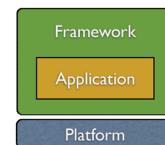


Temas de Hoy



Evoluciones

Frameworks

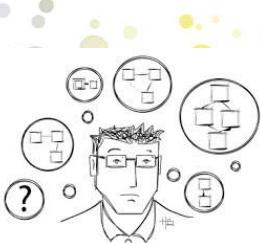


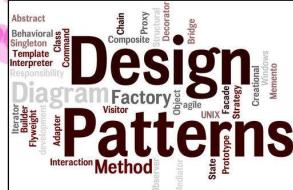
Arquitectura de Aplicaciones Web – 1*C 2018



Retomando...

Diseño de Software



Principios de Diseño

- Descomposición
- Abstracción
- Alta Cohesión
- Bajo Acoplamiento
- Modularidad
- Encapsulamiento

GRASP

- Experto en información
- Creador
- Controlador
- Alta cohesión/bajo acoplamiento
- Polimorfismo
- Fabricación pura
- Indirección

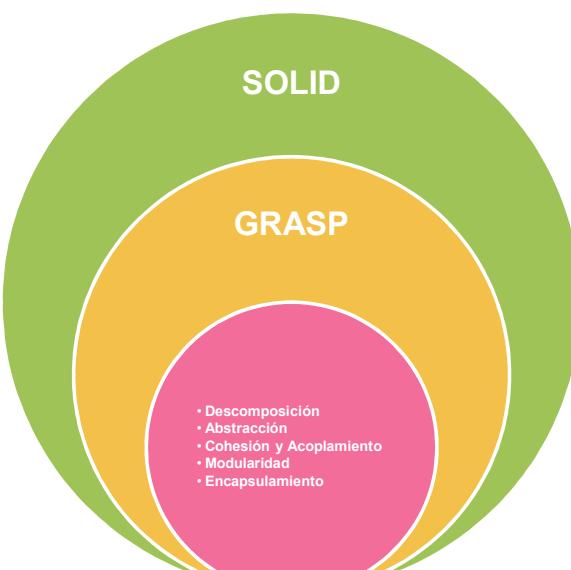
SOLID

- Responsabilidad única.
- Abierto/cerrado
- Sustitución de Liskov
- Segregación de la interfaz
- Inversión de la dependencia

Design Patterns

- De Creación
- Estructurales
- De Comportamiento

¿Cómo diseñar?

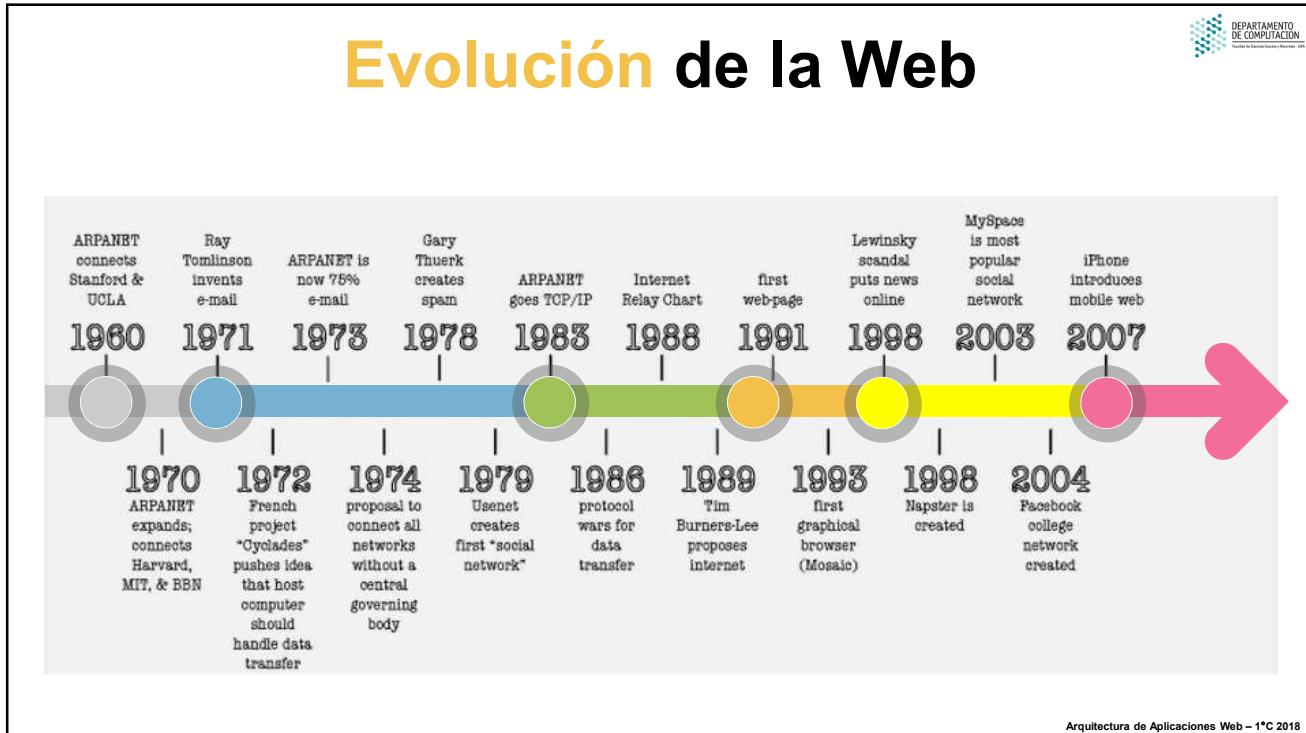


The diagram consists of three nested circles. The outermost circle is green and labeled "SOLID". The middle circle is orange and labeled "GRASP". The innermost circle is pink and contains a bulleted list of principles:

- Descomposición
- Abstracción
- Cohesión y Acoplamiento
- Modularidad
- Encapsulamiento

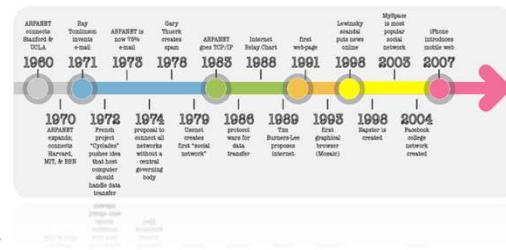
Arquitectura de Aplicaciones Web – 1ºC 2018





Evolución de la Web

- 60-80 Origen militar
 - Protocolos de comunicación (TCP/IP)
 - Seguridad ante ataques (múltiples servidores)
- 80-90 Implantación académica
 - Protocolos de intercambio de información (FTP, SMTP, ...)
- 90-95 World Wide Web
 - HTTP, HTML, etc.
 - Enorme biblioteca con material hipermedia
- 95-00 Acceso comercial
 - Posibilidad de negocio → Dinero!!
 - Boom comercial
- 00-01 Crisis de las punto.com
 - Historias de fracasos → Lecciones aprendidas

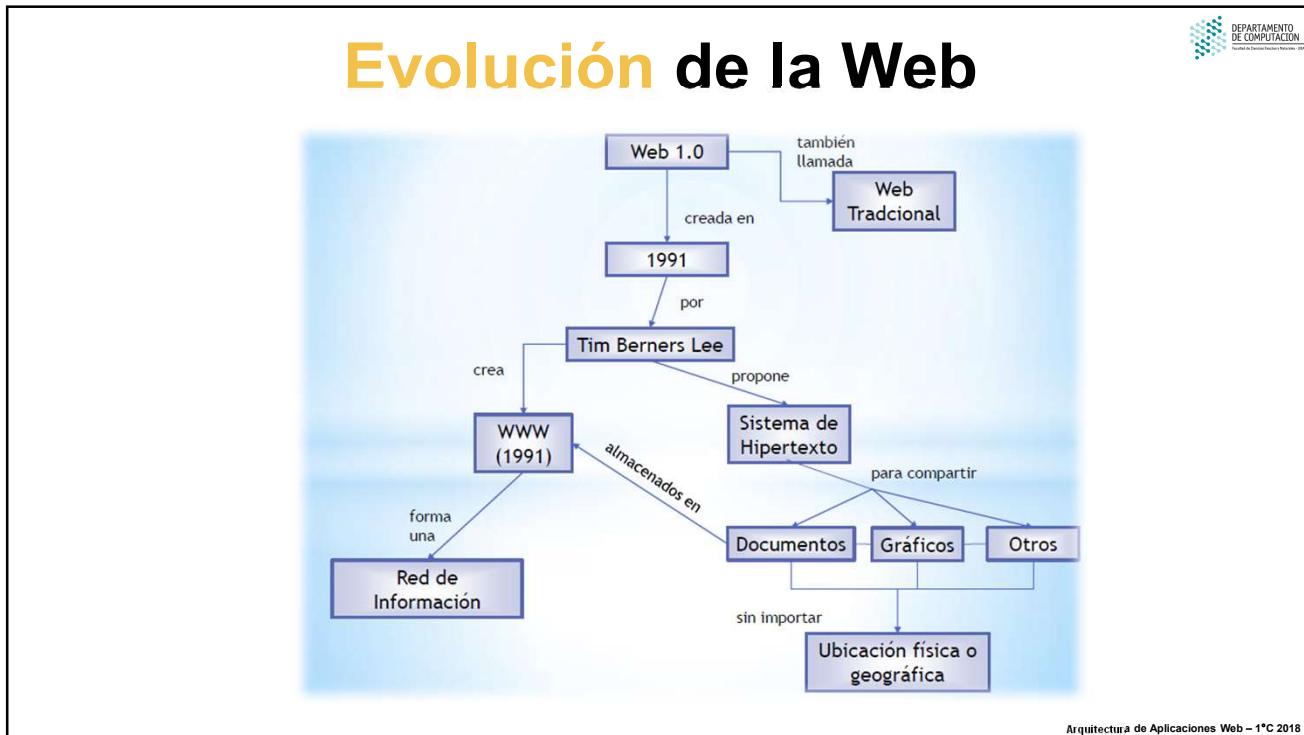
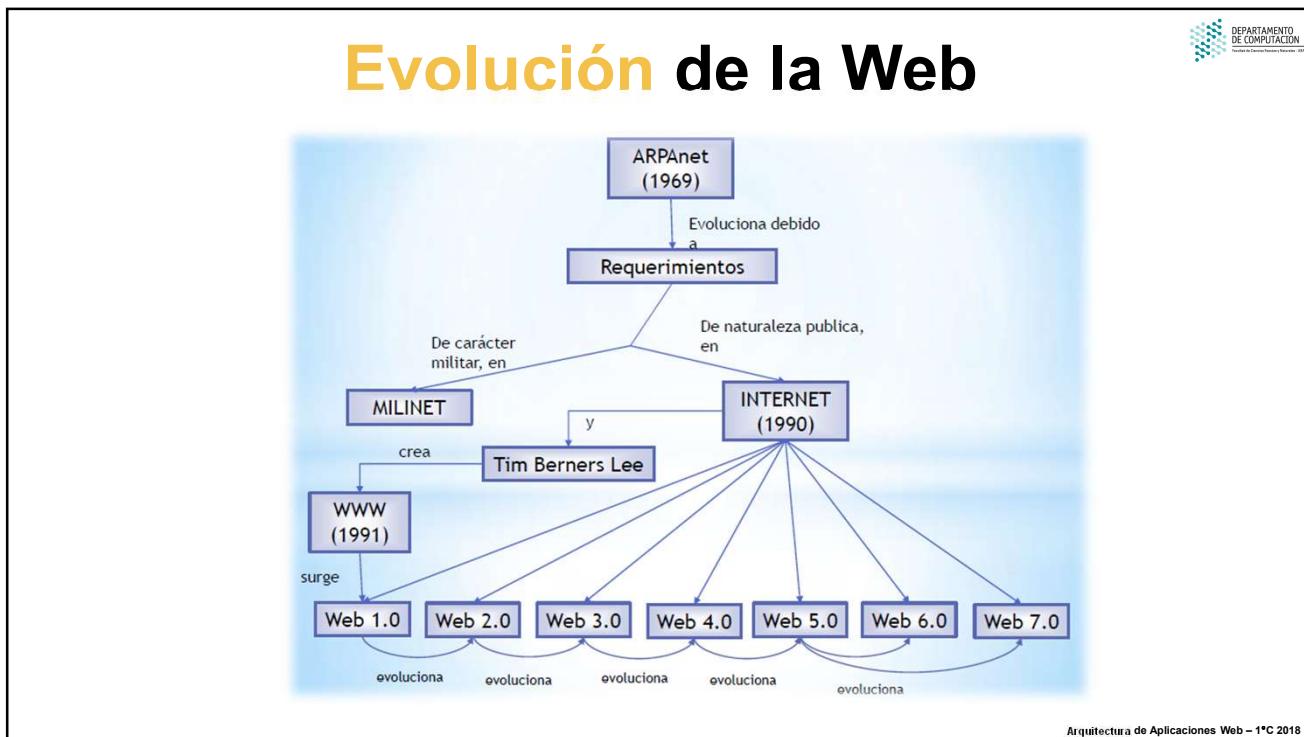


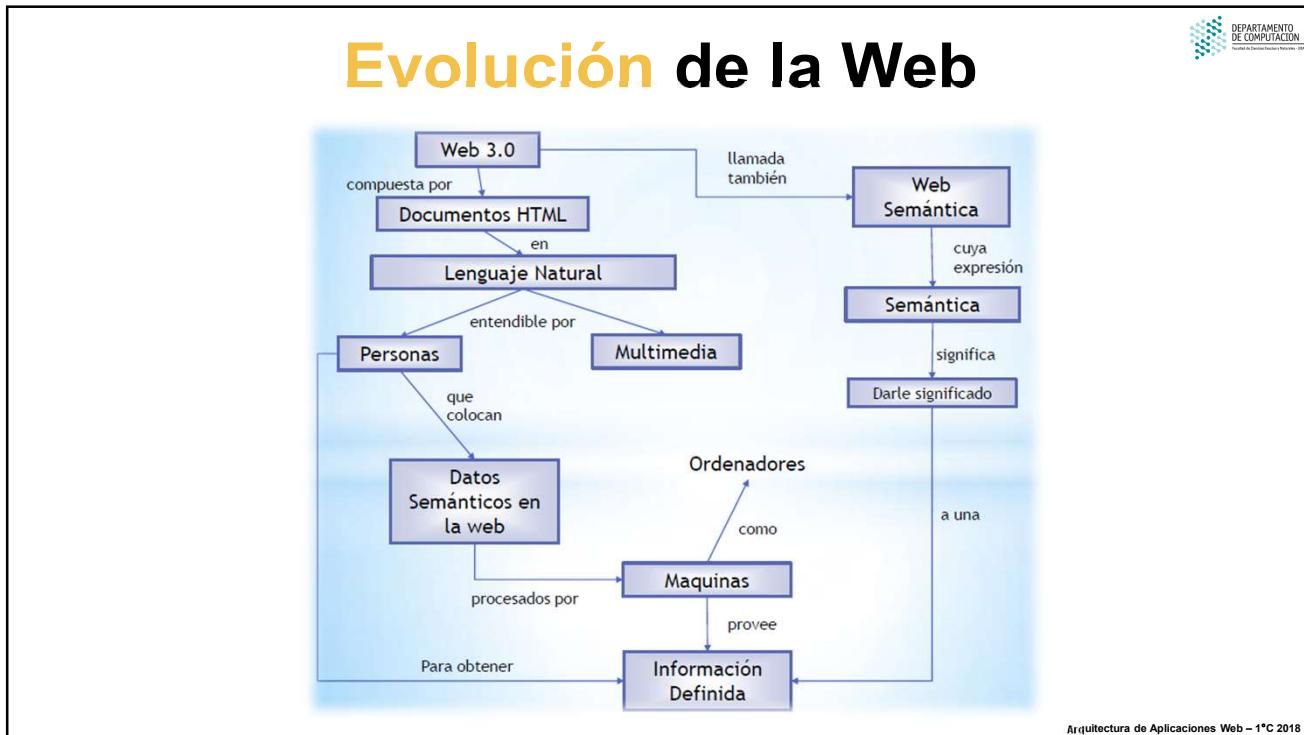
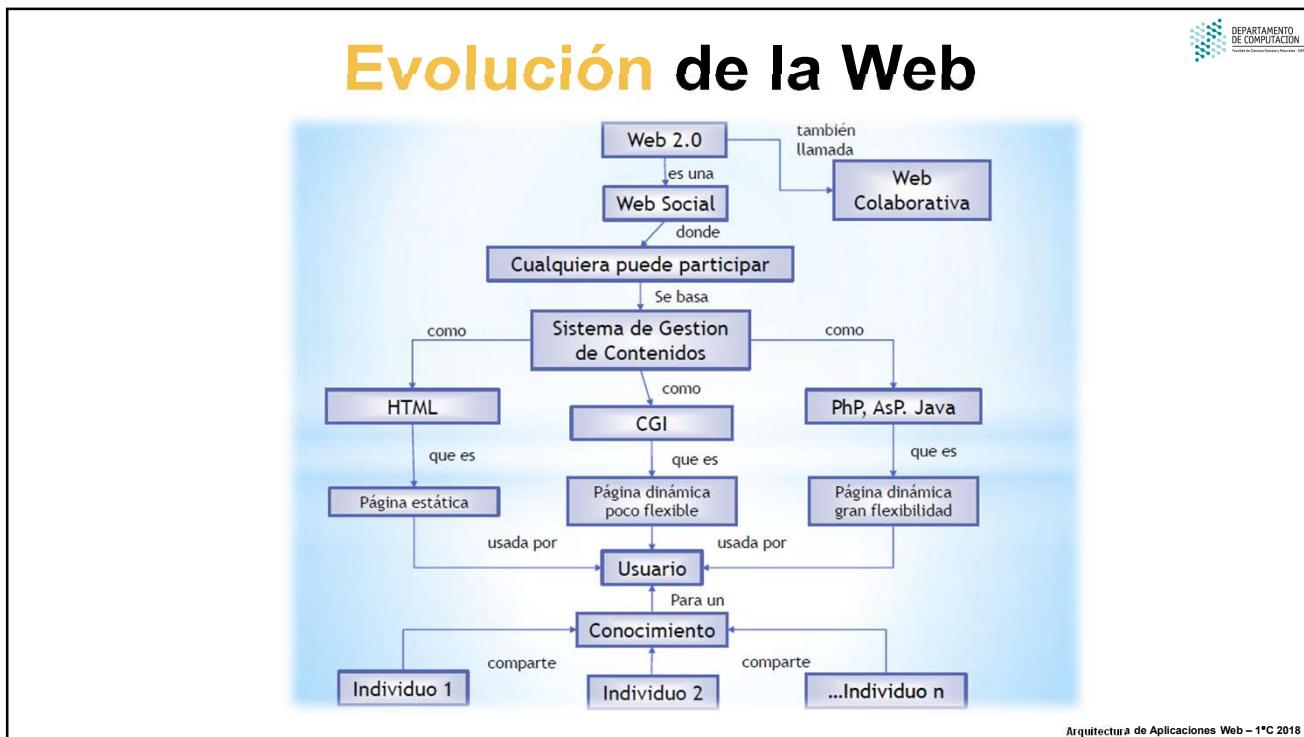
Arquitectura de Aplicaciones Web – 1ºC 2018

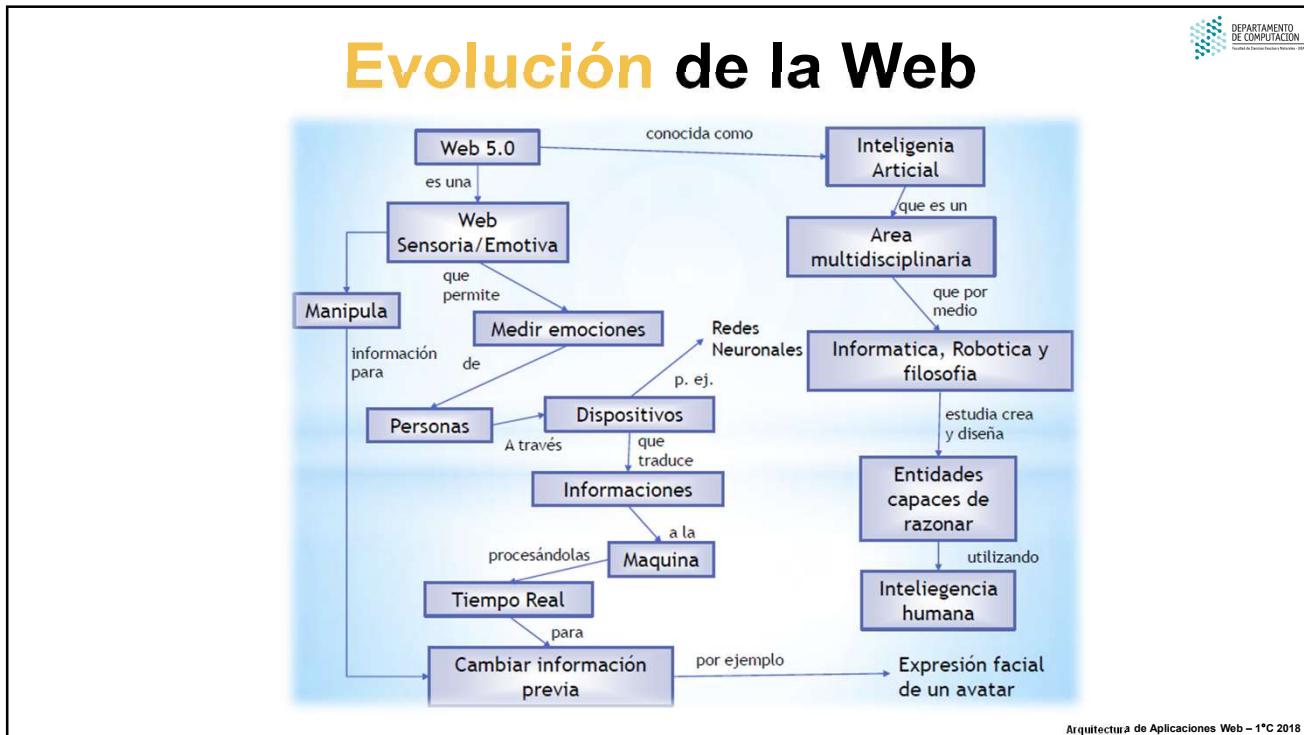
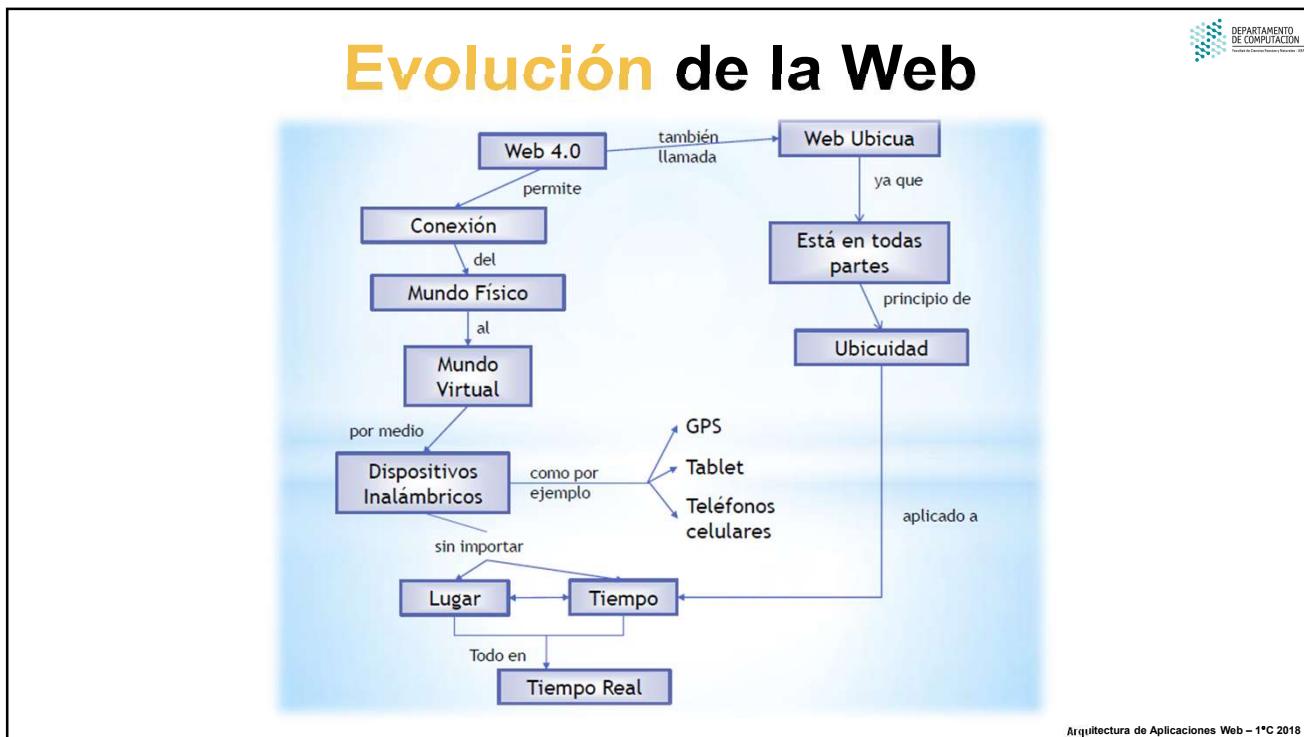
Evolución de la Web

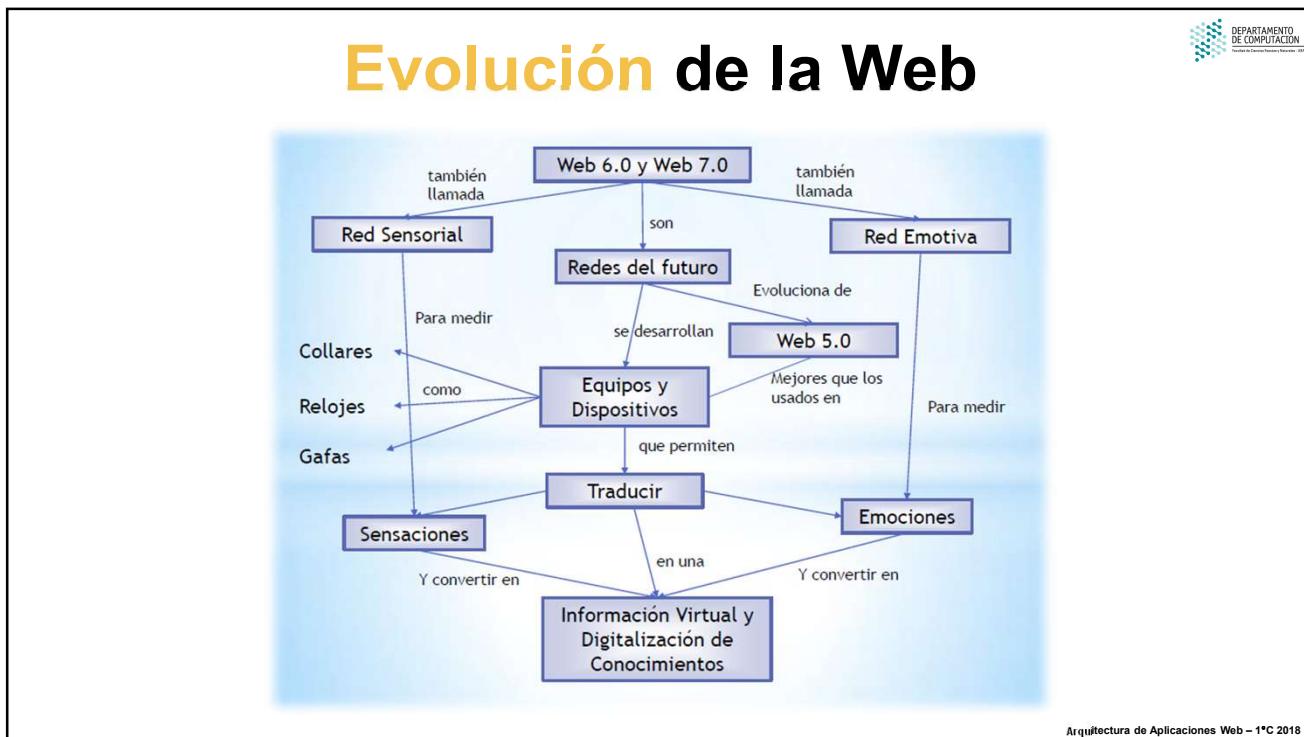


Arquitectura de Aplicaciones Web – 1ºC 2018









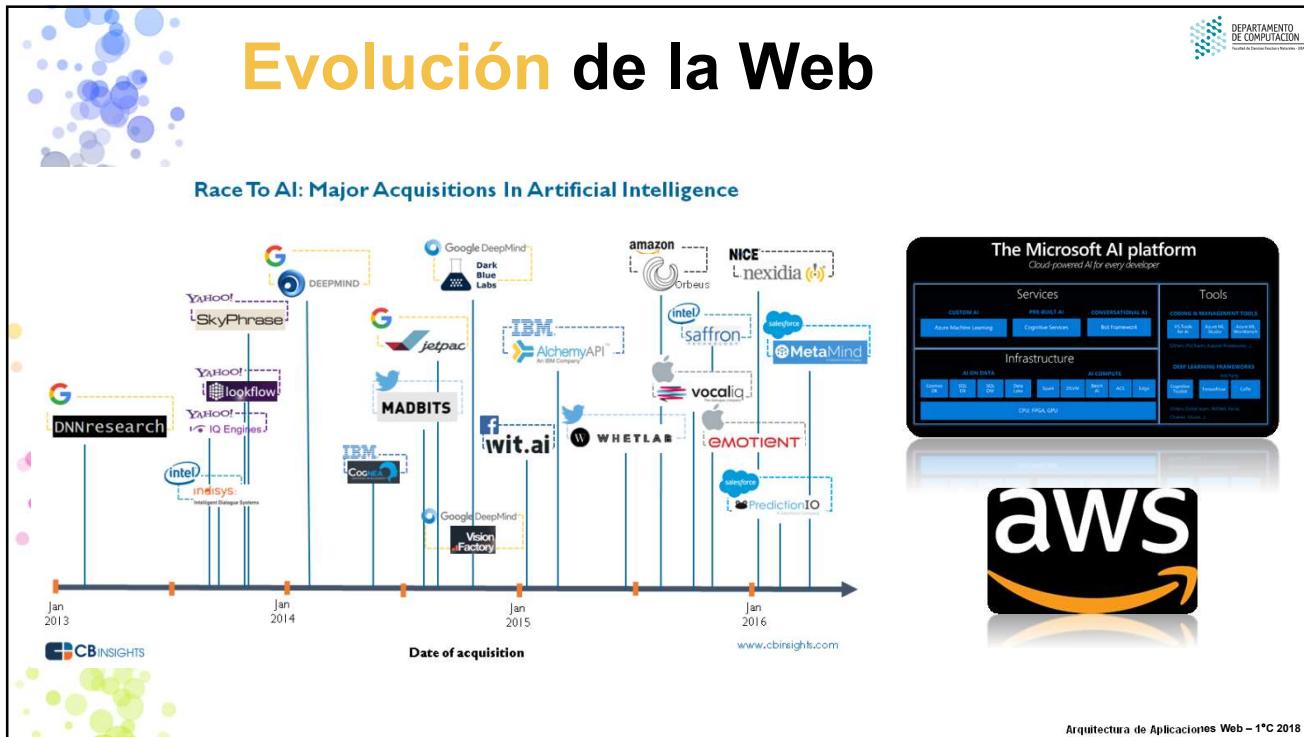
Arquitectura de Aplicaciones Web – 1*C 2018



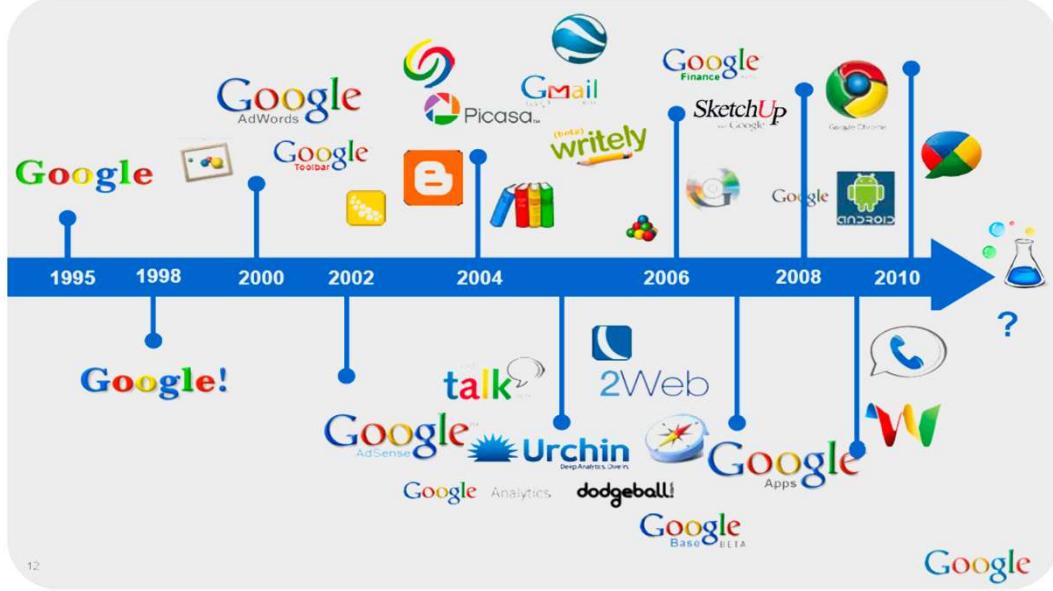
Arquitectura de Aplicaciones Web – 1*C 2018

Evolución de la Web

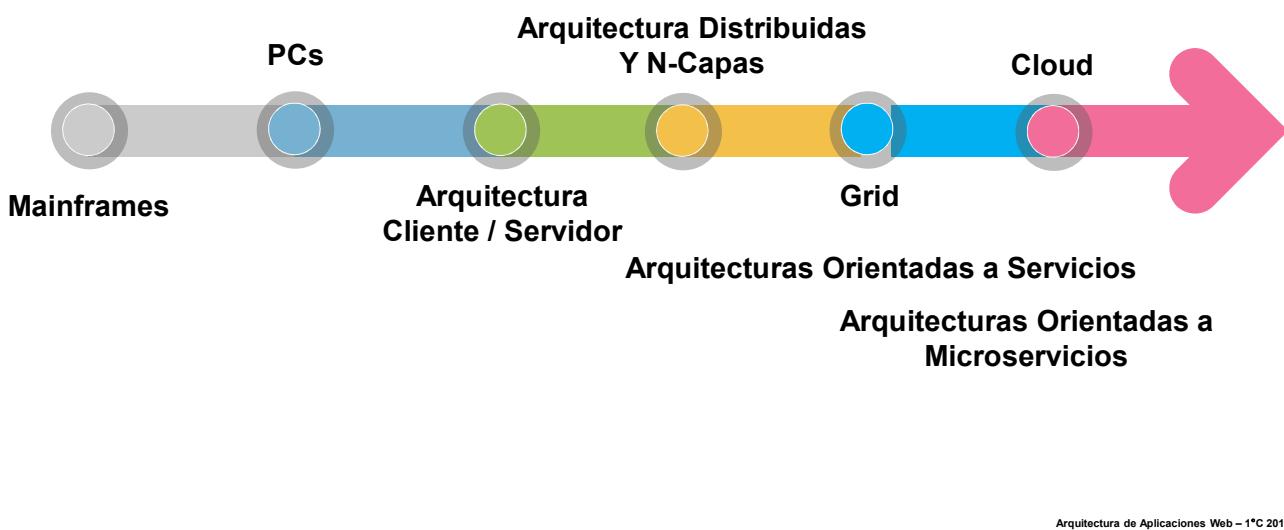
Arquitectura de Aplicaciones Web – 1ºC 2018

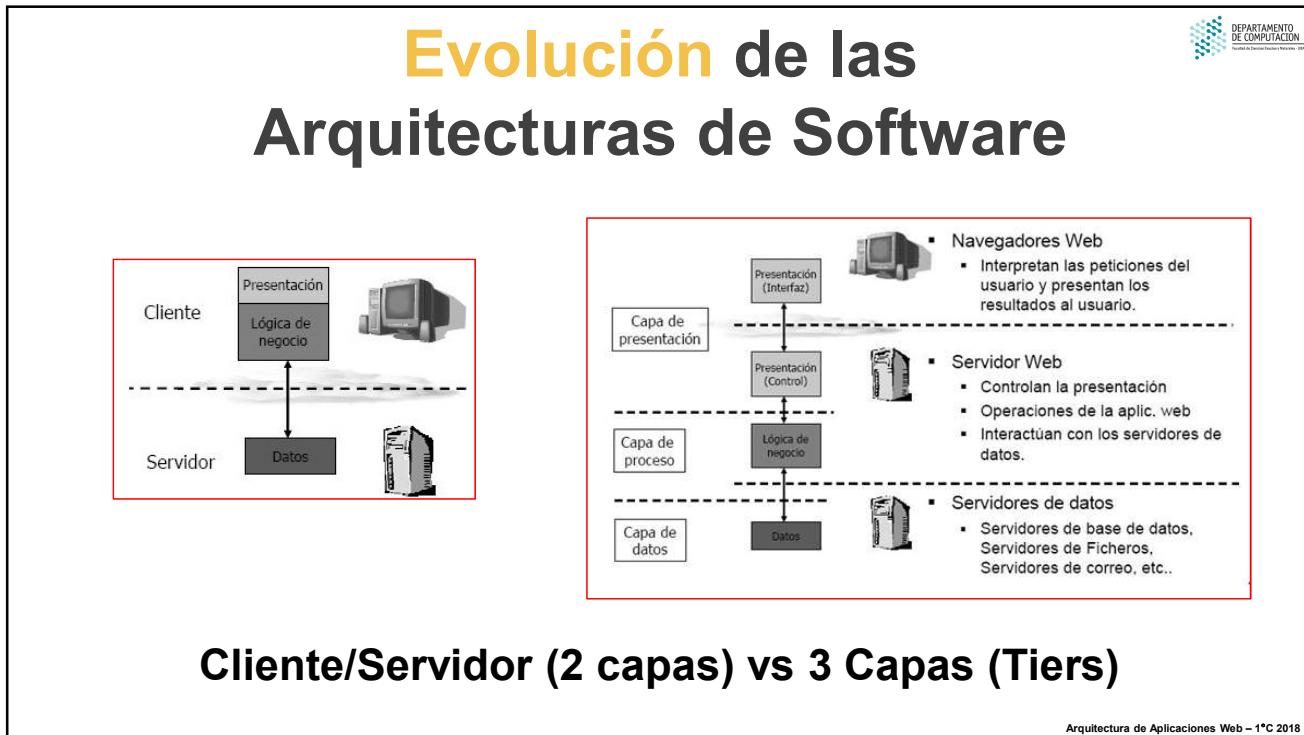
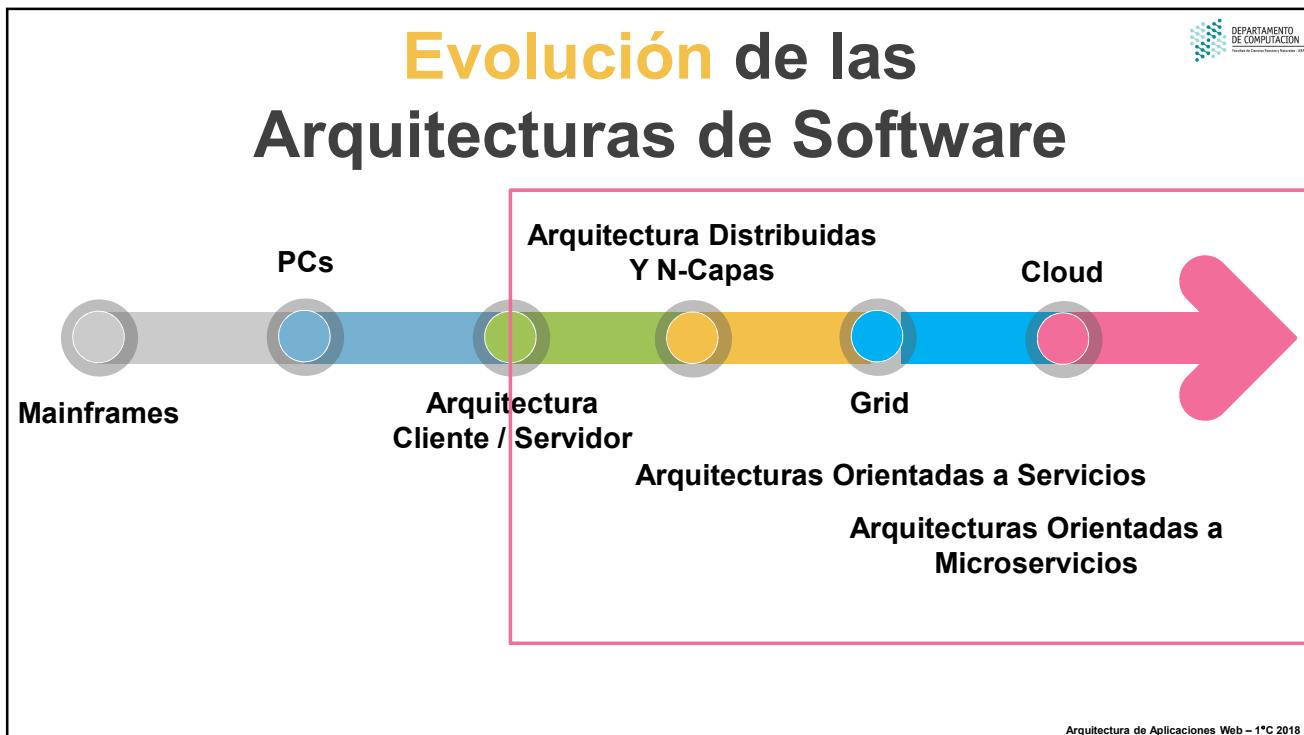


Evolución de la Web



Evolución de las Arquitecturas de Software

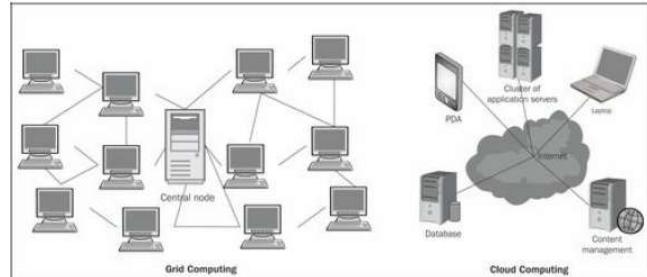




Evolución de las Arquitecturas de Software

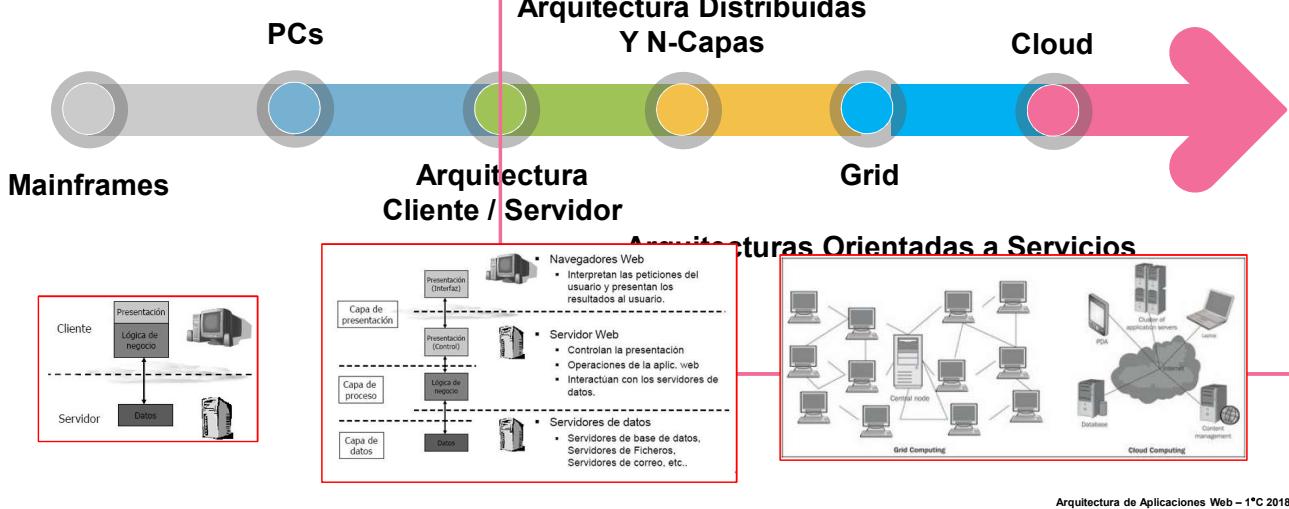
Grid Computing: es un tipo de red diseñado para compartir recursos (computo y almacenamiento) a través de una red.

Cloud Computing: se refiere a una nueva clase de computación basada en tecnologías de redes.

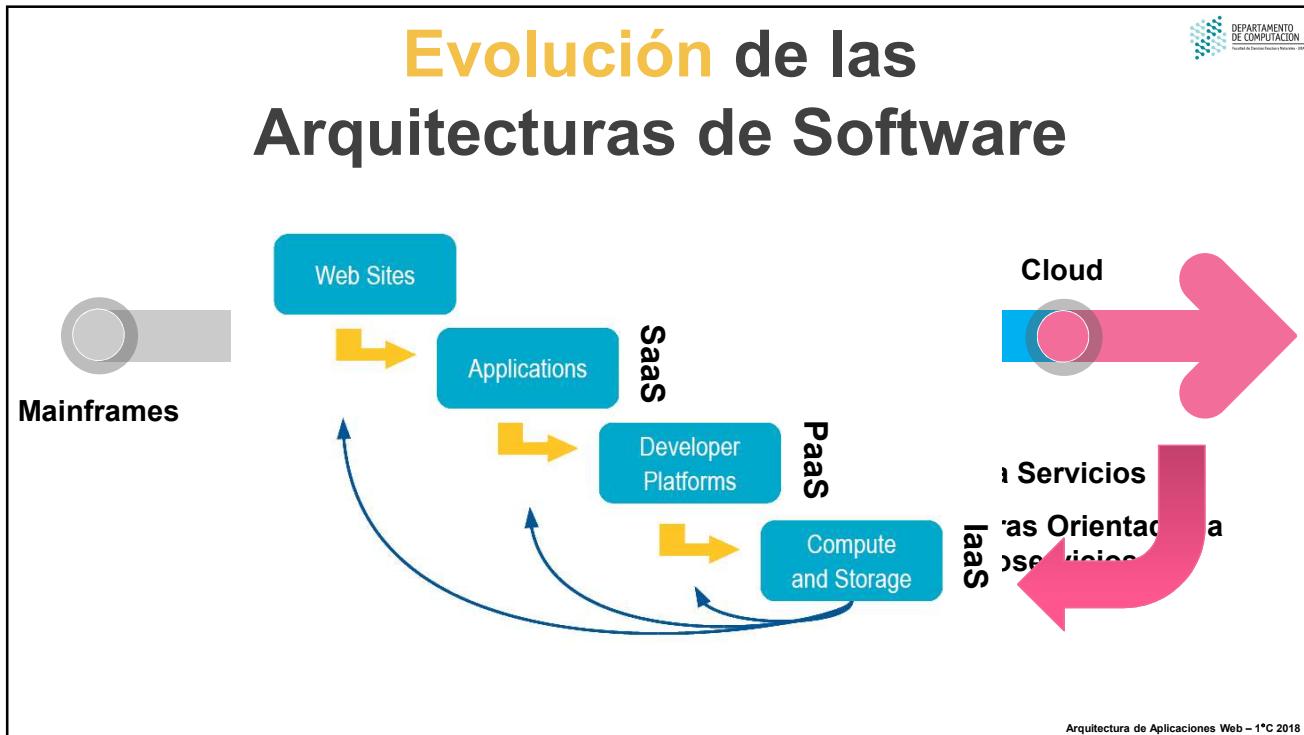
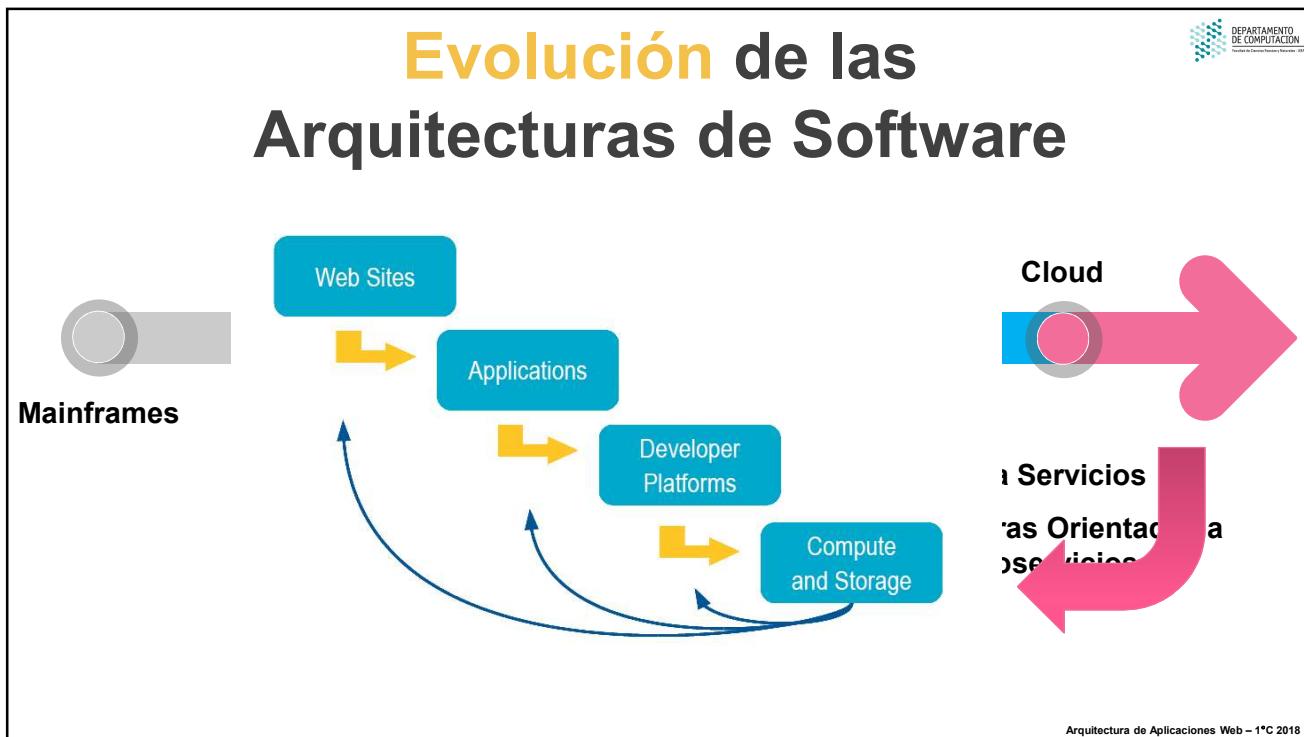


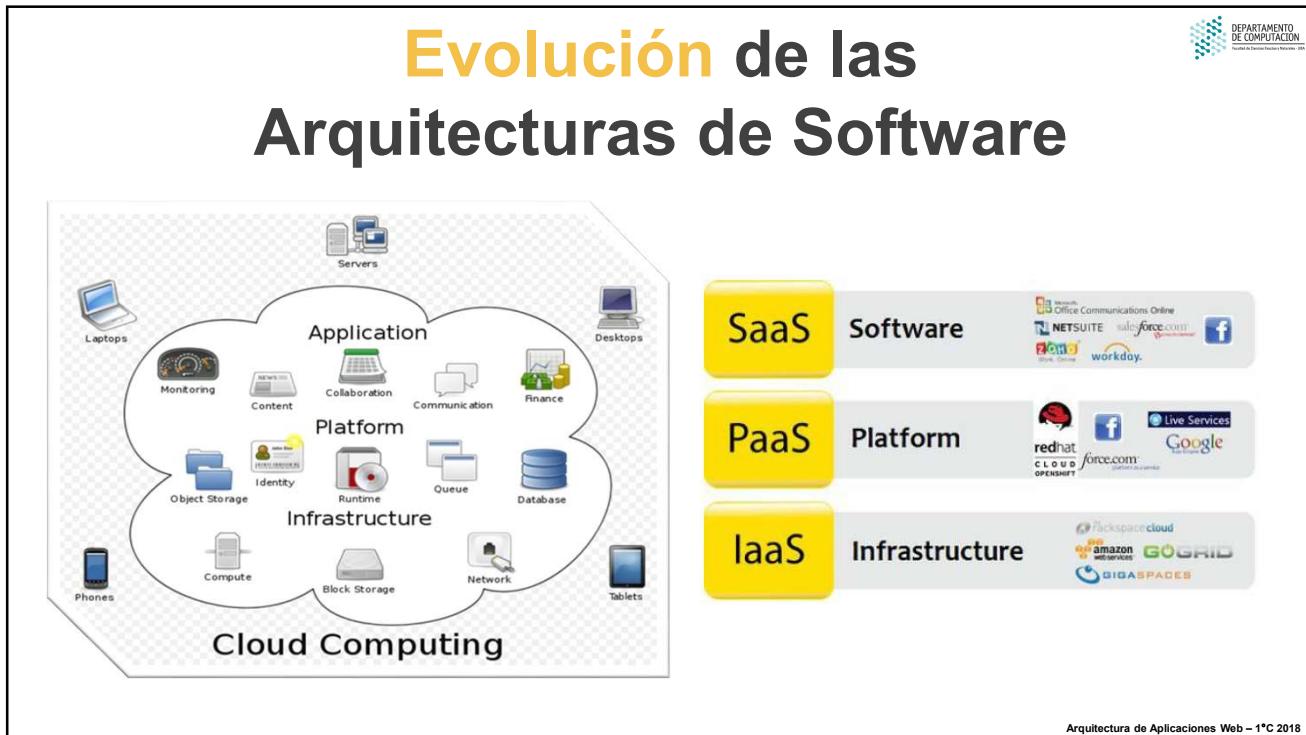
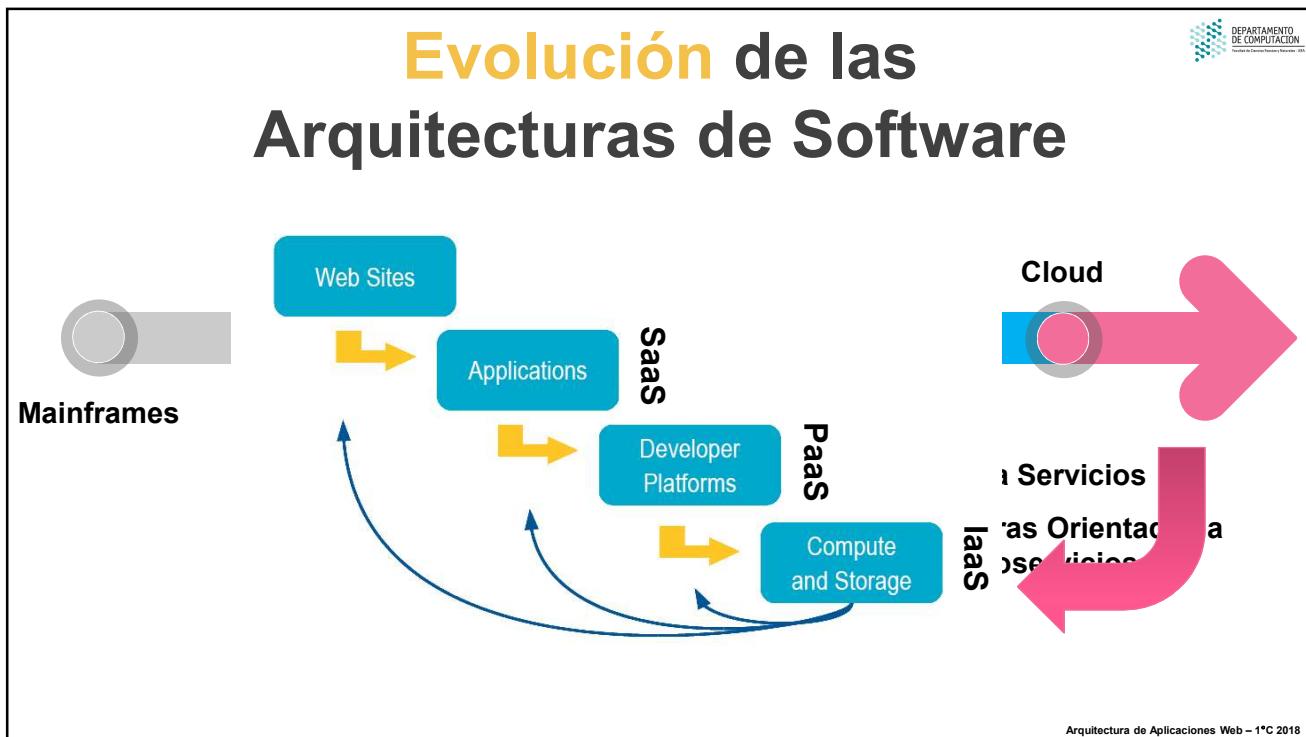
Arquitectura de Aplicaciones Web – 1*C 2018

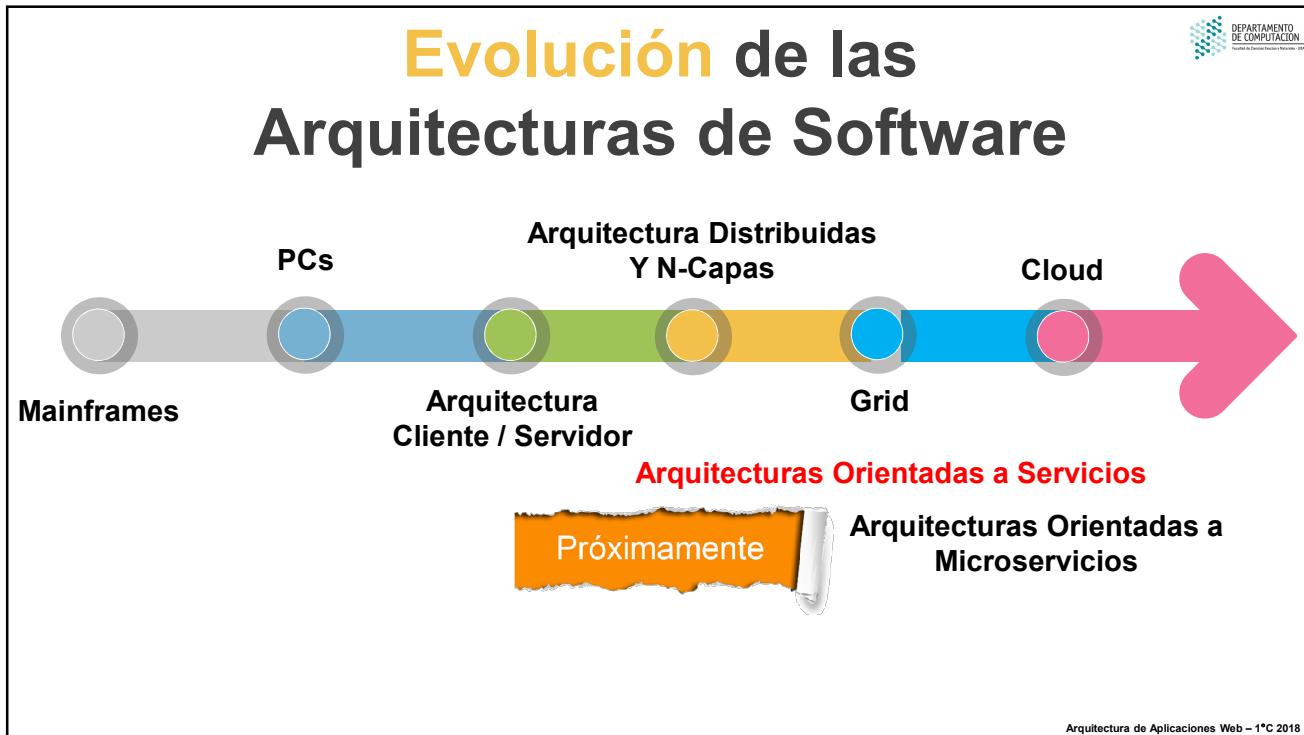
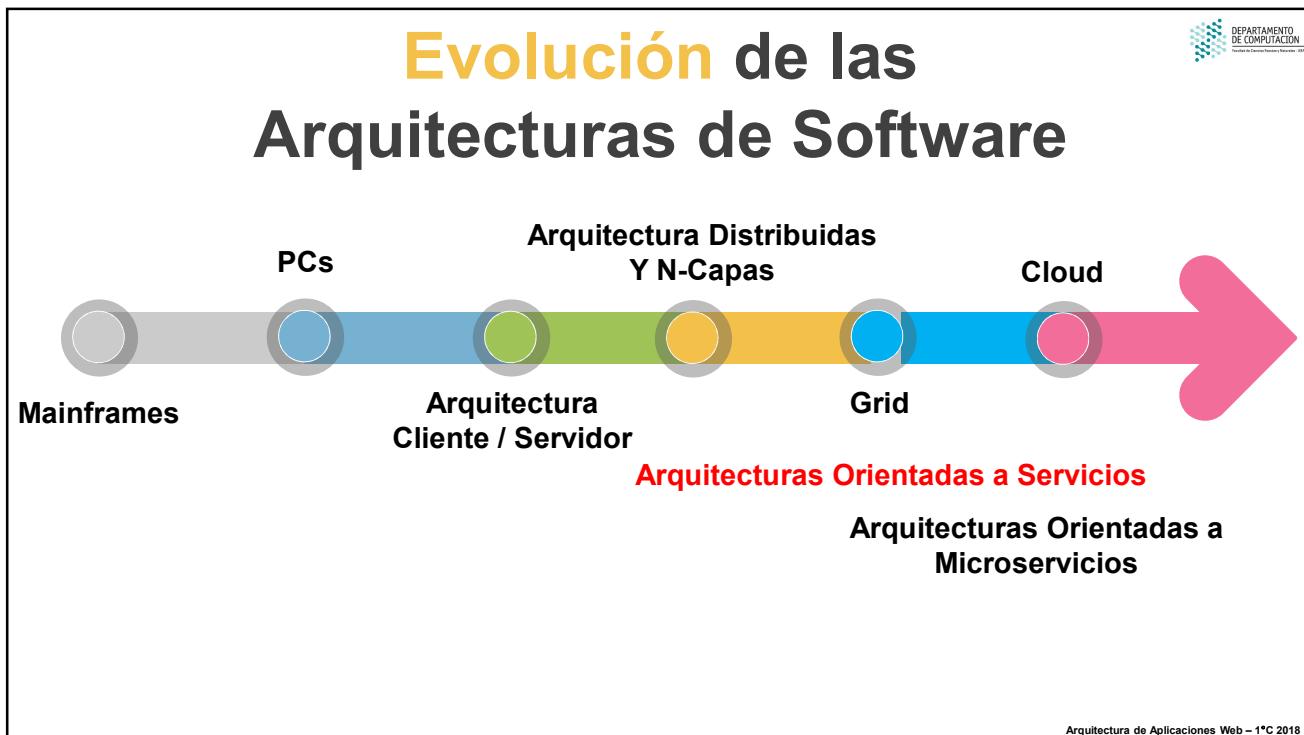
Evolución de las Arquitecturas de Software



Arquitectura de Aplicaciones Web – 1*C 2018





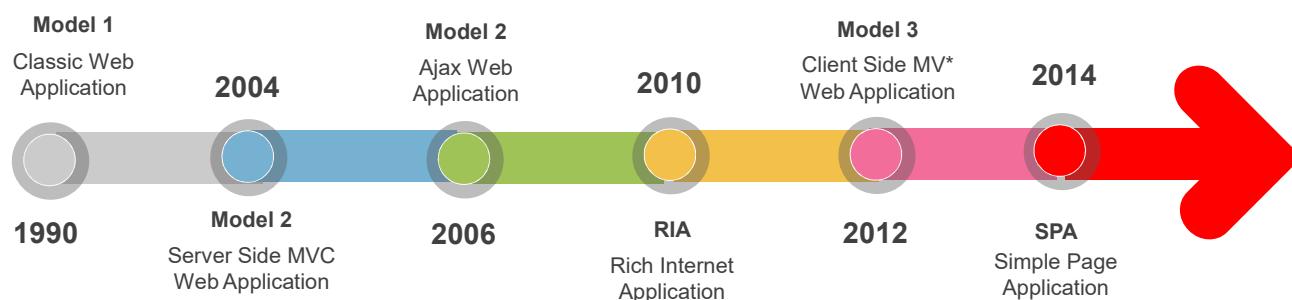




Evolución de Arquitecturas Web



Evolución de las Aplicaciones Web



Arquitectura de Aplicaciones Web – 1ºC 2018

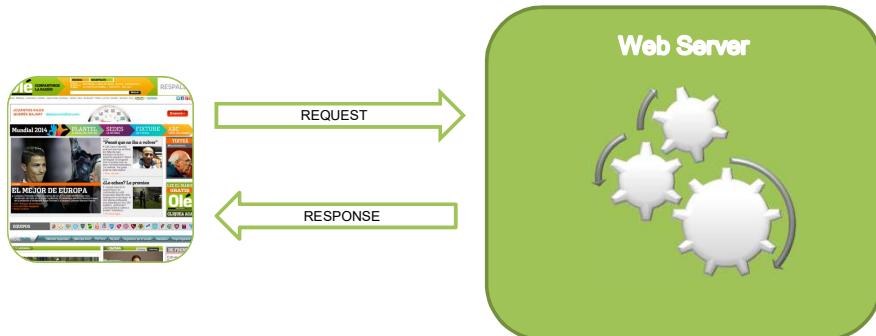
Una posible taxonomía

- Web 1.0 (o 1ra Generación)
 - *Contenido Estático*
- Web 2.0 (o 2da Generación)
 - *Contenido Estático*
 - *Contenido Dinámico*
 - *Servicios*
 - *Herramientas de Colaboración*
- Después vendrían los Portales...
- Después, antes o durante... Web 3.0...

Arquitectura de Aplicaciones Web – 1*C 2018

¿Dónde se genera el contenido?

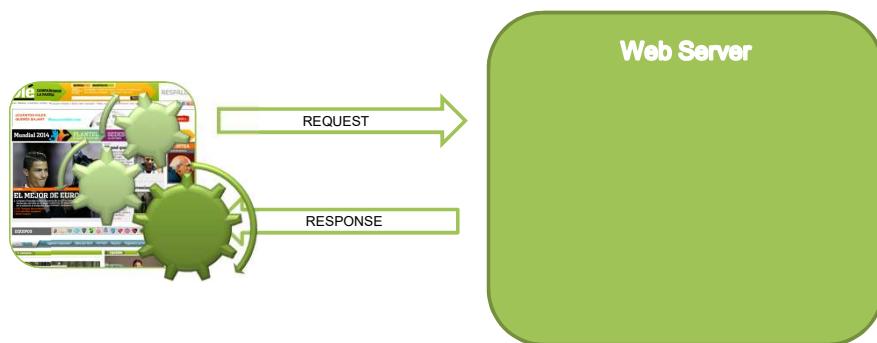
- Server Side
 - JSP, PHP, ASP.NET, ASP.MVC, etc...



Arquitectura de Aplicaciones Web – 1*C 2018

¿Dónde se genera el contenido?

- Client Side
 - DHTML, Javascript, jQuery, AngularJS, etc.



Arquitectura de Aplicaciones Web – 1*C 2018

Modelos arquitectónicos de desarrollo web

- Inicialmente, podríamos decir que existían dos modelos conceptuales en el desarrollo de aplicaciones web
 - *Modelo 1 (Plano)*
 - *Modelo 2 (MVC: Model View Controller)*

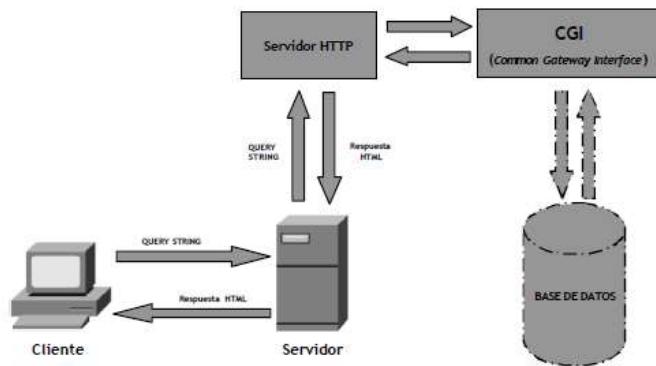
Arquitectura de Aplicaciones Web – 1*C 2018

Modelo 0 - CGI

- **Common Gateway Interface (1996 – NCSA)**
 - Rige como interactúan los servidores HTTP con programas especiales que se ejecutan en la máquina servidora, destinados a procesar la información que envían los clientes Web y generar documentos de forma dinámica.
- A estos programas especiales se les llama **aplicaciones o módulos CGI** (la mayoría de veces, simplemente **CGI**).
 - Son la primera de una serie de soluciones ideadas para publicar información dinámica (extraída normalmente de una BD).
 - El contenido es cambiante con el tiempo, dependiendo del entorno, del usuario, de los datos enviados, etc.

Arquitectura de Aplicaciones Web – 1*C 2018

Modelo 0 - CGI



Arquitectura de Aplicaciones Web – 1*C 2018

Modelo 0 - CGI

Arquitectura de Aplicaciones Web – 1*C 2018

- ❖ El usuario:
 - Rellena un formulario HTML y envía los datos.
 - La acción va asociada a la URL de una aplicación CGI.
- ❖ El servidor Web:
 - Lanza la aplicación CGI como un proceso independiente.
 - Pasa los datos del formulario a través de: variables de entorno, línea de comandos y entrada estándar.
- ❖ La aplicación CGI:
 - Recoge los datos, accede a la BD y, con el resultado, produce un documento HTML.
 - Devuelve al servidor el documento precedido de una cabecera.
 - HTTP (nph o simple), a través de la salida estándar.
- ❖ El servidor Web:
 - Devuelve el resultado, casi directamente, al navegador.

Modelo 0 - CGI

Arquitectura de Aplicaciones Web – 1*C 2018

Datos enviados en la URL y en el cuerpo:

- ❑ El servidor se encarga de hacer llegar la información recibida al CGI:
 - Si el método de envío fue GET
 - Los datos **codificados** son accesible por el CGI a través de la variable de entorno **QUERY_STRING** (texto de la url desde la 1^a '?' hasta el final).
 - Si no se ha utilizado '=' , el servidor también hace llegar los datos **decodificados**, a la aplicación CGI, como argumentos (en C++: **argc, argv**).
 - Si fue POST
 - Llegarán **codificados** por la entrada estándar.
 - En C++: **cin.read**
 - La variable **CONTENT_LENGTH** indicará el número de bytes de la información.



Modelo 0 - CGI

Variables de entorno:

Las variables de entorno son un conjunto de variables que el servidor Web debe poner en el entorno de ejecución de los CGI:

Contienen:

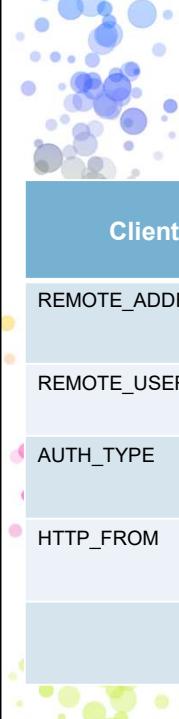
- El resto de información enviada por el cliente (cabecera HTTP del mensaje de petición).
- También incluyen características particulares del entorno (relacionadas con la ejecución del servidor).

La forma de acceder a estas variables depende del lenguaje y del S.O.

- En C++: `getenv(VARIABLE)`
- Las variables se escriben en mayúsculas.
- Un valor nulo equivale a su no definición.

El número de variables depende del servidor.

Arquitectura de Aplicaciones Web – 1*C 2018



Modelo 0 - CGI

Variables de entorno:

Cliente	Servidor	Conexion	Entrada	Localización del CGI
REMOTE_ADDR	SERVER_NAME	HTTP_ACCEPT	REQUEST_METHOD	GATEWAY_INTERFACE
REMOTE_USER	HTTP_HOST	HTTP_REFERER	QUERY_STRING	SCRIPT_NAME
AUTH_TYPE	SERVER_PORT	HTTP_USER_AGENT	CONTENT_TYPE	DOCUMENT_ROOT
HTTP_FROM	SERVER_SOFTWARE	HTTP_COOKIE	CONTENT_LENGTH	PATH_INFO
	SERVER_PROTOCOL	HTTPS		PATH_TRANSLATED

Arquitectura de Aplicaciones Web – 1*C 2018

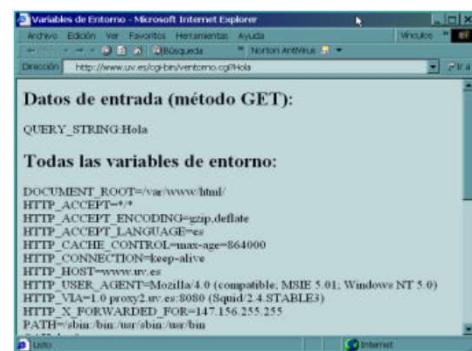
CGI – Web hora y fecha

```
#include <iostream>
#include <ctime>
using namespace std;
int main ()
{
    time_t hora;
    time( &hora ); // Almacena la hora y fecha en la variable
    cout << "Content-Type: text/html" << endl << endl;
    cout << "<html><head>" << endl;
    cout << "<title>Fecha y Hora</title>" << endl;
    cout << "</head><body>" << endl;
    cout << "<p>Fecha y Hora actuales:" ;
    cout << asctime(localtime(&hora)) << "</p>" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```

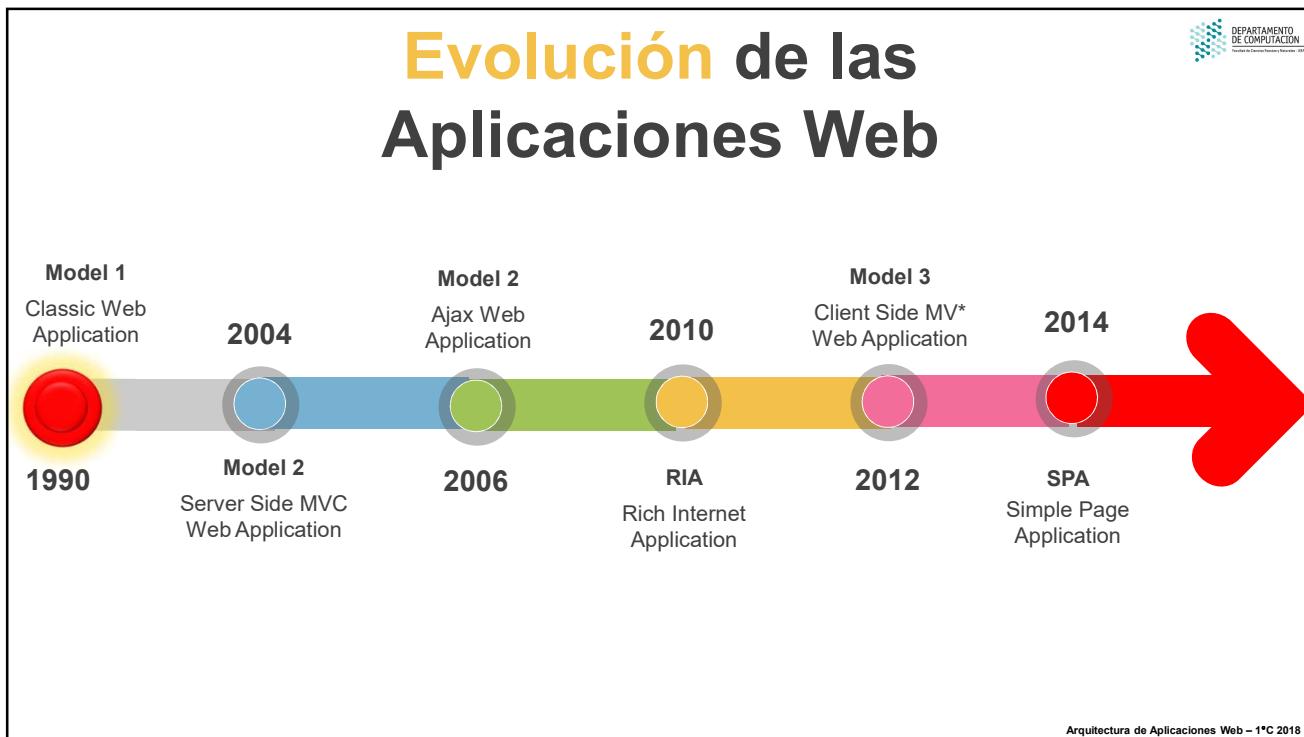
Arquitectura de Aplicaciones Web – 1*C 2018

CGI – Web hora y fecha

```
#include <string>
#include <iostream>
#include <cstdlib>
using namespace std;
int main ()
{
    string vEntorno[24] = { "DOCUMENT_ROOT", "HTTP_ACCEPT",
        "HTTP_ACCEPT_ENCODING", "HTTP_ACCEPT_LANGUAGE", ... };
    cout << "Content-Type: text/html" << endl << endl;
    cout << "<html><head><title>Variables de Entorno</title>" ;
    cout << "</head><body><h2>Datos de entrada (método GET) :</h2>" ;
    cout << "QUERY_STRING:" << getenv("QUERY_STRING") << "<br>" ;
    cout << "<h2>Todas las variables de entorno</h2>" ;
    for ( i = 0; i < 24 ; i++ ) {
        cout << vEntorno[i] << ":" ;
        cout << getenv(vEntorno[i].data()) << "<br>" ;
    }
    cout << "</body></html>" ;
}
```



Arquitectura de Aplicaciones Web – 1*C 2018

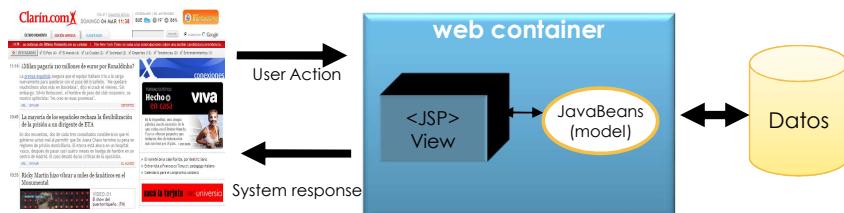


Modelo 1 (Plano)

- A diferencia de los CGIs, el contenido dinámico es generado utilizando un lenguaje propio, embebido dentro de código HTML.
 - Las peticiones son procesadas por el mismo servidor
 - El script que genera el código dinámicamente es ejecutado por el propio servidor HTTP.
 - De ello se encarga un módulo incluido en el mismo proceso.
- Toda la responsabilidad reside en las páginas que generarán el HTML dinámicamente (JSP, ASP, PHP)
 - Reciben el request del cliente
 - Manejan todo el procesamiento del request
 - Se comunican con objetos / capas de negocio.
 - Muestran el output al cliente
 - Seleccionan la próxima página para el cliente

Arquitectura de Aplicaciones Web – 1*C 2018

Modelo 1 (Plano) – Servlet/JSP



Arquitectura de Aplicaciones Web – 1*C 2018

Modelo 1 – Ej: Servlets

- ❑ Los **Java Servlets** surgieron como una alternativa a la programación CGI.
- ❑ Los servlets son programas Java que corren en un servidor web y permiten la construcción de páginas dinámicamente.
- ❑ Son independientes de la plataforma.
- ❑ No deben ser confundidos con los **Java Applets**, que corren en el lado del cliente.

Arquitectura de Aplicaciones Web – 1*C 2018

Servlets

- Las ventajas de los **Java Servlets** sobre CGI, incluyen:
 1. *Eficiencia*: en CGI, un nuevo proceso se inicia ante cada pedido; los servlets se inician una única vez y luego cada nuevo pedido es atendido por un nuevo thread.
 2. *Potencial*: los servlets pueden manejar sesiones y conservar información sobre el usuario.
 3. *Seguridad y portabilidad*: debido a que se ejecutan bajo la máquina virtual de Java, al mecanismo de excepciones y al uso del administrador de seguridad de java.

Arquitectura de Aplicaciones Web – 1*C 2018

Hello World Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

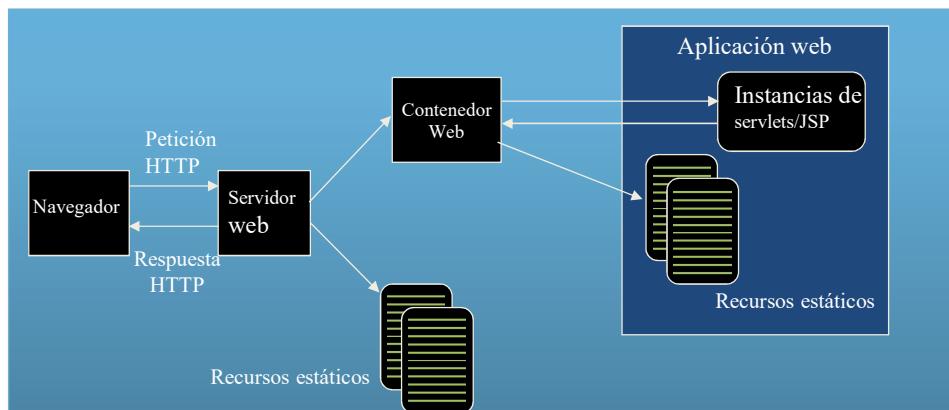
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html> <head> </head> " +
                   "<body> <h1> Hello World </h1> </body> </html>");
        out.close();
    }
}
```

Arquitectura de Aplicaciones Web – 1*C 2018

Servlets

- Los servlets no se pueden invocar directamente por el usuario
- La interacción se lleva a cabo a través del contenedor o motor de servlets en el que la aplicación está desplegada:
 1. Invoca a los servlets
 2. Intercambia con ellos la información de entrada para que pueda analizarla y generar la respuesta

Servlets



Servlets

- El **ciclo de vida de un servlet** se desarrolla con los siguientes métodos:
 1. **Init:** invocado en la creación
 2. **Service:** invocado en los requerimientos subsiguientes
 3. **doGet y doPost:** realizan el *verdadero* procesamiento
 4. **Destroy:** invocado al realizar el unload del servlet

Arquitectura de Aplicaciones Web – 1ºC 2018

Servlets - Init

- El método **init** se llama una única vez cuando se crea el servlet.
- Al registrarse, puede configurarse si ello ocurrirá al invocar el servlet por primera vez o al iniciar el servidor.
- Hay dos versiones de este método:
 1. **Init():** se utiliza cuando no se necesitan configuraciones específicas del servidor.

```
public void init() throws ServletException {
    // Initialization code...
}
```
 2. **Init(ServletConfig config):** se utiliza para brindar configuraciones específicas del servidor. La clase ServletConfig dispone de un método getInitParameter.

```
public void init(ServletConfig config)
throws ServletException {
super.init(config);
// Initialization code...
}
```

Arquitectura de Aplicaciones Web – 1ºC 2018

Servlets - Service

- Cada vez que se invoca un servlet, se genera un nuevo thread y se invoca al método **service**.
- De acuerdo al tipo de requerimiento HTTP (Get, Post, Delete) se invoca al doGet, doPost, doDelete.
- Si el comportamiento es idéntico para diferentes requerimientos, puede sobreescibirse. No es recomendable: dificulta agregar nuevos servicios.
- Si el comportamiento del doGet y el doPost es idéntico, utilizar llamados entre ellos.

```
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// Servlet Code
}
public void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
doGet(request, response);
}
```

Arquitectura de Aplicaciones Web – 1*C 2018

Servlets - doXXX

- Realizan el verdadero procesamiento, de acuerdo a cada tipo de requerimiento HTTP.
- En general se sobreesciben el **doGet** y el **doPost**, pero también pueden codificarse el doDelete, doPut, doOptions y doTrace.
- El requerimiento Get es el requerimiento del browser más usual para páginas web.
- Todo servlet debe sobreescibir los métodos doGet o doPost. Si el procesamiento es idéntico para ambos requerimientos, uno puede llamar al otro (o viceversa).
- Tienen dos argumentos:
 1. HttpServletRequest: encapsula la información de entrada, como formularios, encabezamientos HTTP, hostname del cliente
 2. HttpServletResponse: permite indicar la información de salida, como el status HTTP, encabezamientos y el documento devuelto al cliente.

Arquitectura de Aplicaciones Web – 1*C 2018

Servlets - doXXX



```
public class ServletTemplate extends HttpServlet {  
  
    public void doGet(HttpServletRequest request,  
                      HttpServletResponse response)  
        throws ServletException, IOException {  
  
        // Use "request" to read incoming HTTP headers  
        // (e.g. cookies) and HTML form data (e.g. data the user  
        // entered and submitted).  
        // Use "response" to specify the HTTP response status  
        // code and headers (e.g. the content type, cookies).  
        PrintWriter out = response.getWriter();  
        // Use "out" to send content to browser  
  
    }  
}
```

Arquitectura de Aplicaciones Web – 1*C 2018

Servlets - destroy



- Un server podría remover un instancia de un servlet cargada anteriormente.
- En ese momento, se llama al método `destroy` del servlet.
- Permite realizar tareas como cerrar conexiones a una base de datos o registrar los números de hits.
- Hay que tener en cuenta que puede haber fallas; no confiar en este método como el único método para realizar persistencia.

Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages

- Java Servlets son programas Java puros y pueden ser incómodos de usar: los programas consisten en su mayoría de sentencias para escribir Html (con algunas instrucciones para otorgarle el comportamiento dinámico).
- Es complicado escribir servlets que produzcan páginas web con estilo atractivo.
- **Java Server Pages** (JSP) es una extensión de la tecnología Java Servlet.
- JSP es un híbrido de HTML y Java Servlets: permite la utilización de estándar HTML estático mezclado con contenido dinámico.

Arquitectura de Aplicaciones Web – 1ºC 2018

Java Server Pages

- JSP son documentos de texto consistentes de dos componentes:
 1. **Contenido estático**: provisto por HTML o XML.
 2. **Contenido dinámico**: generado por JSP tags y scriplets escritos en Java para encapsular la lógica de la aplicación.
- El código Java se delimita con **tags** especiales como:
 - `<% .. %>`, `<jsp:useBean>`

Arquitectura de Aplicaciones Web – 1ºC 2018

Java Server Pages



```
<HTML>
  <BODY>
    Hello!
    The time is now <%= new java.util.Date()%>
  </BODY>
</HTML>
```

Arquitectura de Aplicaciones Web – 1*C 2018

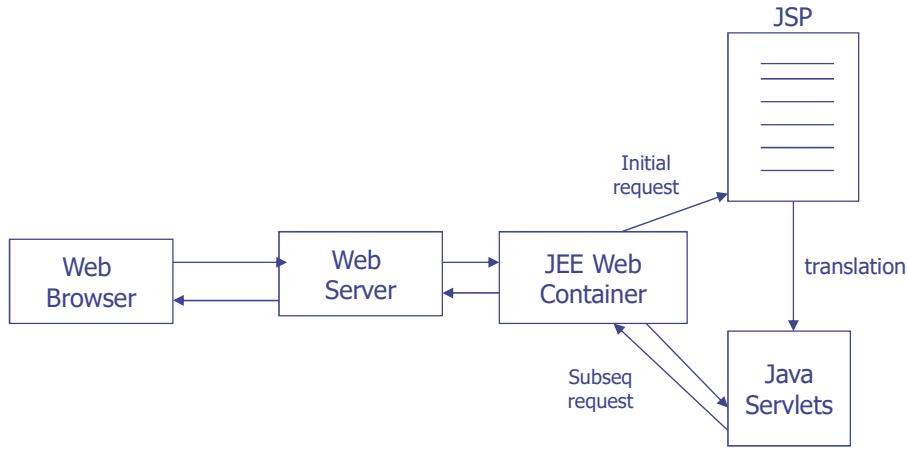
Java Server Pages



- JSP evita tener que escribir servlets consistentes en su mayoría de sentencias para escribir HTML.
- Las páginas JSP son convertidas automáticamente en servlets en el primer acceso y luego almacenadas como servlets en el server.

Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages



Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages

- Para utilizar correctamente JSP, es necesario comprender y utilizar **Java Beans**
- No es considerado una buena práctica escribir demasiado código Java en una página JSP
- En reemplazo, es posible utilizar beans en conjunción con tags especiales, como el `jsp:usebean` o `jsp:getProperty`

Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages

- Los elementos de scripting permiten insertar código Java en el servlet que será generado a partir del JSP:
 1. Expresiones de la forma `<%= expresion Java %>`: evaluadas en tiempo de ejecución, convertidas a string e insertadas en la página.
 2. Scriptlets de la forma `<% código %>`: que permiten insertar código más complejo en el servlet generado.
 3. Declaraciones de la forma `<%! Código %>`: que permiten definir métodos y variables en el cuerpo principal de la clase generada.

Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages

- Las directivas influyen en la estructura del servlet generado:
 1. Page directive `<%@ page directiva %>`: permite importar clases, setear el tipo de contenido, etc.
 2. Include directive `<%@ include directiva %>`: permite insertar un archivo en el servlet (durante su generación).
 3. Taglib directive `<%@ taglib directiva %>`: utilizado para definir custom tags.

Arquitectura de Aplicaciones Web – 1*C 2018

Java Server Pages

- Los tags JSP se utilizan para simplificar la escritura y acceder a diferentes estructuras:
 1. **<jsp:include código />**: para incluir código HTML estático.
 2. **<jsp:useBean código />**: para cargar un bean para utilizar en la página JSP.
 3. **<jsp:getProperty código />**: para obtener el valor de la propiedad de un bean.
 4. **<jsp:setProperty código />**: para setear el valor de la propiedad de un bean.

Arquitectura de Aplicaciones Web – 1*C 2018

JSP vs. ASP (original)

- **JSP:**
 1. Basado en tecnología **Java**
 2. Independiente de plataforma
- **Active Server Pages (ASP):**
 1. Lenguaje de programación de **Microsoft**

Arquitectura de Aplicaciones Web – 1*C 2018

JSP vs. ASP

- **Puntos en común:**

1. Páginas web generadas dinámicamente en el servidor
2. Comparten sintaxis similar en algunos tags, como <% %>
3. Programación modular (ActiveX y JavaBeans)
4. Focalizado en manejo de base de datos

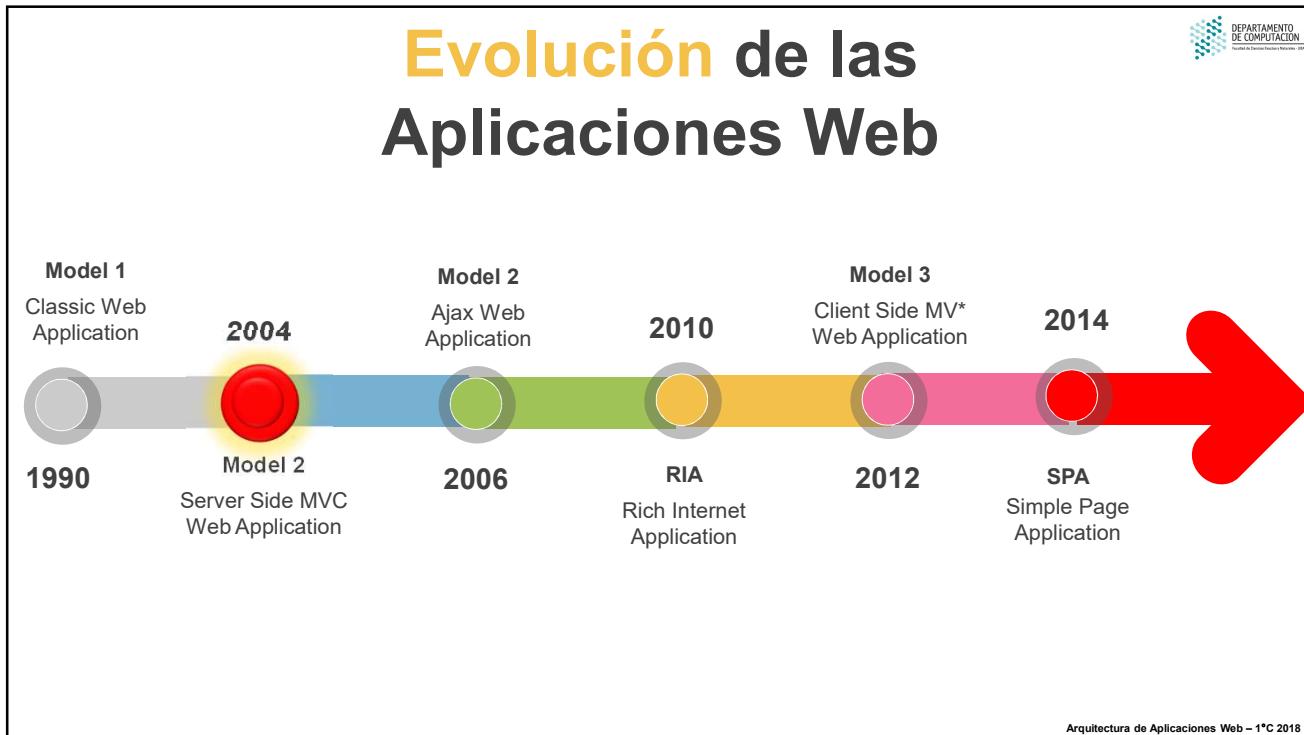
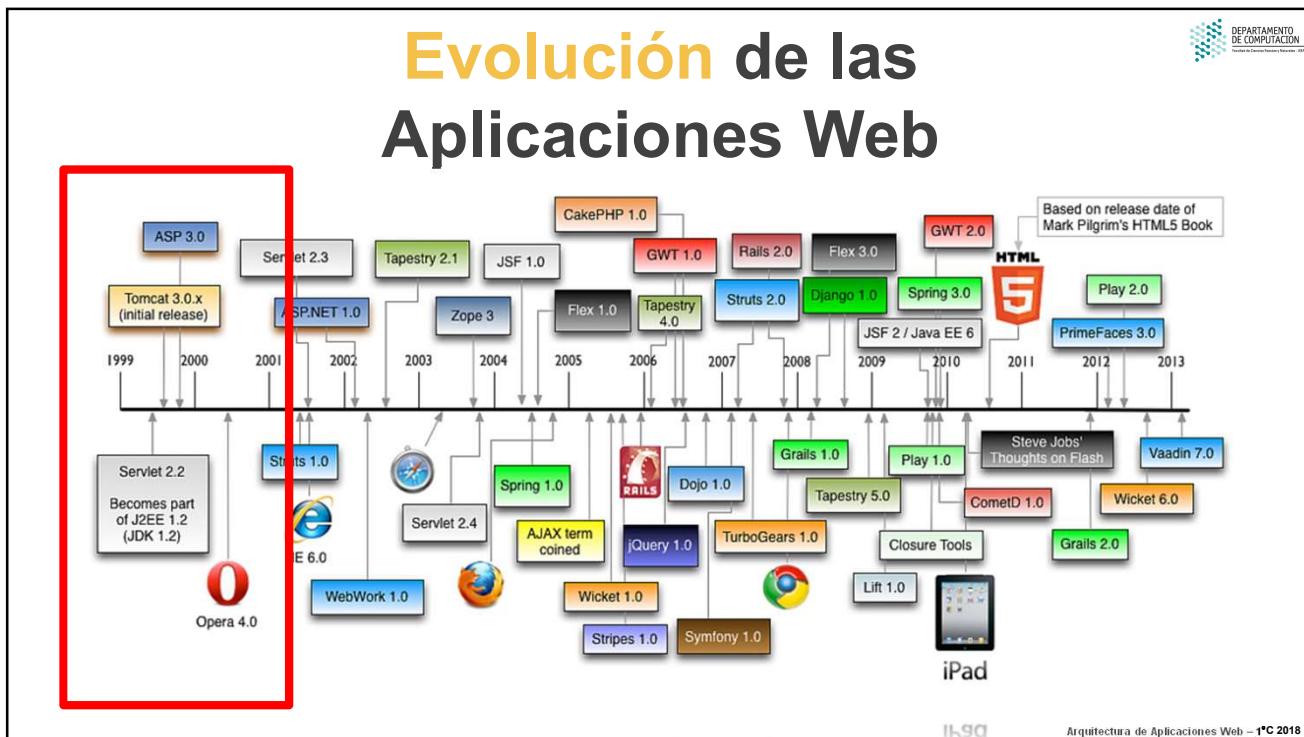
•

Diferencias:

1. ASP es un producto mientras que JSP es una especificación

JSP vs. Servlets

- Las páginas JSP son convertidas a Servlets: ¿son similares?
- **Puntos en común:**
 1. Proveen el mismo resultado a los usuarios finales.
 2. JSP es un módulo adicional de la Servlet Engine
- Servlets: “**HTML en código Java**”
 1. Código HTML inaccesible para diseñadores gráficos
 2. Demasiado acceso a los programadores
- JSP: “**Código Java (Scriptlets) en HTML**”
 1. Código HTML muy accesible para los diseñadores gráficos
 2. Código Java muy accesible para los programadores

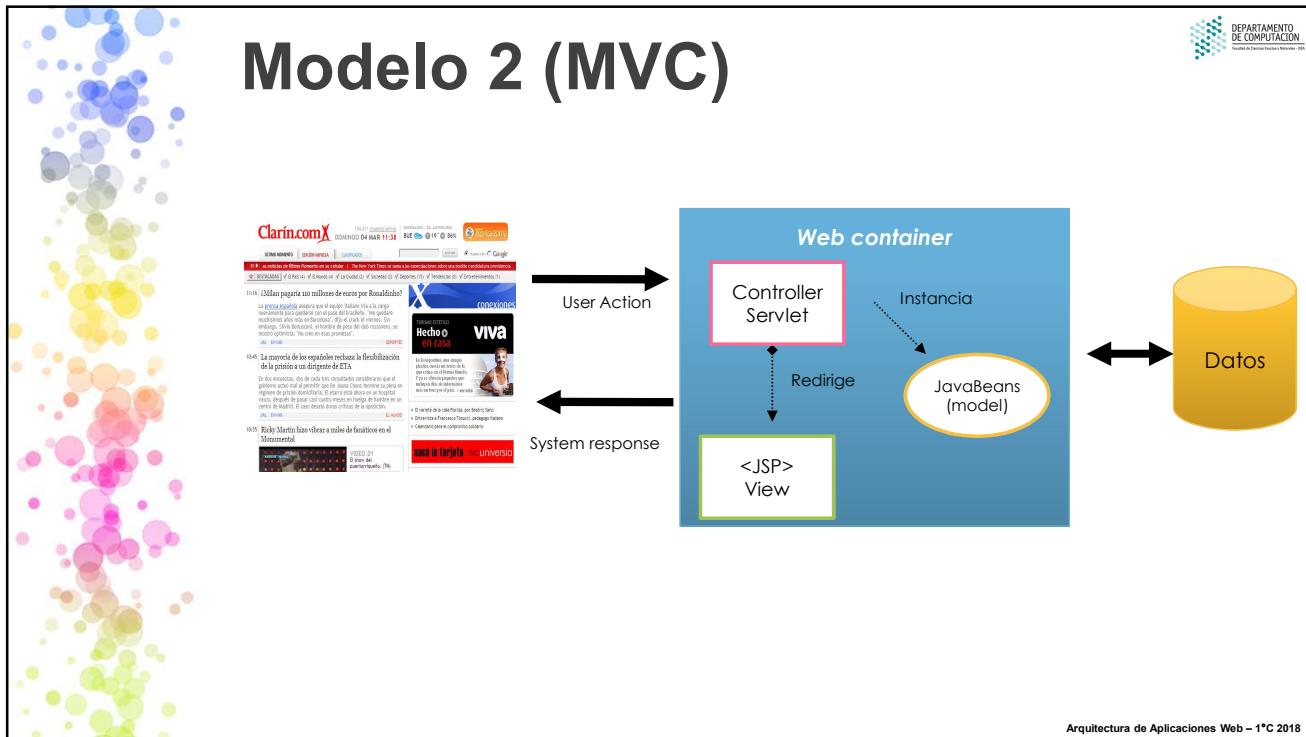


Modelo 2 (MVC)

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- ❖ El **request** del cliente es primero interceptado por un **servlet** : **Controller Servlet**
- ❖ El **Controller Servlet** maneja el procesamiento inicial del **request**
- ❖ El **Controller Servlet** determina cuál será la próxima página JSP

Arquitectura de Aplicaciones Web – 1*C 2018



Ventajas Modelo 2

- Único punto de entrada (control centralizado de la navegación)
- Clara separación de:
 - La lógica de negocio
 - La presentación
 - El procesamiento del request
- Genera páginas mas atractivas, fáciles de entender y mantener (mayor claridad, reutilización de código y mantenibilidad)

Arquitectura de Aplicaciones Web – 1*C 2018

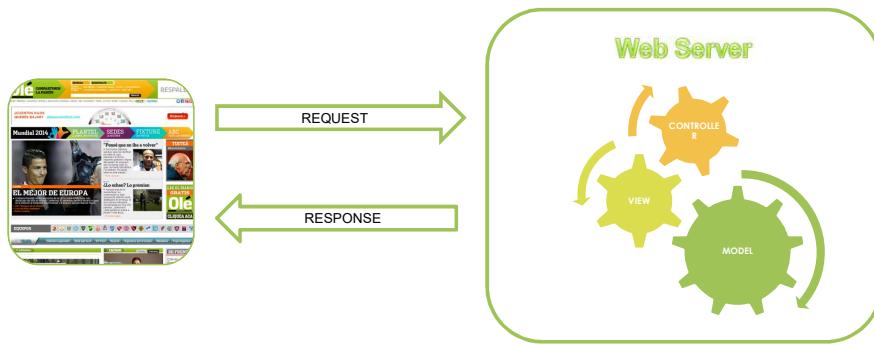
Model - View - Controller

- El patrón MVC tiene 3 componentes principales:
 - Model
 - *Responsable de mantener el estado y la lógica del dominio de negocio. (Objetos java, ORM framework, EJB, etc)*
 - View
 - *Responsable de la presentación de las vistas del dominio de negocio (generalmente constituido por páginas JSP y HTML)*
 - Controller
 - *Responsable de controlar el flujo y procesar el input del usuario*

Arquitectura de Aplicaciones Web – 1*C 2018

Había una vez...

- Model View Controller y derivados...



Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué es un Framework?

- Definición Simple

“Un framework es un conjunto de clases e interfaces que cooperan para resolver un tipo de problema específico de software”

Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué es un Framework?

- Características de un Framework
 - Está compuesto por múltiples clases o componentes, donde cada una provee una abstracción de algún concepto particular.
 - Define cómo esas abstracciones trabajan juntas para resolver el problema.
 - Sus componentes son reutilizables.
 - Organiza patrones a alto nivel.

Arquitectura de Aplicaciones Web – 1*C 2018

Framework vs. Librería

- Existe una sutil diferencia entre una librería y un framework

Librería	Framework
Contiene funciones o rutinas que nuestra aplicación puede invocar	Provee componentes genéricas que nuestra aplicación puede extender para proveer un conjunto particular de funciones
Tiene un rol pasivo	Tiene un rol activo

Arquitectura de Aplicaciones Web – 1*C 2018

¿Qué es un Framework?

Arquitectura de Aplicaciones Web – 1*C 2018

Utilidad de un Framework

- ❖ Es una estructura base
- ❖ Permite a los desarrolladores enfocarse en la problemática de negocio
- ❖ Reducción del tiempo de desarrollo
- ❖ Establece patrones de diseño de alto nivel que estructuran la solución, facilitando la comprensión y la mantenibilidad del producto
- ❖ Es extensible, permitiéndonos cambiar su comportamiento cuando lo consideremos necesario

Arquitectura de Aplicaciones Web – 1*C 2018



Struts

- ¿Qué es Struts?
 - Es un **Framework** de desarrollo **web** que implementa **MVC**
- Origen de Struts
 - El Framework Struts fue creado por Craig R. McClanahan y donado a la ASF (Apache Software Fundation). Craig está profundamente relacionado con el grupo de expertos en las especificaciones de Servlet y JSP. También escribió un gran porcentaje de la implementación del Tomcat 4.0.
 - Struts es un proyecto Open Source donde participa un gran equipo de expertos. Se mejora constantemente aumentando su valor y utilidad.

Objetivos de Struts

- Struts busca solucionar problemas comunes en el desarrollo de aplicaciones *web*
 - Modularización: separación de las responsabilidades en componentes distintas (Model, View, Controller)
 - Redundancia de código en todas las páginas JSP
 - Existencia de código Java en las páginas JSP
 - Demasiado tiempo de desarrollo en la elaboración de vistas
 - Dificultades para validar el *input* del usuario
 - Dificultad para desarrollar aplicaciones que soporten distintos idiomas
 - Falta de patrones de diseño

Arquitectura de Aplicaciones Web – 1*C 2018

Filosofía de Struts

- Struts se basa en una práctica que consistía en un *mapeo* entre una tecla y una función.
- Struts *mapea* una *acción del usuario* con una *componente de software* que atienda este requerimiento.



Ver la ayuda

Acción del usuario → Componente 1

Arquitectura de Aplicaciones Web – 1*C 2018



Componentes de Struts

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

I : Controller
II : Model
III: View

Arquitectura de Aplicaciones Web – 1ºC 2018



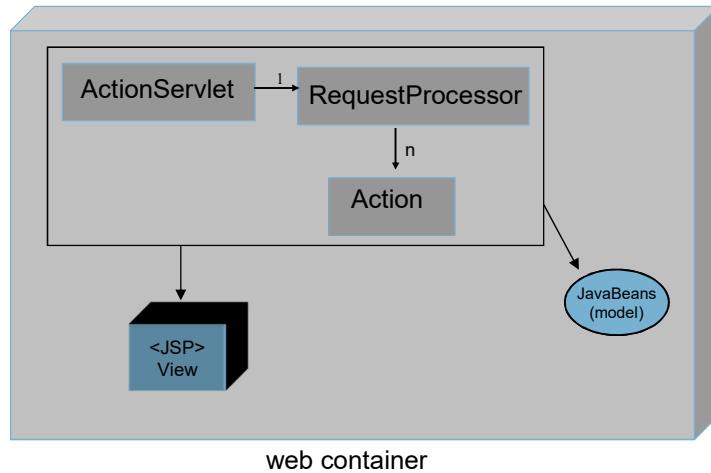
Controller

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Responsabilidades
 - Interceptar el **request** del cliente
 - Mapear cada **request** a una **operación de negocio** específica
 - Recolectar los **resultados** de las operaciones de negocio
 - Determinar la **próxima vista** a mostrar al cliente basándose en el estado actual y en los resultados de las operaciones de negocio

Arquitectura de Aplicaciones Web – 1ºC 2018

Componentes del Controller



Arquitectura de Aplicaciones Web – 1*C 2018

ActionServlet

- **Responsabilidades**
 - Esta clase actúa como un **interceptor**
 - Funcionalidad principal: **inicialización del Framework**
 - Dispara el **procesamiento del request** utilizando la clase **RequestProcessor**
 - Esta no es una clase abstracta. Tiene una **implementación default** que puede ser usada por nuestra aplicación

Arquitectura de Aplicaciones Web – 1*C 2018

RequestProcessor

- Es llamada por el *ActionServlet* cada vez que llega un **nuevo request**
- No es una clase abstracta. Tiene una **implementación default** que puede ser usada por nuestra aplicación
- Esta clase será la encargada de **procesar el request** (Ejemplo: obtener la *url* y **encontrar el handler apropiado** para ese *request* en función de la *url*)

Arquitectura de Aplicaciones Web – 1ºC 2018

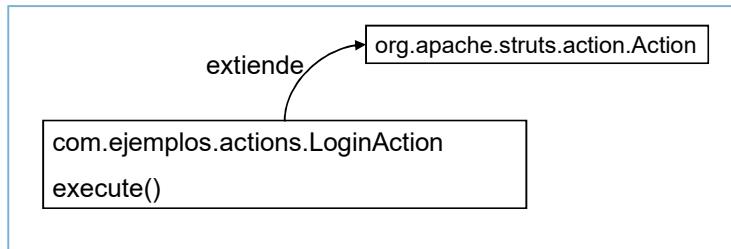
Action

- Es el punto de **extensión del Framework**
- Comprende nuestro mayor aporte de código
- Es el **puente** entre el **request** del cliente y una **operación de negocio**
- Cada **Action** generalmente es diseñado para resolver una **única operación de negocio**
- **Desacopla la vista de la lógica de negocio**
- Los **Action** son los encargados de acceder al **modelo de negocio**

Arquitectura de Aplicaciones Web – 1ºC 2018

Creación de un Action

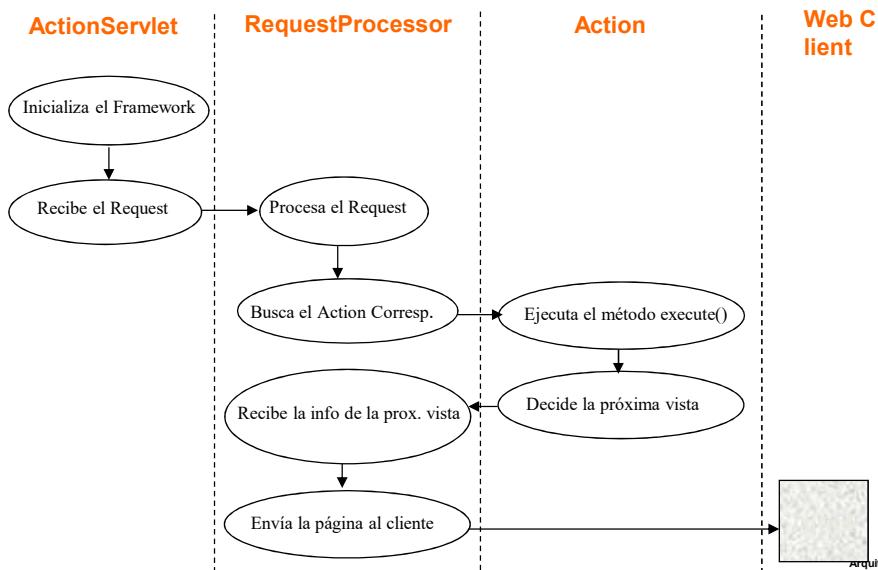
- El Framework nos provee una clase Action abstracta



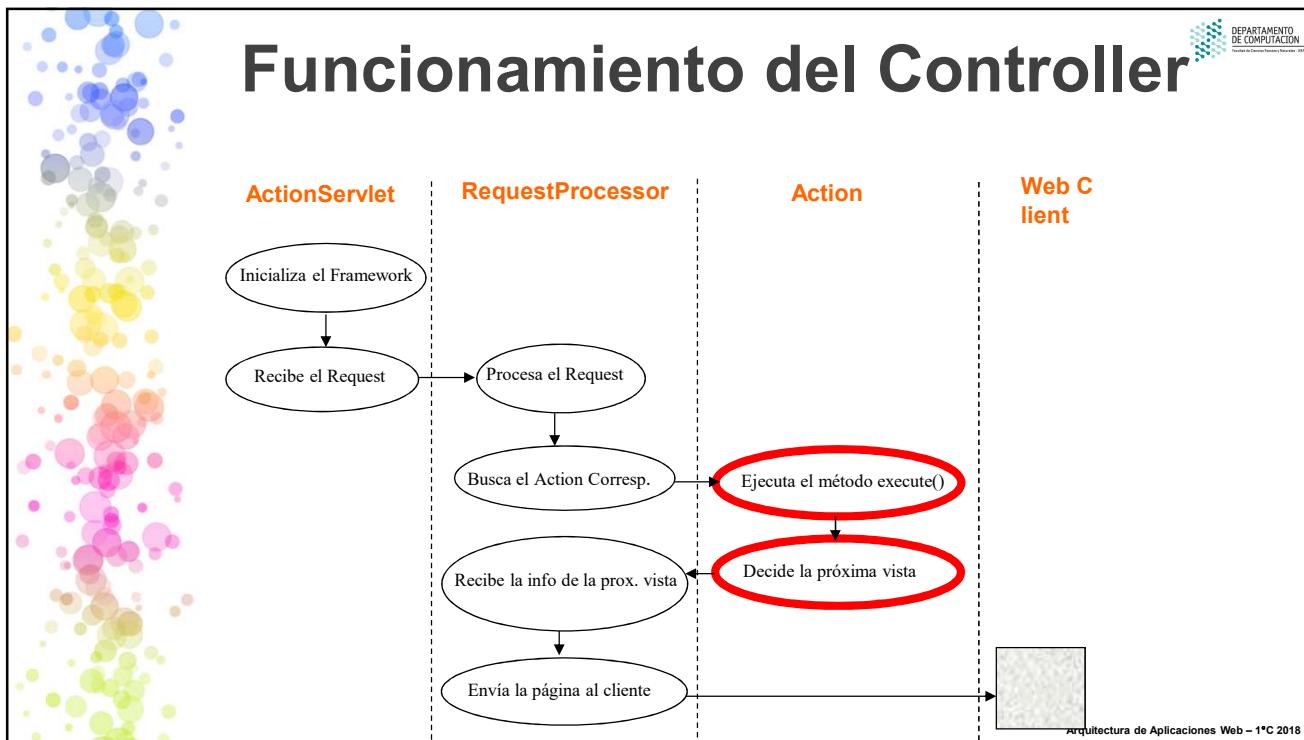
Cuando el **RequestProcessor** recibe un **request** determina que **Action** debe responder a ese **request** y llama al método **execute()** del **Action** correspondiente

Arquitectura de Aplicaciones Web – 1*C 2018

Funcionamiento del Controller



Arquitectura de Aplicaciones Web – 1*C 2018



Mapeando los Actions

- El **RequestProcessor** determina el **Action** que manejará el requerimiento, inspeccionando el **request** y usando un conjunto de **ActionMappings**
- Los **ActionMappings** forman parte de la configuración de **Struts** (XML especial : *struts-config.xml*)
- Esta configuración es cargada al iniciar la aplicación.
- Cada mapeo es representado en memoria por una instancia de la clase **org.apache.struts.action.ActionMapping**

Arquitectura de Aplicaciones Web – 1*C 2018

ActionMapping: Ejemplo

- Este es un fragmento del `struts-config.xml` que contiene un mapeo para un **Action** que valida el login de un usuario

```
<action-mappings>
  <action
    path="/loginAction"
    type="com.ejemplos.actions.LoginAction"
    name="loginForm"
    scope="session"
    input="/login.jsp">
    <forward name="Success" path="/member.jsp"/>
    <forward name="Failure" path="/login.jsp"/>
  </action>
<action-mappings>
```

Arquitectura de Aplicaciones Web – 1*C 2018

Clase ActionForward

- ¿Cómo determina el **RequestProcessor** la próxima vista?
- El método `execute()` de los **Actions** devuelven un objeto **ActionForward**
- La clase **ActionForward** representa el destino al cual el **RequestProcessor** enviará el control una vez que un **Action** complete su ejecución
- En lugar de especificar el destino en el código de un **Action**, se puede configurar declarativamente en el `struts-config.xml`
- Volvamos al ActionMapping que mostramos como ejemplo...**

Arquitectura de Aplicaciones Web – 1*C 2018

ActionMapping : Ejemplo

- El siguiente Action Mapping tiene declarados dos forwards

```
<action-mappings>
    <action
        path="/loginAction"
        type="com.ejemplos.actions.LoginAction"
        name="loginForm"
        scope="session"
        input="/login.jsp">
        <forward name="Success" path="/member.jsp"/>
        <forward name="Failure" path="/login.jsp"/>
    </action>
</action-mappings>
```

Arquitectura de Aplicaciones Web – 1*C 2018

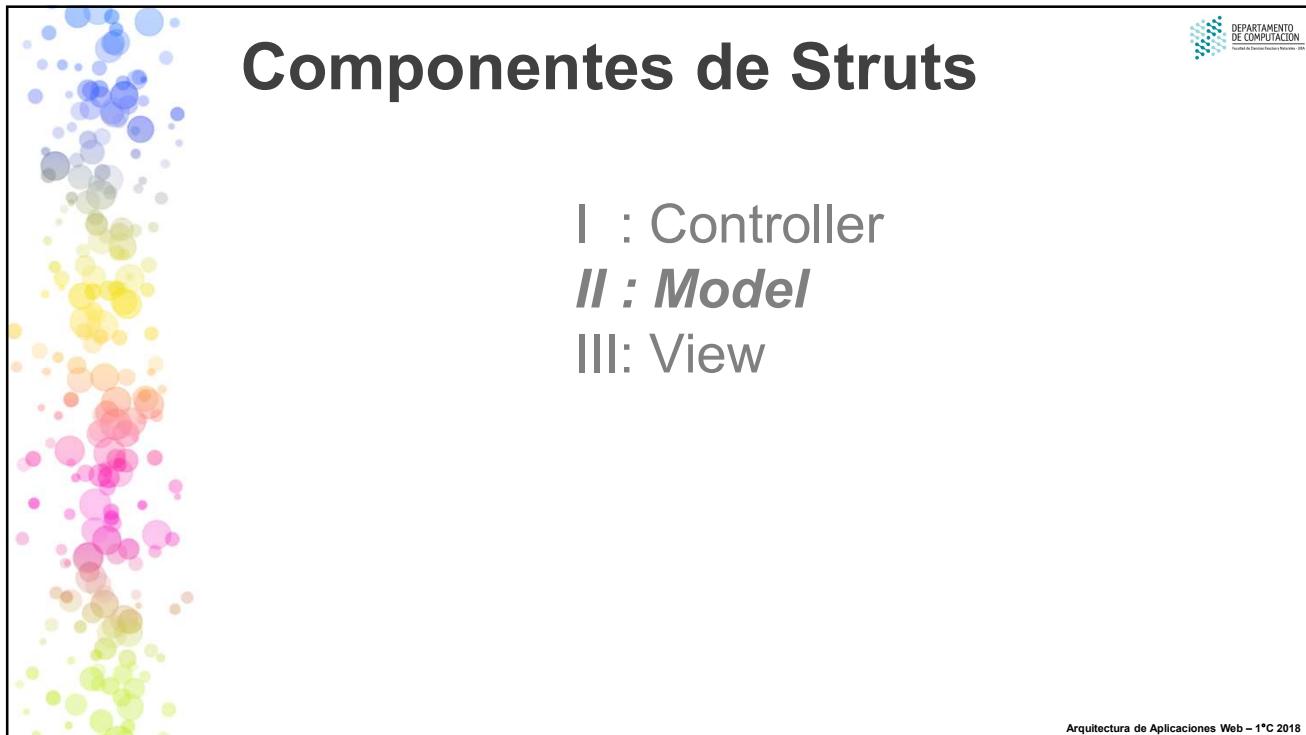
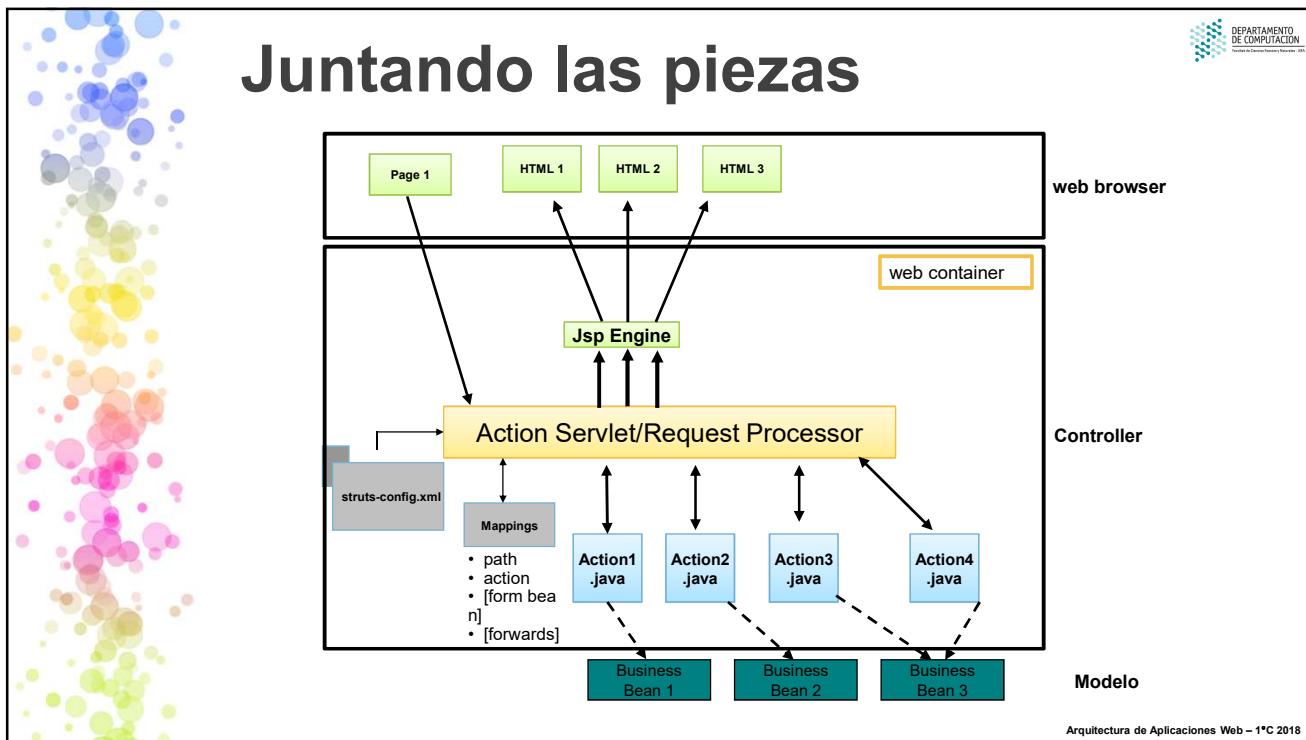
Global Forwards

- En el struts-config.xml se pueden definir forward s globales a la aplicación:

```
<global-forwards>
    <forwards name="error" path="/error.jsp"/>
</global-forwards>
```

- *Prioridad*: El controller busca primero en los forwa rds locales al Action correspondiente y luego busca en los globales

Arquitectura de Aplicaciones Web – 1*C 2018





Model

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Las componentes del **modelo** son **artifacts** muy valiosos
- El **modelo** incluye las **entidades de negocio** y las **reglas** que rigen el acceso y la modificación de los datos
- Las componentes que están dentro del **modelo** no deberían conocer el tipo de *Cliente o Framework* que las están usando
- Axioma de diseño : Las dependencias van hacia abajo y los datos van hacia arriba

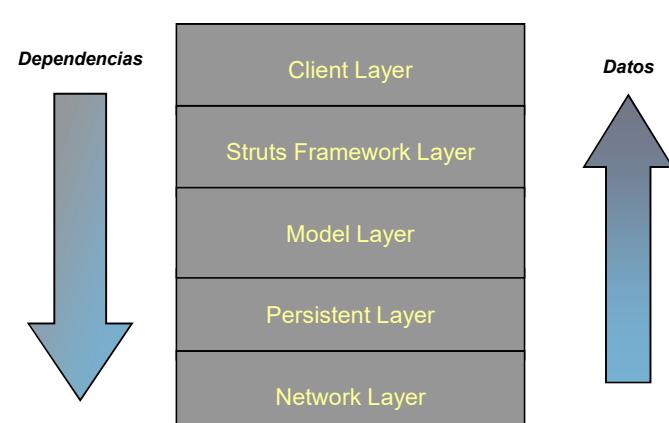
Arquitectura de Aplicaciones Web – 1*C 2018



Model

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Esta figura ilustra el hecho de que las capas de una aplicación sólo deben depender de las capas más bajas



Arquitectura de Aplicaciones Web – 1*C 2018

Struts y el Modelo

- El Framework Struts **no ofrece** nada para la construcción de componentes del modelo
- Existen y están disponibles muchas tecnologías para lidiar con el dominio de negocio de una aplicación. Ejemplo, **Enterprise JavaBeans (EJB)**, **Hibernate**, **Custom (DAO Pattern)**
- Struts tiene su enfoque en las capas **Controller** y **View**

Arquitectura de Aplicaciones Web – 1ºC 2018

Struts y el Modelo

- Struts plantea la **independencia del modelo**
 - Para garantizar esto, el método `execute()` de cada Action debería **delegar** en la/las clases correspondientes la resolución de la lógica de negocio relacionada

Arquitectura de Aplicaciones Web – 1ºC 2018

Componentes de Struts



I : Controller

II : Model

III: View

Arquitectura de Aplicaciones Web – 1*C 2018

View



- **Responsabilidades**
 - Esta capa tiene la responsabilidad de mostrar el modelo en una interfaz de usuario
 - Pueden haber varias vistas del mismo modelo
 - Metafóricamente, una vista es una ventana que los clientes pueden usar para mirar el estado del modelo, y la perspectiva puede ser distinta dependiendo de qué ventana esté utilizando

Arquitectura de Aplicaciones Web – 1*C 2018

View



- Las vistas en Struts se basan en la tecnología JSP
- Pueden utilizarse *componentes adicionales*:
 - *Documentos HTML.*
 - *JSP custom tags libraries.*
 - *JavaScript y Stylesheets.*
 - *Archivos Multimedia*
 - *Message Resource Bundles*
 - *Clases ActionForm*

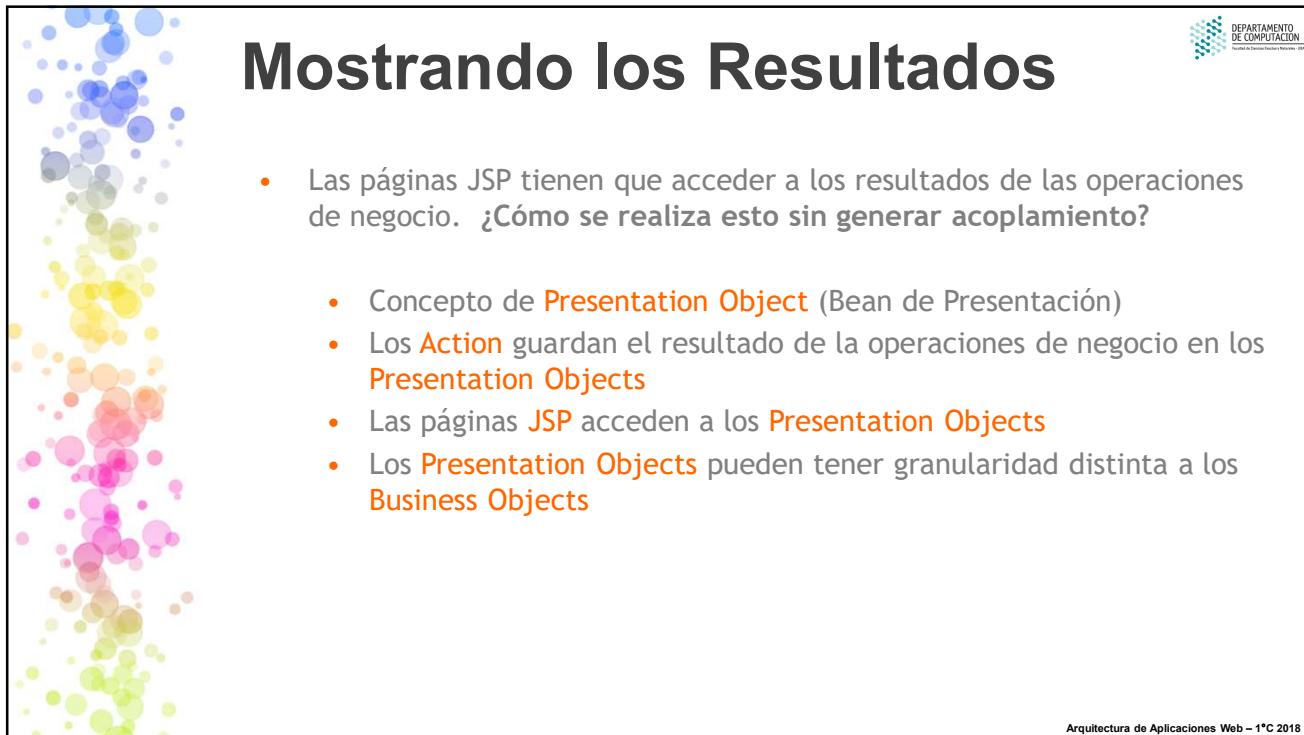
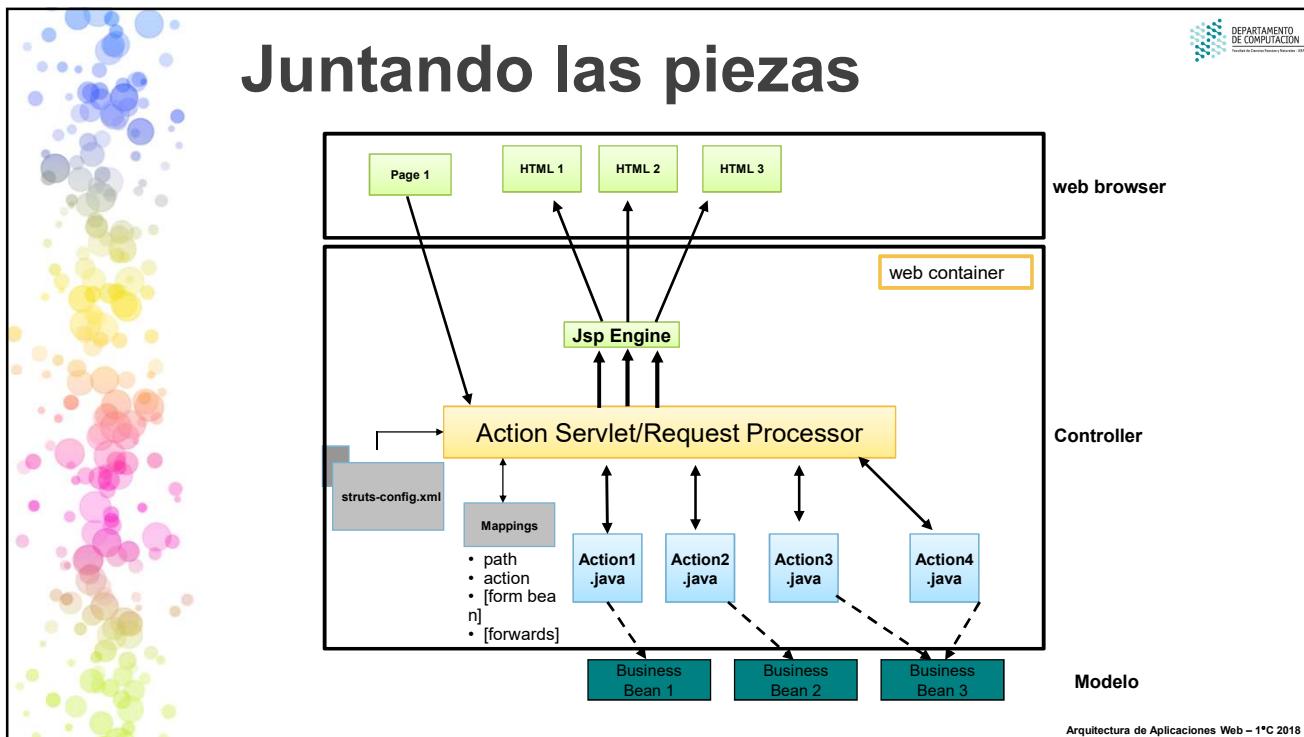
Arquitectura de Aplicaciones Web – 1*C 2018

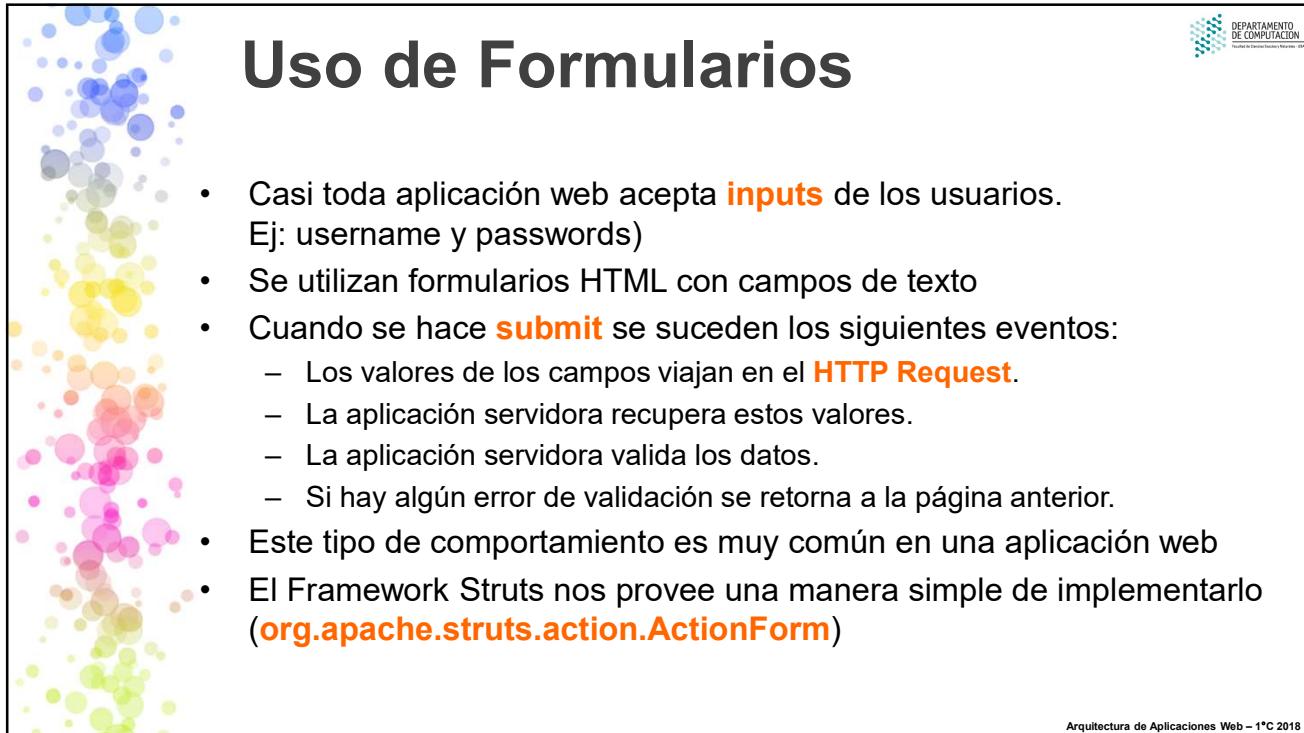
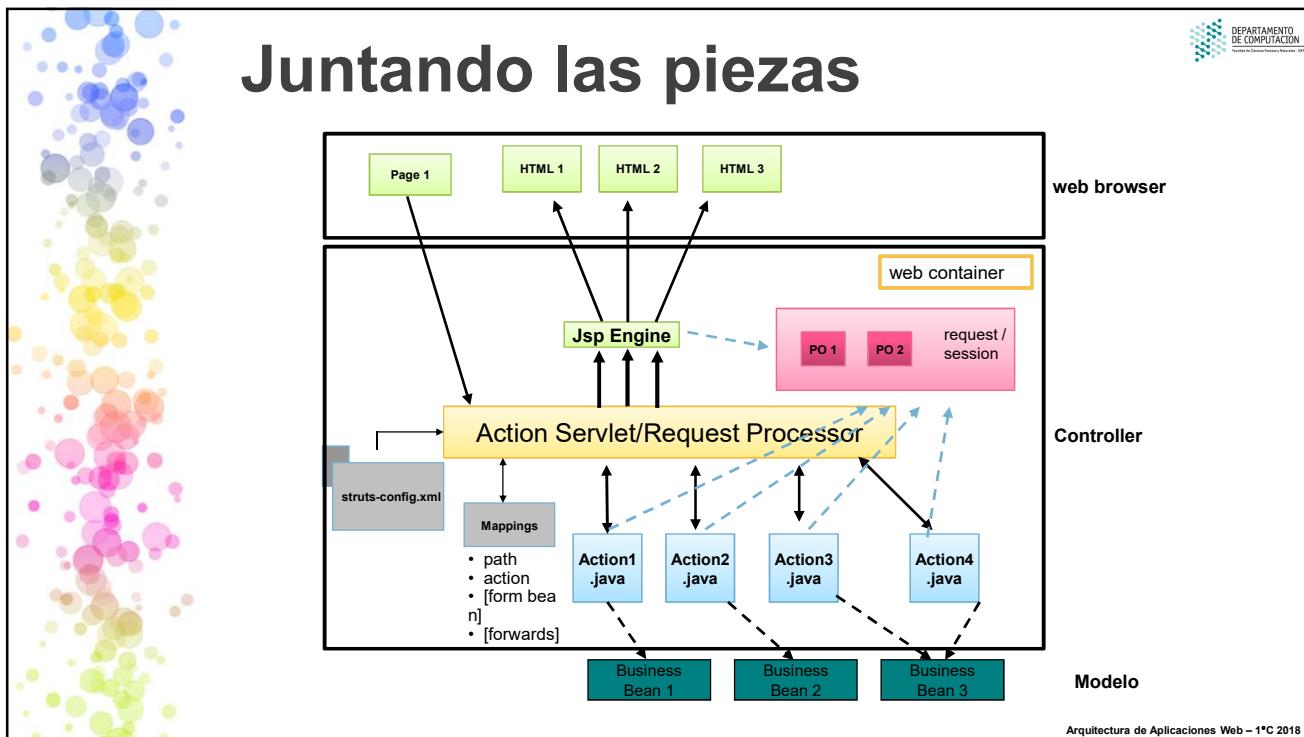
Mostrando los Resultados



- Las páginas JSP tienen que acceder a los resultados de las operaciones de negocio.
- ¿Cómo se realiza esto sin generar acoplamiento?

Arquitectura de Aplicaciones Web – 1*C 2018



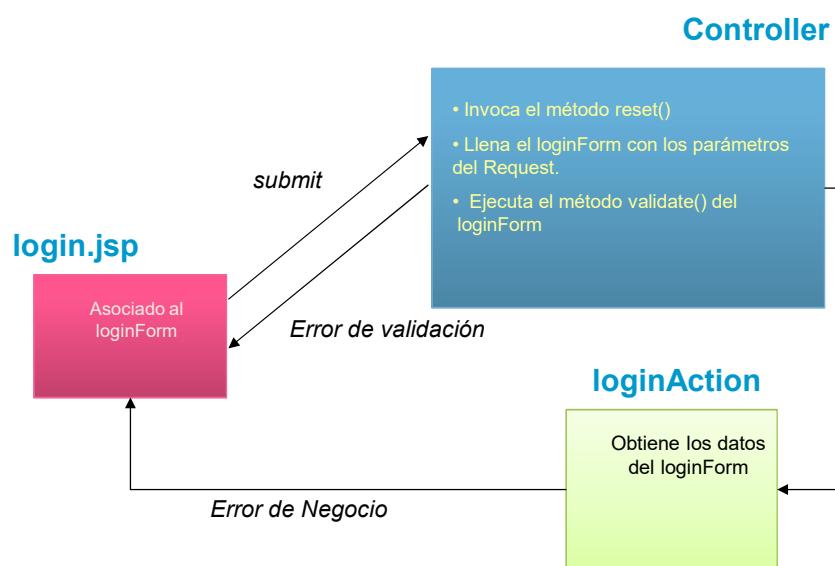


Clase ActionForm

- La clase **ActionForm** es utilizada para capturar el **input** de un formulario HTML y transferirlo a un **Action**
- Surgen por la necesidad de almacenar el **input** en un espacio temporal
- Son un espejo de un formulario HTML en el servidor
- Se puede decir que los **ActionForms** actúan como “**firewalls**”
- Las páginas JSP pueden llenar automáticamente los campos de un formulario utilizando los datos de un **ActionForm**

Arquitectura de Aplicaciones Web – 1*C 2018

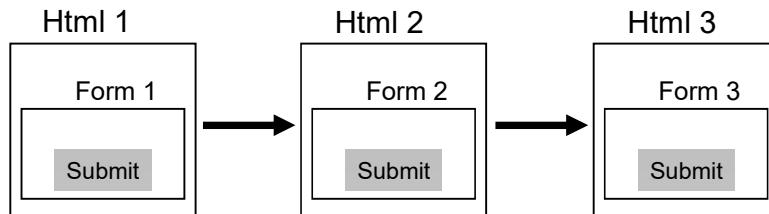
Usando Los ActionForm



Arquitectura de Aplicaciones Web – 1*C 2018

Scope de los ActionForms

- Los Action Form pueden tener 2 scopes distintos:
 - **Request:** Están disponibles durante el ciclo de vida request/response
 - **Session:** Se mantiene en la sesión hasta que sea explícitamente removido o reemplazado por otro objeto



Arquitectura de Aplicaciones Web – 1*C 2018

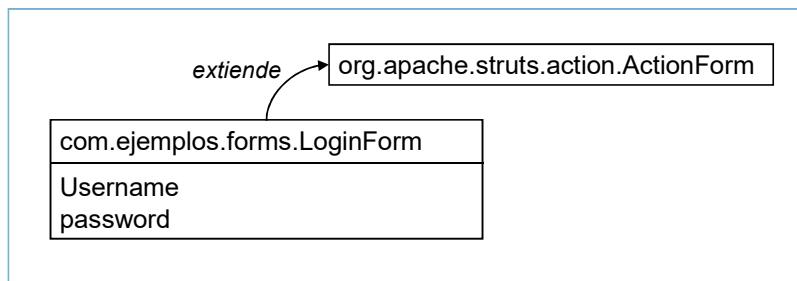
Configurando un ActionForm

```
<form-bean  
      name="loginForm"  
      type="com.nt.forms.LoginForm"/>  
  
<action  
      path="/loginAction"  
      type="com.ejemplos.actions.LoginAction"  
      name="loginForm"  
      scope="session"  
      input="/login.jsp">  
      <forward name="Success" path="/member.jsp"/>  
      <forward name="Failure" path="/login.jsp"/>  
</action>
```

Arquitectura de Aplicaciones Web – 1*C 2018

ActionForms/Actions

Struts provee una clase ActionForm abstracta

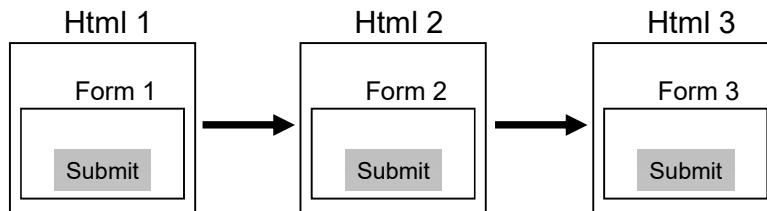


Los **Action** reciben como parámetro un **ActionForm** que será llenado por el controller cuando éste invoca el método `execute()`

Arquitectura de Aplicaciones Web – 1*C 2018

reset()

- Características del método `reset()`
 - Es llamado antes de que se llene el ActionForm con la llegada de cada Request.
 - Se utiliza para resetear las propiedades del Form a un valor default.
 - El comportamiento default es no hacer nada



Arquitectura de Aplicaciones Web – 1*C 2018

validate()

- Características del método validate()
 - Es configurable en cada **action mapping** si será o no llamado el método **validate()** del **ActionForm** asociado al **Action**
 - En este método se pueden realizar las validaciones
 - Retorna un objeto **ActionErrors** o **null** dependiendo si hubo o no errores en la validación

Arquitectura de Aplicaciones Web – 1*C 2018

Invocación del validate()

- **Características del método validate()**
 - Es configurable en cada **action mapping** si será o no llamado el método **validate()** del **ActionForm** asociado al **Action**
 - En este método se pueden realizar las validaciones
 - Retorna un objeto **ActionErrors** o **null** dependiendo si hubo o no errores en la validación

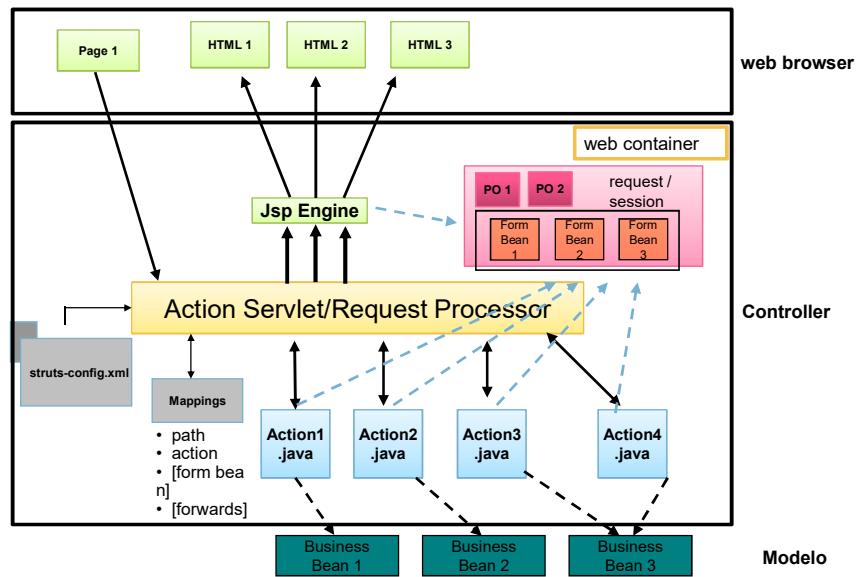
Arquitectura de Aplicaciones Web – 1*C 2018

ActionForms Dinámicos

- Para polemizar...
- Sobre los ActionForms:
 - Cantidad de clases en un Proyecto.
 - Cuando se elimina una propiedad de un Html, hay que modificar la clase y recompilar.
 - Tiempo en la construcción: definición de propiedades, getters y setters.
- Por esto, en *Struts*, aparecen los *ActionForms dinámicos* ...

Arquitectura de Aplicaciones Web – 1*C 2018

Juntando las piezas



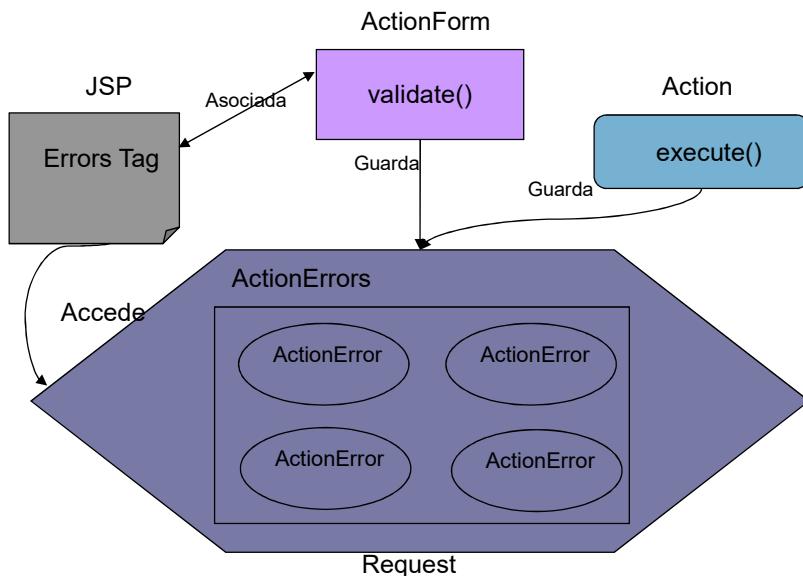
Arquitectura de Aplicaciones Web – 1*C 2018

ActionErrors

- Representa un conjunto de errores
- Cada error descubierto es representado por una instancia de la clase `org.apache.struts.action.ActionError`
- Se utiliza en los métodos `validate()` y `execute()`
- Funcionamiento: se crea un objeto `ActionErrors`, se le agregan los `ActionError` y se guarda el objeto en el `Request`.
- Existe un `Custom Tag Jsp` que muestra los errores que contiene un objeto `ActionErrors` muy fácilmente. Este objeto es obtenido del `Request`
- En las versiones posteriores, en general se utilizan `ActionMessages`

Arquitectura de Aplicaciones Web – 1*C 2018

Usando ActionErrors



Arquitectura de Aplicaciones Web – 1*C 2018

Internacionalización

- **¿Qué es?**

“Es el proceso de diseñar software para soportar múltiples lenguajes y regiones, de manera tal que no se tenga que hacer reingeniería en la aplicación cada vez que un nuevo lenguaje, país necesita ser soportado.”

Arquitectura de Aplicaciones Web – 1*C 2018

Internacionalización

- Para que una aplicación soporte *internacionalización* debe poseer las siguientes características:
 - Lenguajes adicionales pueden ser soportados **sin cambios** en el código
 - Elementos de texto, mensajes, e imágenes residen **frente al código fuente**
 - Datos dependientes de la cultura como las fechas, horas, valores decimales, moneda son formateados correctamente para el lenguaje del usuario y su ubicación geográfica

Arquitectura de Aplicaciones Web – 1*C 2018

Internacionalización y Struts



- Struts se enfoca en la presentación de textos e imágenes en la aplicación:
 - El **Framework** puede determinar el **Locale** preferido del usuario y guardarlo en su sesión
 - Un **Locale** es una región, que comparte costumbres, cultura y lenguaje. Es representado por la clase `java.util.Locale`
 - **Struts** utiliza este **Locale** para buscar texto y otros recursos en los **Resource Bundle**

Arquitectura de Aplicaciones Web – 1*C 2018

ResourceBundles



- Agrupan un conjunto de recursos para un **Locale** específico
- Los recursos son, generalmente, **elementos de texto** como campos, etiquetas de los botones y mensajes de estado. También incluyen nombres de imágenes, mensajes de error y títulos de páginas

Arquitectura de Aplicaciones Web – 1*C 2018

Tag Libraries de Struts

- Los **Custom Tags** provistos por el **Framework Struts** son agrupados en 5 librerías distintas:
 - HTML
 - Bean
 - Logic
 - Template
 - Nested

Arquitectura de Aplicaciones Web – 1ºC 2018

Struts HTML Tags

- Esta librería contiene **tags** usados para la creación de objetos **HTML**
- En especial de formularios HTML y sus componentes
- Estos **tags** fueron diseñados para trabajar en conjunto con otros componentes de **l Framework**, incluyendo los **ActionForms**

Arquitectura de Aplicaciones Web – 1ºC 2018

Struts HTML Tags

errors tag

- Muestra un conjunto de mensajes de error (ActionErrors). Si no existe tal objeto, no muestra nada.
- Atributos principales:
 - `property`: valor que indica qué mensajes de error se van a mostrar.
- Utiliza las siguientes claves del bundle:
 - `errors.header`: HTML que se devuelve antes de la lista de errores
 - `errors.footer`: HTML que se devuelve despues de la lista de errores
 - `errors.prefix`: HTML que se devuelve antes de cada error
 - `errors.suffix`: HTML que se devuelve despues de cada error

Arquitectura de Aplicaciones Web – 1*C 2018

Struts HTML Tags

options tag

- Devuelve conjunto de tags HTML <option> representando las posibles opciones de un select. Estas opciones son obtenidas dinámicamente de una colección. La opción que aparece seleccionada se determina con la propiedad especificada en el tag <select> del ActionForm
- Este tag opera de dos maneras diferentes dependiendo de:
- Si se utiliza el atributo collection
- Si no se utiliza el atributo collection

Arquitectura de Aplicaciones Web – 1*C 2018

Struts Logic Tags

- ***present tag (notPresent)***
 - Evalúa el cuerpo del tag si la variable especificado no se encuentra en el scope especificado
 - Atributos principales:
 - **cookie**: nombre de una cookie que se utilizará para el chequeo
 - **parameter**: nombre de un parámetro del request que se utilizará para el chequeo
 - **name**: nombre del bean que se utilizará para el chequeo
 - **property**: nombre de la propiedad que se utiliza para obtener el valor al cual se le realizará el chequeo
 - **scope**: en donde se encuentra el bean especificado en la propiedad name

Arquitectura de Aplicaciones Web – 1*C 2018

Struts Logic Tags

- ***equal tag***
 - Compara (por igualdad) el valor de una variable contra una constante. Si los valores son iguales evalua el cuerpo del tag
 - Atributos principales:
 - **cookie**: nombre de una cookie que se utilizará para la comparación
 - **parameter**: nombre de un parámetro del request que se utilizará para la comparación
 - **name**: nombre del bean con el que se hará el chequeo
 - **property**: nombre de la propiedad que se utiliza para obtener el valor a comparar
 - **scope**: en donde se encuentra el bean especificado en la propiedad name
 - **value**: valor constante contra el que se realiza la comparación

Arquitectura de Aplicaciones Web – 1*C 2018

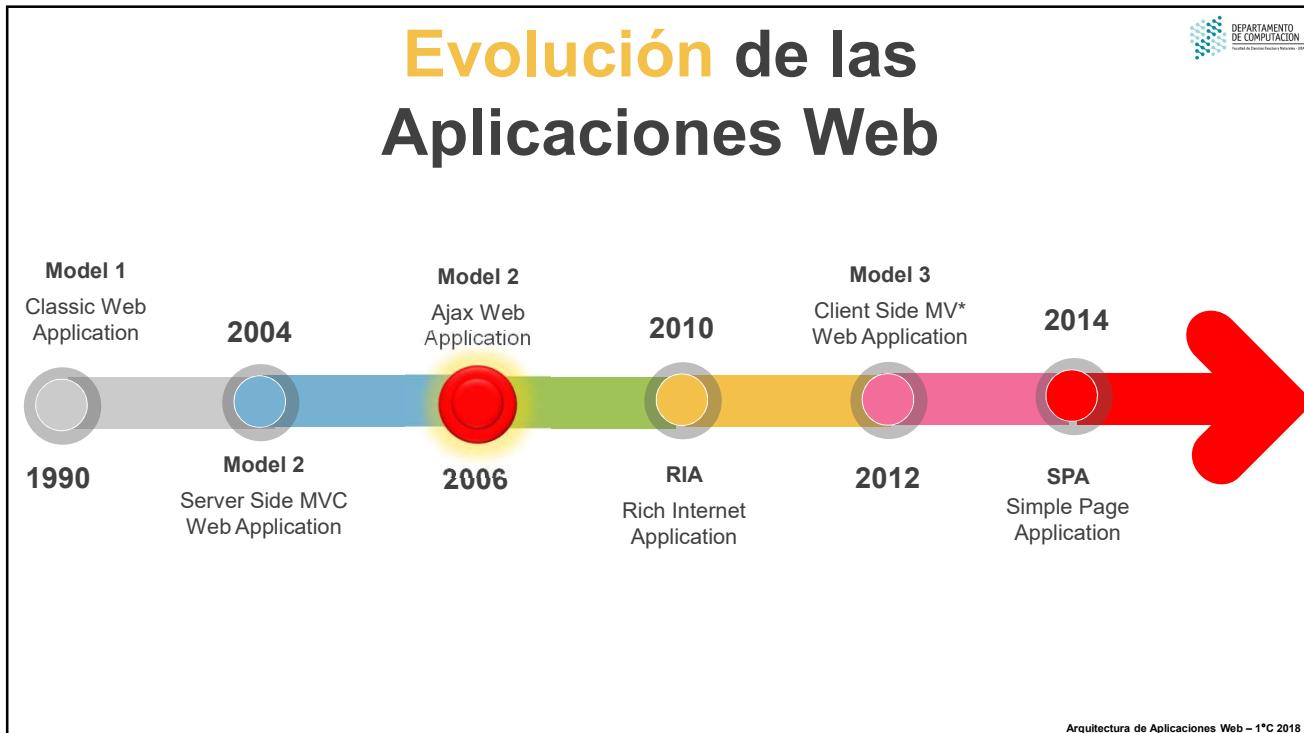


Struts Logic Tags

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- ***iterate tag***
- Repite el contenido del cuerpo del tag una vez por cada elemento perteneciente a un colección.
- Sobre qué se puede iterar?
- Un array de objetos java o tipos primitivos
- Una implementación de la interfaz java.util.Collection (ej:vector)
- Una implementación de la clase java.util.Enumeration
- Una implementación de la clase java.util.Iterator
- Una implementación de la clase java.util.map

Arquitectura de Aplicaciones Web – 1*C 2018



Había una vez...

- Model View Controller y derivados

The diagram illustrates three main architectural patterns:

- Browser-Controller-Model:** Shows a **BROWSER** sending an **HTTP REQUEST** to a **CONTROLLER**. The **CONTROLLER** sends **EXECUTION PARAMETERS** to a **MODEL** and receives **RESULTING DATA ARRAYS** from it. It also sends **HTTP RESPONSE** back to the **BROWSER** and **GUI CONTENT** to a **VIEW**. The **VIEW** sends **RESULTING DATA ARRAYS** back to the **CONTROLLER**.
- MVVM (MODEL VIEW VIEWMODEL):** Shows a **Model** (Geschäfts-Logik und Daten), a **ViewModel** (Presentations-Logik), and a **View** (Presentation XAML). They interact via bidirectional data binding.
- Detailed MVC Diagram:** Shows a **VIEW** interacting with a **MODEL** (which contains a database icon) and a **CONTROLLER** (which contains a mouse icon). The **VIEW** sends **updates** to the **CONTROLLER**, which then sends **writes** to the **VIEW**. The **VIEW** also sends **notifies** back to the **CONTROLLER**. The **MODEL** sends **fills** to the **VIEW**.

Arquitectura de Aplicaciones Web – 1*C 2018

Había una vez...

- Diversos frameworks implementaron las mismas ideas

The diagram shows a **REQUEST** from a **Web Browser** to a **Web Server**. The **Web Server** contains several frameworks:

- Jakarta Struts**
- RichFaces**
- Microsoft ASP.net**
- TakePHP**
- Microsoft .net MVC**
- tapestry**

Arquitectura de Aplicaciones Web – 1*C 2018



Mientras tanto...

- Web 1.0
- Web 2.0
- Surgen nuevas ideas... (2003/2004)
 - SOA
 - Portales

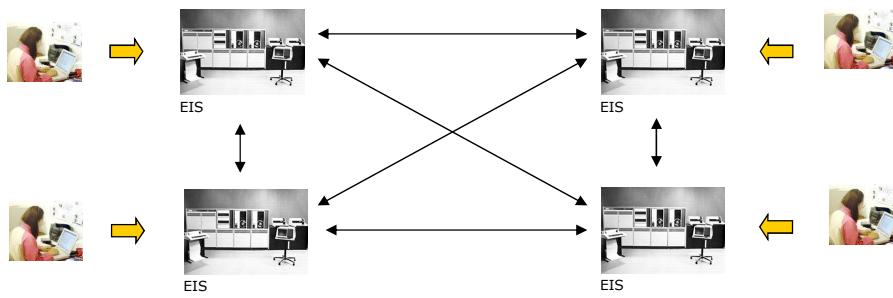
Necesidades

- Implementar arquitecturas que soporten Business On Demand
 - Reducir el Time To Market
- Abandonar la estrategia de grandes proyectos de reimplementaciones
 - Proyectos con rápido retorno de inversión
 - Reutilización de proyectos previos
- Reducir costos a través de lazos más cercanos con clientes, empleados y proveedores

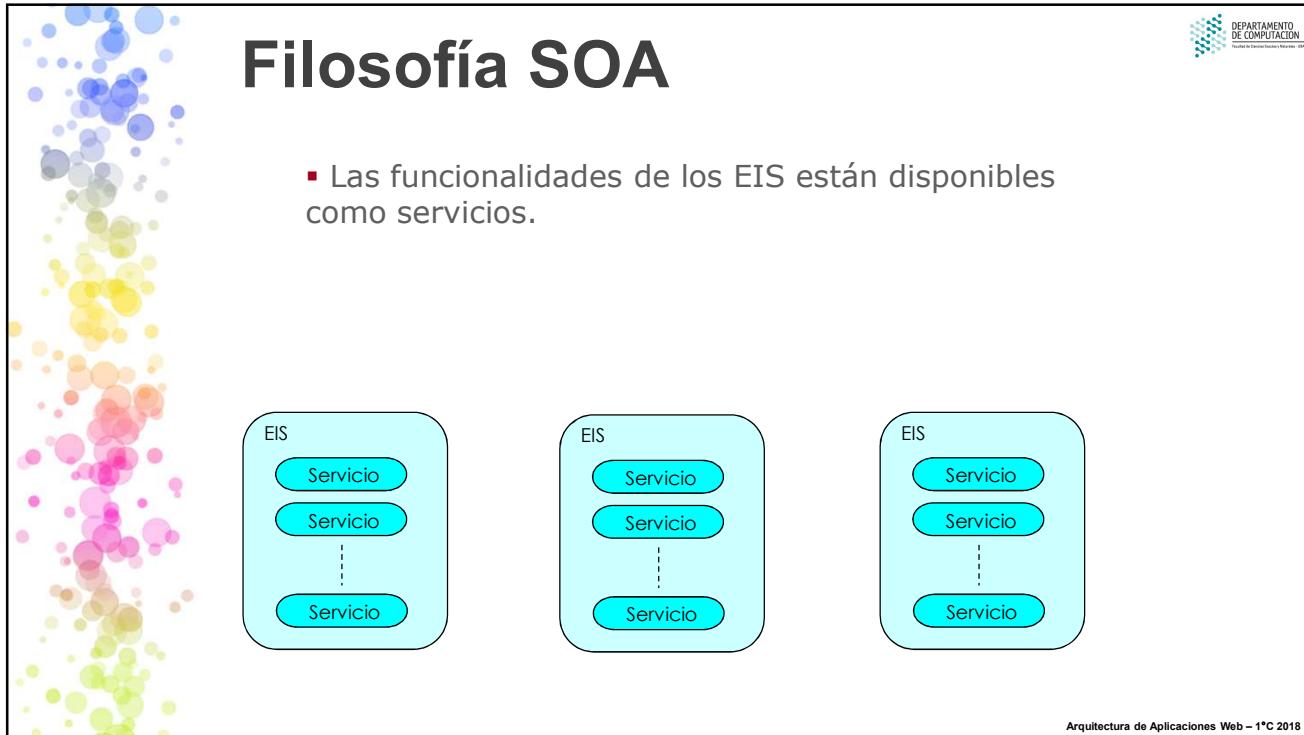
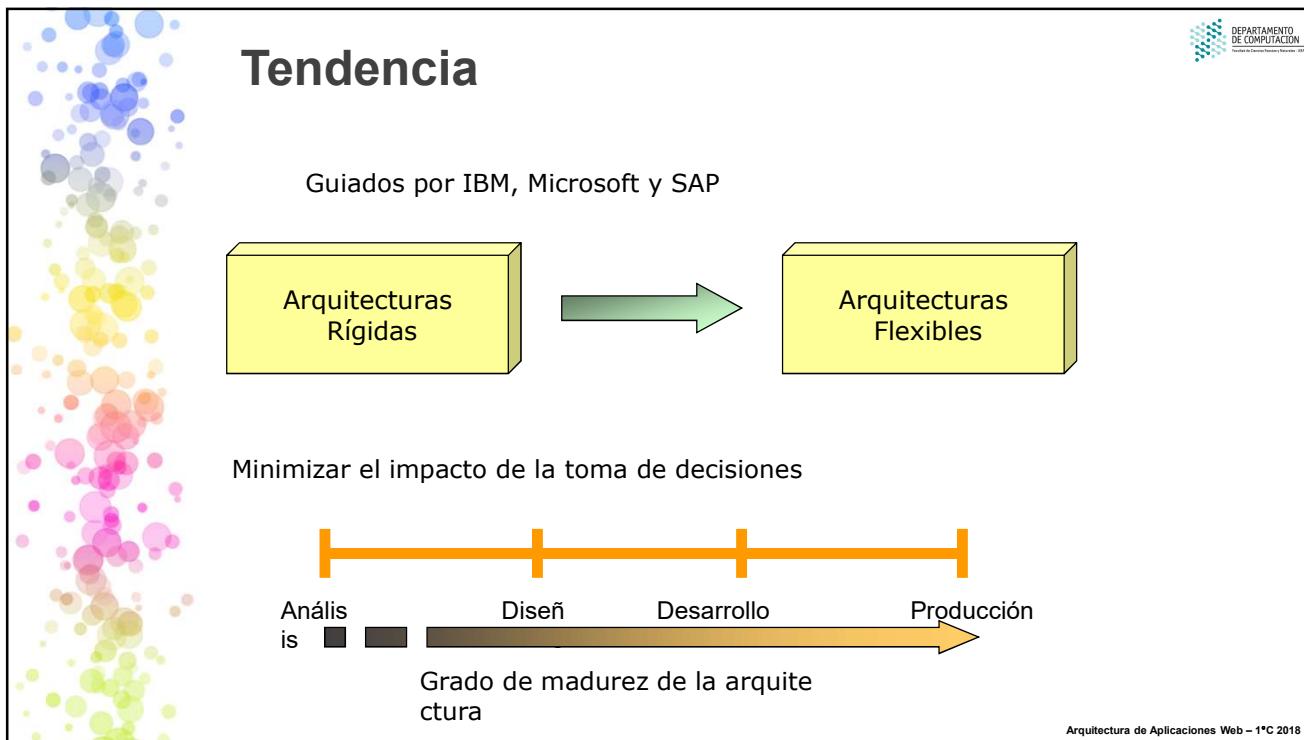
Arquitectura de Aplicaciones Web – 1*C 2018

Arquitectura Actual

- Las compañías necesitan integrar sistemas existentes
- Las interacciones existentes eran:
 - Esquemas muy rígidos (no hay visión ni arquitectura)
 - Múltiples accesos para los usuarios



Arquitectura de Aplicaciones Web – 1*C 2018



Filosofía SOA

- Las nuevas unidades de reutilización son las funciones de negocio.

The diagram illustrates the SOA philosophy. On the left, a vertical column of colored circles represents the 'Base de componentes reutilizables' (Reusable component base). To its right is a light blue rounded rectangle containing four teal rectangular boxes, each labeled 'Función de Negocio' (Business Function). Arrows connect the circles to the first two business functions, indicating their reuse.

Base de componentes reutilizables

Función de Negocio

Función de Negocio

Función de Negocio

Función de Negocio

Arquitectura de Aplicaciones Web – 1*C 2018

Filosofía SOA

- Se implementan procesos de negocio flexibles que utilizan estos componentes.

The diagram illustrates the SOA philosophy. On the left, a vertical column of colored circles represents the 'Base de componentes reutilizables' (Reusable component base). To its right is a light blue rounded rectangle containing four teal rectangular boxes, each labeled 'Función de Negocio' (Business Function). Above this is a light blue rounded rectangle labeled 'Business Process' containing several small gray rectangles connected by arrows. Red arrows point from the business functions to the business process, indicating their reuse.

Flujo y control de la lógica del negocio

Business Process

Business Process

Función de Negocio

Función de Negocio

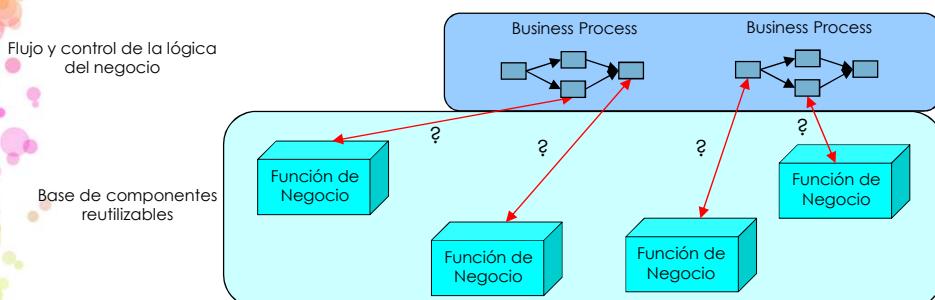
Función de Negocio

Función de Negocio

Arquitectura de Aplicaciones Web – 1*C 2018

Filosofía SOA

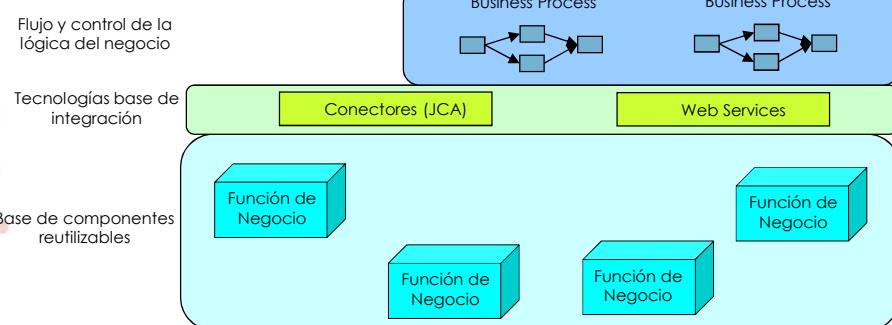
- Se necesitan tecnologías que permitan acceder a las funciones de negocio implementadas en distintas plataformas.



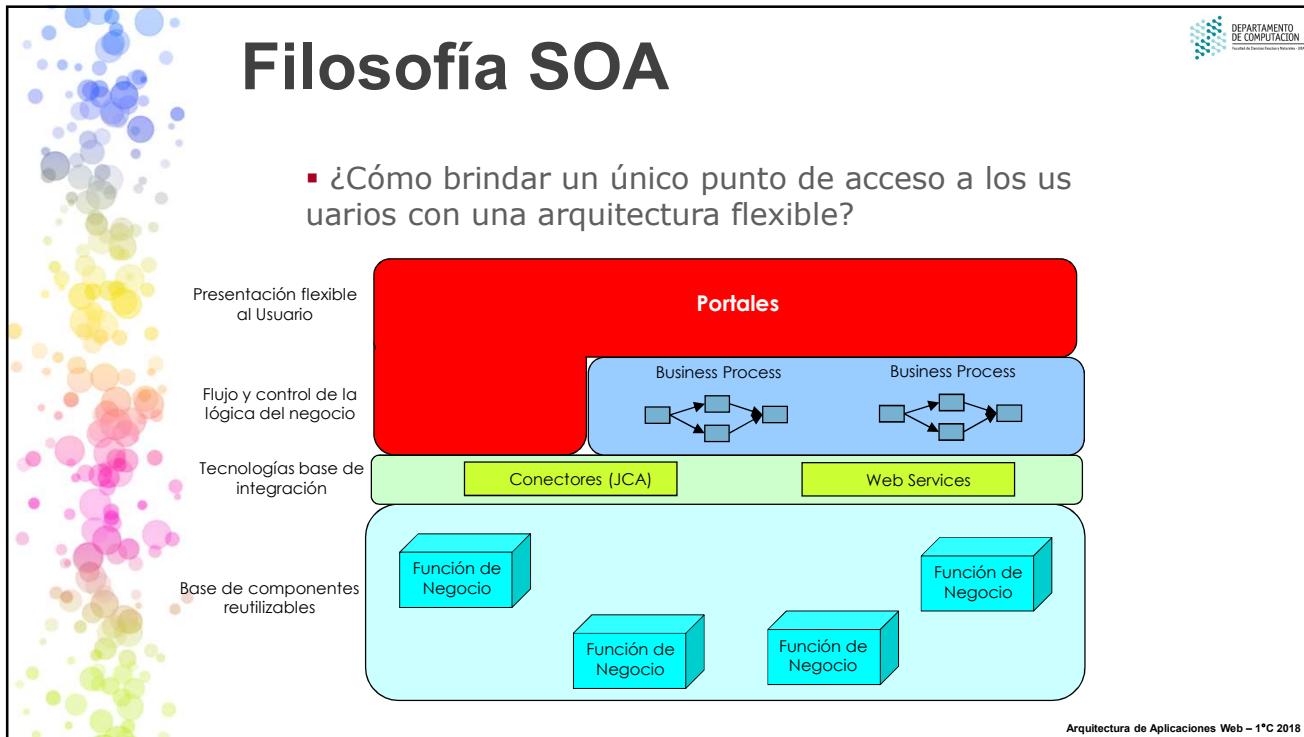
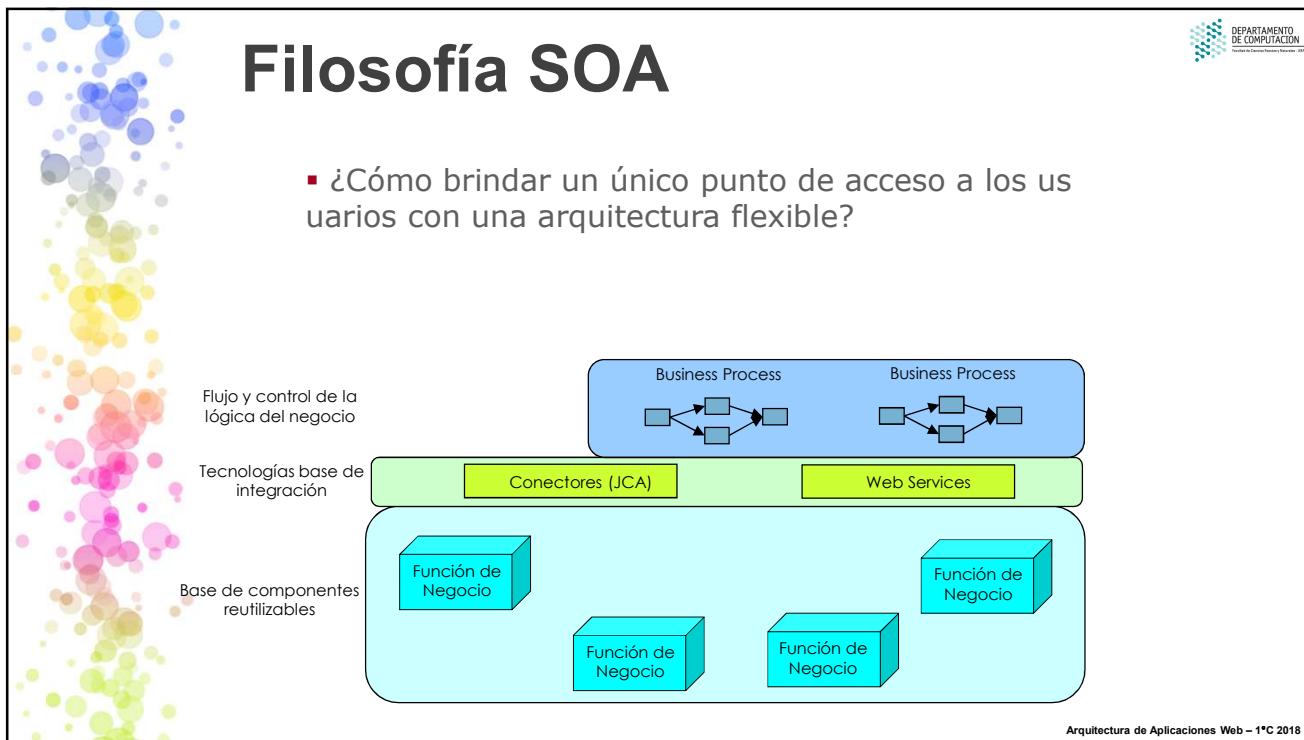
Arquitectura de Aplicaciones Web – 1*C 2018

Filosofía SOA

- Se necesitan tecnologías que permitan acceder a las funciones de negocio implementadas en distintas plataformas.



Arquitectura de Aplicaciones Web – 1*C 2018





Filosofía SOA

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Habilidad de reutilizar:
 - Funcionalidades existentes
 - Información existente
 - Sistemas existentes
- Entorno integrado y flexible
 - Soportar políticas comerciales cambiantes
- Brindar soluciones rápidamente
 - Nuevos mercados
 - Clientes
 - Necesidades internas de la empresa

Arquitectura de Aplicaciones Web – 1ºC 2018



SOA

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

“Para 2008, SOA será la práctica de ingeniería de software prevaleciente, acabando así 40 años de dominación de la arquitectura monolítica de software.”

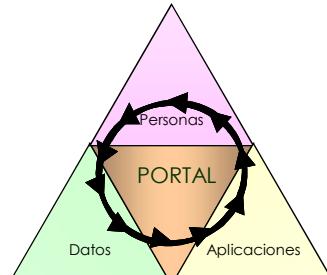
Gartner LE-19-7652
(Abril 2003)

Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué es un Portal?

Un portal es:

- ▶ Una aplicación Web
- ▶ Un único punto de acceso a los recursos de la empresa
- ▶ Personalizable según las necesidades y responsabilidades de los usuarios



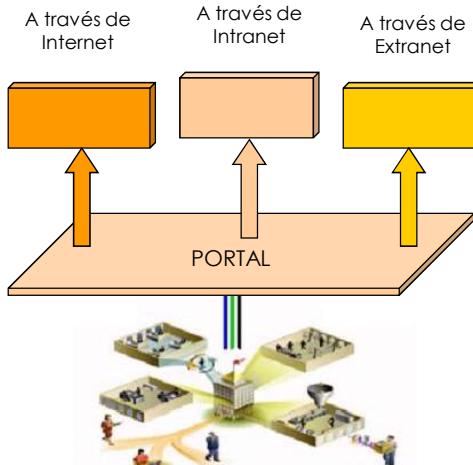
El diagrama muestra un triángulo dividido en tres secciones: la parte superior es rosa y contiene la palabra "Personas"; la parte izquierda es verde y contiene la palabra "Datos"; la parte derecha es amarilla y contiene la palabra "Aplicaciones". En el centro del triángulo, la palabra "PORTAL" aparece rodeada por un círculo que tiene flechas apuntando hacia el centro de cada lado.

- ✓ Integración
- ✓ Accesibilidad
- ✓ Personalización
- ✓ Seguridad

Arquitectura de Aplicaciones Web – 1*C 2018

Beneficios

Mayor disponibilidad de los recursos de la empresa



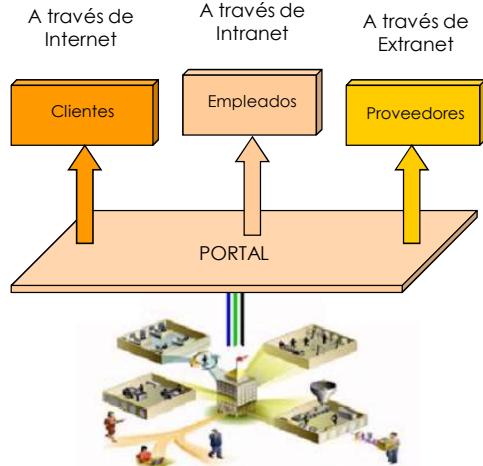
Este diagrama ilustra la funcionalidad centralizada de un portal. En el centro, una plataforma naranja se llama "PORTAL". De este centro parten tres flechas coloridas hacia arriba: una naranja hacia la izquierda (etiquetada "A través de Internet"), una marrón hacia la derecha (etiquetada "A través de Intranet") y una amarilla hacia abajo (etiquetada "A través de Extranet"). Debajo de la plataforma, se muestra un paisaje con edificios y personas, representando la red de usuarios y sistemas que interactúan a través del portal.

Arquitectura de Aplicaciones Web – 1*C 2018

Beneficios

DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UNLP

Mayor integración dentro de la cadena de valor

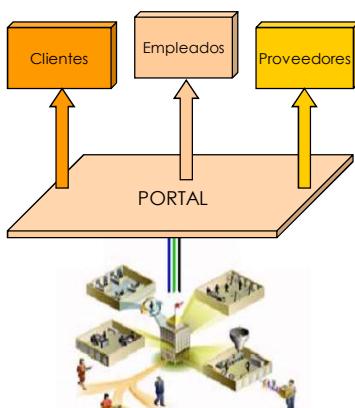


Arquitectura de Aplicaciones Web – 1ºC 2018

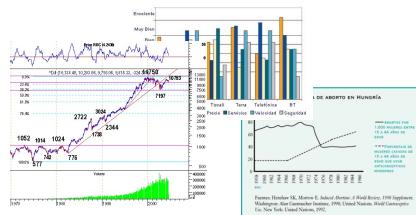
Beneficios

DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Esta tecnología facilita...



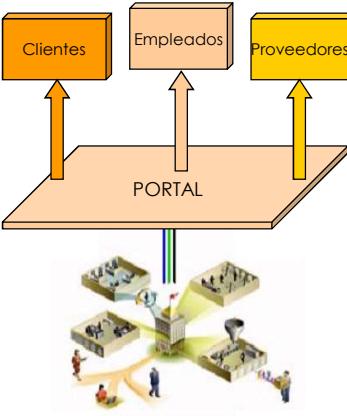
Responder al instante a constantes cambios de la competencia, a las expectativas de sus clientes y a las tendencias del mercado, en tiempo real



Arquitectura de Aplicaciones Web – 1°C 2018

Beneficios

Esta tecnología facilita...



Que todos los miembros de la empresa dispongan de la información correcta, en el momento adecuado



Arquitectura de Aplicaciones Web – 1*C 2018

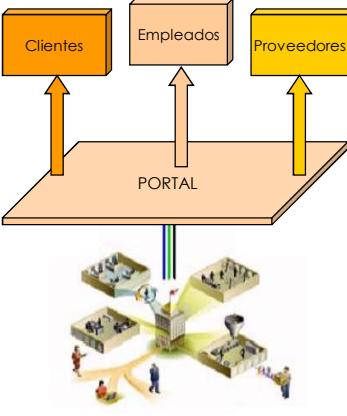
Beneficios

Esta tecnología facilita...

Estar antes en el mercado

Responder más rápido y mejor

Economizar los costos de capacitación



Arquitectura de Aplicaciones Web – 1*C 2018

Beneficios

No consiste en eliminar y reemplazar.
Consiste en trabajar con los recursos d
e los que ya se dispone

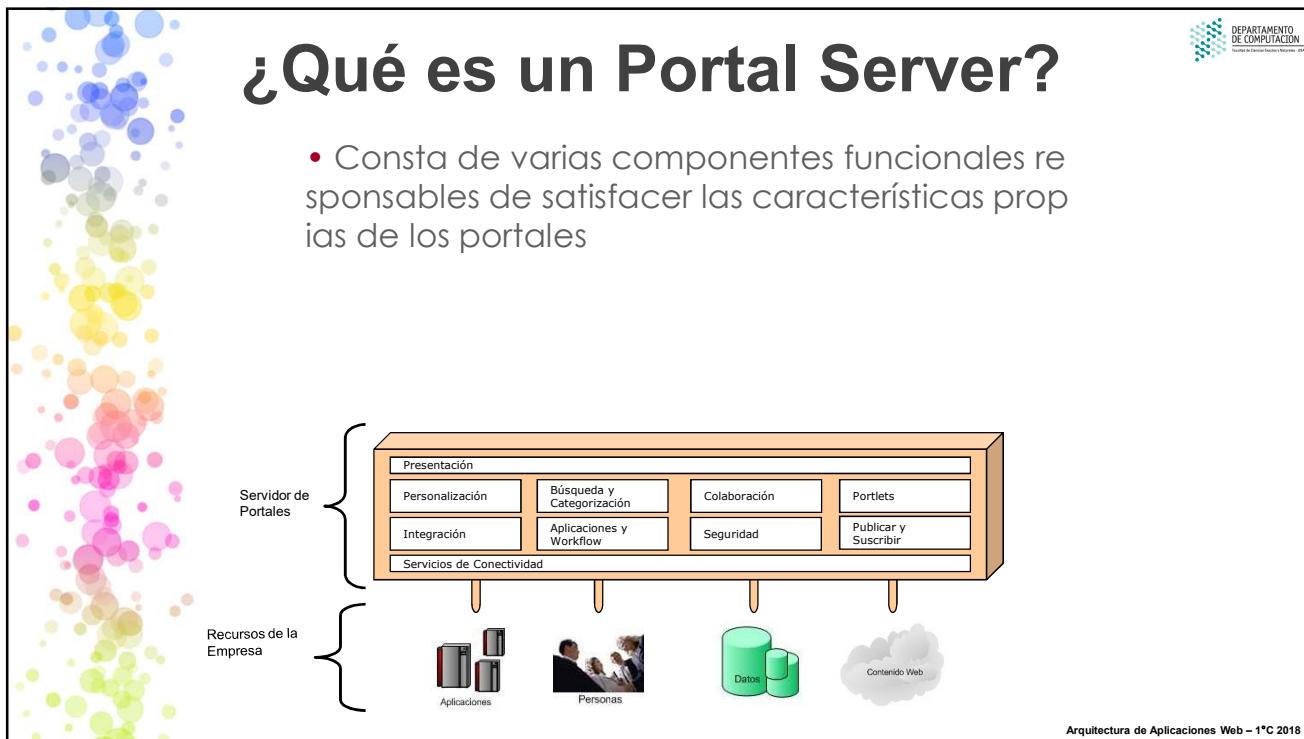
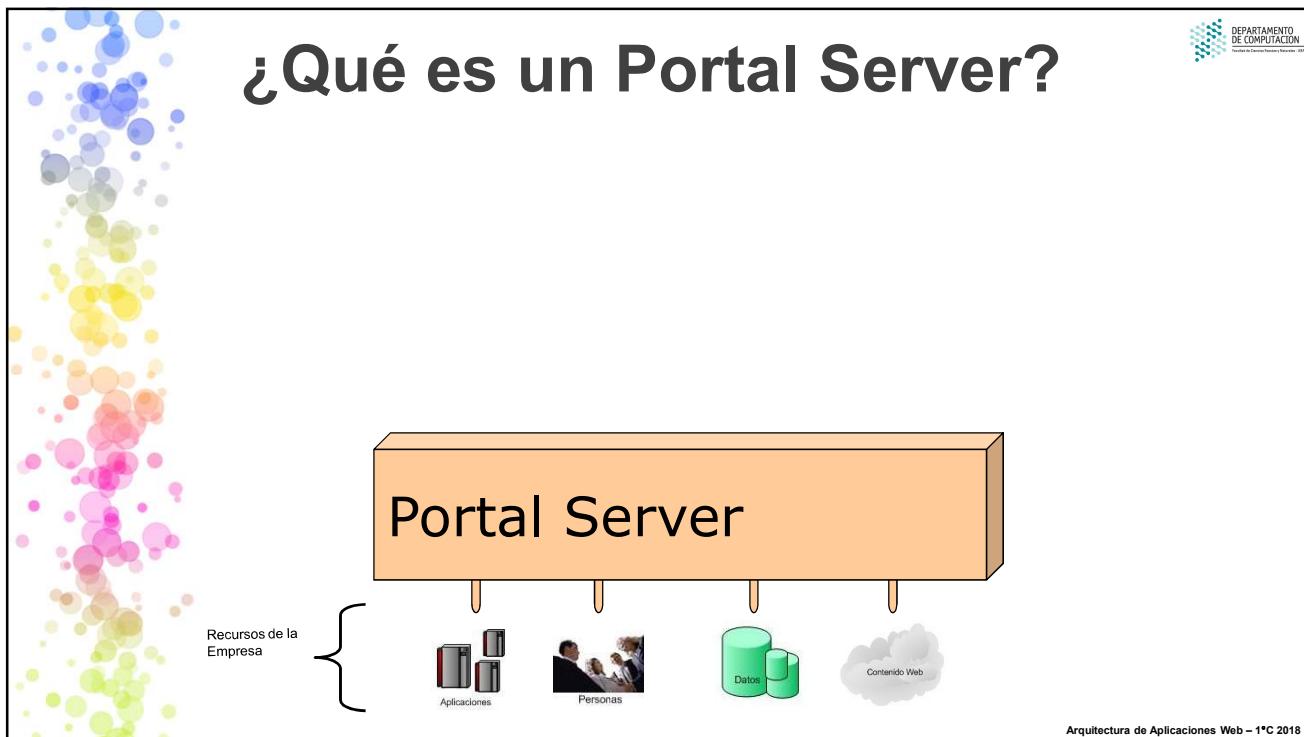


Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué es un Portal Server?

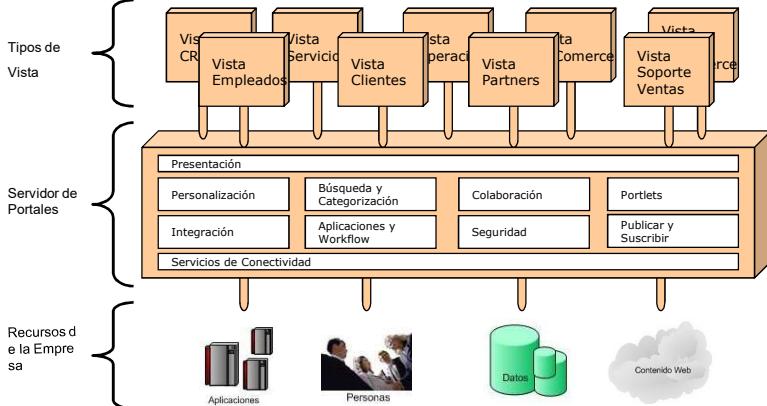
- Es un servidor de aplicaciones especializado, que provee lógica de negocio al portal
- Brinda infraestructura para el desarrollo y ejecución del portal
- Se encuentra apoyado sobre los recursos de la empresa

Arquitectura de Aplicaciones Web – 1ºC 2018



¿Qué es un Portal Server?

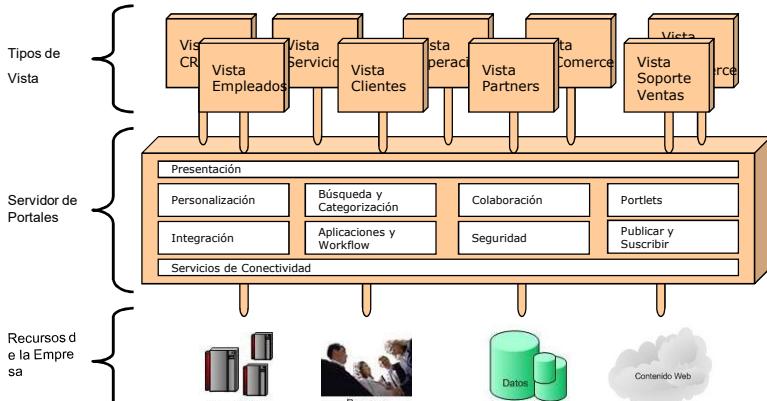
- Sobre él encontramos diferentes vistas, a manera de distintos tipos de portales, dedicados a dominios o unidades funcionales concretas.



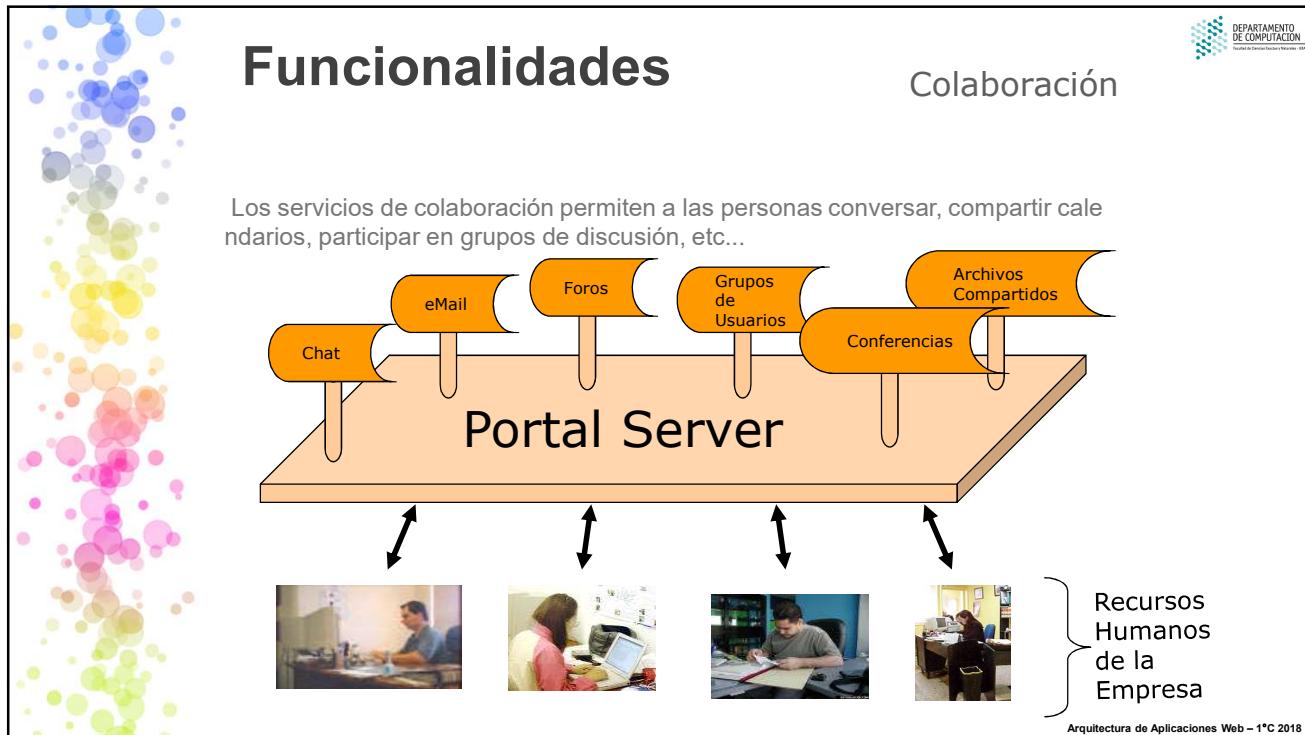
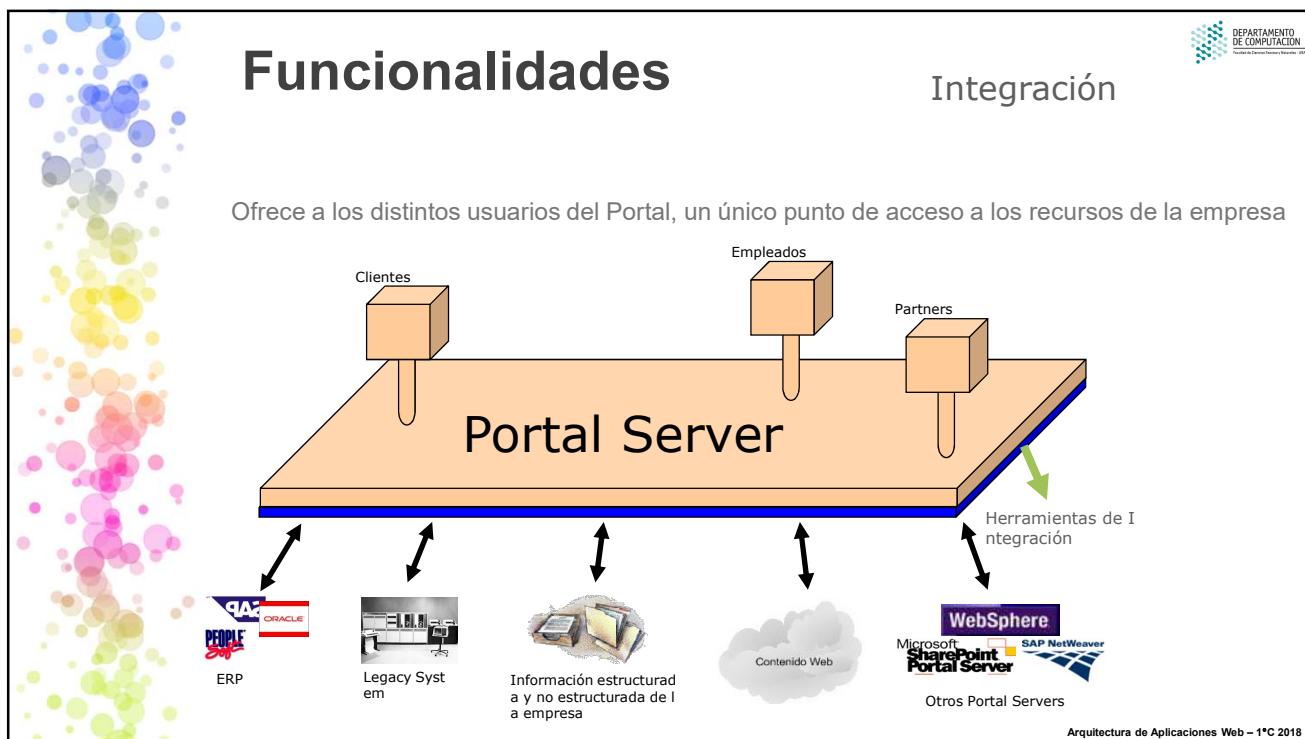
Arquitectura de Aplicaciones Web – 1*C 2018

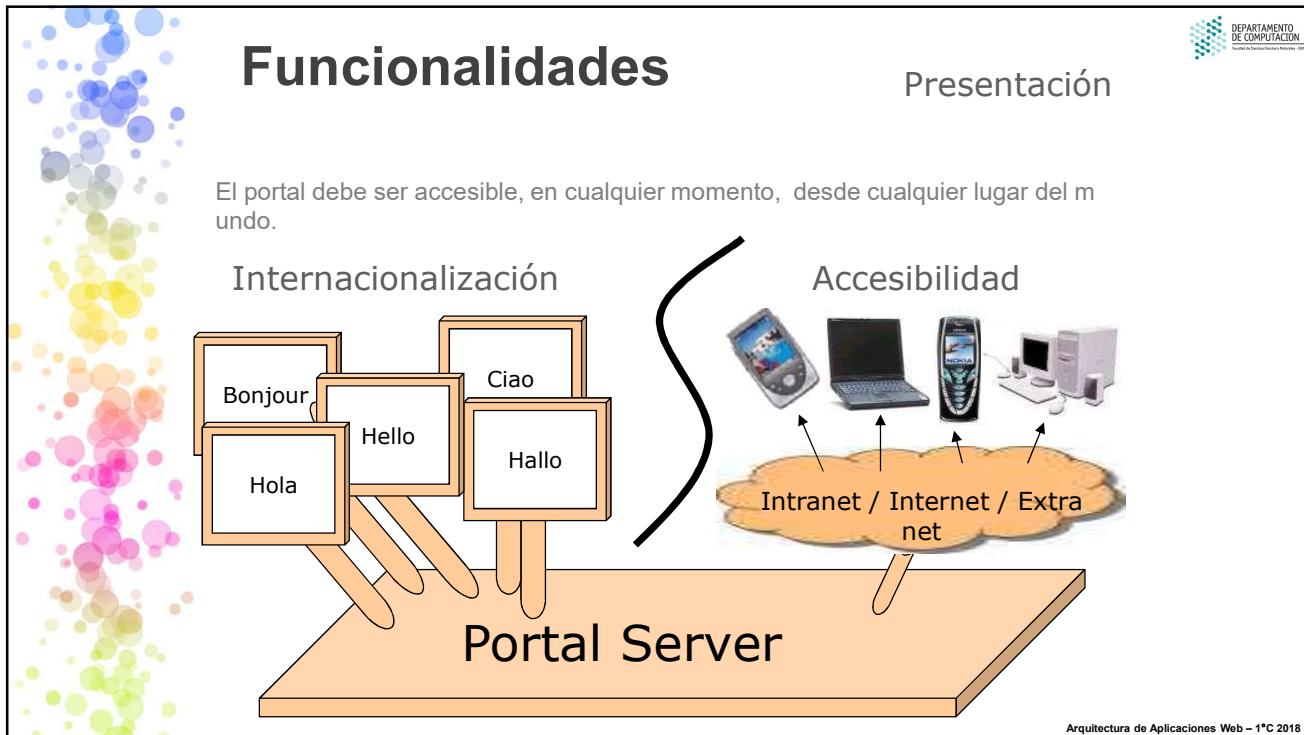
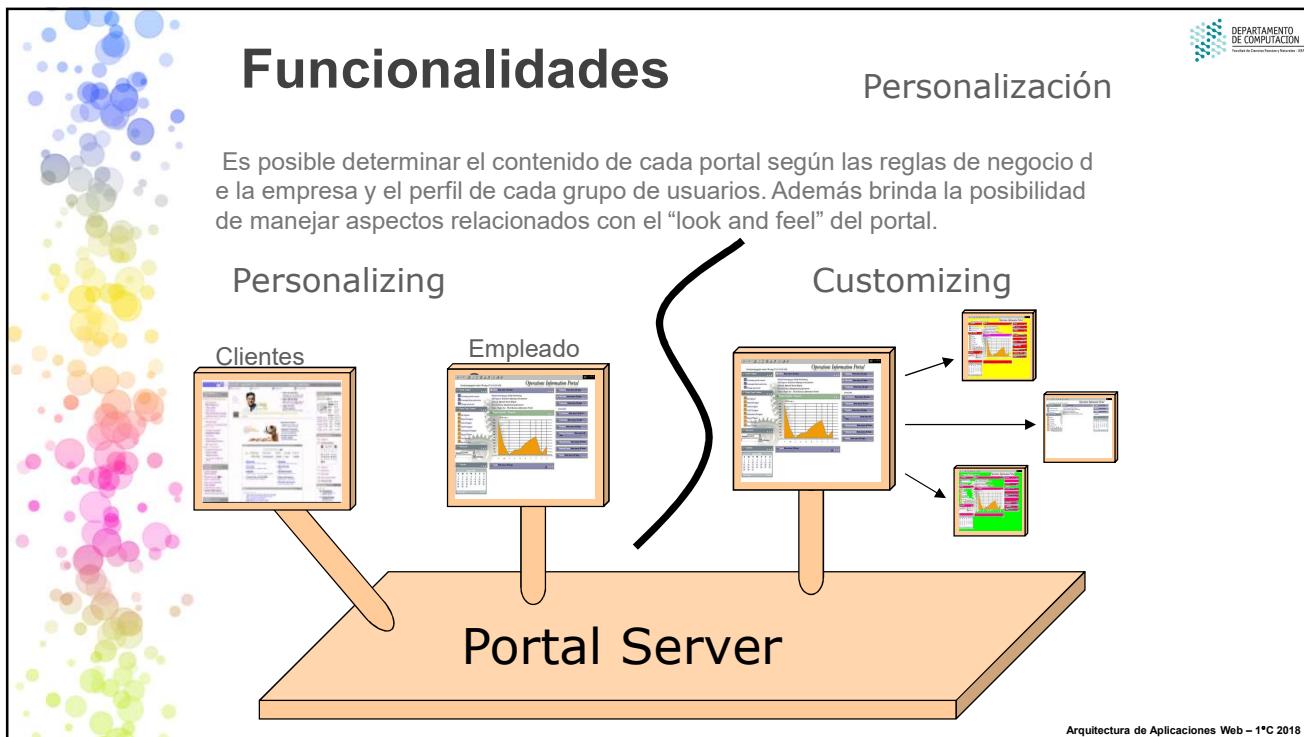
Estandarización

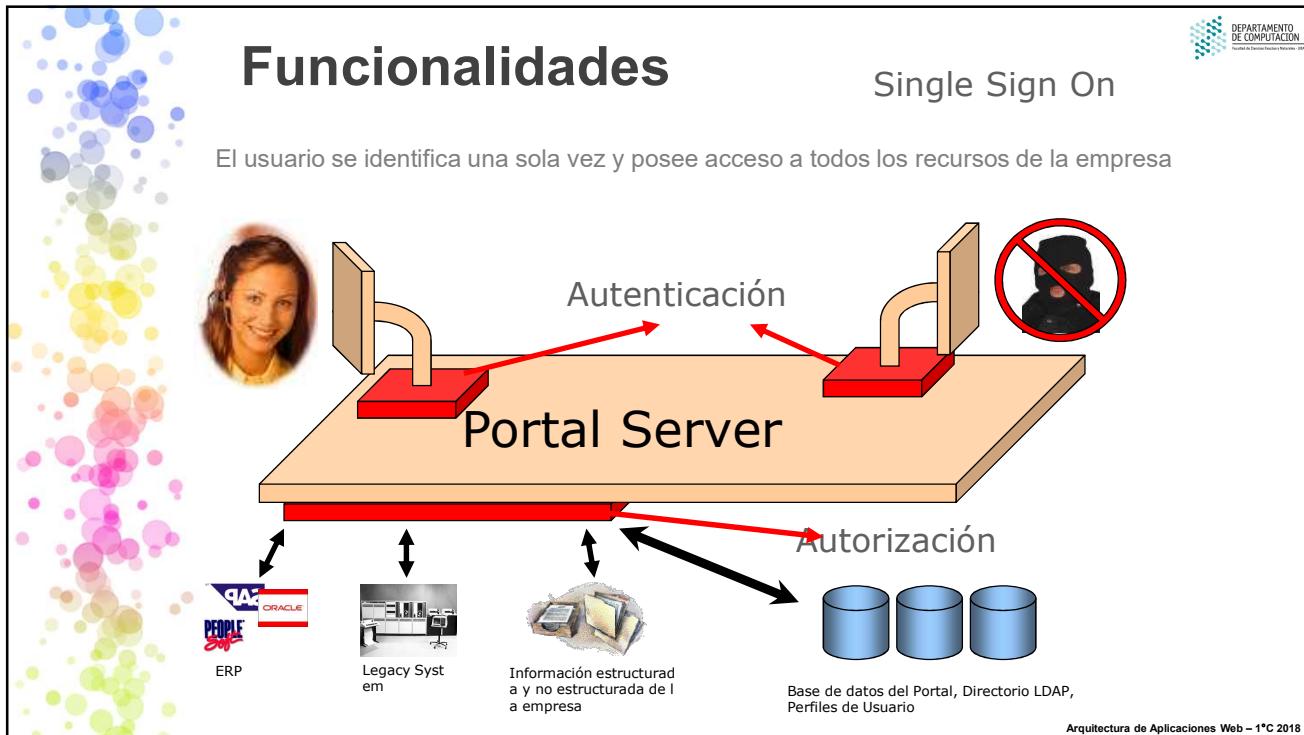
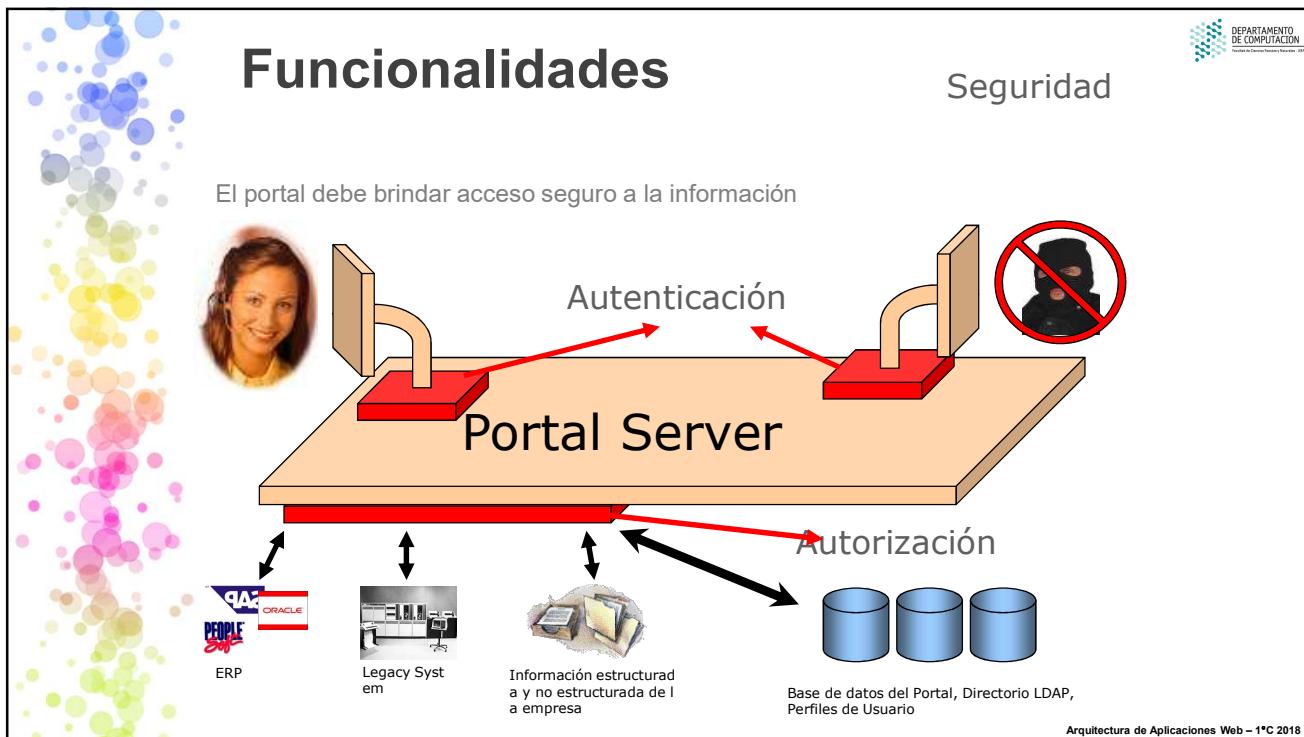
¿Qué servicios debe proveer?



Arquitectura de Aplicaciones Web – 1*C 2018







Funcionalidades

Búsqueda y Categorización

El portal debe facilitar la búsqueda y categorización de contenidos provenientes de diversas fuentes



Arquitectura de Aplicaciones Web – 1*C 2018

Funcionalidades

Publicar y Suscribir

Los usuarios pueden suscribirse a distintos eventos y de esta manera recibir notificaciones cuando esos eventos sucedan



Arquitectura de Aplicaciones Web – 1*C 2018

Funcionalidades

Portlets

- Un portal está compuesto por portlets
- Un portlet es un componente web, capaz de procesar requerimientos y generar contenido dinámico
- Es aplicación en sí misma

Portal Server

Arquitectura de Aplicaciones Web – 1*C 2018

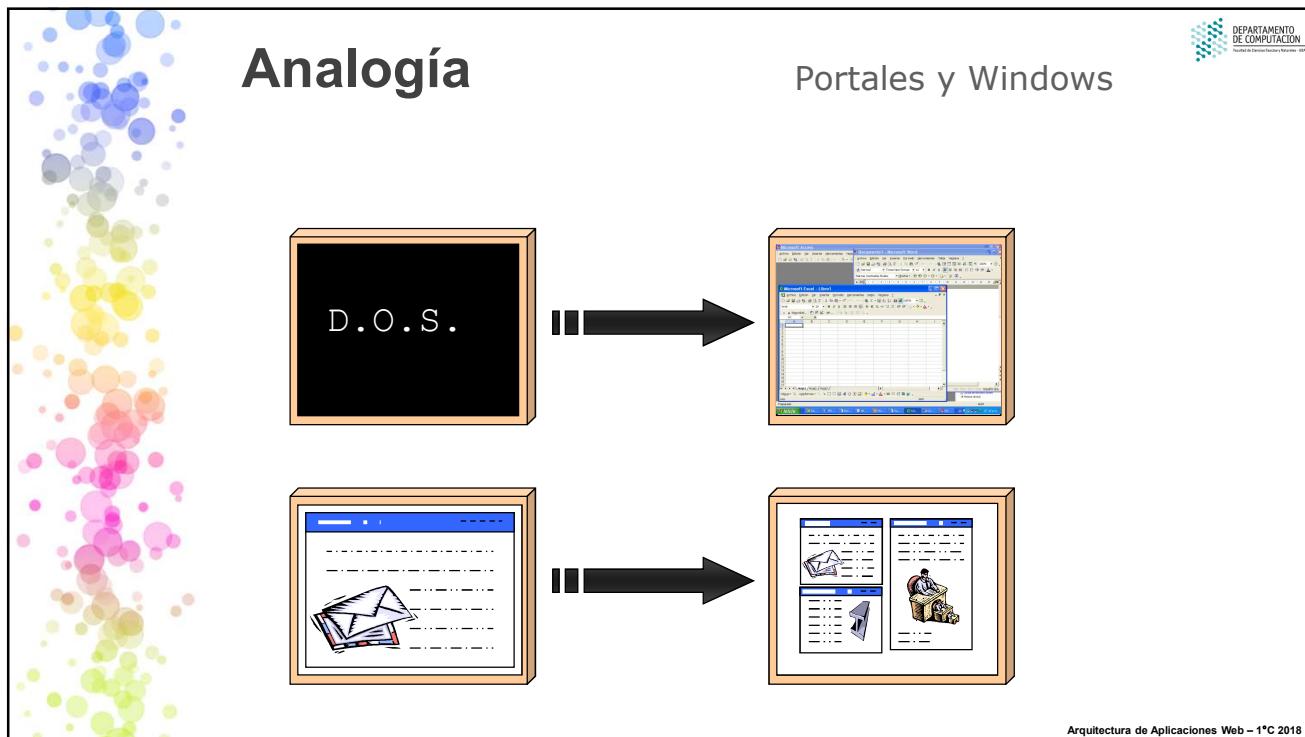
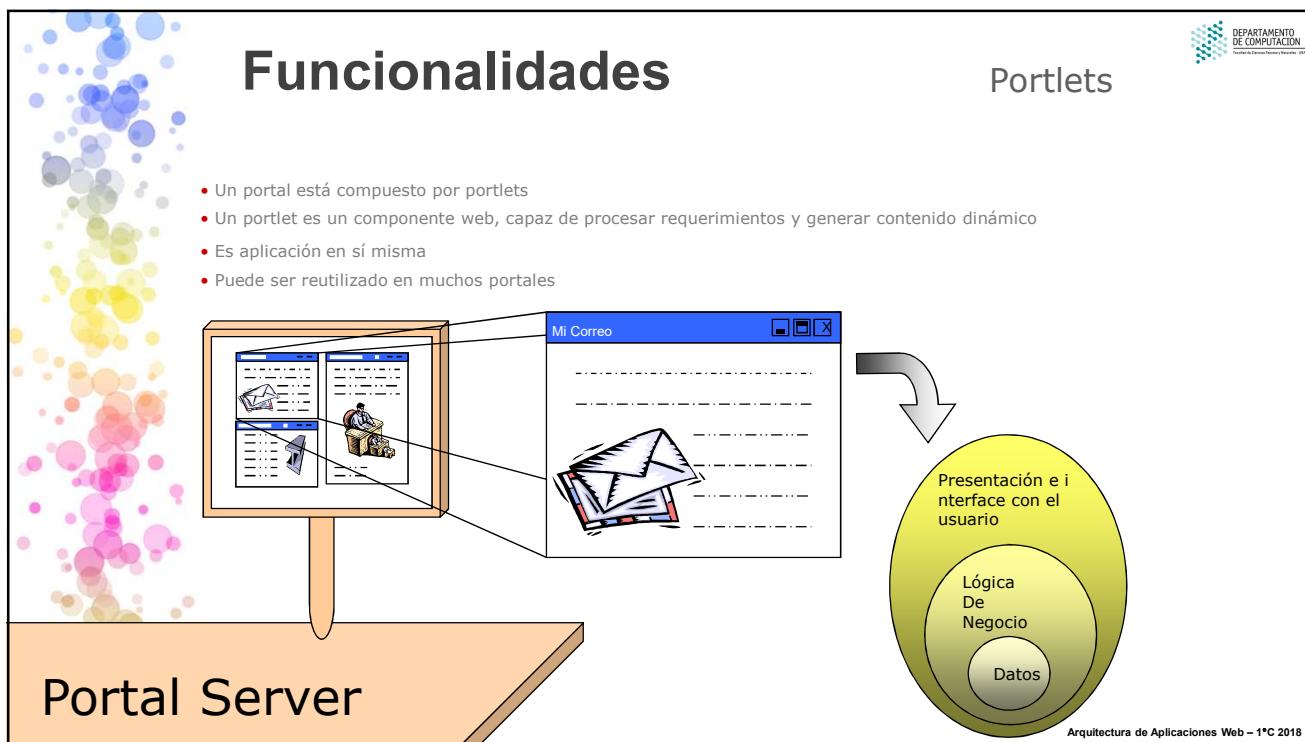
Funcionalidades

Portlets

- Un portal está compuesto por portlets
- Un portlet es un componente web, capaz de procesar requerimientos y generar contenido dinámico
- Es aplicación en sí misma

Portal Server

Arquitectura de Aplicaciones Web – 1*C 2018



Funcionalidades

Portlets

Gracias a su arquitectura, el administrador del portal puede determinar su contenido haciendo "drag and drop" de aplicaciones (portlets)

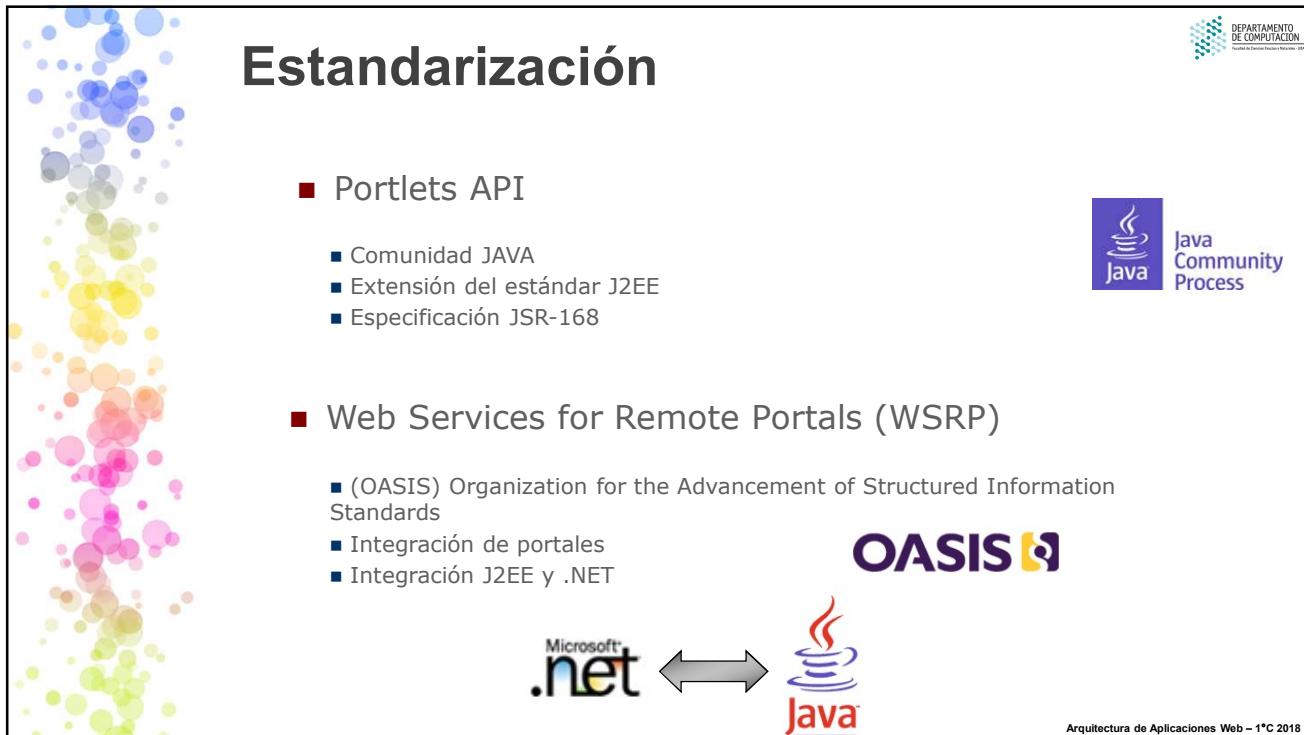
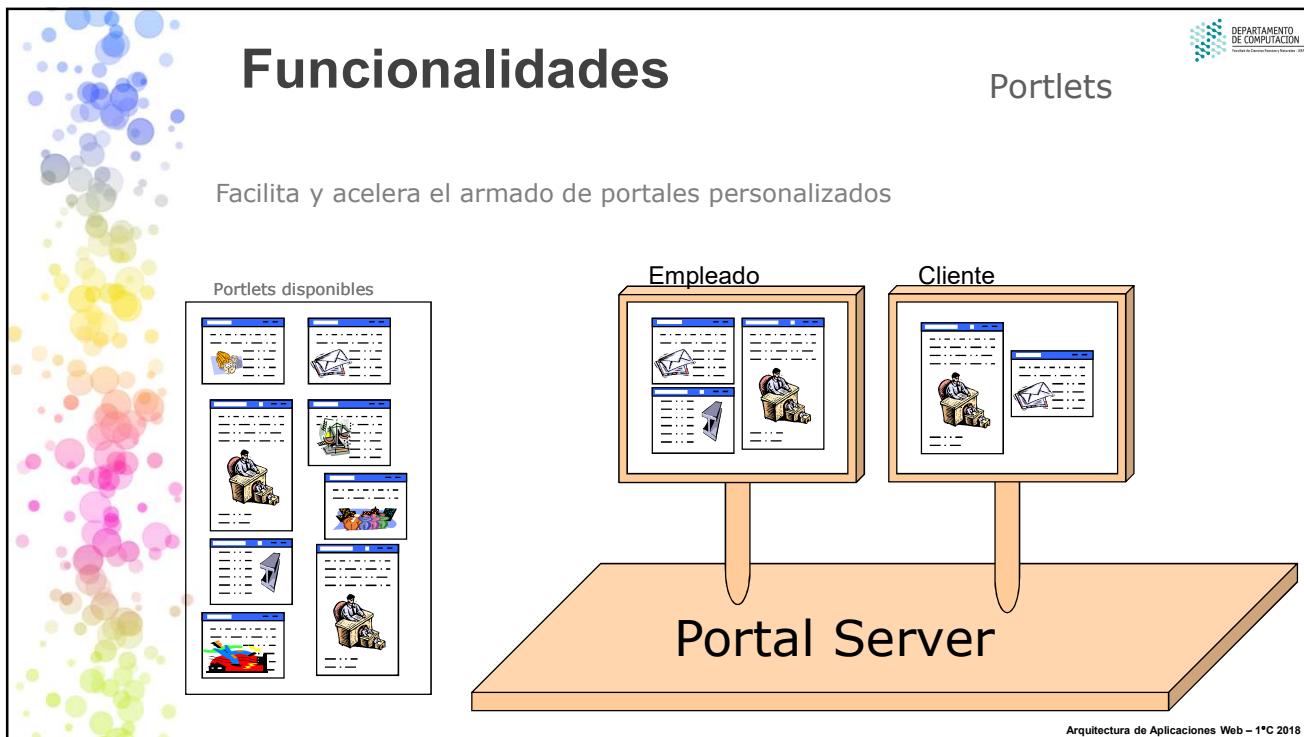
Arquitectura de Aplicaciones Web – 1*C 2018

Funcionalidades

Portlets

Facilita y acelera el armado de portales personalizados

Arquitectura de Aplicaciones Web – 1*C 2018



Tecnologías

- La carrera la empezaron a correr todos los grandes vendors

Microsoft SharePoint Portal Server

SAP NetWeaver

bea WEBLOGIC

APACHE PORTALS Jetspeed

IBM WebSphere Portal

ORACLE

Arquitectura de Aplicaciones Web – 1*C 2018

Mercado de Portales

		Desafiantes	Líderes
Habilidad de ejecución	Visión	Microsoft Novell Computer Associates webMethods SeeBeyond Technology Art Technology Group (ATG) • abaxX Technology • Day	IBM SAP Oracle Sun Microsystems PeopleSoft BroadVision Hummingbird Sybase Open Text BEA Systems Plumtree Software Vignette
		Jugadores de nicho	Visionarios

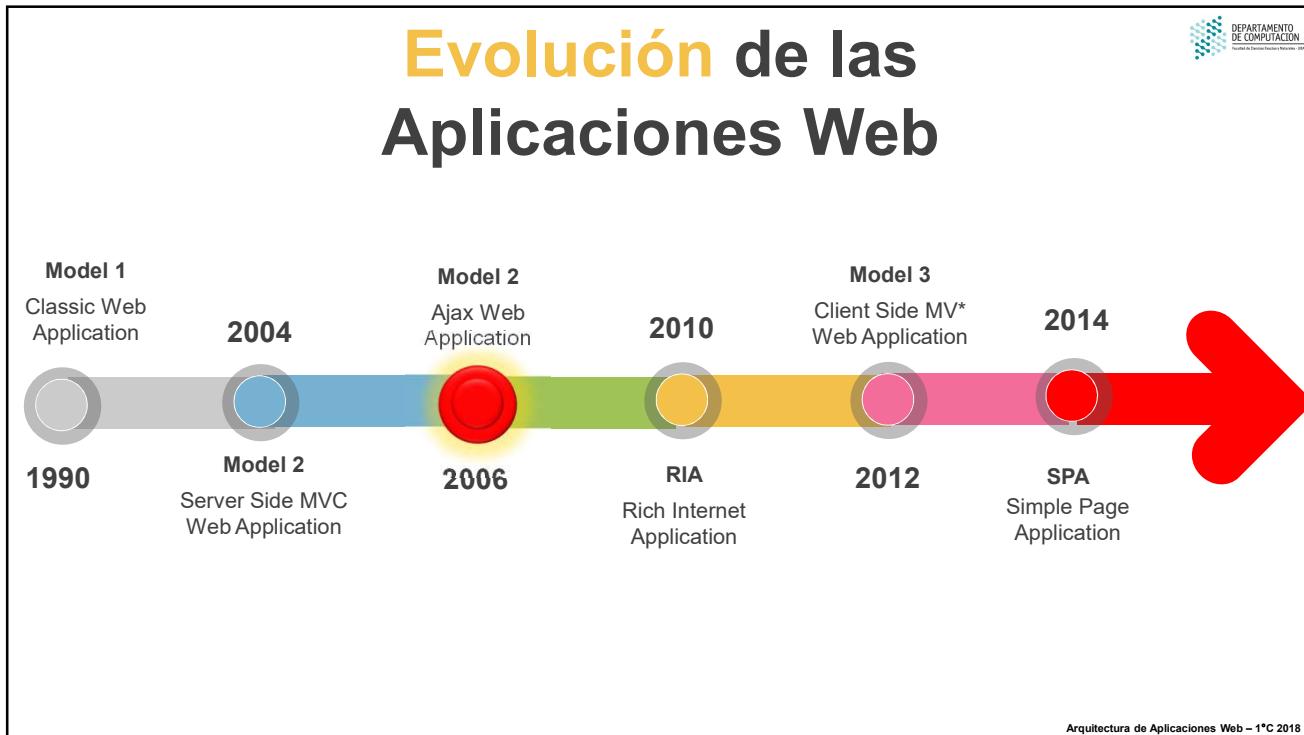
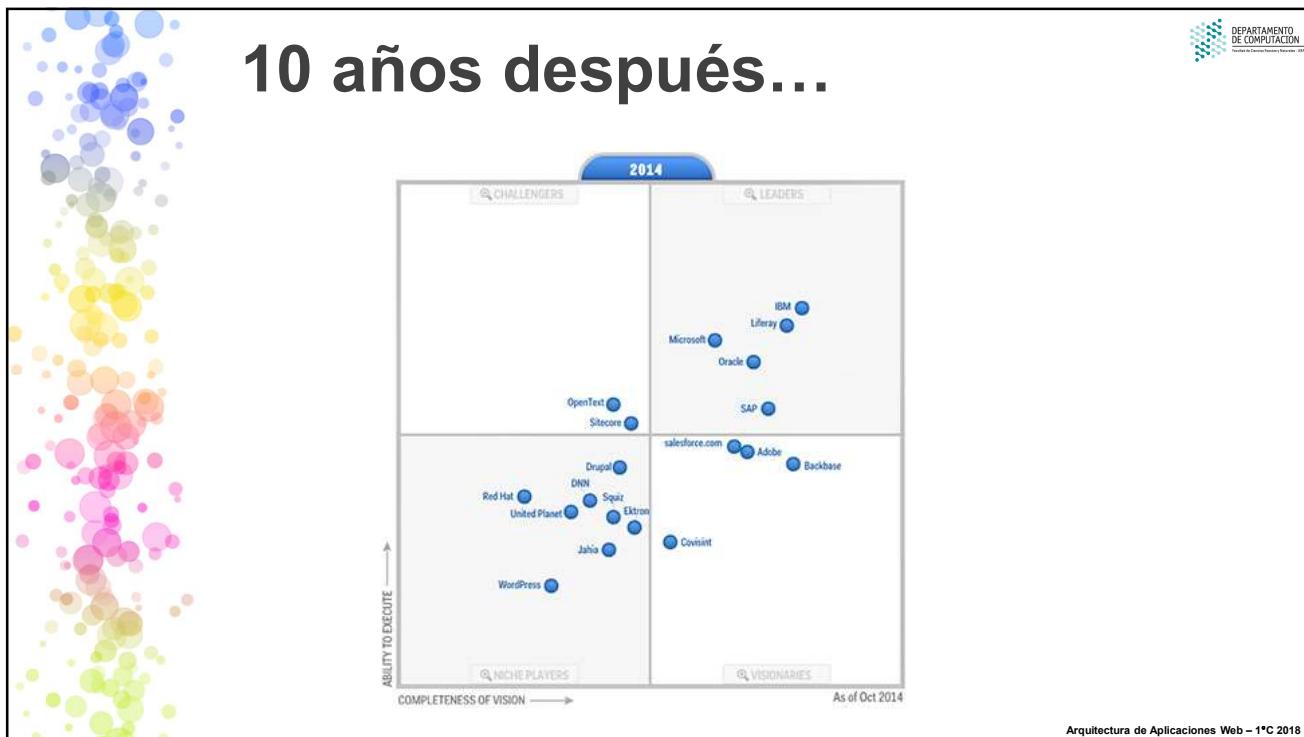
IBM WebSphere Portal

SAP NetWeaver

bea WEBLOGIC

Microsoft SharePoint Portal Server

Arquitectura de Aplicaciones Web – 1*C 2018



Ajax

- Asynchronous JavaScript and XML
- Inicialmente no fue un elemento *First Class*
- Surgió como esfuerzo de recortar el gap entre usabilidad e interactividad, entre las aplicaciones web y las aplicaciones desktop.
- No fue un lenguaje de programación
- No fue una nueva tecnología

Arquitectura de Aplicaciones Web – 1*C 2018

Ajax

Otra manera de pensar la programación Web



Arquitectura de Aplicaciones Web – 1*C 2018

Ajax

Otra manera de pensar la programación Web

Arquitectura de Aplicaciones Web – 1*C 2018

Entonces...

- Ahora las interacciones no son tan “request-response” por pantalla... sino...

Arquitectura de Aplicaciones Web – 1*C 2018

Otro detalle...

- Frameworks “server side” suman Ajax como elemento *First Class*

The slide features a decorative vertical column on the left side composed of a series of overlapping colored circles in shades of blue, yellow, pink, and green. In the top right corner is the logo of the "DEPARTAMENTO DE COMPUTACION Facultad de Ciencias Exactas y Naturales UBA". The footer at the bottom right reads "Arquitectura de Aplicaciones Web – 1*C 2018".

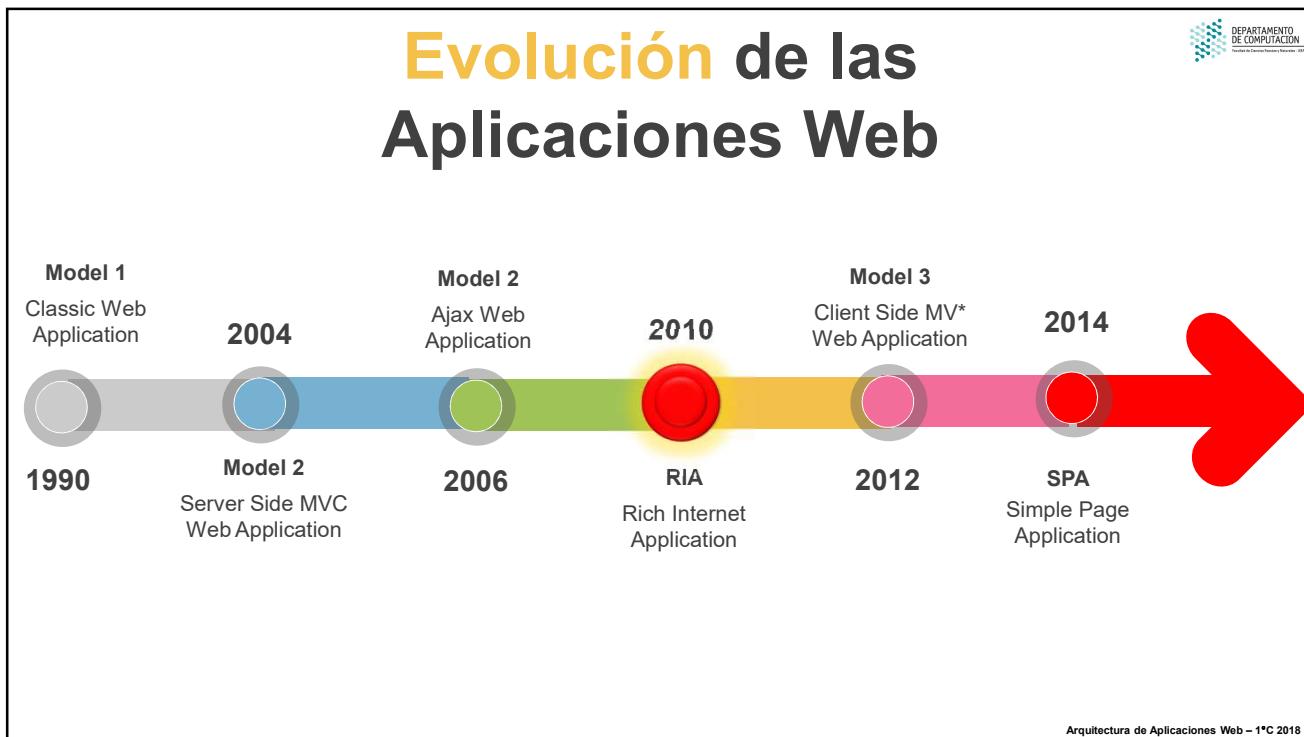
Y más MV*...

- Model View Controller y derivados...

The slide features a decorative vertical column on the left side composed of a series of overlapping colored circles in shades of blue, yellow, pink, and green. In the top right corner is the logo of the "DEPARTAMENTO DE COMPUTACION Facultad de Ciencias Exactas y Naturales UBA". The footer at the bottom right reads "Arquitectura de Aplicaciones Web – 1*C 2018".

The diagram illustrates three architectural patterns:

- MVC (Left):** Shows a Browser sending an HTTP REQUEST to a Controller. The Controller sends EXECUTION PARAMETERS to a Model and RESULTING DATA ARRAYS to a View. The View returns a GUI CONTENT to the Browser.
- MVVM (Center):** Shows a Model (Grundlage Logik und Daten) interacting with a View (Presentation Logic XAML) via a ViewModel (Presentation Logic). The View also interacts with the ViewModel.
- MVCF (Bottom):** Shows a Model filling a View, which then updates a Controller. The Controller writes back to the View and notifies the Model.



Mientras tanto...

- Las Rich Internet Applications (RIA) son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales.
- Las RIA surgen como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales.

The slide features a decorative background on the left side consisting of a vertical column of colorful, semi-transparent circular bubbles in shades of blue, yellow, pink, and green. On the right side, there is a collection of logos for various Rich Internet Application frameworks and technologies, including Adobe AIR, Mozilla Prism, Fx (Firefox), Microsoft Silverlight, AJAX (Asynchronous JavaScript And XML), OpenLaszlo, and a logo featuring a coffee cup with steam.

Arquitectura de Aplicaciones Web – 1*C 2018

Desktop vs Web



Arquitectura de Aplicaciones Web – 1*C 2018

Desktop vs Web

Aplicaciones de escritorio

- Ventajas:
 - Pueden ser más robustas
 - Tiempo de respuesta más rápido
 - Se puede hacer cualquier cosa que permita el SO (cuestión gráfica, control total de las entradas del usuario al momento de capturar)
- Desventajas:
 - Requiere instalación
 - Generalmente se hacen para un SO específico
 - Se requiere actualizar en cada cliente

Arquitectura de Aplicaciones Web – 1*C 2018

Desktop vs Web

Aplicaciones web

- **Ventajas**

- *Se puede usar desde cualquier lugar.*
- *No requiere hacer actualizaciones en los clientes.*
- *No hay problemas de incompatibilidad entre versiones, porque todos trabajan con la misma.*
- *Se centralizan los respaldos.*
- *No necesita instalar nada en el cliente.*
- *No se obliga a usar cierto SO.*

- **Desventajas**

- *Requiere conexión a la red.*
- *Se pierde tiempo de desarrollo haciéndola compatible con los distintos navegadores, los frameworks ayudan a solventar estos problemas.*
- *Su tiempo de respuesta es más lento, esto se mejoró usando tecnologías como AJAX haciéndolas casi tan rápidas como las de escritorio.*

Arquitectura de Aplicaciones Web – 1*C 2018

Aplicaciones RIA

- Son aplicaciones web que tienen la mayoría de las características de las aplicaciones tradicionales, estas aplicaciones utilizan un “navegador web” estandarizado para ejecutarse.
- Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones de escritorio.
- Buscan mejorar la experiencia del usuario
- Normalmente en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces, a recargar la misma página con un mínimo cambio.

Arquitectura de Aplicaciones Web – 1*C 2018

¿Que es Flex?

- Open Source Framework para desarrollar aplicaciones Web interactivas.
- Las aplicaciones Flex funcionan de manera consistente en la mayoría de los web browsers.
- Se ejecuta en la plataforma Adobe Flash.
- Dentro de la familia RIA (Rich Internet Applications).
- Nueva arquitectura web.
- Capa de presentación de la aplicación Web.

Arquitectura de Aplicaciones Web – 1*C 2018

¿Por que usar Flex?

- Multiplataforma
 - Las aplicaciones Flex corren sobre cualquier browser que soporte Flash Player
 - También corren en el escritorio utilizando Adobe AIR.
 - Adobe AIR puede acceder a datos locales y recursos del sistema en el escritorio.
- Interfaz interactiva para el usuario
 - Framework construido sobre Flash Player para brindar una experiencia única al usuario.

Arquitectura de Aplicaciones Web – 1*C 2018

Tecnología de Servidor

- ASP.NET
- ColdFusion
- J2EE
- PHP
- Ruby
- Perl

Arquitectura de Aplicaciones Web – 1ºC 2018

Componentes que integran Flex

- ActionScript 3.0
- Flash Player 9
- Flex Framework (SDK)
- Flex Builder 3
- MXML (Multimedia eXtended Markup Language)
- E4X (ECMAScript for XML)
 - ECMAScript 4 (Abandonado)
 - ECMAScript 6 → Typescript
 - ECMAScript 7 en progreso

Arquitectura de Aplicaciones Web – 1ºC 2018

¿Que es MXML?

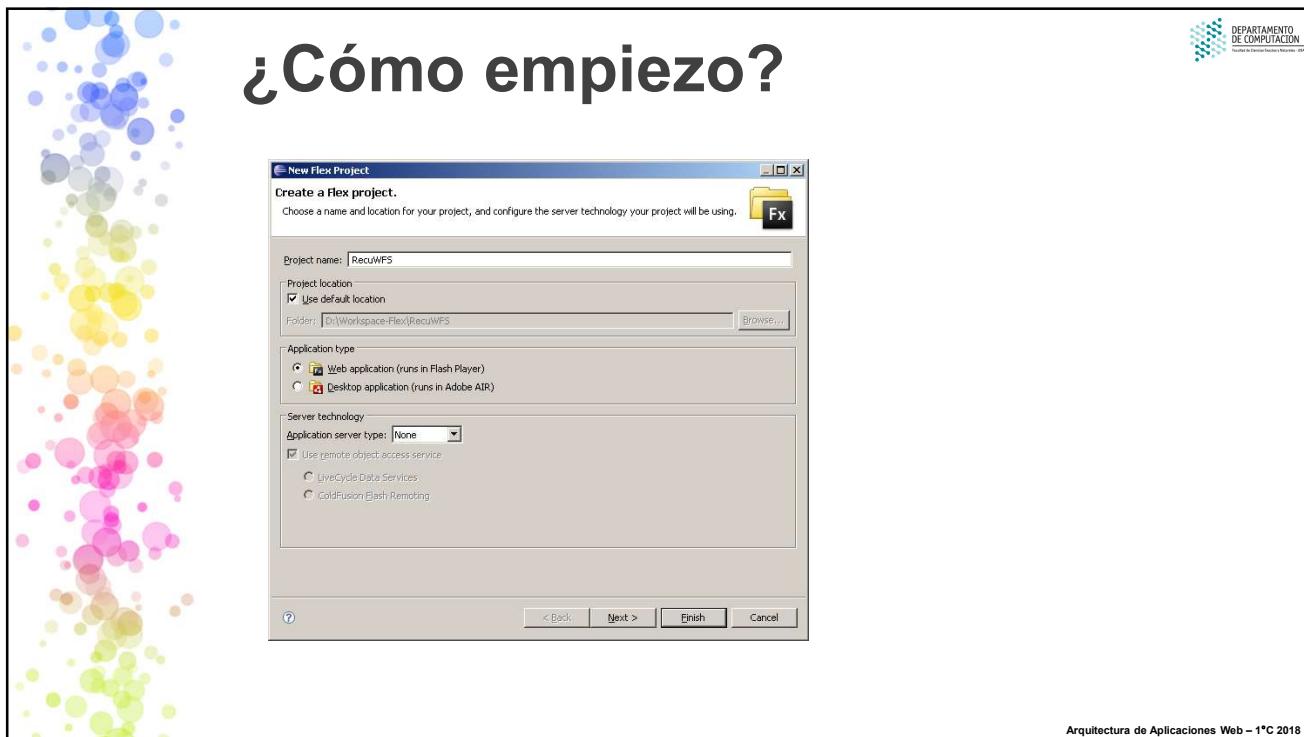
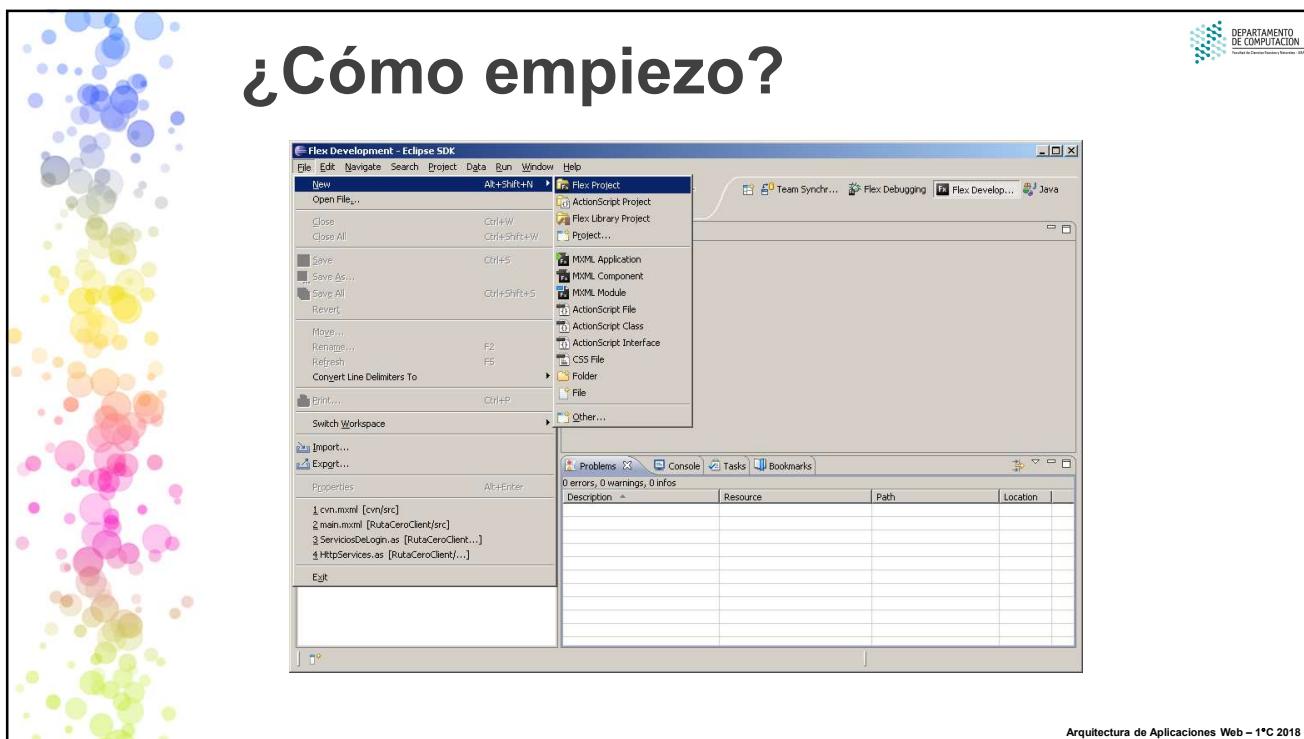
- Adobe Flex 3
- ActionScript 3.0
- **MXML**
- HTML
- JavaScript
- HTML

Arquitectura de Aplicaciones Web – 1ºC 2018

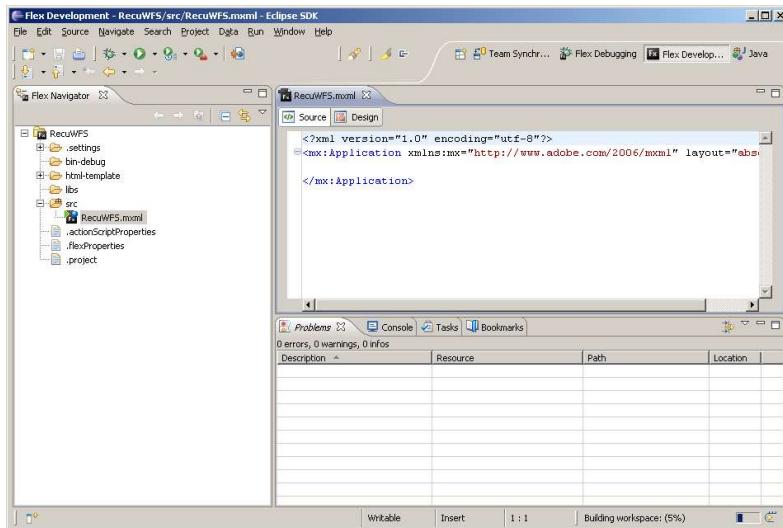
¿Como empiezo?

- Descargar Adobe Flex SDK o Adobe Flex Builder.
- Adobe Flex Builder es el IDE para desarrollar aplicaciones Flex.
- Basado en Eclipse.
- Orientado a Formularios.
- Diseñador Visual.

Arquitectura de Aplicaciones Web – 1ºC 2018



¿Cómo empiezo?



¿Cómo desarrollo en Flex?

- Usando el Flex Builder (IDE basado en Eclipse) o algún editor a elección.
- Flex Builder es el más utilizado para construir aplicaciones Flex.
- En cualquier caso se utiliza el Flex SDK, para generar las aplicaciones Flex.

Componentes gráficos Flex

- Básicos
 - Button, Label, TextInput, CheckBox, Image, RadioButt on, ...
- Listas
 - DataGrid, AdvancedDataGrid, List, Tree, ComboBox, ...
- Layout
 - Canvas, HBox, VBox, TitleWindow, Panel, ...
- Charts
 - AreaChart, BarChart, ColumnChart, LineChart, PieChart, PlotChart, ...

Arquitectura de Aplicaciones Web – 1ºC 2018

Adobe Flex Builder

- Potente editor de texto.
- Refactoring.
- Interfaz visual de diseño.
- Soporte nativo para Adobe AIR.
- Asistentes para Backend.

Arquitectura de Aplicaciones Web – 1ºC 2018

Una aplicación Flex

- MXML
 - Lenguaje basado en XML para describir los componentes visuales de la aplicación.
- ActionScript
 - Lenguaje orientado a objetos similar a Java para escribir el comportamiento de la aplicación.



Arquitectura de Aplicaciones Web – 1ºC 2018

Desarrollo en Flex Builder.



Arquitectura de Aplicaciones Web – 1ºC 2018

Features

- ActionScript 3.0
- MXML
- Bindable
- Data providers (Array/ArrayCollection/XML)
- HTTPService
- MXML Components
- States

Arquitectura de Aplicaciones Web – 1ºC 2018

ActionScript

- Similar a Java, C#.
- Orientado a objetos.
- Clases, interfaces, funciones anónimas.
- Eventos.

Arquitectura de Aplicaciones Web – 1ºC 2018

ActionScript: Eventos

- Flex (y por lo tanto ActionScript) es un lenguaje orientado a eventos.
- El cliente se subscribe al evento que quiere atender.
- Cuando el usuario desencadena una acción que tiene algún subscriptor, este es invocado para realizar alguna tarea deseada. (El subscriptor es mejor conocido como Event Handler)

Arquitectura de Aplicaciones Web – 1ºC 2018

ActionScript

- Lenguaje de programación orientado a objetos usado en aplicaciones Flash (por lo tanto Flex también dispone de ActionScript)

```
/* Hello World module */
var HelloString = "Hello World!";

function getHello(name:String) {
    var welcome:String = " Welcome " + name;
    return welcome + HelloString;
}
```

Arquitectura de Aplicaciones Web – 1ºC 2018

ActionScript

```
public function doInit() : void
{
    goodbyeButton.addEventListener("click", goodbye);
}

public function goodbye(event: Event) : void
{
    myPanel.visible = false;
}
```

Arquitectura de Aplicaciones Web – 1*C 2018

ActionScript

- Se puede incrustar en código MXML.

```
<Application>
    <Style source="style.css"/>
    <Script>
        public function goodbye() : void {
            myPanel.visible = false;
        }
    </Script>

    <Panel id="myPanel" layout="absolute" width="80%" height="80%">
        <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
        <Button label="goodbye" right="30" bottom="41" click="goodbye()"/>
    </Panel>
</Application>
```

Arquitectura de Aplicaciones Web – 1*C 2018

MXML

- Basado en XML

```
<mx:Application>
    <mx:WebService id="ws" wsdl="catalog.wsdl"/>
    <mx:Button label="Get Data" click="ws.getProducts()"/>
    <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
    <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1ºC 2018

MXML - Componentes

```
<mx:Application>
    <mx:WebService id="ws" wsdl="catalog.wsdl"/>
    <mx:Button label="Get Data" click="ws.getProducts()"/>
    <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
    <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1ºC 2018

MXML – Identificadores

```
<mx:Application>
  <mx:WebService id="ws" wsdl="catalog.wsdl"/>
  <mx:Button label="Get Data" click="ws.getProducts()"/>
  <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
  <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1ºC 2018

MXML - Propiedades

```
<mx:Application>
  <mx:WebService id="ws" wsdl="catalog.wsdl"/>
  <mx:Button label="Get Data" click="ws.getProducts()"/>
  <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
  <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1ºC 2018

MXML - Eventos

```
<mx:Application>
    <mx:WebService id="ws" wsdl="catalog.wsdl"/>
    <mx:Button label="Get Data" click="ws.getProducts()"/>
    <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
    <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1*C 2018

MXML - Bindings

```
<mx:Application>
    <mx:WebService id="ws" wsdl="catalog.wsdl"/>
    <mx:Button label="Get Data" click="ws.getProducts()"/>
    <mx:DataGrid dataProvider="{ws.getProducts.result}"/>
    <mx:LineChart dataProvider="{ws.getProducts.result}"/>
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1*C 2018

Bindable



DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Flex Development - RutaCeroClient/src/rc/componentes/RechazoQueryBox.mxml - Eclipse SDK

File Edit Source Navigate Search Project Data Run Window Help

Source Design

Expresión Bindable

```
<mx:Script>
    <![CDATA[
        ]]>
</mx:Script>
<mx:HBox bottom="10" left="10" right="10" horizontalAlign="center">
    <mx:Button id="_actualizarButton" label="Actualizar" click="_actualizarHa
        <mx:icon>@Embed(source='../../imagenes/actions/repeat.png')</mx:icon>
    </mx:Button>
    <mx:Button id="_modificarButton" label="Modificar" click="_modificarHandle
        &enabled="(_rechazozADG.selectedItem != null)">
        <mx:icon>@Embed(source='../../imagenes/actions/note_edit.png')</mx:ic
    </mx:Button>
    <mx:Button id="_eliminarButton" label="Eliminar" enabled="(_rechazozADG.s
        <mx:icon>@Embed(source='../../imagenes/actions/note_remove.png')</mx:
    </mx:Button>
</mx:HBox>
<RCAdvancedDataGrid id="_rechazozADG" designViewDataType="flat" top="10" bott
    <columns>
```

Writable Insert

Arquitectura de Aplicaciones Web – 1ºC 2018

Data providers



DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- Varios controles utilizan proveedores de datos para mostrar resultados.
- XML es un tipo de data provider. (También está ArrayCollection, Array).
- Controles que usan proveedores de datos:
 - DataGrid
 - List
 - Menu
 - Tree

Arquitectura de Aplicaciones Web – 1ºC 2018

Data providers

```
<mx:Script>
<![CDATA[
[Bindable]
public var myArray:Array = ["AL", "AK", "AR"];
]]
</mx:Script>
<mx:ComboBox id="c1" dataProvider="{myArray}" />
```

Arquitectura de Aplicaciones Web – 1*C 2018

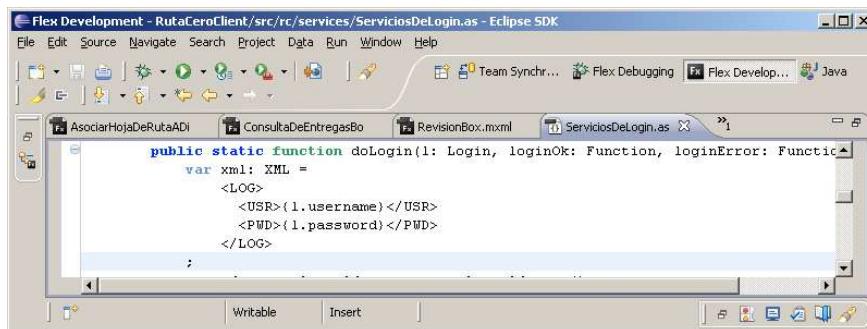
Data providers

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml initialize="initData()">
<mx:Script>
<![CDATA[
import mx.collections.*;
[Bindable]
public var stateArray:ArrayCollection;
public function initData():void {
stateArray=new ArrayCollection( [{label:"AL", data:"Montgomery"}, {label:"AK", data:"Juneau"}, {label:"AR", data:"Little Rock"}]); }
]]
</mx:Script>
<mx:ComboBox id="myComboBox" dataProvider="{stateArray}" />
</mx:Application>
```

Arquitectura de Aplicaciones Web – 1*C 2018

Data providers - XML

- Objetos de primer orden en el lenguaje.
- Manipulación más sencilla.
- Verificación de formato XML en tiempo de compilación.



Arquitectura de Aplicaciones Web – 1ºC 2018

XML - Operadores

```
var xml: XML = <RC>
    <HDR>
        <COD>0</COD>
        <DES>Pase</DES>
    </HDR>
</RC>;
var cod:String= xml.HDR.COD; // Devuelve "0"
var des:String= xml.HDR.DES; // Devuelve "Pase"
```

Arquitectura de Aplicaciones Web – 1ºC 2018

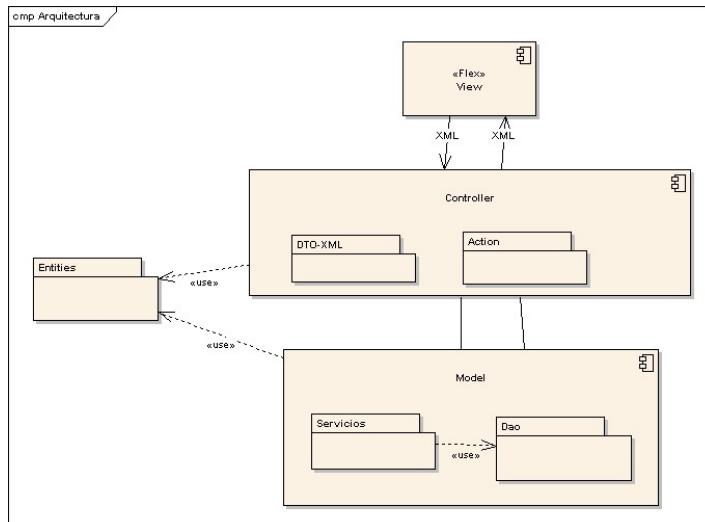
XMLList

```
var xml: XML = <RC>
  <HDR><COD>0</COD><DES>Ok</DES></HDR>
  <DATA>
    <ITEMS>
      <ITEM><ID>1</ID><DES>D1</DES></ITEM>
      <ITEM><ID>2</ID><DES>D2</DES></ITEM>
      <ITEM><ID>3</ID><DES>D3</DES></ITEM>
      <ITEM><ID>4</ID><DES>D4</DES></ITEM>
    </ITEMS>
  </DATA>
</RC>
var xmllist: XMLList = xml.DATA.ITEMS.ITEM;
```

Comunicación

- HTTPService es una de las formas de comunicarse con un servidor de datos.
- Se puede transmitir XML (de hecho cualquier stream de datos)
- Hay otras formas: WebServices, RemoteObjects.
- Acá vamos a ver HTTPService.

Arquitectura clásica MVC



Arquitectura de Aplicaciones Web – 1*C 2018

HTTPService

- La clase HTTPService permite interactuar con un Web Server.
- Usualmente se utiliza con XML como objeto de transporte.
- El Web Server puede ser desarrollado con cualquier tecnología que permita crear XML. (Java, ASP.NET, PHP, Ruby, Python, Perl, etc).

Arquitectura de Aplicaciones Web – 1*C 2018

Backend

- XML via HTTP/HTTPS requests.
- Web Services.
- POJO
 - JavaBeans
 - DAO
 - Hibernate
 - EJBs
- ColdFusion Remote Objects
- Flex Data Services 2

Arquitectura de Aplicaciones Web – 1*C 2018

HTTPService

- Crear un tag dentro de la aplicación o componente Flex.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
3
4     <mx:HTTPService id="myXMLdocument" url="http://www.theflexgroup.org/menu.xml"/>
5
6 </mx:Application>
7
```

Arquitectura de Aplicaciones Web – 1*C 2018

HTTPService

- Parámetros más usados
 - id: Nombre del componente.
 - url: URL completa para el HTTP request.
 - result: Evento que será invocado cuando llegue el response HTTP (Recordar que el manejo es asíncronico)
 - fault: Evento que será invocado cuando haya algún error en la comunicación.
 - showBusyCursor: Muestra el cursor ocupado durante el response/request HTTP.
 - method: Forma de enviar el request (GET o POST).

Arquitectura de Aplicaciones Web – 1ºC 2018

HTTPService

- Método send(parameters: Object = null)
 - Envía el request HTTP.
 - Método asíncronico.
 - Cuando se completa el request se invoca a los listener (RESULT o FAULT según el caso).
 - Se puede enviar datos en el parámetro ‘parameters’.
 - Este parámetro también puede ser un XML.

Arquitectura de Aplicaciones Web – 1ºC 2018

Componentes

- Funcionalidad repetida en la interfaz de usuario sugiere Flex Components.
- Piezas de código creadas por el usuario para crear componentes que interactúan con el usuario más pequeños.
- Se puede crear a partir de cualquier componente Flex: Button, Canvas, TextInput, Panel, etc.

Arquitectura de Aplicaciones Web – 1ºC 2018

Componentes

```
<Application>
    <Script>
        function initStartPanel() {...}
        function initDetailPanel() {...}
    </Script>
    <HBox>
        <Panel id="startPanel" creationComplete="initStartPanel()">
            ...
        </Panel>
        <Panel id="detailPanel" creationComplete="initDetailPanel()">
            ...
        </Panel>
    </HBox>
</Application>
```

Arquitectura de Aplicaciones Web – 1ºC 2018



Componentes

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

MyApp.mxml

```
<Application>
  <HBox>
    <StartPanel id="startPanel"/>
    <DetailPanel id="detailPanel"/>
  </HBox>
</Application>
```

StartPanel.mxml

```
<Panel creationComplete="initStartPanel()">
  <Script>
    function initStartPanel() {...}
  </Script>
  ...
</Panel>
```

Arquitectura de Aplicaciones Web – 1*C 2018



States

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

```
<Application initialize="doInit()">
  <states>
    <State name="small">
      < SetProperty target="{myPanel}" property="height" value="200"/>
      < SetProperty target="{myPanel}" property="width" value="300"/>
    </State>
  </states>
  <Style source="style.css"/>
  <Script source="Step5_code.as"/>
  <Panel id="myPanel" layout="absolute" width="80%" height="80%">
    <TextArea text="Hello, world!" top="10" bottom="71" left="10" right="30"/>
    <Button id="goodbyeButton" label="goodbye" right="30" bottom="41"/>
    <Button id="smallButton" label="small" left="10" bottom="41"/>
  </Panel>
</Application>
```

Arquitectura de Aplicaciones Web – 1*C 2018

States

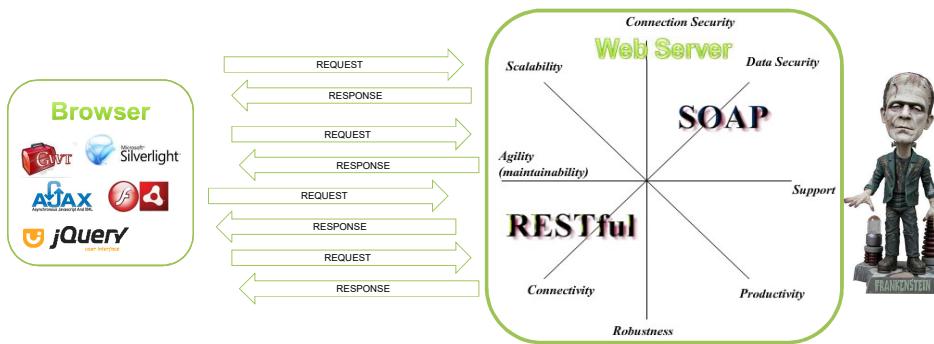
```
public function doInit() : void
{
    goodbyeButton.addEventListener("click", goodbye);
    smallButton.addEventListener("click", makeSmall);
}

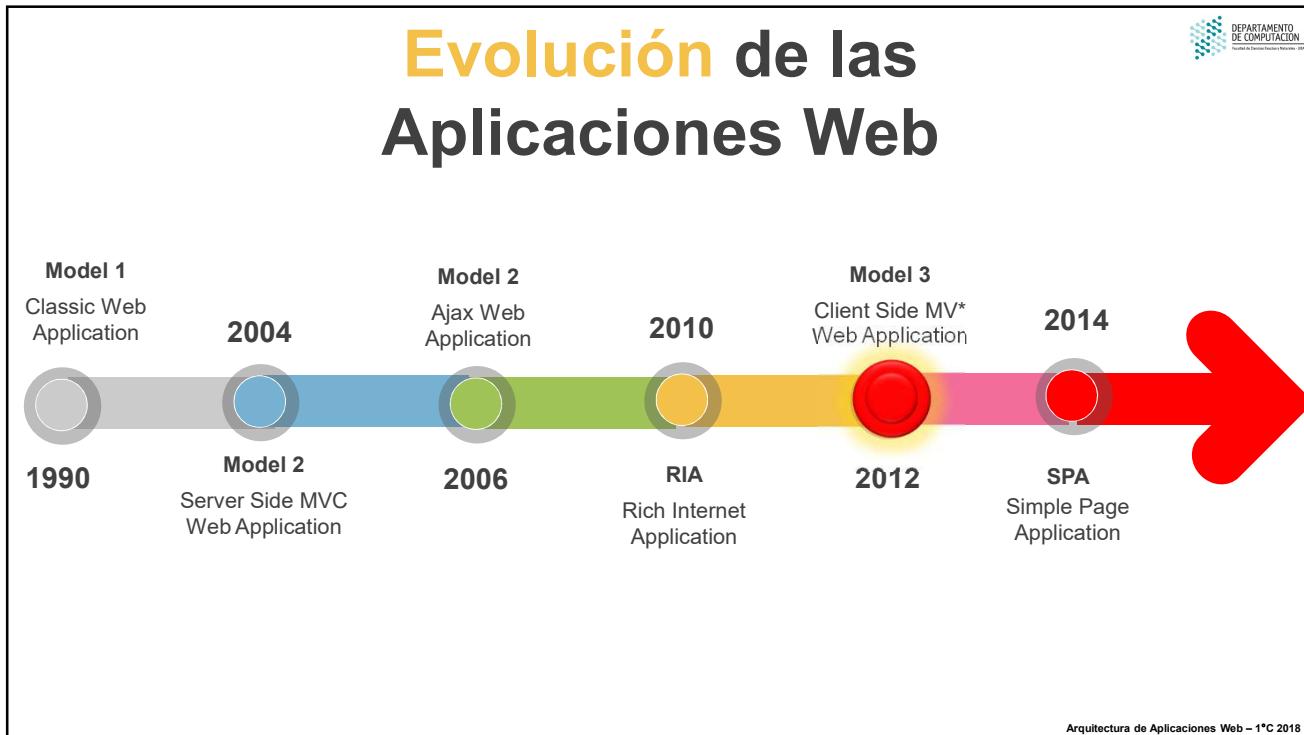
public function goodbye(event: Event) : void
{
    myPanel.visible = false;
}

public function makeSmall(event: Event) : void
{
    currentState = "small";
}
```

Buscando a Frankie...

- Muchas alternativas de arquitectura





Evolución



- La unión hace la fuerza...



Arquitectura de Aplicaciones Web – 1ºC 2018

Evolución



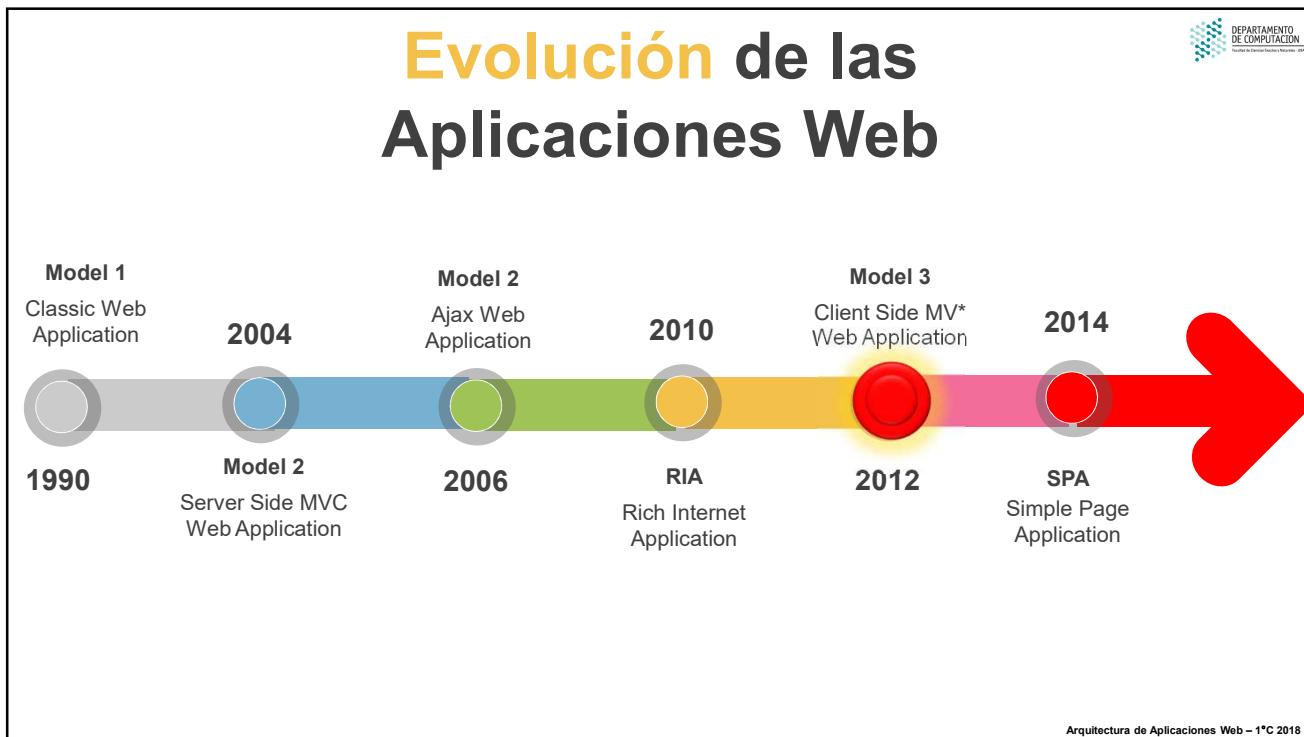
- La unión hace la fuerza...
- El fin de una época...



Arquitectura de Aplicaciones Web – 1ºC 2018



Arquitectura de Aplicaciones Web – 1ºC 2018



Arquitectura de Aplicaciones Web – 1ºC 2018

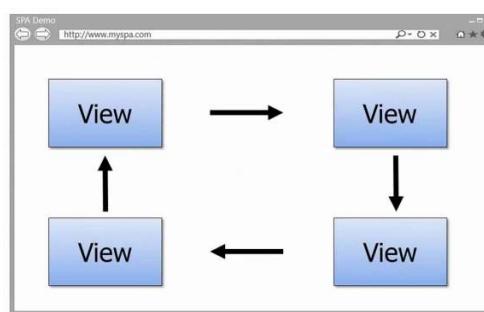
Evolución

- La unión hace la fuerza...
- El fin de una época...
- Y el nacimiento de otros frameworks



Single Page Applications

- En una SPA el usuario no navega por un sistema de enlaces tradicionales si no que en su lugar, mediante el uso de JavaScript, Ajax, HTML5 o una combinación de las anteriores, se actualiza lo que el usuario ve siempre desde la misma página (sin cambiar de URL ni refrescar el contenido entero).



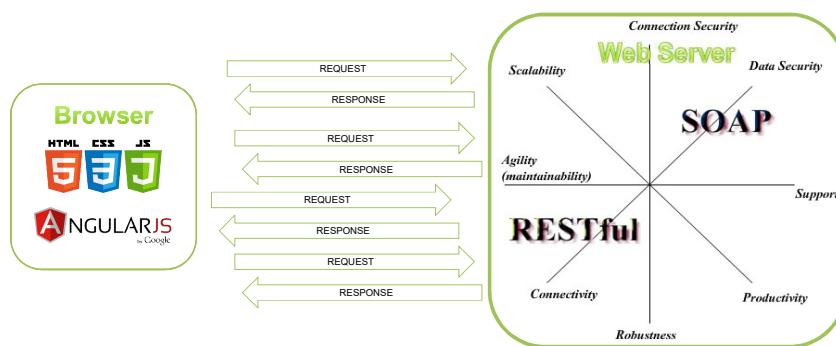
Single Page Applications

- Aplicaciones web que se ejecutan en una única página, logrando así una experiencia de usuario más cercana a una aplicación de escritorio.

Arquitectura de Aplicaciones Web – 1ºC 2018

Y del otro lado?

- REST o NO REST



Arquitectura de Aplicaciones Web – 1ºC 2018

SOAP



- WSDL (Web Services Description Language)
 - Permite especificar en XML las operaciones y tipos de datos de un servicio web.
 - Independiente del lenguaje de programación.
 - Estandarizado por el W3C
- Permiten especificar las operaciones expuestas por el servicio, junto con sus parámetros de entrada y sus valores de respuesta.
- El compilador de WSDL permite generar
 - Un “Stub” (Proxy) del objeto remoto (cliente)
 - Un Skeleton (Adapter) para implementar el interfaz remoto (servidor)
- Estilo arquitectónico “RPC”.

Arquitectura de Aplicaciones Web – 1ºC 2018

REST



- REST, Representational State Transfer, fue ganando amplia adopción en toda la web como una alternativa más simple a SOAP y a los servicios web basados en el Web Services Descripción Language (WSDL)

Arquitectura de Aplicaciones Web – 1ºC 2018

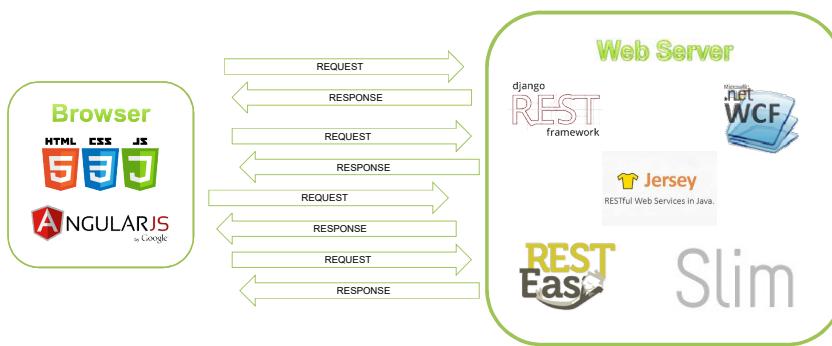
Los 4 principios de REST

- Utiliza los métodos HTTP de manera explícita.
- No mantiene estado.
- Expone URIs con forma de directorios.
- Transfiere XML, JavaScript Object Notation (JSON), o ambos.

Arquitectura de Aplicaciones Web – 1*C 2018

Nos quedamos con REST...

- Otra vez a comparar y elegir frameworks?



Arquitectura de Aplicaciones Web – 1*C 2018

Notas de Color



- Las nuevas versiones de herramientas de portal han incorporado recursos relacionados con SPA.
- Las tecnologías de desarrollo mobile conocidas como híbridas toman este concepto como base.
- Los frameworks SPA poseen una comunidad activa y en constante crecimiento.

Arquitectura de Aplicaciones Web – 1ºC 2018



¿Preguntas?



Arquitectura de Aplicaciones Web – 1ºC 2018

