

***RECUERDA PONER A GRABAR LA
CLASE***





¿DUDAS DEL ON-BOARDING?

MIRALO AQUI



Clase 15. JAVASCRIPT

INTRODUCCIÓN A SPA CON JQUERY



OBJETIVOS DE LA CLASE

- Aprender qué es una SPA y qué beneficios tiene sobre las páginas.
- Entender el concepto MVC.

GLOSARIO:

Clase 14

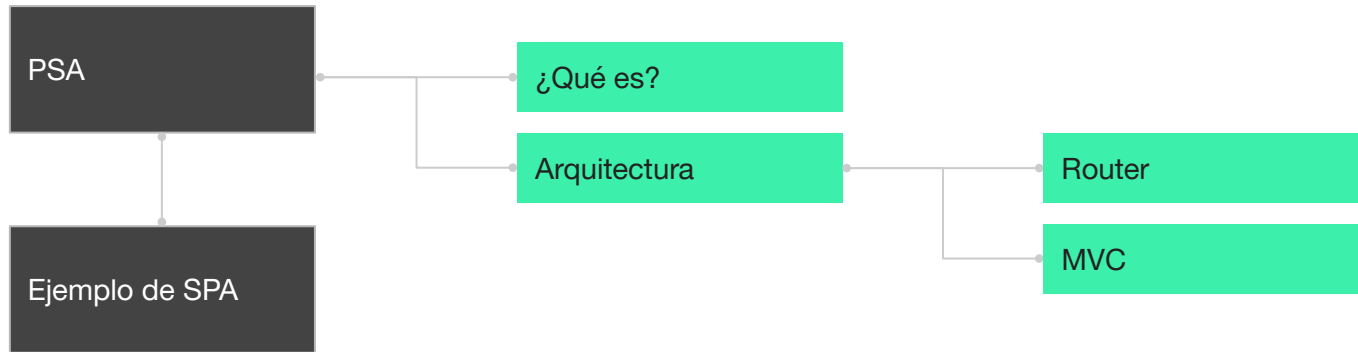
AJAX: significa JavaScript asincrónico y XML (Asynchronous JavaScript and XML). Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asincrónica, procesando cualquier solicitud al servidor en segundo plano.

API o Application Programming Interfaces: es un conjunto de reglas y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas.

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 15

¡Para
recordar!



CRONOGRAMA DEL CURSO

Clase 14



AJAX con jQuery



AJAX EN TU PROYECTO



TERCERA ENTREGA DEL
PROYECTO FINAL

Clase 15



Introducción a SPA con jQuery



EJEMPLO EN VIVO



PRACTICAR SPA

Clase 16



Workshop II



ENTREGA DEL PROYECTO FINAL



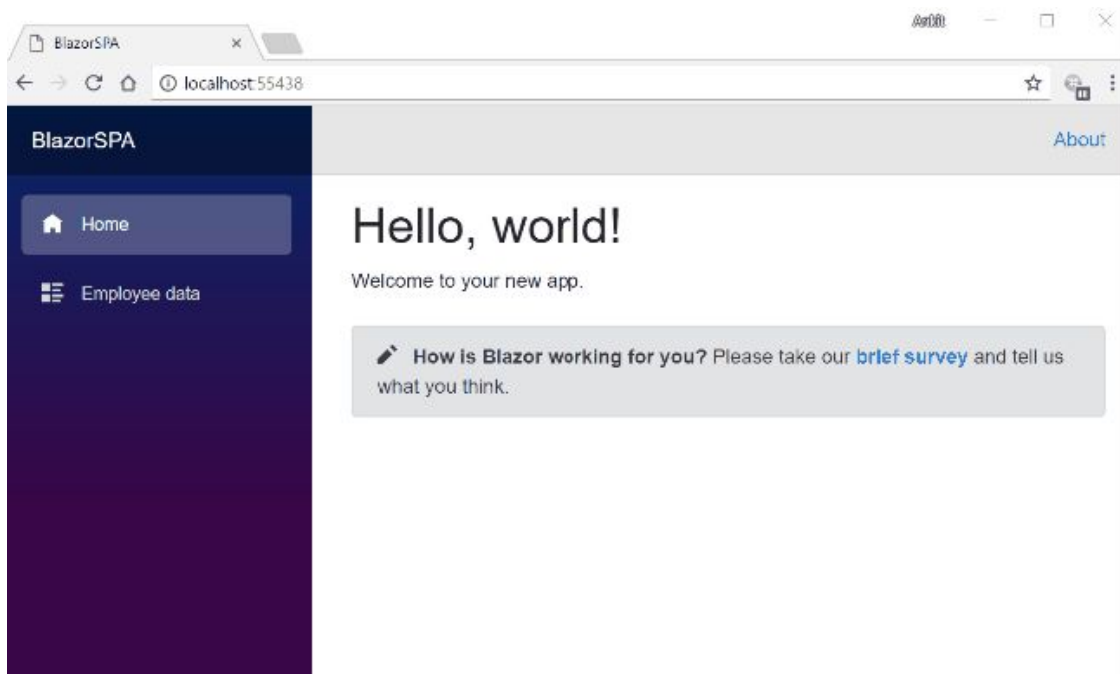
HERRAMIENTAS DE LA CLASE

Les compartimos algunos recursos para acompañar la clase

- Guión de clase N° 15 [aquí](#)
- Quizz de clase N° 15 [aquí](#)
- Booklet de Javascript [aquí](#)
- FAQs de Javascript [aquí](#)

SPA

¿QUÉ ES UNA APLICACIÓN SPA?

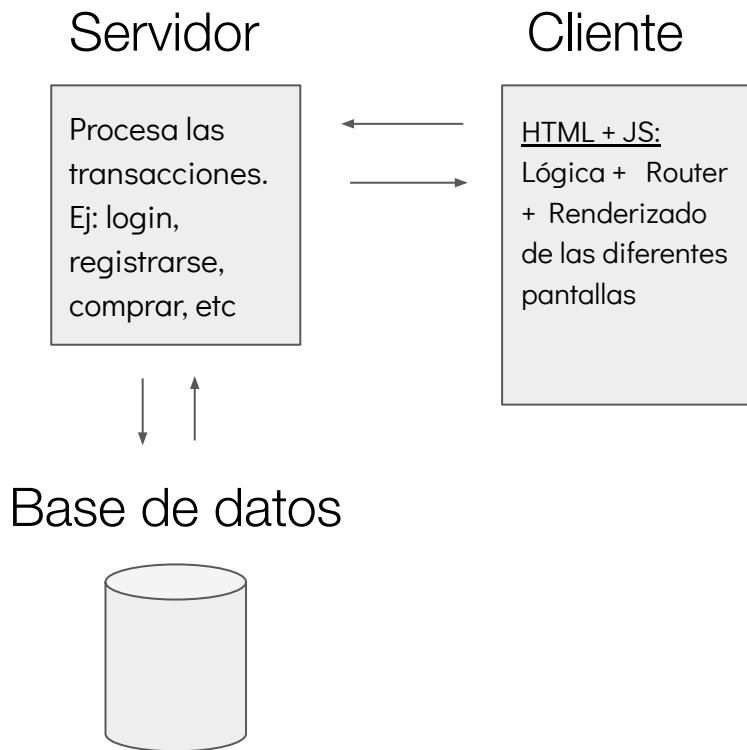


Por SPA se conocen las aplicaciones de una sola página o Single Page Applications.

La aplicación SPA se carga una única vez en el navegador, y luego todo intercambio de contenidos se realiza sin recargar la pantalla.

Ejemplos de aplicaciones SPA son Slack, Discord y Netflix.

ARQUITECTURA DE UN SPA



La mayor parte de la funcionalidad de la aplicación (la lógica) queda en el cliente (navegador).

Se accede y envía datos al servidor mediante AJAX, siendo la arquitectura del backend, generalmente una API REST. Todo la información necesaria para que la aplicación funcione se carga en el cliente durante la petición inicial y el acceso a diferentes páginas de la aplicación las maneja el **router**.

ROUTER

El router o ruteador dentro de una aplicación SPA, cumple la función de controlar e interpretar cada solicitud que el usuario haga en la página, para realizar un cambio en la URL mediante los hash (ejemplo: dominio.com/#seccion).

Ese cambio de URL implica realizar al servidor la solicitud de los datos necesarios para luego renderizar o dibujar mediante JS la nueva página que se mostrará.

Durante este proceso, el usuario no verá recargar la pestaña del navegador, sino que todo el proceso ocurre de manera asincrónica. Sólo notará que el contenido de la página web se modifica.

VENTAJAS DE APLICACIÓN SPA

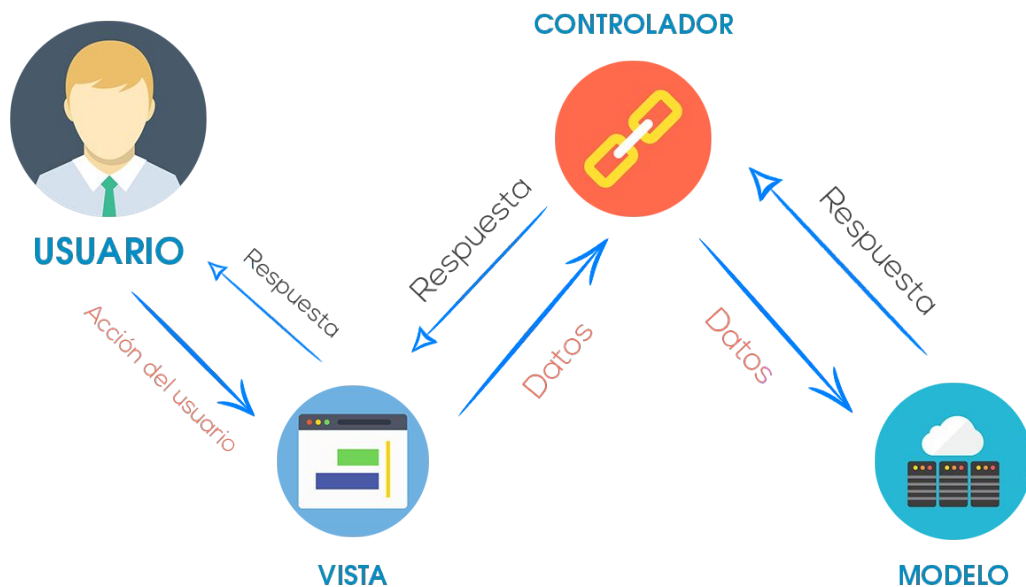
- Es más rápida y flexible.
- Mejor experiencia de usuario.
- Caché más poderoso.
- Simplicidad para implementar en servidores.
- Posibilidad de incorporar UI de Material Design.

MVC

MVC

En la metodología SPA debemos utilizar el patrón MVC (Modelo Vista Controlador). Este **separa la lógica de la aplicación, de la lógica de la vista en una aplicación.**

La mayoría de los frameworks modernos utilizan MVC (o alguna adaptación del MVC) para la arquitectura de aplicación. Algunos frameworks JS MVC son AngularJS, Ember y Backbone.



MODELO

Se encarga de los datos, generalmente (pero no obligatoriamente) consultando la base de datos. Actualizaciones, consultas, búsquedas, etc. todo eso va aquí, en el modelo.

```
class ProductoModel {  
  constructor() {  
    const productos = JSON.parse(localStorage.getItem('productos')) || [];  
    this.productos = productos.map(producto => new Producto(producto));  
  }  
  
  guardarProductos() {  
    localStorage.setItem('productos', JSON.stringify(this.productos));  
  }  
  
  agregarProducto(producto) {  
    this.productos.push(new Producto(producto));  
    this.guardarProductos();  
  }  
}
```

VISTA

Son la representación visual de los datos, todo lo que tenga que ver con la interfaz gráfica. Ni el modelo ni el controlador se preocupan de cómo se verán los datos.

```
class ProductoView {
  listarProductos(padre, data, callback) {
    let html = '';
    for (const producto of data) {
      html+=`<div>
        <input value=${producto.id} type="hidden">
        <h4> Producto:${producto.nombre}</h4>
        <b> ${producto.precio}</b>
        <button class="btnComprar">Comprar</button>
      </div>`;
    }
    $(padre).html(html);
    $(".btnComprar").click(callback);
  }
}
```

CONTROLADOR

Se encarga de controlar , es decir de definir el procesamiento principal en respuesta a eventos de usuario, solicitar los datos al modelo y comunicárselos a la vista.

```
class ProductoController {
  constructor(productoModel, productoView) {
    this.productoModel = productoModel;
    this.productoView = productoView;
    this.productoView.agregarProducto('#pag1', (event) =>{
      let hijos = $(event.target).parent().children();
      this.productoModel.agregarProducto({
        id: this.productoModel.productos.length + 1,
        nombre: hijos[1].value,
        precio: hijos[2].value,
      });
    });
  }
}

const app = new ProductoController(new ProductoModel(), new ProductoView());
```



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

***¡Ya estás llegando al fin de la cursada!
Recordá que luego de la corrección de tu proyecto
final, se notificará por Slack y email si quedaste en el
TOP10***

***No cuelgues, que tenés hasta 2 semanas desde que te
notificamos para solicitar los beneficios.***

¡5/10 MINUTOS Y VOLVEMOS!

CODER HOUSE

SPA: EJEMPLO

Ejemplo
en vivo



***¡VAMOS A PRACTICAR LO VISTO!
ESTA VEZ, TODOS A LA PAR***

CODER HOUSE

EJEMPLO ROUTING SPA

Vamos a ver parte por parte todos los pasos para armar una estructura SPA simple con routing para una web con 3 secciones. Como no estamos utilizando un servidor, utilizaremos información estática y HTML.

No usaremos AJAX para recuperar información del lado servidor, ya que no estamos trabajando con el backend. Esa instancia podría incorporarse al momento de dibujar el HTML.

Recomendamos utilizar la siguiente plantilla para el ejemplo:

[SPA PLANTILLA](#)


```
<!DOCTYPE html>
<html>
  <head>
    <title>EJEMPLO SPA</title>
  </head>
  <body>
    <h2>SPA CODER HOUSE</h2>
    <hr>
    <nav>
      <a href="/" class="button">Agregar Producto</a>
      <a href="#/pagina1" class="button">Ver Productos</a>
      <a href="#/pagina2" class="button">Buscar</a>
    </nav>
    <div id="notificacion"></div>
    <div id="app" style="height: 1000px;">
    </div>
    <!-- SCRIPTS JS -->
    <script src="js\jquery-3.5.1.js"></script>
    <script src="js\productos.js"></script>
    <script src="js\main.js"></script>
  </body>
</html>
```

PASO 1

Analizamos el HTML base donde vemos los links a cada página y un archivo main.js donde programaremos la lógica SPA. Empleamos además un script productos.js con las clases MVC

PASO 2

Dentro del archivo main.js instanciamos el controlador de productos y creamos las siguientes rutas.

```
const app = new ProductoController(new ProductoModel(), new ProductoView());  
// LISTA DE RUTAS (ASOCIAR A CADA ACCION)  
const routes = [  
  { path: '/', action: 'agregar' },  
  { path: '/pagina1', action: 'listar' },  
  { path: '/pagina2', action: 'buscar' },  
];
```

PASO 3

Analizamos el componente a mostrar cuando hay errores en productos.js.

```
const ErrorComponent = (padre) => {  
  $(padre).html("<h2>Error 404</h2>");  
}
```

PASO 4

Programamos una función para obtener el hash actual en el navegador.

```
//OBTENER LA RUTA ACTUAL (USAMOS EL OBJETO LOCATION Y SU PROPIEDAD HASH). SI "" || '/' ENTONCES parseLocation = '/'  
const parseLocation = () => location.hash.slice(1).toLowerCase() || '/';
```

PASO 5

Definimos una función para buscar la acción correspondiente en el array de rutas.
Luego creamos el router y llamamos a las funciones previamente codificadas.

```
//BUSCAMOS LA ACCIÓN EN EL ARRAY routes QUE CORRESPONDE A LA RUTA CON FIND
const findActionByPath = (path, routes) => routes.find(r => r.path == path || undefined);

const router = () => {
  //OBTENER RUTA ACTUAL
  const path = parseLocation();
  //OBTENER ACCIÓN ACTUAL
  const action = findActionByPath(path, routes).action;
};
```

```

const router = () => {
  //OBTENER RUTA ACTUAL
  const path = parseLocation();
  const { action = 'error' } = findActionByPath(path, routes) || {};
  // LLAMAMOS AL MÉTODO CORRESPONDIENTE PARA LA ACCIÓN ENCONTRADA
  switch (action) {
    case 'agregar':
      app.agregar('#app');
      break;
    case 'listar':
      app.listar('#app');
      break;
    case 'buscar':
      app.buscar('#app');
      break;
    default:
      ErrorComponent('#app')
      break;
  }
};

```

PASO 6

Empleamos la estructura switch para determinar qué comportamiento ejecuta el controlador según la acción de la ruta solicitada.

PASO 7

Por último, conectamos nuestra estructura al funcionamiento de la web, mediante los eventos load, y hashchange. De forma que la función router() se ejecute al cargar la página, y cada vez que se produce un cambio de Hash en la URL

```
//CADA VEZ QUE SE DETECTA LA CARGA DE LA VENTANA SE LLAMA A LA FUNCION ROUTER
$( window ).on( 'load', function() {
    router();
});

//CADA VEZ QUE SE DETECTA UN CAMBIO EN EL HASH (EJEMPLO la URL CAMBIA DE #/pagina1 a #/pagina2) SE LLAMA A LA FUNCION ROUTER
$( window ).on( 'hashchange', function() {
    router();
} );
```

PRÓXIMA CLASE: WORKSHOP FINAL

La próxima clase es la última, por lo que haremos un Workshop para que puedas mostrar el estado de avance de tu Proyecto Final. Es el momento de lucir ante todos tus compañeros, tutores y docente todo el trabajo del curso.

Te recomendamos presentar tu proyecto, no solo mostrarlo en funcionamiento, sino también "venderlo". Puedes usar una presentación, un video, o sólo un buen speech

Si te interesa exponer, avísale a tu tutor antes de terminar la clase de hoy.

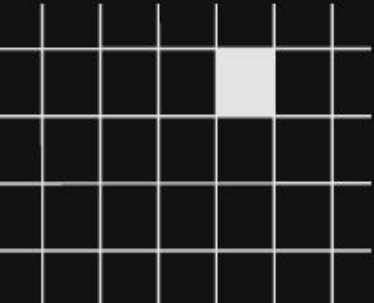
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Concepto de SPA.
 - Concepto de MVC.
 - Ejemplo de SPA.
- 



OPINA Y VALORA ESTA CLASE