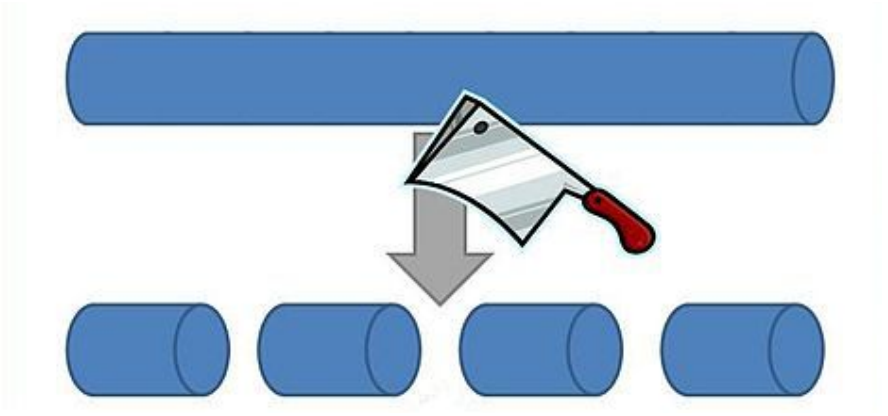
File chunking

Està clar que una part important d'un servei d'emmagatzematge de fitxers al núvol és el procés de pujar arxius. Trobem casos on els fitxers a pujar són de mides d'*1 MB* a *20 MB*, però què passa si són superiors a *100 MB*?

Pujar arxius d'aquestes dimensions de la forma tradicional suggereix un problema:

- En primer lloc ens trobem amb casos on l'arxiu és molt gran o bé la connexió de pujada de l'usuari és molt lenta. El procés pot durar una gran quantitat de temps, fet que produiria que durant tot aquest període, part dels recursos del servidor es destinessin a la pujada d'arxius, deixant amb pocs a les altres funcions.

Per tal de solucionar aquest problema s'utilitza l'anomenat **file chunking**, que consisteix en **dividir** l'arxiu en **diferents parts** d'*1 o 2 MB* (chunks), **pujar-los** al servidor i anar **ajuntant** les **diferents parts** fins a tenir l'**arxiu complet** que l'usuari volia pujar.



Imatge 6: Representació divisió de l'arxiu en "chunks".

Anem a posar un exemple, imaginem que ens trobem al lloc web i volem pujar un vídeo que pesa *200 MB*. Tal i com està programada la web, el procés que seguirà la pujada de l'arxiu serà el següent:

- Primer de tot es realitzaran unes comprovacions per veure si l'arxiu és vàlid tant pel que fa als formats com a les mides.
- Un cop realitzades les comprovacions, s'iniciarià un bucle on s'aniria **dividint l'arxiu** en **chunks** d'*1 MB*, s'aniran **pujant** els **chunks** i s'aniran **ajuntant** fins a obtenir l'arxiu que es volia pujar. Un cop l'arxiu estigués complet, es crearia un nou registre a la taula de Fitxers del servidor on es guardaria tota la informació de l'arxiu.

Sessions

Una **sessió** no és res més que una informació que es guarda al servidor i és accessible a través d'una **Id** que es guarda en una **galeta** en el **navegador** de l'usuari. Aquesta **Id** s'utilitza per verificar si la sessió existeix, si efectivament existeix, es dóna accés a la informació.

Mentre duri la sessió, la informació estarà disponible pel servidor cada vegada que l'usuari sol·liciti la pàgina. Alguns exemples d'informació que pot resultar interessant guardar serien:

- Si l'usuari s'ha autentificat o no.
- Identificador de l'usuari que s'ha autenticat.
- Productes afegits a un carro de la compra.

Aquesta informació es guarda en el servidor, en un fitxer temporal o en una base de dades, com és en el nostre cas, pel que només és accessible des del codi de l'**aplicació**, però no des del **navegador**. Això permet emmagatzemar informació privada sense corre

el risc de que algun pirata informàtic la vegi o modifiqui, cosa que amb les **galetes** podem fer ja que es pot editar posant el contingut que es vulgui.

Vulnerabilitats web

En aquest apartat tractaré diferents vulnerabilitats que pot haver-hi en una pàgina web, que qualsevol usuari malintencionat podria fer ús d'elles per tal de provocar danys.

HTTP i HTTPS

Què són?

Com ja sabem, **HTTP** és el protocol que utilitzen els navegadors per comunicar-se amb els servidors web. Funciona bé, però té un problema: la seguretat.

Amb **HTTP**, qualsevol dada es transmet en text pla, sense xifrar. És a dir, qualsevol que es connecti a la nostra xarxa WiFi, o que tingui accés a la comunicació entre el nostre **ordinador** i el **servidor** pot veure totes les dades que rebem i enviem.

Per tal de transmetre aquestes dades de forma segura, hem de fer servir **HTTPS** (Hyper Text Transfer Protocol Secure), que de fet no és res més que **HTTP** normal sobre **SSL/TLS**.

SSL/TLS (Secure Sockets Layer/Transmission Layer Security) són dos protocols per enviar paquets xifrats a través d'Internet.

Què protegeix?

Un altre dels problemes que trobem en la comunicació per Internet és el de com sabem que ens estem comunicant amb el servidor **example.com** i no es un servidor **fals,** que s'està fent passar per **example.com**?

Per tant, s'han d'autentificar els servidors. No serveix de res tenir les dades xifrades si no ens assegurem que estem connectats al servidor correcte. Per això existeixen els **certificats SSL**, que ens asseguren que ens estem connectant al servidor desitjat. Aquests certificats són emesos per les **Certificate Authorities**, les entitats emisores de certificats **SSL** firmats, que només donen certificats sobre un domini al seu propietari.

HTTPS xifra totes les dades de **HTTP**. Això és, no només la pàgina web, sinó també la URL completa, els paràmetres enviats, les galetes, etc. L'únic que queda al descobert són les dades que necessita el paquet com el servidor i el port.

Per tant, **HTTPS** no només ens impedeix que algú vegi les pàgines web que estem visitant, també impedeix conèixer les URLs que visitem, els paràmetres que enviem (per exemple, els usuaris i les contrasenyes) o les galetes que enviem i rebem.

A l'**Annex 4** podem veure com aprofitar-nos d'aquesta vulnerabilitat.

Injecció SQL

Què és?

Com ja hem explicat en apartats anteriors, la majoria d'aplicacions webs es basen en una base dades on s'hi pot guardar i extreure informació dels usuaris, els arxius que guarden, etc. Quan s'elabora una consulta a partir del llenguatge de programació **SQL**, pot aparèxer el que se'n diu **Injecció SQL**, que no és res més que la inserció de codi no desitjat per part de l'usuari.

Un exemple

Per fer-ho més fàcil posarem un exemple: Tenim una base de dades per la nostra aplicació i dins d'aquesta base de dades hi trobem una taula anomenada **Usuaris** on hi guardem informació de 5 usuaris.

Id	Nom	Adreça	Ciutat	País
1	Maria Anders	Obere Str. 57	Berlín	Alemanya
2	Ana Trujillo	Avda. de la Constitución 2222	Ciutat de Mèxic	Mèxic
3	Antonio Moreno	Mataderos 2312	Ciutat de Mèxic	Mèxic
4	Thomas Hardy	120 Hanover Sq.	Londres	Regne Unit
5	Christina Berglund	Berguvsvägen 8	Luleå	Suècia

Al lloc web, tenim un formulari on pots introduir la **Id** de l'usuari per tal d'aconseguir la seva informació. Anem a provar:

L'usuari introdueix com a paràmetre **Id = 1**

SELECT * FROM Tests **WHERE** Id **=** 1**;**

Id	Nom	Adreça	Ciutat	País
1	Maria Anders	Obere Str. 57	Berlín	Alemanya

Si realitzem la consulta anterior, el que estarem fent serà seleccionar l'usuari que té el número **1** com a **Id**. Fins aquí tot funciona bé, l'aplicació compleix amb la funció que ha de realitzar. Però, anem a provar la següent consulta:

L'usuari introdueix com a paràmetre **Id = 1 OR 1=1**

SELECT * **FROM** Tests **WHERE** Id **=** 1 **OR** 1**=**1**;**

Id	**Nom**	**Adreça**	**Ciutat**	**País**
1	Maria Anders	Obere Str. 57	Berlín	Alemanya
2	Ana Trujillo	Avda. de la Constitución 2222	Ciutat de Mèxic	Mèxic
3	Antonio Moreno	Mataderos 2312	Ciutat de Mèxic	Mèxic
4	Thomas Hardy	120 Hanover Sq.	Londres	Regne Unit
5	Christina Berglund	Berguvsvägen 8	Luleå	Suecia

Com podeu observar, un cop realitzada la consulta anterior, obtenim tots els usuaris de la taula. Com és això? Doncs, és més simple del que podem pensar. L'únic que hem realitzat ha estat una consulta en què li demanem que el paràmetre **Id** sigui **1** o bé que la expressió **1=1** sigui verdadera, i com podem imaginar la expressió **1=1** sempre és vertadera, per tant, obtindrem tots els resultats. En aquest cas, aquest problema no seria una amenaça, ja que l'únic que estem fent és obtenir tots els usuaris, però què passaria si aquesta taula contingués correus electrònics i contrasenyes? La cosa seria diferent. Però, anem més lluny, anem a borrar la taula.

L'usuari introdueix com a paràmetre **Id = 1; DROP TABLE Tests**;

SELECT * FROM Tests **WHERE** Id **=** 1**; DROP TABLE** Tests**;**

El que hem fet ha estat seleccionar un usuari per tal de completar la consulta *SQL* i l'hem concatenat amb una altra consulta *SQL*, que en aquest cas el que farà serà borrar la taula amb la qual hem estat treballant. Ara quan tornem a realitzar la consulta del principi, obtindrem aquest error:

L'usuari introdueix com a paràmetre **Id = 1**

SELECT * FROM Tests **WHERE** Id **=** 1**;**

En realitzar aquesta consulta, obtindrem un missatge d'error, ja que haurem borrat la taula.

Com ens protegim?

Per tal de protegir-nos d'atacs **SQL Injection**, s'ha de fer ús de filtres que eliminen els caràcters problemàtics com les cometes (‘’) o el punt i coma (;).

Atacs DoS/DDoS

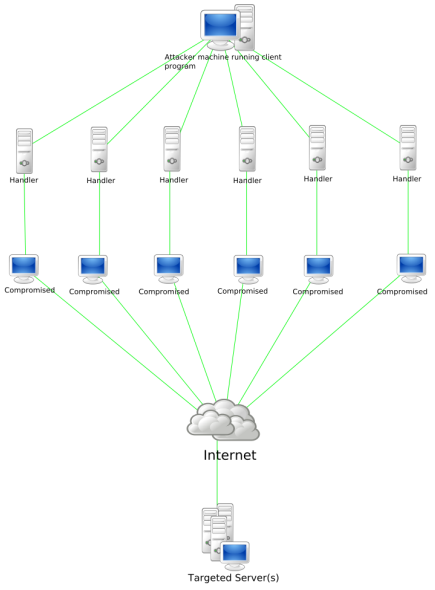
Què són?

DDoS són les sigles de **Distributed Denial of Service**. La traducció és "atac distribuït de denegació de servei", i traduït de nou significa que s'ataca al servidor des de molts ordinadors per tal que deixi de funcionar.

Tot i així, això no ens guia molt sobre que és un **DDoS**. Per explicar-ho recorreré a una simple analogia en la qual el nostre servidor és un auxiliar que atén a persones en una finestreta.

El nostre auxiliar és molt eficient i és capaç d'atendre a vàries persones a la vegada sense cap problema. Però un dia comencen a arribar centenars de persones a la finestreta a demanar-li coses. I com qualsevol persona normal, quan hi ha molta gent demanant-li coses no és capaç d'atendre a tothom i comença a atendre més lent. Si encara ve més gent probablement s'acabarà cansant, marxarà de la finestreta i ja no atendrà ningú més.

En el servidor passa el mateix: quan hi ha moltes peticions, el servidor es queda sense recursos, es penja i deixa de funcionar. Pot ser que s'apagui directament o que només deixi de respondre les connexions. En qualsevol cas, el servidor no tornarà a la normalitat fins que l'atacant pari, ja sigui perquè els atacants han parat o perquè s'ha aconseguit bloquejar les connexions il·legítimes.

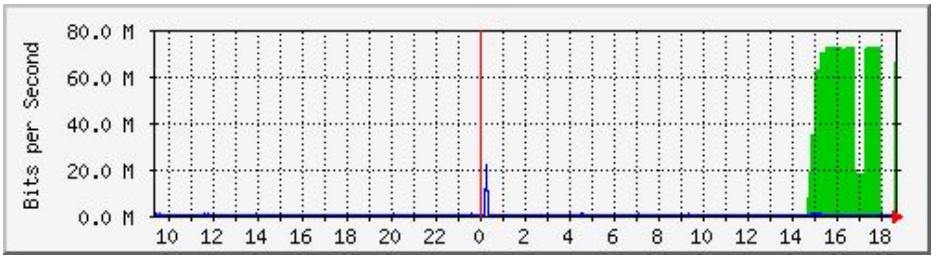


Imatge 7: Representació d'un atac DDoS.

Com afecta un atac DDoS a un lloc web?

Depèn de l'atac i del servidor. Els servidors es poden protegir contra aquests atacs amb filtres que rebutgen els paquets mal formats o modificats amb **IPs** falses, de forma que al servidor només li arriben els paquets legítims. Per descomptat, les mesures no són infal·libles i el servidor sempre pot acabar saturat si l'atac és prou massiu i està ben preparat.

Per fer-nos una idea del volum necessari per a què un **DDoS** sigui efectiu, a baix tenim un gràfic que representa el trànsit d'un servidor al llarg del temps. El trànsit durant l'atac (el verd) és tan gran que quasi bé no es pot apreciar el trànsit normal del servidor.



Imatge 8: Trànsit d'un lloc web durant un atac DDoS

I què passa quan el servidor es satura? Simplement deixa d'estar disponible durant un temps fins que l'atac para. És molt difícil que es produeixin danys físics en el servidor. A més, el atacs **DDoS** no permeten entrar en el servidor.

Així doncs, un **DDoS** només pot provocar la caiguda de la pàgina web, res més. Depenent del tipus de web això pot ser una catàstrofe o no. Si la web genera diners (venta online, publicitat), el propietari deixa de guanyar diners mentre aquesta web està caiguda. Imagineu les pèrdues que pot arribar a tenir *Amazon*, per exemple, si la seva pàgina està caiguda durant un dia.

Programació de l'aplicació

Com hem vist anteriorment, l'objectiu principal d'aquest treball ha sigut la creació d'un lloc web d'emmagatzematge de fitxers. En aquest apartat presentaré d'una forma resumida com funciona el lloc.

El lloc s'ha dissenyat per permetre emmagatzemar als usuaris qualsevol arxiu en una carpeta assignada. Aquest arxiu es sincronitza i es pot visualitzar des de qualsevol ordinador. Els arxius poden ser compartits amb altres usuaris del servei o bé obtenir un enllaç per la descàrrega directa.

Llenguatges de programació

Un llenguatge de programació és un llenguatge formal dissenyat per realitzar processos que poder ser duts a terme per màquines com ordinadors.

Dins de la programació web trobem dos tipus de grups de llenguatges de programació, els anomenats **front-end** (client side) i els **back-end** (server side).

El **front-end** és la part del software que interactua amb l'usuari i el **back-end** és la part que processa l'entrada del front-end. La idea general és que el **front-end** sigui el responsable de recol·lectar les dades de l'usuari, ja sigui informació personal, arxius, i transformar-los, ajustant-los a les especificacions que demana el **back-end** per poder processar-los, tornant generalment una resposta que el **front-end** rep i exposa a l'usuari d'una forma entenedora.

Pel que fa a la part de **front-end**, hem fet servir "*HTML"*, *JavaScript* i "*CSS"* i per la part de **back-end** *PHP* i "*SQL"*.

HTML

És el llenguatge essencial utilitzat en el desenvolupament de pàgines web. És considerat un llenguatge d'etiquetes, que és emprat per marcar com va ordenat el contingut de la pàgina.

Posem un exemple: volem tenir una web i la volem ordenar a partir d'un **títol**, una **introducció**, un **desenvolupament** i una **conclusió**.

```
<html>
  <head></head>
  <body>
    <h1>Títol</h1>
      <h2>Introducció</h2>
      <h2>Desenvolupament</h2>
      <h2>Conclusió</h2>
  </body>
</html>
```

Imatge 9: Codi *HTML*.

En primer lloc trobem les etiquetes <*html*> que ens indiquen que tot el que hi hagi a dins d'aquestes, haurà de ser interpretat com a codi *HTML*. A continuació tenim l'etiqueta <*head*> que es fa servir per introduir informació que l'usuari no veurà, però els cercadors webs com Google faran servir per trobar la pàgina. Tot seguit, apareix l'etiqueta <*body*> en la qual s'introduirà tot el contingut que serà visible per l'usuari. A dins, hi trobem una etiqueta <*h1*> que indica que serà el títol principal, i 3 subtítols que s'indiquen a partir de l'etiqueta <*h2*>. És interessant veure com cada etiqueta que s'obre, com <*h1*>, es tanca a

partir de la mateixa etiqueta, però afegint-hi una "/" al principi. És molt important tancar bé les etiquetes, ja que serveix per delimitar la seva àrea d'actuació.

Si pugéssim aquest codi a un servidor web, la pàgina quedaria de la següent manera:

Títol <h1>Títol</h1>

Introducció <h2>Introducció</h2>

Desenvolupament <h2>Desenvolupament</h2>

Conclusió <h2>Conclusió</h2>

Imatge 10: Vista del codi *HTML* des del navegador.

CSS

És el llenguatge de disseny gràfic utilitzat per definir i crear la presentació d'un document escrit en *HTML*.

Continuant amb l'exemple anterior, si volguéssim augmentar la **mida** del títol a "36px" i **canviar la lletra a color** "vermell", ho hauríem de fer de la següent manera:

```
<style>
  h1 {
     font-size: 36px;
     color: red;
   }
</style>
<body>
  <h1>Títol</h1>
   <!-- Contingut del <body> -->
</body>
```

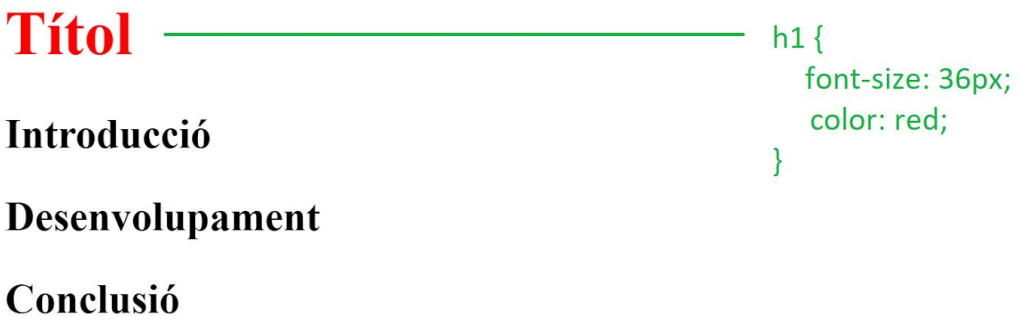
Imatge 11: Codi CSS.

En la imatge anterior, podem veure com a continuació de les etiquetes <*head*>, s'hi col·loquen unes altres anomenades <*style*> que serveixen per marcar que tot el que hi hagi a dins d'aquestes etiquetes es tracta de codi *CSS* i per tant, s'haurà d'interpretar com a tal.

Aquest llenguatge funciona de la següent manera:

- S'indica quina etiqueta es vol modificar, en aquest cas l'etiqueta <*h1*> que correspon a la del títol principal.
- A continuació, s'obren unes claus i a dins va tot el codi, en aquest cas l'atribut "font-size" que significa "mida de lletra", ha de ser de "36px i l'atribut "color" que ha de ser de color vermell.

D'aquesta manera, obtindrem un títol amb una mida de lletra de 36px i de color vermell.



Imatge 12: Vista del codi *HTML* i *CSS* des del navegador web.

JavaScript

JavaScipt és un llenguatge que permet crear diferents programes que poden ser executats en el navegador de l'usuari, per aquesta raó està considerat un llenguatge de **front-end**. Permet fer una gran quantitat de coses, tals com reconèixer **events**, animacions, connexions amb base de dades, etc.

A continuació, ensenyarem un exemple que és utilitzat àmpliament al llarg de l'aplicació, el de reconèixer **events**, que no són res més que accions generades per l'usuari, com la de prémer un botó:

```
<html>
  <head></head>
  <body>
    <button id="Send_Button">Enviar</button>
  </body>
  <script>
    var Button = document.getElementById("Send_Button");
    Button.onclick = function() {
      alert("Has premut el botó d'enviar.");
    }
  </script>
</html>
```

Imatge 13: Codi *JavaScript*.

En aquest cas, podem veure com dins de l'etiqueta <*body*>, hi trobem un botó que es crea a partir de l'etiqueta <*button*> amb un atribut que es diu **id** en verd i que és igual a

“Send_Button”, aquesta **id** serà utilitzada per referenciar-nos al botó. A continuació de l’etiqueta del botó, hi trobem una nova etiqueta anomenada <*script*>. Aquesta serveix per inserir codi *JavaScript*. Dins d’aquesta etiqueta, trobem que es declara una variable, que no és res més que un espai reservat a la memòria on s’hi poden guardar coses, i en aquest cas guardem el botó que recuperem a partir de la **id** “Send_Button” que hem establert anteriorment. Un cop ja tenim la variable, indiquem a quin **event** volem respondre. En aquest cas veiem l’**event** “onclick” que es refereix a què passarà quan premem el botó. En aquesta ocasió, quan el botó sigui premut, ens avisarà que l’hem premut, com podem veure en la imatge següent:

Imatge 14: Vista de l’execució de l’**event**: *onclick*

És important ressaltar que en aquest projecte trobem un munt més d’**events** com: *onmouseover* (quan el cursor estigui sobre d’una determinada zona), *onmouseleave* (quan el cursor marxi d’aquella zona), *onsubmit* (quan s’envia un formulari), etc.

PHP

Aquest és un dels llenguatges de programació back-end que s'ha utilitzat. A partir d'aquest llenguatge, es realitzen totes les tasques per part del servidor:

- Generar contingut dinàmic.
- Crear, obrir, llegir, escriure, eliminar i tancar arxius.
- Processar dades de formularis.
- Enviar i rebre galetes.
- Crear sessions.
- Afegir, eliminar i modificar informació d'una base de dades a partir del llenguatge *SQL*.

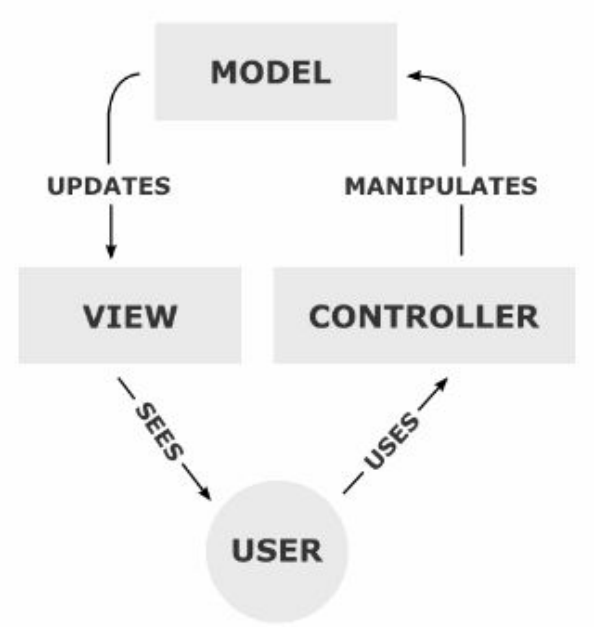
A l'**Annex 1** podem trobar una petita explicació dels conceptes més importants a l'hora de programar amb *PHP*.

SQL

Aquest llenguatge és l'utilitzat per gestionar les base de dades, com ja hem vist anteriorment en l'apartat de **base de dades**.

Patró d’arquitectura

Els patrons d’arquitectura serveixen per organitzar l’aplicació, i per tant, facilitar el desenvolupament d’aquesta.



Imatge 15: Representació del patró d’arquitectura MVC.

L’esquema està diferenciat en quatre parts: l’interfície d’usuari, el controlador, el model i la vista.

Interfície d’usuari: És la part visible per l’usuari i és amb la qual interactua.

El controlador: És l’encarregat de respondre els events (usualment accions de l’usuari) i invoca peticions al “model” quan es realitza alguna sol·licitud sobre la informació (per exemple, editar un document o un registre de la base de dades).

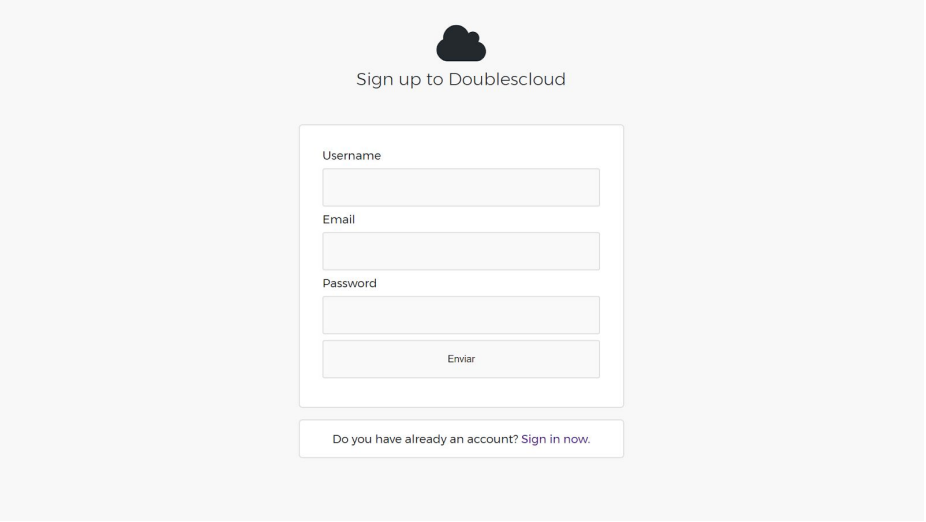
El model: És l'encarregat de realitzar certes accions, com modificar una base de dades, modificar un document, etc.

La vista: És l'encarregada de mostrar la informació i els fitxers de l'usuari a partir del model.

Per entendre millor aquesta metodologia, anem a posar com exemple que ens volem **registrar** en el lloc web.

Interfície d'usuari

Com ja hem comentat, aquesta és la part visible per l'usuari i amb la qual pot interactuar. Seguint l'exemple que hem proposat en l'apartat anterior, en el lloc web, aquesta part es veu de la següent manera:



Imatge 16: Interfície d'usuari de l'**event** *registrar usuari*.

Com es pot observar, aquesta pàgina no és res més que un formulari on l'usuari ha d'introduir les seves credencials i un cop enviï el formulari, es cridarà el controlador corresponent, en aquest cas el de registrar-se, que serà l'encarregat de processar les dades.

El controlador

Com hem explicat abans, el controlador és l'encarregat de respondre als **events**. En aquest cas, l'**event** de **registrar-se**, que serà cridat un cop s'enviï el formulari.

```php
//INCLUDES

  include $_SERVER["DOCUMENT_ROOT"] . "/Application/Configurations/Includes/Header.php";

//FUNCTION - Controllers / User - SignUp

  try {

    $Connection = new PDO("mysql:host=" . DATABASE_host . ";dbname=" . DATABASE_dbname, DATABASE_user, DATABASE_password);

    $User = new \User\Handler($Connection);

      $User->setUsername(filter_input(INPUT_POST, "inputUsername_SignUp", FILTER_SANITIZE_SPECIAL_CHARS));
      $User->setEmail(filter_input(INPUT_POST, "inputEmail_SignUp", FILTER_SANITIZE_SPECIAL_CHARS));
      $User->setPassword(filter_input(INPUT_POST, "inputPassword_SignUp", FILTER_SANITIZE_SPECIAL_CHARS));
      $User->setImage("");
      $User->setSize(0);
      $User->setVOE(false);

        $User->checkUsername();
        $User->checkEmail();
        $User->hashPassword();
        $User->setAccount();
        $User->getIdByEmailOrUsername();
        $User->checkDirectory();
        $User->setDirectory();

    $Session = new \Session\Main($Connection);

      $_SESSION["userId"] = $User->getId();
      $_SESSION["Username"] = $User->getUsername();

    $Log = new \Log\Handler($Connection);

      $Log->setId($User->getId());
      $Log->setDatetime(date("Y-m-d H:i:s"));
      $Log->setEvent("user.signup");
      $Log->setData("Originated from " . $_SERVER["REMOTE_ADDR"]);

        $Log->Set();

    $Connection = null;

    header("Location: " . "/public_html/me/");

  } catch (Exception $e) {

    header("Location: " . "/public_html/signup/");

  }
```

Imatge 17: Controlador de l'**event** *registrar usuari*.

Aquest és el codi corresponent al controlador encarregat de registrar l'usuari. Com es pot observar, hem diferenciat el codi en quatre parts, que no són res més que la crida de les diferents funcions del model.

En primer lloc, s'estableixen les propietats de l'usuari amb els valors que ha introduït l'usuari, que en aquest cas serien el **nom d'usuari**, el **correu electrònic** i la **contrasenya**. Cal tenir en compte que abans d'establir-les, les propietats de l'usuari passen per un filtre per tal d'evitar la vulnerabilitat **Injecció SQL**. A continuació, es criden diferents **funcions** que comproven que el nom d'usuari i correu electrònic que s´han introduït estiguin disponibles. A continuació, si estan disponibles, s'introdueixen els valors a la base de dades i es crea la carpeta on l'usuari guardarà els seus arxius.

En segon lloc, es crea una **sessió** on es guardarà la **Id** de l'usuari, perquè l'aplicació el pugui reconèixer al llarg de les accions. D'aquesta manera, quan entri a la pàgina personal, obtindrà els arxius associats al seu compte.

Finalment es crea una nova entrada a la base de dades que correspon a un registre on es guarden tots els events, per tant, si l'usuari es registre amb èxit, aquesta acció es guardarà.

El model

Com ja hem dit, aquest és l'encarregat de fer les modificacions als documents i a les bases de dades.

```
//checkUsername
  public function checkUsername() {
    $Query = "SELECT 1 FROM Users WHERE Username = :newUsername";
      $sqlQuery = $this->Connection->prepare($Query);
        $sqlQuery->bindValue(":newUsername", $this->Username);
          $sqlQuery->execute();
            if (!$sqlQuery->rowCount()) { return true; } else { throw new \Exception(); }
  }
//checkEmail
  public function checkEmail() {
    $Query = "SELECT 1 FROM Users WHERE Email = :newEmail";
      $sqlQuery = $this->Connection->prepare($Query);
        $sqlQuery->bindValue(":newEmail", $this->Email);
          $sqlQuery->execute();
            if (!$sqlQuery->rowCount()) { return true; } else { throw new \Exception(); }
  }
```

Imatge 18: Codi de dos funcions del model utilitzat en l'**event** de registrar usuari.

Com podem veure amb la captura de pantalla anterior, trobem definides dues de les funcions emprades en l'exemple anterior del controlador. Les dues funcions en aquest cas **checkUsername** i **checkEmail,** no són res més que dos consultes que serveixen per comprovar si el nom d'usuari i correu electrònic elegit per l'usuari estan en ús o no. Si ho estan, les funcions retornaran un valor "true" que significa que els valors estan lliures i es podrà seguir amb l'execució del codi del controlador.

La vista

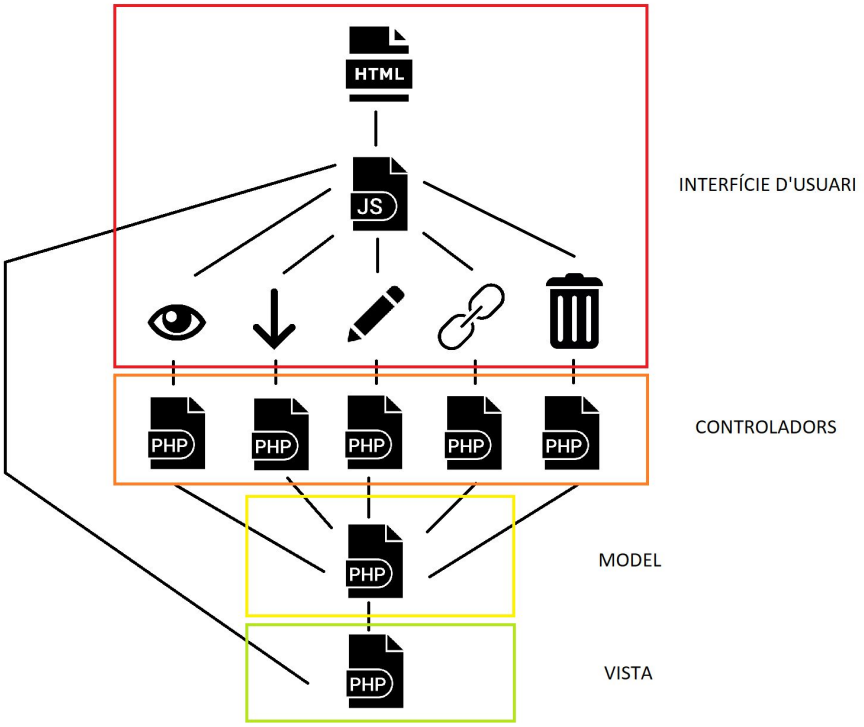
Sabem que la vista és l'encarregada de mostrar la informació a partir de les funcions del model.

```php
//INCLUDES
  include $_SERVER["DOCUMENT_ROOT"] . "/Application/Configurations/Includes/Header.php";
  include $_SERVER["DOCUMENT_ROOT"] . "/Application/Configurations/Includes/Headers/Session.php";
//FUNCTION - Views / getFiles
  try {
    $Connection = new PDO("mysql:host=" . DATABASE_host . ";dbname=" . DATABASE_dbname, DATABASE_user, DATABASE_password);
      $User = new \User\Handler($Connection);
        $User->setId($_SESSION["userId"]);
          $Files = $User->getFiles($_POST["l"], $_POST["Status"]);
            //Procés per imprimir els arxius en format HTML.
  } catch (Exception $e) {
    die($e->getMessage());
  }
```

Imatge 19: Codi de la vista de carregar arxius.

El codi que veiem en la imatge anterior, correspon a la vista encarregada d'obtenir tots els arxius de l'usuari. Per fer-ho, primer obté la **Id** de l'usuari a partir d'una **sessió,** a continuació fa una consulta a la base de dades corresponent, en aquest cas a la d'arxius, per tal d'obtenir tots els arxius. Un cop els tingui, aquests passaran per una sèrie de filtres que els preparà per ser impresos a partir de codi *HTML*. Un cop tinguem aquest codi *HTML*, aquest serà enviat a la interfície d'usuari i l'usuari podrà començar a fer ús de l'aplicació.

Funcionament general de l'aplicació



Imatge 20: Diagrama del funcionament del lloc web.

L'exemple anterior tracta, doncs, del procés que seguiria l'aplicació quan es realitza una acció en concret, en aquest cas l'acció de registrar-se. Però en aquest apartat, veurem d'una forma més global, com funciona l'aplicació.

Veiem com el diagrama de la **Imatge 20** està format per la interfície d'usuari, els controladors, el model i la vista. Aquest és l'esquema que ja havíem vist, però en aquest diagrama, apareix un arxiu després de l'*HTML* (part que l'usuari veu), que està programat amb *JavaScript* i que és l'encarregat de detectar tot el que fa l'usuari i transmetre-ho al controlador corresponent. Per entendre-ho millor, tenim el següent exemple:

- L'usuari realitza algun **event**.
- Aquest **event** és recollit per l'arxiu programat amb *JavaScript* que fa d'intermediari entre la interfície d'usuari i els controladors.
- L'arxiu *JavaScript* detectarà de quin tipus d'**event** es tracta i cridarà el controlador corresponent.
- Com ja sabem, aquest controlador invocarà diferents funcions que recollirà del model per tal de dur a terme l'**event**.
- Un cop s'hagi realitzat l'acció, es cridarà la vista que serà l'encarregada de carregar els arxius de nou per tal de visualitzar els nous canvis.
- La vista passarà el codi *HTML* que produeix a l'intermediari *JavaScript*, i aquest s'encarregarà de mostrar els arxius a la interfície d'usuari.

En resum, aquest diagrama ens serveix per veure com funciona l'aplicació d'una forma més general. Òbviament, està incomplet, ja que li falten un munt d'**events** que podem trobar al llarg de l'aplicació, però que per la seva extensió estan situats a l'**Annex 2** on es pot trobar el codi complet.

Per tal d'entendre millor tot el que s'ha explicat, podem visitar https://doublescloud.com i fer anar l'aplicació.

Conclusió

Una vegada acabat el treball, puc dir que m'ha aportat molts coneixements nous gràcies a les recerques que he fet, que m'han permès la realització de la part pràctica i que de ben segur faré servir al llarg de la meva vida.

També, m'ha servit per veure que darrera d'una aplicació hi ha una quantitat important de feina, que l'usuari no veu. A més, m'ha anat bé per veure la importància que té el fet de planificar bé la feina, ja que si no ho fas i comences a treballar directament, et trobes un munt d'entrebancs.

Igualment haig de dir que a l'hora de la programació de l'aplicació, em vaig trobar diferents problemes que no sabia com solucionar, fet que em va portar a una aturada temporal, havent de buscar informació per diferents llocs webs per tal de trobar la solució més adequada, fent que el temps dedicat al treball s'incrementés notablement. Tot i així, aquestes aturades m'han servit per aprendre altres conceptes que he pogut aplicar en altres parts del projecte.

Pel que fa als objectius, considero que els he pogut realitzar satisfactòriament. Tot i així, la meva intenció és continuar treballant en l'aplicació per desenvolupar-la encara més, i així poder oferir una alternativa, on la privacitat i la seguretat siguin l'element principal.

Com a conclusió final, he pogut realitzar un treball del qual em sento orgullós i que m'ha permès aprendre moltes coses, així que considero que ha estat una experiència molt positiva.

A l'annex hi podem trobar el diferents elements de l'aplicació com les bases de dades i el codi de l'aplicació. Per tal de veure el resultat final, podem visitar https://doublescloud.com i fer anar l'aplicació.

Enllaços d’interès

- Lloc web - https://doublescloud.com
- Lloc web per descarregar el codi de l’aplicació - https://doublescloud.com/downloadl/?u=73fcd1fbd108e0544a5dbf8946b28b49&p=ed4988848f6fe12a22efd51061a7db38

Webgrafia

Pàgines web utilitzades:

- **Curso de PHP/MySQL** - Curs de PHP/MySQL
  https://www.youtube.com/playlist?list=PLU8oAlHdN5BkinrODGXToK9oPAlnJxmW_
- **Curso de SQL** - Curs d'*SQL*
  https://www.youtube.com/playlist?list=PLU8oAlHdN5Bmx-LChV4K3MbHrpZKefNwn
- **Curso de JavaScript** - Curs de JavaScript
  https://www.youtube.com/playlist?list=PLU8oAlHdN5BmpobVmj1IlneKlVLJ84TID
- **HTML Tutorial -** Tutorial d'*HTML*.
  https://www.w3schools.com/html/default.asp
- **CSS Tutorial** - Tutorial de CSS.
  https://www.w3schools.com/css/default.asp
- **PHP** - Pàgina oficial del llenguatge de programació PHP.
  http://php.net/
- **JavaScript** - Pàgina oficial del llenguatge de programació JavaScript.
  https://developer.mozilla.org/es/docs/Web/JavaScript
- **Stackoverflow** - Fòrum de programació on es van visitar prop de 100 entrades.
  https://stackoverflow.com
- **How does the Internet work?** - Com funciona Internet?
  https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm
- **Treehouse** - Diferents tutorials de programació.
  https://teamtreehouse.com/

Bibliografia fotogràfica

Part principal

- Imatge portada: https://www.signiant.com/wp-content/uploads/2015/01/Flight_Hero2.png
- Imatge 1: Elaboració pròpia.
- Imatge 2: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper_files/ruswp_diag9.gif
- Imatge 3: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper_files/ruswp_diag2.gif
- Imatge 4: https://i1.wp.com/informaticacoslada.com/wp-content/uploads/2013/03/dns.png?resize=1080%2C631&ssl=1
- Imatge 5: http://database.guide/wp-content/uploads/2016/06/Relational-Database-Structure-Example-1.png
- Imatge 6: Elaboració pròpia.
- Imatge 7: https://en.wikipedia.org/wiki/File:Stachledraht_DDos_Attack.svg
- Imatge 8: https://i.blogs.es/9366c1/ddos/1366_2000.png
- Imatge 9: Elaboració pròpia.
- Imatge 10: Elaboració pròpia.
- Imatge 11: Elaboració pròpia.
- Imatge 12: Elaboració pròpia.
- Imatge 13: Elaboració pròpia.
- Imatge 14: Elaboració pròpia.

- Imatge 15: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador#/media/File:MVC-Process.png
- Imatge 16: Elaboració pròpia.
- Imatge 17: Elaboració pròpia.
- Imatge 18: Elaboració pròpia.
- Imatge 19: Elaboració pròpia.
- Imatge 20: Elaboració pròpia.

Annexos

- Imatge 21: Elaboració pròpia.
- Imatge 22: Elaboració pròpia.
- Imatge 23: Elaboració pròpia.
- Imatge 24: Elaboració pròpia.
- Imatge 25: Elaboració pròpia.
- Imatge 26: Elaboració pròpia.
- Imatge 27: Elaboració pròpia.
- Imatge 28: Elaboració pròpia.
- Imatge 29: Elaboració pròpia.
- Imatge 30: Elaboració pròpia.
- Imatge 31: Elaboració pròpia.
- Imatge 32: Elaboració pròpia.

Annexos

Annex 1: Funcionament PHP

Sintaxi

Fent ús del llenguatge de programació s'ha de tenir en compte una sèrie de puntuacions que serveixen per indicar el començament i final d'algunes parts del codi. En primer lloc tenim els parèntesis **()**, que s'utilitzen juntament amb la paraula reservada **function** per tal de definir una funció o bé amb les estructures pròpies del llenguatge.

En segon lloc tenim les claus **{}**. Aquestes serveixen per indicar el principi i el final d'una funció, sense aquestes claus, el programa no seria capaç de saber on comença ni on acaba la funció i ens donaria error.

Ja per acabar, tenim els punts i comes **;**. S'utilitzen per separar les diferents línies de codi.

Imatge 21: Sintaxi en el llenguatge *PHP*.

Variables

Una variable no és res més que un espai reservat a la memòria, que pot canviar de contingut al llarg de l'execució d'un programa. Aquesta variable s'identifica amb un nom. En el llenguatge de programació *PHP*, una variable es declara amb el signe **$** seguit del **nom de la variable**.

$nom_variable = valor;

De tal manera que si volem guardar un número en una variable, haurem de fer-ho de la següent manera:

$numero = 2;

Funcions

Una funció és un conjunt de línies de codi que realitzen una feina específica i poden retornar un valor. Les funcions poden prendre paràmetres que modificaran el seu comportament.

```
//Declaració de la funció Sumar
  function Sumar($valor1, $valor2) {
    return ($valor1 + $valor2);
  }
//Crida de la funció Sumar
  Sumar(2, 3); //La funció retornarà 5
```

Imatge 22: Representació d'una funció en el llenguatge PHP.

Podem dividir l'exemple anterior en dues parts, la primera que és la declaració de la funció **Sumar** on li programem que ha de rebre dos valors com a paràmetres i després ha de retornar el resultat de la suma dels dos. Un cop la funció és declarada, aquesta no actuarà fins que se la cridi.

En la segona part, veiem com es crida la funció **Sumar** i se li transmet el número 2 i 3 com a paràmetres, de tal manera, que aquesta funció ens retornarà el número 5.

Classes

Una classe és una plantilla que serveix per la creació d'objectes a partir d'un model predefinit. Aquestes s'utilitzen per representar conceptes. Podríem dir que una classe és una barreja entre variables i funcions que s'uneixen per complir amb un objectiu. Per exemple, si volem crear una classe anomenada **Cotxe**, la podríem definir de la següent manera:

```php
class Cotxe {

  //PROPIETATS

    private $Marca;
    private $Model;
    private $Color;

  //FUNCIONS

    public function __construct($newMarca, $newModel, $newColor) {

      $this->Marca = $newMarca;
      $this->Model = $newModel;
      $this->Color = $newColor;

    }

    public function Engegar() {}

    public function Accelerar() {}

    public function Frenar() {}

    public function Apagar() {}

}
```

```php
$Cotxe = new Cotxe("Audi", "A5", "Blanc");

  $Cotxe->Engegar();
  $Cotxe->Accelerar();
```

Imatge 23: Representació d'una classe en el llenguatge *PHP*.

El contingut enquadrat amb vermell, s'hi defineix la classe **Cotxe** i li definim tres propietats que pot tenir, la **marca**, el **model** i un **color**. Aquestes propietats també les podem dir variables, ja que variaran depenent del cotxe escollit. També definim diferents funcions que un cotxe té**.** La primera d'elles es tracta del **constructor**, que és la funció que es crida quan creem l'objecte. A aquesta funció se li poden donar molts usos; nosaltres en aquest cas la fem servir per donar-li un valor a les variables de l'objecte. A més, tenim 4 funcions que representen 4 accions que pot fer un cotxe: engegar, accelerar, frenar o apagar.

A continuació, enquadrat de color verd, tenim una variable anomenada **Cotxe** i li guardem un Objecte que en aquest cas seria un "Audi A5" de color blanc. A partir d'aquest moment ja tenim un cotxe i li podem aplicar diferents funcions com Engegar i després Accelerar.
Com es pot veure, a partir d'aquesta classe, podríem crear diferents cotxes que tindrien en comú una plantilla (variables i funcions).

Annex 2: Codi de l'aplicació

En aquest annex podem trobar una llista completa de tots els arxius que integren l'aplicació. Primer de tot, trobem les interfícies d'usuari que són les encarregades de cridar als diferents controladors i a continuació hi ha els models, els controladors i les vistes de l'apartat de l'usuari i del maneig d'arxius.

Interfície d'usuari

- **Styles (Estils CSS):** Tot el codi que conforma el disseny del lloc web. (33 pàgines de codi)
- **SignUp (Registrar un nou compte):** Pàgina on hi ha el formulari de registre d'un compte. (2 pàgines de codi)
- **SignIn (Iniciar sessió amb un compte):** Pàgina on hi ha el formulari per tal d' iniciar sessió amb un compte existent. (2 pàgines de codi)
- **PasswordReset (Restablir la contrasenya d'un compte perdut):** Pàgina on trobem un formulari per tal de recuperar un compte del qual s'ha perdut la contrasenya. (2 pàgines de codi)
- **Download (Descarregar un arxiu compartit amb enllaç):** Pàgina que permet descarregar o important un arxiu. (3 pàgines de codi)
- **Me (Pàgina de l'aplicació):** Pàgina on trobem el gruix de l'aplicació. En aquesta part trobem la resposta d'**events** realitzats per l'usuari com pujar un arxiu, eliminar-lo, etc. (35 pàgines de codi)

Usuari

Aquest conjunt d'arxius engloba totes les accions que tenen un efecte en el maneig del compte de l'usuari, ja sigui des de registrar-se a l'aplicació web a canviar el correu electrònic.

Models

- **User.php (Usuari):** Conté totes les funcions relacionades amb la gestió del compte de l'usuari. (16 pàgines de codi)
- **Password.php (Contrasenya):** Conté un seguit de funcions per tal de poder recuperar el compte en cas de perdre'l. (2 pàgines de codi)
- **Session.php (Sessió):** Encarregat de gestionar les diferents sessions dels usuaris. (3 pàgines de codi)
- **Verification.php (Verificació):** Conjunt de funcions encarregades de la verificació de l'usuari per tal de poder atorgar certs privilegis a l'usuari. (2 pàgines de codi)

Controladors

- **checkEmail.php (Comprovar correu electrònic):** Comprovar si el correu electrònic desitjat està disponible. (1 pàgina de codi)
- **checkUsername.php (Comprovar nom d'usuari):** Comprovar si el nom d'usuari desitjat està disponible. (1 pàgina de codi)
- **Delete.php (Borrar compte):** Eliminar el compte de l'usuari (informació, arxius). (1 pàgina de codi)
- **SignIn.php (Entrar):** Comprovar si les credencials facilitades per l'usuari són correctes, i si ho són, crear una sessió per tal que s'identifiqui i així pugui accedir als seus arxius. (1 pàgina de codi)
- **SignOut.php (Sortir):** Eliminar la sessió i així sortir de l'aplicació. (1 pàgina de codi)
- **updatePassword.php (Actualitzar contrasenya):** Actualitzar la contrasenya. (1 pàgina de codi)

- **updateUsername.php (Actualitzar nom d'usuari):** Actualitzar el nom d'usuari. (1 pàgina de codi)
- **updateProfile.php (Actualitzar el perfil):** Actualitzar el perfil públic del compte. (1 pàgina de codi)
- **Verification.php (Verificació):** Verificar a partir del correu que és qui diu ser. (1 pàgina de codi)
- **sendPasswordToken.php (Establir un token de recuperació):** Permet enviar un correu amb un número secret que es farà servir per canviar la contrasenya del compte en cas que s'hagi perdut. (1 pàgina de codi)
- **resetPassword.php (Restablir contrasenya):** Permet canviar la contrasenya d'un compte en el qual s'ha perdut la contrasenya. (1 pàgina de codi)

Vistes

- **getAll.php (Obtenir tot):** Permet obtenir totes del dades del compte de l'usuari a partir d'una consulta a la base de dades. (1 pàgina de codi)
- **getSizes.php (Obtenir mides):** Permet obtenir a partir de vàries consultes a la base de dades l'espai utilitzat per l'usuari. (1 pàgina de codi)

Arxius

Aquest conjunt d'arxius engloba totes les accions que tenen un efecte en el maneig dels arxius, ja sigui des de pujar un arxiu a eliminar-lo.

Models

- **File.php (Arxiu):** Conté totes les funcions utilitzades pel maneig d'arxius (canviar el nom, eliminar-lo, etc.). (10 pàgines)
- **Link.php (Enllaç):** Conté tot un conjunt de funcions utilitzades per tal de crear un enllaç per compartir l'arxiu. (3 pàgines)

Controladors

- **Copy.php (Copiar):** Copiar un arxiu. (1 pàgina de codi)

- **temporaryDelete.php (Eliminar temporalment):** Eliminar un arxiu/carpeta de forma temporal. (1 pàgina de codi)
- **permanentDelete.php (Eliminar permanentment):** Eliminar un arxiu/carpeta de forma permanent. (1 pàgina de codi)
- **Restore.php (Restaurar):** Restaurar un arxiu que ha sigut eliminat de forma temporal. (1 pàgina de codi)
- **Download.php (Descàrrega):** Descarregar un arxiu/carpeta. (1 pàgina de codi)
- **Favorite.php (Favorit):** Posar un arxiu a preferits. (1 pàgina de codi)
- **Folder.php (Carpeta):** Crear una carpeta. (1 pàgina de codi)
- **Import.php (Importar):** Permet importar un arxiu que està compartit o en què hi ha un enllaç. (1 pàgina de codi)
- **Move.php (Moure):** Moure un arxiu/carpeta de lloc. (1 pàgina de codi)
- **Rename.php (Canviar nom):** Canviar el nom d'un arxiu/carpeta. (1 pàgina de codi)
- **Share.php (Compartir):** Compartir un arxiu amb un usuari de l'aplicació. (1 pàgina de codi)
- **Size.php (Mida):** Saber la mida concreta d'un arxiu. (1 pàgina de codi)
- **Upload.php (Pujada):** A partir d'un conjunt d'arxius i fent ús de la tècnica de *file chunking*, permet pujar arxius de grans d'una forma segura. (3 pàgines de codi)
- **setLink.php (Establir enllaç):** Estableix un enllaç per compartir un arxiu/carpeta concrets. (1 pàgina de codi)
- **deleteLink.php (Eliminar enllaç):** Elimina l'enllaç d'un arxiu. (1 pàgina de codi)
- **updateLink.php (Actualitzar enllaç):** Permet actualitzar l'enllaç de l'arxiu/carpeta que es vol compartir. (1 pàgina de codi)

Vistes

- **getOwnFiles.php (Obtenir arxius propis):** Permet obtenir tots els arxius del quals l'usuari n'és propietari. (4 pàgines de codi)
- **getFavorites.php (Obtenir preferits):** Permet obtenir només els arxius que estan marcats com a favorits. (4 pàgines de codi)
- **getSharedWithYou.php (Obtenir compartits amb tu):** Permet obtenir tots els arxius que han estat compartits amb l'usuari. (4 pàgines de codi)

- **getSharedWithOthers.php (Obtenir compartits amb altres):** Permet obtenir tots els arxius que l'usuari ha compartit amb altres. (4 pàgines de codi)
- **getSearch.php (Obtenir cerca):** Permet obtenir els arxius que compleixen amb les especificacions de la cerca. (4 pàgines de codi)
- **getLocation.php (Obtenir localització):** Mostra l'ubicació en la qual l'usuari es troba en aquell moment. (1 pàgina de codi)
- **getDesiredFile.php (Obtenir l'arxiu desitjat):** Permet veure les característiques d'un arxiu que ha estat compartit a partir d'un enllaç. (1 pàgina de codi)

En total hi ha **166** pàgines de codi que permeten el correcte funcionament de l'aplicació. A continuació, deixo un enllaç a l'aplicació on es pot descarregar el projecte complet i on es poden veure els diferents codis:

https://doublescloud.com/downloadl/?u=73fcd1fbd108e0544a5dbf8946b28b49&p=ed4988848f6fe12a22efd51061a7db38

Annex 3: Bases de dades de l'aplicació

En aquest projecte hem utilitzat el sistema de gestió de base de dades *MySQL,* on hem creat una base de dades formada per 6 taules que serveixen per emmagatzemar tota la informació necessària per a l'aplicació:

Files (Arxius)

Aquest taula s'ha utilitzat per emmagatzemar la informació dels fitxers de l'usuari.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	Id	int(10)			No	*Ninguna*		AUTO_INCREMENT
☐	2	Location	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
☐	3	Name	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
☐	4	Type	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
☐	5	Size	int(10)			Sí	*NULL*		
☐	6	Owner	int(10)			No	*Ninguna*		
☐	7	Shared	varchar(200)	utf8_unicode_ci		Sí	*NULL*		
☐	8	Date	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
☐	9	EncryptedName	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
☐	10	Status	varchar(50)	utf8_unicode_ci		No	*Ninguna*		
☐	11	Favorite	tinyint(1)			No	*Ninguna*		

Imatge 24: Taula de Files (Arxius).

- **Id:** Qualsevol número natural, que és únic per a cada cada arxiu (p.e. **1**).
- **Location:** Localització de l'arxiu, o sigui, la carpeta on està (p.e. **/151051241681609379/**).
- **Name:** Nom de l'arxiu (p.e. **nom_imatge.png**).
- **Type:** Format de l'arxiu emmagatzemat (p.e. **.jpeg**).
- **Size:** Mida de l'arxiu en bytes (p.e. **1243**).
- **Owner:** Id de l'Usuari propietari de l'arxiu (p.e. **2**).

- **Shared:** Diferents Ids dels Usuaris que tenen accés a l'arxiu separats per dos punts per facilitar el maneig d'aquests en el codi (p.e. **.2..3..5.**).
- **Date:** Data en què l'arxiu ha estat pujat (p.e. **2017-11-12 18:48:25**).
- **EncryptedName:** Nom únic que se li dona a l'arxiu en el sistema de fitxers per tal que no es sobreescrigui amb un altre (p.e. **15105125021401509064.jpg**).
- **Status:** Comprovar si l'arxiu es troba a la paperera o no. Aquest camp pot tenir 2 valors: Active o Inactive (Paperera). (p.e. **Active**).
- **Favorite:** Comprovar si l'arxiu està a la carpeta de favorits: true o false (p.e. **true**).

Links (Enllaços)

Aquesta taula s'ha utilitzat per guardar els diferents enllaços creats per l'usuari per poder compartir els seus fitxers mitjançant l'opció de crear un enllaç per compartir un fitxer.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	Id	int(10)			No	*Ninguna*		
☐	2	Link 🔑	varchar(256)	utf8_unicode_ci		No	*Ninguna*		
☐	3	Password	varchar(256)	utf8_unicode_ci		No	*Ninguna*		

Imatge 25: Taula de Links (Enllaços).

- **Id:** Qualsevol número natural únic que té cada enllaç (p.e. **1**).
- **Link:** Enllaç que li correspon a l'arxiu (p.e. **762015a2842066c9213a2bba7498947d**).
- **Password:** Contrasenya que pot tenir l'arxiu per tal d'accedir a ell (p.e. **6d4f02eb0e6d4e22575206b972f378ba**).

Logs (Registers de navegació)

Aquesta taula serveix per emmagatzemar les diferents accions que l'usuari realitza en l'aplicació: des de iniciar sessió a borrar el compte.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**Id**	int(10)			No	*Ninguna*		
2	**Datetime**	datetime			No	*Ninguna*		
3	**Event**	text	utf8_unicode_ci		No	*Ninguna*		
4	**Data**	text	utf8_unicode_ci		No	*Ninguna*		

Imatge 26: Taula de Logs (Registres de navegació).

- **Id:** Qualsevol número natural únic que té cada registre (p.e. **3**).
- **Datetime:** Dia i hora que l'event ha tingut lloc (p.e. **2017-11-12 18:48:25**).
- **Event:** L'acció o event que s'ha produït (p.e. **user.login**).
- **Data:** Informació extra de l'acció o event (p.e. **User logged from 88.1.54.12**).

Password Recovery (Recuperació contrasenyes)

Taula que serveix per emmagatzemar els diferents codis de recuperació dels comptes de l'usuari en cas de perdre la contrasenya.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	**Id**	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	**Token**	varchar(128)	utf8_unicode_ci		No	*Ninguna*		
3	**Access**	int(11)			No	*Ninguna*		

Imatge 27: Taula de Password_Recovery (Recuperació contrasenyes).

- **Id:** Qualsevol número natural únic que té cada recuperació de contrasenya (p.e. **12**).
- **Token:** Unió de 2 *hashes sha256* creat aleatoriament que representa la clau per canviar la contrasenya (p.e. **6a38f071eb350cf55620882be69a3ab44c17c2762b5b5f5f03816d3bce32a808eadfc2670acbd0229973655aa15bdd5eccc9cfb0b4812e35fd1d1d3af8ad44ee**).

- **Access:** Hora en que s'ha demanat la recuperació de contrasenya en format UNIX i mil·lisegons (p.e. **1509558011**).

Sessions (Sessions)

Taula que serveix per guardar les diferents sessions iniciades en l'aplicació.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
☐	1	Id 🔑	varchar(32)	utf8_unicode_ci		No	*Ninguna*		
☐	2	Access	int(11)			No	*Ninguna*		
☐	3	Data	text	utf8_unicode_ci		No	*Ninguna*		

Imatge 28: Taula de Sessions (Sessions).

- **Id:** Unió de diferents caràcters únics que té cada sessió (p.e. **0441i2v8u0ltvin51hbs0quei7**).
- **Access:** Hora en que s'ha iniciat la sessió en format UNIX i mil·lisegons (p.e. **1509214390**).
- **Data:** Informació guardada en la sessió (p.e. **userId|s:3:"106";Username|s:19:"desteba21@gmail.com";**).

Users (Usuaris)

Taula que serveix per registrar els diferents usuaris de l'aplicació amb les seves dades.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	Id	int(10)			No	*Ninguna*		AUTO_INCREMENT
2	Name	varchar(200)	utf8_unicode_ci		Sí	*NULL*		
3	Username	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
4	Email	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
5	Password	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
6	Location	varchar(200)	utf8_unicode_ci		Sí	*NULL*		
7	Description	text	utf8_unicode_ci		Sí	*NULL*		
8	Gender	varchar(20)	utf8_unicode_ci		No	*Ninguna*		
9	Image	varchar(200)	utf8_unicode_ci		No	*Ninguna*		
10	Size	int(10)			No	*Ninguna*		
11	VOE	tinyint(1)			No	*Ninguna*		

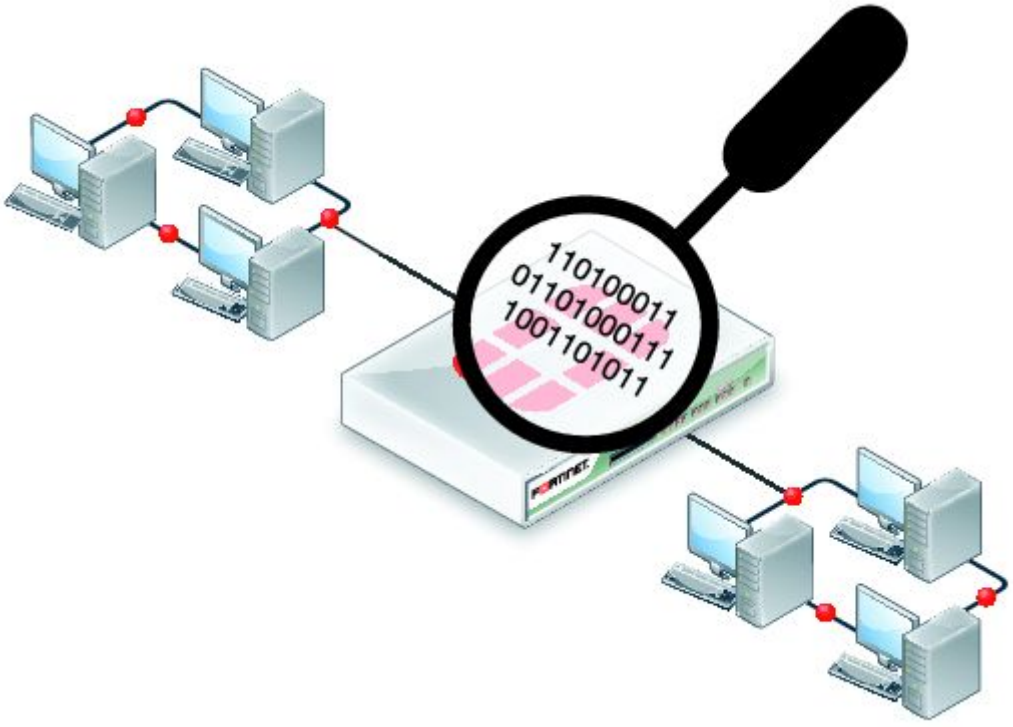
Imatge 29: Taula de Users (Usuaris)

- **Id:** Número natural únic que té cada usuari (p.e. **3**).
- **Name:** Nom del propietari del compte (p.e. **David Esteba Fàbrega**).
- **Username:** Nom d'usuari únic (p.e. **desteba**).
- **Email:** Correu electrònic de l'usuari (p.e. **desteba21@gmail.com**).
- **Password:** Contrasenya guardada amb *hash* + *salt* (p.e. **$2y$10$HwHvp7nMJSbTXKdet8WQW.vH0r.PcQz4T5uayUuq/3fazmbXAMQlW**).
- **Location:** Lloc de residència de l'usuari (p.e. **Canet d'Adri**).
- **Description:** Descripció de l'usuari (p.e. **Apassionat per la tecnologia**).
- **Gender:** Gènere de l'usuari (p.e. **Masculí**).
- **Image:** Imatge de perfil de l'usuari (p.e. **15160508176989306l9.jpg**).
- **Size:** Quantitat d'espai utilitzat per l'usuari en bytes (p.e. **123512**).
- **VOE:** Verificació del correu electrònic que pot ser *true* en cas que el compte estigui verificat o *false* en cas que no ho estigui (p.e. **true**).

Annex 4: Exemples de vulnerabilitats

Packet sniffing degut a la manca d'HTTPS

S'anomena *Packet sniffing* a l'acció de posar-se al mig de la comunicació entre una màquina A i una màquina B i fer passar tots el paquets que s'intercanvien per la teva màquina.

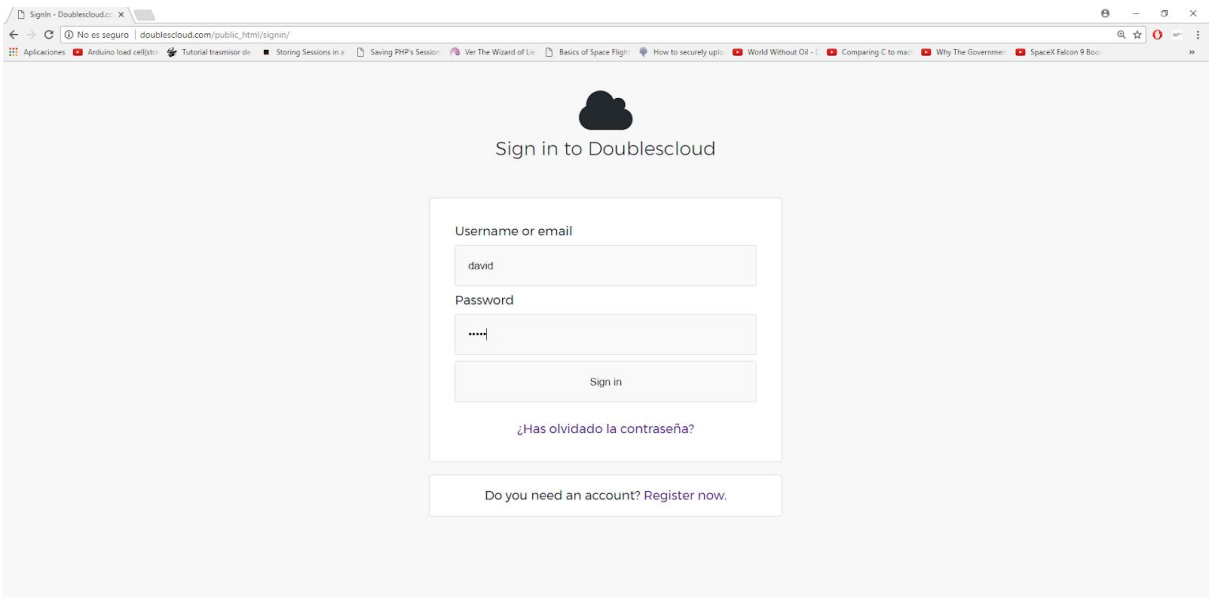


Imatge 30: Representació gràfica de *packet sniffing*.

Aquesta acció és possible sempre que la màquina A intercanviï informació amb la maquina B i aquesta comunicació no estigui xifrada amb algun tipus de protocol, com per exemple el **HTTPS**.

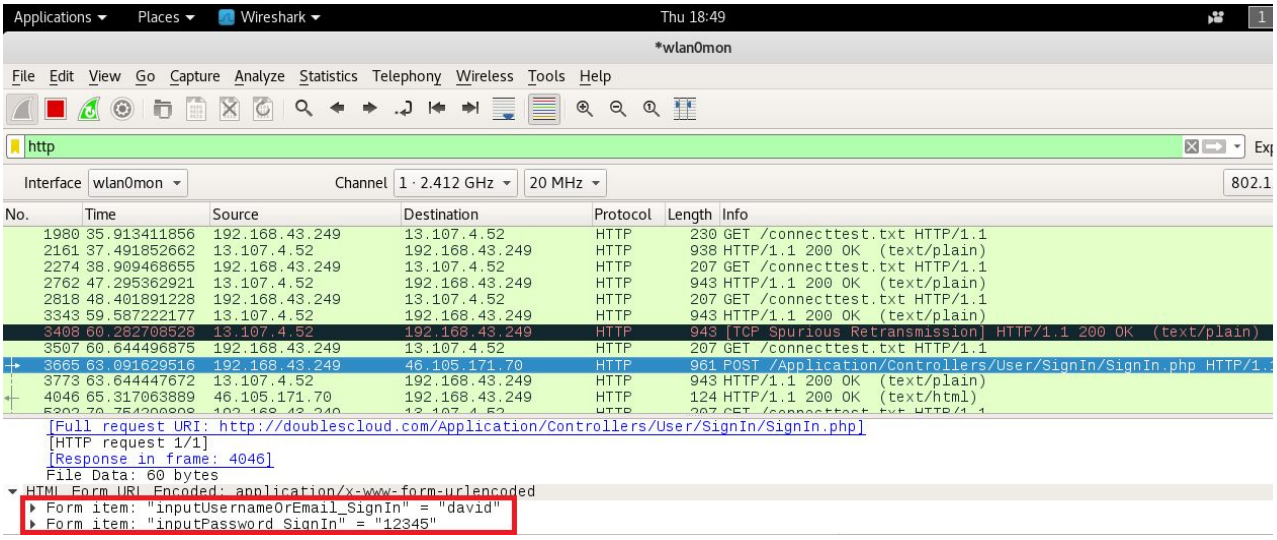
A continuació elaborarem un exemple en el que l'usuari A vol visitar la pàgina que es troba a la màquina B i per fer això es connectarà a una **Wi-Fi**. El que faré serà posar la

meva màquina en mode monitor i així aconseguir tots els paquets que s'intercanviïn. Com hem dit, si la connexió no està xifrada per un protocol **HTTPS** o bé la xarxa **Wi-Fi** a la que està connectat l'usuari A té una contrasenya que l'atacant desconeix, serà capaç de interceptar tot el trànsit. Anem a provar amb un formulari del lloc web que no està protegit amb **HTTPS**. En aquest exemple enviarem un nom d'usuari que serà *david* i una contrasenya que serà *12345* i intentarem interceptar-los.



Imatge 31: Formulari web.

Un cop enviada la informació, el que faré serà comprovar si el programa que fem servir, en aquest cas *Wireshark*, ha pogut capturar el paquet amb la informació del formulari.



Imatge 32: Programa Wireshark amb les dades interceptades.

Podem observar que el programa ha sigut capaç d'interceptar diferents paquets i en aquest cas hem pogut interceptar el paquet corresponent a la informació desitjada.

Això implica que l'atacant podrà llegir correus electrònics, contrasenyes, números de targetes, pins, etc.

Per sort, gran part dels llocs webs que trobem avui en dia i que requereixen d'una autentificació, disposen d'un xifrat que ens permet una transmissió segura de dades com és en el nostre cas.