

Laboratorio de Computación IV

Clase 1

Andrés Fortier

Antes de comenzar



- Último semestre de la tecnicatura.
- Tienen los conceptos básicos, los vamos a poner en práctica en un proyecto.
- Piensen en una pasantía
 - No van a tener todo masticado. Internet + traducción si lo necesitan.
 - Trabajen desde el día uno. Asumo que van a estar al día con lo pedido.
 - Repasar las transparencias al otro día.

Antes de comenzar



- Las transparencias no van a ser tan detalladas como el año pasado
 - Tomar notas.
 - Nuevamente, internet + traducción si lo necesitan.
- Página:
 - <http://andres-fortier.github.io/laboratorio4/>
- Foro (no escriban al mail personal):
 - utn-sma-poo@googlegroups.com

Objetivos de la materia



- Desarrollar una aplicación web que contemple un conjunto de funcionalidades típicas.
- Exponerlos a un conjunto de herramientas comunes dentro de la industria del desarrollo de software.
- Poner un sistema en producción en un servidor de acceso público y mantenerlo.

Contenidos de la materia



- Aplicaciones web
 - Protocolo HTTP.
 - Lenguajes HTML, CSS y Javascript.
 - Servidores web.
 - MVC en la web.
 - Autenticación y conceptos básicos de seguridad.
 - Autorización. Roles de usuarios y administradores.
 - Persistencia y evolución incremental.
 - Testing de aplicaciones web.

Contenidos de la materia




- Herramientas:
 - Versionamiento de código: Git y Github.
 - Uso de tickets para la organización del trabajo. Desarrollo de software incremental.
 - Interfaz de línea de comandos: Terminales linux, SSH y comandos comunes.
 - Editores de texto de línea de comando e IDEs.

Contenidos de la materia

- Utilización de un servicio de hosting para colocar la aplicación en producción (Openshift).

Aprobación y promoción



- Centramos la materia en un trabajo que engloba los contenidos descriptos
 - No hay parciales o finales.
 - Tres entregas.
 - Se aprueba y promociona con promedio mayor o igual a 5.
 - Sólo se permite una entrega con nota menor a 5.

Aprobación y promoción


- Fechas de entrega:
 - 21 de Abril.
 - 2 de Junio.
 - 3 de Julio.

Sobre las entregas



- Código por medio de Github. Se toma lo publicado en “master” antes de que comience el día de la entrega
 - Ej: para 21/04/2015 se toma hasta 20/04/2015 23:59:59.
- 2da y 3ra entrega: sumar demo publicado en Openshift.
- Utilicen el sentido común para las entregas.

Organización de las clases



- Comando linux.
- Conceptos relacionados a desarrollo de aplicaciones web y/o Ruby.
- Opcional: alguna herramienta.
- “Tarea” para la clase que viene en base a lo visto.

Sobre las aplicaciones



- Se desarrollan individualmente.
- Deben implementar una funcionalidad mínima (nota 5).
- Funcionalidades extras consensuadas suman puntos.
- Distintas complejidades.
- Carta de presentación.
- Pueden armar un modelo de negocios.


Denuncia de baches/grafitis/etc.

- Estilo <http://www.buenosairesbache.com/>.
- Aplicación web móvil + web desktop.
- Técnicamente compleja
 - Acceso a la cámara y al GPS en mobile (html 5).
 - Upload de archivos.
 - Mapas
- Requiere 3G (no importa para la materia).
- Impacto en la sociedad.

Denuncia de baches/grafitis/etc.

- Opcionales
 - Funcionamiento offline.
 - Reportes: queries y exportación a CSV.
 - Votos por usuarios.
 - “Likes” y publicación en redes sociales.

Registro de niñeras/personal



- Trabajos sensibles donde la recomendación y el boca a boca pesan.
- Perfiles de las personas que brindan servicios.
- Costos por hora.
- Buscador básico.
- Recomendaciones y “reviews”.

Registro de niñeras/personal



- Opcionales
 - Manejo de horarios (disponibles y necesitados).
 - Buscador avanzado.
 - Notificación (ej. buscar al final del día).
 - Cálculos de costo por hora en base a estadísticas.

Armado de bandas



- Contactar músicos para el armado de bandas
 - Registro de músicos disponibles.
 - Registro de bandas que buscan músicos.
- Perfiles de músicos (instrumentos y géneros musicales).
- Búsquedas ordenadas por “matching”.

Armado de bandas

- Opcionales
 - Reviews.
 - Horarios disponibles de ensayo.
 - Ubicación geográfica.
 - Publicidad.

Equipos de desarrollo distribuidos



- Sincronizar a programadores distribuidos.
- Concepto de standup (que hice ayer / que voy a hacer hoy / blockers).
- Trazabilidad y estadísticas.
- Dashboard.
- Complejidad en trabajar en un dominio que no manejan.

Equipos de desarrollo distribuidos



- Opcionales
 - Enviar por mail.
 - Integrar herramientas externas (ej. github).
 - Manejo de grupos.
 - “Pausar” un item.

Todas las aplicaciones - mínimo




- Desarrollaremos sobre Ruby, usando RoR (Ruby on Rails).
- Funcionalidad de usuario básica
 - Usuarios y cuentas (id + password).
 - Backend de administración.
 - Al menos dos niveles de visibilidad, según la app
 - Roles de usuarios.
 - Usuario anónimo vs. Registrado.

Todas las aplicaciones - mínimo




- A nivel implementación
 - Utilización de un estilo (CSS) en forma consistente.
 - Tests del modelo.
 - Persistencia
 - Mantener información.
 - Migrar información conforme la app crece.

Todas las aplicaciones - opcional



- Algunos items dependen de la aplicación.
- Social login (facebook, twiter, gmail, etc).
 - Múltiples perfiles por usuario
- Google maps.
- File uploads.
- Javascript.
- Charts.

Todas las aplicaciones - opcional



- Notificaciones por e-mail.
- Log de errores.
- Tests de UI.
- Tests de integración.
- Métricas de código.
- Análisis de seguridad.
- Cosas que los motiven, bienvenidas sean.

Comando del día: cd

- cd == change directory.
- Se utiliza para cambiar el **directorio actual** de trabajo.

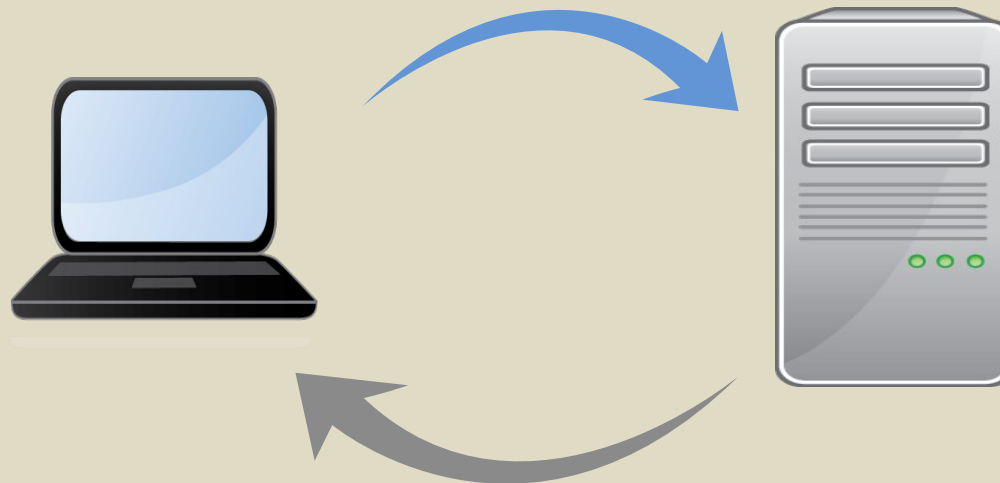
```
[andres@CrazyGoat:~] █
```

Comando del día: cd

- Ejemplos:
 - `cd .`
 - `cd ..`
 - `cd /home`
 - `cd ~` (o simplemente ``cd``)
 - `cd ../../home`
 - `cd -`
 - `cd /ho<tab>`

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.

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

```
archivo o direct... Laboratorio 4 Clase 1.00p-LibreO... Calendario con Nota... Programa.00c-Libre... [D.0.0K 0.0.0K-Beta... Terminal
Google Chrome
Terminal
File Edit View Search Terminal Help
</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

<div id="content">
  <div class="content-container"><fieldset>
    <h2>404: archivo o directorio no encontrado.</h2>
    <h3>Puede que se haya quitado el recurso que est buscando, que se le haya cambiado el nombre o que no est disponible temporalmente.</h3>
  </fieldset></div>
</div>
</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/>
 - Y en el navegador.

Tarea para el hogar



- Instalar linux.
- Tener a mano un editor de texto (ej. gedit).
- Investigar / repasar sobre HTTP.
 - 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`