Meteoroid

Tesina de grado de Licenciatura en Informática

Integrantes:

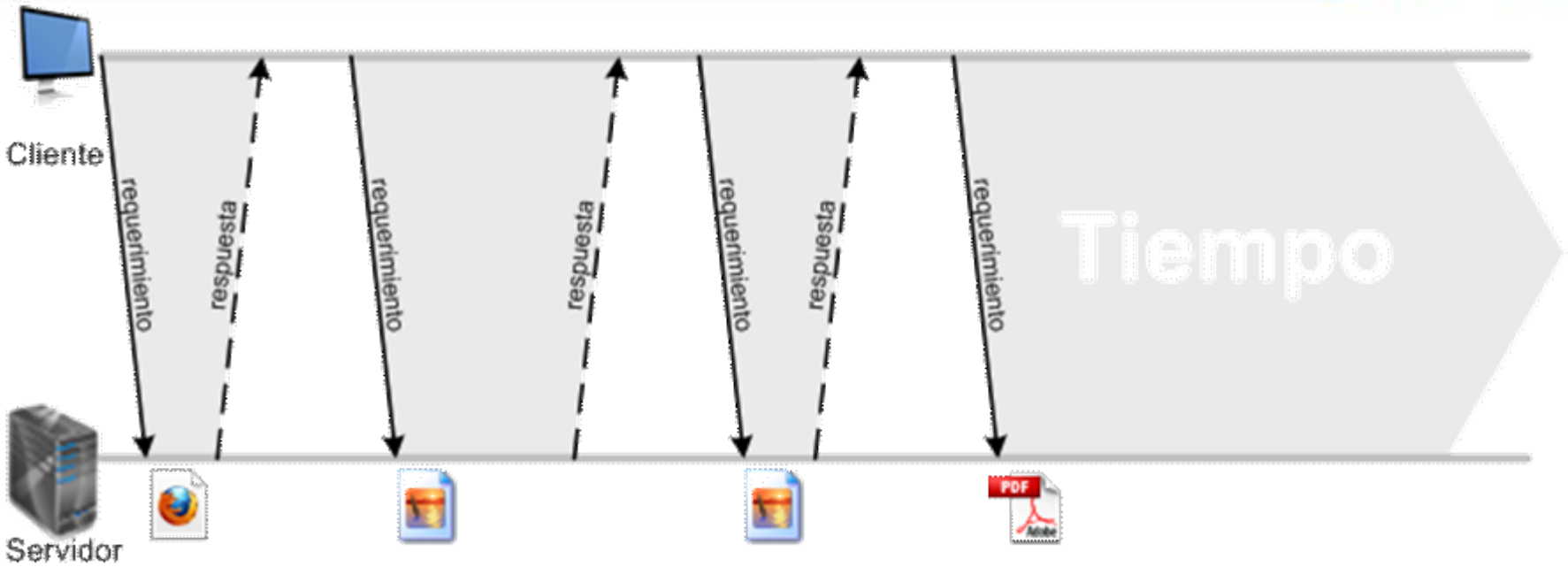
Lautaro Fernández
Santiago Robles

Director: Gustavo Rossi
Co-director: Silvia Gordillo

Web: Al infinito y más allá!

- Request/response
- Barata, rápida y accesible
- Muchos servicios se proveen vía Web
  - Mail
  - Home banking
  - Editores colaborativos
  - Redes sociales
  - E-Commerce
  - Etc.
- Utilizado en variedad de dispositivos

¿Cómo trabaja HTTP?

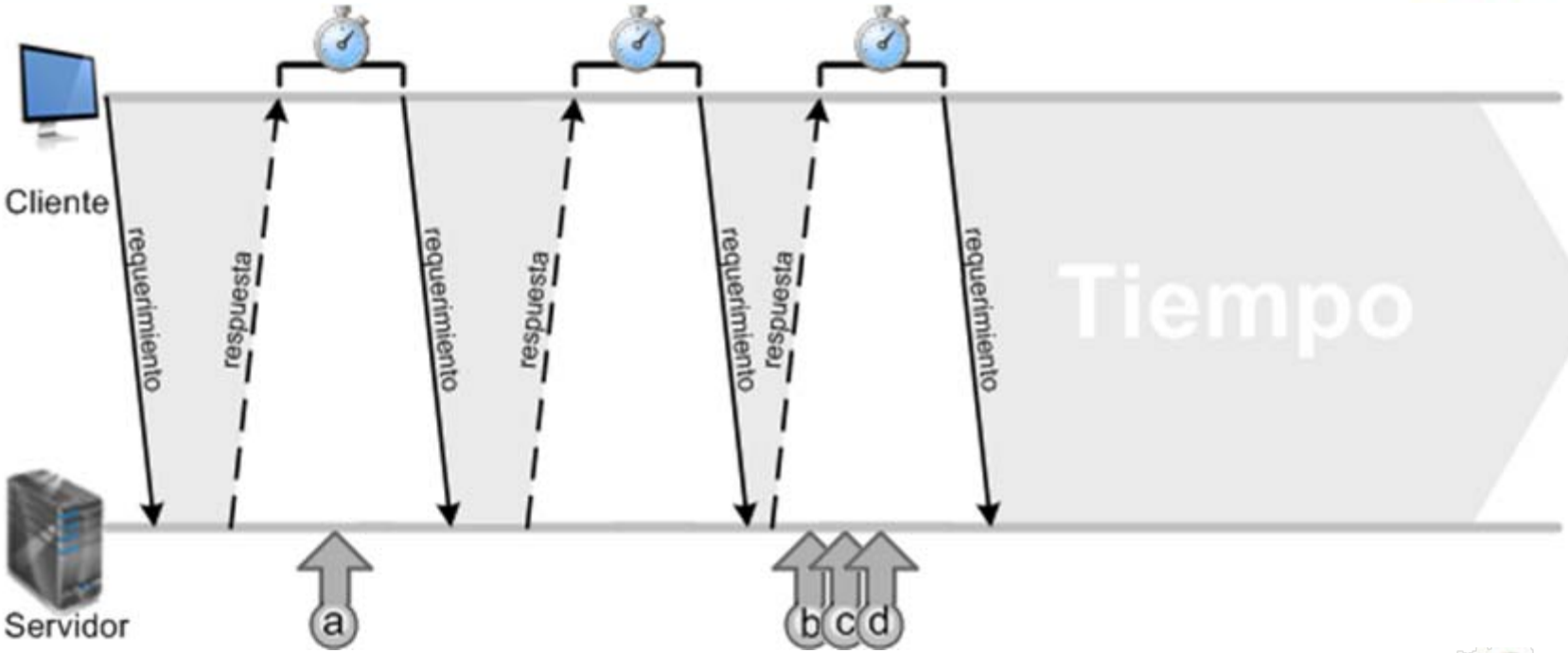


- Es un estándar
- Cada respuesta se debe a un requerimiento
- HTTP 1.0
  - Una conexión por requerimiento

Evolución de la Web

- Actualmente evolucionó para ser más interesante, requiriendo
  - Dominios de aplicación complejos
  - Información dinámica
  - Más interacción cliente-servidor
  - Necesidad de actualizaciones en el cliente sin un requerimiento explícito

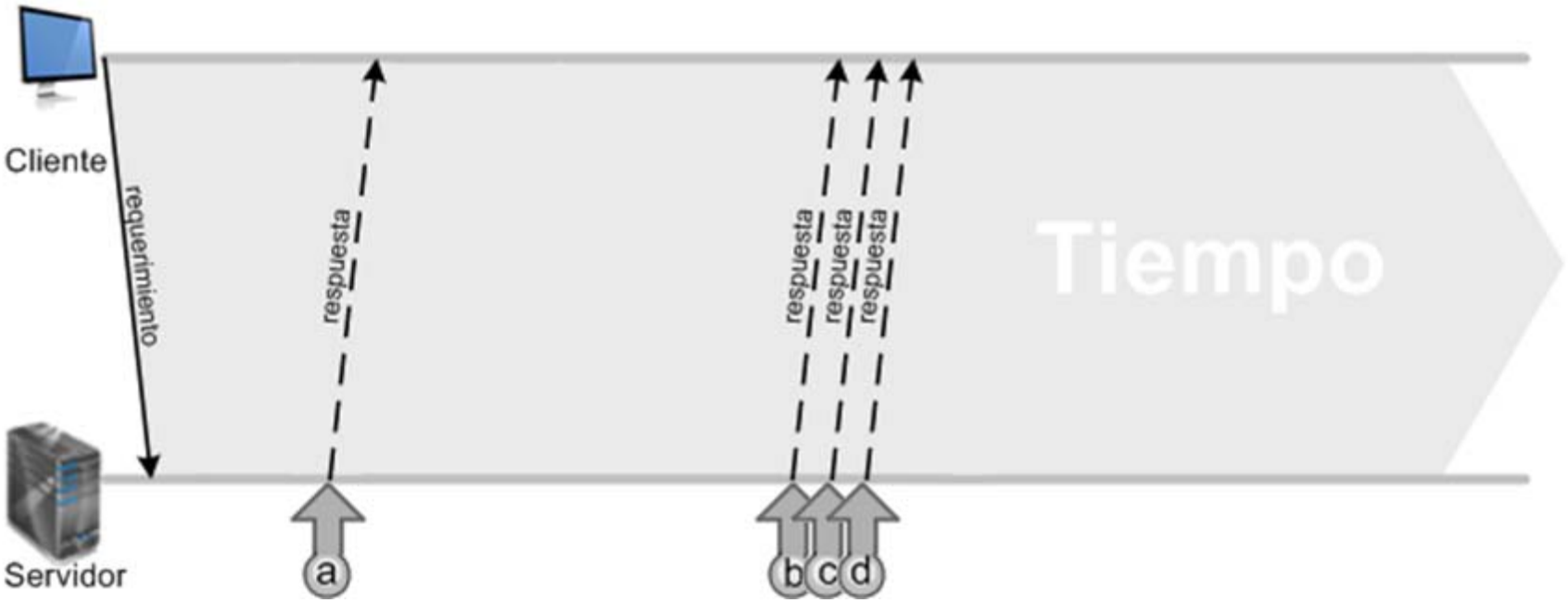
HTTP en aplicaciones de tiempo real



- Exigen Ajax + timers Javascript
- Metodología pobre de programación
  - Difícil estimar la frecuencia
  - Sobrecarga de requerimientos
  - Uso de cola de espera (por cliente)

Para este último es necesario un nuevo enfoque

Comet



- Envío de información desde el servidor a los clientes
- Conjunto de tecnologías
  - Javascript + DOM
  - Ajax
  - Streaming

Comet (cont.)

- Por el momento no es un estándar
- Técnica común para todos los navegadores
  - Problema del Waiting cursor 
  - Problema del throbber 
  - Problema de la barra de estado 
- Mejor técnica para cada navegador

	Forever IFrame	XML HttpRequest	Server-Sent Events	ActiveX + IFrame	WebSockets
					
					
					
					
 (others)					

Entendiendo las bases


```
= {
  Comet, Ajax,
  Seaside,
  Observer Pattern,
  Announcements,
  MVC, Value Models
}
```

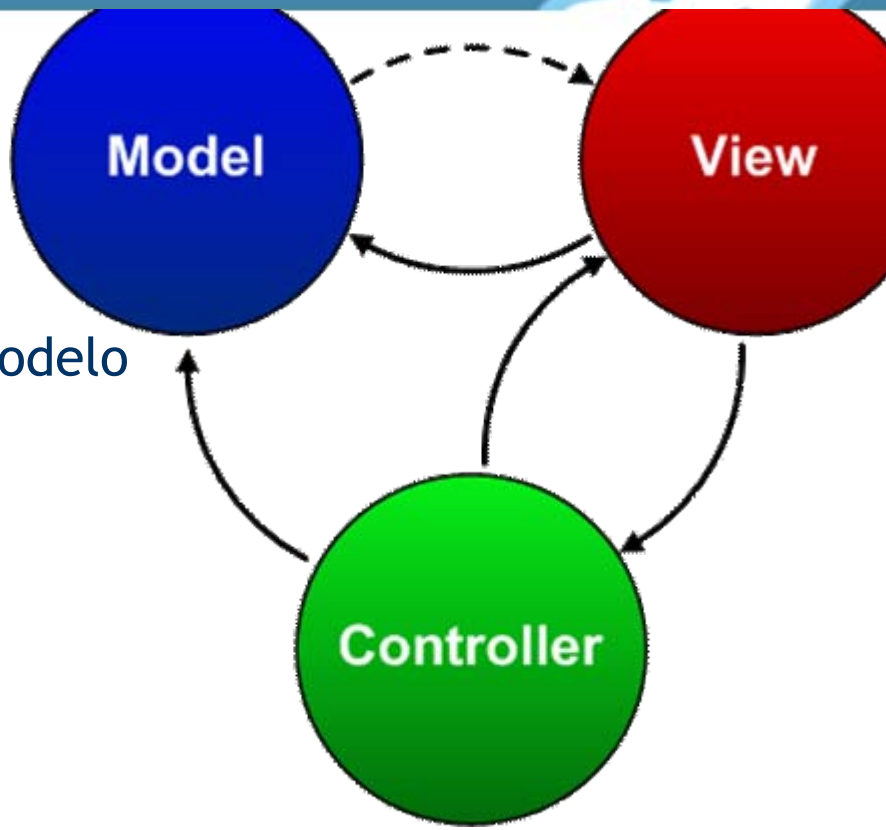
Seaside

- Framework para crear aplicaciones Web
- Características:
  - Componentes anidadas
  - Hot debugging and recompilation
  - Múltiples control flows
  - Action callbacks
  - Etc.
- #renderContentOn: html

Demo

Model-View-Controller

- Model
  - Independiente de las vistas
- View
  - Muestra la información del modelo
- Controller
  - Procesa eventos



Announcements

- Implementación del patrón Observer, más simple y potente que sus predecesores
- Utiliza objetos en vez de símbolos, lo que facilita el comportamiento del objeto interesado

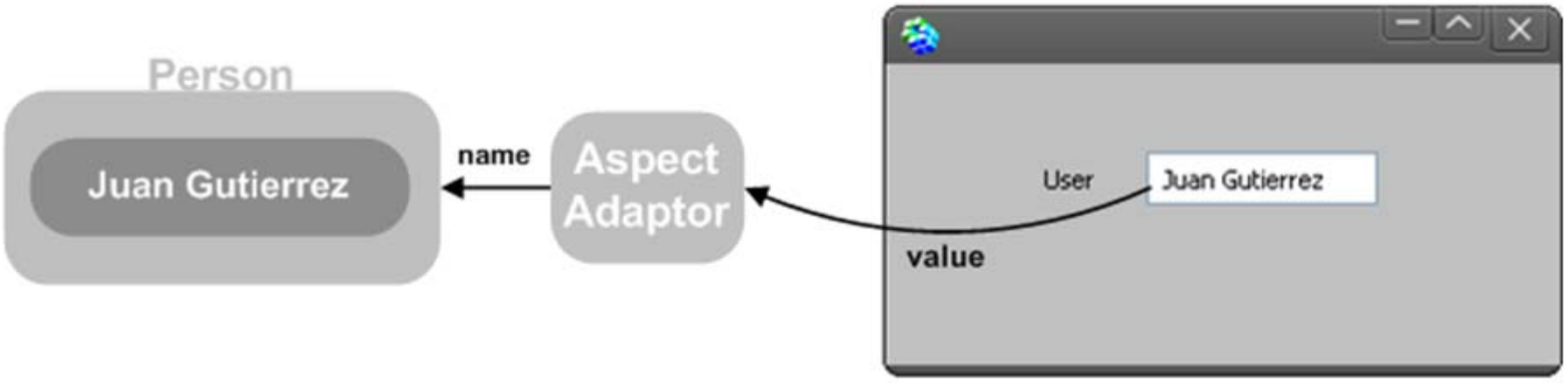
```
self change: #value with: aValue
```

Vs.

```
self announce: (ValueAnnouncement new: aValue)
```

Value Models

- Idea general
  - Contiene un modelo
  - Entiende el protocolo #value y #value:
  - Notifica a sus dependientes cuando su valor es modificado
  - Simple, estándar
- Útil para widgets, acceden al valor
  - Independientemente del modelo subyacente
  - De forma uniforme

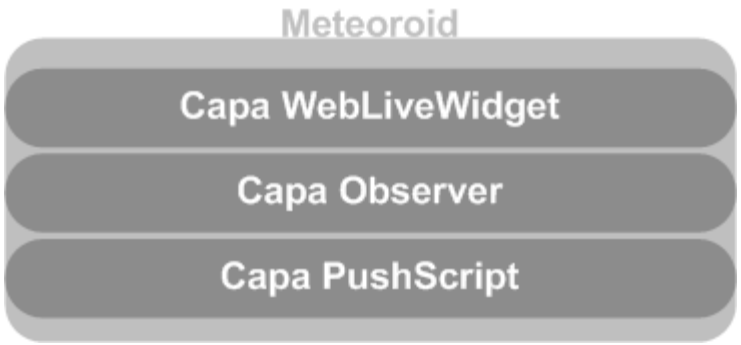


Meteoroid

- Framework para la creación de aplicaciones Web Vivas
- Combina
  - Comet
  - Seaside
  - Announcements
  - Patrón MVC
  - Value Models

Capas de Meteoroid

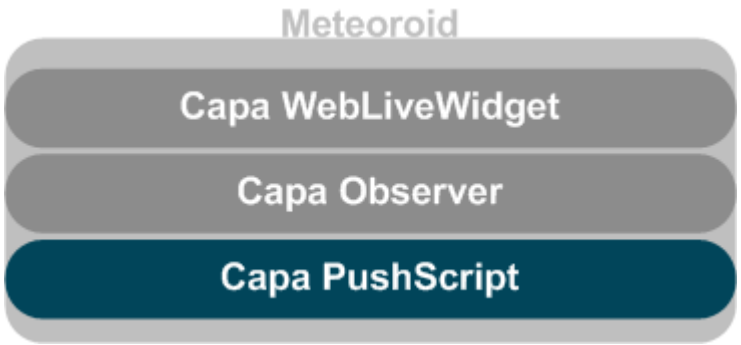
- Meteoroid esta dividido en capas
- Cada capa provee
  - Más funcionalidad
  - Más abstracción
- Fácil de instanciar
  - Heredar de componente Meteoroid
  - Utilizar sesión MeteoroidSession
  - API



Capa PushScript

- Capa base, menor abstracción
- Provee
  - Conexión Comet
  - El mensaje #pushScript: aScript
  - Manipulación de DOM

```
Componente>> sendHello
    self pushScript: 'alert("Hello World");'
```



Capa PushScript (cont.)

- **Elección de mejor técnica de conexión**
- Mantiene el protocolo Seaside (#renderContentOn:)
- Soporta anidación de componentes
- Única conexión por aplicación

	Forever IFrame	XML HttpRequest	Server-Sent Events	ActiveX + IFrame
		●	β	
			●	
				●
	●		β	
(others)	●			

Capa PushScript (cont.)

- Elección de mejor técnica de conexión
- **Mantiene el protocolo Seaside** (#renderContentOn:)
- Soporta anidación de componentes
- Única conexión por aplicación

con seaside

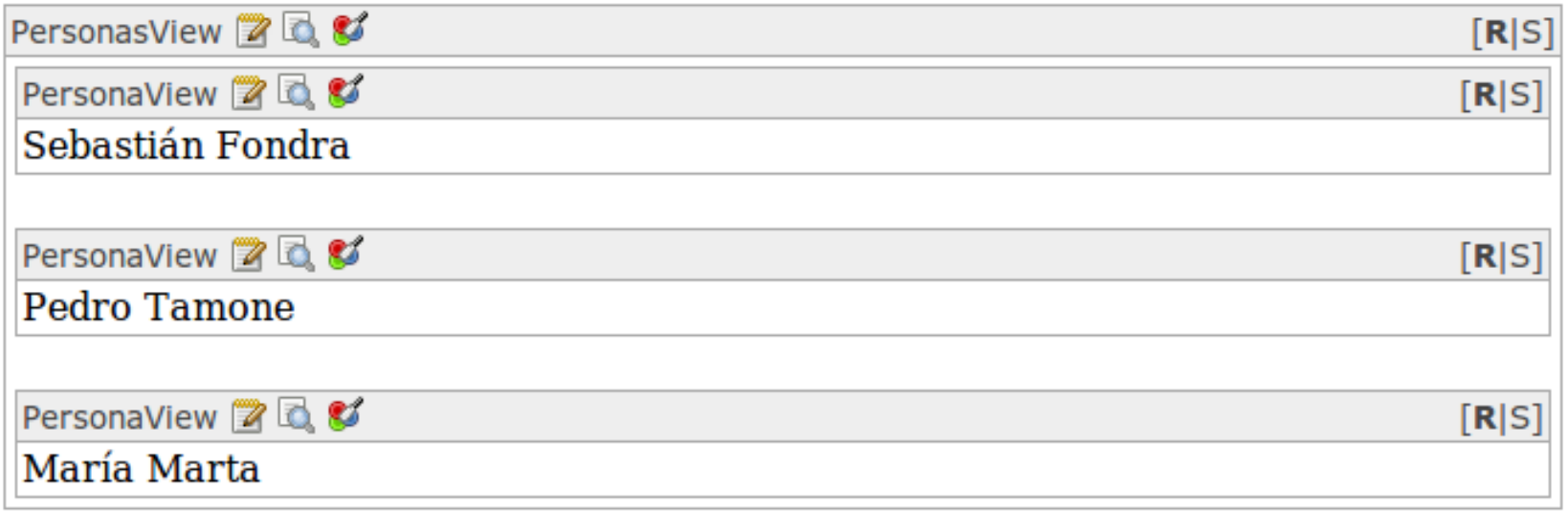
```
renderContentOn:html
    html paragraph: [
        html text: 'Hola '.
        html strong: 'mundo'
    ].
```

con Meteoroid

```
renderContentOn:html
    html paragraph: [
        html text: 'Hola '.
        html strong: 'mundo'
    ].
```

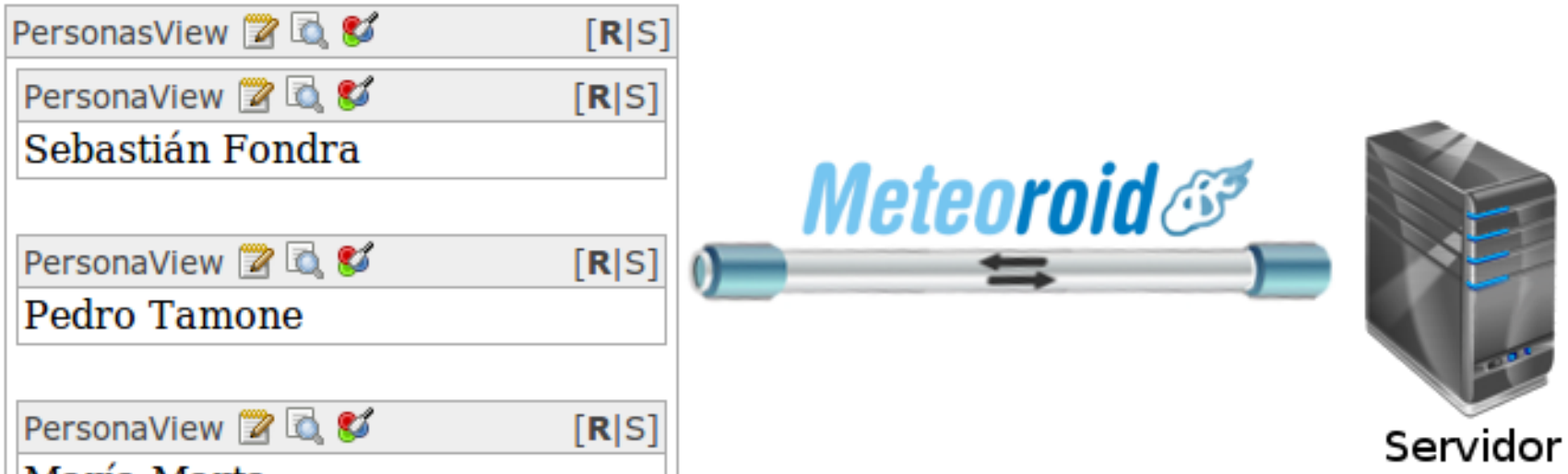
Capa PushScript (cont.)

- Elección de mejor técnica de conexión
- Mantiene el protocolo Seaside (#renderContentOn:)
- **Soporta anidación de componentes**
- Única conexión por aplicación



Capa PushScript (cont.)

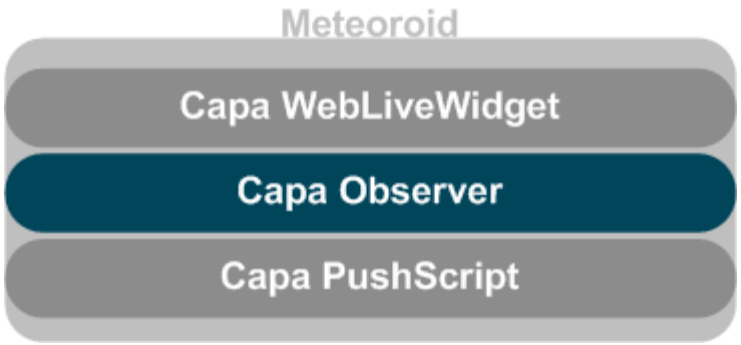
- Elección de mejor técnica de conexión
- Mantiene el protocolo Seaside (#renderContentOn:)
- Soporta anidación de componentes
- **Única conexión por aplicación**



Demo

Capa Observers

- Mayor abstracción
  - Permite el uso de dependencias necesarias en el patrón MVC
  - Agrega y remueve dependencias View-Model automáticamente
  - Mensajes para actualizar información automáticamente
- Tipos diferentes, según el caso
  - **Updaters**, reemplazan un elemento DOM
  - **Insertions**, agregan contenido
  - **Javascripting**, ejecuta Javascript



Capa Observers (cont.)

- ¿Cómo usarlo? 2 pasos!
  - Crear observer en la inicialización de la componente
  - Dibujar el tag a modificar
- El modelo lanza cambios (Announcements)

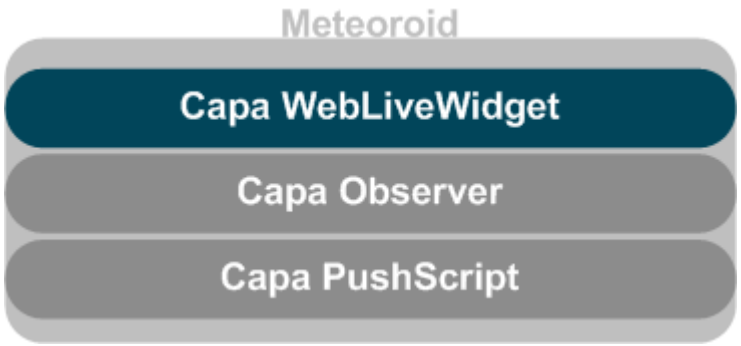
```
Componente>>initialize
  ...
  self on: AnnouncementX
      of: self model
      update: 'aDOMId'
      callback: [:html  :announcement :announcer |
                              html text: announcer value
      ]
```

```
Componente>>renderContentOn: html
   ...
   html div id: 'aDOMId'.
```

Demo

Capa Web Live Widgets

- Combina
  - Capa PushScript y Observers
  - HTML widgets
  - Value Models
- Permite crear aplicaciones Web como se crean aplicaciones de escritorio



Capa Web Live Widgets (cont.)

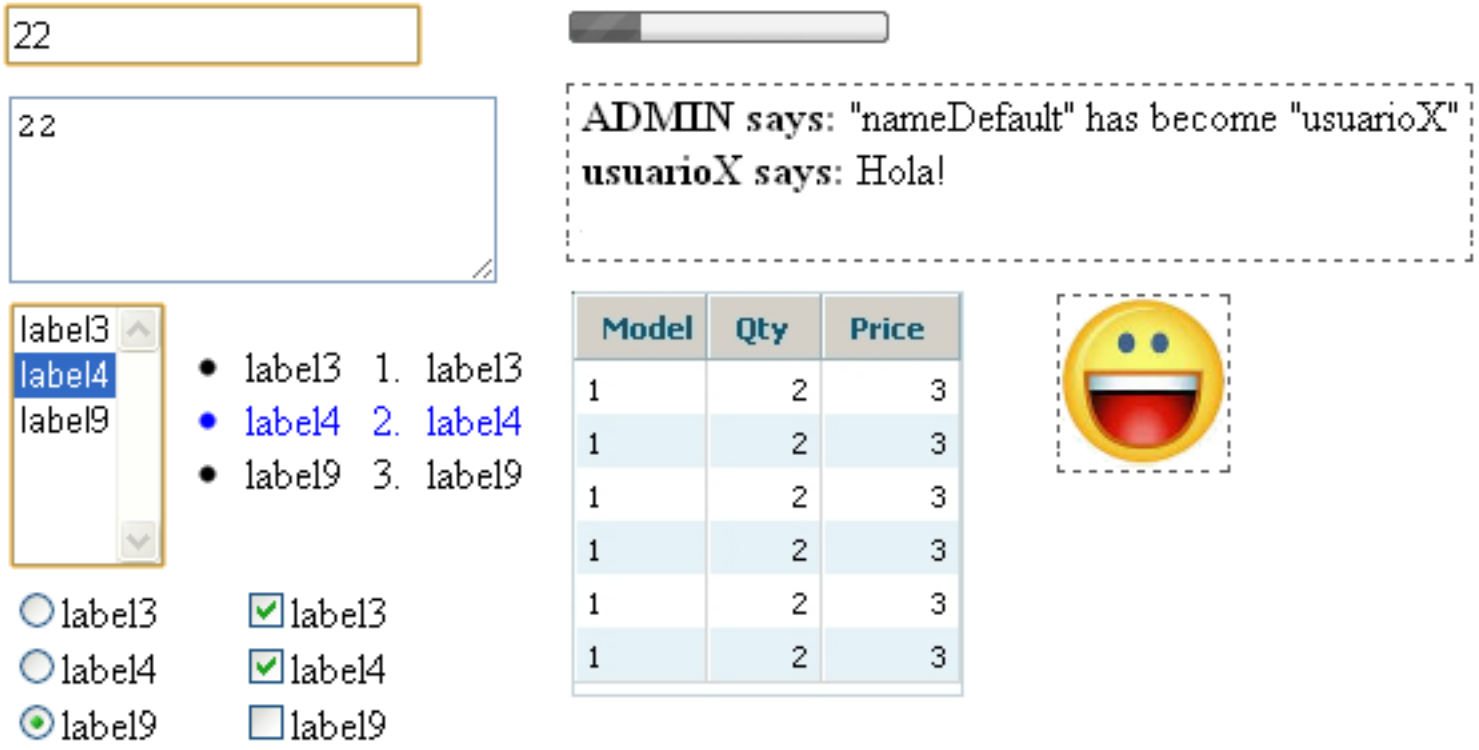
- ValueModelWrapper
  - Enmascaran ValueModels para agregar comportamiento
  - Dispara Announcements posibilitando el uso de capa Observer
- Librería Javascript WebLiveWidgetsLibrary
  - Abstrae código Javascript para manipular widgets
  - Crossbrowser

Capa Web Live Widgets (cont.)

- Live Widgets
  - Acceden a ValueModelWrappers
  - Responsables de definir las actualizaciones y saber cómo pintarse
  - Permiten definir vistas Web de la misma forma que de escritorio

Capa Web Live Widgets (cont.)

- Widgets implementados

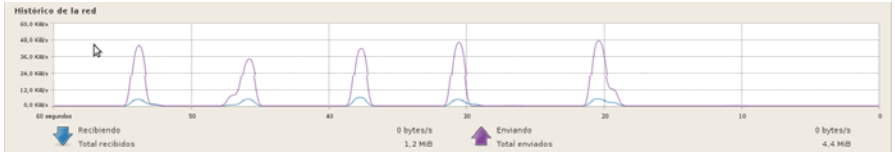


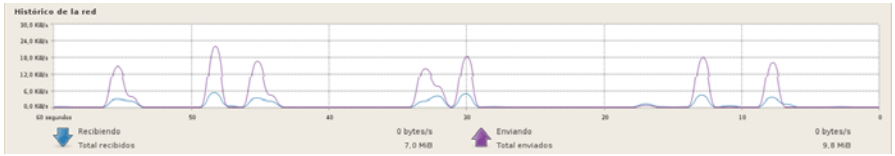
- input, textArea, div, span
- select, unorderedList, oderedList, divList
- radiobutton, checkbox
- image, progress bar, table

Demo

Conclusiones

- Aprender tecnologías nuevas, Comet en particular
- Generar un framework de alto nivel para crear aplicaciones Web
  - MVC real
  - Simple
  - “Backward compatibility”
  - Extensible
  - Posibilita migrar aplicaciones de escritorio a entornos Web
- Generar papers durante el proceso de investigación
- Benchmarkings empíricos





Publicaciones y presentaciones

- Publicaciones
  - “Meteoroid Towards a real MVC for the Web”
    - Proceedings of the International Workshop on Smalltalk Technologies 2009, páginas 28-37
    - Año: 2009
  - “Exploiting Personal Web Servers for Mobile Context-Aware Applications”
    - Knowledge Engineering Review (KER)
    - Año: 2010
  - “Meteoroid: a real MVC for the Web”
    - Korean Society Internet Information (KSII)
    - Año: 2010/en proceso
- Presentaciones
  - European Smalltalk User Group (ESUG), Brest France 2009
  - Smalltalks, Buenos Aires 2009

Trabajo a futuro

- Actual
  - Refactorizar las tres capas
  - Más ejemplos funcionales
  - Tests
  - Bugfixes
- Futuro
  - Hacerlo andar con Seaside 3.0 (o la versión más nueva)
  - Hacerlo más portable (Pharo/Squeak)
  - Más benchmarks
  - Extender los widgets vivos

Preguntas

Gracias!