

The diagram illustrates various network components and protocols. On the left, a green box contains the text "Redes de Comunicación" and "TCP/IP - HTTP", next to a white network node icon. In the center, a large orange box contains the text "Arquitecturas de Software" and a white puzzle piece icon. To the right, a yellow box contains a diagram showing the layers of the OSI model: Application, Transport, Network, and Link, each connected to specific protocols like Telnet, DNS, TCP, UDP, IP, ICMP, Ethernet, and Token Ring.

Temas de Hoy

Arquitecturas de Software

Redes de Comunicación

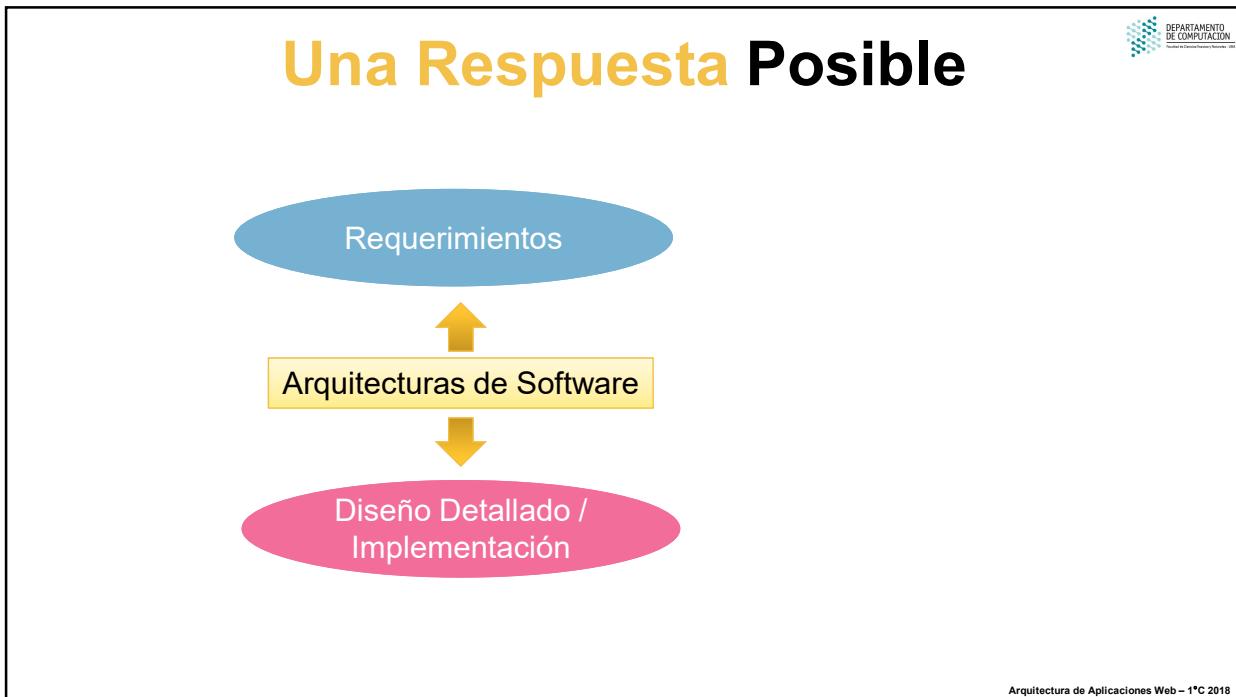
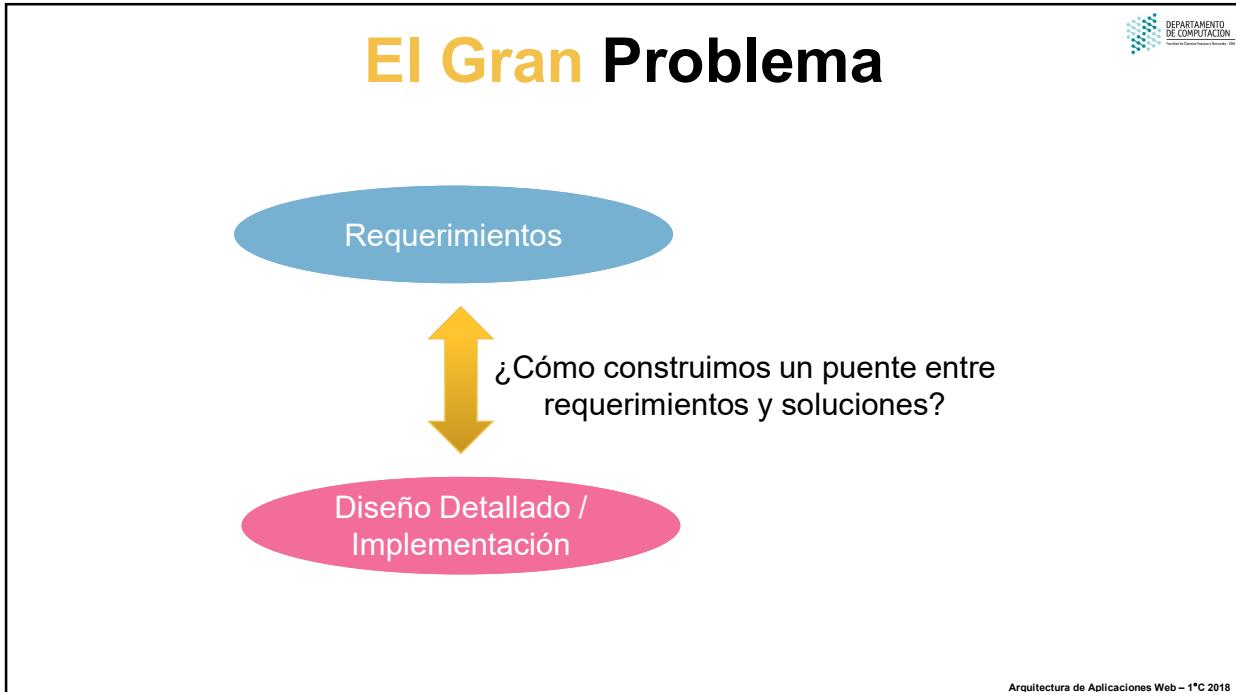
TCP/IP - HTTP

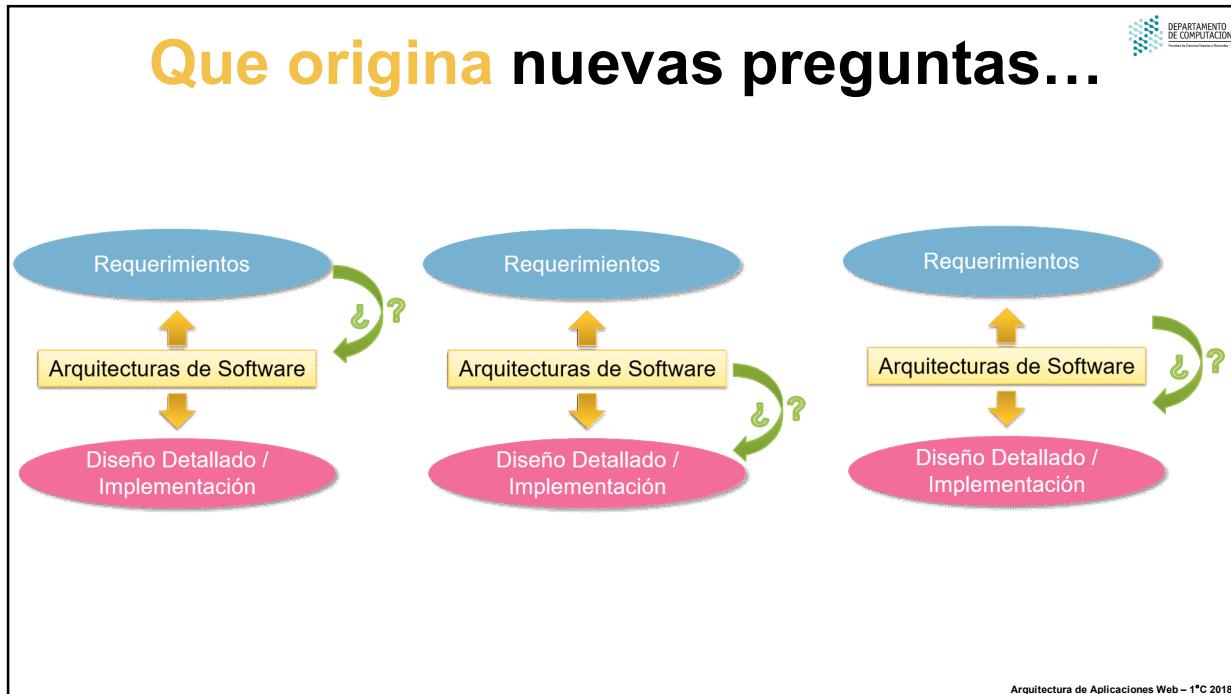
Arquitectura de Aplicaciones Web – 1ºC 2018

A computer monitor displays a detailed diagram of a network architecture, featuring a cityscape on a server, various network components like routers, switches, and databases, and icons representing different network types such as Intranet, Internet, and Firewalls. The monitor sits on a desk with a keyboard and mouse, all set against a background of colorful, floating circular bubbles.

Arquitecturas de Software

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA





Reseña Histórica

1968 – Dijkstra introduce la noción de “niveles de abstracción” y de Sistemas operativos organizados en “capas”.

1969 – Conferencia NATO*, P. I. Sharp sostiene:

Pienso que tenemos algo, aparte de la ingeniería de software: algo de lo que hemos hablado muy poco pero que deberíamos poner sobre el tapete y concentrar la atención en ello. Es la cuestión de la arquitectura de software. La arquitectura es diferente de la ingeniería. Como ejemplo de lo que quiero decir, echemos una mirada a OS/360. Partes de OS/360 están extremadamente bien codificadas.

Partes de OS, si vamos al detalle, han utilizado técnicas que hemos acordado constituyen buena práctica de programación. La razón de que OS sea un amontonamiento amorfo de programas es que no tuvo arquitecto. Su diseño fue delegado a series de grupos de ingenieros, cada uno de los cuales inventó su propia arquitectura. Y cuando esos pedazos se clavaron todos juntos no produjeron una tersa y bella pieza de software.

* La primera conferencia sobre Ingeniería de Software fue en 1968, financiada por la OTAN

Arquitectura de Aplicaciones Web – 1ºC 2018



Reseña Histórica

Además sostuvo (1969):

Lo que sucede es que las especificaciones de software se consideran especificaciones funcionales. Sólo hablamos sobre lo que queremos que haga el programa.

Es mi creencia que cualquiera que sea responsable de la implementación de una pieza de software debe especificar más que esto. Debe especificar el diseño, la forma; y dentro de ese marco de referencia, los programadores e ingenieros deben crear algo. Ningún ingeniero o programador, ninguna herramienta de programación, nos ayudará, o ayudará al negocio del software, a maquillar un diseño feo.

El control, la administración, la educación y todas las cosas buenas de las que hablamos son importantes; pero la gente que implementa debe entender lo que el arquitecto tiene en mente.

Luego, por unos años, “arquitectura” fue una metáfora de la que se echó mano cada tanto, pero sin precisión semántica ni consistencia pragmática.

Arquitectura de Aplicaciones Web – 1ºC 2018



Reseña Histórica

1975 – Brooks* (diseñador del OS/360)

Utilizaba el concepto de arquitectura del sistema para designar “la especificación completa y detallada de la interfaz de usuario” y consideraba que el arquitecto es un agente del usuario, igual que lo es quien diseña su casa. El concepto de AS actual está alejado de esa línea de pensamiento.

Distinguía entre arquitectura e implementación; mientras aquella decía qué hacer, la implementación se ocupa de cómo.

A diferencia de Dijkstra (formalismo matemático), Brooks considera las variables humanas.

**The mythical man-month*

Arquitectura de Aplicaciones Web – 1ºC 2018



Reseña Histórica

• Década del 70 – **Diseño estructurado** y los primeros **modelos explícitos** de desarrollo de software.

• David Parnas desarrolla temas como:

- **Descomposición** de Sistemas
- **Ocultamiento** de la Información
- **Estructuras** de Software
- Familias de Programas

*Enfatizando siempre la búsqueda de **calidad del software**, medible en términos de economías en los procesos de desarrollo y mantenimiento.*

Arquitectura de Aplicaciones Web – 1ºC 2018



Reseña Histórica

• Decía Parnas:

“Las decisiones tempranas de desarrollo serían las que probablemente permanecerían invariantes en el desarrollo ulterior de una solución

*Esas “decisiones tempranas” constituyen de hecho lo que hoy se llamarían **decisiones arquitectónicas**. ”*

La elección de la estructura correcta sintetiza, como ninguna otra expresión, el programa y la razón de ser de la Arquitectura de Software.

Arquitectura de Aplicaciones Web – 1ºC 2018



Reseña Histórica



- Sobre las familias de programas
(o un anticipo de los estilos arquitectónico)

“Una familia de programas es un **conjunto de programas** (no todos los cuales han sido construidos o lo serán alguna vez) a los cuales **es provechoso o útil considerar como grupo**. Esto evita el uso de conceptos ambiguos tales como “similitud funcional” que surgen a veces cuando se describen dominios.

Por ejemplo, los ambientes de ingeniería de software y los juegos de video no se consideran usualmente en el mismo dominio, aunque podrían considerarse miembros de la misma familia de programas en una discusión sobre herramientas que ayuden a construir interfaces gráficas, que casualmente ambos utilizan.”

Arquitectura de Aplicaciones Web – 1ºC 2018

Reseña Histórica



- 1984, Mary Shaw

Reivindica las **abstracciones de alto nivel**, reclamando un espacio para esa reflexión y augurando que el uso de esas abstracciones en el proceso de desarrollo pueden resultar en “**un nivel de arquitectura de software en el diseño**”.

- Mary Shaw. “Abstraction Techniques in Modern Programming Languages”. *IEEE Software*, Octubre, pp. 10-26, 1984.
- Mary Shaw. “Large Scale Systems Require Higher- Level Abstraction”. *Proceedings of Fifth International Workshop on Software Specification and Design*, IEEE Computer Society, pp. 143-146, 1989.

Años 80': Programación Estructurada → Programación Orientada a Objetos

Arquitectura de Aplicaciones Web – 1ºC 2018

Reseña Histórica

- 1992, Perry y Wolf
 - El primer estudio en que aparece la expresión “arquitectura de software” en el sentido en que hoy lo conocemos.
 - Proponen concebir la AS por analogía con la arquitectura de edificios, una analogía de la que luego algunos abusaron, otros encontraron útil y para unos pocos consideraron inaceptable.
- Presentan un modelo para la arquitectura de software que consiste en tres componentes: elementos, forma y razón (rationale).
 - **Elementos:** son elementos ya sea de procesamiento, datos o conexión.
 - **Forma:** las propiedades de, y las relaciones entre, los elementos, es decir, restricciones operadas sobre ellos.
 - **Razón:** proporciona una base subyacente para la arquitectura en términos de las restricciones del sistema, que lo más frecuente es que se deriven de los requerimientos del sistema.

La década de 1990, creemos, será la década de la arquitectura de software. Usamos el término “arquitectura” en contraste con “diseño”, para evocar nociones de codificación, de abstracción, de estándares, de entrenamiento formal (de los arquitectos de software) y de estilo.

Reseña Histórica

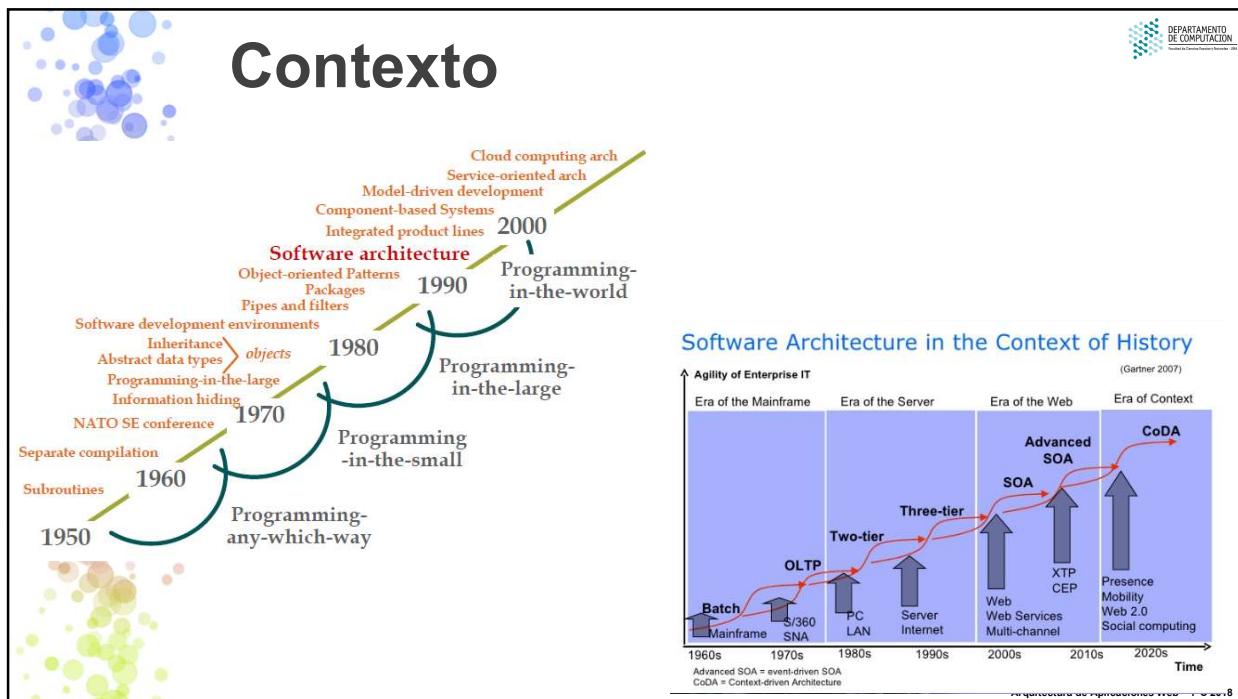
- 1994, Surge también la programación basada en **Componentes**.
- 1995, Surgimiento de los patrones, cristalizada en dos textos fundamentales:
 - Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. *Design Patterns: Elements of reusable object-oriented software*. Reading, Addison-Wesley, 1995.
 - Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal. *Pattern-oriented software architecture – A system of patterns*. John Wiley & Sons, 1996.
- 1996, Paul Clements, promueve un modelo donde la AS debe ser más de **integración de componentes** pre-programados que de programación.
- Finales de los 90's:
 - Homogenización de la terminología en Arquitectura de Software
 - Tipificación de los estilos arquitectónicos
 - Lenguajes de descripción de arquitectura (ADLs)
 - Se consolidó la concepción de las vistas arquitectónicas, reconocidas en todos y cada uno de los frameworks generalizadores que se han propuesto (4+1, TOGAF, RM/ODP, IEEE)

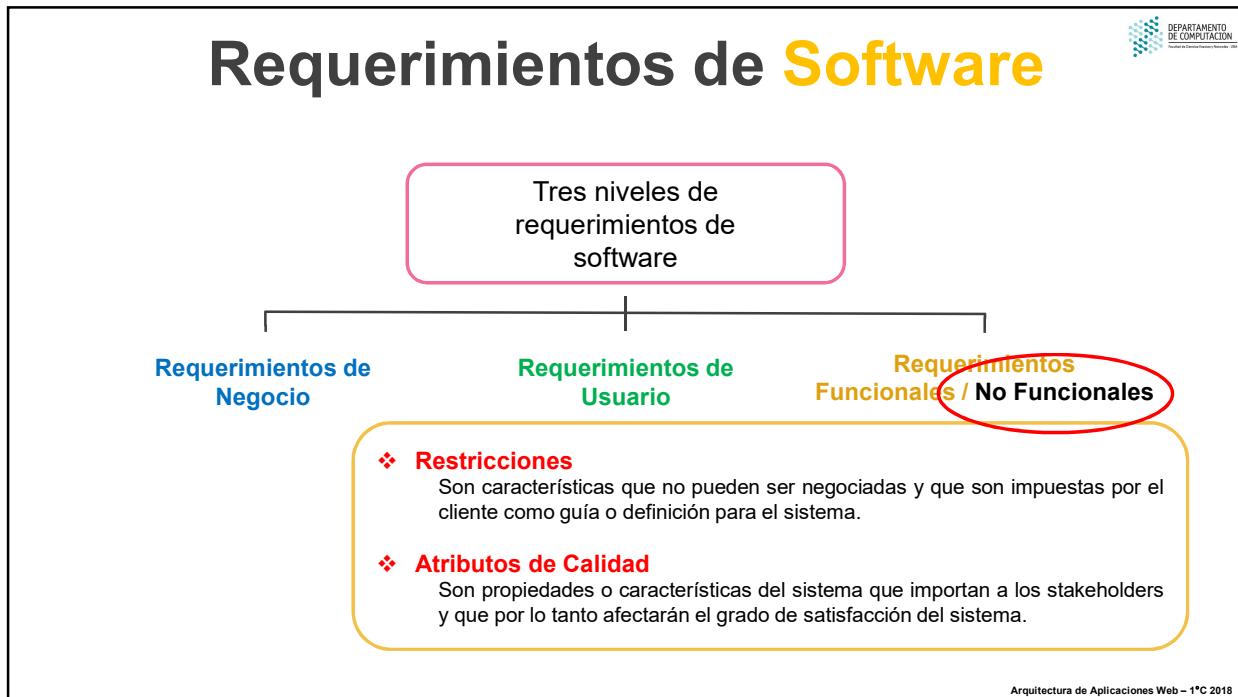
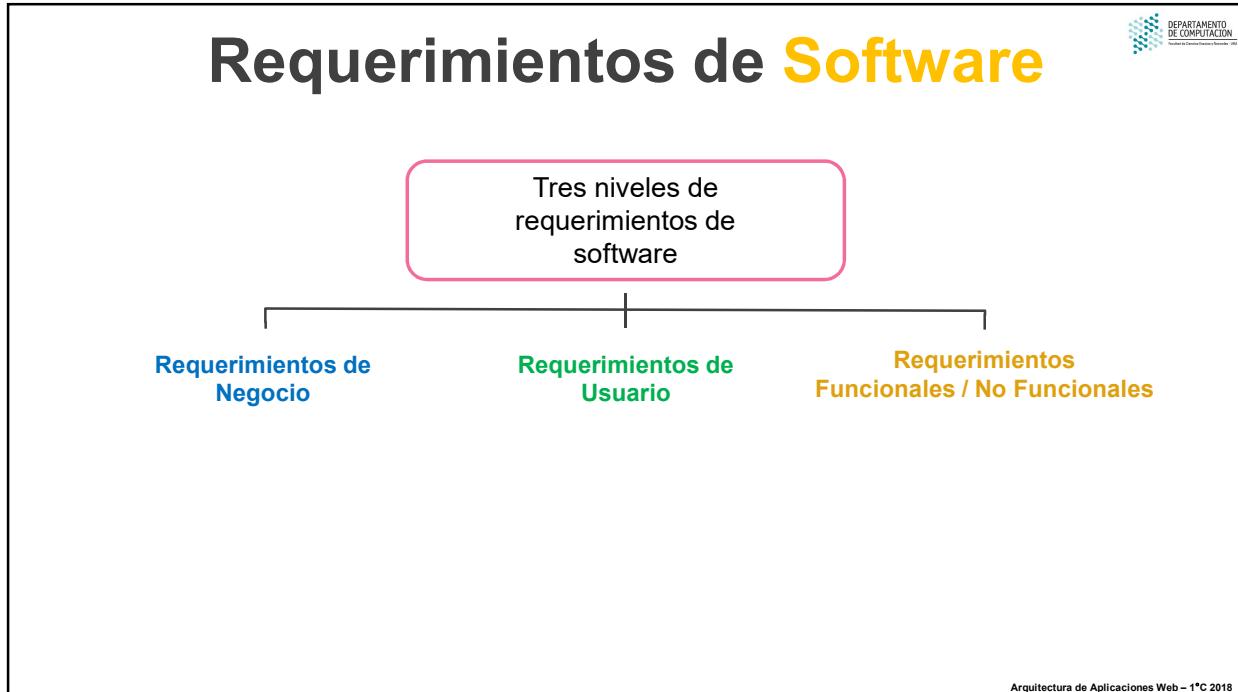
Reseña Histórica

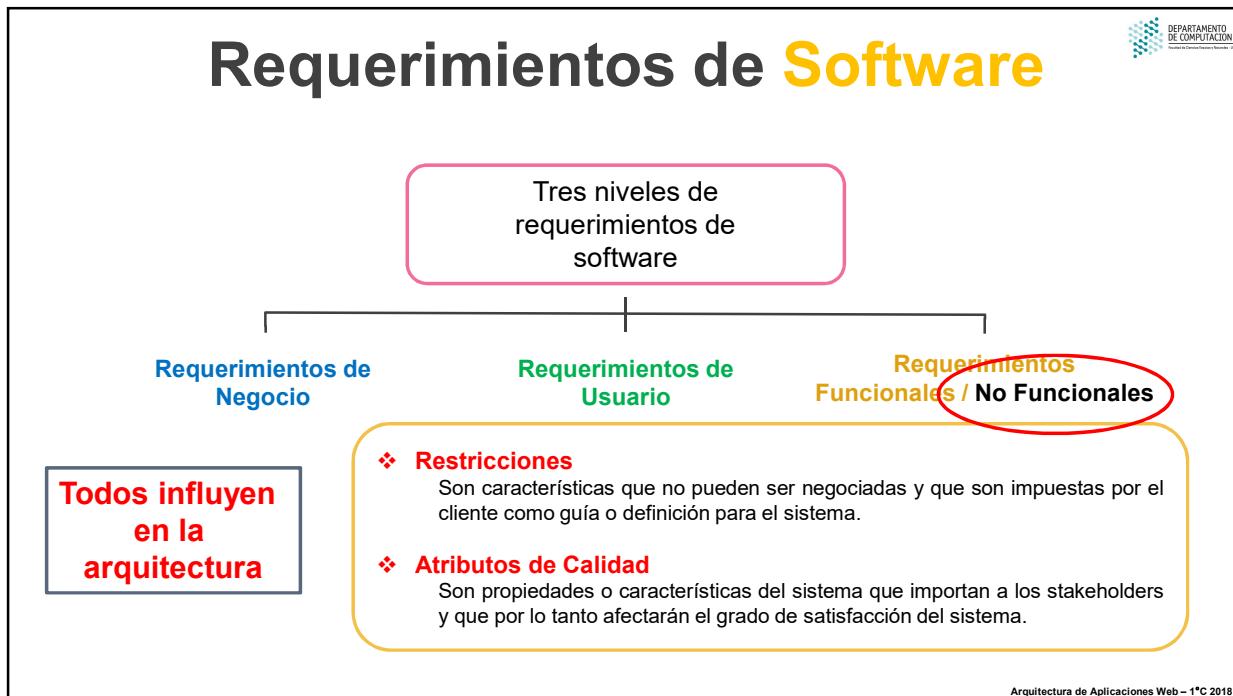
DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- 2000, Roy Fielding, presentó el modelo REST, el cual establece definitivamente el tema de las tecnologías de Internet y los modelos orientados a servicios y recursos.
- Se publica la versión definitiva de la recomendación IEEE Std 1471, que :
 - Procura homogenizar y ordenar la nomenclatura de descripción arquitectónica.
 - Homologa los estilos como un modelo fundamental de representación conceptual.
 - ANSI/IEEE 1471-2000, *Recommended Practice for Architecture Description of Software-Intensive Systems*.
- Siguiente paso...
 - Vistas, Viewtypes y estilos...

Arquitectura de Aplicaciones Web – 1°C 2018









Atributos de Calidad

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- 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 – 1ºC 2018



Atributos de Calidad

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- Suelen estar pobemente 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 – 1ºC 2018

Atributos de Calidad

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- ✓ 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 – 1ºC 2018

Atributos de Calidad

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

ISO 9126 CERTIFIED

- ✓ Distintas clasificaciones de atributos de calidad
 - ✓ Estándares y Certificaciones
 - ✓ SEI, IEEE, etc...

El diagrama ilustra la jerarquía y la interacción entre los atributos de calidad. Se muestra un flujo vertical de "Proceso" a "Producto" y finalmente al "Efecto del Producto".

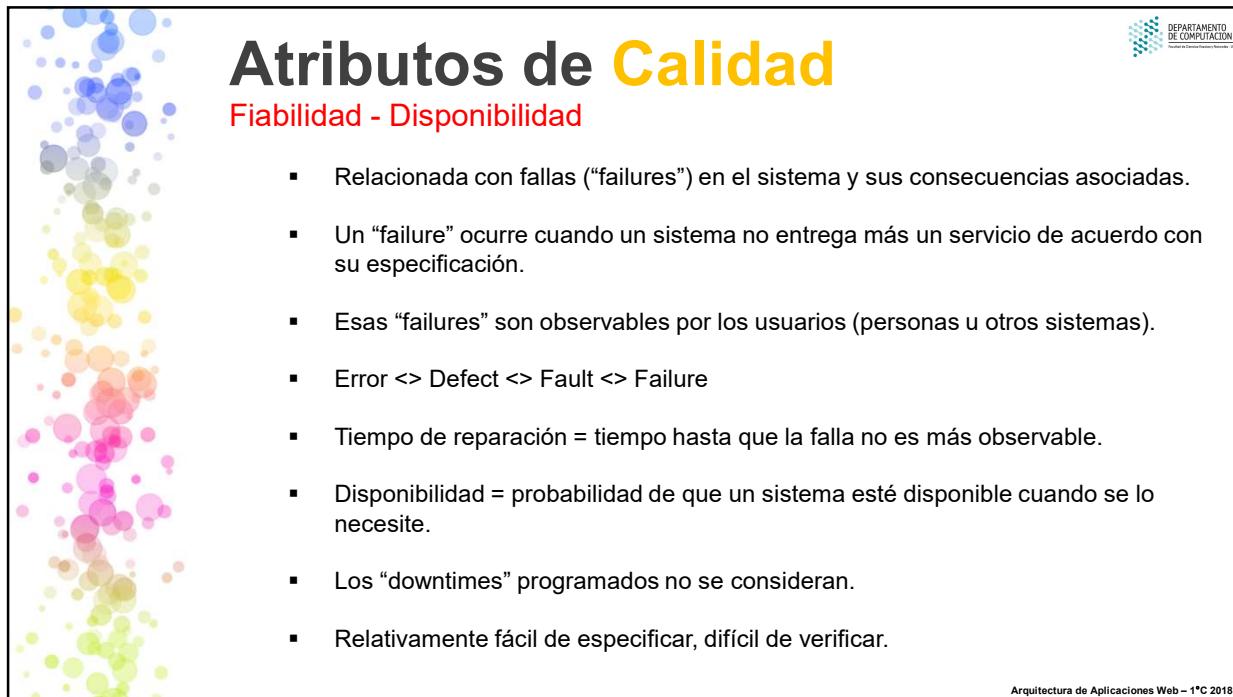
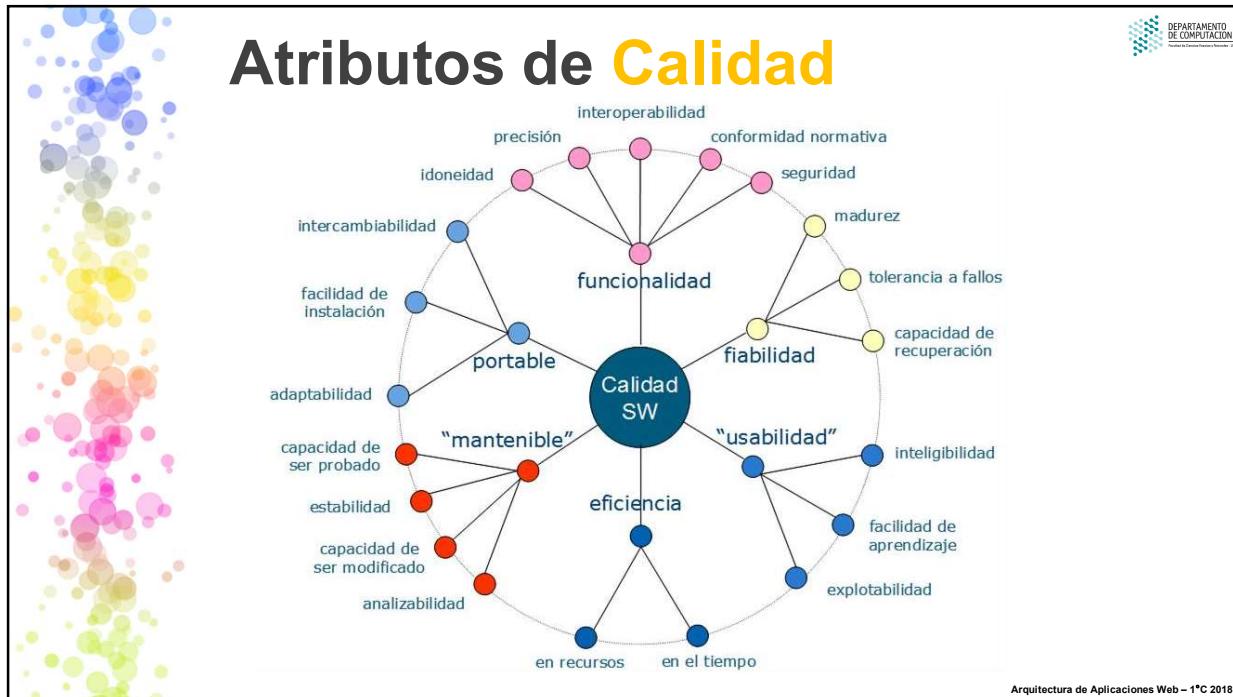
- Proceso:** Un círculo azul que incluye el "Proceso de Calidad".
- Producto:** Tres círculos azules que representan la "Calidad Interna", la "Calidad Externa" y la "Calidad en Uso".
- Efecto del Producto:** Un cuarto círculo azul que incluye la "Calidad en Uso".

Las interacciones se representan mediante flechas:

- Entre el Proceso y la Calidad Interna: "Influye" (arriba) y "Depende de" (abajo).
- Entre la Calidad Interna y la Calidad Externa: "Influye" (arriba) y "Depende de" (abajo).
- Entre la Calidad Externa y la Calidad en Uso: "Influye" (arriba) y "Depende de" (abajo).
- Entre la Calidad en Uso y el Efecto del Producto: "Influye" (arriba) y "Depende de" (abajo).
- Entre el Proceso y el Efecto del Producto: Una flecha horizontal amarilla que pasa por la Calidad Interna, la Calidad Externa y la Calidad en Uso, etiquetada como "Desarrollo".
- Entre el Efecto del Producto y el Usuario: Una flecha horizontal amarilla que pasa por la Calidad en Uso, etiquetada como "Usuario".
- Entre el Proceso y el Usuario: Una flecha vertical amarilla que pasa por la Calidad Interna, la Calidad Externa y la Calidad en Uso, etiquetada como "Medición del Proceso".
- Entre el Efecto del Producto y la Calidad en Uso: Una flecha vertical amarilla que pasa por la Calidad Interna, la Calidad Externa y la Calidad en Uso, etiquetada como "Medición de la Calidad en Uso".
- Entre la Calidad Interna, la Calidad Externa y la Calidad en Uso: Flechas horizontales amarillas que conectan directamente entre ellos, etiquetadas como "Medición Interna", "Medición Externa" y "Medición de la Calidad en Uso".

Fundamentos de ISO 9126

Arquitectura de Aplicaciones Web – 1ºC 2018





Atributos de Calidad

Facilidad de Cambios

- Relacionada con el costo de los cambios. Uno de los atributos de calidad más difíciles de expresar.
- Temas importantes:
 - ¿Qué puede cambiar?
 - Funcionalidad
 - Plataforma
 - Otros atributos de calidad
 - Interfaces
 - ¿Quién y dónde se hace el cambio?
 - Usuarios, desarrolladores, administradores
 - Código, configuración, parametrización
- Una vez que un cambio se especifica, debe ser diseñado, implementado, probado y liberado.

Arquitectura de Aplicaciones Web – 1ºC 2018



Atributos de Calidad

Performance

- Relacionada con el tiempo que le lleva al sistema responder a un evento que ocurre (interrupciones, mensajes, pedidos de usuarios o paso del tiempo).
 - Latencia: tiempo entre la llegada del estímulo y el inicio de la respuesta del sistema
 - “Jitter”: variación en la latencia
 - Deadlines: límites de tiempo para un proceso
 - Throughput: cantidad de transacciones que el sistema puede procesar en un período de tiempo
 - Eventos no procesados
- Difícil de expresar. Depende de volúmenes del sistema, equipamiento en uso y versiones de sistema operativo y otros software de base.

Arquitectura de Aplicaciones Web – 1ºC 2018



Atributos de Calidad

Seguridad

- Habilidad de un sistema para resistir usos no autorizados y seguir proveyendo sus servicios a usuarios legítimos. Algunos temas que incluye:
 - *Nonrepudiation*: mecanismos para asegurar que quienes hicieron algo no puedan negarlo.
 - *Confidencialidad*: propiedad por la cual datos o servicios son protegidos de accesos no autorizados.
 - *Integridad*: propiedad por la cual datos o servicios se brindan como fue previsto.
 - *Disponibilidad* (en el contexto de seguridad): que un sistema esté disponible para su uso legítimo.
 - *Auditabilidad*: habilidad de un sistema para hacer un seguimiento de actividades realizadas.



Arquitectura de Aplicaciones Web – 1ºC 2018

Atributos de Calidad

Usabilidad

- Relacionada con la facilidad con la cual un usuario puede cumplir una tarea o utilizar un servicio ofrecido por el sistema y el tipo de soporte que provee el sistema.
 - Aprender la funcionalidad del sistema.
 - Usar el sistema eficientemente.
 - Minimizar el impacto de los errores.
 - Adaptar el sistema a las necesidades de los usuarios.
 - Aumentar confianza y satisfacción.



Arquitectura de Aplicaciones Web – 1ºC 2018

Atributos de Calidad

Escalabilidad

- Una medida de qué tan bien una solución sigue cumpliendo con sus requerimientos al cambiar los volúmenes del problema que resuelve.

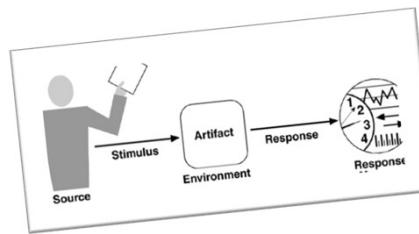
Portabilidad

- Facilidad de un sistema para poder ser operado en distintas plataformas.

Facilidad de Testing

- Posibilidad de ver el estado interno de la aplicación.

Especificación de Atributos de Calidad

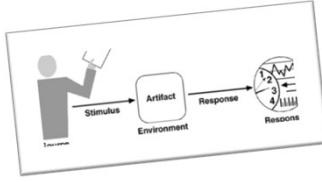


Quality Attribute Scenario (SEI)

- Fuente del estímulo: Interna o externa
- Estímulo: condición que debe ser tenida en cuenta al llegar al sistema
- Entorno: condiciones en las cuales ocurre el estímulo
- Artifact: el sistema o partes de él afectadas por el estímulo
- Response: qué hace el sistema ante la llegada del estímulo
- Response measure: cuantificación de un atributo de la respuesta



Especificación de Atributos de Calidad

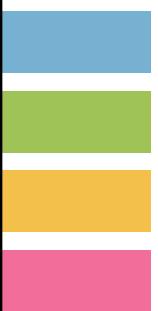


Escenario de performance:

"Se requiere que el sistema sea capaz de realizar las operaciones de autenticación y cobro en a lo sumo 1 segundo ya que, de otro modo, no sería bien recibido por los usuarios, ni los choferes de colectivos. Esto es especialmente crítico en horas pico."

- Fuente: Pasajero
- Estímulo: Se aproxima o inserta una tarjeta para realizar el pago de un pasaje
- Entorno: En operación normal y hora pico
- Artefacto: Terminal de Cobro
- Respuesta: Se concreta la operación de la venta del pasaje.
- Medición de la respuesta: tiempo de respuesta total <= 1 seg.

Arquitectura de Aplicaciones Web – 1ºC 2018



La arquitectura de un sistema de software:



- Define el sistema en términos de elementos e interacción entre ellos.
- Muestra correspondencia entre requerimientos y elementos del sistema construido.
- Resuelve atributos de calidad en el nivel del sistema, como escalabilidad, flexibilidad, confiabilidad y performance.

Arquitectura de Aplicaciones Web – 1ºC 2018

Analogías con la ingeniería civil...

- ❑ Estilos arquitectónicos: colonial, victoriano, griego
 - Paradigmas de organización de sistemas de software: pipes, layers, events, repositories.
- ❑ Conocimientos específicos para un estilo en particular: cárceles, fábricas automotrices, hospitales, hoteles 5 estrellas
 - Arquitecturas para un dominio específico, llamadas arquitecturas de referencia.

La estructura de los sistemas

- ❑ La arquitectura trata sobre la estructura de los sistemas
 - ❑ Cómo el sistema se descompone en partes
 - ❑ Cómo esas partes interactúan
- ❑ Pero esto lleva a la pregunta: ¿Qué tipos de estructuras?
 - Del código
 - Run-time
 - De deployment
 - Del entorno de desarrollo
 - Work breakdown structures
- ❑ Cada una de estas estructuras puede ser la base para una vista arquitectónica (architectural view)
 - Históricamente el foco estuvo en vistas de código.

Las definiciones más aceptadas

(Bass, Clements)



La arquitectura de software de un sistema de computación es el conjunto de estructuras necesarias para razonar sobre el sistema, y comprende elementos de software, relaciones entre ellos y propiedades de ambos.

- Estilo o patrón arquitectónico
 - ✓ Una descripción de tipos de relaciones y elementos, junto con restricciones sobre cómo deben usarse (ej. "client server").
- Arquitectura de referencia
 - ✓ Una división común de funcionalidad mapeada a elementos que cooperativamente implementan esa funcionalidad y flujos de datos entre ellos.

Arquitectura de Aplicaciones Web – 1ºC 2018

Tres Principios Fundamentales



- Toda aplicación tiene una arquitectura.
- Cada aplicación tiene al menos un arquitecto.
- La "Arquitectura" no es una fase del desarrollo.

La arquitectura es el conjunto de decisiones principales de diseño de un sistema de software.

Arquitectura de Aplicaciones Web – 1ºC 2018

Temas fundamentales (Clemens – 1996)

- Diseño o selección de la arquitectura
 - Cómo crear o elegir.
- Representación de la arquitectura
 - Cómo comunicar.
- Evaluación y análisis de la arquitectura
 - Cómo validar y verificar.
- Desarrollo y evolución basados en arquitectura
 - Cómo construir.
- Recuperación de la arquitectura
 - Cómo descubrir arquitecturas subyacentes (legacy)

Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué hace que una arquitectura sea “buena”?

- ✓ Producto de un **único arquitecto o un pequeño grupo de arquitectos** con un claro líder (Brooks, Mills y otros). “Integridad conceptual”.
- ✓ El equipo de arquitectura debe contar con **requerimientos funcionales y atributos de calidad** requeridos que sean claros.
- ✓ La arquitectura debe estar **documentada**.
- ✓ La arquitectura debe ser **revisada** por los “stakeholders”.
- ✓ Debe ser **evaluada cuantitativamente** antes de que sea tarde.
- ✓ Debe permitir una implementación **incremental**.
- ✓ Módulos bien definidos basados en el **ocultamiento** de la información.
- ✓ Interfaces claramente definidas.
- ✓ Usa un grupo **pequeño y claro** de patrones de interacción.

Arquitectura de Aplicaciones Web – 1ºC 2018

Arquitecturas de Software - Características

- **Representación de alto nivel** de la estructura del sistema describiendo las partes que lo integran.
- Puede incluir los **patrones** que supervisan la composición de sus componentes y las restricciones al aplicar los patrones.
- Trata **aspectos del diseño y desarrollo** que no pueden tratarse adecuadamente dentro de los módulos que forman el sistema.

Arquitecturas de Software - Objetivos

- **Comprender** (abstracción) y mejorar la estructura de las aplicaciones complejas.
- **Reutilizar** dicha **estructura** (o partes de ella) para resolver problemas similares.
- Analizar la **corrección** de la aplicación y su grado de cumplimiento respecto a los requisitos iniciales.
- Permitir el estudio de algunas propiedades específicas del dominio.

Arquitecturas de Software - Objetivos



- Planificar la **evolución** de la aplicación, identificando las partes mutables e inmutables de la misma, así como los costos de los posibles cambios.
- Facilitar la **adaptación al cambio**:
 - Composición
 - Reconfiguración
 - Reutilización
 - Escalabilidad
 - Mantenibilidad, etc.

Arquitectura de Aplicaciones Web – 1ºC 2018

Arquitecturas de Software - Objetivos



- Organización a alto nivel del sistema, incluyendo aspectos como:
 - La descripción
 - Análisis de propiedades relativas a su estructura y control global
 - Los protocolos de comunicación
 - Protocolos de sincronización utilizados
 - La distribución física del sistema y sus componentes, etc

Arquitectura de Aplicaciones Web – 1ºC 2018

Arquitecturas de Software – Fuera del alcance



➤ ¿De qué no se ocupa?

- Diseño detallado
- Diseño de algoritmos
- Diseño de estructuras de datos

Arquitectura de Aplicaciones Web – 1ºC 2018

Más definiciones aceptadas

(Garlan y Shaw, 1993)

Una colección de componentes computacionales - o, simplemente, **componentes** - en conjunto con una descripción de las **interacciones** entre estos componentes, es decir, de los **conectores**.

- La arquitectura de software define
 - **Componentes:** lugar de almacenamiento o computo:
 - Filtros, bases de datos, objetos, TADs
 - **Conectores:** Mediadores entre componentes
 - Llamadas a procedimientos
 - Pipes
 - Broadcast
 - **Propiedades:** Información para construcción y análisis
 - Pre/Post condiciones, invariantes

Arquitectura de Aplicaciones Web – 1ºC 2018



Vistas

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- ❖ Las vistas arquitectónicas representan un aspecto parcial de una arquitectura de software que muestran propiedades específicas del sistema.
- ❖ Una única representación resultaría demasiado compleja.
- ❖ Cada vista representa un comportamiento particular del sistema.
- ❖ La **descripción** de un sistema complejo **no es unidimensional**.
- ❖ Surgen de la agrupación de **elementos arquitectónicos** en “tipos”.
- ❖ Relevancia: depende del **propósito**. Por ejemplo:
 - enunciar la misión de implementación,
 - análisis de atributos de calidad,
 - generación automática de código,
 - planificación,
 - etc.

Arquitectura de Aplicaciones Web – 1ºC 2018



Vistas

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- ❖ Las vistas exponen **atributos de calidad** en diferente grado.
Ejemplos:
 - Vista modular: portabilidad...
 - Vista de deployment: performance, confiabilidad...
- ❖ Reflejan **decisiones arquitectónicas**.
- ❖ Enfatizan **aspectos** e ignoran otros para que el problema sea abordable.
- ❖ Es clave saber cuáles son las vistas relevantes y vincularlas.
- ❖ **Ninguna vista es “LA” arquitectura.**

Arquitectura de Aplicaciones Web – 1ºC 2018

Vistas

(David Parnas
1974)

- ❖ Las vistas arquitectónicas representan un aspecto parcial de una arquitectura de software que muestran propiedades específicas del sistema.
- ❖ Una única representación resultaría demasiado compleja.
- ❖ Cada vista representa un comportamiento particular del sistema.
 - **Estructura de módulos** (*asignación de trabajo, es parte de o comparte el mismo secreto que*).
 - **Estructura de uso** (*programas, depende de la corrección de*).
 - **Estructura de procesos** (*procesos, brinda trabajo computacional a*).

Arquitectura de Aplicaciones Web – 1ºC 2018

Vistas

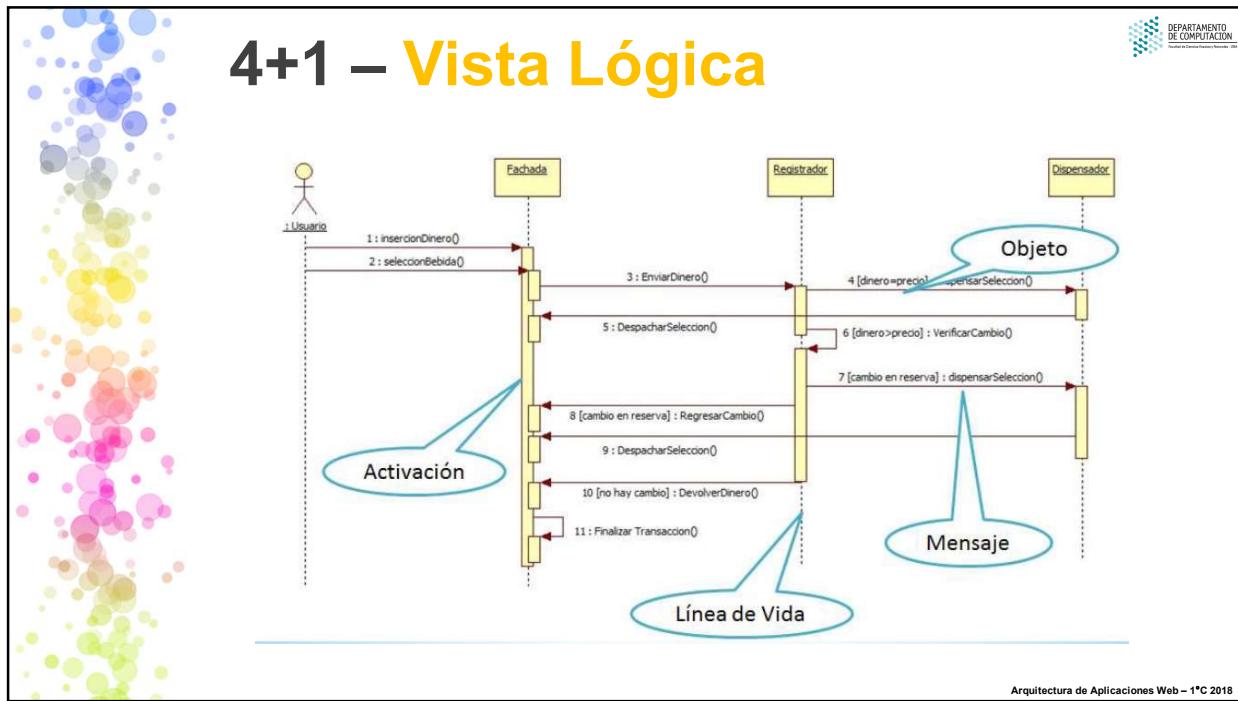
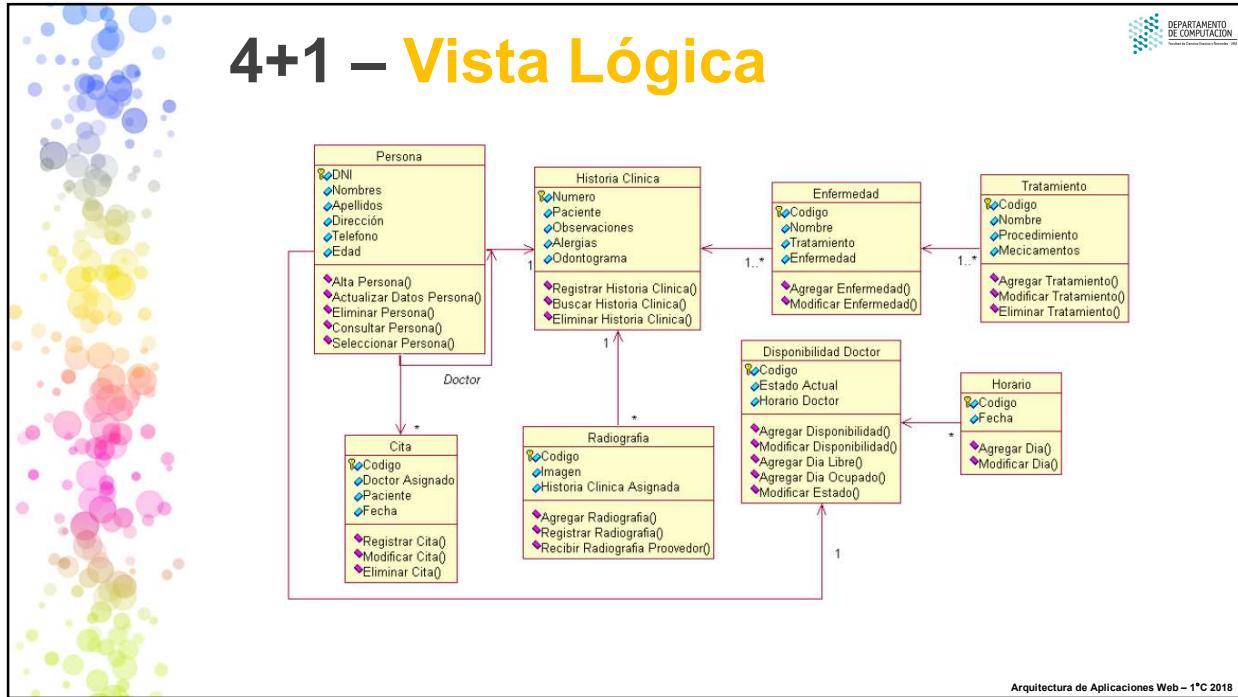
Vista 4+1
(Kruchten, 1995)

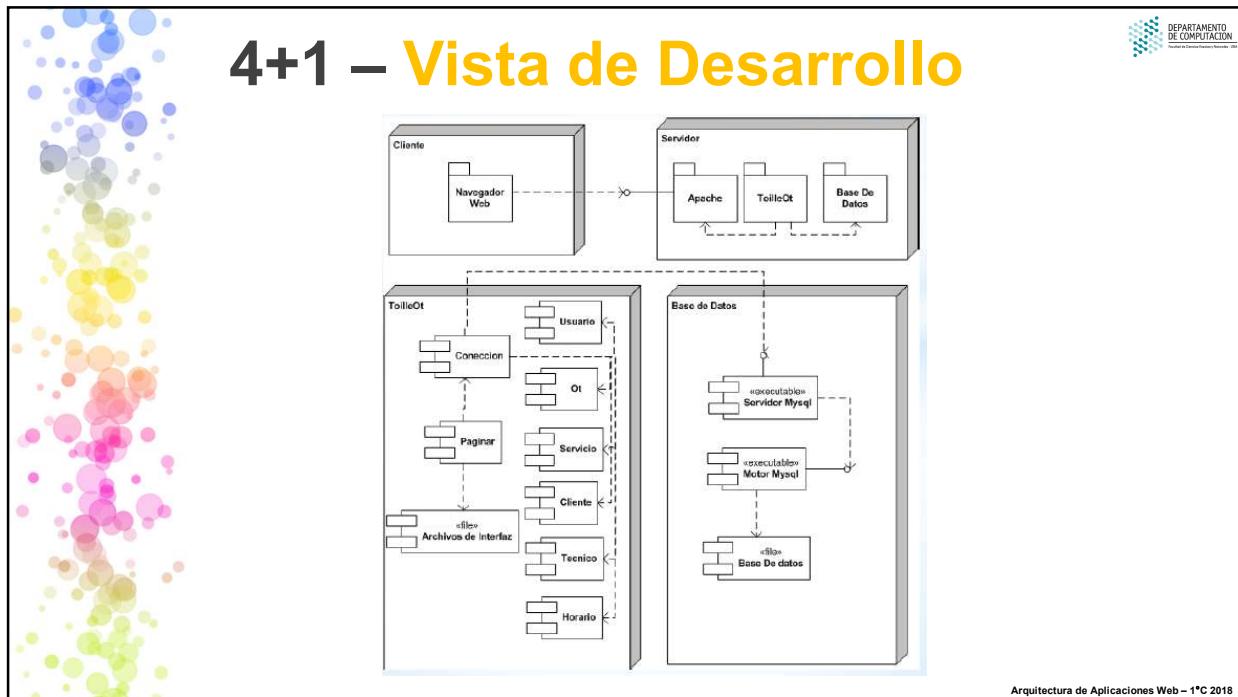
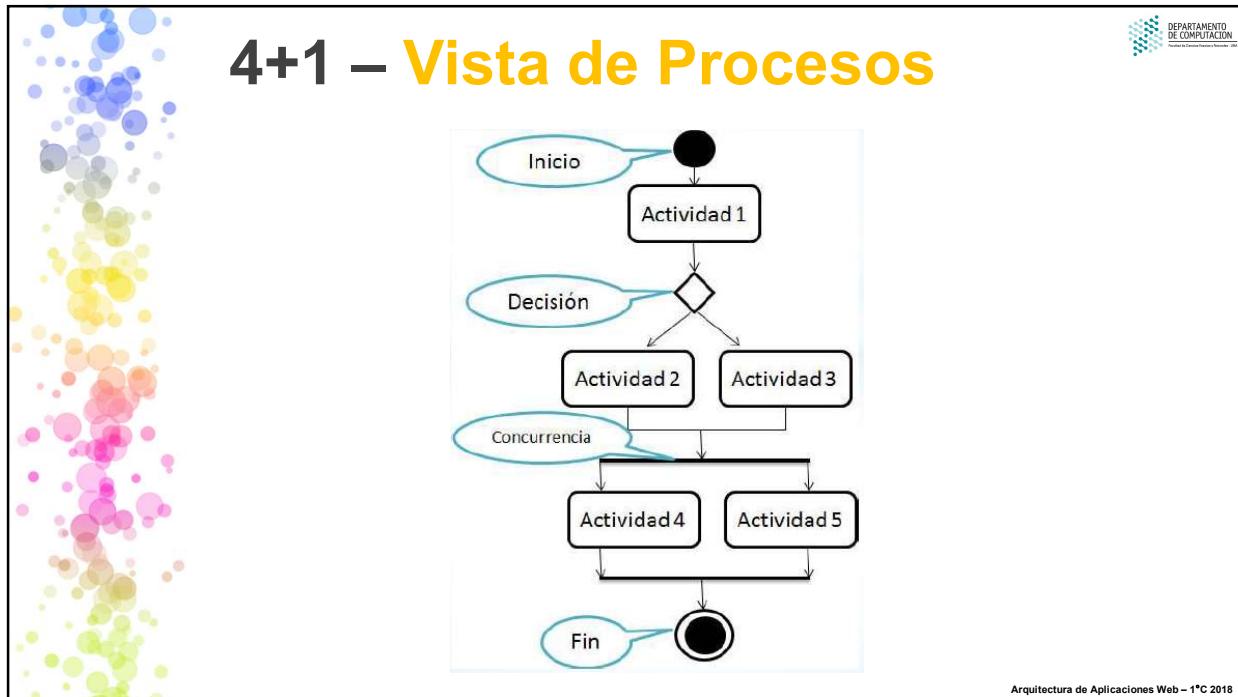
- ❖ Las vistas arquitectónicas representan un aspecto parcial de una arquitectura de software que muestran propiedades específicas del sistema.
- ❖ Una única representación resultaría demasiado compleja.
- ❖ Cada vista representa un comportamiento particular del sistema.

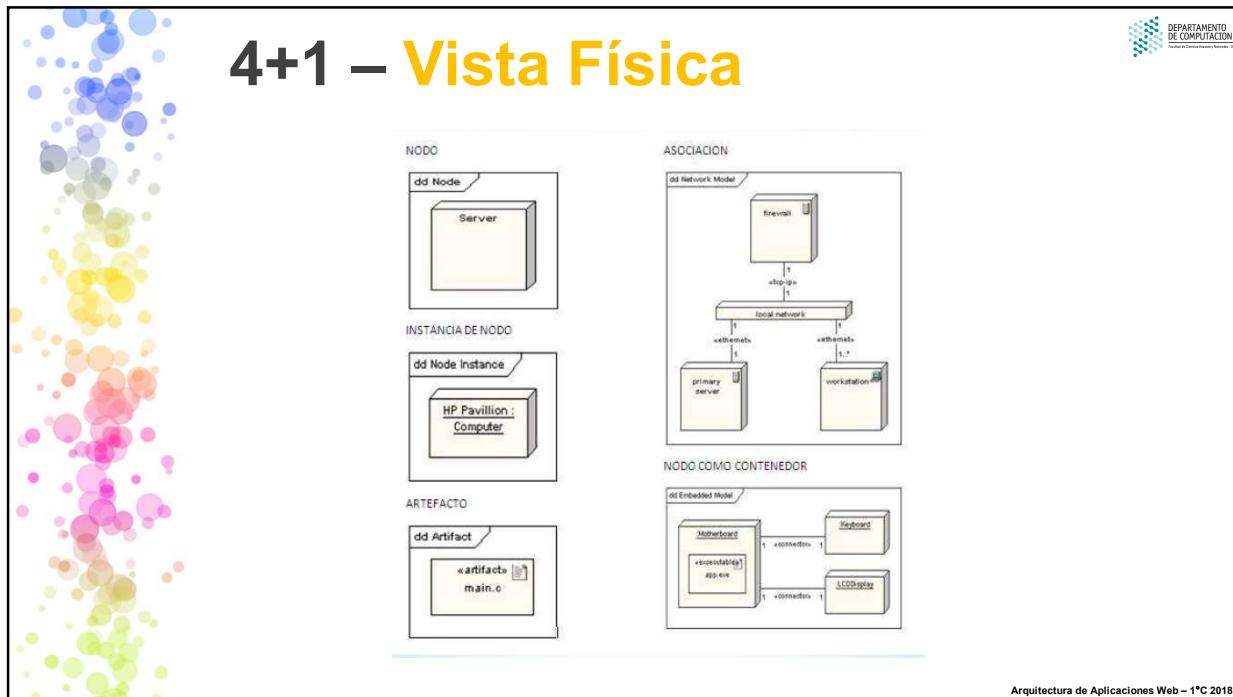
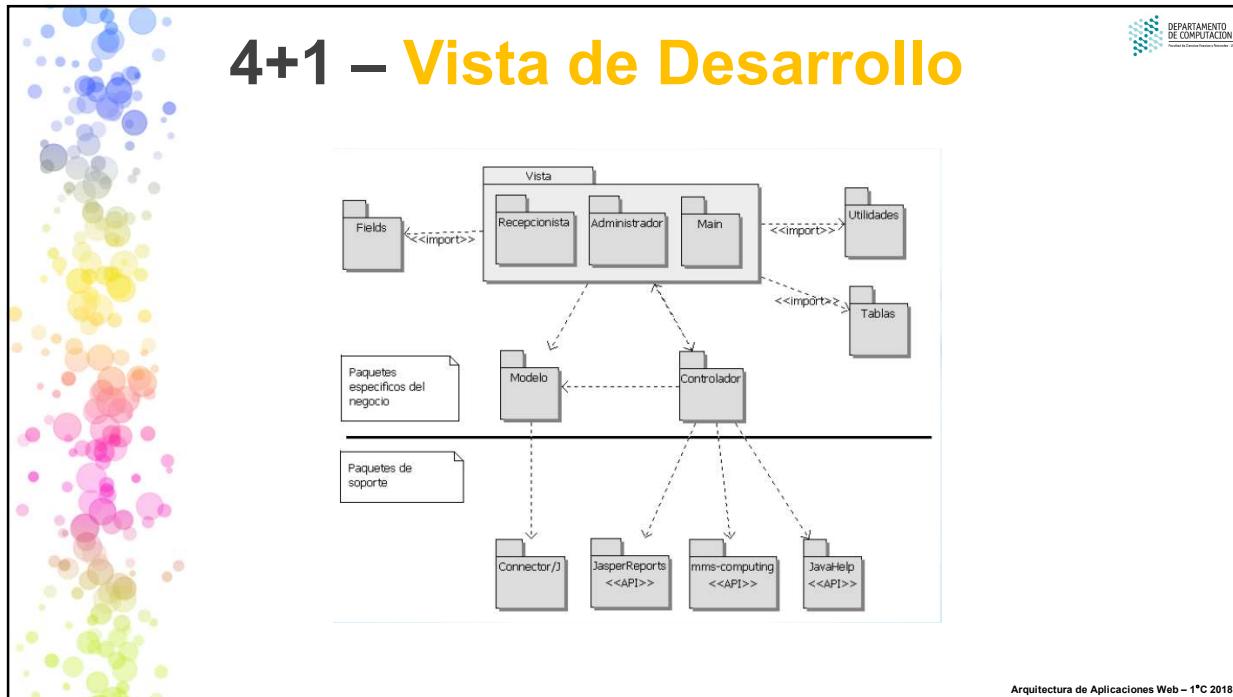
```

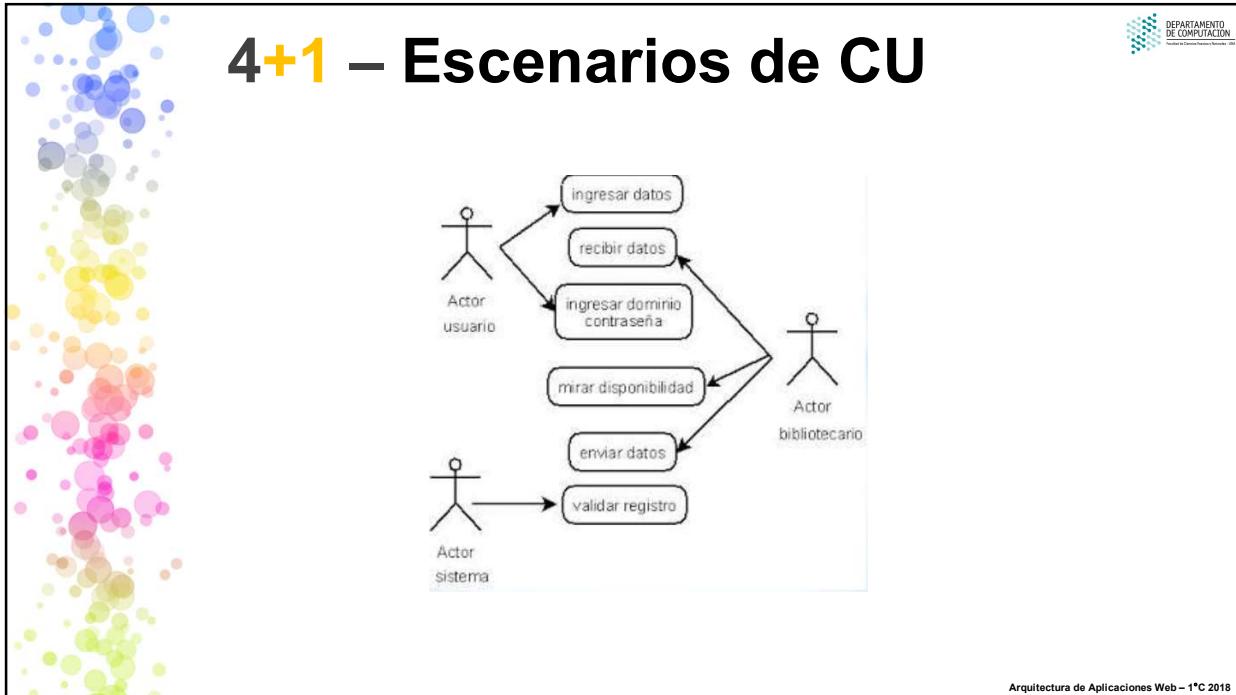
graph TD
    VL[Vista Lógica] --> VE[Vista de Escenarios]
    VL --> VP[Vista de Procesos]
    VL --> VF[Vista Física]
    VE --> VP
    VE --> VF
    VP --> VF
  
```

Arquitectura de Aplicaciones Web – 1ºC 2018









Arquitectura de Aplicaciones Web – 1ºC 2018

Viewtypes

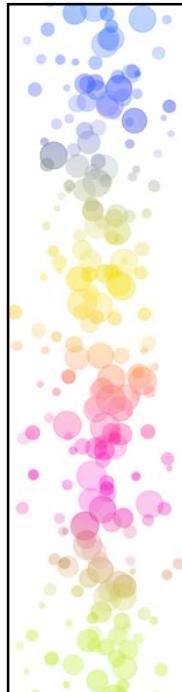
Un sistema tiene varias estructuras

Las estructuras pueden dividirse principalmente en tres grupos:

- *De Módulos*: los módulos son unidades de implementación, una forma de ver al sistema basada en el código (visión estática).
- *De Componentes y Conectores*: aquí los elementos son unidades de run-time **independientes** y los conectores son los mecanismos de comunicación entre esos componentes (visión dinámica).
- *Estructuras de asignación o asignación ("allocation")*: Muestra la relación entre elementos de software y los elementos en uno o más entornos externos en los que el software se crea y ejecuta.

Llamamos “**Viewtypes**” a las vistas de arquitecturas orientadas a estas tres estructuras

Arquitectura de Aplicaciones Web – 1ºC 2018

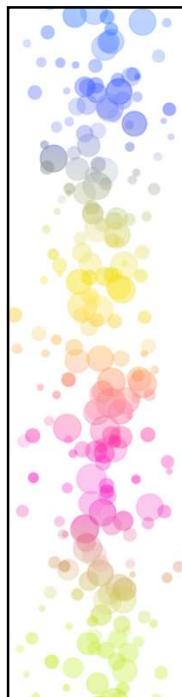


Module Viewtype

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- ❖ Un **módulo** es una unidad de código que implementa un conjunto de responsabilidades.
 - Una clase, una colección de clases, una capa o cualquier descomposición de la unidad de código.
 - Nombres, responsabilidades, visibilidad de las interfaces.
- ❖ Tipos de diagramas:
 - Descomposición (“es parte de”): los módulos tienen relación del tipo “es un submódulo de”
 - Usos (“depende de”): los módulos tienen relación del tipo “usa a”. Se dice que un módulo A usa a B, si la correcta ejecución de B es necesaria para la correcta ejecución de A (no es lo mismo que invocación)
 - Clases (“se comporta como”): los módulos en este caso son clases y las relaciones son de herencia.

Arquitectura de Aplicaciones Web – 1ºC 2018



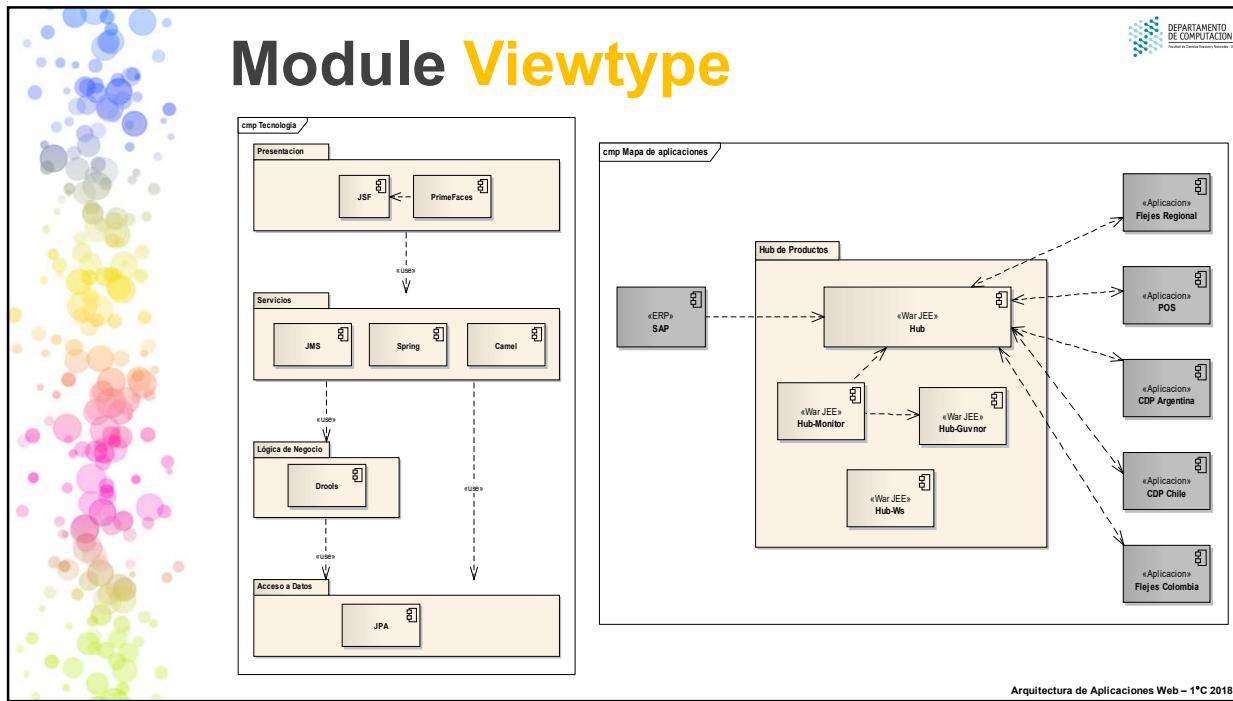
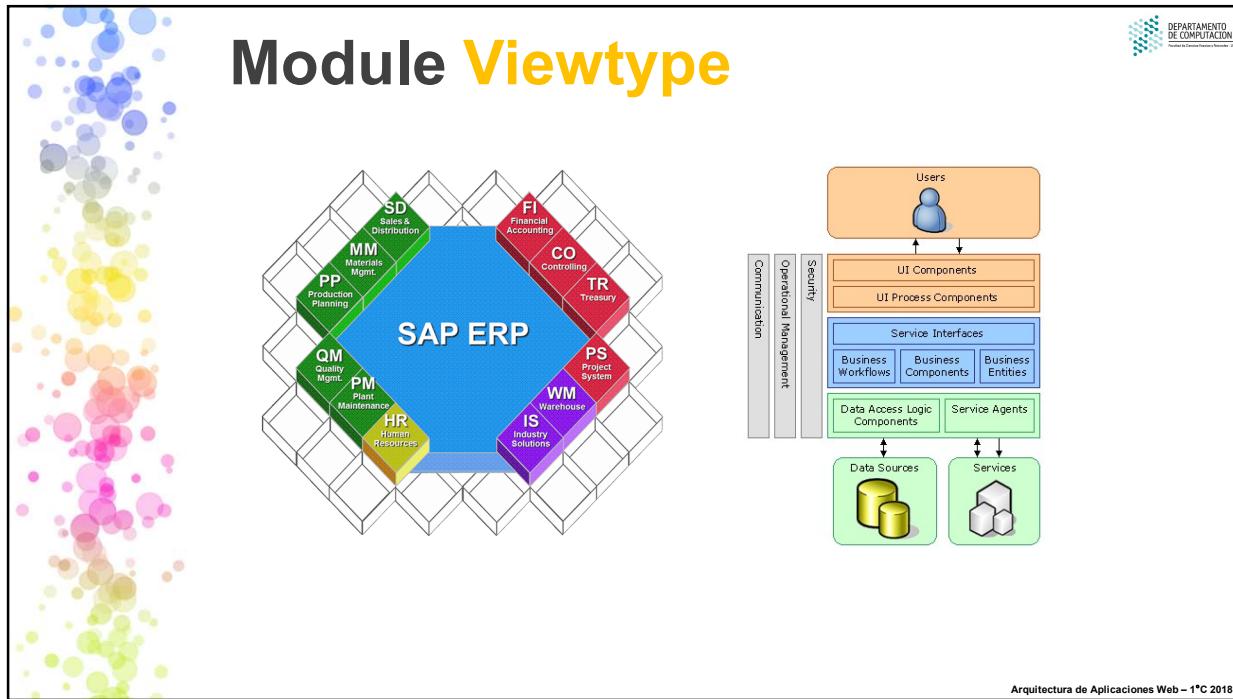
Module Viewtype

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Utilidad

Construcción	Análisis	Comunicación
Organizar el código fuente.	Trazabilidad de Requerimientos. Análisis de Impacto.	Pueden ser utilizadas para explicar las funcionalidades del sistema a alguien no familiarizado con el mismo.

Arquitectura de Aplicaciones Web – 1ºC 2018



C&C Viewtype

- ❖ Estas estructuras están centradas en **procesos** que se comunican.
- ❖ Sus elementos son entidades con manifestación runtime que consumen recursos de ejecución y contribuyen al comportamiento en ejecución del sistema.
- ❖ La configuración del sistema es un grafo conformado por la asociación entre **componentes** y **conectores**.
- ❖ Las entidades runtime son **instancias** de tipos de conector o componente.
- ❖ Los componentes son entidades independientes y sólo se relacionan e interactúan a través de conectores.

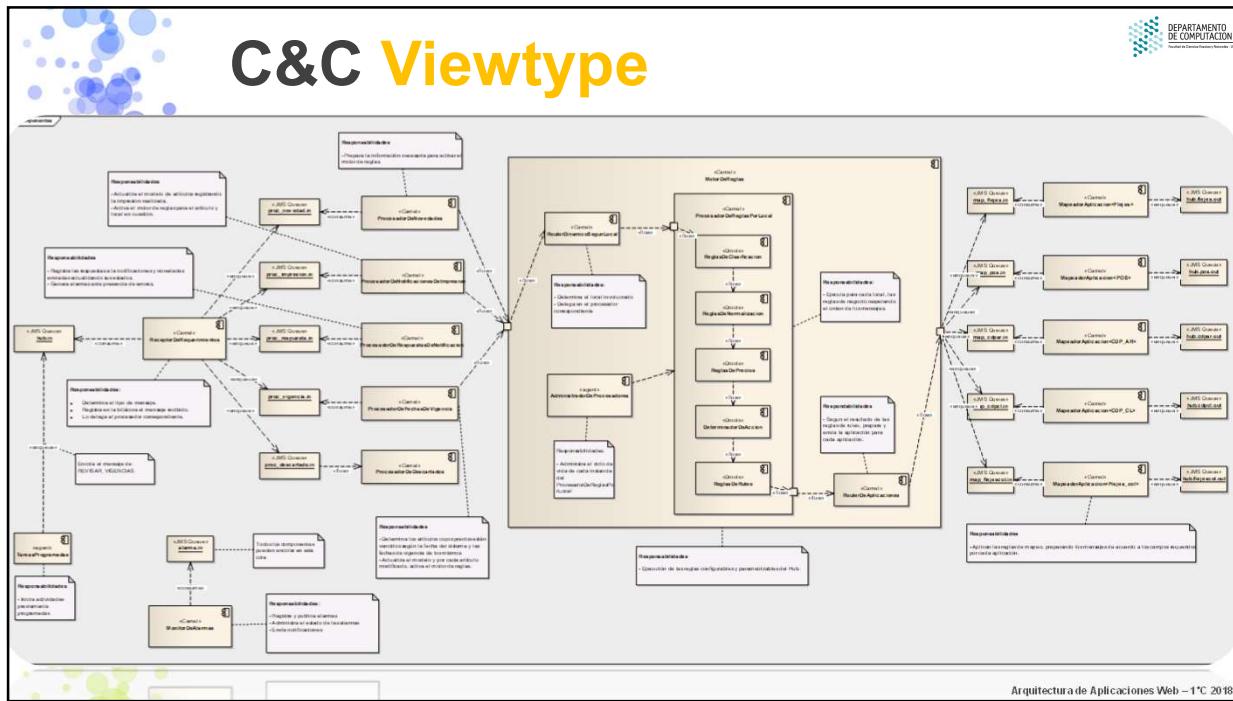
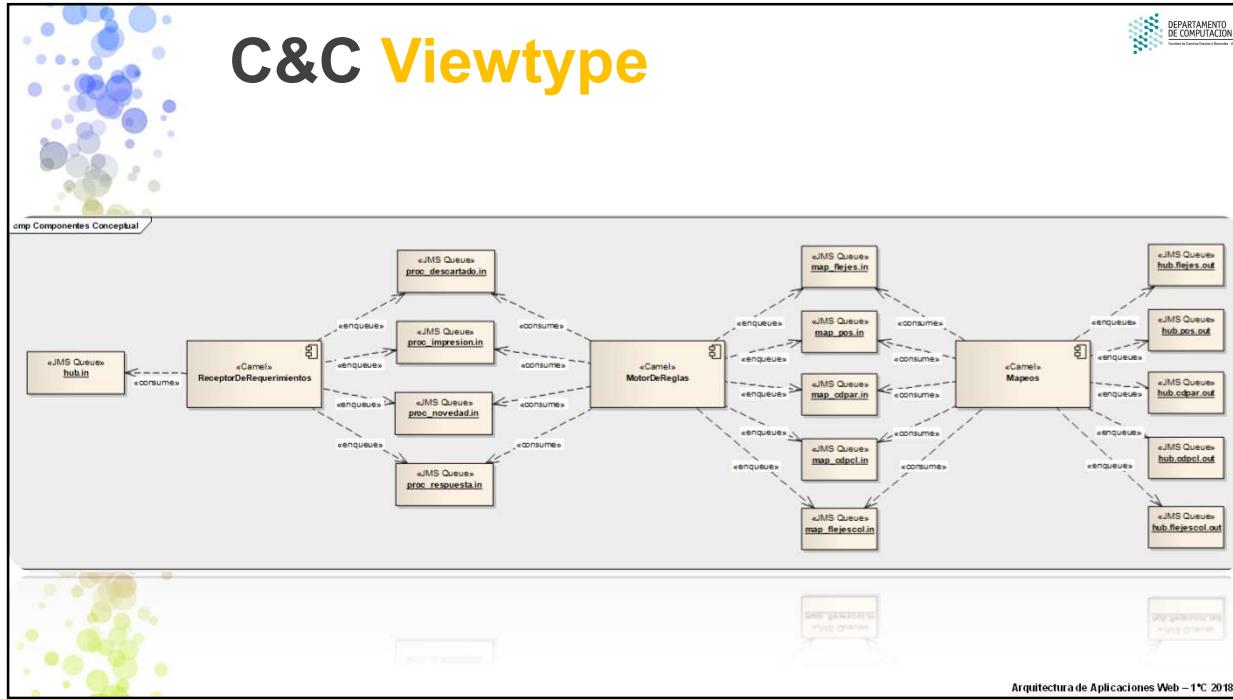
Arquitectura de Aplicaciones Web – 1ºC 2018

C&C Viewtype

- ❖ La relación es **attachment**: Indica qué componentes están vinculados con qué conectores.
- ❖ Formalmente siempre se asocian puertos de componentes con puertos de conectores (llamados roles)



Arquitectura de Aplicaciones Web – 1ºC 2018

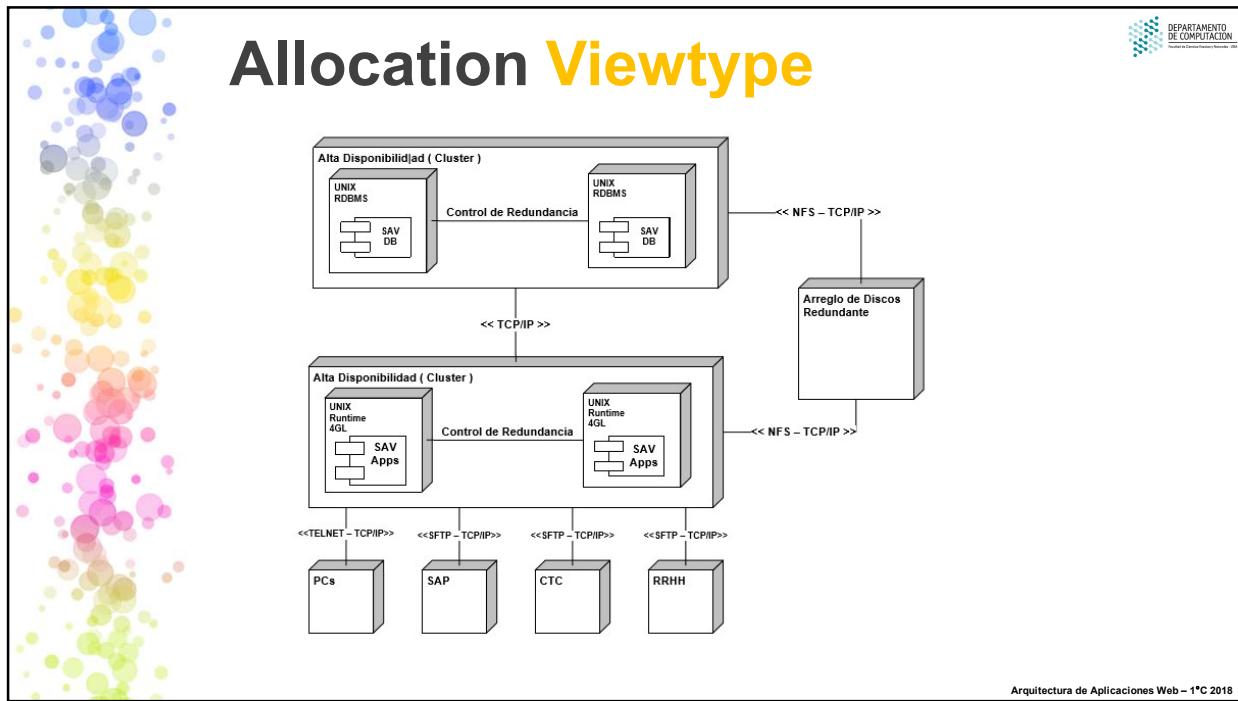


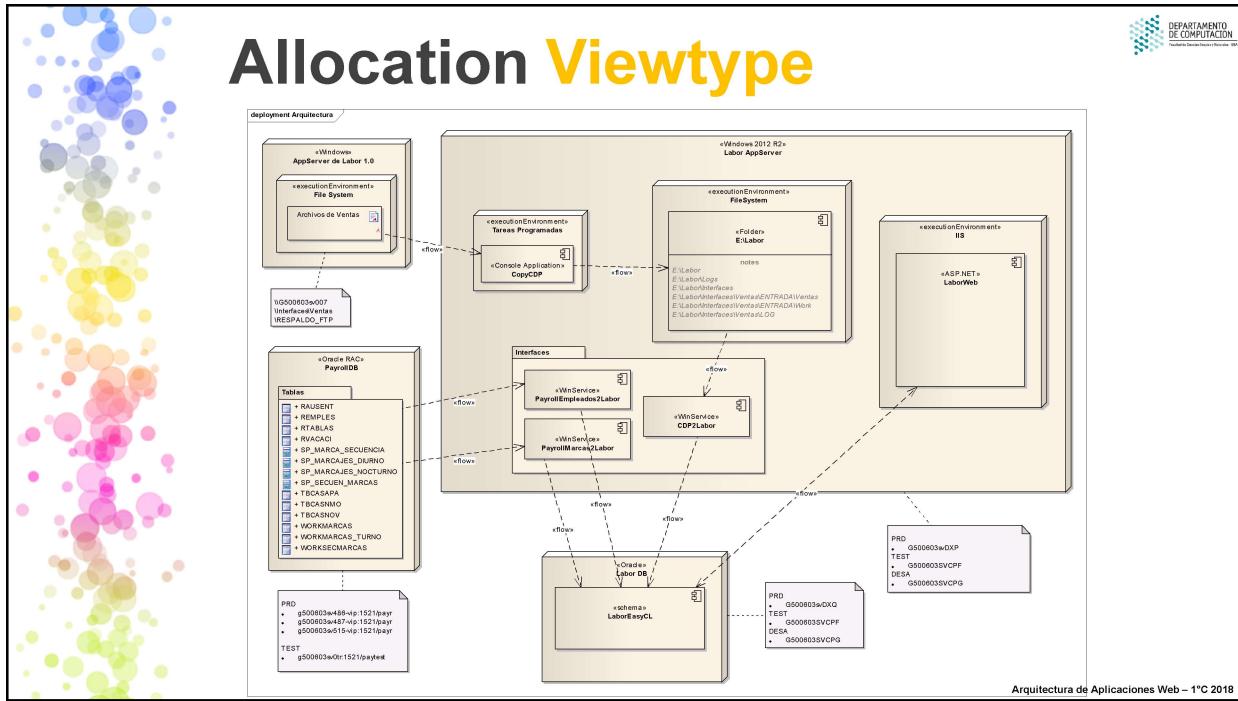
Allocation Viewtype

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

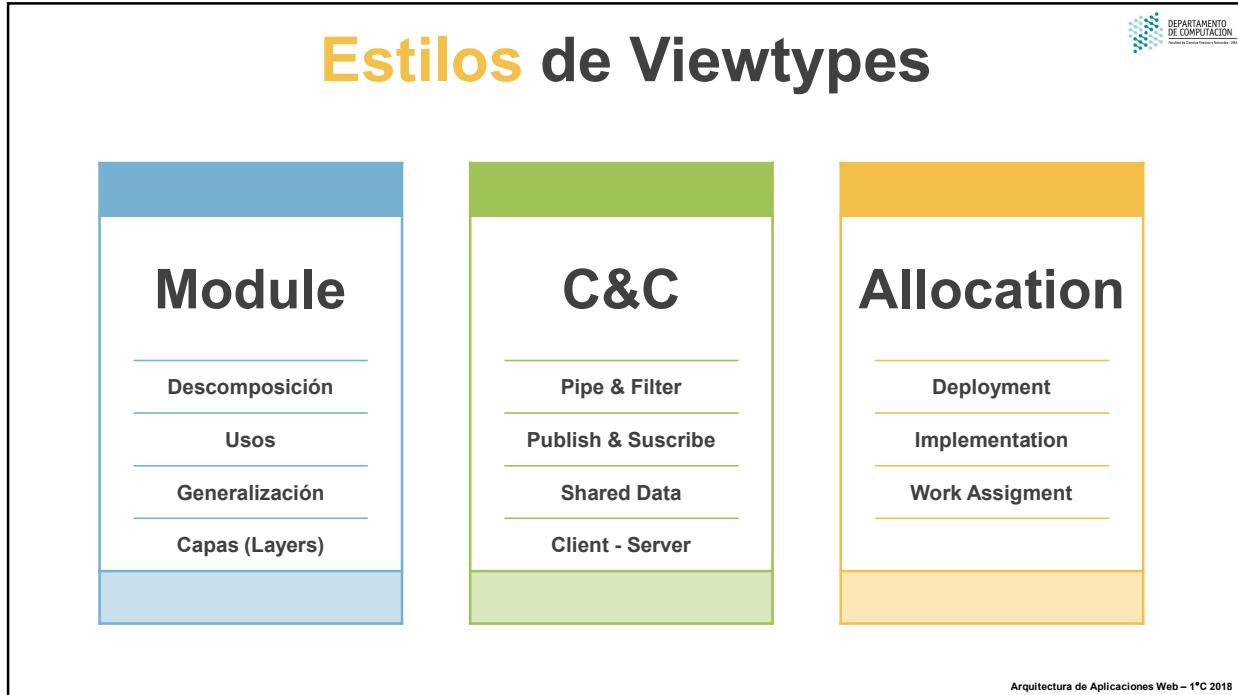
- ❖ **Deployment:** muestra cómo el software se asigna a hardware y elementos de comunicación.
- ❖ **Implementación:** muestra cómo los elementos de software se mapean a estructuras de archivos en repositorios de control de la configuración o entornos de desarrollo.
- ❖ **Asignación de trabajo (“work assignment”):** asigna la responsabilidad del desarrollo y la implementación a equipos de programadores.

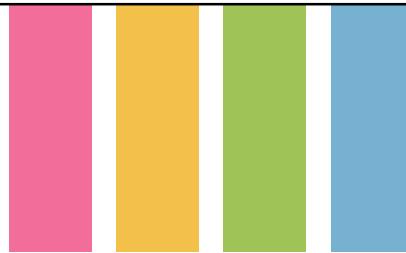
Arquitectura de Aplicaciones Web – 1ºC 2018





Estilos de Viewtypes





Estilos Arquitectónicos C&C

Un estilo arquitectónico determina el vocabulario de componentes y conectores que puede ser usado, así como un conjunto de restricciones de cómo pueden ser combinados.

Un estilo arquitectónico define una familia de sistemas en términos de patrón de organización estructural.

Arquitectura de Aplicaciones Web – 1ºC 2018

Propiedades

- Un vocabulario para los elementos de diseño
 - Tipos de componentes y conectores
 - Por ejemplo: clases, invocaciones, “pipes”, clientes
- Reglas de composición
 - Un estilo tiene restricciones topológicas que determinan cómo se puede hacer la composición de los elementos.
 - Por ejemplo: los elementos de un “layer” se pueden comunicar sólo con los del “layer” inferior.
- Semántica para esos elementos
- Idealmente, criterios para la evaluación de una arquitectura o formas de analizarla; generación de código.
- Importante: un estilo arquitectónico no define la funcionalidad de un sistema. Desde ese punto de vista es algo “abstracto”.

Arquitectura de Aplicaciones Web – 1ºC 2018

Arquitecturas Heterogéneas



- Resultan de la combinación de distintos estilos
- Por ejemplo:
 - Los componentes de un sistema “layered” pueden tener una estructura interna que use otro estilo.
 - Una arquitectura hecha con JEE probablemente resulte en una arquitectura heterogénea que incluya:
 - Layered
 - Repository
 - Independent components
 - Information hiding → Objects

En sistemas medianos / grandes, es más probable que un estilo arquitectónico describa una parte de un sistema que al sistema completo.

Arquitectura de Aplicaciones Web – 1ºC 2018

Módulos vs Componentes



- **Módulos:** entidad en tiempo de diseño.
Enfatiza en encapsulamiento: “information hiding” e interfaces
- **Componentes:** tienen entidad en tiempo de ejecución y de despliegue.

Arquitectura de Aplicaciones Web – 1ºC 2018



Componentes y Conectores

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Colección de módulos de software (**Componentes**) interactuando a través de un paradigma de comunicación bien definida (**conectores**)
- Los componentes son los bloques de construcción para describir una arquitectura.
- No existe aun una notación estándar.

Arquitectura de Aplicaciones Web – 1ºC 2018



Componentes y Conectores

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

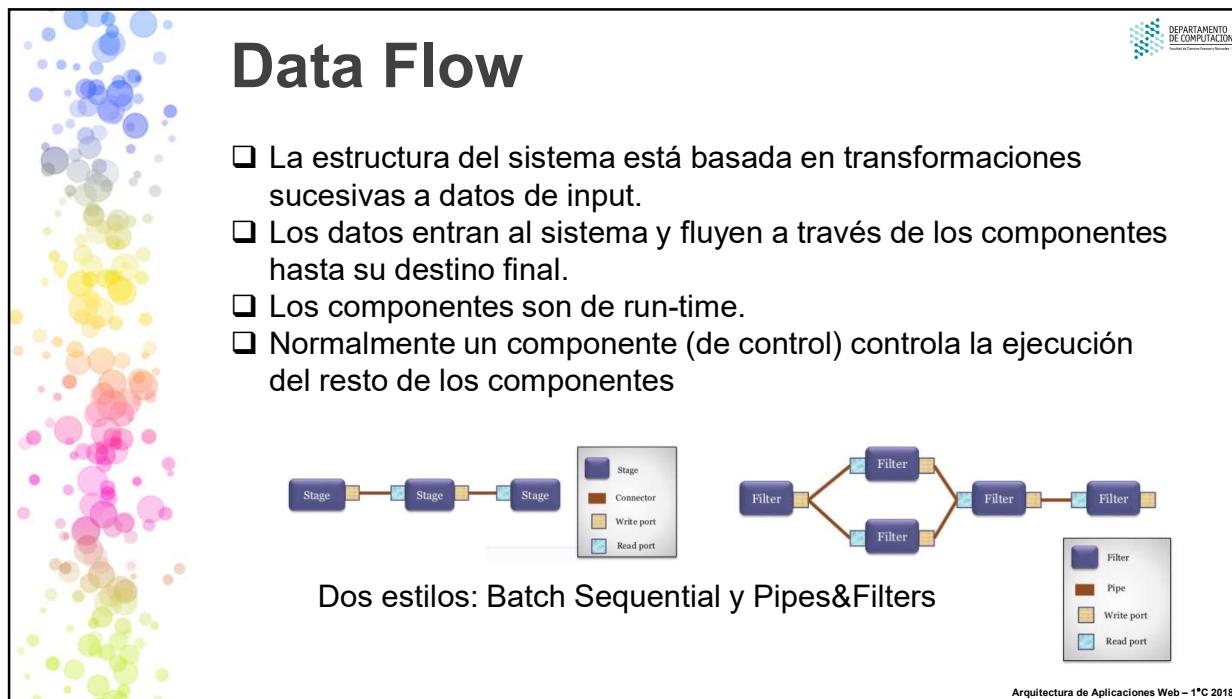
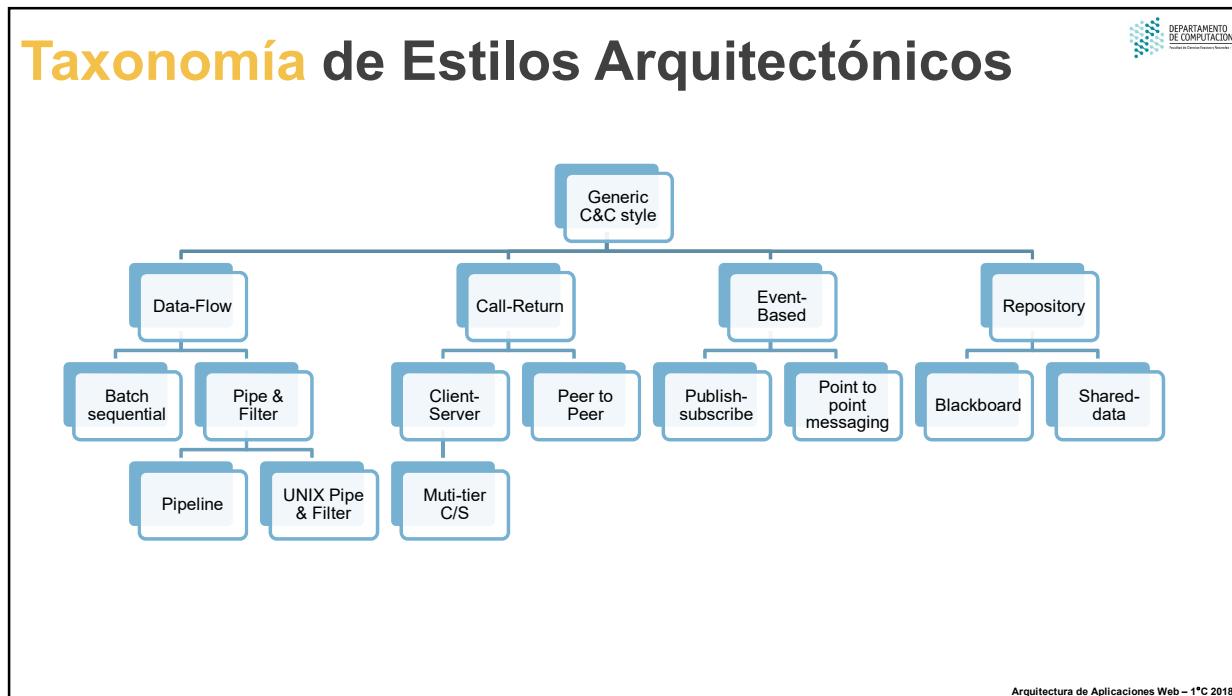
Tipos de Componentes

- **Computacional:** realiza el procesamiento en algún orden.
Ej. función matemática, filtros.
- **Memoria:** mantiene una colección de datos persistentes.
Ej. bases de datos, sistemas de archivos, tablas de símbolos.
- **Manejador:** contiene estado + operaciones asociadas. El estado es mantenido entre invocaciones de operaciones.
Ej. TAD, Servidores.
- **Controlador:** gobierna la secuencia de tiempo de otros eventos.
Ej. módulo de control de alto nivel, scheduler.

Tipos de Conectores

- **Procedure call**
Simple thread de control entre el invocado (called) y el invocador (callee). Ej. RPC.
- **Data flow**
Interacción de procesos a través de flujos de datos. Ej. pipes
- **Implicit invocation**
El proceso se inicia hasta que un evento ocurra. Ej. listas de correo.
- **Message passing**
La interacción se realiza a través de transferencia explícita o de datos discretos. Ej. TCP/IP.
- **Shared data**
El acceso a datos es concurrente, con algún esquema de bloqueo para prevenir los conflictos. Ej. Pizarra, bases de datos compartidas.

Arquitectura de Aplicaciones Web – 1ºC 2018

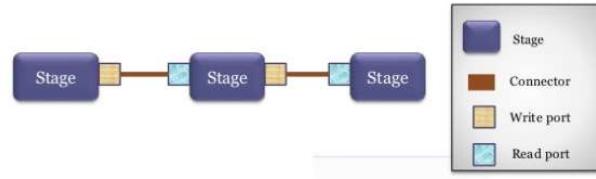


Data Flow - Batch



 DEPARTAMENTO
DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- ❑ Cada paso se ejecuta hasta ser completado, y recién después puede comenzar el siguiente paso.
- ❑ Usado en aplicaciones clásicas de procesamiento de datos.
- ❑ Muchas veces usadas a partir de un “proceso off line”.



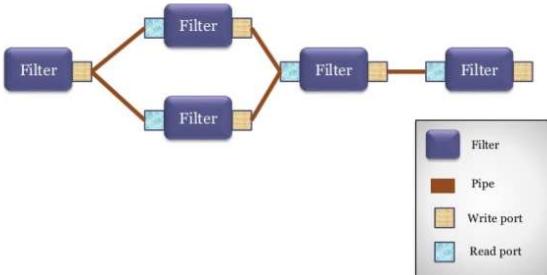
Arquitectura de Aplicaciones Web – 1ºC 2018

Data Flow – Pipes&Filters



 DEPARTAMENTO
DE COMPUTACIÓN

- ❑ Cada componente tiene inputs y outputs. Los componentes leen “streams” de datos de su input y producen streams de datos en sus outputs de forma continua.
- ❑ Filters: ejecutan las transformaciones.
- ❑ Pipes: conectores que pasan streams de un filtro a otro.



Arquitectura de Aplicaciones Web – 1ºC 2018

Call Return

□ Un componente llama o invoca a otro y se queda esperando la respuesta.

```

graph LR
    A[Componente A] -- CALL --> B[Componente B]
    B -- RETURN --> A
  
```

Arquitectura de Aplicaciones Web – 1ºC 2018

Layered o Multi-tier

- Cada nivel provee servicios
 - Oculta en nivel siguiente
 - Provee servicios al nivel anterior
- En muchos casos el “bajar” de nivel implica acercarse al hardware o software de base
- Los niveles van formando “virtual machines”
 - Ventajas: portabilidad, facilidad de cambios, reuso.
 - Desventajas: performance, difícil de encontrar la abstracción correcta, puede implicar salteo de niveles.
- Ejemplos:
 - arquitectura de 3 capas (presentación, reglas de negocio, acceso a datos)

Layer: capa lógica.

Tier: capa física o entorno de ejecución.

```

graph TD
    PL[Presentation Layer] --> BL[Business Logic Layer]
    BL --> DAL[Data Access Layer]
    DAL --> DS[Data Source]
  
```

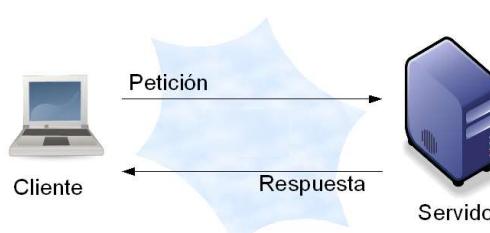
```

graph TD
    PGT[Presentation / GUI Tier] <--> ALT[Application Logic Tier]
    ALT <--> DT[Data Tier]
  
```

Arquitectura de Aplicaciones Web – 1ºC 2018

Client / Server

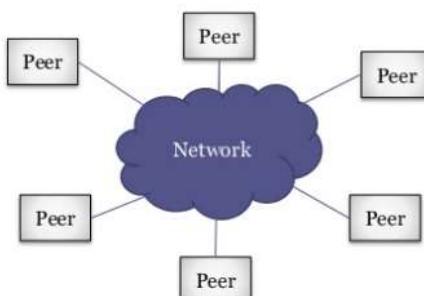
- 2-Tier
- Los componentes son clientes (acceden a servicios) y servidores (proveen servicios)
- Los servidores no conocen la cantidad o identidad de los clientes.
- Los clientes saben la identidad de los servidores.
- Los conectores son protocolos basados en Call/Return



Arquitectura de Aplicaciones Web – 1ºC 2018

Peer to Peer

- Nodos autónomos e iguales (peers) se comunican entre sí a través de la red.



Arquitectura de Aplicaciones Web – 1ºC 2018

Event Based

➤ EDA

- Event Driven Architecture

```
graph LR; A([Event Producer]) --> B[Event Processor]; B --> C([Event Consumer]);
```

The diagram illustrates the flow of events in an Event-Based architecture. It starts with an 'Event Producer' (yellow oval), which sends an event to an 'Event Processor' (blue rectangle). The 'Event Processor' then sends the event to an 'Event Consumer' (pink oval).

Arquitectura de Aplicaciones Web – 1ºC 2018

Publish & Subscribe

➤ Componentes se suscriben a un canal para recibir mensajes de otros componentes.

```
graph TD; subgraph EB [Event Bus]; direction LR; C1[Component] --- P1[Publish Port]; C2[Component] --- P2[Publish Port]; C3[Component] --- P3[Subscribe Port]; C4[Component] --- P4[Subscribe Port]; end;
```

The diagram illustrates the Publish & Subscribe architecture. Components publish messages to an 'Event Bus' (represented by a brown bar). Other components can subscribe to the 'Event Bus' to receive messages. The legend indicates that a blue square represents a 'Subscribe Port' and a yellow square represents a 'Publish Port'.

Arquitectura de Aplicaciones Web – 1ºC 2018

Centradas en datos

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Estructura de datos central (normalmente una base de datos) y componentes que acceden a ella.
- Gran parte de la comunicación está dada por esos datos compartidos.

```
graph TD; SD[Shared Data] --> AM1((Application Module)); SD --> AM2((Application Module)); SD --> AM3((Application Module)); SD --> AM4((Application Module))
```

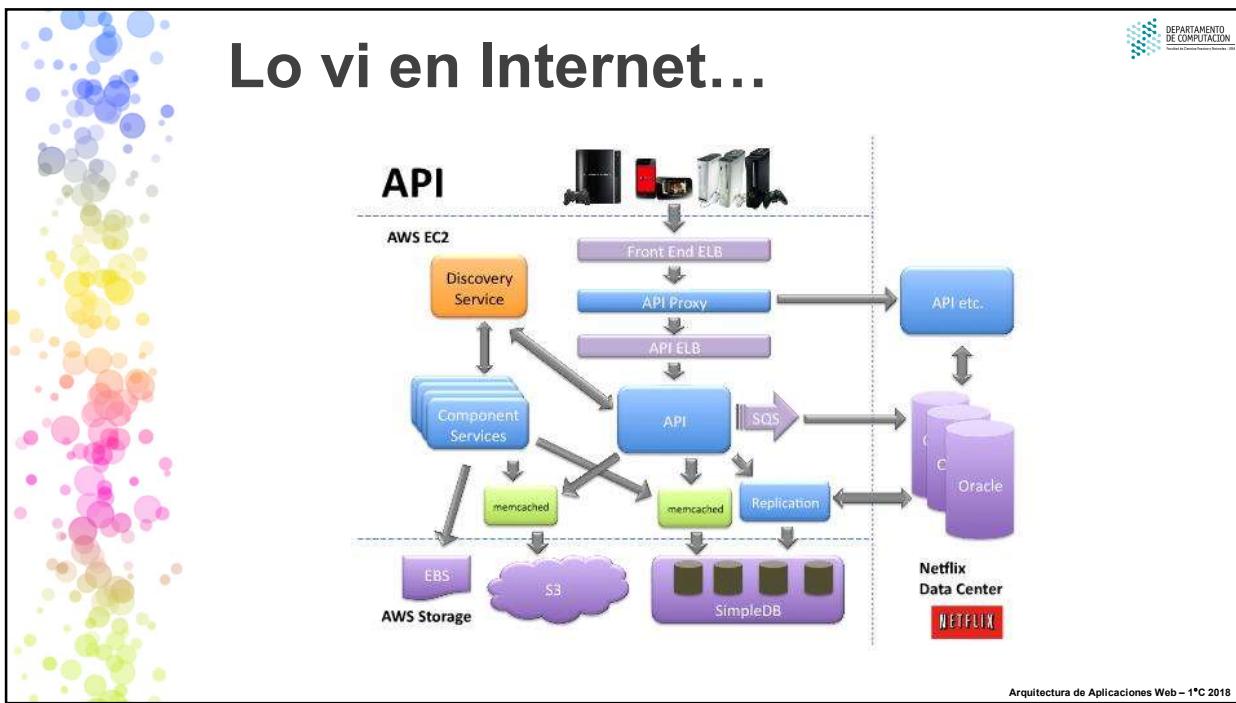
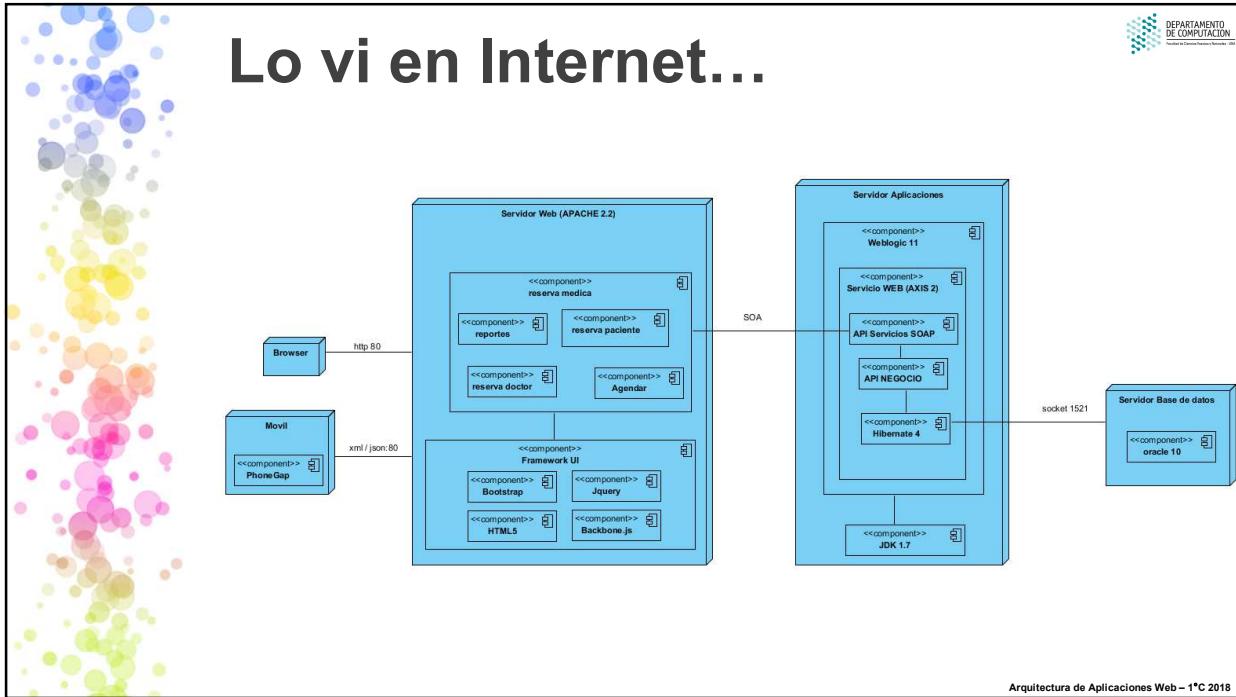
Arquitectura de Aplicaciones Web – 1ºC 2018

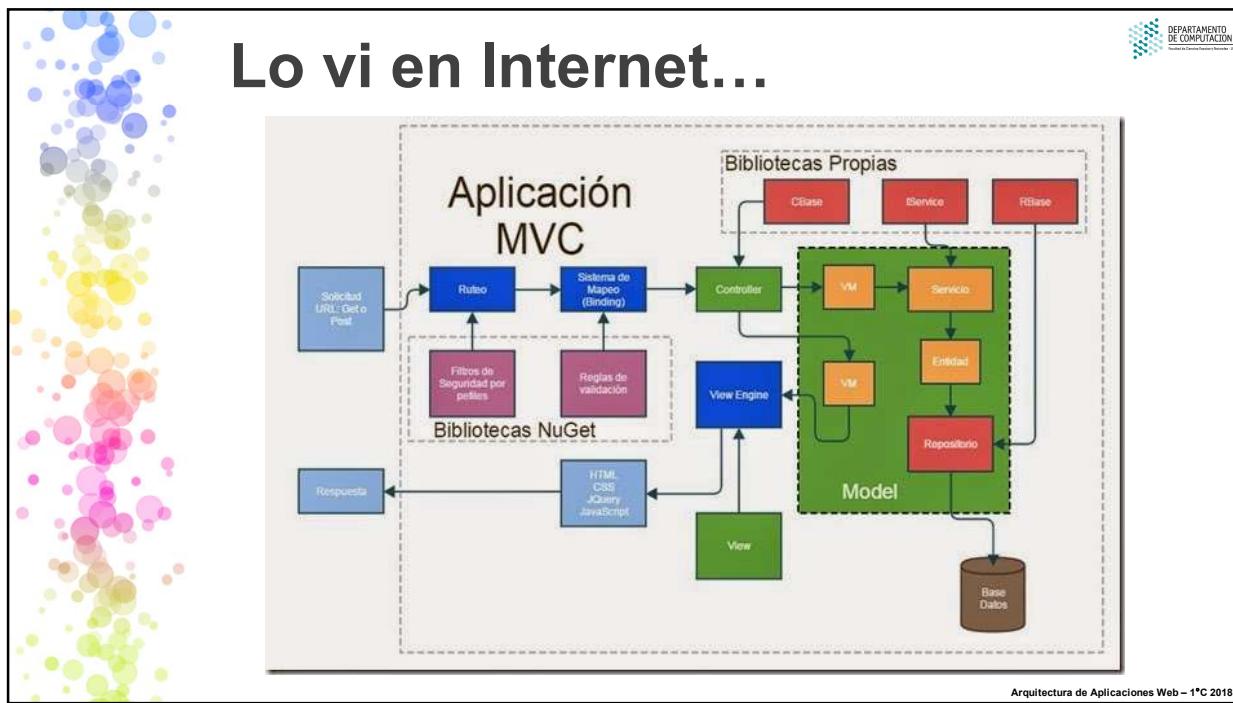
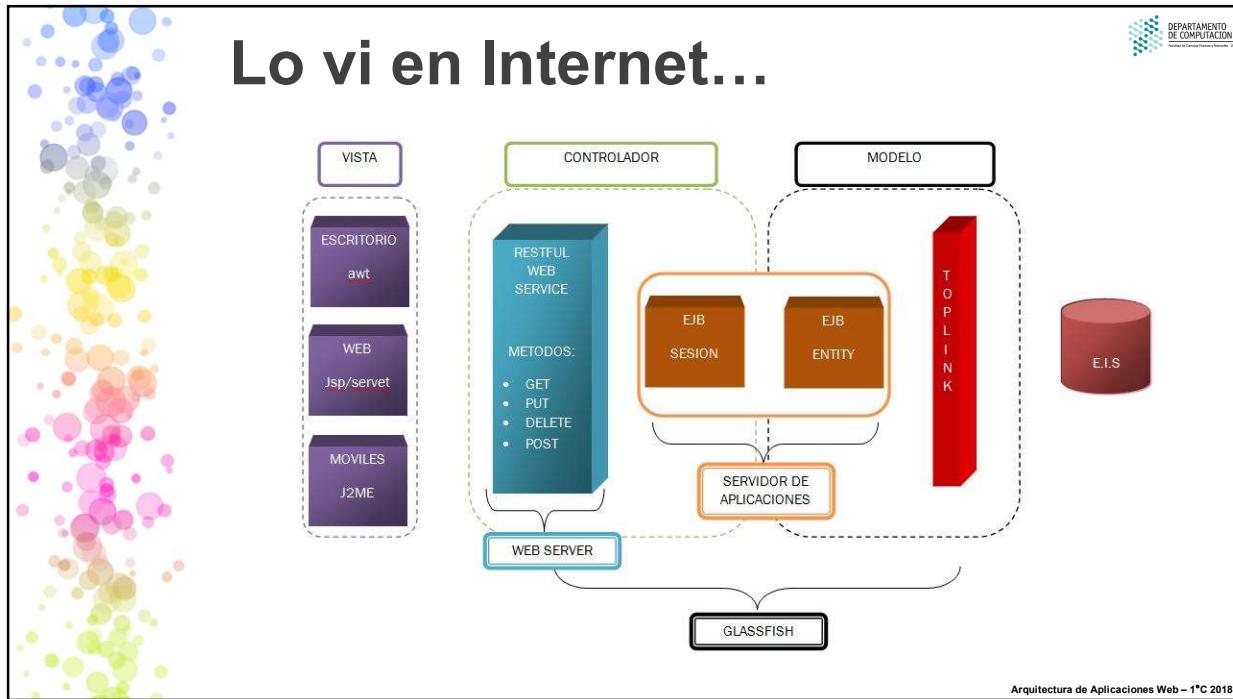
Documentación de Arquitecturas

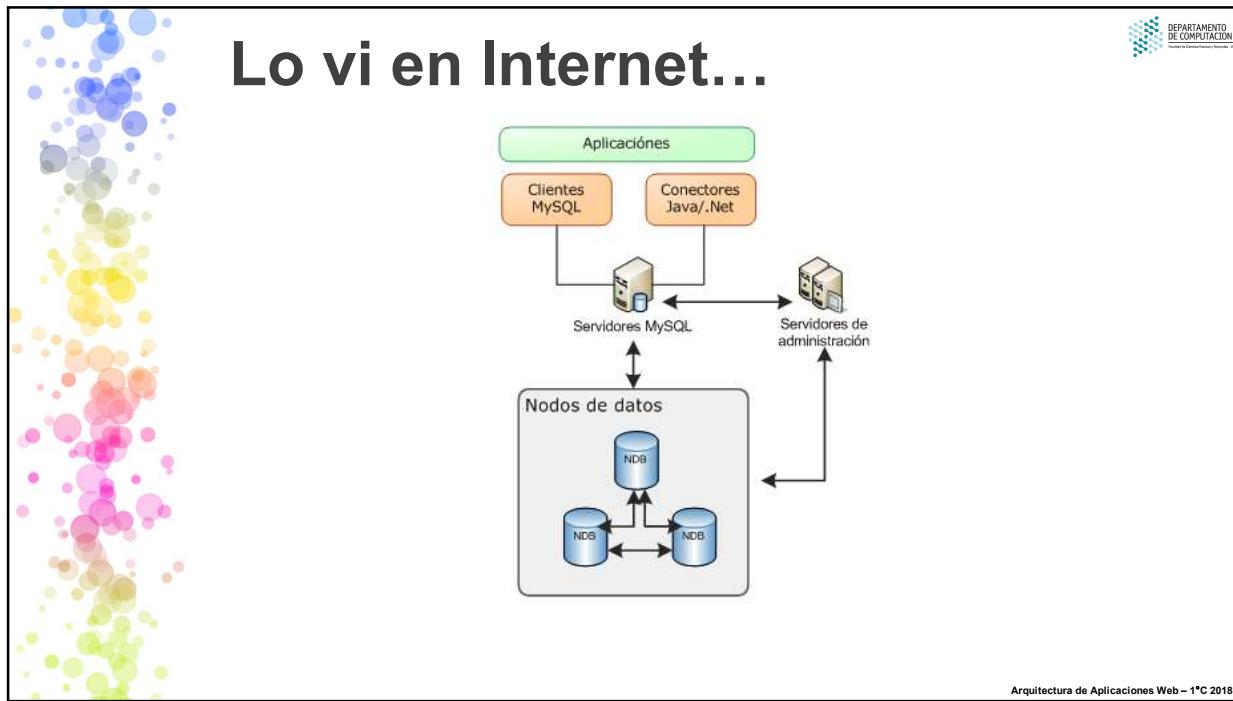
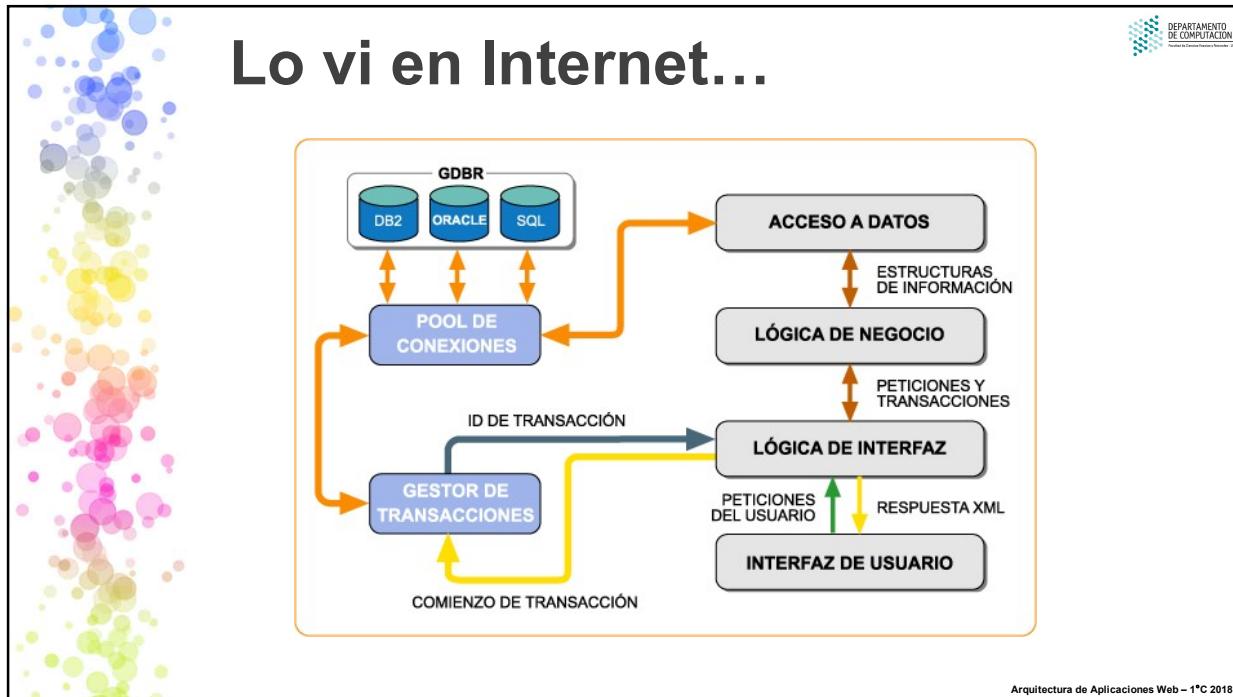
DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

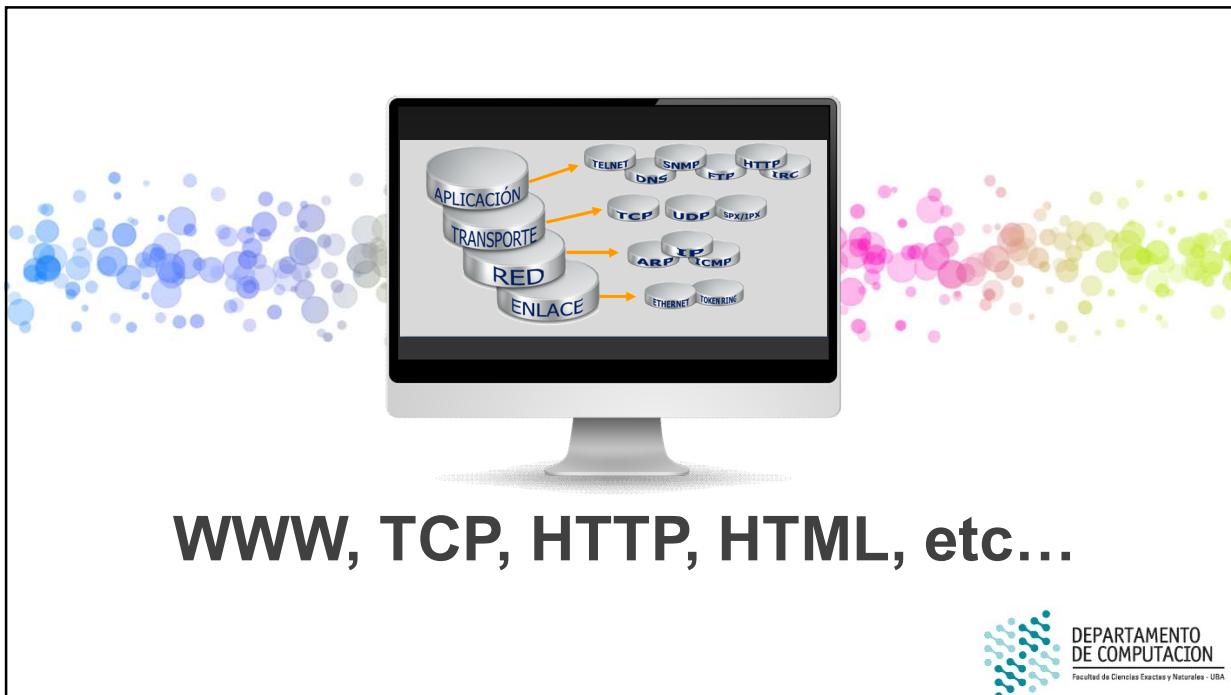
- ❖ Descripción de los requerimientos
 - Contexto del negocio, razón de ser para el producto, dominio
- ❖ Descripción del contexto
 - Sistemas con quienes interactúa, interfaces externas
- ❖ Uso de diagramas de arquitectura
 - Con prosa y descripción de cajas y líneas
- ❖ Consideración de restricciones de implementación
 - En la medida en que impactan la arquitectura
- ❖ Explicación del diseño arquitectónico
 - Como ataca los requerimientos y las restricciones de diseño
 - Alternativa

Arquitectura de Aplicaciones Web – 1ºC 2018









Un poco de historia

- **World Wide Web (WWW)**
 - Inventada por Tim Berners Lee en 1989
 - Mientras trabajaba European Organization for Nuclear Research (CERN)
 - <http://www.w3.org/Consortium/history.html>
 - <http://www.w3.org/History/1989/proposal.html>

Arquitectura de Aplicaciones Web – 1ºC 2018

Terminología básica

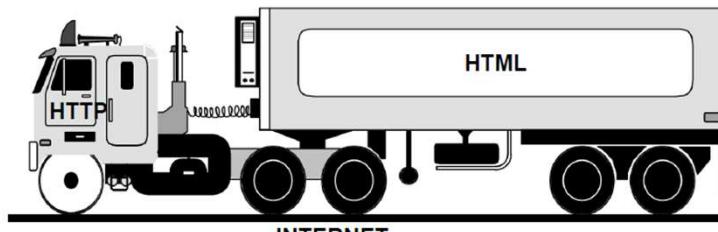
La WWW es una combinación de 4 ideas básicas

- **Hipertexto**
 - Habilidad de navegar desde un documento a otro a través de conexiones → “hiperenlaces”
- **Identificadores de Recursos**
 - Permite encontrar un recurso particular (un documento, imagen) en la red a través de dicho identificador
 - UniformResourceIdentifier(URI)
 - UniformResourceLocator(URL)
- **Modelo cliente servidor**
 - Un cliente software demanda servicios o recursos a un servidor software
- **Un lenguaje de marcado**
 - Además de texto incluyen conjuntos de caracteres especiales que indican al navegador como presentar dicho texto → HTML

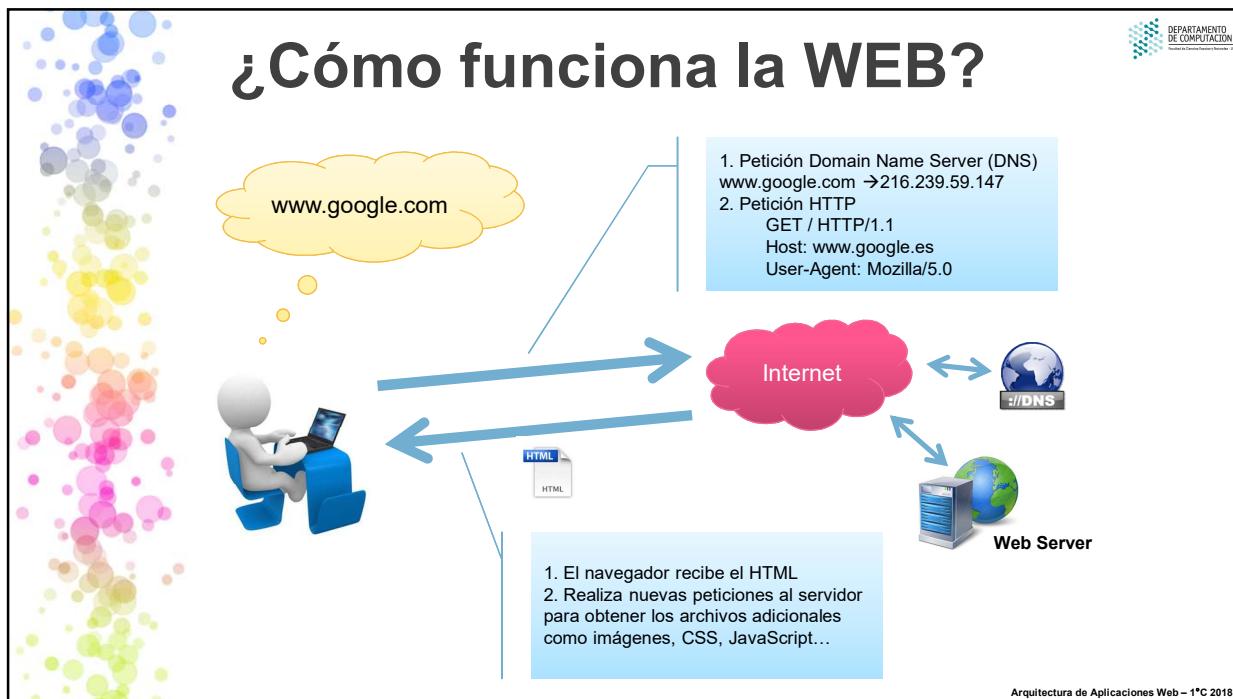
Arquitectura de Aplicaciones Web – 1ºC 2018

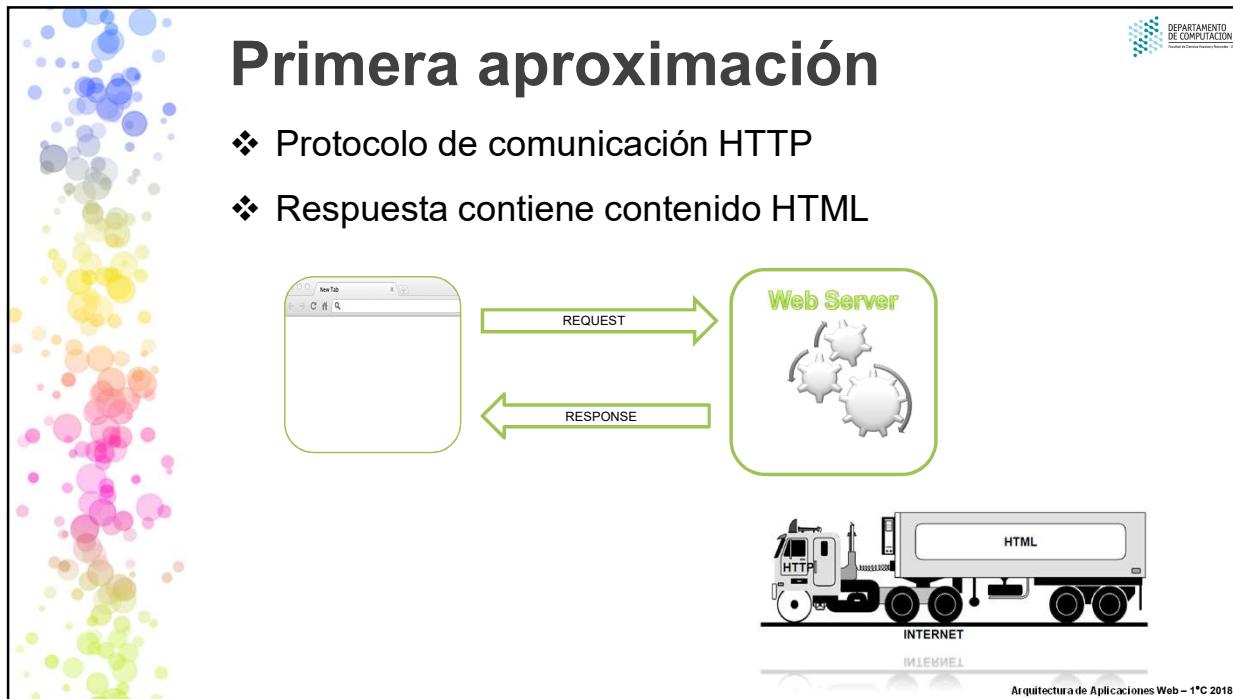
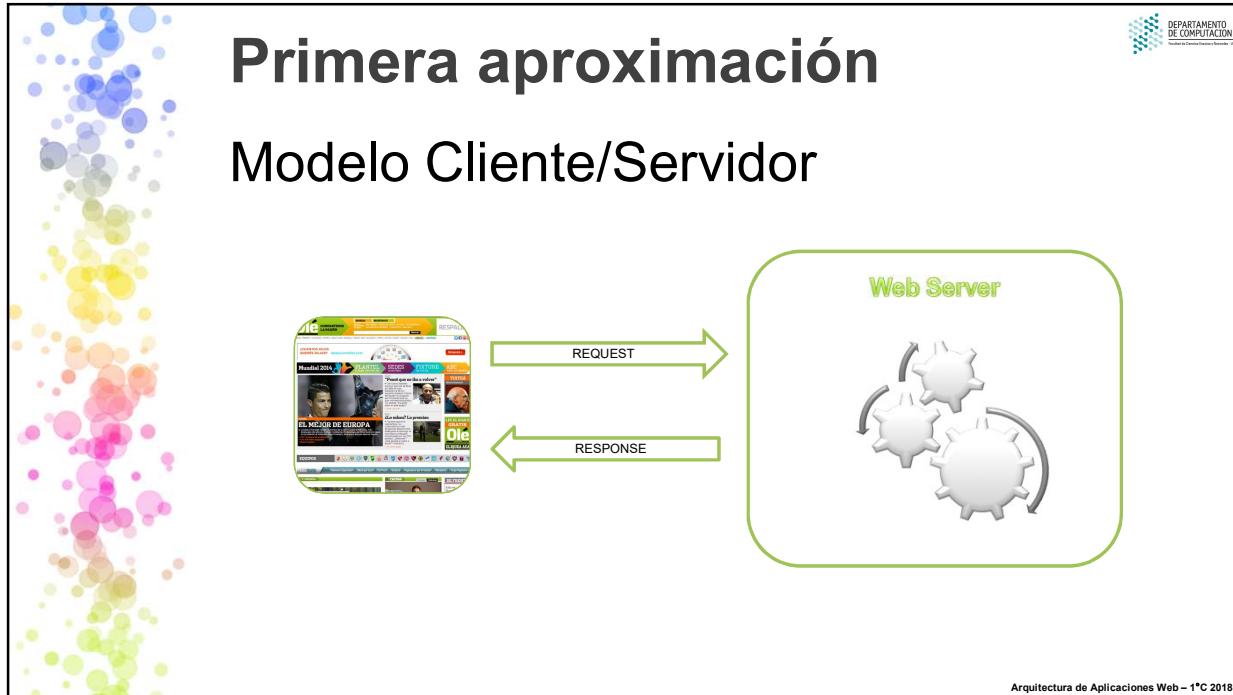
Terminología básica

La WWW no es Internet, es un servicio que está montado sobre Internet
Internet (la red) está formado por el conjunto de ordenadores que están interconectados entre sí.



Arquitectura de Aplicaciones Web – 1ºC 2018





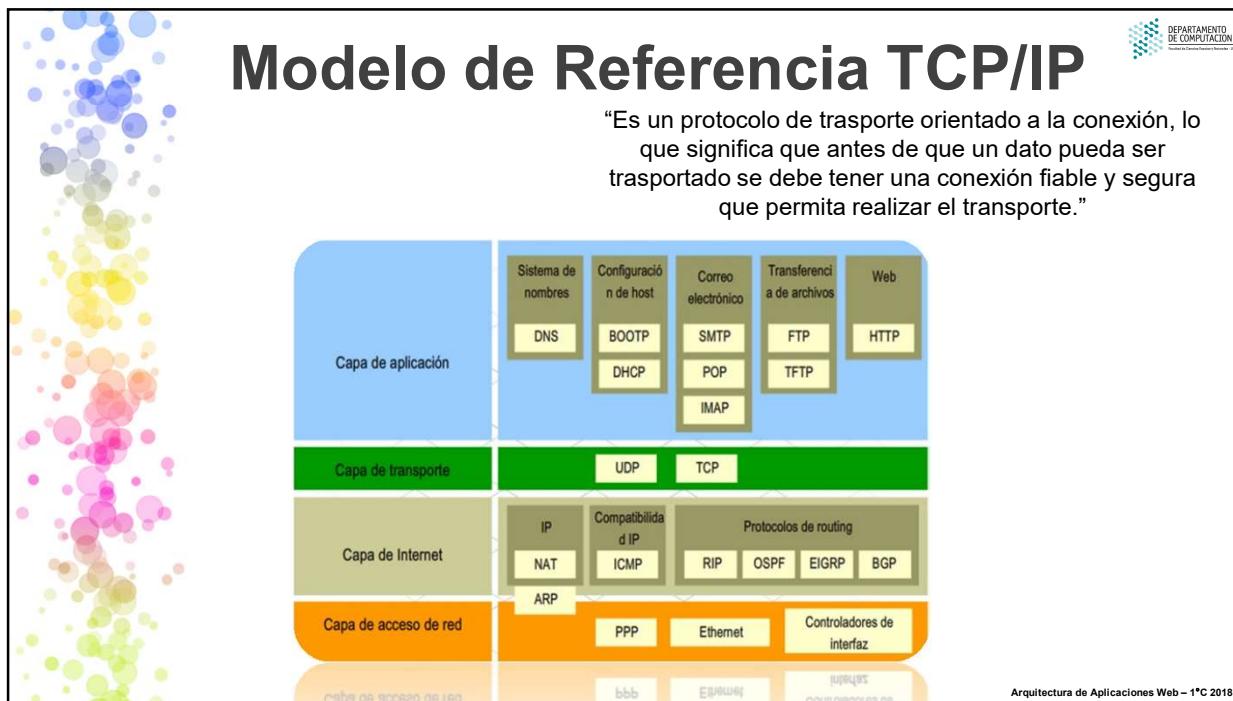
HTTP

DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- ❑ Hypertext Transfer Protocol o HTTP (en español *protocolo de transferencia de hipertexto*) es el protocolo usado en cada transacción de la World Wide Web.
- ❑ HTTP define la *sintaxis* y la *semántica* que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.
- ❑ Es un *protocolo orientado a transacciones* y sigue el esquema petición-respuesta entre un cliente y un servidor.
 - Al cliente que efectúa la petición (un navegador web) se lo conoce como "user agent" (agente del usuario).
 - A la información transmitida se la llama recurso y se la identifica mediante un localizador uniforme de recursos (URL)



Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP

- ❖ TCP port 80 (443 secure)
- ❖ Estándares
 - [RFC 1945 \(HTTP/1.0, 1996\)](#)
 - [RFC 2616 \(HTTP/1.1, 1999\)](#)
 - [RFC 7540 \(HTTP/2.0, 2015\)](#)

HTTP/1.1

HTTP/2

HTTP/1.1 Baseline

HTTP/2 Multiplexing

Time

Client Renders Page

Connection Remains Open

Connection Closed

Arquitectura de Aplicaciones Web – 1ºC 2018

HTTP/2

- 1. One TCP connection
- 2. Request → Stream
 - Streams are multiplexed
 - Streams are prioritized
- 3. Binary framing layer
 - Prioritization
 - Flow control
 - Server push
- 4. Header compression (HPACK)

HTTP/1.1

POST /upload HTTP/1.1
Host: www.example.org
Content-Type: application/json
Content-Length: 15
{"msg": "hello"}

HTTP/2

HEADERS frame

DATA frame

Arquitectura de Aplicaciones Web – 1ºC 2018

URI = URL + URN



• Uniform Resource Identifier

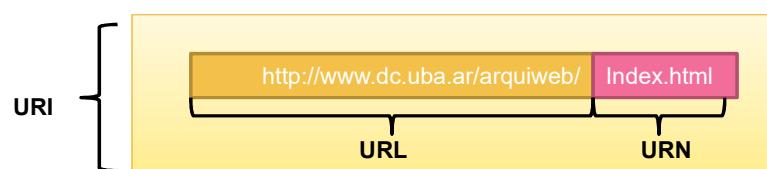
- Es una cadena de caracteres que identifica los recursos de una red de forma única.

• Uniform Resource Name

- Es una cadena de caracteres que identifican recursos en la web, pero no indican exactamente dónde se encuentra ese objeto.

• Uniform Resource Locator

- Es una secuencia de caracteres que determina cómo encontrar el recurso.



Arquitectura de Aplicaciones Web – 1ºC 2018

URIs



- URI
 - *Protocolo (http, ftp, news)*
 - *Host name (name.domain name)*
 - *Puerto (usualmente 80)*
 - *Ruta al recurso*
 - *Nombre del Recurso*
 - http://www.myplace.com/www/index.html
 - http://www.myplace.com:80/cgi-bin/t.exe

Mensajes HTTP



DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- Pueden ser
 - *Solicitudes (Request)*
 - *Respuestas (Response)*
- Tanto las solicitudes como las respuestas utilizan el formato genérico de e-mails (RFC-822)
- Ambos tipos de mensajes consisten de
 - *Una línea inicial*
 - *Cero o más encabezados (headers)*
 - *Una línea en blanco*
 - *Un cuerpo del mensaje (opcional, ej. archivo, datos de una consulta).*

Arquitectura de Aplicaciones Web – 1ºC 2018

HTTP – Request & Response



DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

- Poseen las siguientes partes:
 - *Línea Inicial:*
 - Determina la naturaleza del mensaje.
 - *Encabezados HTTP (Headers):*
 - Por ejemplo: *Accept-Language:* es
 - *Línea vacía*
 - *Cuerpo del mensaje (opcional)*

Requests

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; ... ) AppleWebKit/537.36
Accept: text/html,application/xhtml+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=12656974
Content-Length: 345
```

HTTP headers

empty line

body

Responses

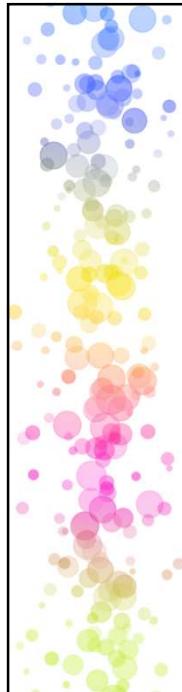
```
HTTP/1.1 403 Forbidden
Server: Apache
Content-Type: text/html; charset=iso-8859-1
Date: Wed, 10 Aug 2016 09:23:25 GMT
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
Age: 3464
Date: Wed, 10 Aug 2016 09:46:25 GMT
X-Cache-Info: caching
Content-Length: 220
```

HTTP headers

empty line

body

Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP – Request & Response

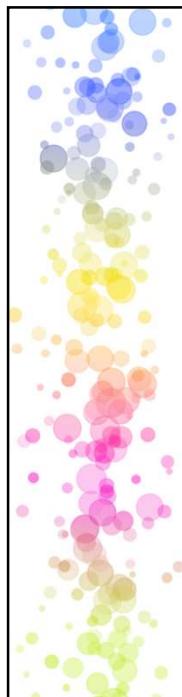
**DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA**

- Línea Inicial
 - **Solicitudes**
 - Tiene tres partes separadas entre sí por un espacio.
 - *El método (GET, PUT, POST, OPTIONS, TRACE, DELETE,...)*.
 - *El identificador del recurso (URI).*
 - *La versión del protocolo HTTP en uso.*

Ejemplo: GET /images/logo.png HTTP/1.1
 - **Respuestas**
 - Tiene tres partes separadas entre sí por un espacio.
 - *Versión de HTTP*
 - *Código de estado*
 - *Frase explicativa (legible por humanos)*

Ejemplo: HTTP/1.0 200 OK HTTP/1.0 404 Not Found

Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP – Métodos de petición

**DEPARTAMENTO DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA**

Método	Significado
GET	Devuelve el recurso identificado en la URL pedida.
HEAD	Funciona como el GET, pero sin que el servidor devuelva el cuerpo del mensaje. Es decir, sólo se devuelve la información de cabecera.
POST	Indica al servidor que se prepare para recibir información del cliente. Suele usarse para enviar información desde formularios.
PUT	Envía el recurso identificado en la URL desde el cliente hacia el servidor.
OPTIONS	Pide información sobre las características de comunicación proporcionadas por el servidor. Le permite al cliente negociar los parámetros de comunicación.
TRACE	Inicia un ciclo de mensajes de petición. Se usa para depuración y permite al cliente ver lo que el servidor recibe en el otro lado.
DELETE	Solicita al servidor que borre el recurso identificado con el URL.
CONNECT	Este método se reserva para uso con proxys. Permitirá que un proxy pueda dinámicamente convertirse en un túnel. Por ejemplo para comunicaciones con SSL.

Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP – Códigos de estados

• El código de estado es un entero de 3 dígitos

Rango	Descripción
1XX	Informativos
2XX	Éxito
3XX	Redirección
4XX	Error de cliente
5XX	Error de servidor

Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP – Códigos de estados

• Los más comunes

Código	Descripción
200	OK Solicitud exitosa. La respuesta se envía en el cuerpo
404	Not Found El recurso no existe.
303	See Other El recurso se ha movido a otra URL (Dada en el header Location)
500	Server Error Error no esperado en el servidor.

Arquitectura de Aplicaciones Web – 1ºC 2018



HTTP – Códigos de estados



Arquitectura de Aplicaciones Web – 1ºC 2018

- 1xx: Mensaje informativo
- 2xx: Éxito
 - 200 OK
 - 201 Created
 - 202 Accepted
 - 204 No Content
- 3xx: Redirección
 - 300 Multiple Choice
 - 301 Moved Permanently
 - 302 Found
 - 304 Not Modified
- 4xx: Error del cliente
 - 400 Bad Request
 - 401 Unauthorized
 - 403 Forbidden
 - 404 Not Found
- 5xx: Error del servidor
 - 500 Internal Server Error
 - 501 Not Implemented
 - 502 Bad Gateway
 - 503 Service Unavailable



Http - Encabezados



Arquitectura de Aplicaciones Web – 1ºC 2018

- Formato de los encabezados
 - Nombre : Valor
- Clasificación
 - Genéricos: cliente y servidor
 - Exclusivos de la petición (información del cliente)
 - Exclusivos de la respuesta (información del Servidor)
 - Entidad del cuerpo del mensaje
- HTTP 1.0 define 16 headers (ninguno es obligatorio).
- HTTP 1.1 define 46 headers (solo Host: es obligatorio).

Encabezados genéricos

Encabezado	Significado
Cache-Control	Permite especificar distintas directivas para controlar la caché, tanto del cliente como de servidores proxy.
Connection	Especifica opciones de la conexión de red.
Date	Envía una fecha en la representación estándar definida en el protocolo.
Pragma	Transporta información no HTTP a un receptor que sea capaz de entenderla.
Trailer	Indica que ciertas cabeceras HTTP pueden encontrarse en el final de un mensaje con múltiples partes (multipart).
Upgrade	Información sobre protocolos adicionales soportados por el cliente.
Via	Añadidos al mensaje de proxys o gateways para indicar que pasó por ellos.
Warning	Información adicional sobre un estado o transformación de un documento que podría no estar reflejado en el cuerpo del mismo. Por ejemplo, transformaciones que se hacen en servidores caché.

Manejan información que puede ser utilizada tanto por clientes como por servidores, ya que se aplican a una sesión completa de comunicación

Encabezados de Solicitud

Encabezado	Significado
Accept	Listado de tipos MIME que el cliente soporta. Hay otros encabezados relacionados como Accept-Charset , Accept-Encoding , Accept-Language .
Authorization	Indica las credenciales de acceso a un recurso que presenta el usuario.
Expect	Indica que comportamiento del servidor necesita el cliente.
From	Dirección de correo que controla el cliente (navegador).
Host	Nombre o IP del host desde donde se conecta el cliente.
If-Match	Un cliente que tiene recursos en cache puede verificar si están actualizados incluyendo este encabezado. Hay otros encabezados que también tienen que ver con la caché. If-Modified-Since , If-None-Match , If-Range , If-Unmodified-Since .
Max-Forwards	Cuántas veces la petición del cliente puede ser reenviada por proxys.
Proxy-Authorization	Indica las credenciales de acceso a un proxy que presenta el usuario.
Range	Indica que porción de recurso (rango de bytes) recuperar.



Encabezados de Solicitud

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Encabezado	Significado
Referer	Es el URI del recurso desde donde la petición se ha realizado (generalmente por provenir de un enlace HTML).
TE	Que codificaciones de transferencia está dispuesto a recibir el cliente.
User-Agent	Información sobre el agente de usuario (generalmente navegador) que origina la petición.

Los utiliza el cliente para enviar (en sus peticiones de servicio) información adicional al servidor

Arquitectura de Aplicaciones Web – 1ºC 2018



Encabezados de Respuesta

DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Encabezado	Significado
Accept-Ranges	Indica las unidades en las que el servidor acepta peticiones de rangos.
Age	Tiempo estimado por el servidor para cumplir la petición.
Etag	Valor actual de la etiqueta de entidad solicitada.
Location	Contiene un URI al que el cliente debe ser redireccionado.
Proxy-Authenticate	Indica el esquema de autenticación que acepta un servidor proxy.
Retry-After	Cuanto tiempo se espera que el servicio no esté disponible.
Server	Información del software servidor.
Vary	Indica que un recurso tiene múltiples fuentes que pueden variar de acuerdo a la lista de encabezados de petición.
WWW-Authenticate	Indica que el recurso solicitado necesita de credenciales de autorización.

Los utiliza el servidor para enviar información adicional al cliente.

Arquitectura de Aplicaciones Web – 1ºC 2018

Encabezados de Entidad



Encabezado	Significado
Allow	Informa al cliente de los métodos válidos asociados con el recurso.
Content-Type	Indica el tipo MIME de los contenidos. Hay otros encabezados muy relacionados como Content-Language , Content-Length , Content-Location , Content-MD5 , Content-Range o Content-Encoding .
Expires	Indica la fecha y hora en la que el recurso se considerará obsoleto.
Last-Modified	Indica la fecha y hora en la que el recurso original fue modificado por última vez.

Contienen información relacionada directamente con el recurso que se le va a proporcionar al cliente.

Arquitectura de Aplicaciones Web – 1ºC 2018

Custom Header



- ❑ Son headers que no pertenecen al estándar.
- ❑ Tradicionalmente, se los denominaba con el prefijo X-
- ❑ El uso del prefijo x- quedó *deprecated* en 2012.

Arquitectura de Aplicaciones Web – 1ºC 2018

HTTP - Request

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0;
Windows NT; DigExt)
Host: www.yahoo.com
Connection: Keep-Alive
Cookie: B=2td79o0sj1f5r&b=2
```



Arquitectura de Aplicaciones Web – 1ºC 2018

HTTP - Cuerpo

- Luego de las líneas de encabezado un mensaje HTTP puede contener un cuerpo (body).
- En las respuestas el cuerpo es la sección en donde se envía el recurso solicitado.
- En las solicitudes el cuerpo se utiliza para subir datos que ingresó el usuario o para transferir archivos hacia el servidor.
- Las líneas de encabezado más comunes que definen el cuerpo son:
 - Content-Type: (Da el tipo MIME de los datos del cuerpo, ejemplo: text/html image/gif).
 - Content-Length: (Especifica el número de bytes en el cuerpo).



Arquitectura de Aplicaciones Web – 1ºC 2018

HTML

- El *HiperText Markup Language* (HTML) es el lenguaje utilizado para diseñar páginas web.
- Es el lenguaje comprendido por los *browsers*.
- Una página HTML consiste en texto delimitado por *tags de marcado* que describen la apariencia, el formato y la ubicación del contenido.
- Puede ser editado utilizando cualquier procesador de texto.

HTML

```
<HTML>
  <HEAD>
    <TITLE> Home Page</TITLE>
  </HEAD>
  <BODY>
    <H1 ALIGN=CENTER>
      Welcome to XYZ's Home Page
    </H1>
    Hi<BR>there<BR>Joe<BR>
  </BODY>
</HTML>
```



HTML



Welcome to XYZ's Home Page
Hi,
there
Joe

Más adelante retomaremos este tema...



Arquitectura de Aplicaciones Web – 1ºC 2018



Session & Cookies



- HTTP es un protocolo *sin estado*
- Existen diversos *mecanismos* útiles para recordar el estado:
 - HTTP Headers
 - FAT URLs
 - Session
 - Cookies
 - Entre otros...

Arquitectura de Aplicaciones Web – 1ºC 2018

Session

- ❑ Una sesión (session) es un período de tiempo dentro del cual, todas las actividades realizadas por un cliente (user-agent) se consideran asociadas a él.
- ❑ Session tracking: es un mecanismo que permite almacenar en el servidor, durante el período de tiempo que dure la sesión, cierta información a través de las siguientes solicitudes realizadas por el usuario.

Cookies

- Pequeños archivos de texto almacenados en los browsers.
- Se submittean en cada request.
- Permiten contar con trazabilidad intersetión.

Session & Cookies

Session	Cookie
Server Side	Client Side
Al cerrar el browser se da por terminada la session	Soporta múltiples sesiones.
Permite almacenar objetos	Permite almacenar Strings

¿Qué es Javascript?

- Es un lenguaje *Script*
- Extiende las capacidades de las páginas Web
- El código está integrado en el HTML
- Se interpreta en el ordenador que recibe el HTML, no se compila
- Ejecución dinámica
 - Los programas y funciones no se chequean hasta que se ejecutan
- Tiene programación orientada a objetos
- Trabaja con los elementos del HTML
- No se declaran los tipos de variables

Generalidades de *Javascript*



- Modelo orientado al WWW
- Elementos de una página HTML pueden causar un evento que ejecutará una acción
- Esa acción se ejecutará a través de una serie de sentencias *JavaScript*
- Comandos de *JavaScript*:
 - Variables
 - Expresiones
 - Estructuras de control
 - Funciones (bloques de sentencias)
 - Clases, objetos y arrays (agrupaciones de datos)

Arquitectura de Aplicaciones Web – 1ºC 2018

¿Qué se puede hacer con *Javascript*?



- Chequear Formularios
 - Comprobar que se han llenado correctamente antes de enviarlos y que el servidor de error
- Realizar cálculos simples
- Hacer interactiva una página web
- Juegos...
- Muchas cosas!

Arquitectura de Aplicaciones Web – 1ºC 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

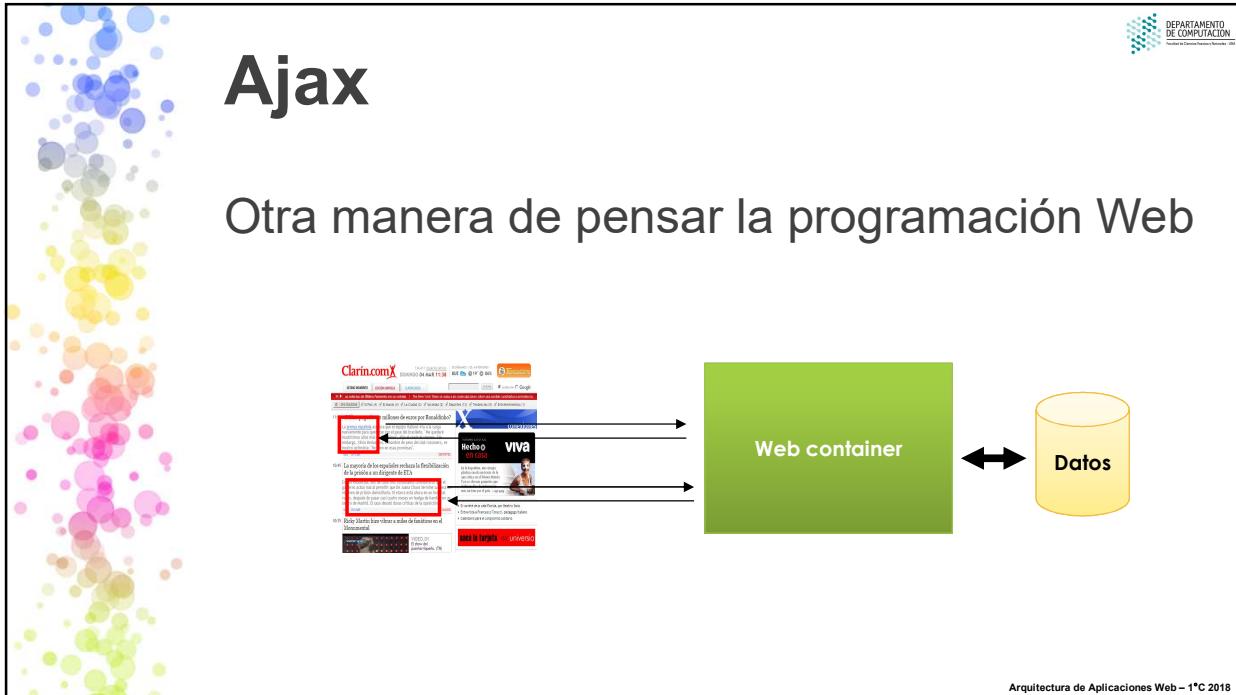
Arquitectura de Aplicaciones Web – 1ºC 2018

Ajax

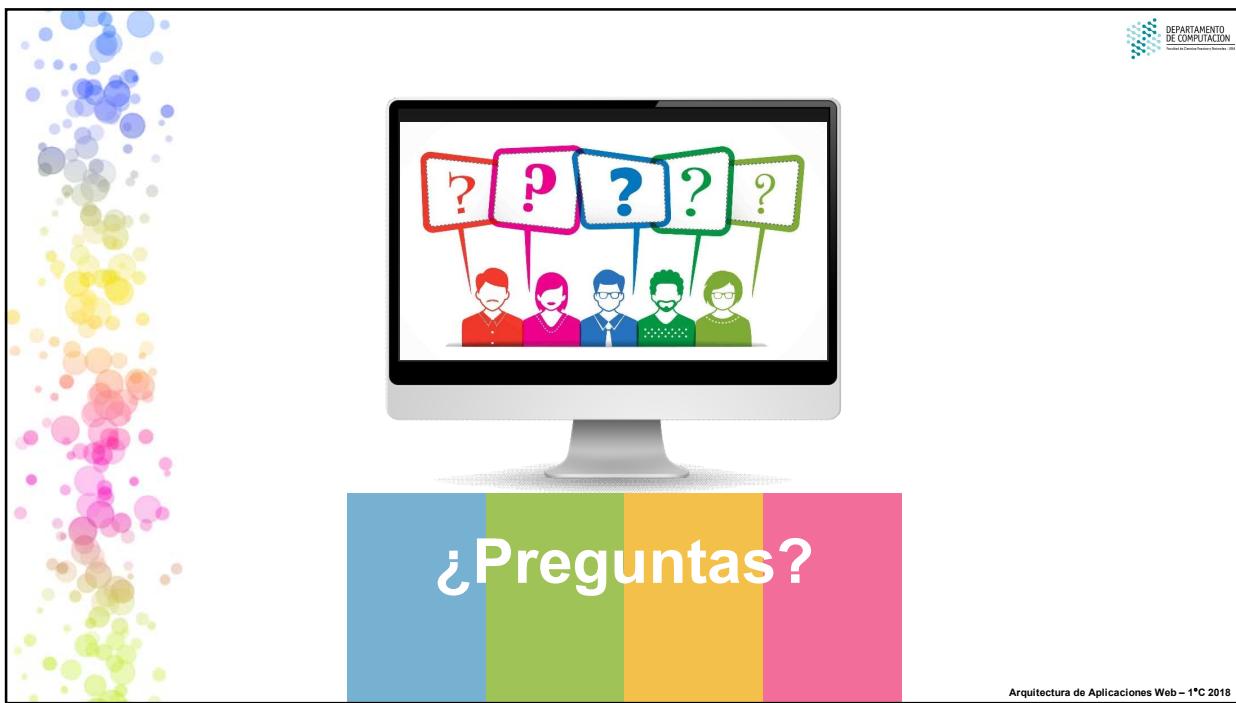
Otra manera de pensar la programación Web

The diagram illustrates the Ajax architecture. On the left, a screenshot of the Clarín.com website shows a news article. An arrow labeled "Request" points from the website to a green box labeled "Web container". Another arrow labeled "Response" points from the "Web container" back to the website. A double-headed arrow connects the "Web container" to a yellow cylinder labeled "Datos".

Arquitectura de Aplicaciones Web – 1ºC 2018



Arquitectura de Aplicaciones Web – 1ºC 2018



Arquitectura de Aplicaciones Web – 1ºC 2018

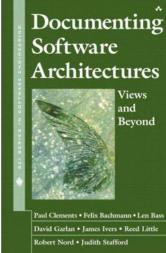
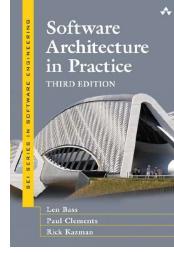
Bibliografía






DEPARTAMENTO
DE COMPUTACIÓN
Facultad de Ciencias Exactas y Naturales - UBA

- Documenting Software Architecture. Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford. 2002.
- Software Architecture in Practice. Len Bass, Rick Kazman, Paul Clements. 2012.
- Carlos Billy Reynoso, Introducción a la Arquitectura de Software, 2004.

Arquitectura de Aplicaciones Web – 1ºC 2018

Papers Fundacionales






DEPARTAMENTO
DE COMPUTACIÓN

- Frederick Brooks Jr. *The mythical man-month*. Reading, Addison-Wesley, 1975.
- David Parnas. "On the Criteria for Decomposing Systems into Modules." *Communications of the ACM* 15(12), pp. 1053-1058, Diciembre de 1972.
- David Parnas. "On a 'Buzzword': Hierarchical Structure", *Programming Methodology*, pp. 335-342. Berlin, Springer-Verlag, 1978.
- David Parnas. "On the Design and Development of Program Families." *IEEE Transactions on Software Engineering* SE-2, 1, pp. 1-9, Marzo de 1976.
- Mary Shaw. "Abstraction Techniques in Modern Programming Languages". *IEEE Software*, Octubre, pp. 10-26, 1984.
- Mary Shaw. "Large Scale Systems Require Higher- Level Abstraction". *Proceedings of Fifth International Workshop on Software Specification and Design*, IEEE Computer Society, pp. 143-146, 1989.
- Dewayne E. Perry y Alexander L. Wolf. "Foundations for the study of software architecture". *ACM SIGSOFT Software Engineering Notes*, 17(4), pp.40–52, 1992.
- Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. *Design Patterns: Elements of reusable object-oriented software*. Reading, Addison-Wesley, 1995.
- Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal. *Pattern-oriented software architecture – A system of patterns*. John Wiley & Sons, 1996.

Arquitectura de Aplicaciones Web – 1ºC 2018

