APLIKAZIO MAILA: HTTP HYPERTEXT TRANSFER PROTOCOL

Konputagailu Sareen Oinarriak

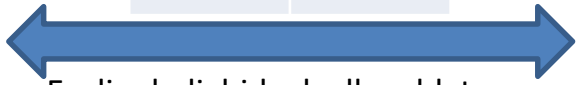
17. ASTEA (2018/01/29)

REST ARKITEKTURA

Protokoloa: HTTP
Ekintzak: CRUD

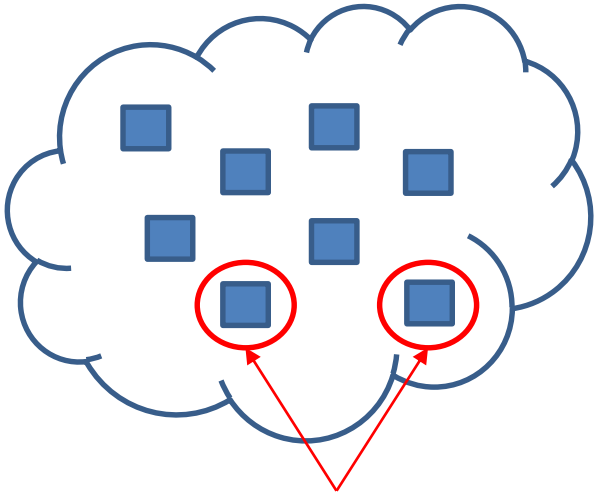
Create	POST
Read	GET
Update	PUT
Delete	DELETE

BEZEROA



Ez dira baliabideak elkaraldatzen, baliabideen adierazpenak baizik

Ez dago egoerarik, hau da, ekintzak ez daude erlazionaturik → Cookie



Baliabideak, hauen izendatzailea URI deitzen delarik.

http://api.dropboxapi.com/2/files/upload
http://api.thingspeak.com/channels/376999/feeds.json

HTTP SARRERA

- RFC (Request For Comments) dokumentua aditu batzuek IETF (Internet Engineering Task Force) elkarteari bidaltzen dioten zirkularra da, elkarrekikotasunean eztabaidatua eta adostua izan daiten, Internet-en inguruko estandarrak garatzea helburu duelarik.

- Jatorriz, HTTP v1.1 (HTTP/1.1) RFC 2616-an definitu zen.
  Gaur egun, HTTP/1.1-en zehaztapena ondoko RFC-etan biltzen da:
  - RFC 7230: HTTP/1.1 Message Syntax and Routing
  - RFC 7231: HTTP/1.1 Semantics and Content
  - RFC 7232: HTTP/1.1 Conditional Requests
  - RFC 7233: HTTP/1.1 Range Requests
  - RFC 7234: HTTP/1.1 Caching
  - RFC 7235: HTTP/1.1 Authentication

  Gai honetan, RFC hauetan eta hauekin erlazionatuta dauden beste RFC batzutan bilduta dauden hainbat alderdi ikasiko ditugu.

- 2015ko maiatzean RFC 7540 argitaratu zen: HTTP/2.
  - Bertsio berri hau gaur egungo web orri berrien karga bizkortzea du helburu, zeintzuk:
    - irudi, javascript eta CSS kopuru handi batez osoturik daude.
    - AJAX bitartez eskaera asinkronoak burutzen dituzte.
  - HTTP/2 Google-n SPDY protokoloan oinarrituta dago.
  - **Mezuek HTTP/1.1-ek duten semantika eta sintaxi bera jarraitzen dute. HTTP/1.1-ekiko aldatzen den alderdi bakarra mezuak saretik ("on the wire") bidaltzean erabiltzen den formatua da.**

- IETF-aren HTTP protokoloari buruzko lan taldearen webgunea: http://httpwg.github.io/

- Web Sistemekin erlazionatutak dauden beste teknologia batzuk arautzeko beste talde edo erakunde batzuk daude; adibidez, W3C (World Wide Web Consortium) elkartea HTML, CSS eta DOM estandarrak kudeatzeaz arduratzen da.

HTTP SARRERA

- Jatorriz, HTTP hipertestu motako baliabideen transferentziarako diseinatutako aplikazio mailako protokoloa da.
  - *Aplikazio mailako protokoloa:* HTTP-ek aplikazioei (adibidez, nabigatzaile edo web zerbitzari bati) datuen bidaltze eta jasotzea egiteko zuzeneko euskarria emoten die.
  - *Transferentzia:* HTTP-ek eskaera-erantzun eredu bat jarraitzen duen transakzio eskema darabil: bezero aplikazioak eskaerak egiten ditu eta zerbitzari aplikazioak eskaera horiei erantzuten die.
  - *Baliabidea:* HTTP-ek URI (Universal baliabidea Identifier) izeneko identifikadore ereduaren bitartez baliabideak izendatzen ditu.
  - *Hipertestua:* beste informazio bat lortzeko estekak dituen testua.
    - Adibidez: web orria (--> wikia --> Wikipedia).
    - Web orri baten kasuan, hipertestua adierazteko lengoaia HTML da.

BALIABIDEEN IZENDAPENA: URI

- URI eta URL terminoei buruzko argibideak.
  - **URI-a (Universal Resource Identifier)** Internet-en baliabide bat izendatzeko aukera ematen duen US-ASCII katea da. RFC 3986, 3. atala-ren arabera, bere sintaxia ondorengoa da:

    URI = scheme ":" "//" authority ["/" path] ["?" query] ["#" fragment]

  - **URL-a (Universal Resource Locator)** baliabide bat identifikatzeaz aparte, baliabide hori Internet-en aurkitzeko balio duen URI-a da. Adibidez:

    https://egela.ehu.eus/course/view.php?id=3032 (eGela-ko ikasgai baten web orria)

    URL-aren irakurketa: *egela.ehu.eus* zerbitzarian *HTTPS* protokoloa erabilita eskuratu daitekeen baliabidea dago, baliabide horren bide osoa */course/view.php* delarik. Baliabide honen bitartez, eGela web zerbitzariak ikasgai baten web orria sortu dezake (hots, ikasgaiaren web adierazpena), ikasgaiaren edukiei egokitua. Horretarako, *id* izeneko parametroa pasatu beharra dago, parametro honen balioak web orria sortzeko datu basetik irakurri beharreko ikasgaiaren gako nagusia adierazten duelarik.

- URI terminoa erabiltzea gomendatzen da. Hau da, errekurtso bat bere URI-a erabilita identifikatzen da. Eskemaren arabera (“scheme”), URI-a baliabidea aurkitzeko erabili ahal izango da (adibidez, *http* eskemaren kasuan).

HELBURUAK

TEORIA

- HTTP protokoloaren funtzionamendua deskribatu.
  - **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - Zelan egiten dira berhelbideraketak? (3xx erantzun kodeak)
  - Zelan kargatzen da web orri bat nabigatzailean?
- Galdera horiek erantzuteko, adibideen bitartez HTTP protokoloaren semantika eta sintaxia deskribatuko dira:
  - Eskaera eta erantzunaren egitura.
  - Protokoloaren funtzionamendua inplementatzeko metodoak eta goiburuak.

PRAKTIKA

- Python-en HTTP bezeroak programatu:
  - Google-era konektatu.

HTTP-REN FUNTZIONAMENDUA

Jarraian, HTTP protokoloaren funtzionamendua deskribatuko da adibide baten bitartez. Ondorengo galderak erantzungo dira, hain zuzen ere:

- **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - Zer egiten du nabigatzaileak?
  - Zein formatu (sintaxi eta semantika) dauka eskaerak?
  - Zelan prozesatzen da eskaera zerbitzarian?
  - Zein formatu (sintaxi eta semantika) dauka erantzunak?

HTTP-REN FUNTZIONAMENDUA:

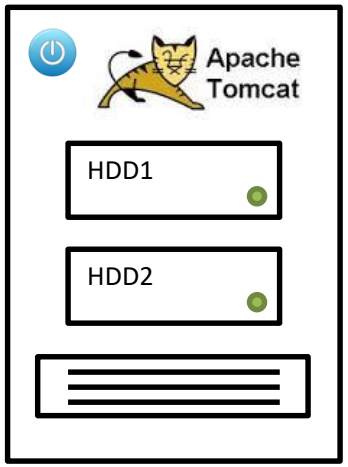
PARTE HARTZEN DUTEN ENTITATEAK

BEZEROA

Bezeroari buruzko oharrak:

- Firefox aplikazioa bezero motako aplikazioa da, HTTP protokoloa inplementatzen duena.
- Firefox-ek gzip formatuan konprimatutako edukia dekodifikatu dezake.
- Bezeroaren sistema eragileak TCP/IP protokolo multzoa darabil.
- Sistema eragileak DNS zerbitzari baten helbidea konfiguratuta dauka.

ZERBITZARIA



Zerbitzariari buruzko suposaketak:

- Zerbitzariaren alias-a **kso2018.com** da.
- Tomcat aplikazioa zerbitzari motako aplikazioa da, HTTP protokoloa inplementatzen duena.
- Tomcat 8080. portuan entzuten dago.
- Tomcat-ek http://kso2018.com:8080/baliabidea URI-arekin identifikatutako baliabidea eskeintzen du.
- Baliabide hori testu lauean eta HTML-n, euskeraz eta gazteleraz, mahaigaineko eta mugikorrentzako bertsioetan, eskuragarri dago.
- Tomcat-ek ezin dezake testua konprimatu.
- Zerbitzariaren sistema eragileak TCP/IP protokolo multzoa darabil.

HTTP-REN FUNTZIONAMENDUA

Jarraian, HTTP protokoloaren funtzionamendua deskribatuko da adibide baten bitartez. Ondorengo galderak erantzungo dira, hain zuzen ere:

- **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - **Zer egiten du nabigatzaileak?**
  - Zein formatu (sintaxi eta semantika) dauka eskaerak?
  - Zelan prozesatzen da eskaera zerbitzarian?
  - Zein formatu (sintaxi eta semantika) dauka erantzunak?

HTTP-REN FUNTZIONAMENDUA:

TCP KONEXIOAREN EZARPENA

BEZEROA

Bezeroari buruzko oharrak:

- Firefox aplikazioa bezero motako aplikazioa da, HTTP protokoloa inplementatzen duena.
- Firefox-ek gzip formatuan konprimatutako edukia dekodifikatu dezake.
- Bezeroaren sistema eragileak TCP/IP protokolo multzoa darabil.
- Sistema eragileak DNS zerbitzari baten helbidea konfiguratuta dauka.

Erabiltzaileak URI-a nabigatzailearen helbide-barran sartzen duenean, nabigatzaileak bere zati ezberdinak aztertzen ditu:

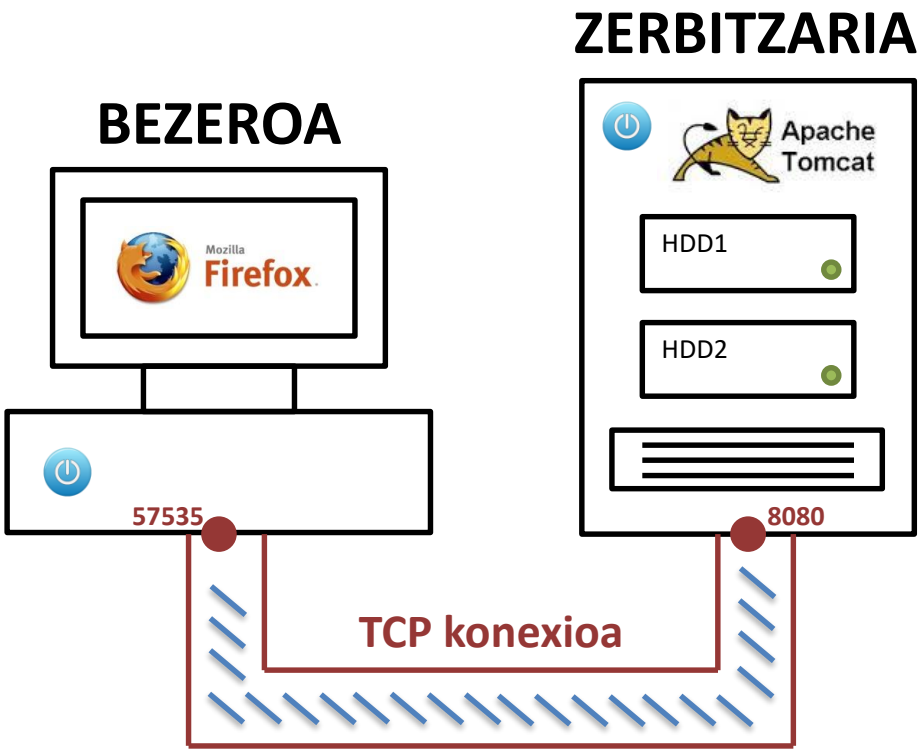
http://kso2018.com:8080/baliabidea
Scheme: http
Authority: Host: kso2018.com
Port: 8080
Path: /baliabidea

Nabigatzaileak sistema eragileari zerbitzariaren host izenaren ebazpena eskatzen dio. Sistema eragileak eskaera hau DNS zerbitzariaren bitartez ebatzi eta nabigatzaileari IP helbidea itzultzen dio.

Datu honekin, nabigatzaileak sistema eragileari TCP konexio bat (SYN, SYN-ACK eta ACK) sortzeko eskatzen dio, portu lokal batetik zerbitzariaren 8080. portura.

HTTP-REN FUNTZIONAMENDUA:

TCP KONEXIOAREN EZARPENA



JARDUERA:
PYTHON-EN TCP KONEXIO BAT EZARRI

- *httplib* liburutegia erabiliz, www.google.com zerbitzariaren 80. portuarekin TCP konexio bat ezartzen duen Python script-a programatu ezazu.

JARDUERA:

PYTHON-EN TCP KONEXIO BAT EZARRI

```
# -*- coding: UTF-8 -*-

import httplib # HTTP protokoloa inplementatzen duen liburutegia

print "\r\n---> TCP konexioa ezartzen... "
zerbitzaria = 'www.google.com'
conn = httplib.HTTPConnection(zerbitzaria) # TCP konexioa definitu
conn.connect() # TCP konexioa ezarri
print "---> TCP konexioa ezarrita!!!"

socketa = conn.sock.getsockname() # socketa IP_helbide-TCP_portu
                                  # bikoteaz osoturik dago
print "     Local IP address is " + str(socketa[0])
print "     Local TCP port is " + str(socketa[1])
```

KODEA

```
---> TCP konexioa ezartzen...
---> TCP konexioa ezarrita!!!
     Local IP address is 158.227.70.162
     Local TCP port is 57535
```

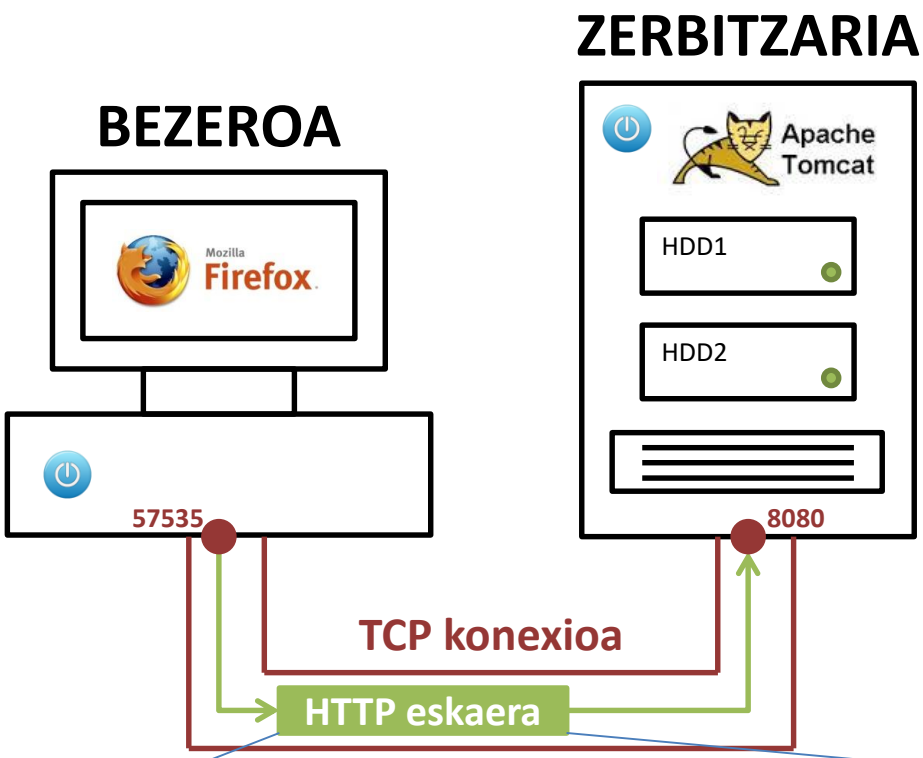
STDOUT

HTTP-REN FUNTZIONAMENDUA

Jarraian, HTTP protokoloaren funtzionamendua deskribatuko da adibide baten bitartez. Ondorengo galderak erantzungo dira, hain zuzen ere:

- **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - Zer egiten du nabigatzaileak?
  - **Zein formatu (sintaxi eta semantika) dauka eskaerak?**
  - Zelan prozesatzen da eskaera zerbitzarian?
  - Zein formatu (sintaxi eta semantika) dauka erantzunak?

HTTP-REN FUNTZIONAMENDUA: BEZEROAREN ESKAERA



HTTP eskaeraren sintaxia

Metodoa URIa HTTP/1.1
Goiburuak
CRLF*
Mezuaren gorputza (zortzikoteetan **)

* CRLF = Carriage Return + Line Feed = \r\n = 0x0D 0x0A
** zortzikote = 8 bit-eko sekuentzia

Adibidearen HTTP eskaera

GET /baliabidea HTTP/1.1
Host: kso2018.com:8080
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Desktop

GET /baliabidea HTTP/1.1\r\nHost: kso2018.com:8080\r\nAccept: text/html\r\nAccept-Encoding: gzip,identity;q=0.5\r\nAccept-Language: en-US,es-ES;q=0.8\r\nUser-Agent: Mozilla Windows Desktop\r\n\r\n

HTTP-REN FUNTZIONAMENDUA: BEZEROAREN ESKAERA

HTTP eskaeraren sintaxia

Metodoa URIa HTTP/1.1
Goiburuak
CRLF
Mezuaren gorputza

Adibidearen HTTP eskaera

GET /baliabidea HTTP/1.1
Host: kso2018.com:8080
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Desktop

Metodoa: GET

Metodoak baliabidearen gainean zein CRUD (Create, Read, Update and Delete) ekintza mota burutu nahi den adierazten du. Kasu honetan, GET → irakurketa.

URIa: /baliabidea

Baliabidearen identifikazioa URI osoarekin edo URI erlatiboarekin egin daiteke.

GET http://kso2018.com:8080/baliabidea HTTP/1.1

GET /baliabidea HTTP/1.1
Host: kso2018.com:8080

Goiburuek bezeroaren ezaugarriak eta erantzunarekiko lehentasunak adierazten dituzte.

Accept: eduki bezela HTML onartzen dela adierazten da.
Accept-Encoding: nabigatzailearentzako eduki konprimatuak (gzip formatuan) lehentasuna dauka, baina konprimatu gabeko edukia (identity) ere onartzen du.
Accept-Language: nabigatzaileak bere hizkuntz nagusia *en* dela adierazten du, bigarren aukera *es* delarik.
User-Agent: nabigatzaileak bere burua mahaigaineko Windows batean dagoen Mozilla bezela aurkezten du.

Mezuaren gorputza: kasu honetan hutsik dago.

JARDUERA:
PYTHON-EN HTTP ESKAERA BAT BIDALI

- *httplib* liburutegia erabiliz, www.google.com zerbitzariaren 80. portuan entzuten dagoen web aplikazioari **/** baliabidea eskatzen dion Python script-a programatu ezazu.

JARDUERA:

PYTHON-EN HTTP ESKAERA BAT BIDALI

```
# -*- coding: UTF-8 -*-

import httplib # HTTP protokoloa inplementatzen duen liburutegia

print "\r\n---> TCP konexioa ezartzen... "
zerbitzaria = 'www.google.com'
conn = httplib.HTTPConnection(zerbitzaria) # TCP konexioa definitu
conn.connect() # TCP konexioa ezarri
print "---> TCP konexioa ezarrita!!!"

socketa = conn.sock.getsockname() # socketa IP_helbide-TCP_portu
                                  # bikoteaz osoturik dago
print "     Local IP address is " + str(socketa[0])
print "     Local TCP port is " + str(socketa[1])

print "---> HTTP eskaera: metodoa, URIa, goiburuak, edukia"
metodoa = 'GET'
uria= '/'
eskaeraren_goiburuak = {'Host': zerbitzaria,
                        'User-Agent': 'Nire Python bezeroa',}
eskaeraren_edukia = ''
conn.request(metodoa, uria, headers=eskaeraren_goiburuak, body=eskaeraren_edukia)
print "---> HTTP eskaera bidali da”

conn.close()
```

HTTP-REN FUNTZIONAMENDUA

Jarraian, HTTP protokoloaren funtzionamendua deskribatuko da adibide baten bitartez. Ondorengo galderak erantzungo dira, hain zuzen ere:

- **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - Zer egiten du nabigatzaileak?
  - Zein formatu (sintaxi eta semantika) dauka eskaerak?
  - **Zelan prozesatzen da eskaera zerbitzarian?**
  - Zein formatu (sintaxi eta semantika) dauka erantzunak?

HTTP-REN FUNTZIONAMENDUA:

ESKAERAREN PROZESAKETA ZERBITZARIAN

Zerbitzariak eskaera jasotzen duenean, metodoa eta URIa aztertzen ditu ondorengoa jakiteko:

1. Baliabidea existitzen ote den.
2. Eskatutako ekintza baliabideari aplikatu ahal zaion.

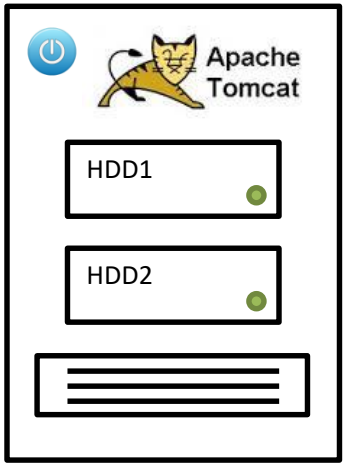
Edukiaren negoziaketa: baliabidea existitzen bada eta eskatutako ekintza aplikatu ahal bazaio, web zerbitzariak eskaeraren goiburuak aztertzen ditu bezeroaren beharrei hobetoen doakion baliabidearen bertsioa itzultzeko:

- Accept: text/html
- Accept-Encoding: gzip,identity;q=0.5
- Accept-Language: en-US,es-ES;q=0.8
- User-Agent: Mozilla Windows Desktop

Kasu honetan, web zerbitzariak ondorengo ezaugarriak dituen erantzuna itzultzen du:

- HTML-en kodifikatua
- konpresiorik gabe
- gazteleraz
- bertsio klasikoa (mugikorrak ez diren gailuentzako)

ZERBITZARIA



Zerbitzariari buruzko suposaketak:

- Zerbitzariaren alias-a **kso2018.com** da.
- Tomcat aplikazioa zerbitzari motako aplikazioa da, HTTP protokoloa inplementatzen duena.
- Tomcat 8080. portuan entzuten dago.
- Tomcat-ek http://kso2018.com:8080/baliabidea URI-arekin identifikatutako baliabidea eskeintzen du.
- Baliabide hori testu lauean eta HTML-n, euskeraz eta gazteleraz, mahaigaineko eta mugikorrentzako bertsioetan, eskuragarri dago.
- Tomcat-ek ezin dezake testua konprimatu.
- Zerbitzariak TCP/IP protokolo multzoa darabil.

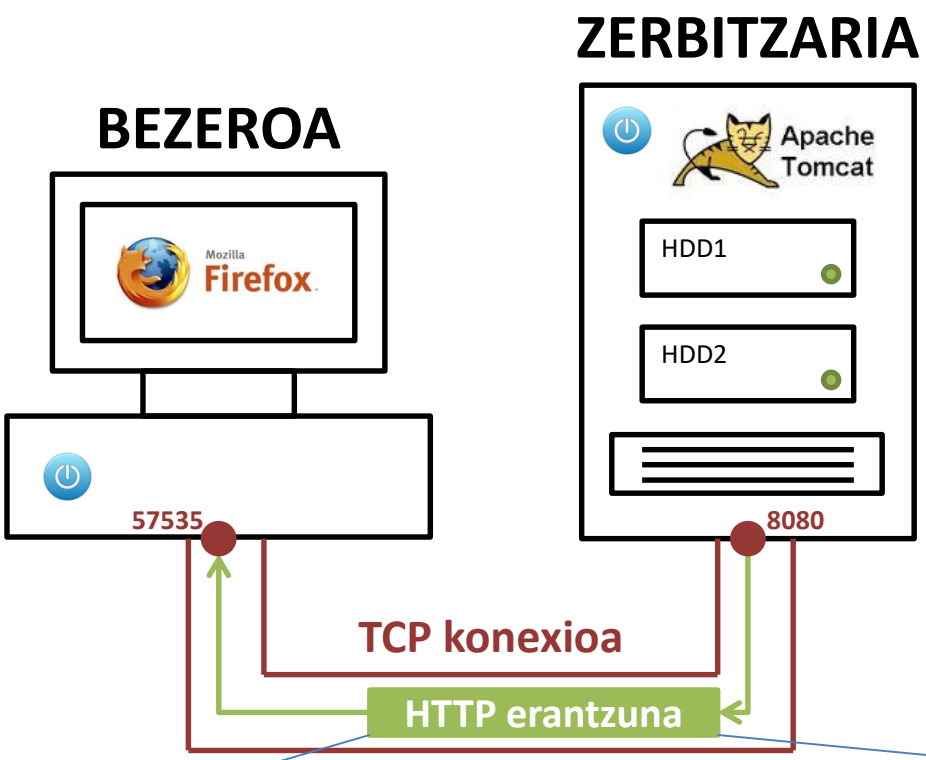
HTTP-REN FUNTZIONAMENDUA

Jarraian, HTTP protokoloaren funtzionamendua deskribatuko da adibide baten bitartez. Ondorengo galderak erantzungo dira, hain zuzen ere:

- **Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?**
  - Zer egiten du nabigatzaileak?
  - Zein formatu (sintaxi eta semantika) dauka eskaerak?
  - Zelan prozesatzen da eskaera zerbitzarian?
  - **Zein formatu (sintaxi eta semantika) dauka erantzunak?**

HTTP-REN FUNTZIONAMENDUA:

ZERBITZARIAREN ERANTZUNA



HTTP erantzunaren sintaxia

HTTP/1.1 Status Deskribapena
Goiburuak
CRLF
Mezuaren gorputza (zortzikoteetan)

Adibidearen HTTP erantzuna

HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>

HTTP/1.1 200 OK\r\nDate: Thu, 20 Mar 2014 20:25:52 GMT\r\nLast-Modified: Tue, 17 Sep 2013 13:00:02 GMT\r\nETag: "1a968-3ec-4e693e61bb8b6"\r\n
Content-Length: 76\r\nContent-Type: text/html; charset=ISO-8859-1\r\n\r\n<html><head><title>index.html</title></head><body>Hello World!</body></html>

HTTP-REN FUNTZIONAMENDUA:

ZERBITZARIAREN ERANTZUNA

HTTP erantzunaren sintaxia

```
HTTP/1.1 Status Deskribapena
Goiburuak
CRLF
Mezuaren gorputza (zortzikoteetan)
```

Adibidearen HTTP erantzuna

```
HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

Status: 200

Eskaera ulertzeko eta burutzeko ahalegina deskribatzen duen kodea.

200 kodeak eskaera ondo osoturik dagoela eta zuzen prozesatu dela adierazten du.

Programei zuzendua.

Deskribapena: OK

Status-ari loturiko testu deskribapena.

Gizakiei zuzendua.

Goiburuel erantzunaren alderdi ezberdinak bereizten dituzte.

Content-Length: zerbitzariak mezuaren gorputzeko luzeera (zortzikote kopurua) adieratzen du.

Content-Type: zerbitzariak edukia HTML motakoa dela eta bere zortzikoteak *latin-1*-en (ISO-8859-1) kodifikatuta daudela adierazten du.

HTTP-REN FUNTZIONAMENDUA:

ZERBITZARIAREN ERANTZUNA

HTTP erantzunaren sintaxia

HTTP/1.1 Status Deskribapena
Goiburuak
CRLF
Mezuaren gorputza (zortzikoteetan)

Adibidearen HTTP erantzuna

```
HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

Goiburuak: (jarraipena)

Date: zerbitzariak erantzuna sortu zueneko data (RFC 822, 5. atala formatuan, 1s-ko bereizmenarekin).

Last-Modified: baliabidea azkenengoz aldatu zeneko data.

ETag: entitate* bereizlea; baliabide berdinaren bi bertsio bereizteko erabiltzen da, adibidez:

URIa: http://kso2018.com:8080/baliabidea

```
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
Content-Length: 12
Content-Type: text/plain; charset=ISO-8859-1

Hello World!
```

```
Last-Modified: Tue, 17 Sep 2013 13:00:02 GMT
Content-Length: 76
Content-Type: text/html; charset=ISO-8859-1

<html><head><title>index.html</title></head>
<body>Hello World!</body></html>
```

* entitatea: goiburu jakin batzuk eta mezuaren gorputzak osotzen duten multzoa (RFC 2616, 7. atala).

Mezuaren gorputza: edukia; kasu honetan, HTML dokumentua (web orria).

JARDUERA:
PYTHON-EN HTTP ESKAERA BAT BIDALI

- *httplib* liburutegia erabiliz, www.google.com zerbitzariaren 80. portuan entzuten dagoen web aplikazioari **/** baliabidea eskatzen dion eta HTTP eskaera horri dagokion HTTP erantzunaren status kodea bistaratzen duen Python script-a programatu ezazu.

JARDUERA:

PYTHON-EN HTTP ESKAERA BAT BIDALI

```
# -*- coding: UTF-8 -*-

import httplib # HTTP protokoloa inplementatzen duen liburutegia

print "\r\n---> TCP konexioa ezartzen... "
zerbitzaria = 'www.google.com'
 conn = httplib.HTTPConnection(zerbitzaria) # TCP konexioa definitu
conn.connect() # TCP konexioa ezarri
print "---> TCP konexioa ezarrita!!!"

socketa = conn.sock.getsockname() # socketa IP_helbide-TCP_portu
                                  # bikoteaz osoturik dago
print "     Local IP address is " + str(socketa[0])
print "     Local TCP port is " + str(socketa[1])

print "---> HTTP eskaera: metodoa, URIa, goiburuak, edukia"
metodoa = 'GET'
uria= '/'
eskaeraren_goiburuak = {'Host': zerbitzaria,
                        'User-Agent': 'Nire Python bezeroa',}
eskaeraren_edukia = ''
conn.request(metodoa, uria, headers=eskaeraren_goiburuak, body=eskaeraren_edukia)
print "---> HTTP eskaera bidali da”

print "---> HTTP erantzuna jasotzen…"
erantzuna = conn.getresponse()
print "     Status: " + str(erantzuna.status)

conn.close()
```

HELBURUAK

TEORIA

- HTTP protokoloaren funtzionamendua deskribatu.
  - Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?
  - **Zelan egiten dira berhelbideraketak? (3xx erantzun kodeak)**
  - Zelan kargatzen da web orri bat nabigatzailean?
- Galdera horiek erantzuteko, adibideen bitartez HTTP protokoloaren semantika eta sintaxia deskribatuko dira:
  - Eskaera eta erantzunaren egitura.
  - Protokoloaren funtzionamendua inplementatzeko metodoak eta goiburuak.

PRAKTIKA

- Python-en HTTP bezeroak programatu:
  - Google-era konektatu.

HTTP-REN FUNTZIONAMENDUA: BERHELBIDERAKETAK

- **Zelan egiten dira berhelbideraketak? (3xx erantzun kodeak)**

  – Batzutan, baliabide baten URI-a aldatu daiteke, edo web zerbitzari batek web bezeroa erantzun “hobeago” bat jasoko duen beste URI batetara berhelbideratu dezake.

  – HTTP protokoloak beste URI batetara berhelbideraketa adierazteko modua eskeintzen du. Horretarako,
    - 301, 302 edo 303 erantzun kodeak
    - eta “Location” goiburua erabiltzen dira.

HTTP-REN FUNTZIONAMENDUA: BERHELBIDERAKETAK

- Demagun Madrilen kokatutako bezero batek ondorengo URI-a duen baliabidea eskatzen duela: http://www.google.com/
- HTTP eskaera hori erantzuten duen zerbitzariak paketearen jatorrizko IP helbidea Espainiakoa dela antzematen du: bezeroari http://www.google.es/ URI-ra berhelbideraketa egiteko erantzuna bidaltzen dio.
- Bezeroak, erantzunean 302 kodea detektatzean, “Location” goiburuaren balioa atera eta URI berrira beste eskaera bat egiten du. Nabigatzailean prozesu hau erabiltzailearentzako modu gardenean gertatzen da.

Eskaera baten adibidea

```
GET / HTTP/1.1
Host: www.google.com
Accept: text/html
Accept-Encoding: identity
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla Windows Desktop
```

Erantzun baten adibidea

```
HTTP/1.1 302 Found
Content-Length: 137
Content-Type: text/html; charset=UTF-8
Location: http://www.google.es/

<html><head><title>Redirection
302</title></head><body><a
href=“http://www.google.es/>Redirect to
http://www.google.es/</a></body></html>
```

HELBURUAK

TEORIA

- HTTP protokoloaren funtzionamendua deskribatu.
  - Zer gertatzen da erabiltzaile batek baliabide bat (adibidez, web orrialde bat) nabigatzailearen bitartez eskatzen duenean?
  - Zelan egiten dira berhelbideraketak? (3xx erantzun kodeak)
  - **Zelan kargatzen da web orri bat nabigatzailean?**
- Galdera horiek erantzuteko, adibideen bitartez HTTP protokoloaren semantika eta sintaxia deskribatuko dira:
  - Eskaera eta erantzunaren egitura.
  - Protokoloaren funtzionamendua inplementatzeko metodoak eta goiburuak.

PRAKTIKA

- Python-en HTTP bezeroak programatu:
  - Google-era konektatu.

JARDUERA:
PYTHON-EN HTTP BERHELBIDERAKETA BAT BURUTU

- *httplib* liburutegia erabiliz, www.google.com zerbitzariaren 80. portuan entzuten dagoen web aplikazioari **/** baliabidea eskatu eta fitxategi batean gordetzen duen Python script-a programatu ezazu.

HTTP-REN FUNTZIONAMENDUA:

WEB ORRIALDE BATEN KARGA

GET / HTTP/1.1
Host: www.ehu.es
Accept: text/html
Accept-Encoding: gzip,identity;q=0.5
Accept-Language: en-US,es-ES;q=0.8
User-Agent: Mozilla/5.0
(Windows NT 6.1; WOW64; rv:26.0)
Connection: keep-alive*

www.ehu.es
158.227.0.65

* "Connection" goiburuak erantzuna bidali/jaso ostean TCP konexioarekin zer egin adierazten du.
Connection: close
→ TCP konexioa itxi
Connection: keep-alive
→ TCP konexioa zabalik mantendu

bezeroa

zerbitzaria

HTTP-REN FUNTZIONAMENDUA:

WEB ORRIALDE BATEN KARGA

bezeroa

* “Keep-Alive” goiburuak TCP konexioa mantentzeko baldintzak zehazten ditu.

timeout=2 → TCP konexio bat HTTP trafiko gabe mantendu daitekeen denbora (s-tan) adierazten du.

max=500 → TCP konexio baten barnean bidali daitekeen HTTP eskaera kopuru maximoa.

HTTP/1.1 200 OK
Date: Thu, 20 Nov 2015 20:25:52 GMT
Last-Modified: Tue, 17 Sep 2015 13:00:02 GMT
ETag: "1a968-3ec-4e693e61bb8b6"
Content-Length: 1004
Content-Type: text/html; charset=ISO-8859-1
Keep-Alive: timeout=2, max=500*

WEB ORRIA (HTML DOKUMENTUA)

www.ehu.es
158.227.0.65

zerbitzaria

HTTP-REN FUNTZIONAMENDUA:

WEB ORRIALDE BATEN KARGA

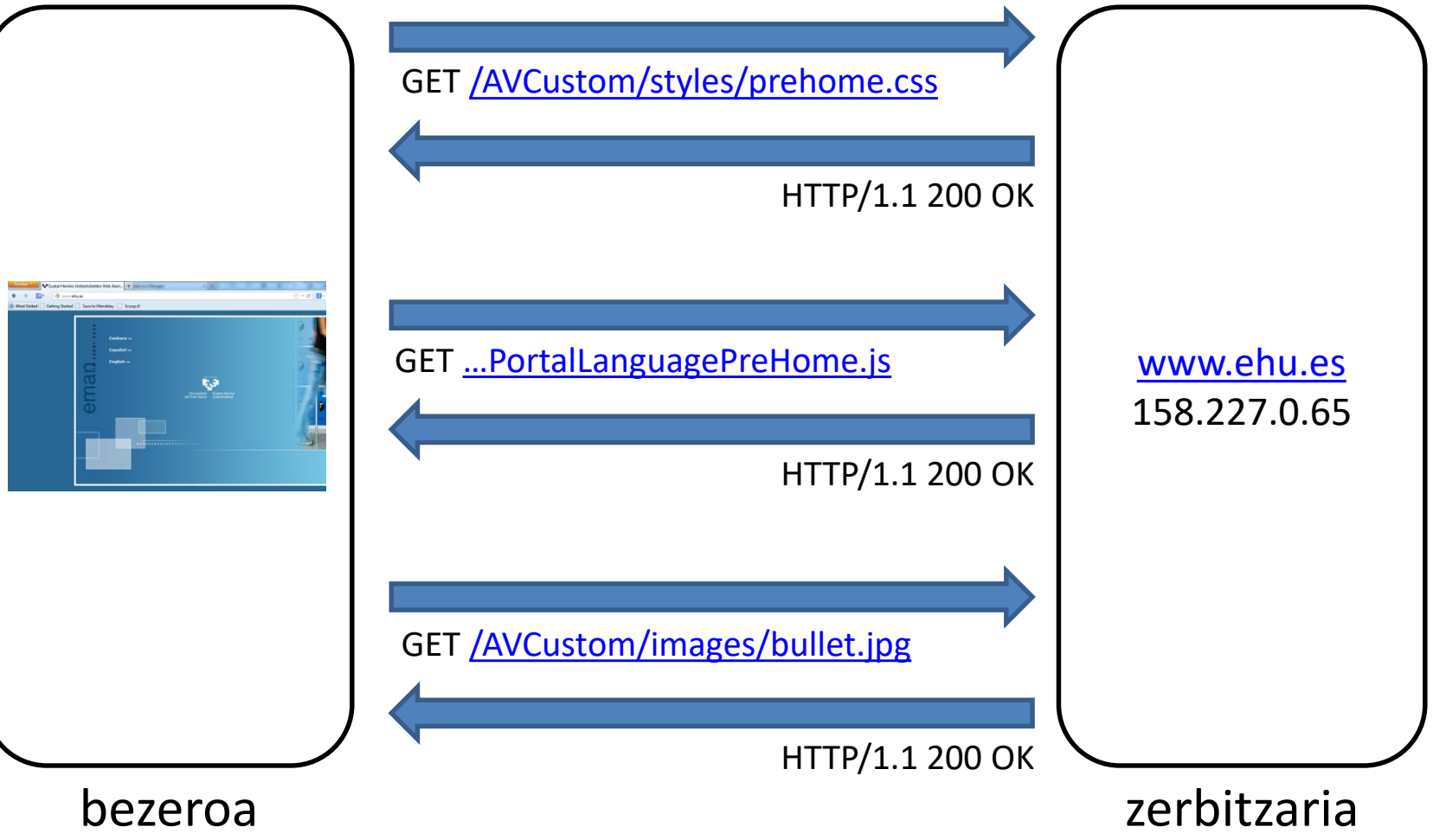
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
Transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="eu">
   <head>
      <title>Euskal Herriko Unibertsitateko Web Ataria/Portal web de la Universidad del País Vasco</title>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1“/>
      <link href="/AVCustom/styles/prehome.css" rel="stylesheet" type="text/css“/>
      <script type="text/javascript" src="/AVCustom/r01gLangSelectorVA/scripts/PortalLanguagePreHome.js“/>
   </head>
   <body>
      <script type="text/javascript"> testPreHome("p200","home","home","shenhm"); </script>
      <div id=container>
         <ul>
            <li><a href="/p200-home/eu/">Euskara </a><img src="/AVCustom/images/bullet.jpg"></li>
            <li><a href="/p200-home/es/">Español </a><img src="/AVCustom/images/bullet.jpg"></li>
            <li><a href="/p200-shenhm/en">English </a><img src="/AVCustom/images/bullet.jpg"></li>
         </ul>
      </div>
   </body>
</html>
```

WEB ORRIA

(HTML DOKUMENTUA)

HTTP-REN FUNTZIONAMENDUA:

WEB ORRIALDE BATEN KARGA



HTTP-REN FUNTZIONAMENDUA: TCP KONEXIOAREN KUDEAKETA

- Aurreko adibideko web orria kargatzeko 4 HTTP transferentzia egin behar dira:
  - HTML orria
  - irudia
  - CSS estilo orria
  - JavaScript kodea

- HTTP eskaera bati lotutako beste baliabideen deskarga ondorengo moduetan egin daiteke:
  - TCP konexio berdina erabilita, konexio iraunkorra ere deitua (eraginkorra*)
  - TCP konexio bereizietan (ez eraginkorra*)
  - edo modu mistoan.

- Azpiko TCP konexioak kudeatzeko, HTTP protokoloak bi goiburu ditu:
  - Eskaeran: “Connection”
  - Erantzunean: “Keep-Alive”

HTTP-REN FUNTZIONAMENDUA: TCP KONEXIOAREN KUDEAKETA

- Eraginkortasuna TCP konexio ezarpen (SYN, SYN-ACK, ACK) kopuruaren arabera neurtzen da.

- HTTP/1.1-ek TCP konexio iraunkorrak erabiltzen dituen arren, praktikan, nabigatzaile batek jatorri bakoitzeko batazbesteko 4-8 konexio zabaltzen ditu.

- Web orri bat jatorri askotako edukiak hartuta osotzen dela kontutan izanda, ohiko web orri batek 30 TCP konexio baino gehiago ezar ditzake, honek suposatzen duen gainkargarekin.

- Arazo hau ekiditzeko, jatorri bakoitzeko TCP konexio bakarra erabili behar da.