Interfaz: Funcionalidades visuales accesibles a través del navegador

Lógica de negocio: “Inteligencia” de la aplicación. Incluye los procesos que involucran a la aplicación. Conjunto de operaciones requeridas para proveer el servicio.

Administración de los datos: Gestión de las BBDD y archivos

Modelo de dos capas:

* Gran parte de la aplicación corre en el lado del cliente (cliente pesado)
* La lógica de negocio se encuentra en el lado del cliente
* Inconvenientes
  + Difícilmente escalable
  + Número reducido de conexiones
  + Carga de la red alta
  + Funcionalidad limitada

Modelo de tres capas:

* Diseñada para superar limitaciones de modelo de dos capas
* Una capa intermedia contiene la lógica del negocio
* Las aplicaciones web actuales se ajustan a este modelo

Detalle modelo de tres capas:

Capa de presentación (parte en el cliente y parte en el servidor)

* Recoge la información del usuario y la envía al servidor para que la capa de negocio la procese
* Procesa determinada información
* Recibe los resultados de la capa de proceso
* Generan la presentación y la muestran

Capa de proceso o negocio (parte en el cliente y parte en el servidor)

* Recibe datos de la capa de presentación
* Interactúa con la capa de datos para realizar validaciones y operaciones
* Contiene flujos y procesos
* Manda los resultados procesados a la capa de presentación

Capa de datos (servidor/es de datos)

* Almacena y gestiona los datos
* Asegura su integridad

Navegadores: Arquitectura de ejecución:

Un navegador es la aplicación informática que permite visualizar un recurso identificado por una URL.

Dentro de la diversidad existente los hay que priorizan una respuesta rápida. La calidad del resultado, o que priman la seguridad de las comunicaciones, entre otras cosas.

Navegadores: Características:

* Plataformas
* Características propias: gestor descargas, almacenamiento de contraseñas, formularios, búsquedas, traducciones
* Personalización de la interfaz: navegación por pestañas, ventanas emergentes, visualizaciones de PDF, zoom
* Soporte de tecnologías Web: CSS, Java, RSS, XTML
* Tipo de licencia: libres o propietarios

Navegadores: Arquitectura de ejecución:

Para ello todos los navegadores siguen una arquitectura de referencia que dispone de los siguientes componentes básicos:

User Inteface 🡪 Browse engine 🡪 Rendering engine 🡪 Networking,

JavaScript Interpreter

UI Backend

Data Persistence

Interfaz de usuario:

Permite la interacción del usuario con los elementos visuales disponibles:

Motor de navegación (buscador):

Es quien obtiene la dirección solicitada (URL) y provee los mecanismos de navegación (proceso de carga, ir a la página siguiente)

Subsistema de persistencia de datos: Los datos que los subsistemas del navegador necesitan almacenar localmente se guardan en ese sistema de almacenamiento persistente. Por ejemplo:

* Gestión del historial
* Mantenimiento de sesiones de usuarios en disco
* Almacenamiento de las preferencias de configuración del usuario
* Gestión del listado de marcadores
* Administración de certificados de seguridad y cookies

Motor de renderizado: Interpreta el código y realiza la representación visual del recurso obtenido a partir del acceso a la url.

Es capaz de mostrar documentos HTML, XML, hojas de estilo CSS e incluso contenido embebido (audios/videos).

Cada navegador tiene su propia forma de interpretar las páginas web en la pantalla de un usuario, por lo que el desarrollador web debe comprobar la compatibilidad entre navegadores…

Comunicaciones:

Implementa los protocolos de transferencia de ficheros y documentos de Internet (HTTP, FTP, ...)

Es responsable de la codificación usada en texto, audio, video… (MIME)

Puede almacenar una caché de elementos a recientemente

Gestiona la seguridad

UI Backend: Parser XML:

Este modulo carga en memoria una representación en árbol (DOM, DocumentObjectModel) de la pagina para proporcionar un acceso más rápido a los distintos elementos…

Motores de renderizado algunos motores:

Google Chrome: Blink

Opera v, 15+\*: ~~Presto~~, Blink, ~~Webkit~~ \*ver versiones

Internet Explorer: Trident

Mozilla Firefor: Gecko, Webrender

Chrome para iOS y Safari: WebKit

Edge: ~~edgeHTML~~, Blink

Blink (The Chromium Project)

<https://www.chromium.org/blink/>

<https://vivaldi.com/es/>

Vivaldi: Basado en Chromium, pero distinto a Chrome | Vivaldi Browser

Lenguaje de programación en entorno cliente: Son los que se ejecutan en el navegador, i.e., en el lado del cliente de una arquitectura cliente/servidor

JavaScript: Lenguaje de programación del lado del cliente: se ejecuta en el navegador del cliente. NO en el servidor web.

* Interpretado. No hay compilación línea a linea
* Orientado a objetos
* Objetual
* Imperativo
* Estructurado
* Tipificación débil
* Tipificación dinámica

Escribir en cualquier elemento HTML con: *document.getElementByID(“identificador id”).innerHTML*

Generar HTML con el método: *document.writer*

Generar mensaje de alerta con: *Windows.alert o alert*.

Reglas básicas:

* Recomendable usar punto y coma al final de cada línea
* Se pueden agrupar varias instrucciones en la misma línea separándolas con “;”
* Los decimales con “.”
* Comentarios // o /\* \*/ si son varias líneas
* Tamaño ilimitado de líneas. Si se divide, hacerlo después de un operador.
* Var o let declaran las variables. No es imprescindible
* El operador de asignación es =. El resto +,-,\*,/,%... En versiones posteriores \*\*. También ++ y –
* Los bloques (funciones, bloques…) de códgio se limitan con {}
* Recomendado **camelCase** para definición de variables. Tambien “\_” y “$”.
* Hay una serie de palabras reservadas que no pueden ser usadas como identificadores: break, continue, switch, for, do… while, if… else, debugger, try…catch, var, function, return.

Variables:

Declaración:

Se declaran con **var** y pueden contener o no un valor

**Let** tiene en cuenta el ámbito donde se declara la variable

**Use strict** obliga a declarar variables

**Hoisting:** es un mecanismo que consiste en que el intérprete de JavaScript escanea el programa antes de ejecutarlo, buscando errores en la sintaxis.

Tipos de datos:

Ejemplo:

Var altura:

Console.log(altura); // - > undefined

Console.log(peso); // - > Uncaught ReferenceError: peso is not defined

Let someResource;

Console.log(someResource); // - > undefined

Console.log(typeof someResource); // - > undefined

someResource = null;

Console.log(someResource); // - > null

Console.log(typeof someResource); // - > object

Conversiones implícitas:

Const STR1 = 42 + “1”;

Console.log(STR1); // - > 421

Console.log(typeof STR1); // - > string

Const STR1 = 42 - “1”;

Console.log(STR1); // - > 41

Console.log(typeof STR1); // - > number

Objetos en JS:

String: Metodos:

CharAt(num)

localCompare(cadena)

includes(cadena)

startsWith(cadena)

endsWith(cadena)

toLowerCase()

toUpperCase()

Estructuras de control:

For in

Vectores:

.push(\_) -> mete otro valor en el array

.pop() -> elimina

.splice(\_,\_,\_) -> añadimos en una posición concreta

.splice(\_,\_) -> eliminamos en una posición concreta

.length -> saca la longitud del array

Number: Metodos:

toFixed -> cadena con el numero especifico de decimales indicado

toPrecision -> cadena con el numero especifico de cifras

toExponential -> cadena con el numero redondeado a notación científica

valueOf -> devuelve el valor primitivo de un objeto

toString -> devuelve la cadena de un numero