



UNIVERSIDAD
DE GRANADA



Diseño y Desarrollo de Sistemas de Información

Grado en Ingeniería Informática

Seminario 2 – Lenguajes y entornos de desarrollo

© I. J. Blanco, C. Cruz, C. Fernández, C de Mesa, M. J. Martín, A. Morales, D. Peralta, D. Sánchez

Departamento de Ciencias de la
Computación e Inteligencia Artificial
<http://decsai.ugr.es>

- Objetivos
- Un lenguaje específico: COBOL
- Lenguajes de propósito general
 - Arquitectura cliente-servidor de dos capas
 - Arquitectura cliente-servidor de tres capas
- SI basados en Aplicaciones Web
 - Front-End y Back-End
 - Lenguajes para desarrollo Front-End
 - Lenguajes para desarrollo Back-End
 - Web Frameworks
 - Aplicaciones de Página Única (SPA)
- Trabajo a realizar
- Bibliografía adicional

- Hemos visto cómo **es posible implementar un SI** utilizando **cualquier lenguaje de programación de propósito general** que disponga de bibliotecas y mecanismos para comunicarse con un SGBD.
- Pero también existen **lenguajes de propósito más específico** que se usan para la implementación de SI, entre los que sobresalen lenguajes para el Desarrollo Web. Algunos de éstos han ido incorporando elementos para convertirse en lenguajes de propósito general.
- También existen **entornos de desarrollo** y otras herramientas para facilitar la tarea de implementar SI, incluyendo algunos proporcionados como parte de distribuciones de ciertos SGBD.
- En este seminario revisaremos brevemente estos y otros conceptos y realizaréis un trabajo como parte de la evaluación de la asignatura.

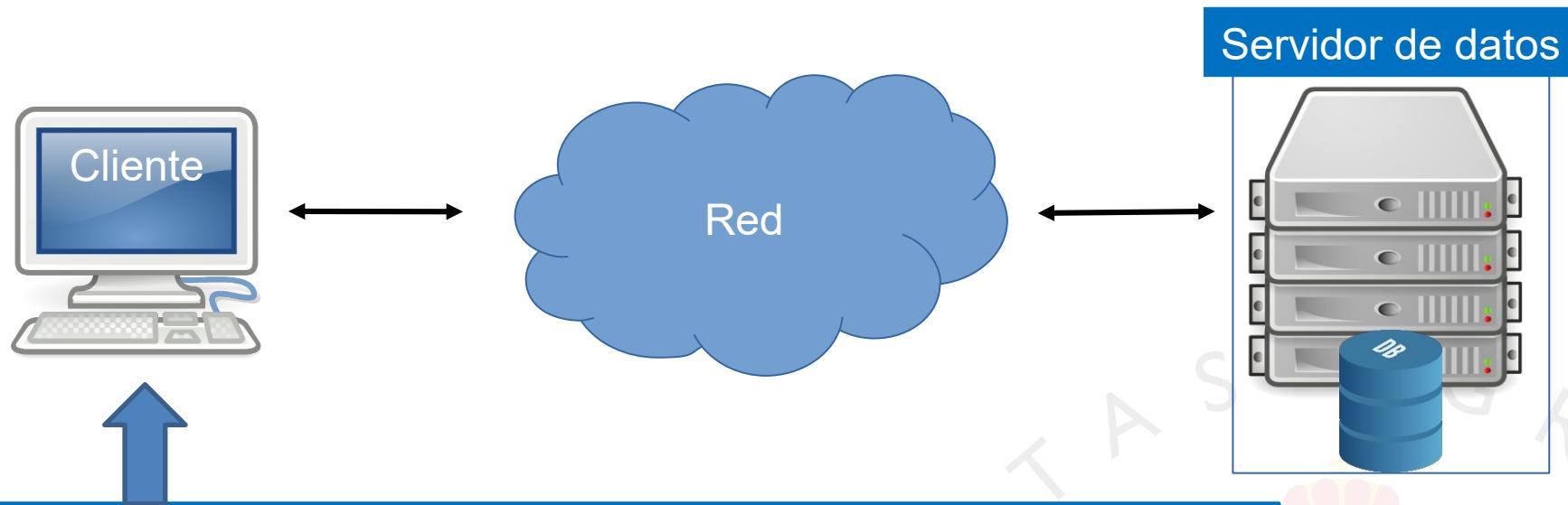
- **COBOL** (COmmon Business Oriented Language) es un lenguaje creado en 1959 para la programación de aplicaciones de gestión en los ordenadores disponibles en aquella época: Mainframes en arquitecturas centralizadas.
- Es un lenguaje muy bueno en la gestión rápida y eficiente de transacciones en Mainframes.
- Aunque ya nadie usa arquitecturas centralizadas ni COBOL para desarrollar un sistema nuevo, siguen existiendo muchos Mainframes (modernos) funcionando como servidores de datos realizando transacciones en bancos, venta de billetes de avión, etc.
- Esto se debe al coste de migrar los millones de líneas de código escrito en COBOL a una nueva arquitectura con tecnologías y lenguajes modernos (aparte de que sigue siendo muy seguro, efectivo y rápido).
- Existe una cierta demanda de conocimientos de COBOL para el mantenimiento de los sistemas existentes, así como cuando se quiere migrar un sistema de ese tipo a tecnologías más modernas.

- Usa el siguiente prompt en tu IA favorita:

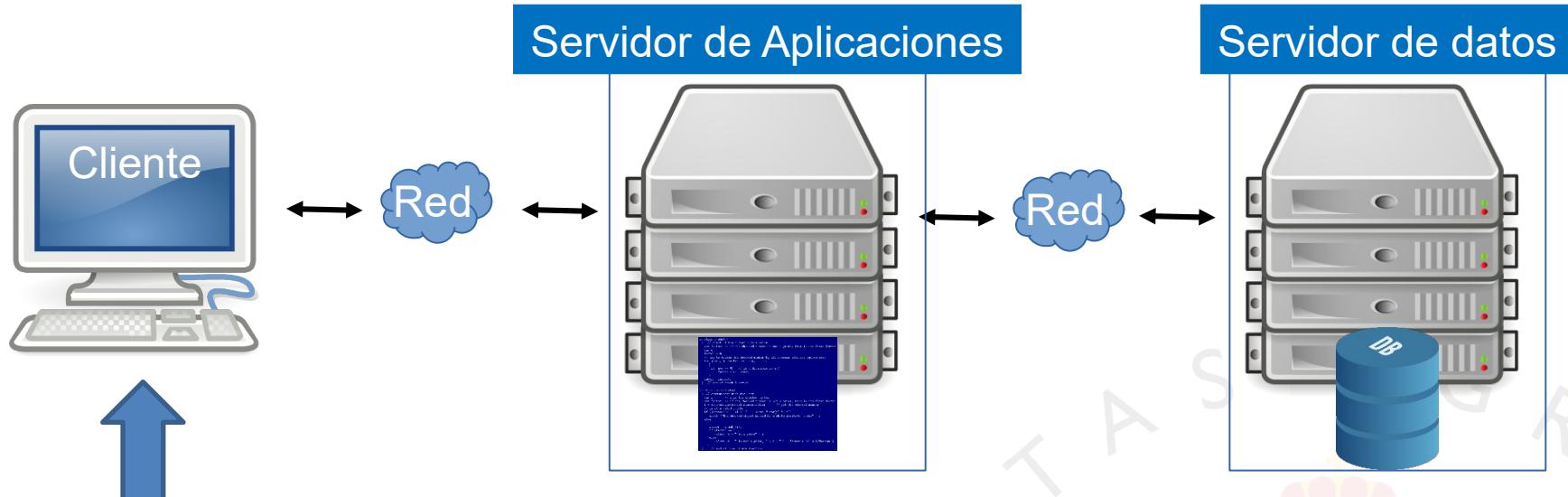
"Sin inventarte nada, atendiendo a la información disponible en Internet y actuando como experto en Sistemas de Información, dime cómo de importante es el lenguaje COBOL en la actualidad en el desarrollo de Sistemas de Información. Dame algunos enlaces donde se explique este tema."

- También puedes ver información sobre COBOL en

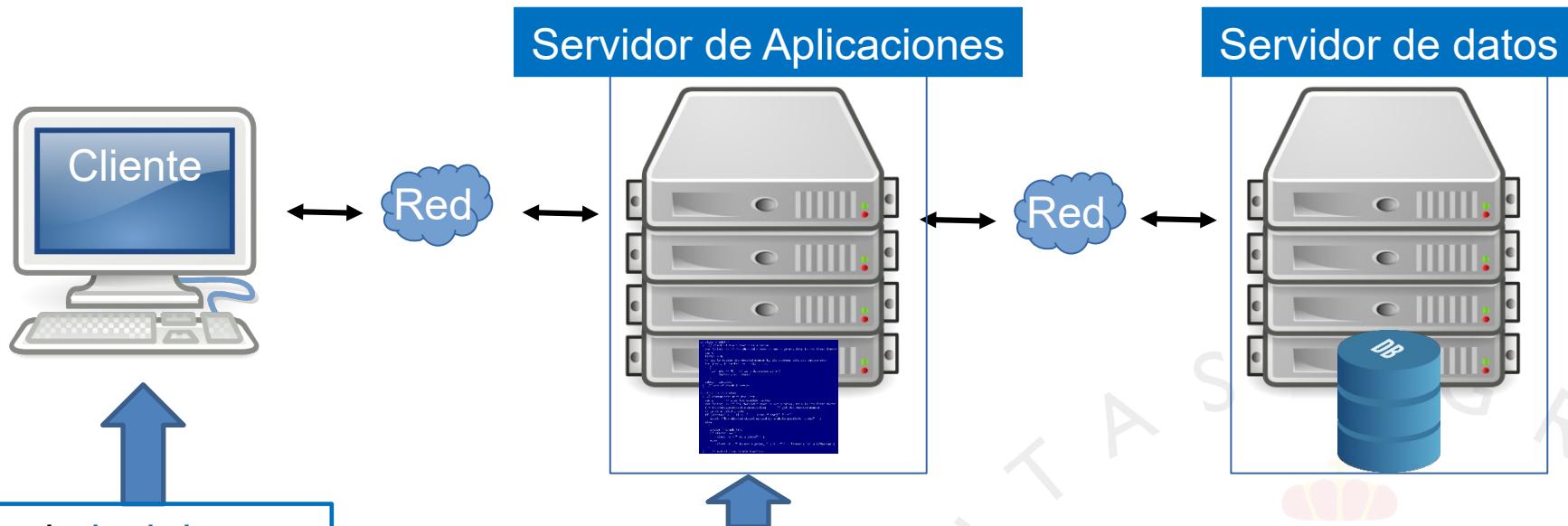
<https://es.wikipedia.org/wiki/COBOL>



- Podemos programar **aplicaciones en el cliente** utilizando lenguajes de propósito general, como hemos visto (**C, C++, Java, Python, Ruby, etc.**)
- Tal y como estás haciendo en el Seminario 1, el interfaz de usuario se puede implementar a través de un terminal de texto o mediante bibliotecas para el diseño de GUIs (Interfaces Gráficos de Usuario), y la conexión al SGBD puede realizarse mediante ODBC, JDBC u otros mecanismos, bien generales o bien específicos de un SGBD concreto.
- La aplicación y el SGBD pueden también estar en la misma computadora.

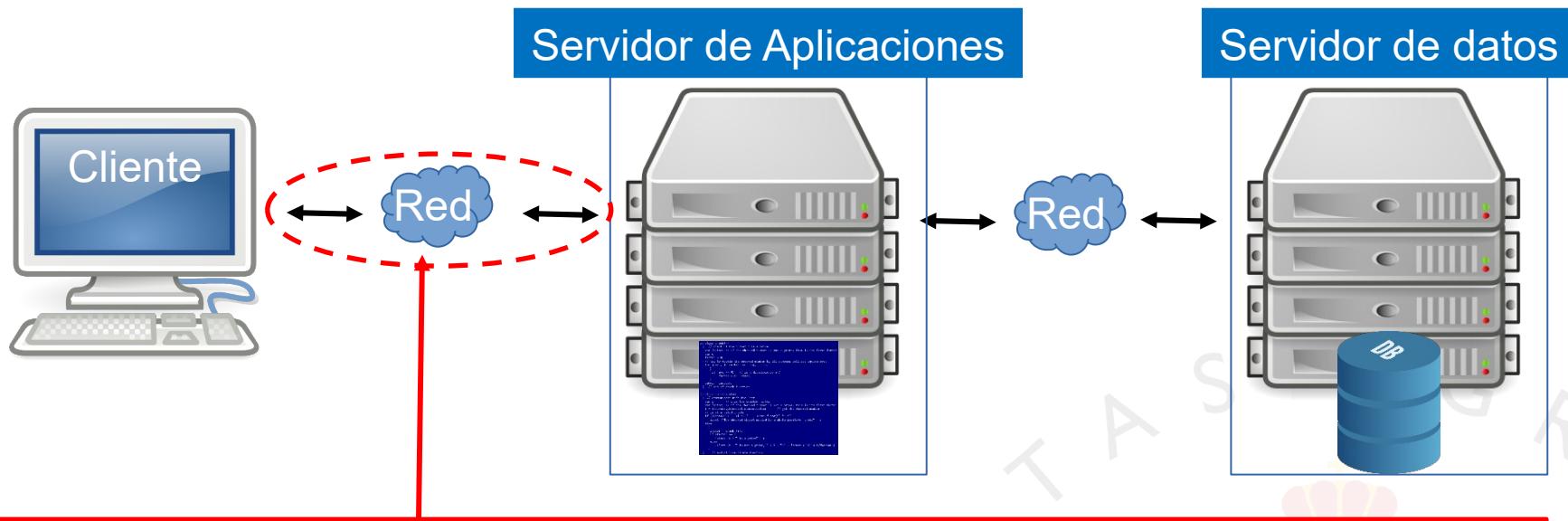


En el nivel de presentación se programa el interfaz de usuario mediante terminal o, más usual, como GUI.

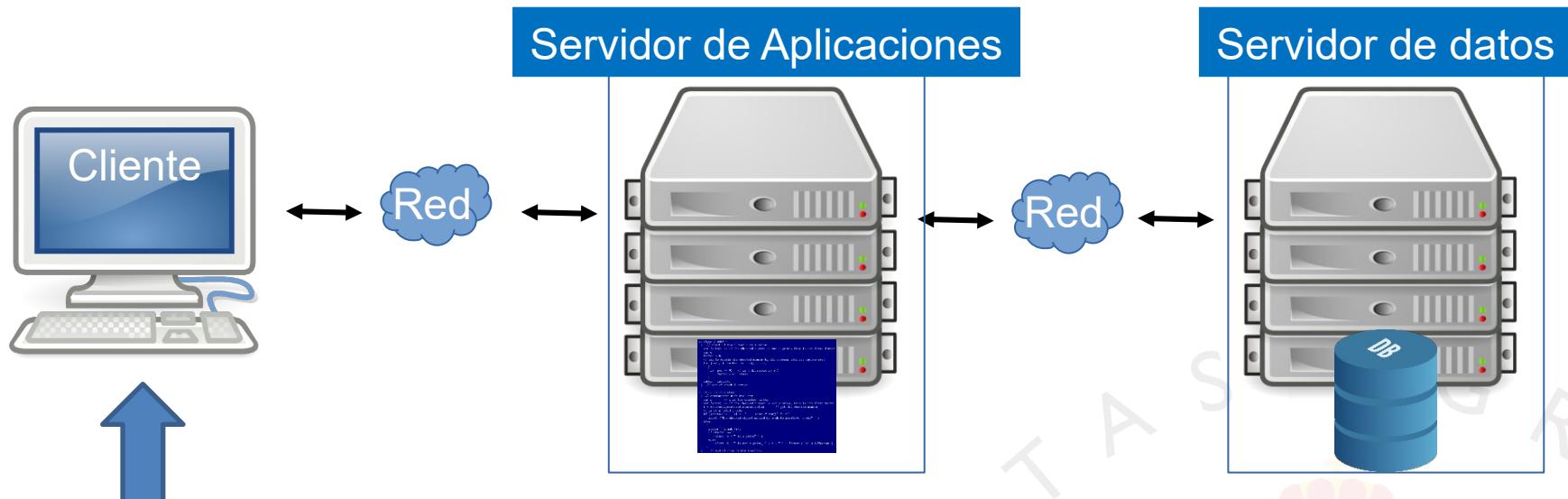


En el **nivel de presentación** se programa el interfaz de usuario mediante terminal o, más usual, como GUI.

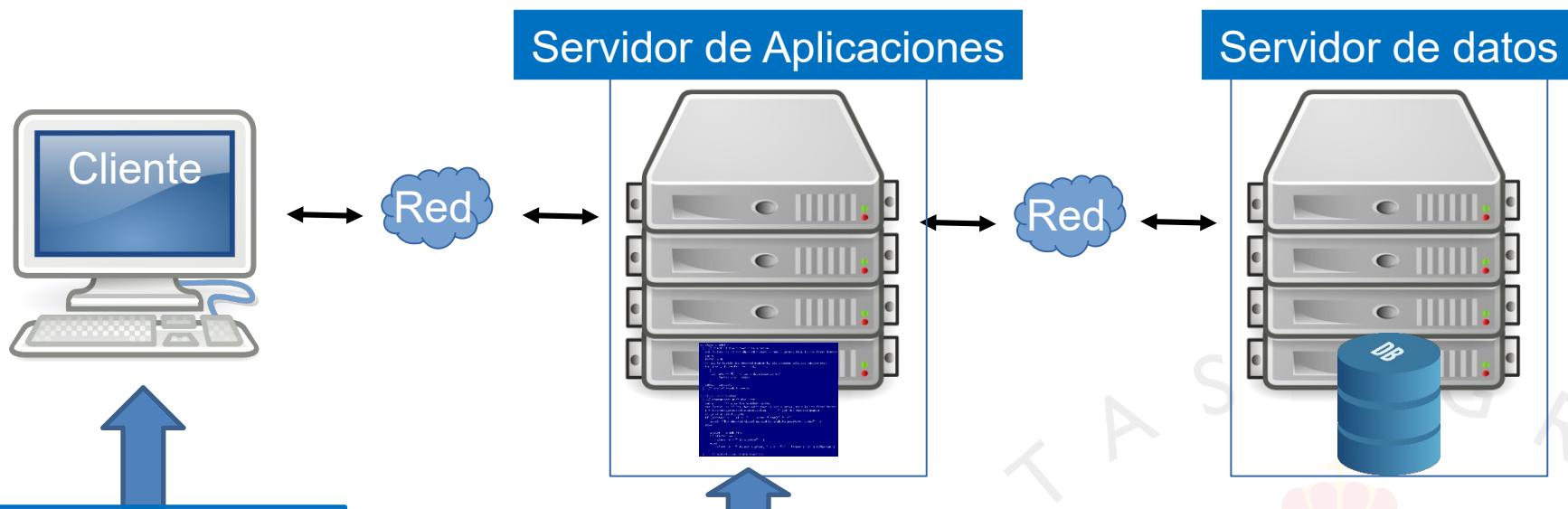
En el **nivel de aplicación** se programa el código que opera con los datos según las peticiones del usuario a través del interfaz, y se generan los resultados que se le envían al GUI para que los muestre como salida. Este código implementa lo que se llama el **Business Logic** (lógica de negocio; a veces, según sea la arquitectura del sistema, parte de ésta se programa en el nivel de presentación o, cuando está fuertemente ligada a los datos, en la propia BD).



- Se requiere el uso de **mecanismos de comunicación síncronos y/o asíncronos** entre las aplicaciones de los niveles de presentación y de aplicación. A estos mecanismos se les suele llamar (**Distribution**) **Middleware**.
- Los lenguajes de programación de propósito general suelen disponer de bibliotecas para programar con **distintos modelos de Middleware** (Sockets TCP/IP, RPC [Remote Procedure Call], Colas de mensajes, orientados a objetos como CORBA y DCOM, basado en componentes como EJB y .NET).
- **Alternativa: SI basados en Aplicaciones Web, vamos a verlos.**

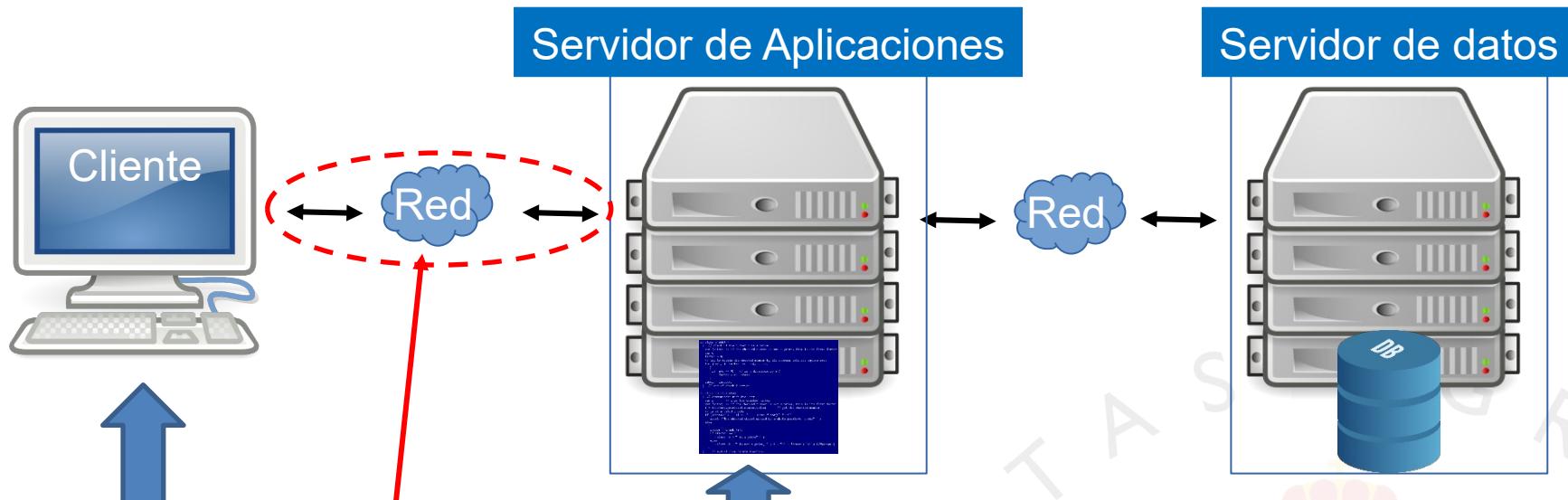


En el **nivel de presentación** tenemos un Navegador Web. El interfaz de usuario del SI toma la forma de una página Web, accesible mediante una URL.



En el **nivel de presentación** tenemos un Navegador Web. El interfaz de usuario del SI toma la forma de una página Web, accesible mediante una URL.

- En el **nivel de aplicación** tenemos un **servidor Web** que recibe las peticiones del interfaz desde el Navegador y devuelve páginas Web a visualizar en el cliente, descritas típicamente mediante los lenguajes **HTML, CSS y JavaScript**, así como datos descritos típicamente mediante **XML y JSON**.
- En el servidor se implementa la lógica de negocio mediante lenguajes de propósito general (**Java, Python, Ruby, C, C++, C#, etc.**), algunos muy especializados para Web como **PHP**.



En el **nivel de presentación** tenemos un Navegador Web. El interfaz de usuario del SI toma la forma de una página Web, accesible mediante una URL.

- En el **nivel de aplicación** tenemos un **servidor Web** que recibe las peticiones del interfaz desde el Navegador y devuelve páginas Web a visualizar en el cliente, descritas típicamente mediante los lenguajes **HTML, CSS y JavaScript**, así como datos descritos típicamente mediante **XML y JSON**.
- En el servidor se implementa la lógica de negocio mediante lenguajes de propósito general (**Java, Python, Ruby, C, C++, C#, etc.**), algunos muy especializados para Web como **PHP**.

Esta comunicación se establece de forma transparente a desarrollador y usuario, típicamente mediante **HTTP(S) (Hypertext Transfer Protocol)**

- Las aplicaciones Web se suelen dividir en dos componentes lógicos llamados **Front-End** y **Back-End**. No siempre son diseñados o programados por la misma persona.
- El **Front-End** es todo el software que tiene que ver con el interfaz de usuario, por tanto con la gestión del contenido y apariencia de la(s) página(s) Web que forman el interfaz en una aplicación Web.
- El **Back-End** es todo el software que se encarga de la lógica de negocio, el intercambio de datos con el SGBD y la gestión de los mismos.
- **No confundir con cliente-servidor!** Dependiendo del tipo de lenguajes y entornos utilizados, el software de ambos **Front-End** y **Back-End** puede ejecutarse total o parcialmente en el cliente o en el servidor, aunque lo “habitual” es que la mayor parte del **Front-End** se ejecute en el cliente (una vez descargado por el navegador) y que el **Back-End** se ejecute en el servidor de aplicaciones, comunicándose con el **Front-End** mediante ficheros XML y JSON.



Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

<https://francescolelli.info/software-engineering/front-end-development-back-end-development-and-full-stack-developers/>

- Podemos ver las principales tecnologías y lenguajes que deben dominar los desarrolladores de **Front-End** y **Back-End**, respectivamente. **Las personas que controlan ambos aspectos** se suelen denominar desarrolladores Full-stack (**Full-Stack developers**).

HTML



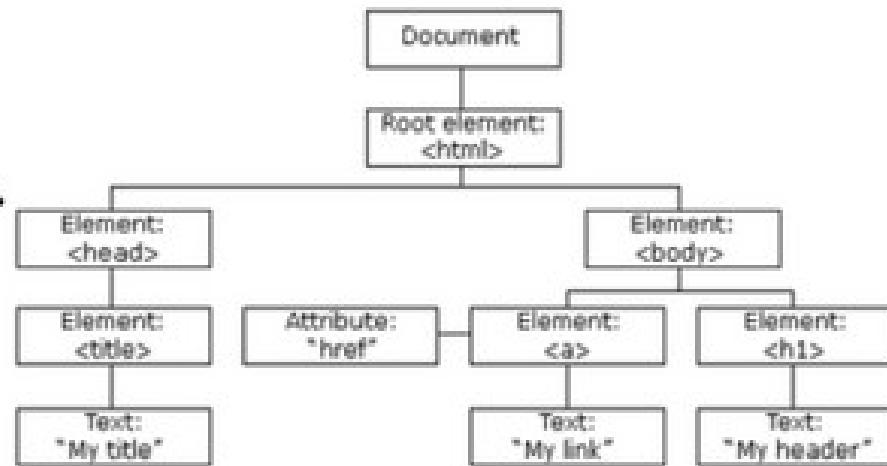
- **HTML** (HyperText Markup Language) es un lenguaje de marcado para codificar documentos (particularmente y sobre todo páginas Web), su estructura y componentes (texto, imágenes, video, ...).
- Es un **estándar del World Wide Web Consortium** (W3C) utilizado por todos los navegadores actuales para la codificación de páginas Web.
- El proceso por el que un navegador muestra una página web en HTML se denomina **rendering**. Para ello, el navegador obtiene a partir de HTML una representación del documento en forma de árbol, donde cada nodo es un objeto que corresponde a una parte de la página web.
- Esta representación se basa en el interfaz **DOM (Document Object Model)**, que proporciona un conjunto estándar de objetos para representar documentos HTML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. También es un estándar W3C.

HTML

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="href">My link</a>
    <h1>My header</h1>
  </body>
</html>
```



DOM



<https://medium.com/the-school-of-js/dom-manipulation-1-ee42bc0a9f41>

- **CSS** (Cascading Style Sheets), es un lenguaje para definir y crear el diseño gráfico de documentos escritos en un lenguaje de marcado como XML o HTML, entre otros. También W3C.
- Separa formato del contenido, por lo que el mismo código CSS puede ser usado por distintas páginas HTML.



https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

```
1 /* Hoja de estilos CSS */
2
3 body {
4     font-family: inherit;
5     background-color: lightblue;
6     color: #2f2f2f;
7     font-size: 1em;
8     font-weight: normal;
9 }
10
11 .wrapper {
12     border: 1px solid #999;
13     border-radius: 5px;
14     color: #222;
15     padding: 10px;
16     background: #ddd;
17 }
18
19 .element-1 {
20     color: #fff;
21     font-size: 22px;
22     line-height: 32px;
23     background: #569dc5;
24     padding: 25px 30px 20px 30px;
25     font-weight: 400;
26 }
27
28 .box-1 {
29     margin: 0;
30     min-width: 190px;
31     border-radius: 0;
32     box-shadow: none;
33 }
34
35 /* https://es.wikipedia.org/wiki/CSS */
```

CSS



JS

- **JavaScript** es un lenguaje que **se ejecuta en los Navegadores Web** (por tanto **en la capa de presentación**) para aportar **características dinámicas a páginas web**, es decir, cambios en la apariencia y comportamiento de éstas ante eventos como pulsar un botón con el ratón, sin necesidad de recargar nuevos documentos HTML desde el servidor Web.
- La mayoría de los Navegadores dispone para ello de un **motor Javascript** para la ejecución de ese código.
- La biblioteca **jQuery** permite **modificar el DOM** de la página para modificar la apariencia de la misma desde el cliente, ahorrando la petición al servidor de un nuevo HTML y la descarga y rendering del mismo. También permite animación CSS, gestión de eventos y comunicación asíncrona con el servidor.
- Dispone de APIs para trabajar con texto, fechas, expresiones regulares y estructuras de datos estándar. No tiene APIs de E/S, la hace el Navegador.
- Aunque por el nombre y la sintaxis pueda entenderse lo contrario, **es muy diferente al lenguaje Java**. Sí comparten algunas bibliotecas estándar.



- El código **JavaScript** se inserta en un documento HTML utilizando la etiqueta <script>.

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

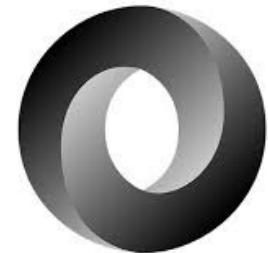
<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```



Para verlo funcionar, https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto_head

- **XML** (Extensible Markup Language) es un lenguaje para representar el contenido de documentos de forma que sea legible tanto para personas como para computadoras.
- Es un estándar W3C que **se utiliza fundamentalmente para la representación de estructuras de datos** de diverso tipo mediante documentos de texto, que pueden usarse **para la transmisión de datos entre aplicaciones en base a APIs**.
- **JSON** (JavaScript Object Notation) es un estándar abierto alternativo a XML, que surge inicialmente para almacenar y transmitir objetos JavaScript, aunque hoy en día se utiliza como un lenguaje independiente para representar y transmitir datos.
- Al igual que ocurre con XML, muchos lenguajes de programación disponen de APIs para la generación y parsing de ficheros JSON.

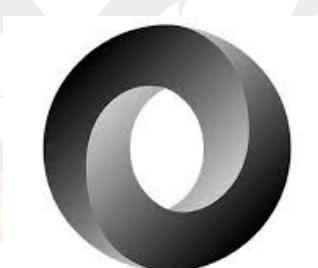


JSON

```
<person firstName="John" lastName="Smith" age="25">
  <address
    streetAddress="21 2nd Street"
    city="New York"
    state="NY"
    postalCode="10021" />
  <phoneNumbers>
    <phoneNumber type="home" number="212 555-1234"/>
    <phoneNumber type="fax" number="646 555-4567"/>
  </phoneNumbers>
</person>
```

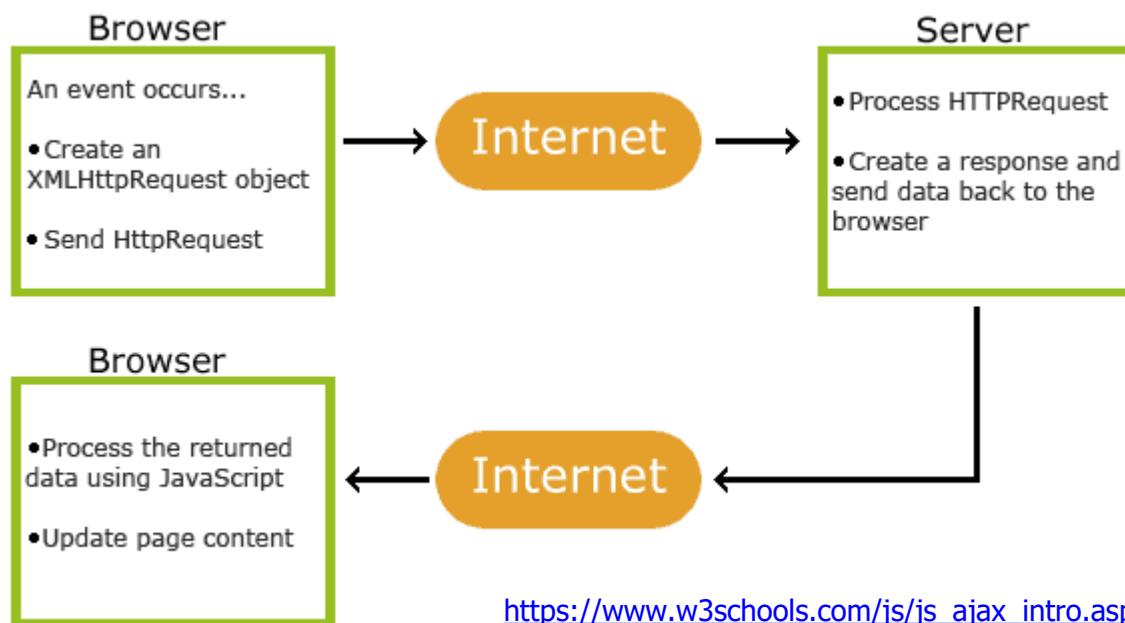


```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```



<https://www.startuprocket.com/articles/a-quick-introduction-to-xml-and-json>

- **AJAX** (Asynchronous JavaScript And XML) no es un lenguaje, sino que representa el uso conjunto de JavaScript y XML (o JSON) para realizar comunicaciones asíncronas con el servidor Web.
- Asíncrono significa que la petición y recepción de datos al servidor se realizan en segundo plano, permitiendo que la página Web siga activa y funcionando hasta que se reciben los datos y se actualiza la página. Estas peticiones se hacen usando el objeto JavaScript llamado **XMLHttpRequest**.



https://www.w3schools.com/js/js_ajax_intro.asp

- El comportamiento dinámico de una página Web puede generarse o bien en el cliente o bien en el servidor mediante lo que se denomina respectivamente “client-side scripting” y “server-side scripting”. La programación de este comportamiento dinámico se lleva a cabo mediante “lenguajes de scripting” (*scripting languages*).
- JavaScript es un ejemplo de lenguaje de scripting en el cliente.
- En el lado del servidor, existen asimismo muchos lenguajes de scripting (interpretados), que se utilizan no sólo para generar el contenido del interfaz Web, sino para otros aspectos importantes como la gestión de sesiones de usuario, conexión a base de datos, implementación del flujo de control del sistema, etc.
- Uno de los más conocidos y utilizados es PHP.

- **PHP** es un lenguaje de programación de propósito general que se adapta especialmente bien al desarrollo Web, donde se utiliza en el desarrollo del Back-End. La forma habitual de usarlo en este ámbito es que el resultado de ejecutar un fichero .php se envía al Navegador en forma de código HTML.
- La implementación de referencia es la que produce “[The PHP Group](#)”, la última versión estable es la 8.0.12 (21 de octubre de 2021). La mayoría de Webs actuales utilizan versiones anteriores (7.3, 7.4, etc.).
- El código **PHP** suele ser [procesado en un servidor web por un intérprete PHP](#). El motor de procesamiento más utilizado para ello es [Zend](#), software libre bajo la licencia **PHP**, disponible para la mayoría de servidores y SO.
- Existen otros lenguajes de scripting similares a **PHP** que se utilizan para los mismos propósitos. Ejemplos:
 - Microsoft **ASP** (Active Server Pages)
 - **JSP** (Java Server Pages), utiliza el lenguaje Java para programar [contenedores servlet](#), por lo que requiere servidores web compatibles como [Apache Tomcat](#).



- Se puede incrustar código **PHP** dentro de HTML utilizando los delimitadores `<?php` para abrir una sección **PHP** y `?>` para cerrarla. El intérprete sólo ejecuta el código que encuentra entre los delimitadores.



```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Para verlo funcionar, https://www.w3schools.com/php/phptryit.asp?filename=tryphp_intro

- Con **PHP** podemos:

- Generar páginas Web de manera dinámica
- Llevar a cabo cualquier tipo de operación con ficheros en el servidor (crear, abrir, leer, etc.)
- Recolectar datos a través de formularios
- Enviar y recibir cookies
- Acceder a la BD y llevar a cabo operaciones
- Controlar el acceso de usuarios
- Encriptar datos
- Etc.

- **PHP** ofrece:

- Multiplataforma (Windows, Linux, Unix, Mac OS X, etc.)
- Compatible con la mayoría de servidores Web (Apache, IIS, etc.)
- Permite conectar a las principales BD relacionales usando su propio Driver Manager PDO. Entre otros drivers, dispone de uno para conectar a cualquier fuente de datos a través de ODBC.
- Se aprende rápido y se ejecuta de forma eficiente



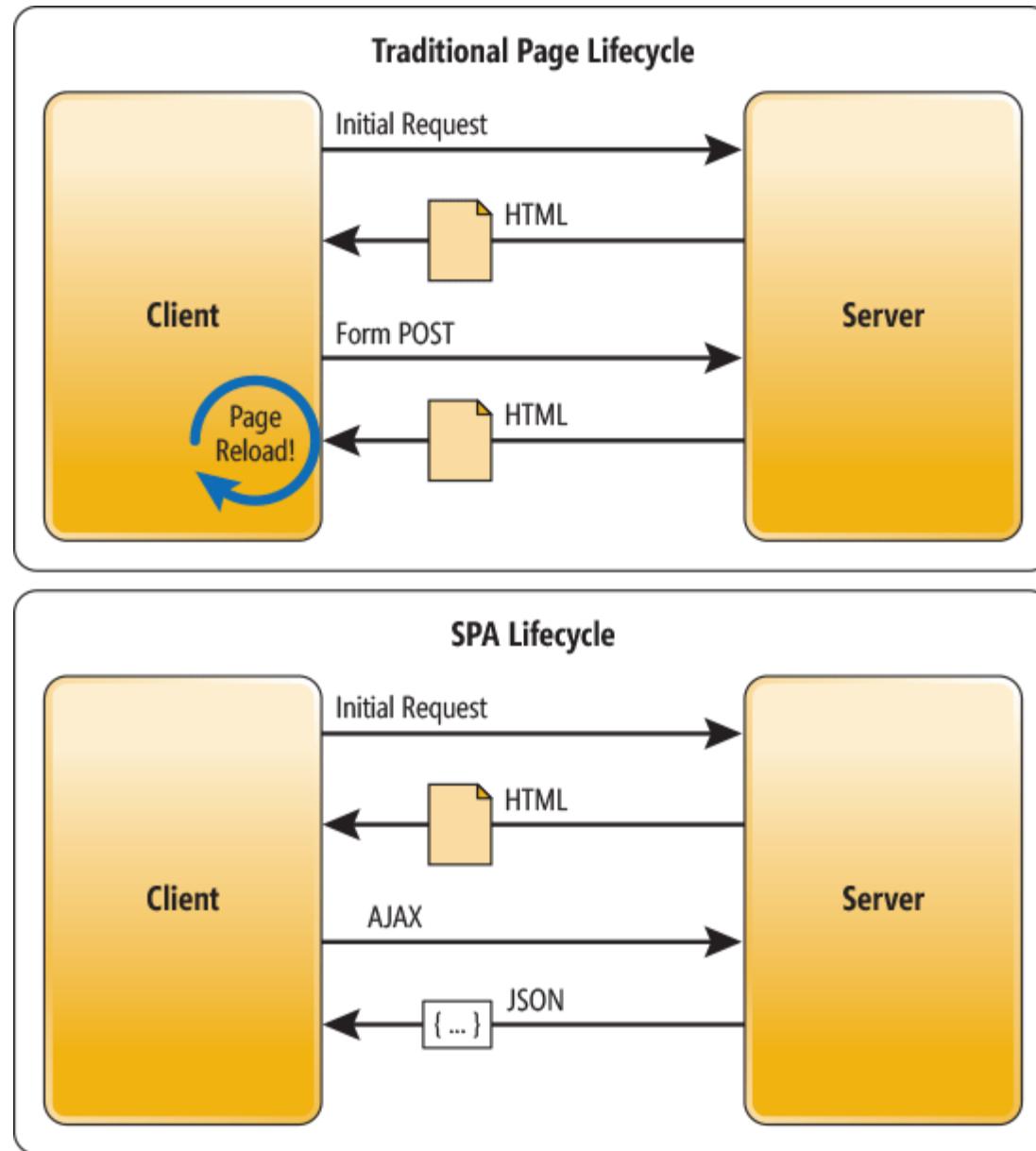
- Hay muchos otros lenguajes, algunos bien conocidos, que pueden emplearse a través de Entornos de Desarrollo Web (Web Frameworks o **Web Application Frameworks, WAF**).
- Los **WAF** facilitan el desarrollo y despliegue de aplicaciones Web, proporcionando bibliotecas para tareas comunes como acceso a bases de datos, gestión de sesiones, plantillas de documentos y páginas y reutilización de código, entre otros. **Suelen usar un único lenguaje de programación.**
- Los **WAF** suelen usar para la implementación de la aplicación Web el patrón de diseño **MVC (Model-View-Controller)**, permitiendo definir tres tipos de componentes:
 - **Modelo (Model)**: componente que gestiona los datos y sus restricciones.
 - **Vista (View)**: cualquier representación visible de los datos en cualquier formato (tabla, diagrama, etc.) en el interfaz
 - **Controlador (Controller)**: recibe comandos y datos y los convierte en instrucciones y/o datos para el Modelo. También en instrucciones para que el View muestre los datos del Modelo.

Lenguaje	WAFs
Python	Django, Flask, Jam.py, Pyramid, web2py
Ruby	Ruby on Rails
Java	Struts, Wicket, JavaServer Faces, Tapestry
PHP	Laravel, Symfony, CakePHP
C#	ASP.NET Core
C++	Dragon, CppCMS
Perl	Catalyst, Mojolicious
JavaScript (sobre Node.js)	Express, Sails.js, Meteor

Aunque JavaScript es un lenguaje estándar para programación del Front-End, puede usarse también en el Back-End gracias a Node.js, que es un entorno de ejecución de código JavaScript fuera de un navegador Web. Esto permite utilizar el mismo lenguaje de scripting en Back-End y Front-End.

... y muchos más, https://en.wikipedia.org/wiki/Comparison_of_web_frameworks

- Las **Aplicaciones de Página Única** (Single Page Applications, **SPA**) son aplicaciones Web que implementan toda la funcionalidad en el Front-End, excepto el acceso a los datos. Tratan de dar la sensación de ser una aplicación de escritorio convencional, pero usando el navegador como interfaz para mostrar una única página. Ejemplo: GMail
- El código **HTML, CSS y JavaScript** se carga una única vez desde el servidor y se ejecuta en el cliente, ganando en rapidez.
- Es un enfoque muy extendido en la actualidad que puede implementarse mediante AJAX, pero para el cual existen diversos **WAF** que facilitan mucho el desarrollo, todos ellos basados en el lenguaje **JavaScript**.
- Entre los **WAF** más utilizados para programación Front-End de SPA están **AngularJS de Google, ReactJS de Facebook, EmberJS, Vue.js y Backbone.js**. Otro **WAF** de Google es **Angular**, basado en el lenguaje **TypeScript**, un lenguaje que añade diversas capacidades a JavaScript, y que transcompila a código JavaScript. Y, como siempre, hay más ...



- El trabajo se realizará **por subgrupos** (los mismos de prácticas).
- El trabajo se realizará con la herramienta **Oracle APEX**.
Opcionalmente también se admitirá que se realice un estudio de un WAF, previa consulta con el profesor.
- En el caso de **Oracle APEX**, se dispone de cuentas en la nube de Oracle, una por grupo de prácticas. Para recibir los datos de acceso, una persona por cada grupo deberá ponerse en contacto con su profesor/a de prácticas y solicitar la información de enlace y datos de la cuenta.

- El trabajo consistirá en.

- Instalar el software en su caso, no es necesario con APEX
- Buscar y ejecutar al menos un tutorial que permita implementar una aplicación Web que incluya conexión a base de datos y tenga un comportamiento muy sencillo. En el caso de APEX, pueden encontrarse varios tutoriales al entrar en la cuenta.
- Cada persona del grupo implementará una sola funcionalidad de su subsistema, a acordar con el profesorado de prácticas.
- La implementación se hará creando según sea necesario un conjunto de tablas preliminar que no tiene por qué ser la que se obtenga finalmente en la práctica 2. La propia implementación de la funcionalidad puede ser también aproximada, no teniendo que ser necesariamente ajustada exactamente a la práctica 1.

- Se entregará a través de Prado un único **fichero .pdf** explicando el trabajo realizado, que contendrá:
 - Referencia a los tutoriales utilizados (no es necesario describir el proceso de realización).
 - Descripción de las tablas creadas
 - Funcionalidad implementada por cada persona, indicando los cambios respecto a lo especificado en la práctica 1 cuando haya sido necesario.
 - Conceptos y tecnologías utilizados de entre los estudiados en este Seminario.

● Evaluación:

- Se recuerda que el trabajo a realizar en este Seminario 2 forma parte de la evaluación de la asignatura, ver Guía Docente.
- La calificación final se obtendrá como resultado de:
 - La evaluación por el profesor de la entrega, y
 - La defensa grupal del trabajo, donde a cualquier persona del grupo se le puede preguntar por cualquier aspecto del trabajo.
 - Se hará una demostración de las funcionalidades implementadas.
 - Se pretende sobre todo comprobar, a través del trabajo entregado, que se han comprendido los conceptos vistos en el Seminario.

Lenguajes y Entornos para el desarrollo de Front-End y Back-End en Aplicaciones Web, con tutoriales y pruebas:

- <https://www.w3schools.com/default.asp>

Lenguajes de scripting en el servidor:

- https://en.wikipedia.org/wiki/Server-side_scripting

Aplicaciones de Página Única (SPA)

- https://es.wikipedia.org/wiki/Single-page_application