

Laboratorio de Computación IV

Clase 7

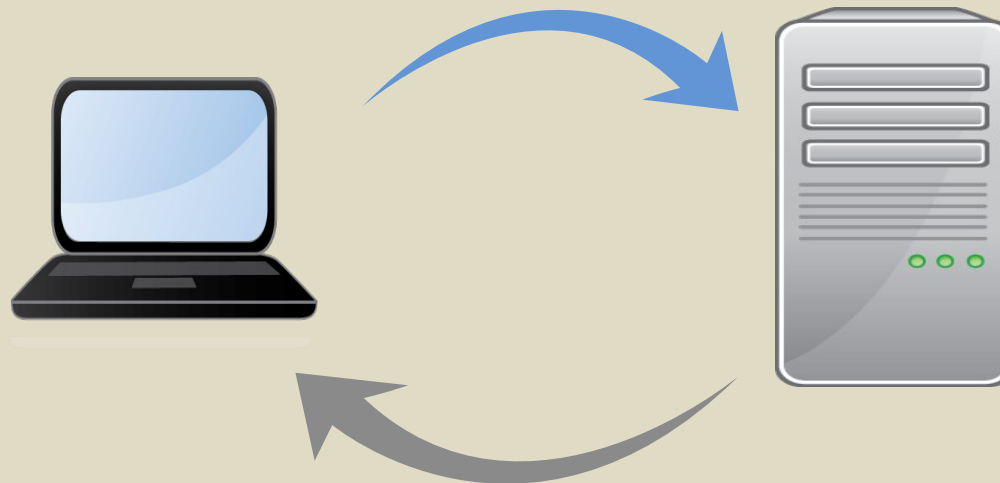
Andrés Fortier

¿Consultas?

- Almacenamiento de passwords
 - Texto plano.
 - Encriptación bidireccional.
 - Hashing criptográfico (bcrypt).
- Entrega 1.
- Git y Github para manejo de código.
- Github issues.

Protocolo HTTP

- Protocolo a nivel de aplicación.
- La base de la comunicación para la World Wide Web (www).
- Se basa en el concepto de *request-response* en un modelo cliente-servidor.



Protocolo HTTP



- *Stateless / Connectionless*
 - El cliente y el servidor no “recuerdan” nada del otro fuera del ciclo request-reponse.
- Diferentes tipos de datos (*media*).
 - Puede transportar distintos tipos de datos, siempre que el cliente y el servidor los puedan manejar.
 - MIME-types (ej. text/plain, image/bmp, etc).

Uniform Resource Identifiers



- *URI*

- Un string que se utiliza para identificar, en forma unívoca, un recurso.

- Ejemplo: <http://www.example.com/index.html>

- Forma genérica:

- `"http://" host [":" port] [abs_path ["?" query]]`

Protocolo HTTP



- HTTP: Acción sobre un recurso
 - Acción == método (*method*) o verbo (*verb*).
 - Recurso == URI.
- Verbos HTTP que mas vamos a utilizar:
 - GET
 - POST

Protocolo HTTP

- Otros muy frecuentes
 - HEAD
 - PUT
 - PATCH
 - DELETE
- Los que seguramente no veamos
 - TRACE
 - OPTIONS
 - CONNECT

Protocolo HTTP



- Métodos seguros (ej. HEAD, GET, OPTIONS).
- Métodos que generan una modificación (PUT, POST, PATCH).
- Métodos idempotentes
 - Los seguros (HEAD, GET, OPTIONS).
 - PUT y DELETE.
 - **POST no lo es.**

Protocolo HTTP

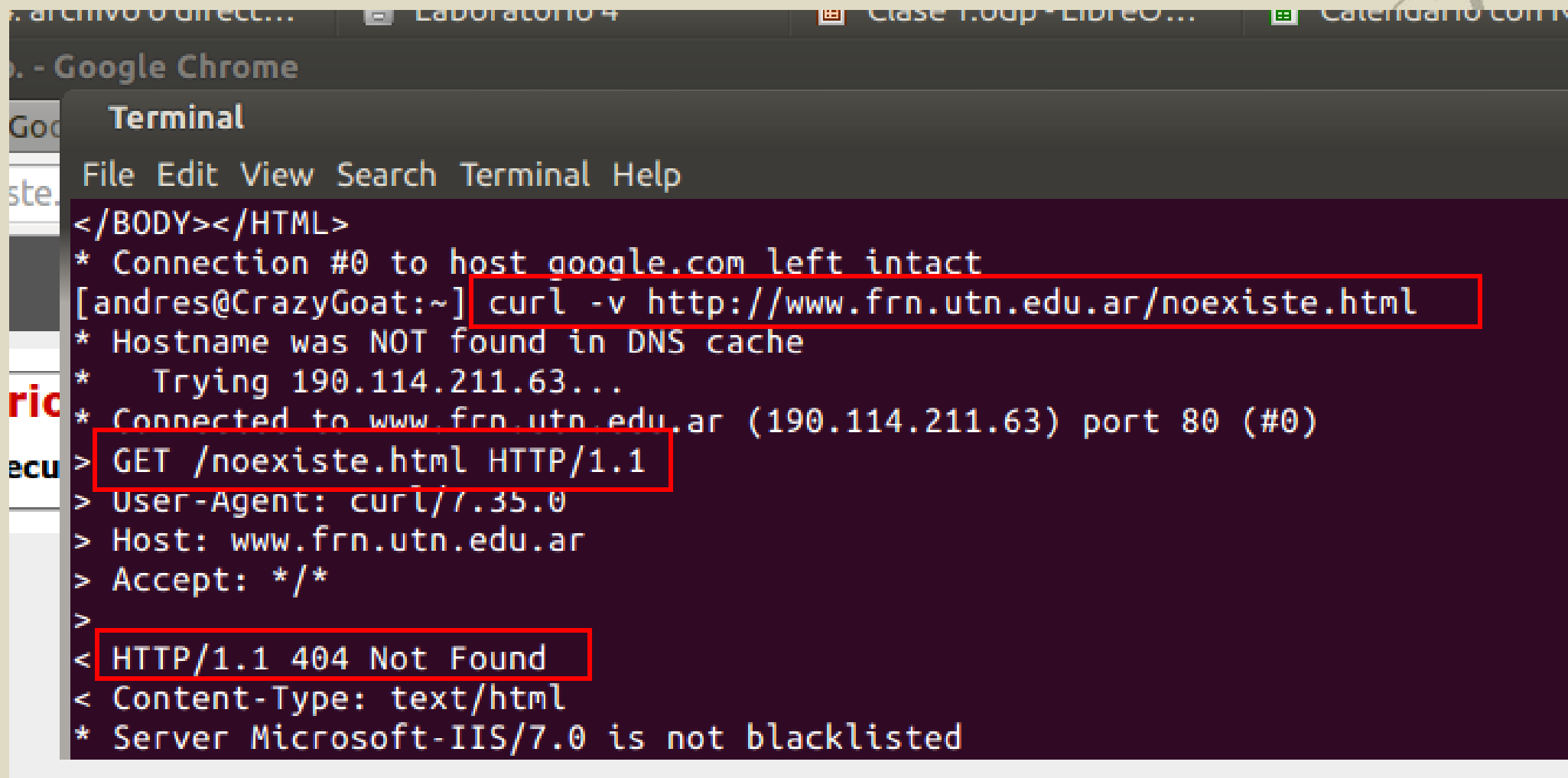


- Respuesta: código + contenido
 - Código de status
 - 1xx - Información.
 - 2xx - Exitosa.
 - 3xx - Redirección.
 - 4xx - Error del cliente.
 - 5xx - Error del servidor.
 - Contenido
 - Header
 - Body

Protocolo HTTP

```
Terminal
File Edit View Search Terminal Help
[andres@CrazyGoat:~] curl -v http://google.com/
* Hostname was NOT found in DNS cache
*   Trying 173.194.42.110...
* Connected to google.com (173.194.42.110) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.35.0
> Host: google.com
> Accept: */*
>
< HTTP/1.1 302 Found
< Cache-Control: private
< Content-Type: text/html; charset=UTF-8
< Location: http://www.google.com.ar/?gfe_rd=cr&ei=p7_8VNmHIsiB8QeG3oC4Cg
< Content-Length: 262
< Date: Sun, 08 Mar 2015 21:31:19 GMT
* Server GFE/2.0 is not blacklisted
< Server: GFE/2.0
< Alternate-Protocol: 80:quic,p=0.08
<
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.com.ar/?gfe_rd=cr&ei=p7_8VNmHIsiB8QeG3oC4Cg">here</A>.
</BODY></HTML>
* Connection #0 to host google.com left intact
[andres@CrazyGoat:~] █
```

Protocolo HTTP



The image shows a terminal window with a dark background and light-colored text. The terminal is titled "Terminal" and has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The prompt is "[andres@CrazyGoat:~]". The command being executed is "curl -v http://www.frn.utn.edu.ar/noexiste.html". The output shows the connection process, including DNS lookup and connection to the host. The request is a GET request for "/noexiste.html" using HTTP/1.1. The response is an HTTP/1.1 404 Not Found status, with a Content-Type of text/html. The server is identified as Microsoft-IIS/7.0.

```
</BODY></HTML>
* Connection #0 to host google.com left intact
[andres@CrazyGoat:~] curl -v http://www.frn.utn.edu.ar/noexiste.html
* Hostname was NOT found in DNS cache
*   Trying 190.114.211.63...
* Connected to www.frn.utn.edu.ar (190.114.211.63) port 80 (#0)
> GET /noexiste.html HTTP/1.1
> User-Agent: curl/7.35.0
> Host: www.frn.utn.edu.ar
> Accept: */*
>
< HTTP/1.1 404 Not Found
< Content-Type: text/html
* Server Microsoft-IIS/7.0 is not blacklisted
```

Protocolo HTTP

```
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /noexiste.html was not found on this server.</p>
<hr>
<address>Apache/2.2.22 (Debian) Server at www.frn.utn.edu.ar Port 80</address>
</body></html>
* Connection #0 to host www.frn.utn.edu.ar left intact
[andres@CrazyGoat:~] █
```

Protocolo HTTP



- Veamos
 - curl -v
<http://andres-fortier.github.io/laboratorio4-2016/>
 - Y en el navegador.

HTML

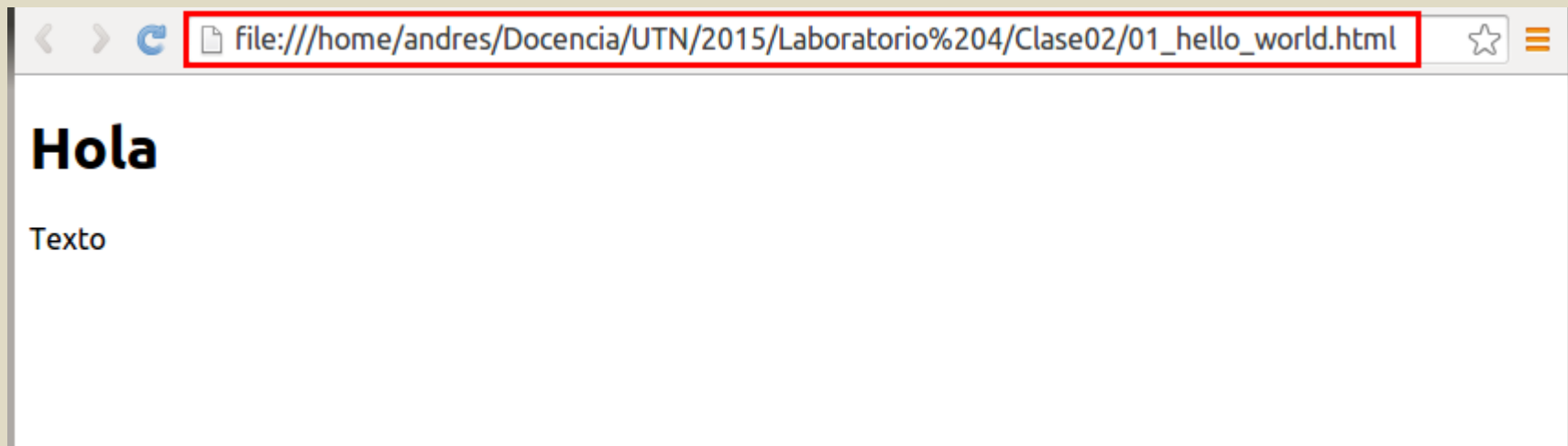
- HTML == *Hyper Text Markup Language*.
 - *Hypertext*: Texto con enlaces a otros textos.
 - *Markup*: Anotar un texto con etiquetas (*tags*).
- Ejemplo

01_hello_world.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Laboratorio 4</title>
  </head>
  <body>
    <h1>Hola</h1>
    <p>Texto</p>
  </body>
</html>
```

HTML + HTTP

- Formas de verlo:
 - Abrir el archivo con un navegador



- HTTP?!

HTTP + HTML

– Ejecutar un servidor HTTP y servir la página

- `cd <directorio dónde se encuentra el archivo>`
- `python -m SimpleHTTPServer 8000`

```
$ python -m SimpleHTTPServer 8000
Serving HTTP on 0.0.0.0 port 8000 ...
```

- En el navegador, ir a

http://localhost:8000/01_hello_world.html

```
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [19/Mar/2015 22:22:10] "GET /
01_hello_world.html HTTP/1.1" 200 -
127.0.0.1 - - [19/Mar/2015 22:22:11] code 404, message
File not found
127.0.0.1 - - [19/Mar/2015 22:22:11] "GET /favicon.ico
HTTP/1.1" 404 -
```


HTTP + HTML

- `curl -v localhost:8000/01_hello_world.html`

```
Serving HTTP on 0.0.0.0 port 8000 ...
127.0.0.1 - - [19/Mar/2015 22:22:10] "GET /
01_hello_world.html HTTP/1.1" 200 -
127.0.0.1 - - [19/Mar/2015 22:22:11] code 404, message
File not found
127.0.0.1 - - [19/Mar/2015 22:22:11] "GET /favicon.ico
HTTP/1.1" 404 -
127.0.0.1 - - [19/Mar/2015 22:24:23] "GET /
01_hello_world.html HTTP/1.1" 200 -
```

HTTP + HTML

```
[andres@CrazyGoat:~/Docencia/UTN/2015/Laboratorio 4/Clase02] curl -v localhost:8000/01_hello_world.html
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8000 (#0)
> GET /01_hello_world.html HTTP/1.1
> User-Agent: curl/7.35.0
> Host: localhost:8000
> Accept: */*
>
* HTTP 1.0. assume close after body
< HTTP/1.0 200 OK
< Server: SimpleHTTP/0.6 Python/2.7.6
< Date: Sun, 15 Mar 2015 20:50:22 GMT
< Content-type: text/html
< Content-Length: 137
< Last-Modified: Sun, 15 Mar 2015 20:25:25 GMT
<
<!DOCTYPE html>
<html>
  <head>
    <title>Laboratorio 4</title>
  </head>
  <body>
    <h1>Hola</h1>
    <p>Texto</p>
  </body>
</html>
* Closing connection 0
```

HTTP + HTML

- SimpleHTTPServer es un servidor HTTP.
 - Da acceso a todo el directorio en forma recursiva.
 - No sólo sirve html (ej. txt, png, etc.).

Etiquetas HTML

- Títulos: <h1> ... <h6>

```
<h1>Hola</h1>  
<h2>Hola</h2>  
<h3>Hola</h3>  
<h4>Hola</h4>  
<h5>Hola</h5>  
<h6>Hola</h6>
```

- Párrafos: <p>

```
<p>En un párrafo</p>
```

Etiquetas HTML

- Líneas horizontales: `<hr />`

```
<p>Párrafo 1</p>  
<hr />  
<p>Párrafo 2</p>
```

- Preservar formato: `<pre>`

```
<pre>Prueba    de    espacios</pre>
```

- Atributos básicos: ``, `<i>`, `<u>`

```
<b>Texto</b>
```

Etiquetas HTML

- Recordemos que los elementos HTML se pueden anidar

```
<p>Texto <b>con</b> muchas <i><u>cosas</u></i></p>
```

```
<p>Texto  
  <b>con</b>  
  muchas  
  <i>  
    <u>cosas</u>  
  </i>  
</p>
```

Etiquetas HTML

- Listas: `` y `` + ``

```
<ol>  
  <li>Elemento 1</li>  
  <li>Elemento 2</li>  
  <li>Elemento 3</li>  
</ol>
```

- Listas de definciones: `<dl>`, `<dt>` y `<dd>`

```
<dl>  
  <dt>Elemento A</dt><dd>Definicion A</dd>  
  <dt>Elemento B</dt><dd>Definicion B</dd>  
</dl>
```

Etiquetas HTML

- Tablas: <table>, <tr>, <th>, <td>

```
<table>
  <tr>
    <th>Columna 1</th>
    <th>Columna 2</th>
  </tr>
  <tr>
    <td>(1,1)</td>
    <td>(1,2)</td>
  </tr>
  <tr>
    <td>(2,1)</td>
    <td>(2,2)</td>
  </tr>
</table>
```


Etiquetas HTML

- Los elementos HTML poseen atributos
 - Algunos son comunes a todos.
 - Otros dependen del elemento en cuestión.
- Anchors: <a>

```
<a href="http://www.google.com">Google</a>
```

- Imágenes:

```

```

Etiquetas HTML

- Forms: <form>
 - Indicar una url para que procese los datos.
 - Y el método HTTP a usar.

```
<form action="procesar.php" method="post">  
  ...  
</form>
```

- Se pueden usar distintos tipos de componentes para pedir información al usuario:
 - Textbox:

```
<input type="text" name="Nombre" value="Pepe">
```

Etiquetas HTML

- Textarea:

```
<textarea name="descripcion" rows="5" cols="20">  
  La descripción  
</textarea>
```

- Radiobuttons:

```
<p>  
  <input type="radio" name="edad" value="A">1-20  
</p>  
<p>  
  <input type="radio" name="edad" value="B">21-40  
</p>  
<p>  
  <input type="radio" name="edad" value="C">41- ...  
</p>
```

Etiquetas HTML

- Select:

```
<select>
  <option value="1">Op 1</option>
  <option value="2" selected>Op 2</option>
  <option value="3">Op 3</option>
</select>
```

Tarea para el hogar



- Investigar / repasar sobre HTTP
 - Leer sobre los distintos verbos HTTP
 - ej.
https://es.wikipedia.org/wiki/Hypertext_Transfer_Protocol
 - Ejemplos:
 - `curl -v http://google.com/`
 - `curl -v http://www.frn.utn.edu.ar/noexiste.html`
 - `curl -v http://www.december.com/html/demo/hello.html`

Tarea para el hogar



- Crear un nuevo repositorio de git.
- Crear en el repositorio las siguientes páginas HTML y servir las usando el servidor HTTP python
 - Árbol genealógico partiendo de ustedes usando listas anidadas. Utilizar `<h1>` para un título y `<p>` para descripción.
 - Una página que utilice una imagen pública en internet y una local.

Tarea para el hogar



- Una página con los ejemplos descriptos en <http://www.htmlgoodies.com/tutorials/tables/article.php/3479791>
- Recuerden ejercitar git
 - Hagan *commits* periódicos y vean los *diffs*.
 - Pierdan el miedo a mover un archivo entre el *working directory* y *staging area*.