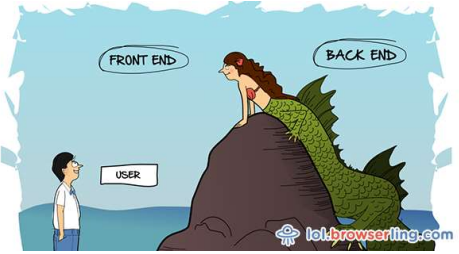


Temas de Hoy




USER

FRONT END


BACK END

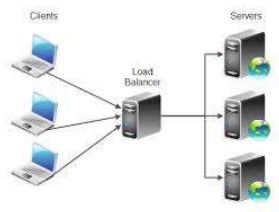
telbrowserling.com



Frontend / Backend

Infraestructura IT





Clients

Load Balancer

Servers

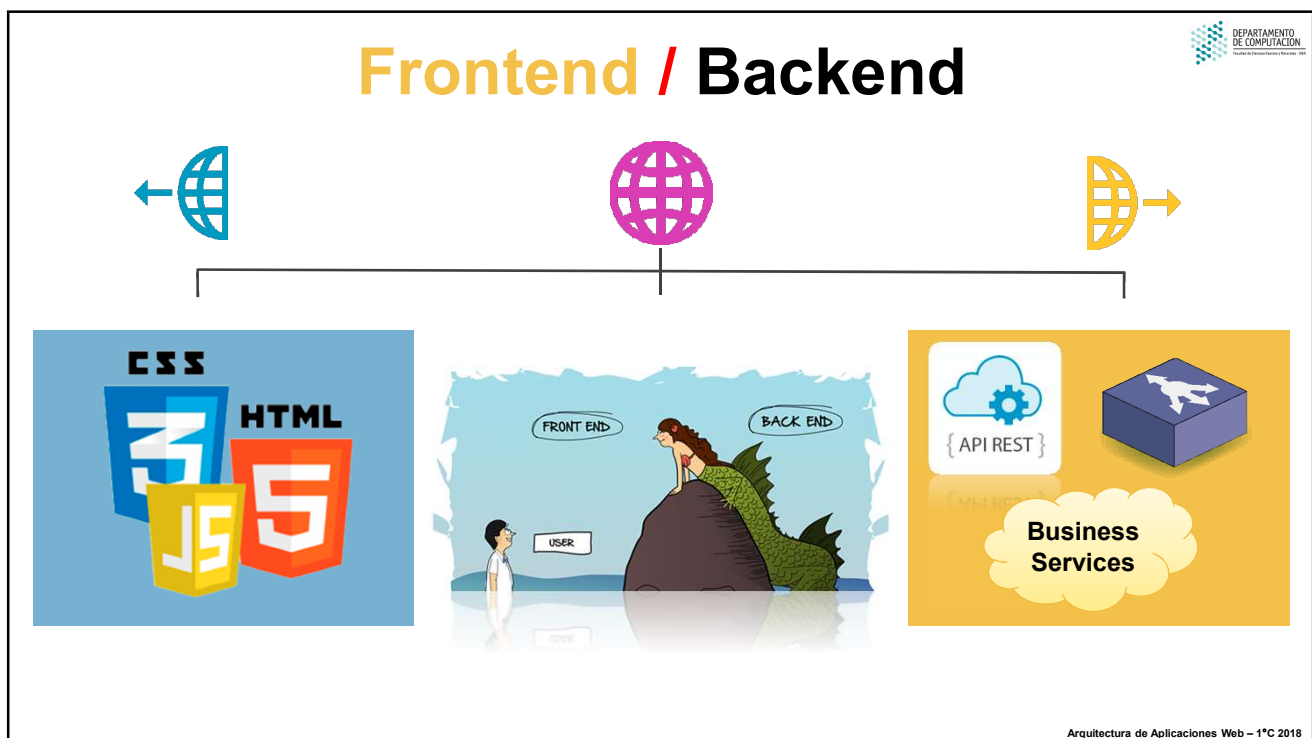
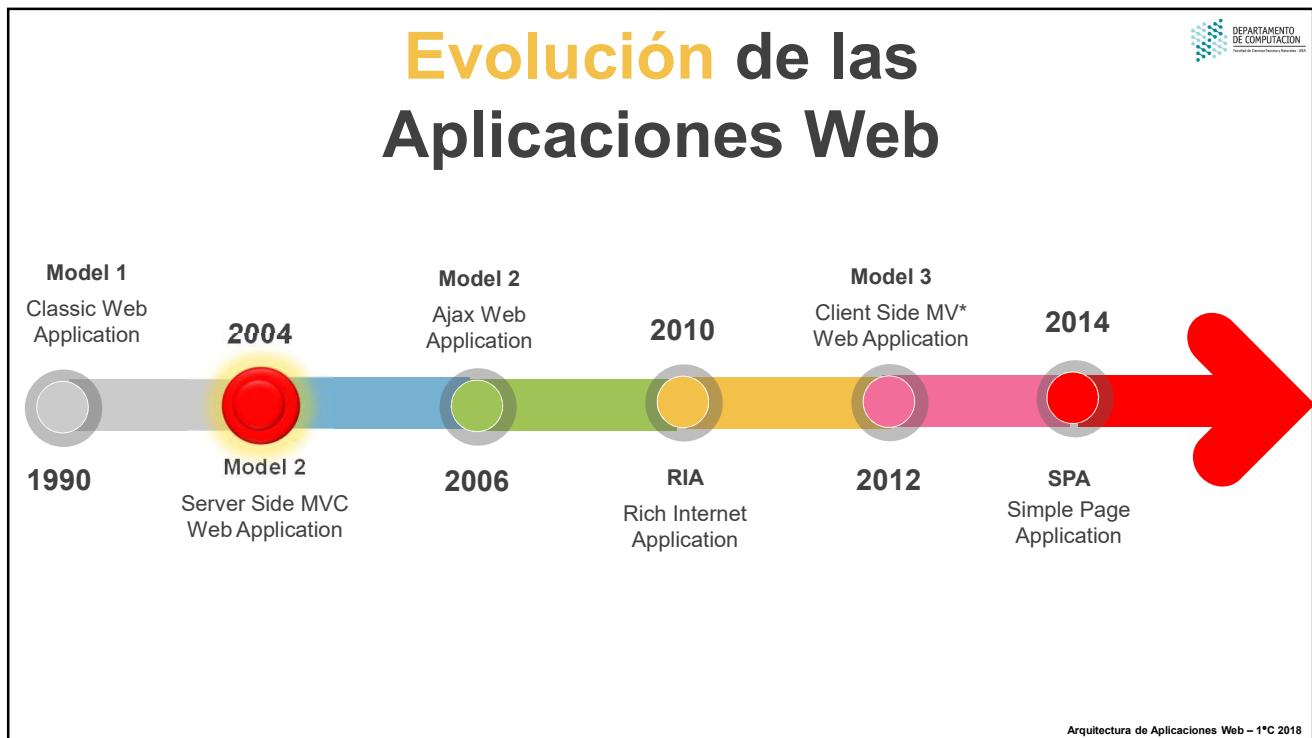
Arquitectura de Aplicaciones Web – 1°C 2018



PREVIOUSLY...

Retomando...

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA



Frontend / Backend

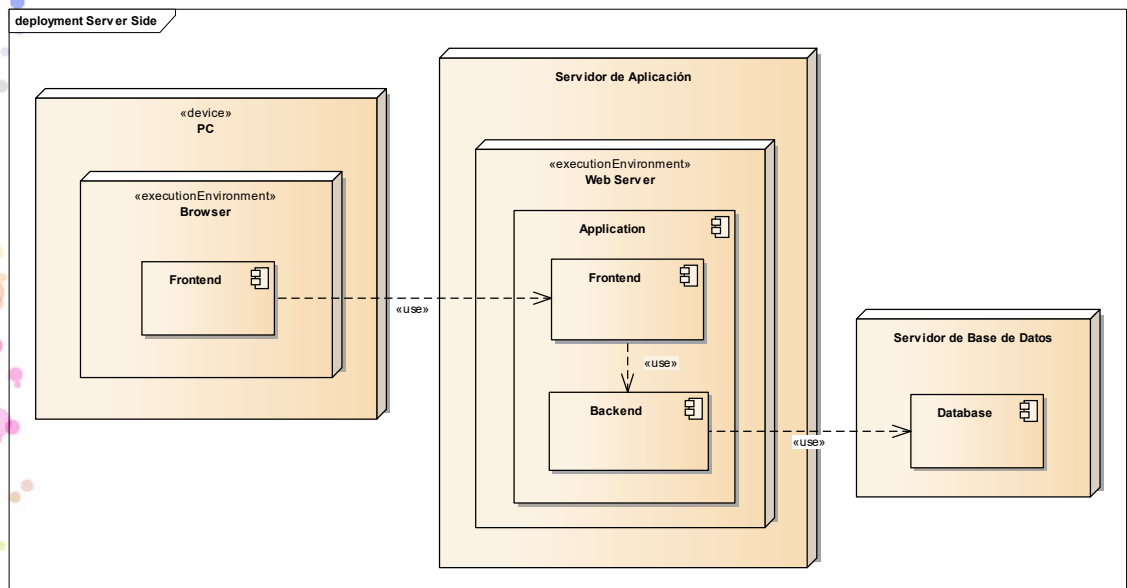
Frontend \leftrightarrow interacción con el usuario

Backend \leftrightarrow lógica de negocio consumida por el frontend

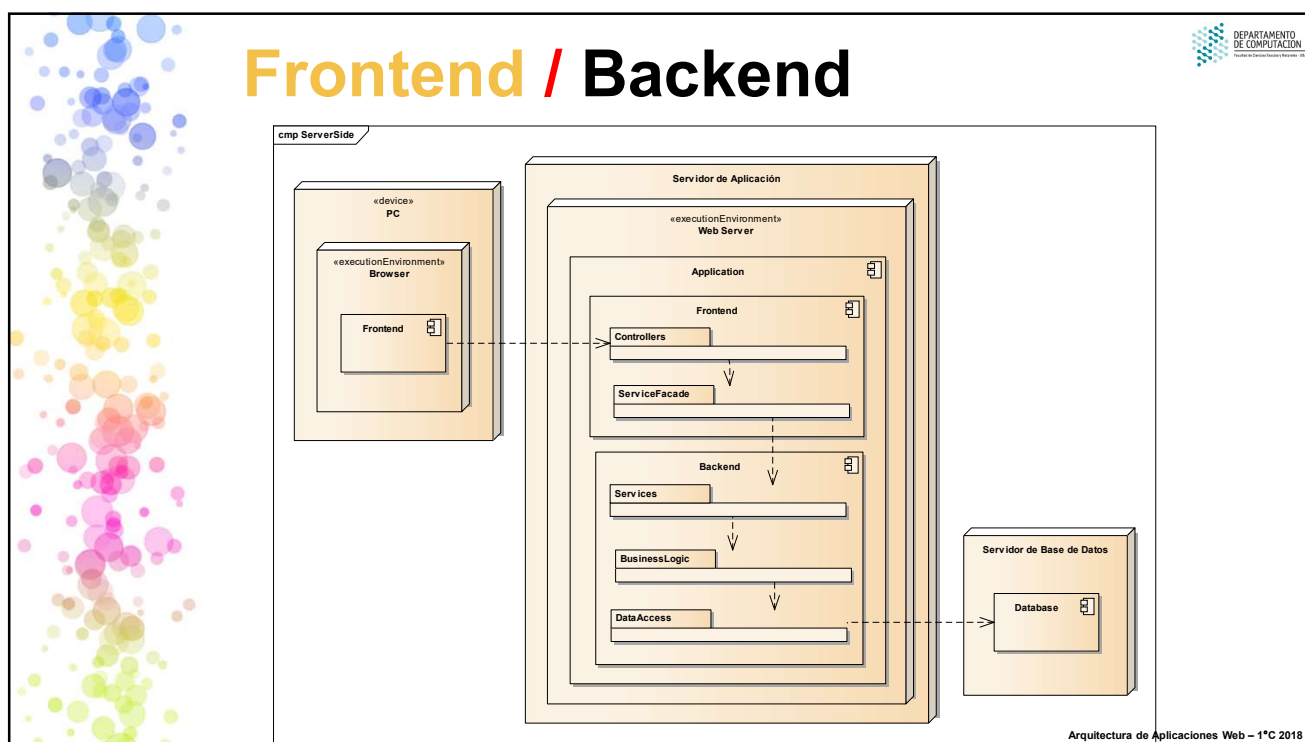
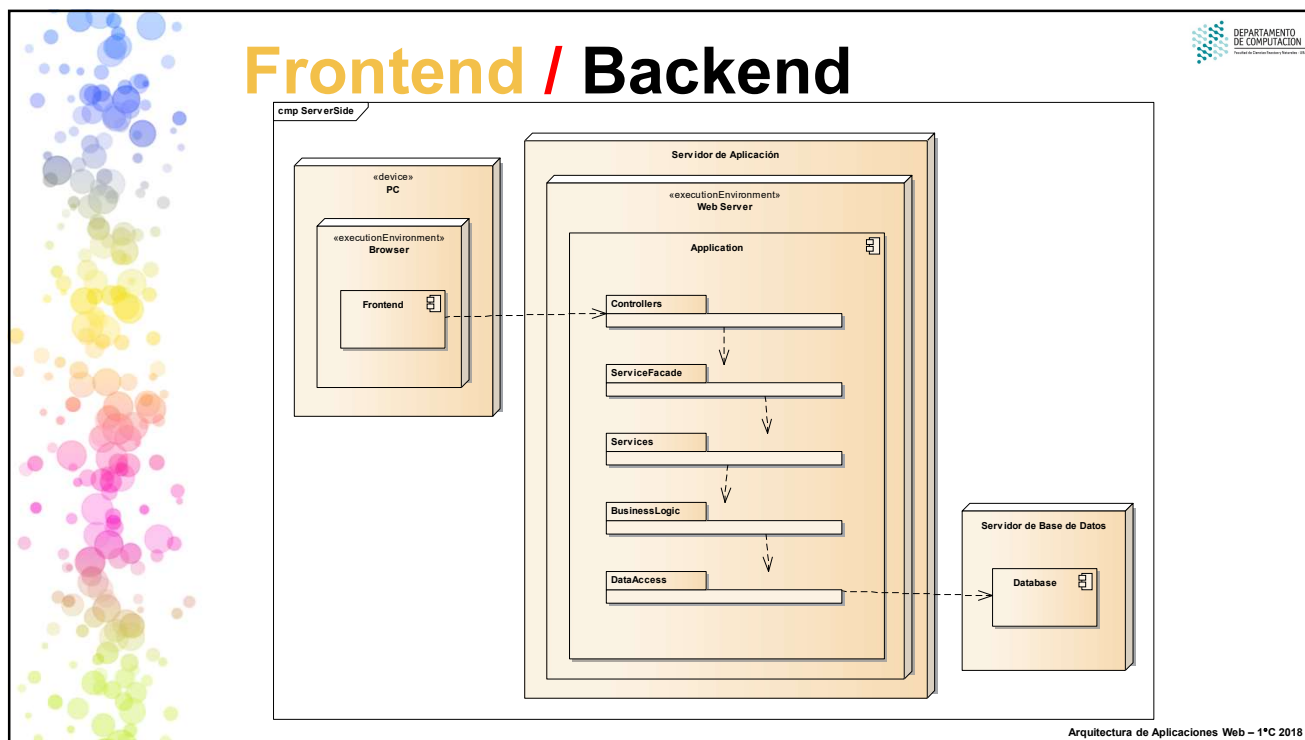


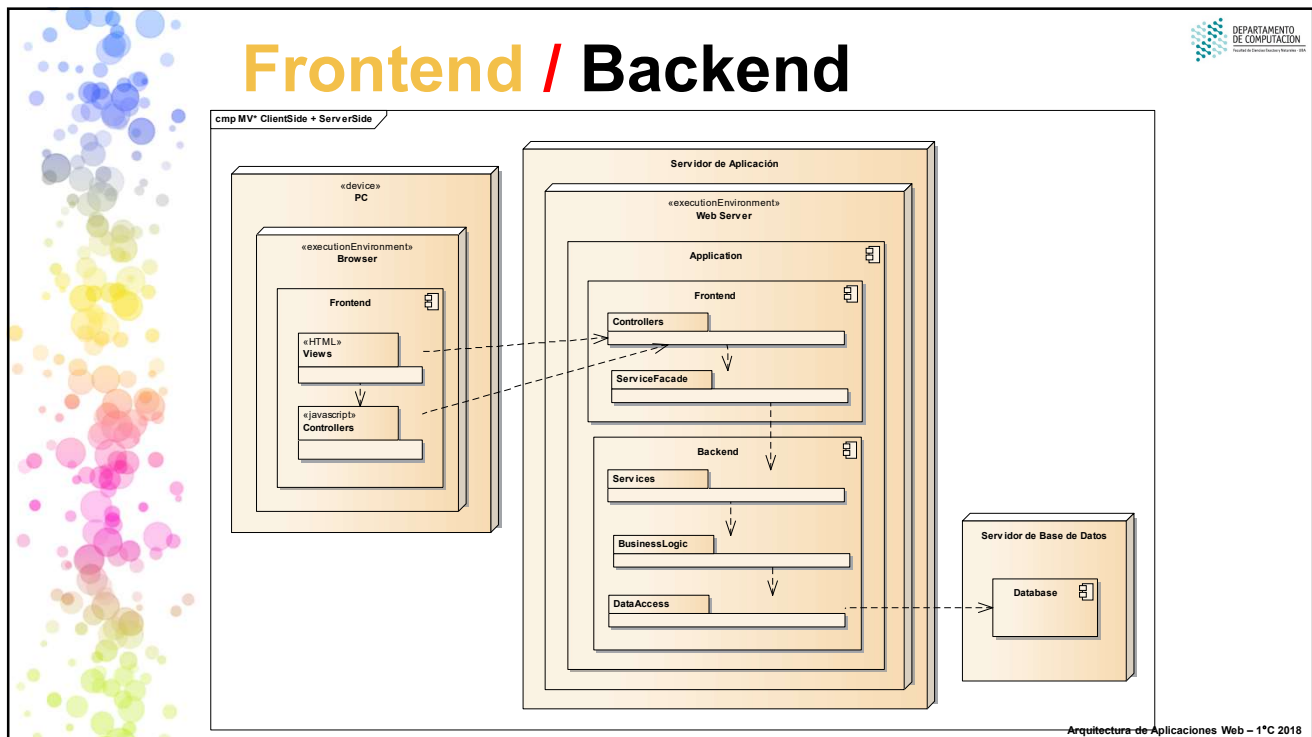
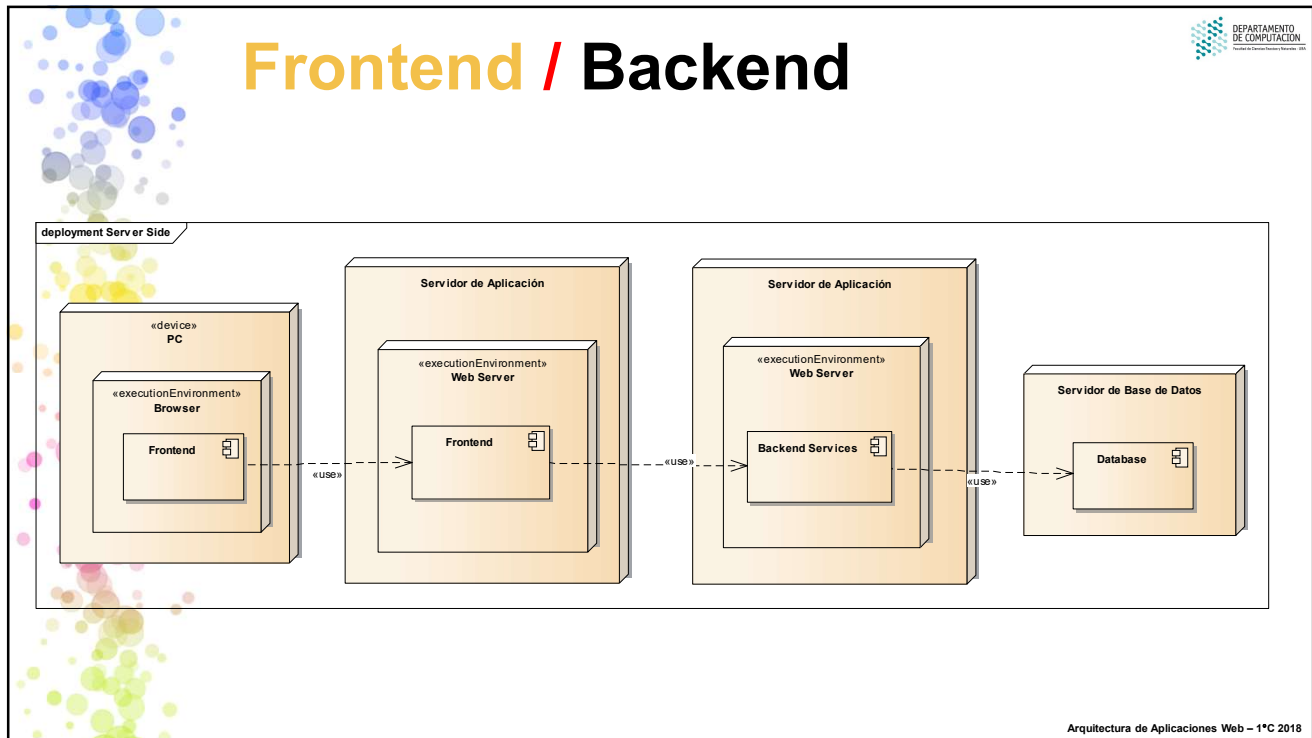
Arquitectura de Aplicaciones Web – 1°C 2018

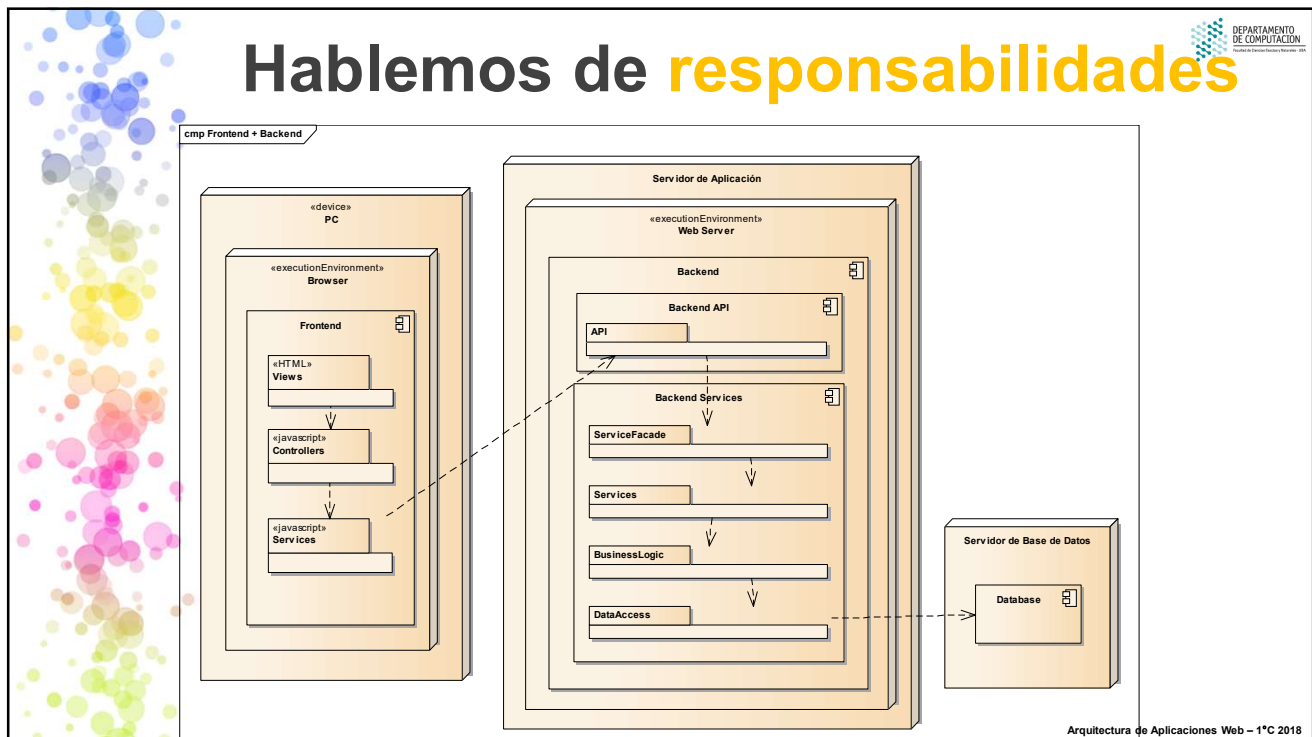
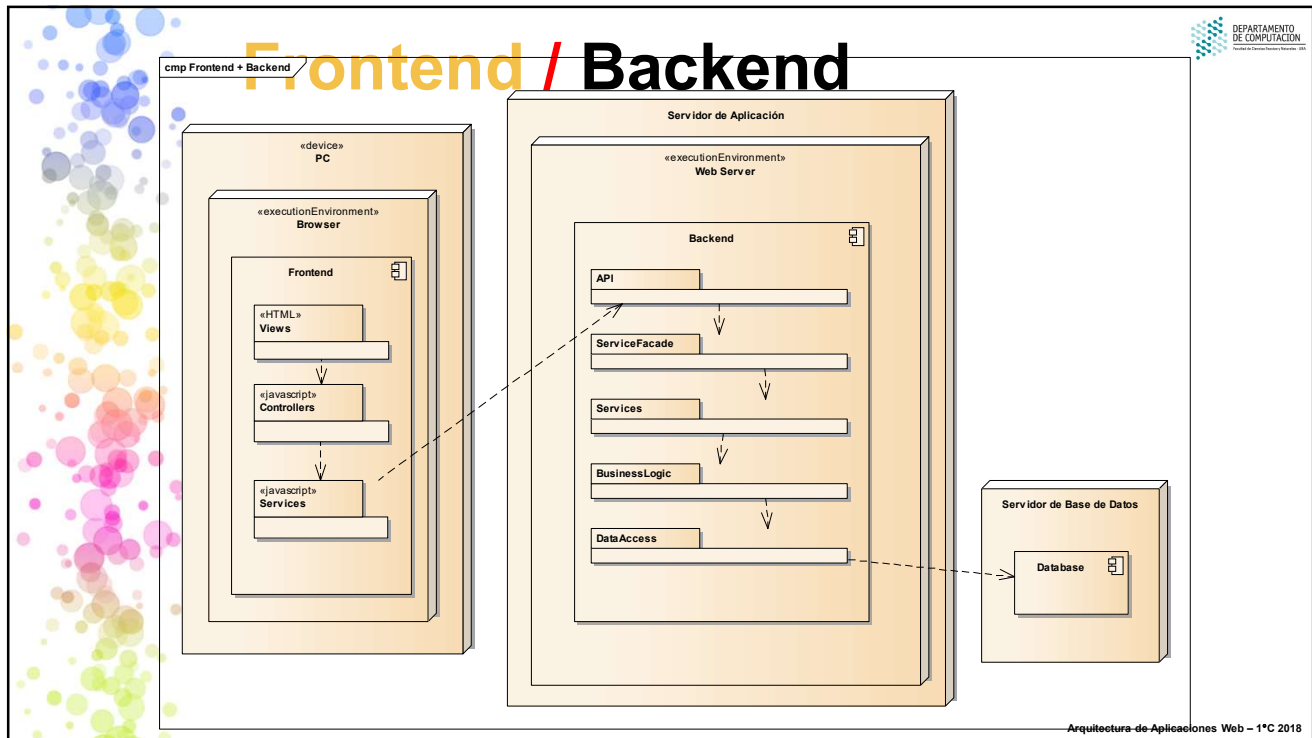
Frontend / Backend

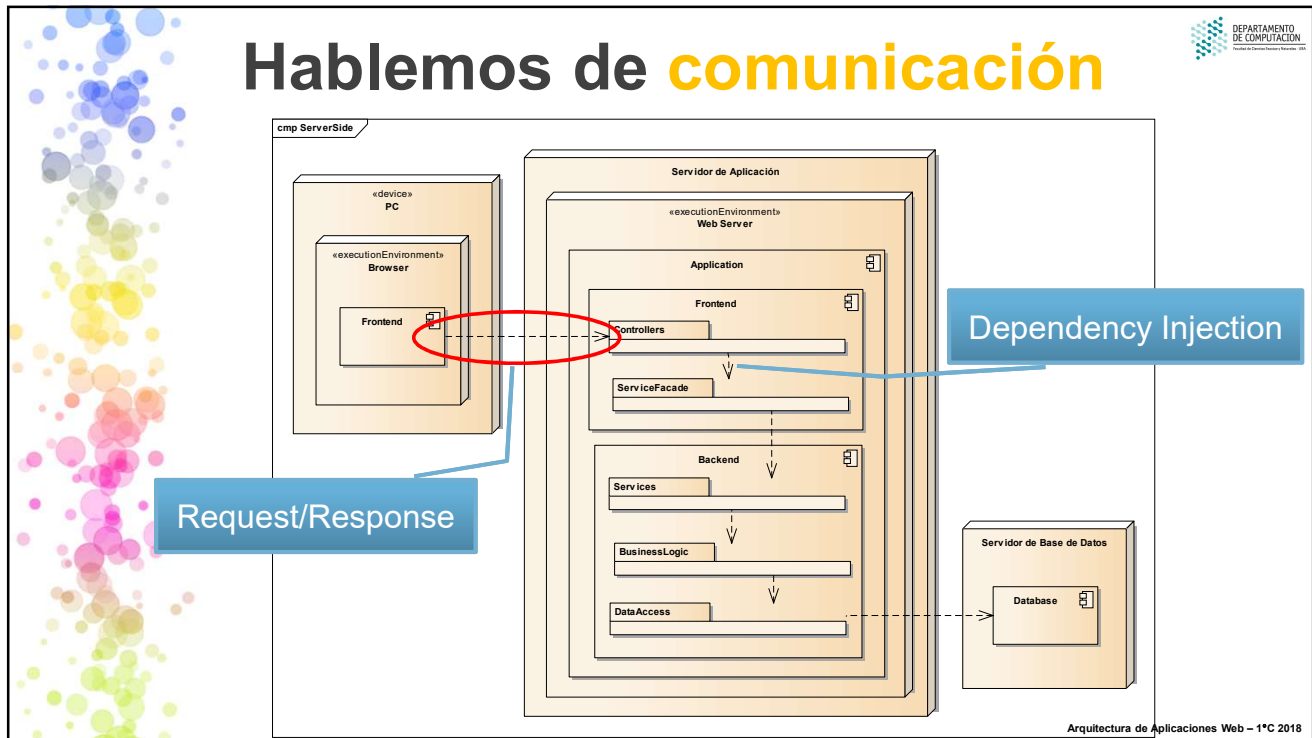


Arquitectura de Aplicaciones Web – 1°C 2018











Servicios Web

Arquitectura de Aplicaciones Web – 1°C 2018

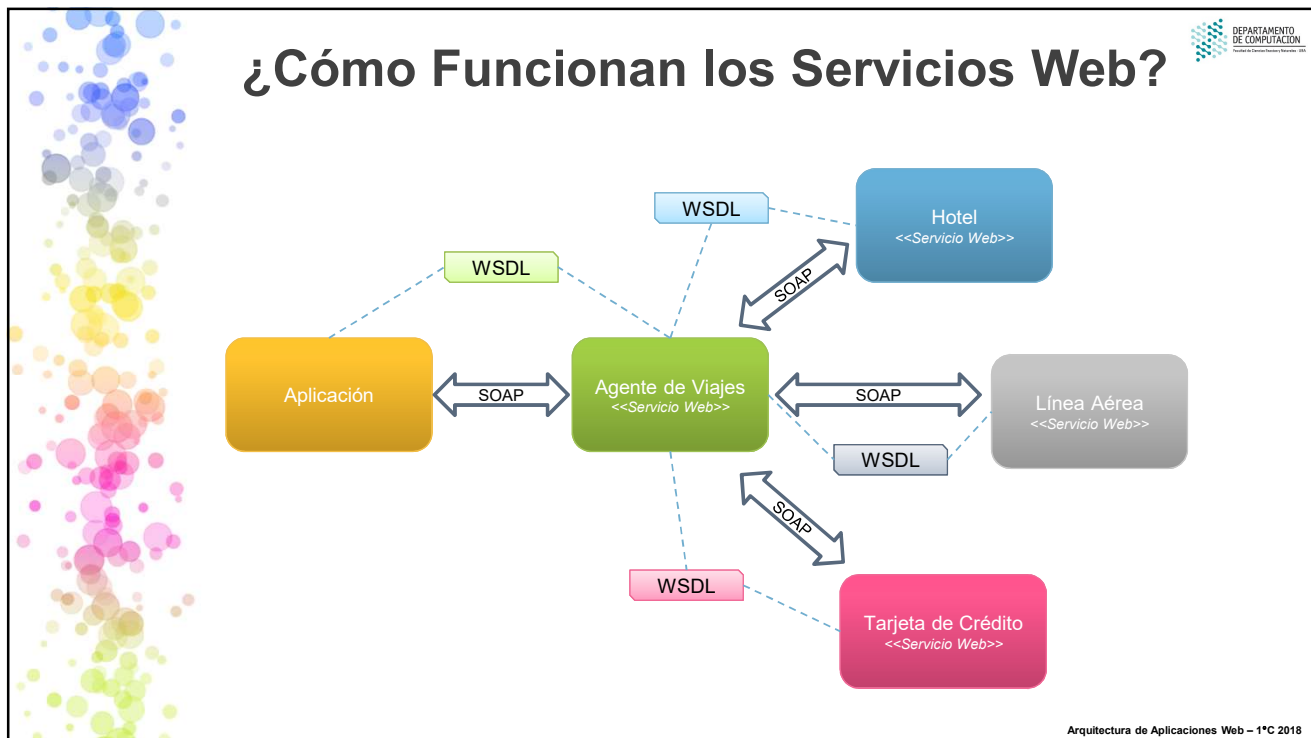


Web Services


Un Web Service es un componente de software que se comunica con otras aplicaciones codificando los mensaje en XML y enviando estos mensaje a través de protocolos estándares de Internet tales como el Hypertext Transfer Protocol (HTTP).

Intuitivamente es similar a un sitio web, pero no existe interacción con una persona a través de un web browser, sino que la interacción es entre aplicaciones

Arquitectura de Aplicaciones Web – 1°C 2018








SOAP


- ❑ Primer aparición: 1998
- ❑ Está basado en XML
- ❑ Es plataforma independiente
- ❑ Provee una forma estándar de estructurar mensajes XML

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Arquitectura de Aplicaciones Web – 1°C 2018



SOAP



Aplicación1 Mensajes XML Aplicación2

Es un protocolo estándar de mensajería entre procesos.
Un framework de mensajería basado en XML, que ofrece las siguientes características:

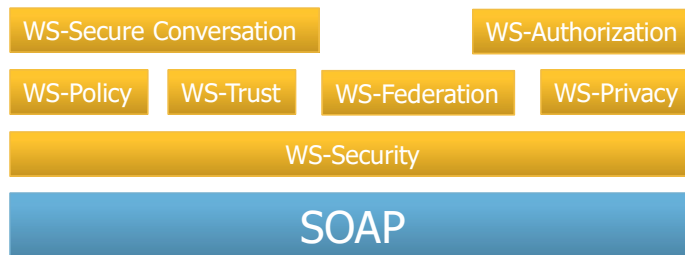
- Extensible
- Interoperable
- Independiente

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Arquitectura de Aplicaciones Web – 1°C 2018

Extensible

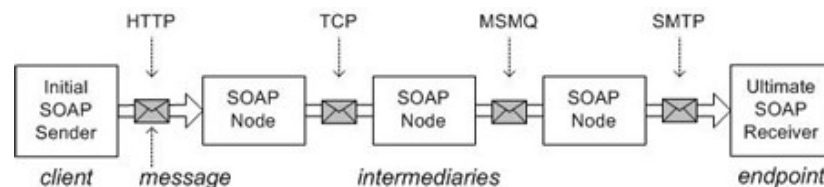
- ❑ Simplicidad es uno de sus objetivos de diseño.
- ❑ El framework de comunicación definido, permite agregar características como seguridad, ruteo y confiabilidad como extensiones (en forma de capas)



Arquitectura de Aplicaciones Web – 1°C 2018

Interoperable

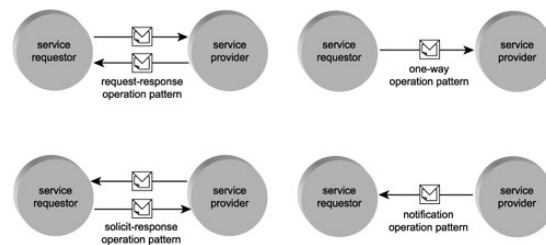
- ❑ SOAP puede ser usado sobre cualquier protocolo de transporte: TCP, HTTP, SMTP
- ❑ SOAP provee un *binding* explícito para HTTP



Arquitectura de Aplicaciones Web – 1°C 2018

Independiente

- ❑ SOAP no depende de un lenguaje de programación.
- ❑ SOAP define un modelos de procesamiento individual de one-way messages (se envía el mensaje en una dirección y no se espera respuesta)
- ❑ SOAP permite implementar distintos Message Exchange Patterns (MEPs)




Arquitectura de Aplicaciones Web – 1°C 2018

SOAP Message Format

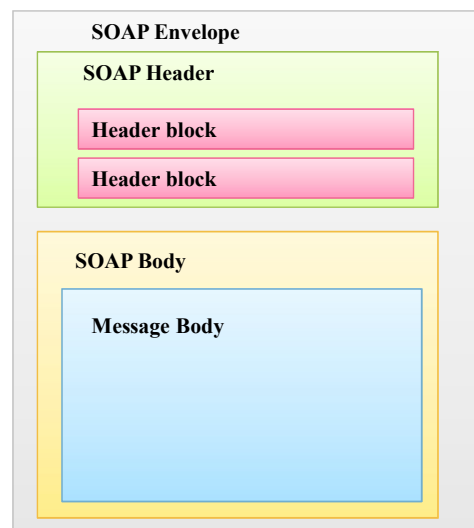
- SOAP message consists of three parts:
 - SOAP Envelope
 - SOAP Header (opcional)
 - SOAP Body

From the <http://schemas.xmlsoap.org/soap/envelope/> namespace

Arquitectura de Aplicaciones Web – 1°C 2018



SOAP Messages



Header contiene bloques de información relacionada a cómo se debe procesar el mensaje:

- Routing y Delivery
- Authentication/authorization
- Transaction contexts

Body contiene el mensaje en sí que debe ser entregado y procesado.

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Arquitectura de Aplicaciones Web – 1°C 2018



Ejemplo SOAP: Orden de Compra

```
<s:Envelope xmlns:s=http://www.w3.org/2001/06/soap-envelope>
  <s:Header>
    <m:transaction xmlns:m="soap-transaction">
      <transactionID>1234</transactionID>
    </m:transaction>
  </s:Header>
  <s:Body>
    <n:purchaseOrder xmlns:n="urn:OrderService">
      <from><person>Christopher Robin</person>
        <dept>Accounting</dept></from>
      <to><person>Pooh Bear</person>
        <dept>Honey</dept></to>
      <order><quantity>1</quantity>
        <item>Pooh Stick</item></order>
    </n:purchaseOrder>
  </s:Body>
</s:Envelope>
```

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Arquitectura de Aplicaciones Web – 1°C 2018

WSDL

- **Web Service Description Language.**
- Lenguaje basado en XML usado para **describir y localizar** Servicios Web.
 - Escritos XML.
 - Describe la funcionalidad de un Servicio Web.
 - Especifica cómo se accede al servicio: protocolo, mensajería, tipos de datos, etc.)



Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL

```
<definition namespace = "http/... ">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>
```



Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<definition namespace = "http/... ">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>
```

- <types> define los tipos utilizados en los mensajes
- XML Schema, DTD, and etc.

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<types>
  <schema targetNamespace="http://example.com/stockquote.xsd" xmlns="http://www.w3.org/2000/10/XMLSchema" >
    <element name="TradePriceRequest">
      <complexType>
        <all>
          <element name="tickerSymbol" type="string"
            minOccurs = "1" maxOccurs="10"/>
          <element name = "payment">
            <complexType> <choice>
              <element name = "account" type="string">
              <element name = "creditcard" type="string">
            </choice> </complexType>
          </element>
        </all>
      </complexType>
    </element>
  </schema>
</types>
```

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<definition namespace = "http/...">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>
```

<message> define los elementos de las operaciones. Cada mensaje posee partes. Las partes podrían pensarse como los parametros de una invocación a una función.

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<message name="GetLastTradePriceInput">
  <part name="body" element="TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="TradePrice"/>
</message>
```

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<definition namespace = "http/... ">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>
```

<port> define las operaciones en sí.

```
<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>
```

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<definition namespace = "http/... ">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>
```

<binding> define cómo el mensaje será transmitido y la ubicación del servicio.

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
```

Arquitectura de Aplicaciones Web – 1°C 2018

Estructura de un WSDL



```
<definition namespace = "http/... ">
  <type> xschema types </type>
  <message> ... </message>
  <port> a set of operations </port>
  <binding> communication protocols </binding>
  <service> a list of binding and ports </service>
</definition>

<service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
    <soap:address location="http://example.com/stockquote"/>
  </port>
</service>
```

Arquitectura de Aplicaciones Web – 1°C 2018



REST

REpresentational State Transfer

Arquitectura de Aplicaciones Web – 1°C 2018



¿Qué es REST?

- Origen: Fielding, Roy T. “Architectural Styles and the Design of Network-based Software Architectures.” Tesis Doctoral, Universidad de California, 2000.
- Describe un **estilo de arquitectura** que utilizar como modelo en los servicios de computación Web.
- Estilo de arquitectura: Conjunto coordinado de **restricciones** que controlan el funcionamiento y características de los elementos de la arquitectura y permite las relaciones de unos con otros.
- Describe **cómo debería comportarse la Web**
- **NO** es un estándar

Arquitectura de Aplicaciones Web – 1°C 2018

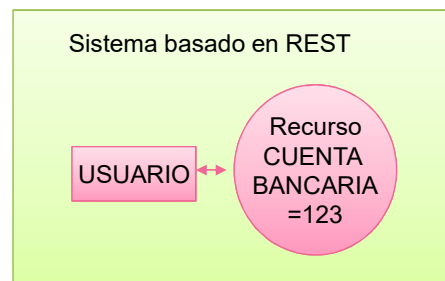
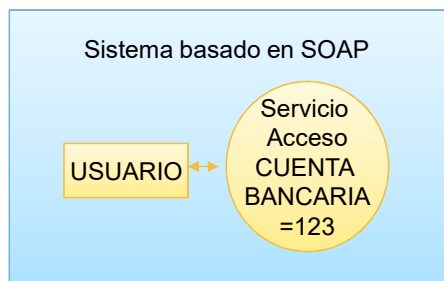
¿Por qué ha triunfado la Web?

- Escalabilidad en interacciones entre componentes
- Generalidad en las interfaces
- Desarrollo independiente de componentes
- Existencia de componentes intermediarios (proxys)

Arquitectura de Aplicaciones Web – 1°C 2018

Principios de REST

- El estado y la funcionalidad de las aplicaciones se divide en **recursos**
 - REST es orientado a recursos y no a métodos
 - No se accede directamente a los recursos, sino a representaciones de los mismos



Arquitectura de Aplicaciones Web – 1°C 2018



Principios de REST

- Todo recurso es identificado de **forma única global** mediante una sintaxis universal. Como en HTTP los recursos se identifican mediante URIs (*Uniform Resource Identifier*).
 - ❑ **Conjunto potencialmente infinito de recursos.**
- Todos los recursos comparten un **interfaz uniforme** formado por:
 - **Conjunto de operaciones** limitado para transferencia de estado
 - En HTTP GET, PUT, POST, DELETE
 - **Conjunto limitado de tipos de contenidos**
 - En HTTP se identifican mediante tipos MIME: XML , HTML...

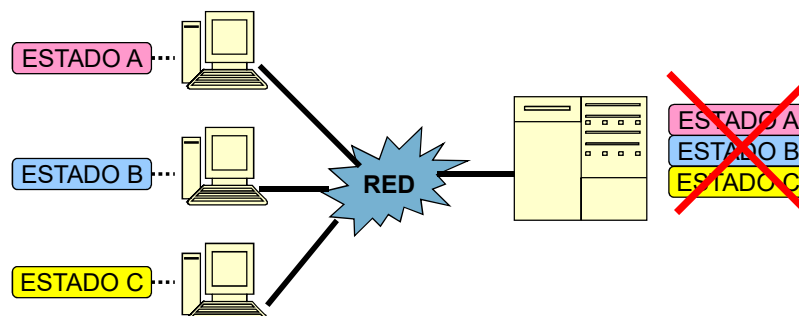
MÉTODO	FUNCIÓN
GET	Solicitar recurso
POST	Crear recurso nuevo
PUT	Actualizar o modificar recurso
DELETE	Borrar recurso
DELETE	Borrar recurso

Arquitectura de Aplicaciones Web – 1°C 2018



Principios de REST

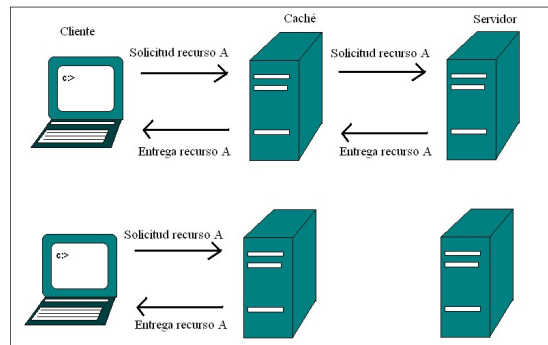
- **Un protocolo cliente/servidor, sin estado y basado en capas**
- **Cada mensaje contiene la información necesaria para comprender la petición (mensajes autocontenidos, como HTTP)**



Arquitectura de Aplicaciones Web – 1°C 2018

Principios de REST

- Promueve mecanismos caché y sistemas intermedios



Arquitectura de Aplicaciones Web – 1°C 2018

Ventajas de REST

- Mejora el tiempo de respuesta gracias al mecanismo Caché y los mensajes auto-descriptivos.
- Disminución de carga en servidor
- Mayor escalabilidad al no requerir mantenimiento de estado en el servidor
- Facilita desarrollo de clientes (menor dependencia del servidor).
- Mayor estabilidad frente a futuros cambios
 - Permite evolución independiente de los tipos de documentos al procesar éstos en el cliente.

Arquitectura de Aplicaciones Web – 1°C 2018

Diferencias entre **REST** y SOAP

SOAP	REST
Orientado a RPC	Orientado a recursos
Servidor almacena parte del estado	El estado se mantiene sólo en el cliente, y no se permiten las sesiones
Usa HTTP como túnel para el paso de mensajes	Propone HTTP como nivel de aplicación

Arquitectura de Aplicaciones Web – 1°C 2018

Ejemplo

- Sistema basado en SOAP
 - Énfasis en diversidad de operaciones (verbos)

```

getUser()
addUser()
removeUser()
updateUser()
getLocation()
addLocation()
removeLocation()
updateLocation()
listUsers()

```

- Sistema REST
 - Énfasis en diversidad de recursos (nombres)

User {} Location{}

- Registro del recurso User (accesible con HTTP GET):

```

<usuario>
<nombre>Benito Pérez</nombre>
<genero>masculino</genero>
<localizacion
href="http://www.example.org/locations/spain/oviedo">
    Oviedo, Spain
</localizacion>
</usuario>

```

Arquitectura de Aplicaciones Web – 1°C 2018

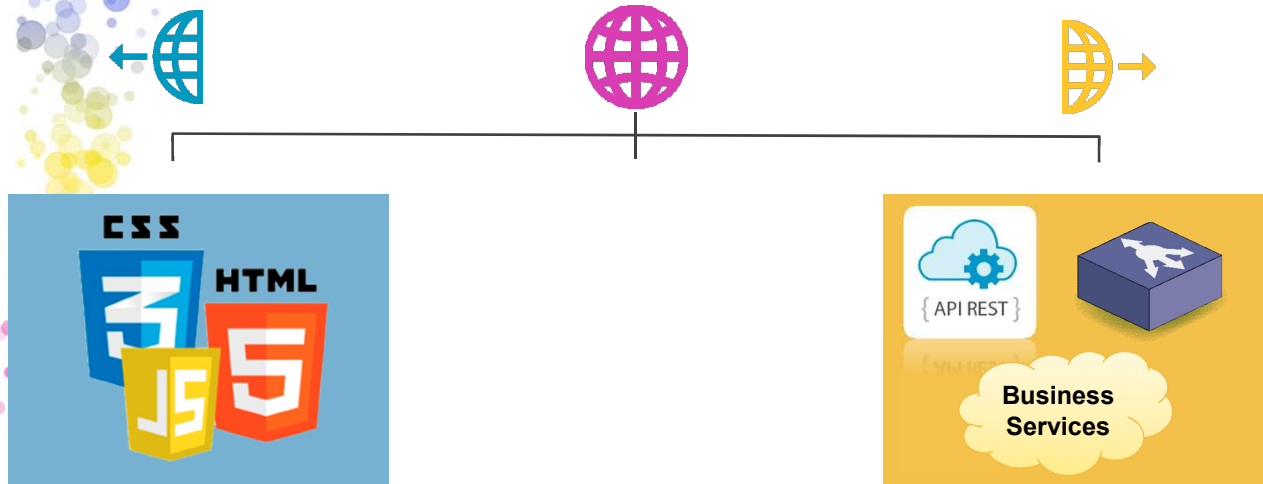
SOAP vs REST: Críticas



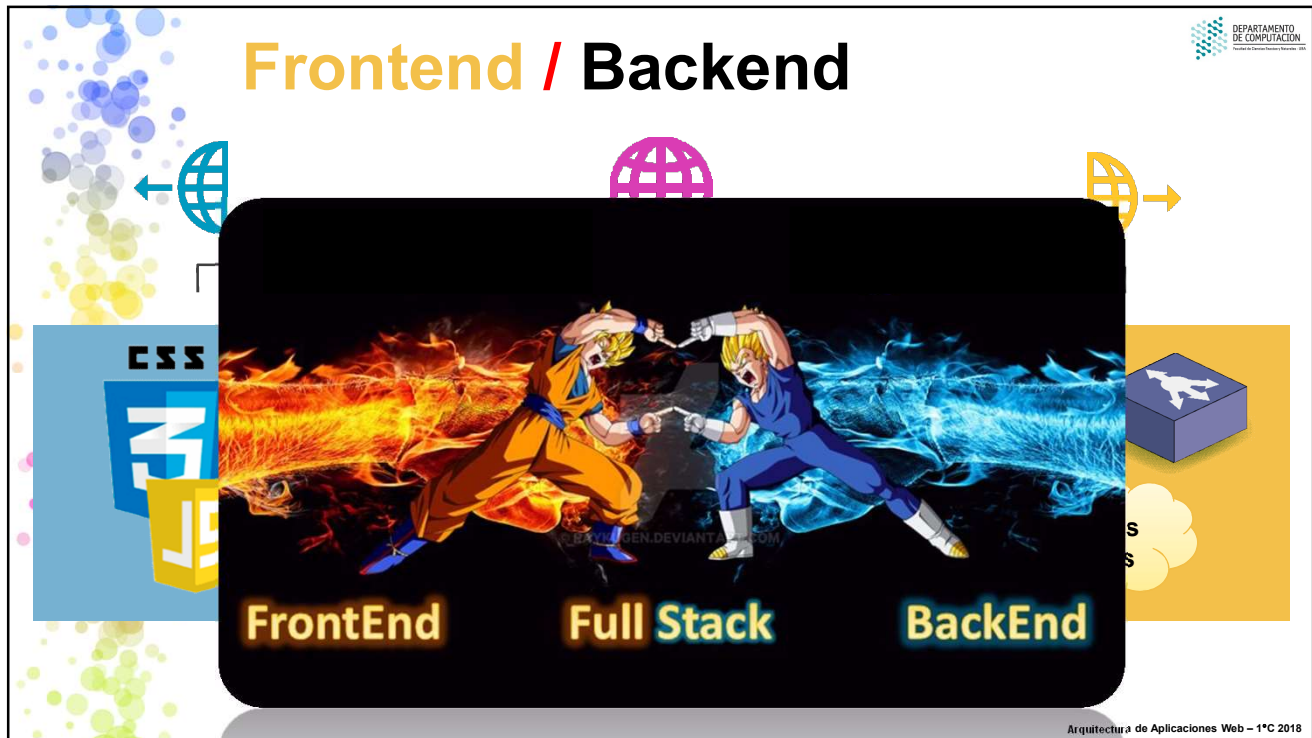
- SOAP no es transparente, apuesta por el encapsulamiento
- SOAP no dispone de un sistema de direccionamiento global
- SOAP puede derivar en agujeros de seguridad
- SOAP no aprovecha muchas de las ventajas de HTTP al usarlo solamente como túnel.
- SOAP no puede hacer uso de los mecanismos Caché.
- REST es poco flexible
- REST no está preparado para albergar Servicios Web de gran complejidad como las aplicaciones B2B
- REST tiene grandes problemas de seguridad al no soportar el concepto de sesión

Arquitectura de Aplicaciones Web – 1°C 2018

Frontend / Backend




Arquitectura de Aplicaciones Web – 1°C 2018



Web Developers

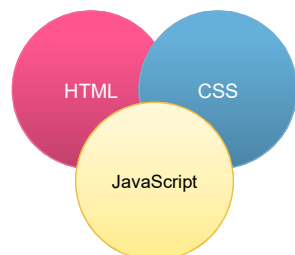
- Front-End Developer
- Web Designer
- Back-End Developer
- Web Programmer
- Full Stack Developer

A cartoon illustration of a superhero's chest, with a blue shirt that says "WEB DEVELOPER" in white capital letters. The superhero is tearing open a white jacket to reveal the shirt. The bottom right corner contains the text "Arquitectura de Aplicaciones Web - 1°C 2018".




Front-End Developer

- Su foco está en el look&feel de la aplicación.
- Herramientas: HTML, CSS, and JavaScript
- Esquema de colores, diseño gráfico, flujo de la información
- User Experience

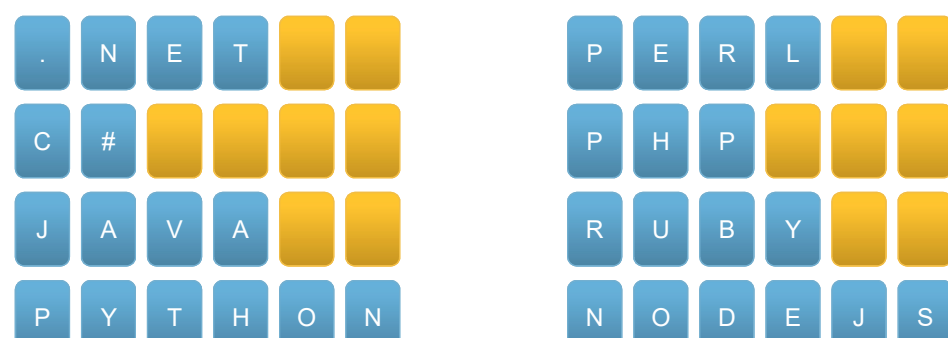


Arquitectura de Aplicaciones Web – 1°C 2018



Back-End Developer

- Implementa el comportamiento de la aplicación.
- Conocimientos de lenguajes de programación, base de datos, redes, sistemas operativos, etc...



Arquitectura de Aplicaciones Web – 1°C 2018





Full Stack Developer

- Maneja ambas tecnologías:


Frontend y Backend

Arquitectura de Aplicaciones Web – 1°C 2018



¿Qué es un Full Stack?


- **System Administration**
 - Shell Scripting (Linux)
 - Cloud Computing (Amazon, Rackspace)
 - Search Engine Integration (ElasticSearch, Sphinx)
 - Caching
 - Monitoring



Arquitectura de Aplicaciones Web – 1°C 2018

What is a Full Stack?

- **Web Development Tools**
 - Version Control (Git, Mercurial)
 - Virtualization (VirtualBox, Vagrant, Docker)




DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

1°C 2018

What is a Full Stack?

- **Back-end Technologies**
 - Web Servers (Apache, Nginx)
 - Programming Languages (PHP, NodeJS, Ruby)
 - Databases (MySQL, PostgreSQL, MongoDB, Redis) – General (SQL, JSON, XML)



DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

1°C 2018

What is a Full Stack?

- **Front-end Technologies**

- HTML / HTML5 Semantic Web
- CSS / CSS3: LESS, SASS, Media Queries
- JavaScript: JQuery, AngularJS, ...
- Browser Compatibility
- Responsive Design
- AJAX, JSON, XML, WebSocket



What is a Full Stack?

- **Design**

- Convert design to front-end code (PHP-HTML/CSS)
- User Interface (UI)
- User Experience (UX)





Algunos temas de IT

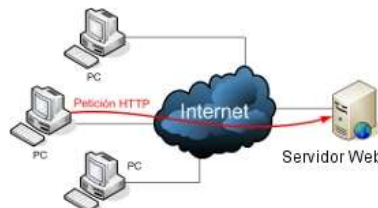
Infraestructura

Arquitectura de Aplicaciones Web – 1°C 2018



Web Server

Servidores Web (Web Servers): Básicamente, un servidor web sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP.



Arquitectura de Aplicaciones Web – 1°C 2018

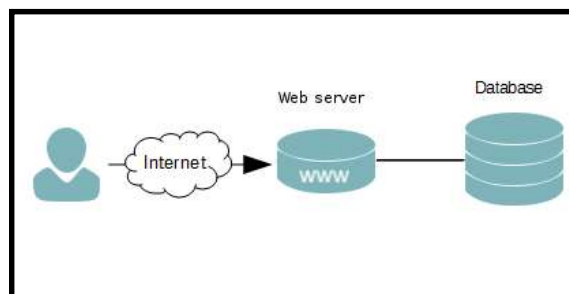
¿Qué le pedimos IT?

- ❑ Las aplicaciones web tienen que atender a todos los usuarios que la estén usando (aunque sean muchos): **escalabilidad**
- ❑ Hay veces que hay muchos usuarios y otras veces que hay pocos usuarios: **elasticidad**
- ❑ El hardware falla, pero la aplicación web tiene que seguir prestando servicio a los usuarios: **tolerancia a fallos**

Arquitectura de Aplicaciones Web – 1°C 2018

¿Cómo escalamos?

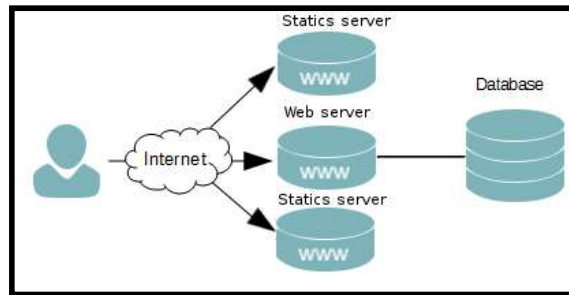
Separar el servidor de base de datos.



Arquitectura de Aplicaciones Web – 1°C 2018

¿Cómo escalamos?

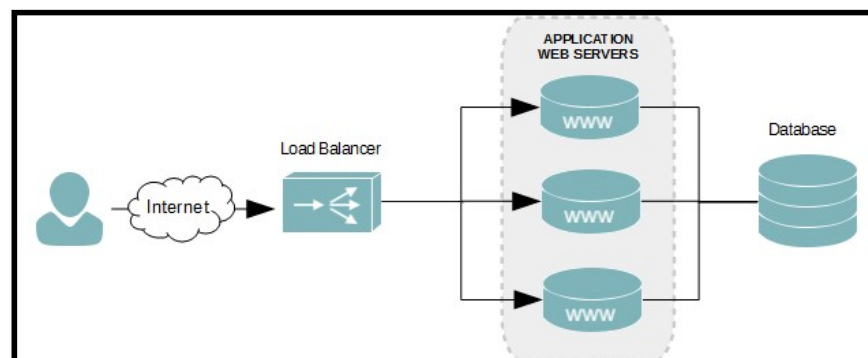
Separar el contenido estático y dinámico



Arquitectura de Aplicaciones Web – 1°C 2018

¿Cómo escalamos?

Balanceando la carga



Arquitectura de Aplicaciones Web – 1°C 2018




Web Server & Application Server... + servers

¿Es lo mismo pero de otro color?




Arquitectura de Aplicaciones Web – 1°C 2018



Plataforma JEE

- Es la plataforma para desarrollar aplicativos de alto porte. Apareció en los '90 motivado por:
 - Necesidad de migrar los EIS (Enterprise Information Systems) de arquitecturas en dos capas (cliente/servidor) a arquitecturas más flexibles en tres o más capas, permitiendo satisfacer los requerimientos de los clientes.
 - Evolución de los servicios de **middleware** OTM, MOM, ORBs
 - Aumento en el uso de internet e intranets para aplicaciones empresariales.
- Existía la necesidad de un estándar para el desarrollo de aplicaciones basadas en arquitecturas de componentes en varias capas. En ese momento Sun y sus partners industriales daban los primeros pasos en generar un estándar.


Arquitectura de Aplicaciones Web – 1°C 2018



Plataforma JEE

- La plataforma JEE define un mecanismo estándar basado en el concepto portabilidad de Java.
- Pretende facilitar y disminuir el costo de desarrollo de EIS de múltiples capas que permitan adaptarse a los requerimientos actuales.
- Fomenta la separación en capas y el desarrollo especializado de componentes.
- Disminuye el código necesario para desarrollar los sistemas a través de: servicios ya provistos y APIs que implementan las funcionalidades normalmente requeridas por los EIS.


Arquitectura de Aplicaciones Web – 1°C 2018




JEE – Es un software?

- JEE es una especificación, NO un producto
- La definición de un estándar permite
 - Que distintos proveedores implementen productos compatibles con el mismo, por lo que lo apoyarán y esto evitará estar atado a un proveedor dado.
 - Establece un modelo claro de construcción de aplicaciones, lo que hace que sea más fácil contar con desarrolladores capacitados

Arquitectura de Aplicaciones Web – 1°C 2018




¿Qué hay dentro de JEE?




- **Web**
 - **Servlets**: abstracción orientada objetos para la generación de páginas Web dinámicas.
 - **JSP**: generación dinámica de páginas Web a través de templates, minimizando la necesidad de programación
- **Lógica de Negocio**
 - **EJB**: modelo de componentes para representar la lógica de negocio y correrla dentro de un servidor de aplicaciones
- **General**
 - **RMI**: API para la invocación remota de métodos con soporte para protocolos IIOP y JRMP
 - **JMS**: API estándar para el acceso a MOM con 2 modelos de mensajería Point-to-Point (Queue) y publish-subscriber (Topics)
 - **JavaMail**: API estándar independiente del protocolo que permite el envío y la recepción de mails

Arquitectura de Aplicaciones Web – 1°C 2018



¿Qué hay dentro de JEE?



- **General**
 - **RMI**: API para la invocación remota de métodos con soporte para protocolos IIOP y JRMP
 - **JMS**: API estándar para el acceso a MOM con 2 modelos de mensajería Point-to-Point (Queue) y publish-subscriber (Topics)
 - **JavaMail**: API estándar independiente del protocolo que permite el envío y la recepción de mails

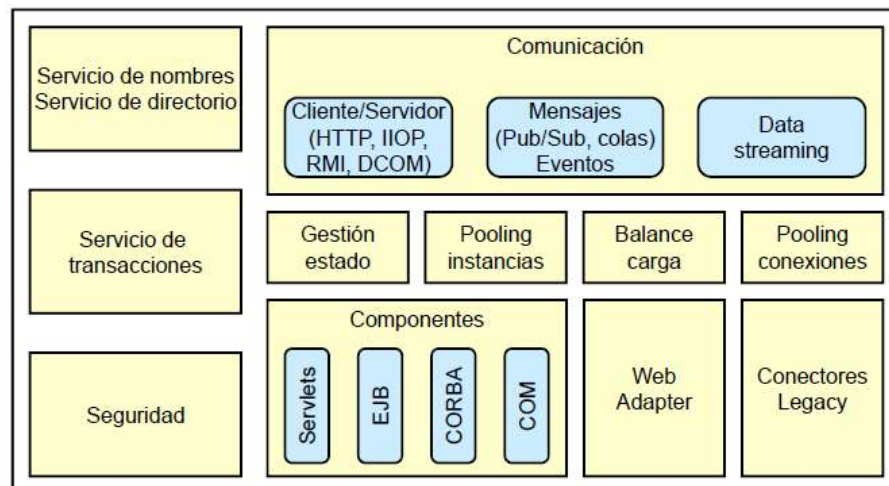
Arquitectura de Aplicaciones Web – 1°C 2018

¿Qué hay dentro de JEE?

- General
 - **JTA**: API de alto nivel que permite controlar Transacciones Distribuidas y comunicarse con un Transaction Manager.
 - **JCA**: estándar para la integración de servidores de aplicaciones J2EE con Enterprise Information Systems (Mainframes, Sistemas Legados, Enterprise Resource Planning o Bases de datos).
 - **JDBC**: API para el acceso a base de datos relacionales con soporte de pool de conexiones y transacciones distribuidas.
 - **JNDI**: Permite el acceso a servicios de Nombres o Servicios de Directorios con soporte de LDAP, CORBA naming service, RMI.
 - **JAXP**: Provee acceso a APIs estándares para SAX y DOM para parsing de documentos XML, y también provee soporte para XSLT.
 - **Java IDL**: permite la invocación de métodos CORBA por parte de clientes Java.

Arquitectura de Aplicaciones Web – 1°C 2018

JEE App Server




Arquitectura de Aplicaciones Web – 1°C 2018



Atributos de Calidad



Arquitectura de Aplicaciones Web – 1°C 2018




Atributos de Calidad


- La funcionalidad «de negocio» es sólo una parte de lo que un sistema debe hacer.
- Además, están los atributos de calidad (“ilities”), que hablan de características específicas que debe tener el sistema (anteriormente llamados “requerimientos no funcionales”).
 - Ejemplo: portabilidad, flexibilidad, usabilidad.}
- Necesitamos conocerlos para definir una arquitectura.
- En muchos casos, se afectan entre si. Por ejemplo:
Portabilidad vs. Performance o Flexibilidad vs. Performance

“Software quality is the degree to which software possesses a desired combination of attributes.”

[IEEE Std. 1061]




Arquitectura de Aplicaciones Web – 1°C 2018



Atributos de Calidad

- Suelen estar pobremente especificados, o directamente no especificados (“un requerimiento que no es testeable, no es implementable”).
- En general no se analizan sus dependencias.
- La importancia de estos atributos varía con el dominio para el cual se construye el software.
- Además de requerimientos funcionales y atributos de calidad, el ingeniero de software debe identificar correctamente restricciones.
- Las “tácticas” de arquitectura no son fines en si mismas, son formas de alcanzar atributos de calidad deseados.
- El atributos de calidad que suele ser más importante: la flexibilidad (“facilidad de cambios”).

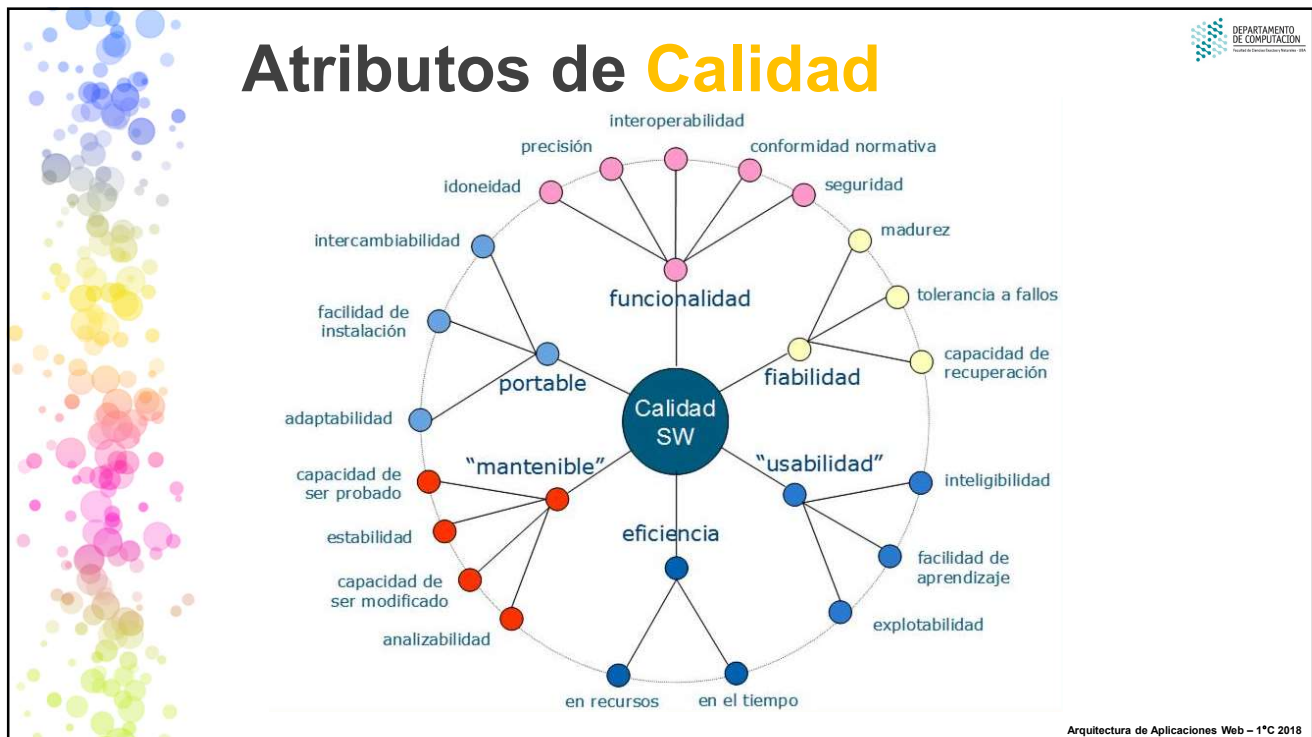
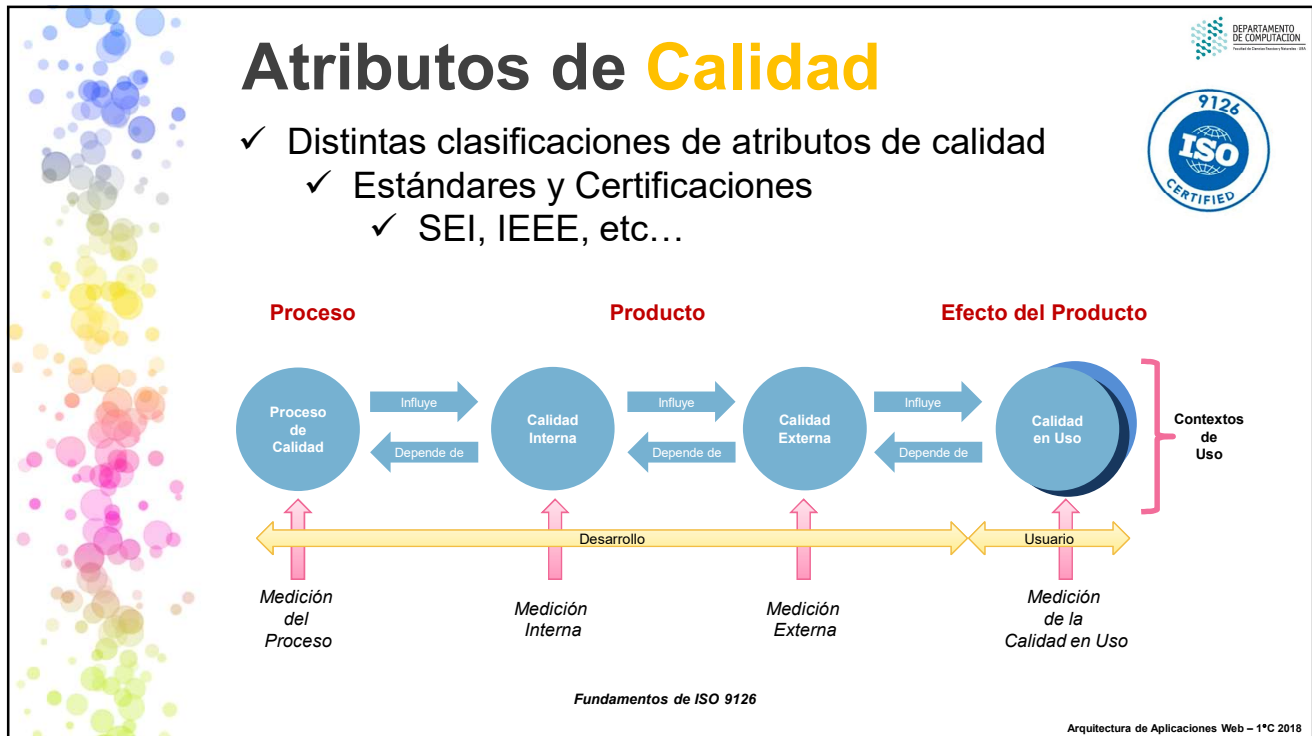
Arquitectura de Aplicaciones Web – 1°C 2018

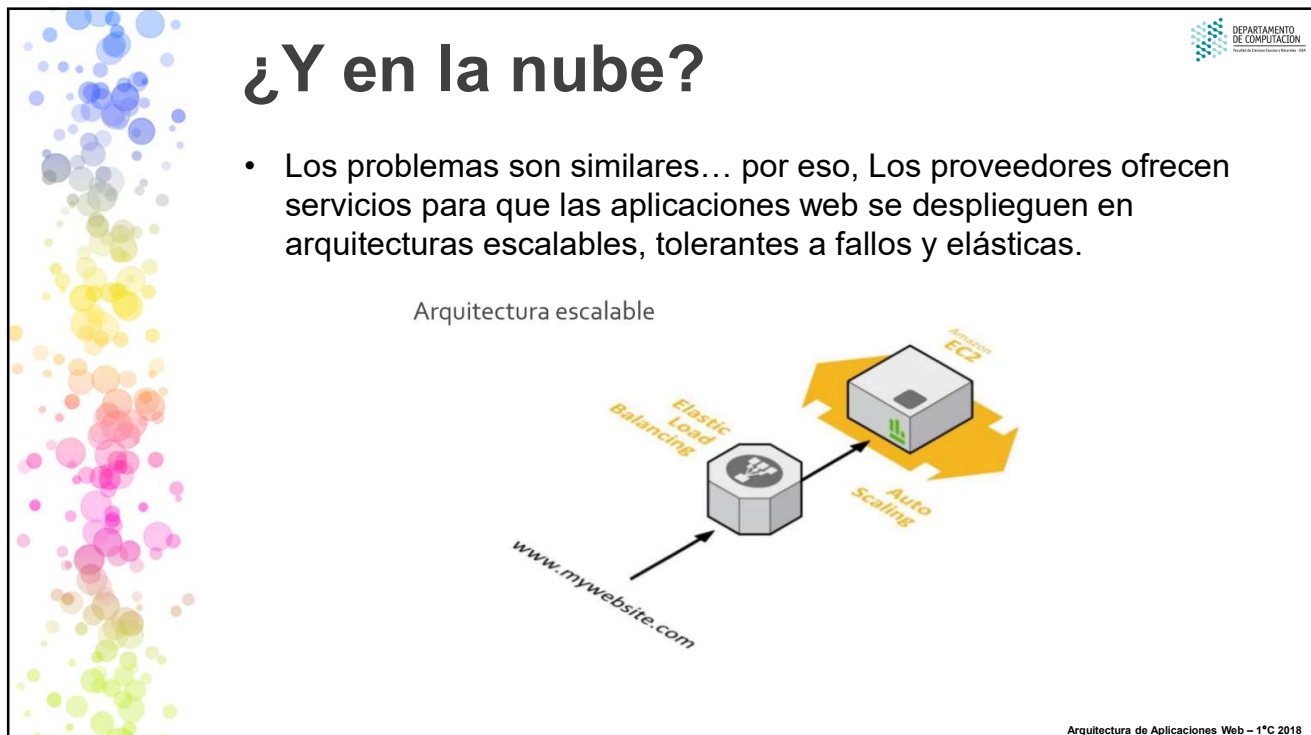
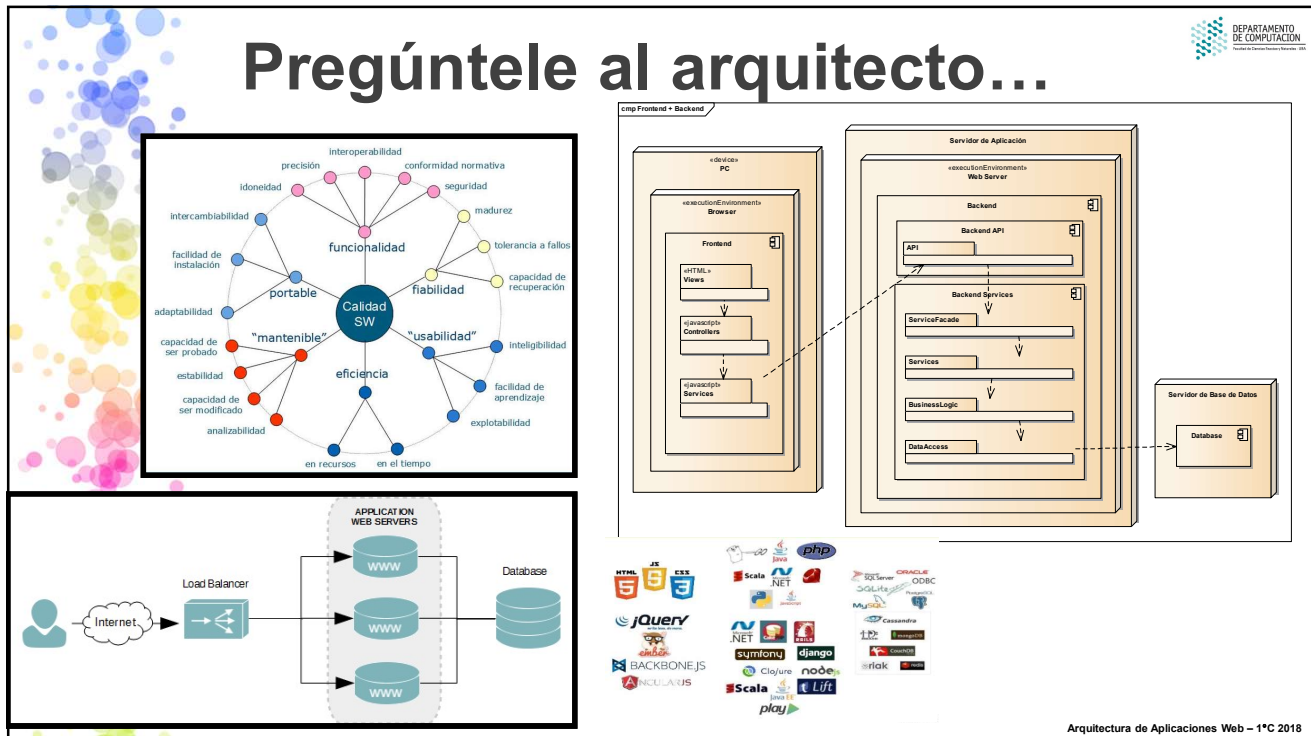


Atributos de Calidad

- ✓ Diferentes aspectos de la calidad:
 - ☐ **Interna:** medible a partir de las características intrínsecas, como el código fuente.
 - ☐ **Externa:** medible en el comportamiento del producto, como en una prueba.
 - ☐ **En uso:** durante la utilización efectiva por parte del usuario.

Arquitectura de Aplicaciones Web – 1°C 2018

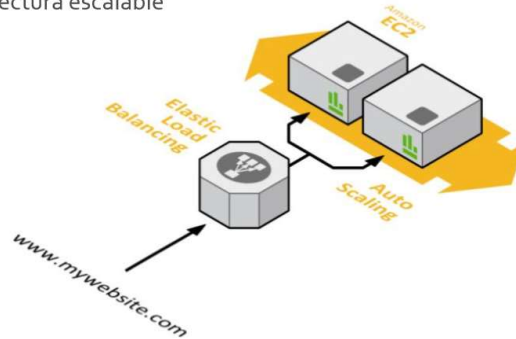




¿Y en la nube?

- Los problemas son similares... por eso, Los proveedores ofrecen servicios para que las aplicaciones web se desplieguen en arquitecturas escalables, tolerantes a fallos y elásticas.

Arquitectura escalable

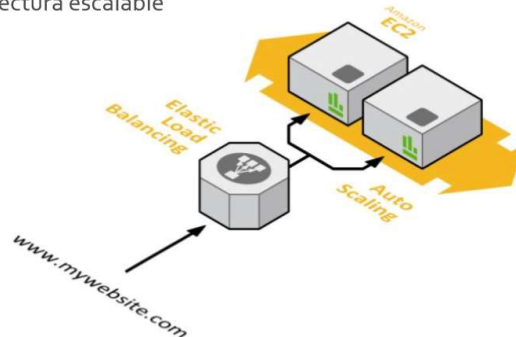


Arquitectura de Aplicaciones Web – 1°C 2018

¿Y en la nube?

- La tolerancia a fallos y la capacidad de escalamiento también aplica a las bases de datos.

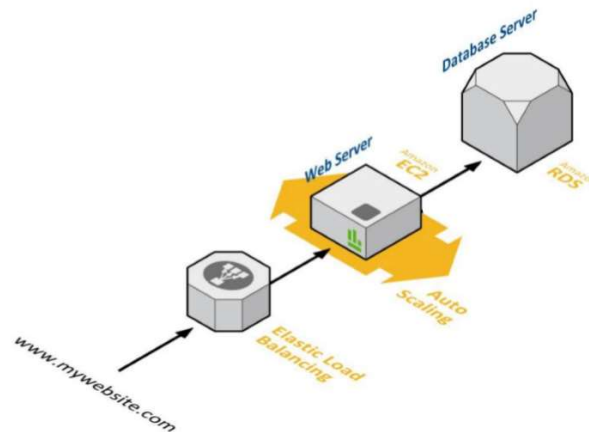
Arquitectura escalable



Arquitectura de Aplicaciones Web – 1°C 2018

¿Y en la nube?

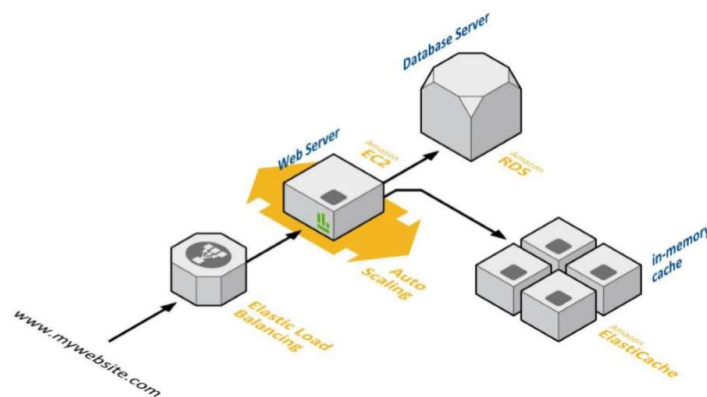
- La tolerancia a fallos y la capacidad de escalamiento también aplica a las bases de datos.



Arquitectura de Aplicaciones Web – 1°C 2018

¿Y en la nube?

- El uso de memorias cache también es una alternativa...



Arquitectura de Aplicaciones Web – 1°C 2018

