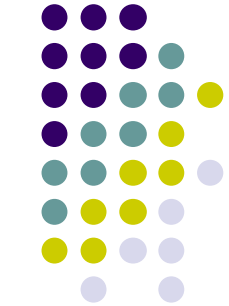


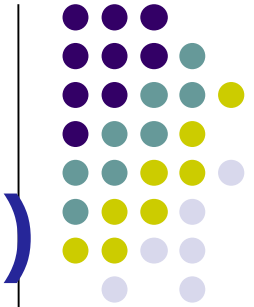
G.7: Cookie-ak, Sesioak eta Egoeraren mantentzea HTTPn.

Webeko informazio-sistemen segurtasuna

Rosa Arruabarrena, Jose Ángel Vadillo

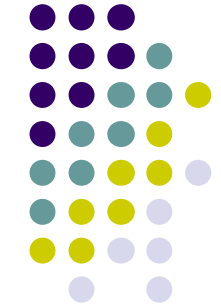
LSI, UPV/EHU

Sesioa (saioa) eta *session* objektua (I)



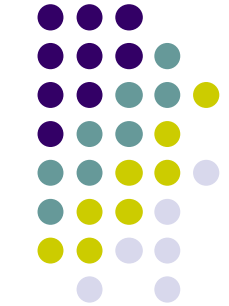
- HTTP protokoloak ez du eskaeren arteko egoera mantentzen
- Hala ere, maiz, arakatzaile-zerbitzari elkarren artean erlazionatuak dauden (request-response) interakzio multzoek, euren arteko taldekatze logikoa behar dute.
- Taldekatze hori **sesioa (saioa)** kontzeptuarekin bat dator

Sesioa eta *session* objektua (II)



- Sesio kontzeptua gauzatzeko ***session*** objektua erabiltzen da
  - *Session* objektu bat sortuko da web-zerbitzari eta bezero arteko interakzioa multzo bat talde bezala maneiatu behar den aldi bakoitzean
  - Web-aplikazioak **zerbitzarian** (aldi baterako) *session*-aren instantzia bat sortzen du, identifikatzaile bakarrekoa
  - *Session*-aren instantzia honetan balio globalak gordetzen dira, orri desberdinek sesio beraren barnean erabiliko dituztenak (adib. , sesioaren identifikatzaile bera, erabiltzaileren izena, balio akumulatuak, ...).

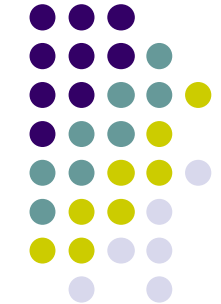
session objektua (III)



- Erreminta hau, normalki *kautotzea* (autentikazioa) eta erabiltzaileen jardueren jarraipena egiteko erabiltzen da, atal pribatuak dituzten webetan haien atzipen-kontrola egin behar denean
  - Behin erabiltzailea kautotu denean, **sesioaren identifikatzailea joan-etorri** tiketa bat balitz bezala erabil liteke, zenbait orritara sartzeko baimena emango diolarik, berriz ere kautotu gabe

- Sesioaren maneiuak atzipenen kontrola eta ikuskapena errazten eta bateratzen ditu. Baina, web-aplikazioak ahuleziaren bat balu, aplikazio osoaren segurtasuna honda lezake

Sesioaren IDentifikadorea (SID)



- Luzera handiko ausazko sekuentzia bat izan ohi da, aplikazioaren atzipena eskatu duen nabigatzaileari *Cookie* bidez igortzen zaiona

- SID-ari esker une zehatz batean aplikazio berdinarekin interakzionatzen/elkarreragiten ari diren bezeroak identifika edota bereiz litezke

- Zerbitzari aldeko aplikazioek datuak gordetzen dituzte *session* objektuan, nabigatzaile horrek exekuzio “berean” egiten dituen atzipen desberdinetan datu horiek eskuragarri egon daitezen

SID-a egoera gordetzeko

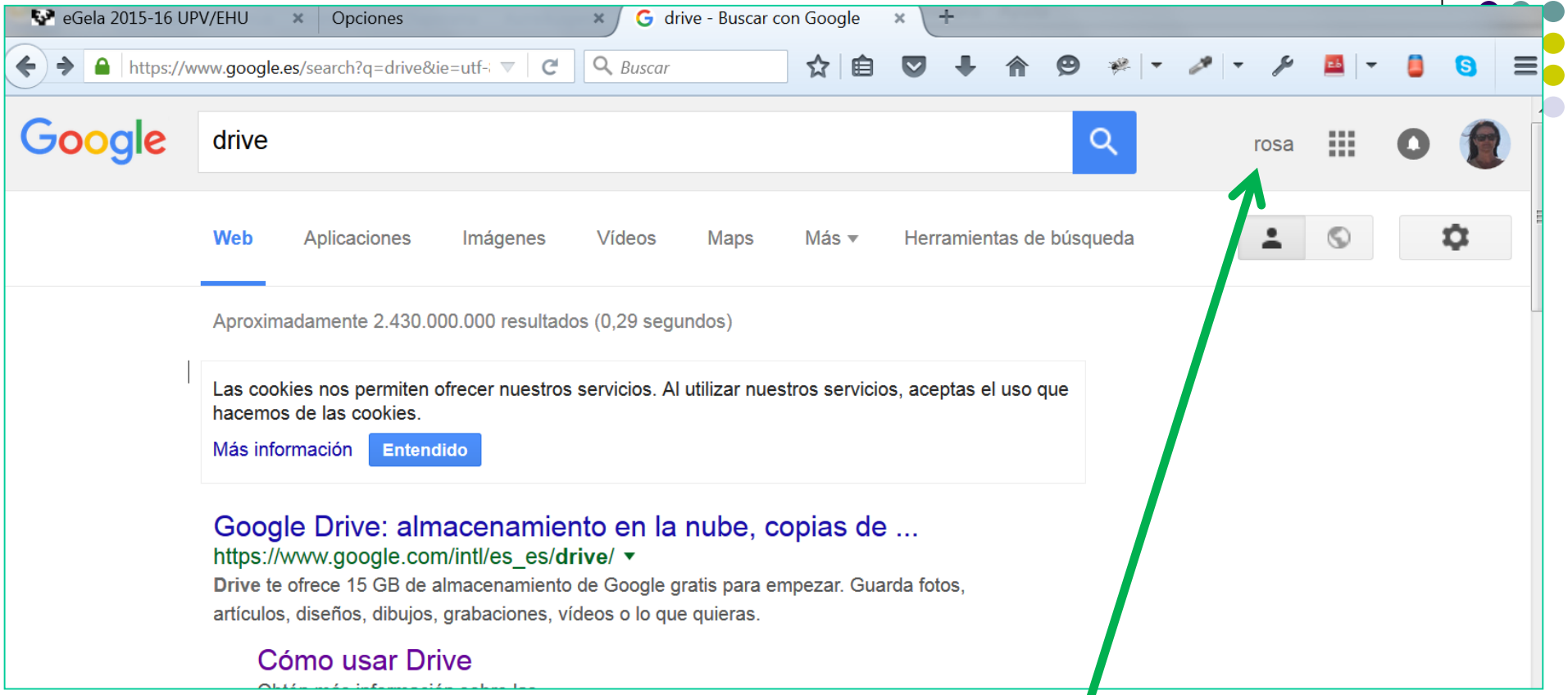
- SID-ak bidaltzeko eta jasotzeko aukerak:

  Lehen

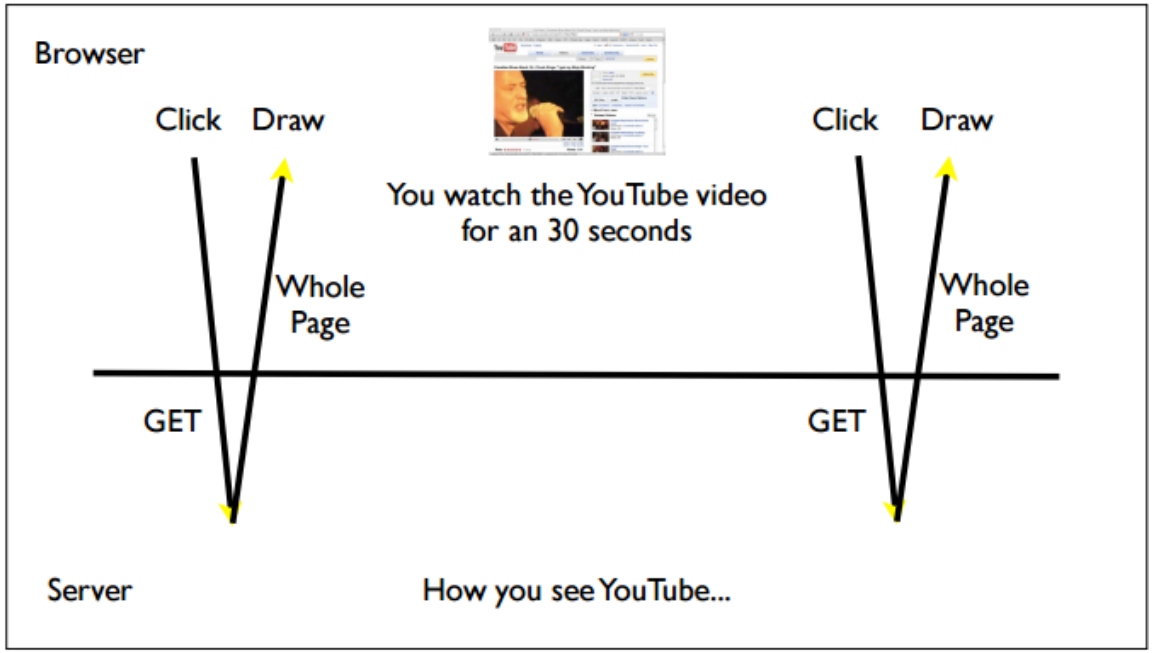
  - Orriaren URLean SIDa gehituz (GET erabilitz)
  - Formularioko zenbait eremutan SIDa metatuz, POST metodo bidez igorriko dena. Normalki *hidden* moduko eremuak erabili ohi dira informazio hau metatzeko

  Egun

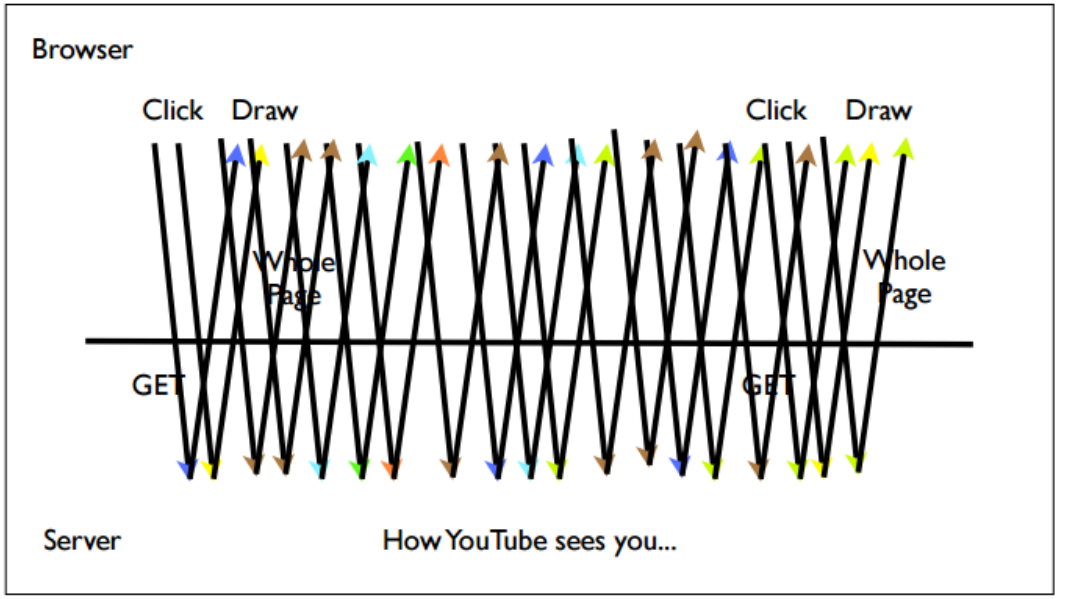
  - *Cookie-n erabilpen bidez.*



Google-k, logina egin baduzu, hura “gogoratzen” du



Itxuraz badirudi ere …



Errealitatea bestelakoa da …

Iturria: http://www-personal.umich.edu/~csev/books/gae/lectures/07-Cookie-Session-Print.pdf

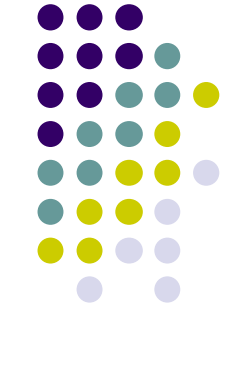
Cookie-ak

- Cookie-ak HTTP protokoloaren gabezi bat gainditzeko gehitu ziren. Hain zuzen, protokolo horrek bezero-zerbitzarien arteko elkarrekintzen arteko konexioa ezin gorde izanagatik

- Cookie-ak informazio puskak dira (ez kodea), gehienez 4KB-eko tamaina izango dutenak eta zerbitzaritik bezerora eta alderantziz igorriko direnak http trama barnean

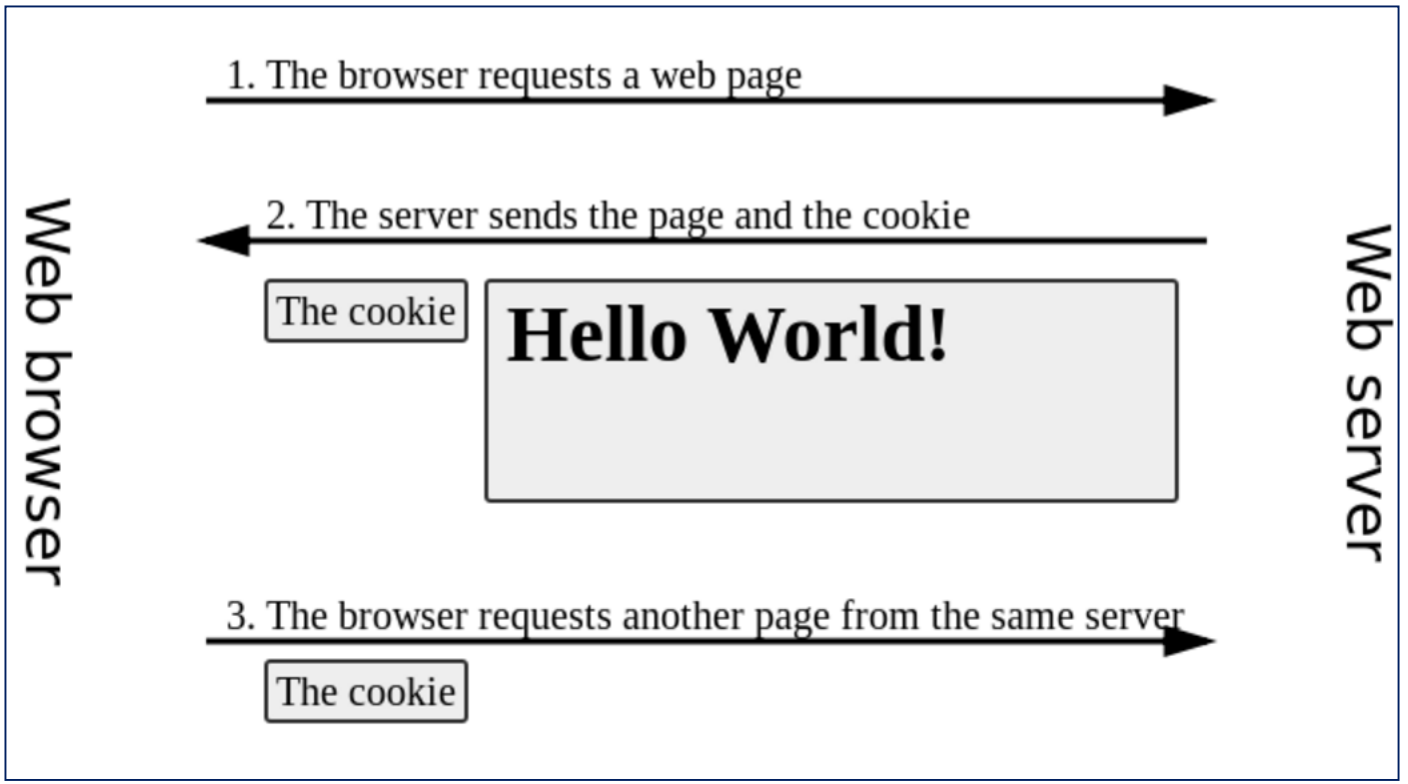
- Nabigatzaileek cookientzat biltegi bat dute. Cookie-ak domeinu bati erlazionatzen/lotzen dira. Horrela, nabigatzaileak domeinu bati eskaera bat luzatu behar dioenean, cookie-rik erlazionaturik duen baieztatuko du eta, hala balitz, *http-request*-ren atal bat bezala gehituko lioke informazio hori.

Cookie-ak(II)



- Zerbitzariak *http-response*-an cookie bat gehitu badu, nabigatzaileak *response*a prozesatu ostean, cookie-aren balio berria eguneratuko du, igorri duen domeinuari lotuz

- Konputagailu berean arakatzaile bat baino gehiago erabiltzen bada, bakoitzak bere cookie biltegia dauka.
  - Eta cookieek ez dute pertsona bat identifikatzen: erabiltzailearen kontua, konputagailu eta arakatzailearen konbinazioa identifikatzen dute.
  - Hori horrela, hainbat kontu, hainbat konputagaiu edota hainbat arakatzaile erabiltzen dituen edonork, cookie multzo anitz ditu.

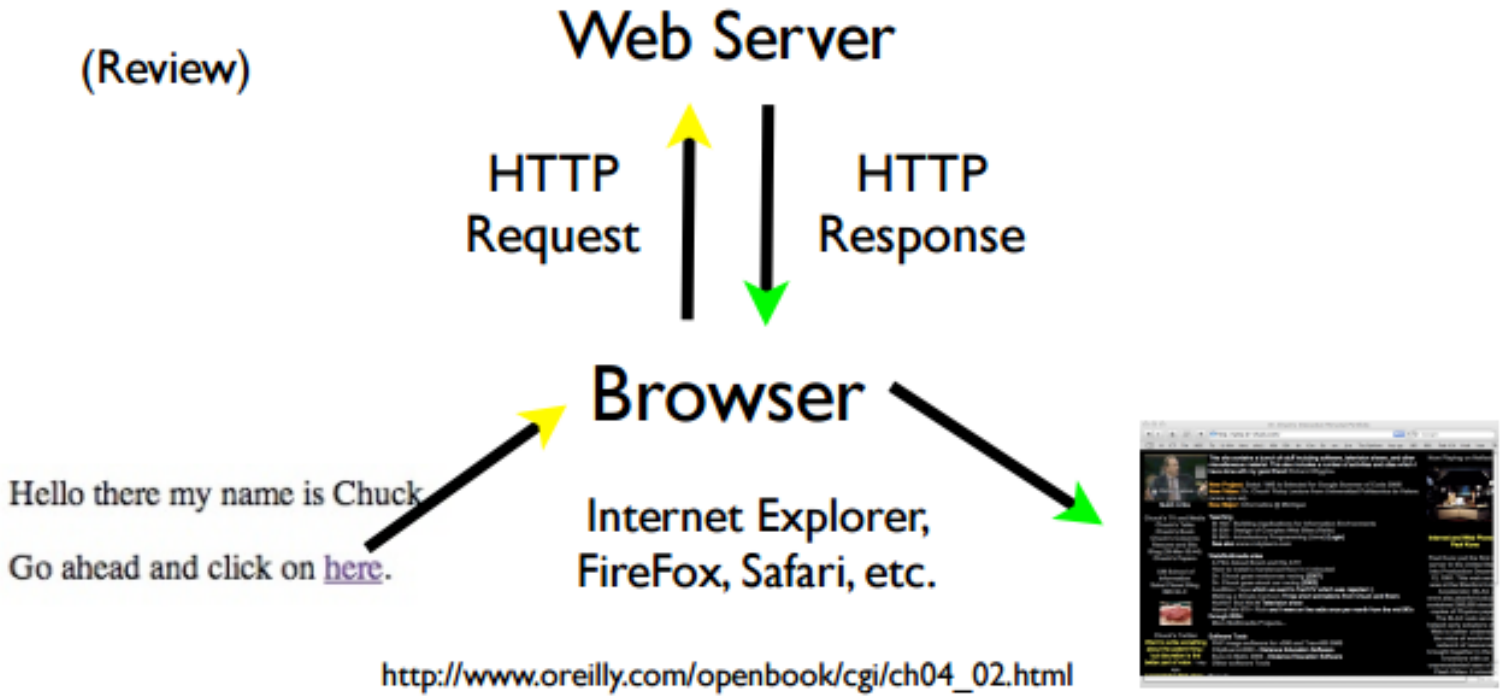


A possible interaction between a web browser and a server holding a web page in which the server sends a cookie to the browser and the browser sends it back when requesting another page.

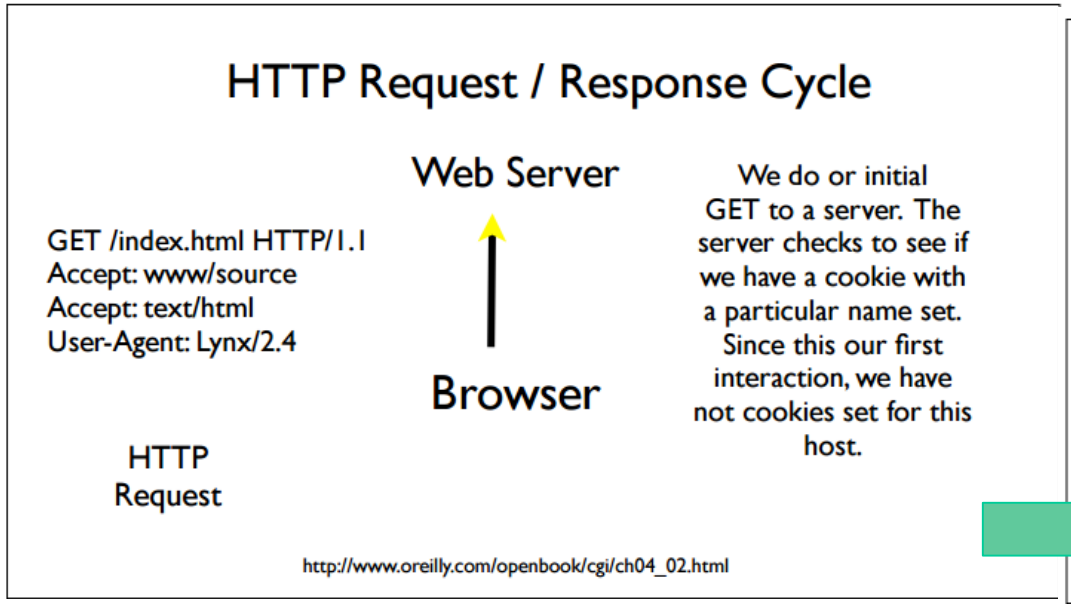
Source: https://en.wikipedia.org/wiki/HTTP_cookie

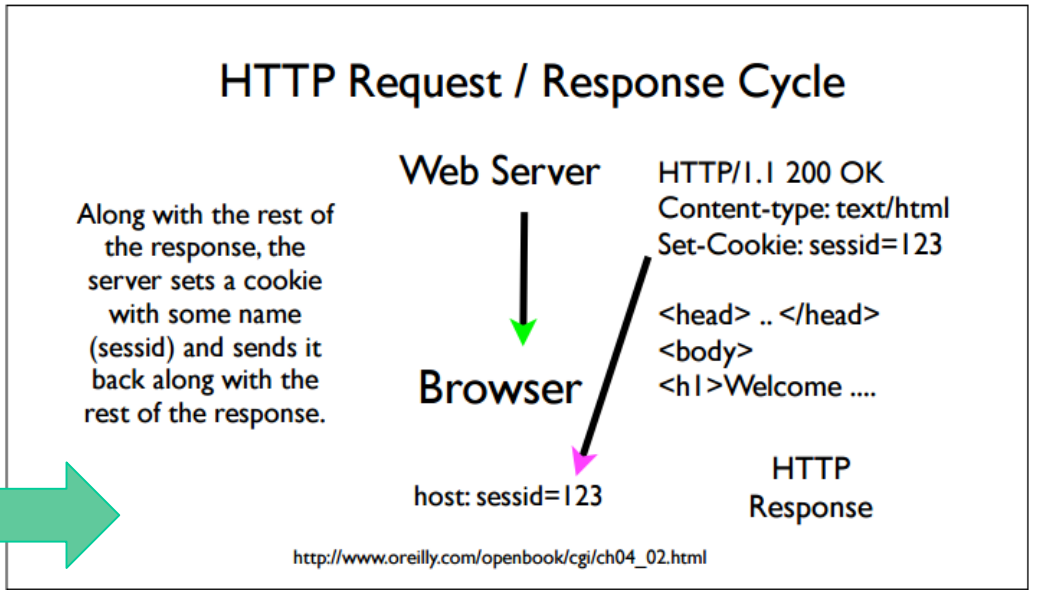
HTTP Request / Response Cycle

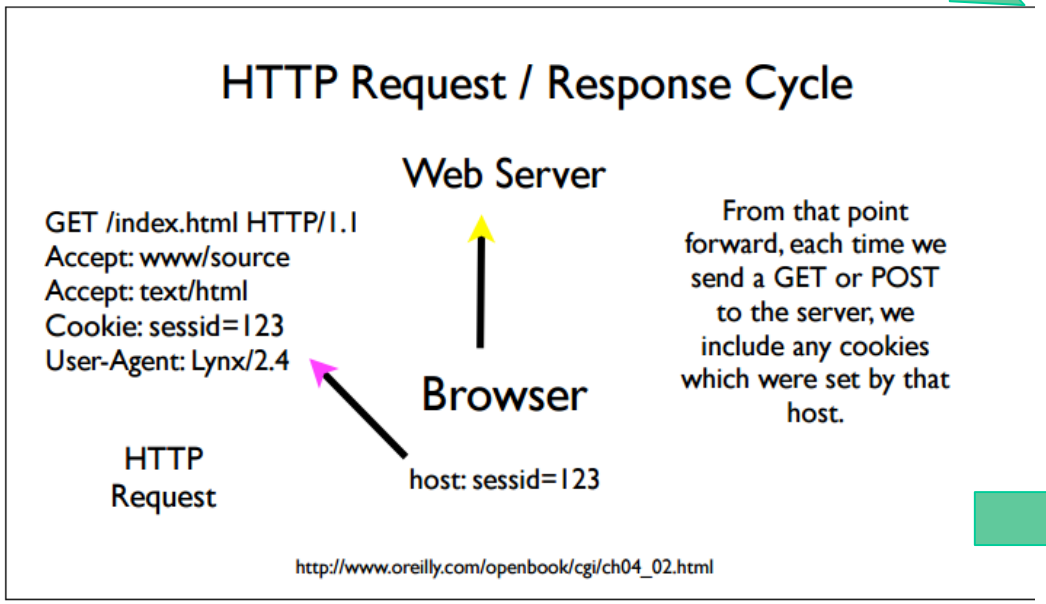
(Review)

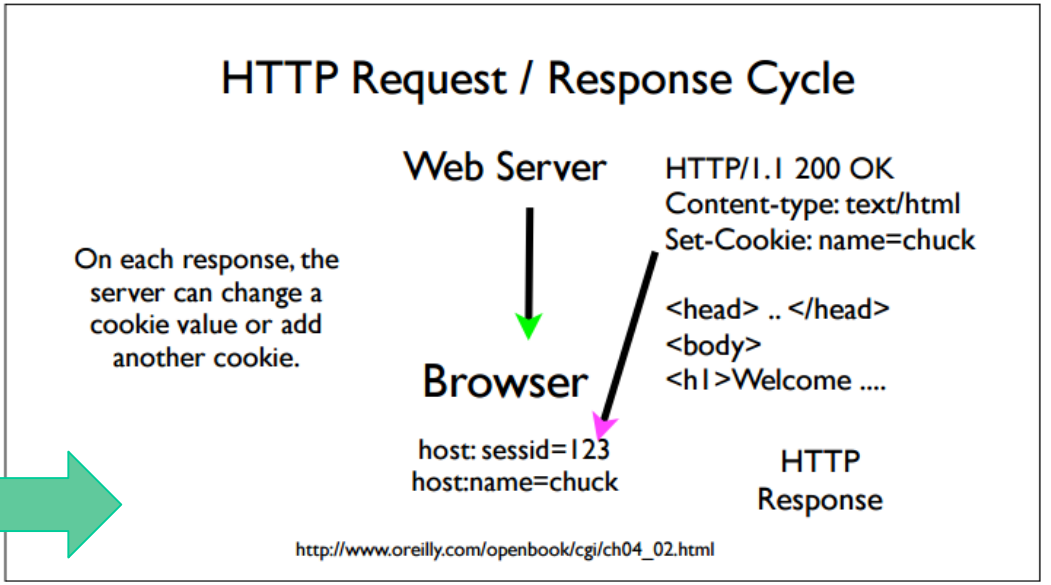


http://www.oreilly.com/openbook/cgi/ch04_02.html

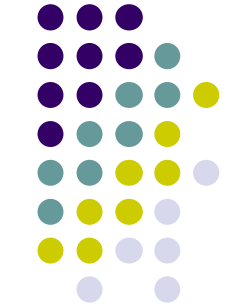

HTTP Request / Response Cycle
Web Server
GET /index.html HTTP/1.1
Accept: www/source
Accept: text/html
User-Agent: Lynx/2.4
Browser
HTTP
Request
We do or initial GET to a server. The server checks to see if we have a cookie with a particular name set. Since this our first interaction, we have not cookies set for this host.
http://www.oreilly.com/openbook/cgi/ch04_02.html


HTTP Request / Response Cycle
Web Server
Along with the rest of the response, the server sets a cookie with some name (sessid) and sends it back along with the rest of the response.
Browser
host: sessid=123
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: sessid=123
<head> .. </head>
<body>
<h1>Welcome ....
HTTP
Response
http://www.oreilly.com/openbook/cgi/ch04_02.html


HTTP Request / Response Cycle
Web Server
GET /index.html HTTP/1.1
Accept: www/source
Accept: text/html
Cookie: sessid=123
User-Agent: Lynx/2.4
Browser
HTTP
Request
host: sessid=123
From that point forward, each time we send a GET or POST to the server, we include any cookies which were set by that host.
http://www.oreilly.com/openbook/cgi/ch04_02.html


HTTP Request / Response Cycle
Web Server
On each response, the server can change a cookie value or add another cookie.
Browser
host: sessid=123
host:name=chuck
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name=chuck
<head> .. </head>
<body>
<h1>Welcome ....
HTTP
Response
http://www.oreilly.com/openbook/cgi/ch04_02.html

Nabigatzaileak eta cookie-ak



- Nabigatzaileak *cookie* bat jasotzen duenean, domeinu bati erlazionaturik gordeko du hura, izen bat eta balio bat izango dituelarik gutxienez

- Cookie-ak "*iraungitze-data*" izan dezakete edo iraunkorrak izan litezke

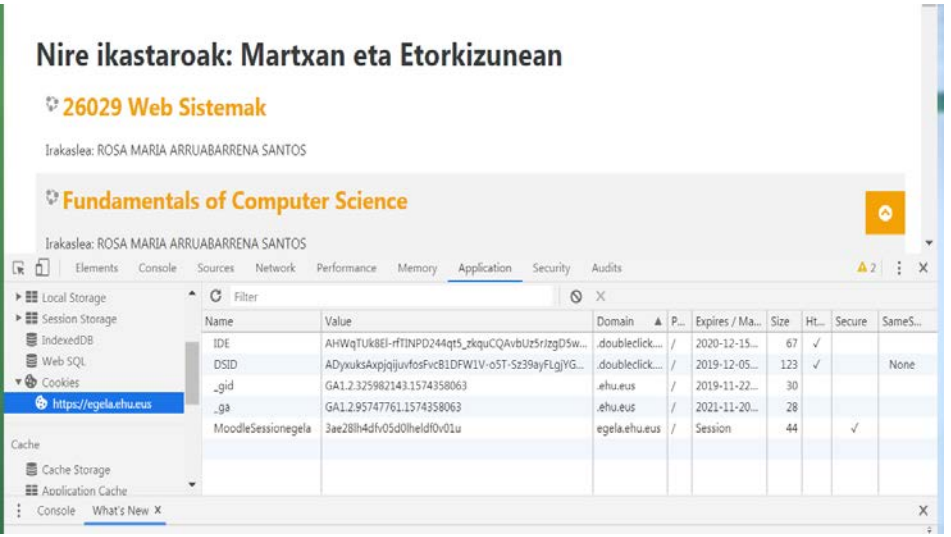
- Cookie-ak domeinu bateko *path* batetara mugatuak egon litezke
- Bestalde, gerta liteke cookie-a "*blindatua*" egotea ere, eta soilik http barneko protokoloaren testuinguruan atzigarri izatea (eta ez, adibidez, JavaScriptatik)

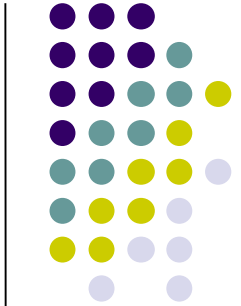
Nabigatzaileak eta cookie-ak (II)

- Nabigatzaileek erabilgarritasun espezifikoak eta *pluggin*-ak dituzte cookie-ak kontsultatzeko eta eguneratzeko, bai eta haien erabilera mugatzeko.

- Erabiltzaileen sesioak kudeatzeko, ezarpen pertsonalizatuak metatzeko, eta erabiltzailearen portaeraren jarraipena egiteko erabilgarriak

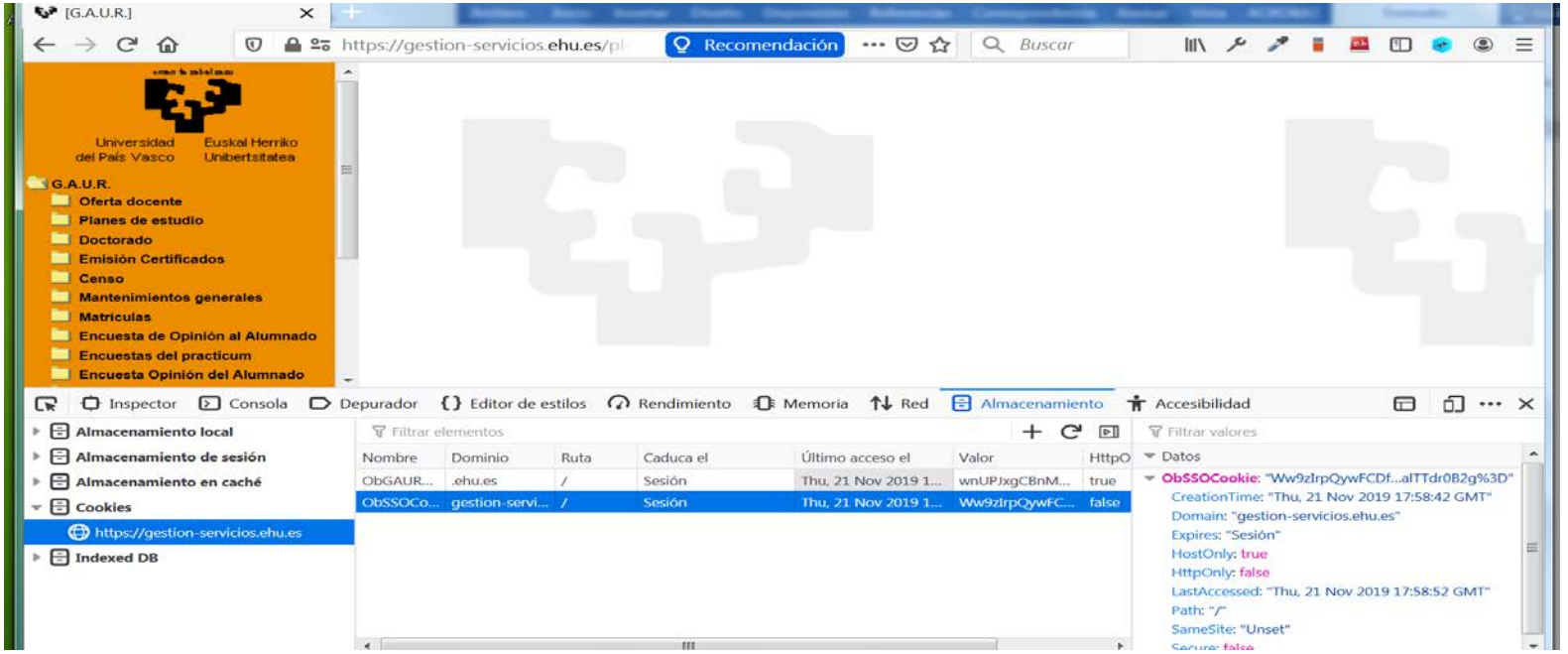
- Gogoratu: Cookie-ak datuak dira, ez kodea
  ➔ ezin dute erabiltzaileen konputagailuko informazioa ez irakurri ez ezabatu



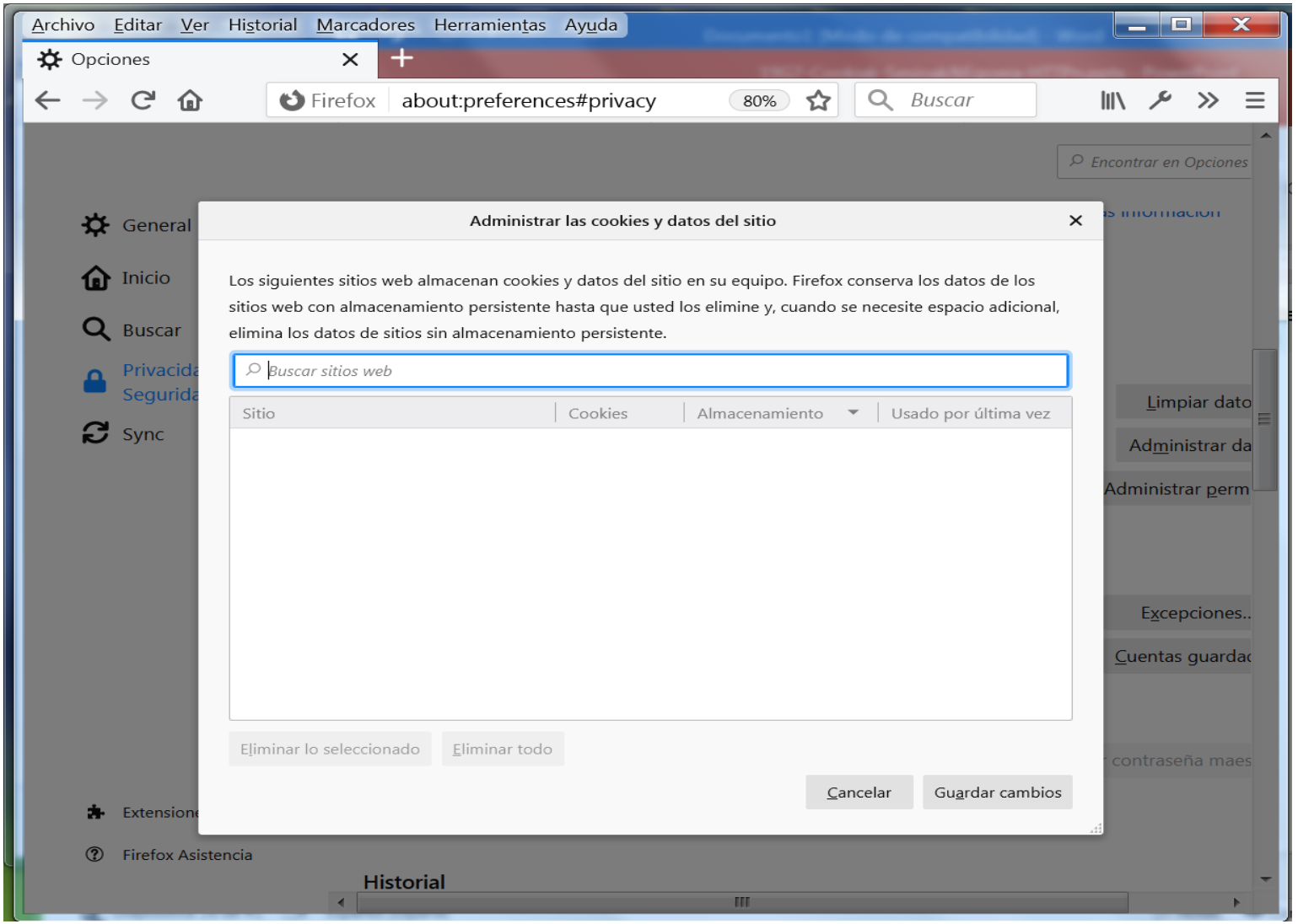


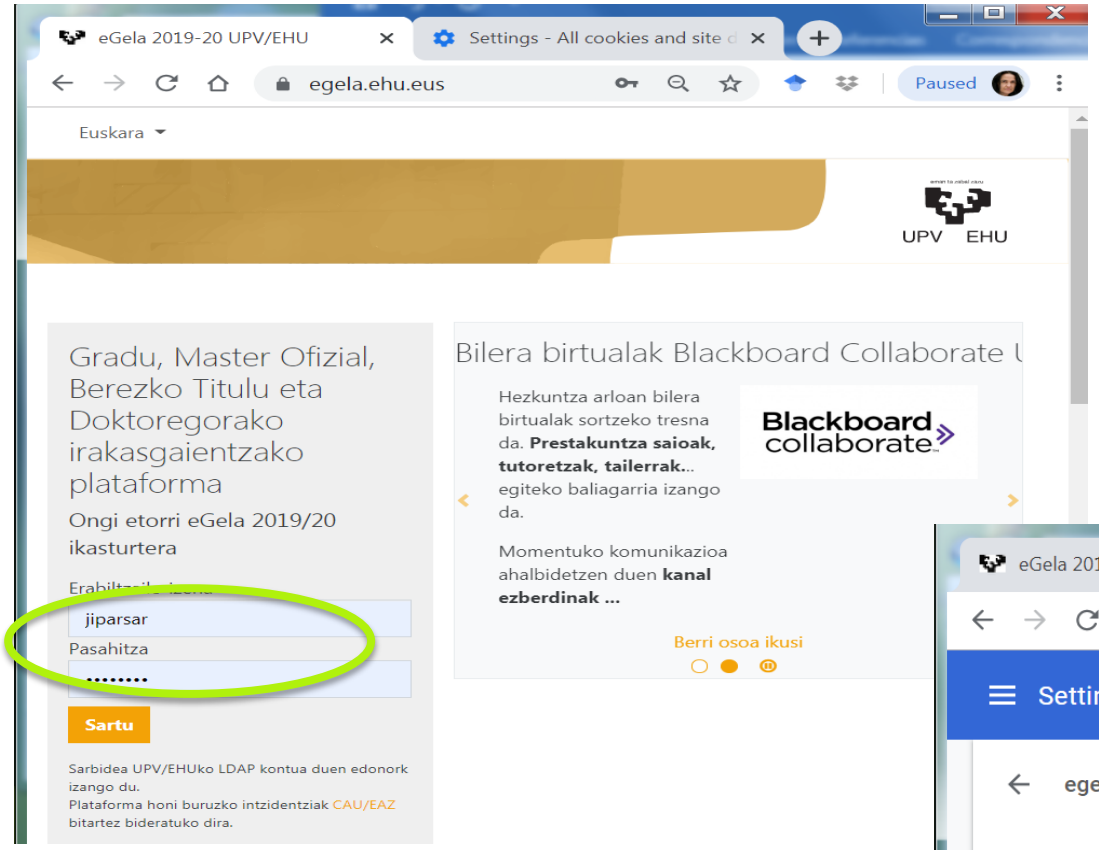
[Chrome](): consola + application + storage + Cookies

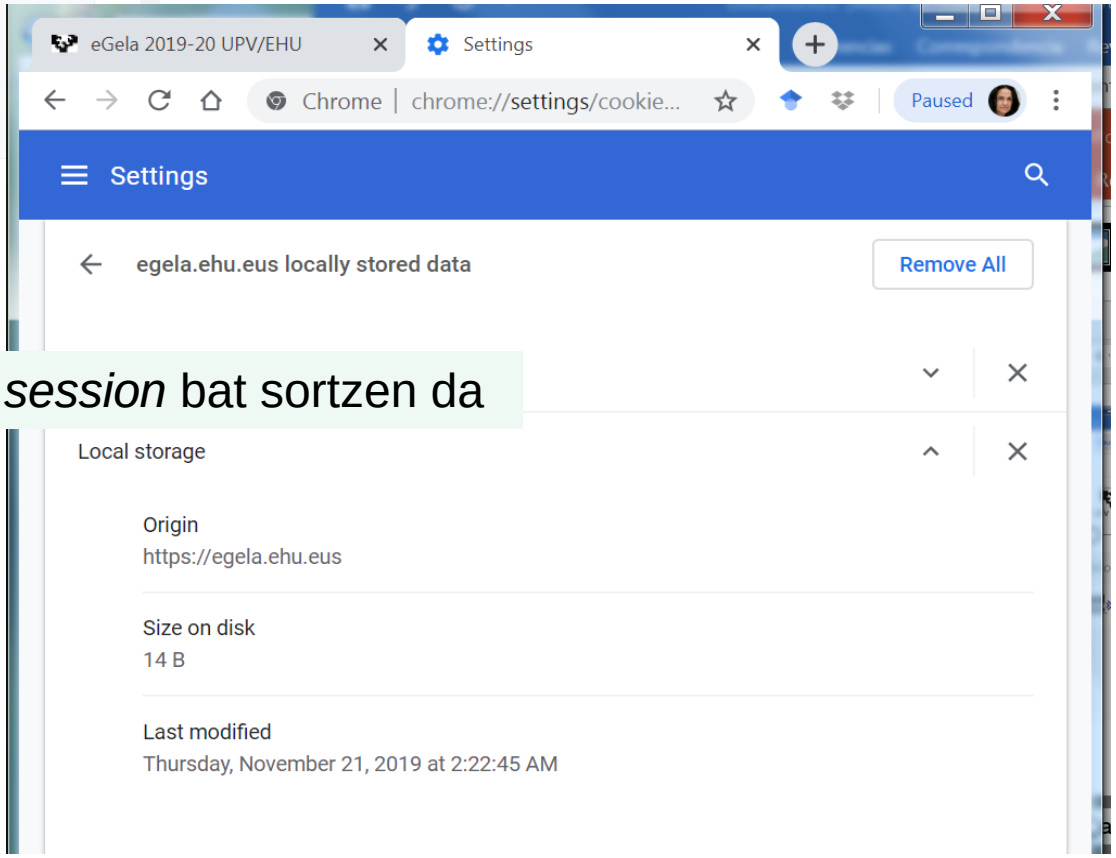
[Firefox]()



Nabigatzailea zabaldu hala, oraindik cookie-rik ez izatea gerta liteke (Edo aurreko konexioren bateko datuak izan ditzake)

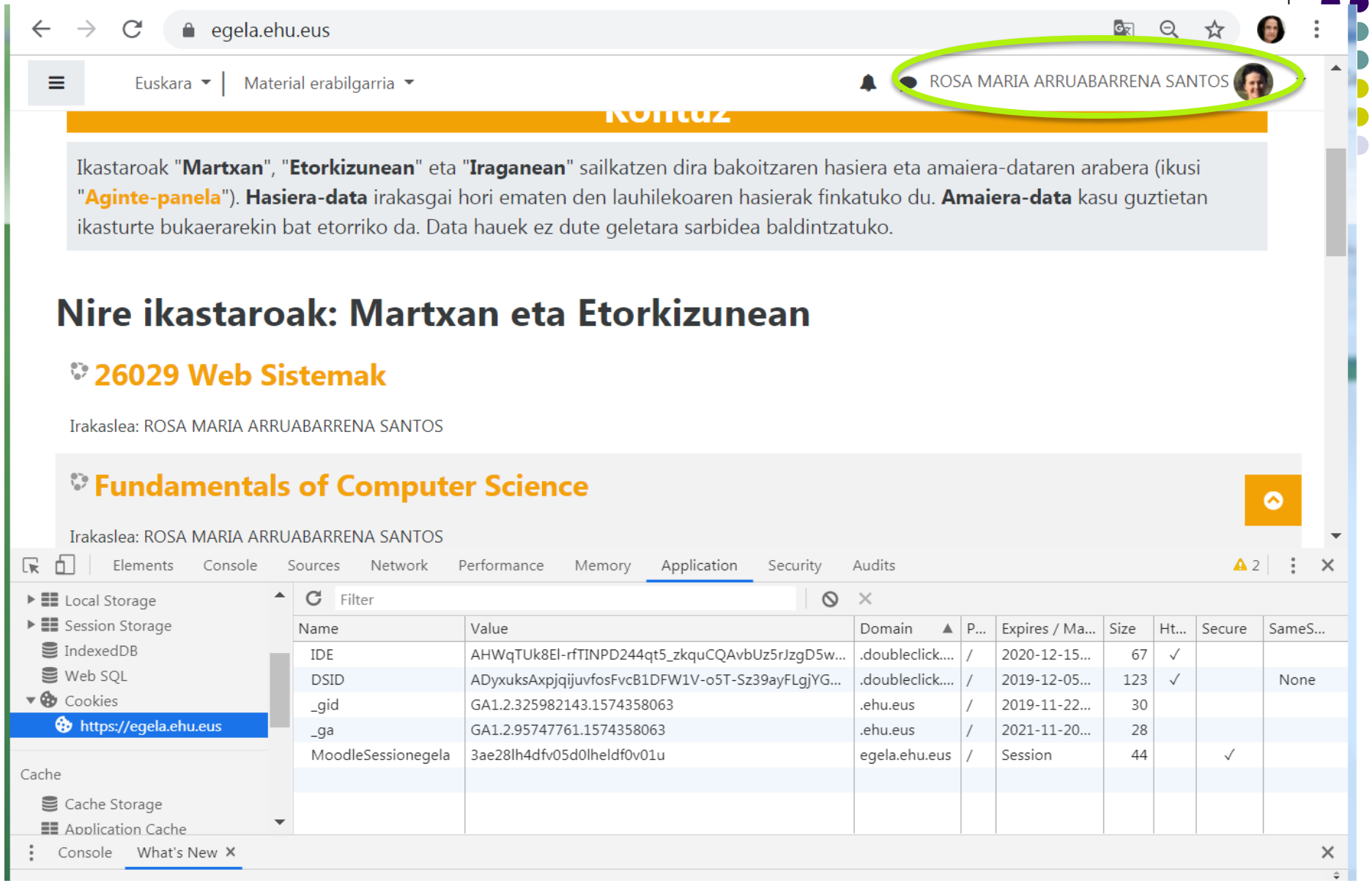






Egelara konektatzean (login egin gabe ere), *session* bat sortzen da

Nola kudeatzen dira Sesioak?



Egelan logina egitean cookiaren edukia alda liteke

Login / Logout. Adi !!!

- Sesio (edo saio) bat aktibo izateak ez du esan-nahi kautotzerik/ autentikaziorik egin denik
- Normalki, sesioa web-aplikaziora konektatzerakoan sortzen da
- Sesioaren IDa cookie bat balitz bezala igortzen zaio nabigatzaileari, lehenengo HTTP Responsean
- Logineko aplikazioak erabiltzailearen informazioa jartzen du zerbitzariaren *session* objektuan
- Logout aplikazioak erabiltzailearen *session*-etik datuak ezabatzen ditu

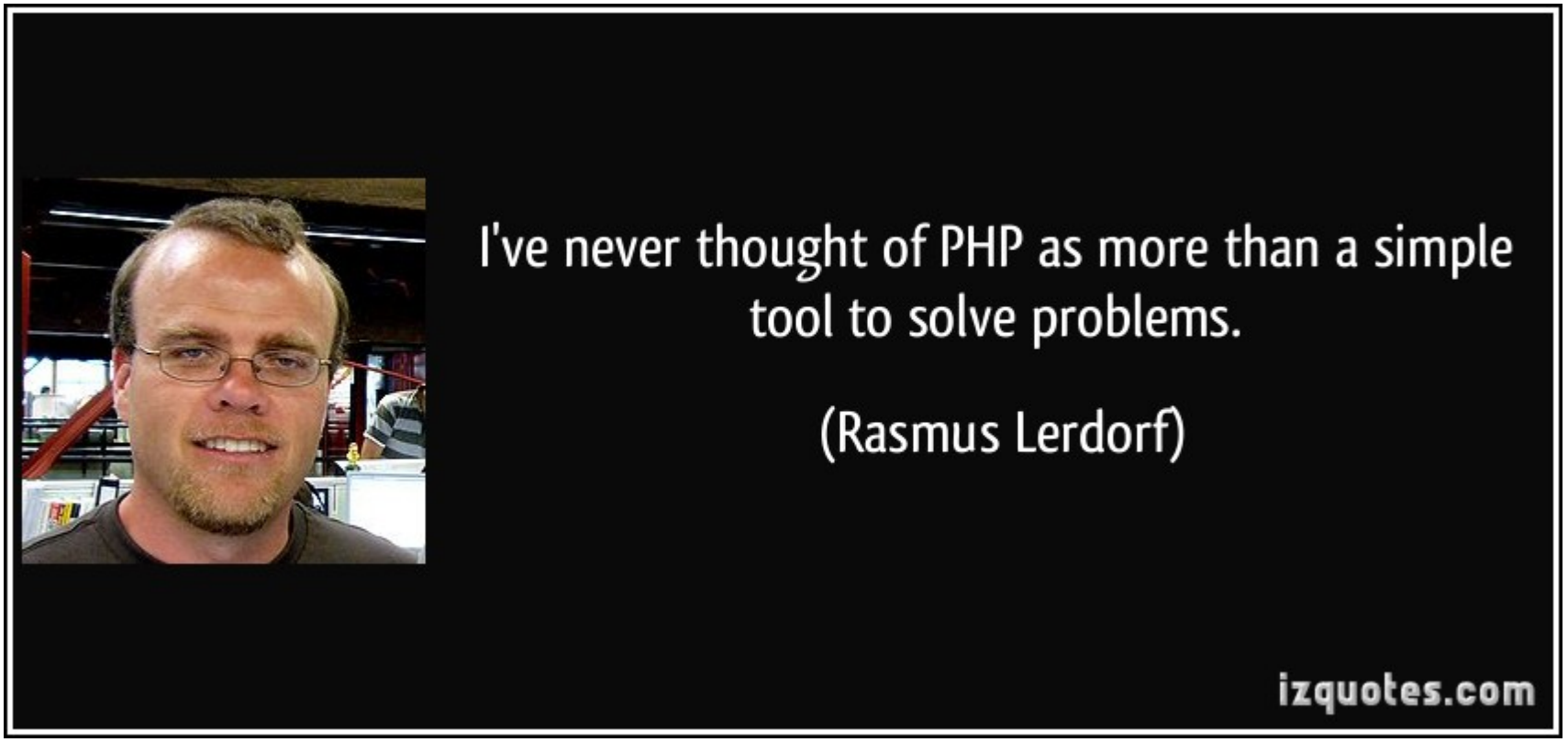
Laburpena

- Nabigatzaileek, cookiei esker, zerbitzari-aplikazio (web zerbitzu) bateko bezero desberdinen informazioa propioa meta ditzatela ahalbidetzen dute

- Sesioaren identifikatzailea, normalki, karaktere anitzeko katea da, cookie batean jasotzen da; eta aplikazioaren erabiltzaile bakoitzari zein sesio dagokion jakiteko erabiltzen da

- Zerbitzariak session objektua gauza txikien lan-espazio global bat balitz bezala erabiltzen du; bestela, aplikazioaren orri berri baten eskaera bakoitzarekin galduko litzateke eta.

Nola inplementatzen dira kontzeptu hauek PHPn?



Sesioen tratamendua

- Besterik esan ezean, PHP aplikazio batek sesio bat sortzen duenean bere identifikadorea cookie batean metatzen du eta sesioa sortu duen orriaren HTTP Response bidez igortzen/hedatzen da,

- Eta nabigatzaileak ez balitu cookie-ak onartzen?

  Irtenbideak egon badaude:

  - Sesioaren ID url-ean itsasten da (GET). Nahi izanez gero, PHP-ek metodo hori inplementa lezake
  - SID-a *input hidden* batean idazten da eta POST bidez pasatu (programatzaileak egingo luke hori)

  Baina aurreikusi behar dira, eta programatu; eta egun, gehienetan, eskaera amaitutzat ematen da.

PHP

Sesioaren ID-a

- **`SID`**: konstante bat da “sesio_izena=sesio_identifikatzailea”
- **`Session_id():`** sesioaren identifikadorea itzultzen du

PHP.ini-ren konfigurazio aukerak

- **`session.use_cookies = 1 / 0`**
  Bezeroari cookie-ak igorri ahal izatea Gaitzen/Desgaitzen du
- **`session.use_trans_sid = 1 / 0`**
  Eskaeraren URLean SID-a gordetzea era automatikoan Gaitzen/Desgaitzen du
- **`session.use_only_cookies = 1 / 0`**
  Gaitzen bada (1), SID-a soilik cookien bidez iraunarazi ahal izango da.

PHP funtzioak, sesioen maneiurako

- ***session_start ():***
  - Sesio bat hasieratzen du eta identifikatzaile bakarra esleitzen dio. Sesioa jada hasieraturik balego, sesioko aldagai guztiak kargatuko lituzke
- ***$_SESSION['izena'] = balioa;***
  - Session objektuko 'izena'-dun aldagaiari 'balioa' esleitzen
- ***unset ($_SESSION['izena']);***
  - Sesioko 'izena' aldagaia ezabatzen du
- ***if (isset($_SESSION['izena']))***
  - Aldagai bat erregistraturik dagoen egiaztazen du. TRUE itzultzen du baiezko kasuan eta FALSE bestela.
- ***session_destroy()***: sesioa isten du (errekuperragarria izanik)

https://www.php.net/manual/es/function.session-destroy.php

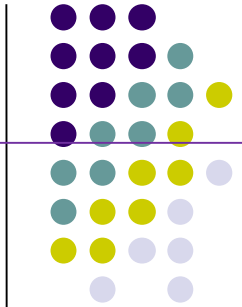
Sesioen maneiua

- Orri guztiek ***`session_start()`***-i dei bat egin behar diote sesioko aldagaiak kargatzeko. Sesioak existituko ez balu, sortu egiten da

- Deiak **edozein HTML kode aurretik** egon behar du

- *Logout* egiterakoan, ***`session_destroy()`***-i dei egitea komeni da.

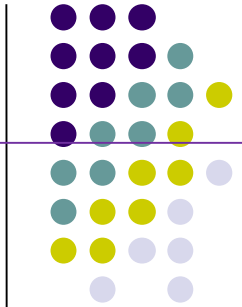
adibidea1a.php



```
<?PHP    session_start ();  ?>
<HTML LANG="es">
<HEAD> <TITLE>Sesioen maneiua</TITLE>
 <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>

<BODY>
 <H1>Sesioen maneiua </H1>
 <H2>Pausoa 1: sesio aldagaia sortu eta gorde egiten da</H2>

<?PHP
   $var = "Miren";
   $_SESSION['var'] = $var;
   print ("<P>Sesioko aldagaiaren balioa: $var</P>\n");
?>
 <A HREF="adibidea1b.php">Pausoa 2</A>.
</BODY>
</HTML>
```

http://www.lsi.us.es/cursos/cursophp/ejercicios/tema5/ejercicio1.php

adibidea1b.php

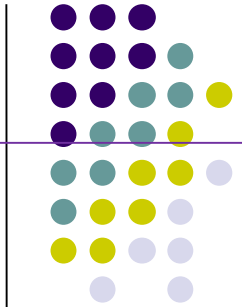


```
<?PHP    session_start ();  ?>
<HTML LANG="es">
<HEAD> <TITLE>Sesioen maneiua</TITLE>
 <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>

<BODY>
 <H1>Sesioen maneiua<H1>
 <H2>Pausoa 2: metatutako sesioko aldagaia eskuratu eta desegiten du</H2>
<?PHP
 $var = $_SESSION['var'];
 print ("<P>Sesioko aldagaiaren balioa: $var</P>\n");
 unset ($_SESSION['var']);
?>
 <A HREF="adibidea1c.php">Pausoa 3</A>.
</BODY>
</HTML>
```

adibidea1c.php

```
<?PHP    session_start ();  ?>
<HTML LANG="es">
<HEAD> < TITLE>Sesioen maneiua</TITLE>
 <LINK REL="stylesheet" TYPE="text/css" HREF="estilo.css">
</HEAD>

<BODY>
 <H1>Sesioen maneiua </H1>
 <H2>Puasoa 3: aldagaia desegina izan da eta bere balio galdu da</H2>

<?PHP
 $var = $_SESSION['var'];
 print ("<P> Sesioko aldagaiaren balioa: $var</P>\n");
 session_destroy();
?>
 <A HREF="adibidea1a.php">"Pausoa 1"-ra itzuli</A>.
</BODY>
</HTML>
```

Adibidea (2)

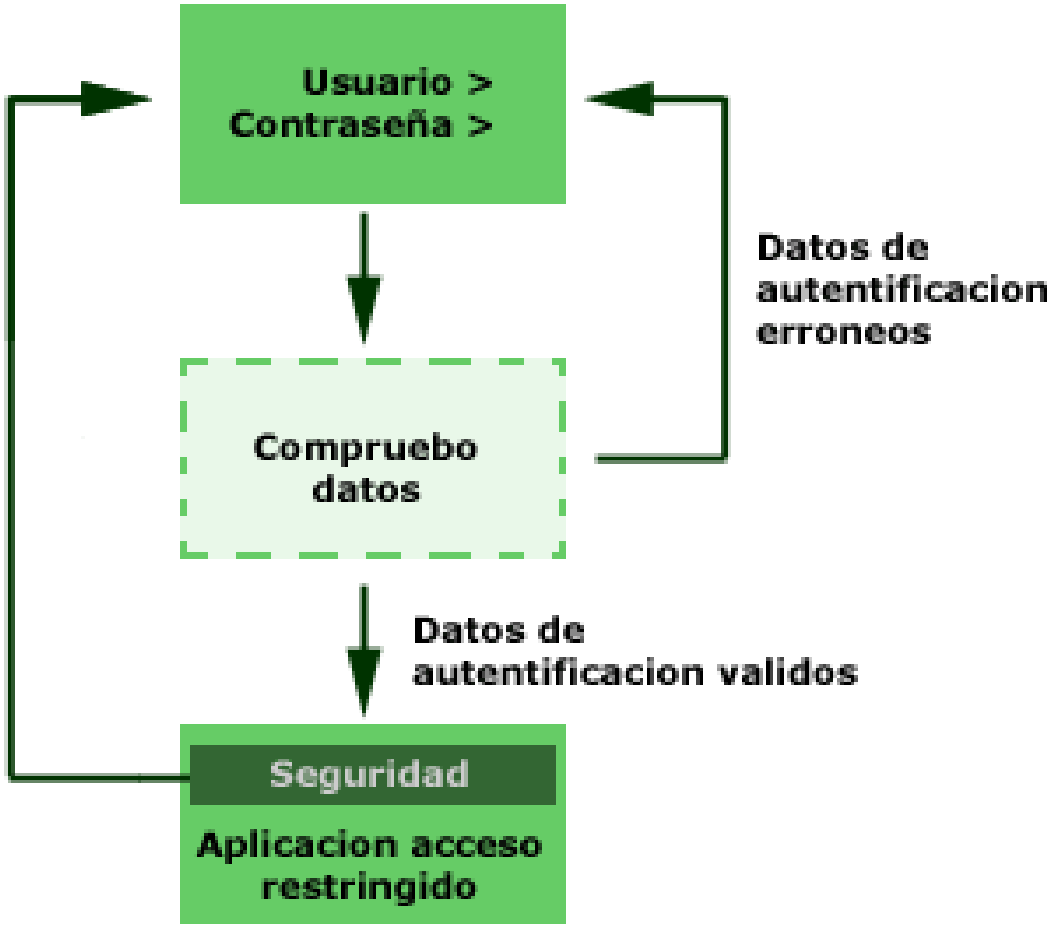


```
<?php
ini_set('session.cookie_lifetime',60);
echo ‘After 60 minute the session will be finished
   '.ini_get("session.cookie_lifetime")/60 . ' minute/s';
session_start();
if (empty($_SESSION['count'])) { $_SESSION['count'] = 1;
} else { $_SESSION['count']++;}
?>
<p>
Hello visitor, you have seen this page
   <?php echo $_SESSION['count'];?> times.
</p>
<p> Session number: <?php echo '<b>'. session_id() .'</b>'; ?> <p>
To continue,
<a href=“sesionexample.php?<?php echo htmlspecialchars(SID);
   ?>">click here</a>.
</p>
```

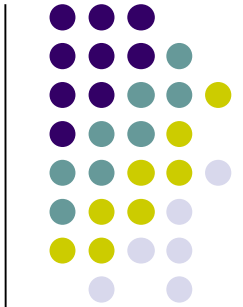
Session-etan oinarritutako kautotzea,

(nabigazio murriztua/baldintzatua web-guneko orrietan)

http://www.desarrolloweb.com/articulos/1007.php



Index.php (hasierako formularioa)

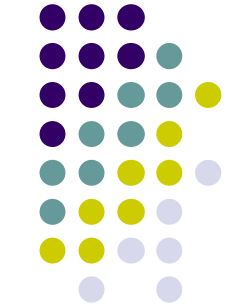


```
<html>
   ……
 <form action="kontrola.php" method="POST">
   ….
      <?php if ($_GET["erabiltzaileerrorea"]=="bai"){?>
   bgcolor=red><span style="color:ffffff"><b>Datu okerrak</b> </span>
   <?php }
      else
       {?>
   bgcolor=#cccccc>Sakatu zure atzipen gakoa
   <?php }?>
    USER:
    <input type="text" name="erabiltzailea" size="8" maxlength="50">

    PASSWD:</td>
    <input type="password" name="pasahitza" size="8" maxlength="50">

    <input type="Submit" value="SARTU">
</form>
   ….
```

kontrola.php

```
<?php
  //Erabiltzailea eta pasahitza zuzenak diren aztertu
  if ($_POST["erabiltzailea"]=="mikel" &&
  $_POST["pasahitza"]=="qwerty"){
      // Erabiltzailea eta pasahitza baliozkoak
      // sesio bat definitzen dut eta datuak metatzen ditut
      session_start();
      $_SESSION["kautotua"]= "BAI";
      header ("Location: aplikazioa.php");
  }else {
      // existitzen ez bada, berriro atarira bidaltzen dut
      header("Location: index.php?erabiltzaileerrorea=bai");
  }

?>
```

segurtasuna.php



```php
<?php
  //sesio hasiera
  session_start();

  // ERABILTZAILEA KAUTOTURIK DAGOELA EGIAZTATU
  if ($_SESSION["kautotua"] != "BAI") {
    // existitzen ez bada, berriro kautotzera bidaltzen dut
    header("Location: index.php");
    //gainera, script-atik irtetzen gara
    exit();
  }

?>
```

Aplikazioaren fitxategietan

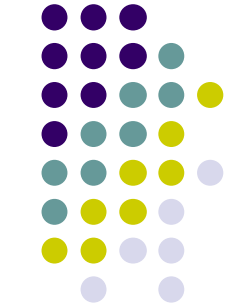
```
<?php include ("segurtasuna.php");?>
  <html>
  <head>
  <title>Aplikazio segurua</title>
  </head>
  <body>
  <h1>Hemen bazaude, kautotu zarelako da</h1>
  <br>
  ----
  <br>
  Aplikazio segurua: atzipen murriztuko ingurunea
  <br>
  ----
  <br>
  <br>
  <a href="irten.php">Irten</a>
  </body>
  </html>
```

irten.php

```
<?php
   session_start();
   session_destroy();

?>
   <html>
   <head>
       <title> Irten zara!!</title>
   </head>
   <body>
       Mila esker zure atzipenagatik
   <br>
   <br>
   <a href="index.php">Kautotze formulariora</a>
   </body>
   </html>
```

http://www.desarrolloweb.com/articulos/ejemplos/autentificacion-php/

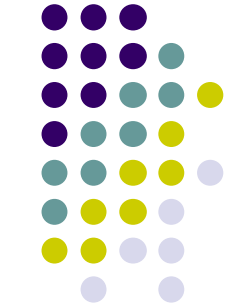
Web aplikazioen garapenean: Neurriak



- Zentzuz jokatu
- Inputetatatik jasotako datuak: Inoiz ez fidatu zuzenean erabiltzailetik datozen datuetaz, beti aztertu:
  - patroi zorrotzetara doitu (bezero, zerbitzari alde bietan)
  - karaktere bereziak izan litezkeenak ezabatu (‘ -- ; ,…)

  mysqli_real_escape_string, addslashes, stripslashes,htmlspecialchars, …
  - PDO eta MySQLi paketeek erremintak dituzte. Adib: “PDO” bidezko sql galderen parametroen “serializazioa”
- Pasahitzak critpografiatu: crypt, sha2, …
- Adi bereziki scriptetan eraikitzen diren galderen parametroekin
- Adi kanpoko fitxategien kargarekin

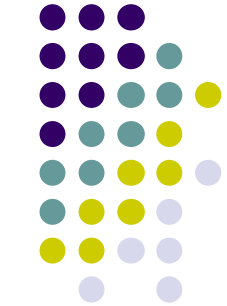
Web aplikazioen garapenean: erasoen arriskuak -> ondorioak



- Cookien faltsifikazioa → online erosketen fraudea
- Cross-site cooking → sesio lapurreta
- Query string → URL modifikazioa, ezeztatutako atzipenak
- SQL injection / blind SQLi injection → scripting, spying
- Kanpoko fitxategien karga zuzena -> scripting

- https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL
- https://es.wikipedia.org/wiki/Cookie_(inform%C3%A1tica)
- Google: prevent sql injection php + PDO + mysqli
- OWASP: https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project

- **Cookien faltsifikazioa**: erasotzaile batek, zerbitzarirako bidean cookie-a harrapatzen eta modifikatzen duenean. Adibidez, online erosketaren salneurria aldatu (txikitu)



- ***cross-site cooking*: web-gune desberdinen arteko cookieak.** Gune bakoitzak bere cookiak izan behar ditu, eta *gune.erasotzailea*k ez luke aukerarik izan behar beste gune bateko cookieak atzitzeko/aldatzeko edo definitzeko.Cookien faltsifikazioaren antzeko da, baina kasu honetan erasotzaileak, intentzio txarrik ez duten erabiltzaileak dabiltzan nabigatzaileen ahulezitaz baliatzen dira *gune.ona* zuzenean erasotzeko. Eraso hauen helburua, adibidez, sesio lapurreta izan liteke

- **query string** URL katean informazio txertatzen denean.

- **SQL txertaketa (SQL injection)** kode arrotza txertatzeko modu bat da. Ahuldutasuna programako aldagaien azterketen edo iragazki okerrean dago, aldagai horiek hain zuzen SQL sortzen duten edo duten programan erabiltzen direlarik. Txertaketa kalte egiteko edo espiatzeko egiten da eta hortaz, segurtasun informatikoaren problema da. Ondorioz, programatzaileak neurriak hartu behar ditu diseinatzeko/programazio garaian behar diren neurriak hartuz

  https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL

- **Blind SQL injection (SQL txertaketa itsua)**. Beti egiazkoa den baldintza bat gehitzen dio eraikitzen ari garen sql galderan (Adibidez: " Or 1=1" edo "having 1=1". Horrelako eraso mota bidez lor litezke datu bateetako erabiltzaileen pasahitzak eta superuser kontua, besteak beste.