UTN FACULTAD REGIONAL CÓRDOBA
INGENIERÍA EN SISTEMAS DE INFORMACIÓN

Desarrollo de Software
Notas de Clase

Clase 18 - Single Page Application

¿Qué es una SPA? (*)

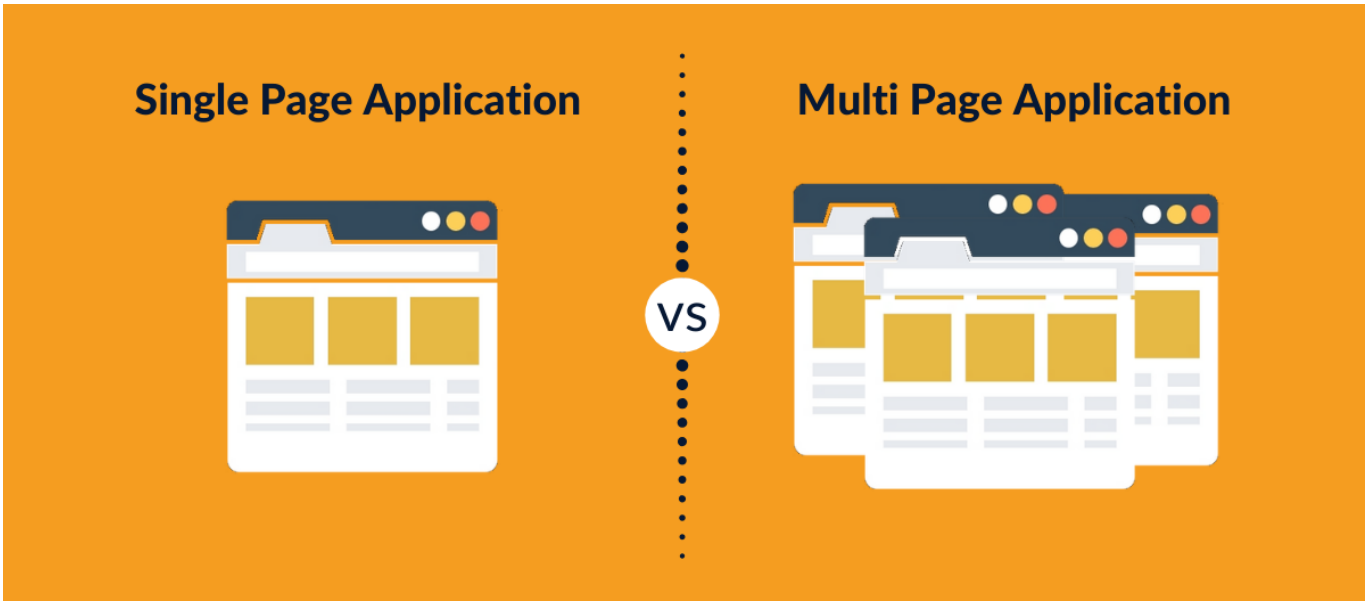
Una **SPA** o **Aplicación de una Sola Página** es una web que todo su contenido carga una sola vez, es decir, el navegador descarga un sólo archivo html con todos los recursos y dependencias que necesite para funcionar: estilos, scripts, imágenes, fuentes, etc.

Una vez que carga todo el contenido de la aplicación la navegación de esta suele ser muy rápida y fluida, pues el contenido ya ha sido previamente cargado y sólo se necesita intercambiar contenido a través de las interacciones del usuario.

Dependiendo de la complejidad y robustez de la aplicación también podría cargarse contenido de forma dinámica sin la necesidad de recargar la página mediante peticiones asíncronas con **AJAX** de tal manera que sólo se tiene que cargar el nuevo contenido.

Varias vistas, no varias páginas (*)

El hecho de tener una sola página, no implica no tener tener varias vistas, es decir los diferentes apartados que podría tener el sitio Web. A ojos del usuario es como si tuviese varias páginas que cargan muy rápido, pero en realidad son vistas en la misma página.



Entonces, si todo está en la misma página, *¿quiere decir esto que no cambia la URL del navegador?*

No, no quiere decir eso, es más, lo más normal es que vaya cambiando según nos movemos entre vistas. Pero la clave aquí, es que en realidad no estás cargando una página, tan solo sustituyendo el contenido de la misma.

Entonces, ¿para qué cambiar la URL?

Rutas en las SPA (*)

Por funcionalidad quizas NO sería necesario que cambiáse la url, pero por usabilidad es una muy buena opción que **si** cambie, ya que el navegador va guardando en su **historial** todas las páginas (en este caso todas las vistas) que el usuario va visitando, lo que podría permitir al usuario regresar o adelantar vistas a través de los botones de atrás y adelante de su navegador web.

También en necesario pensar en la posibilidad de que el usuario en vez de navegar por la aplicación para llegar a un contenido en particular, decida acceder a este a través de una url que decida escribir manualmente en la barra de dirección de su navegador web, si no se implementa un sistema de rutas esto podría ser un gran problema de accesibilidad en el contenido de la SPA.

Pero si el contenido se carga en una sóla página, **¿cómo cambiamos la url?**

Gracias al uso del hash de la url y el paso de parámetros en la misma.

Hash URL (*)

El final de una URL puede contener una palabra separada por un símbolo de almohadilla **(#)**. Esto es lo que se conoce como **elemento hash de la URL** y permite posicionarse dentro de una parte determinada de la página. Esto lo conseguimos de forma sencilla mediante HTML.

La URL **miweb.html#elementoA** se va a posicionar en la página en el elemento HTML que contenga el identificador **elementoA**.

Pero, **¿cómo podemos acceder al hash de la URL con Javascript?** Es decir, qué hacer si queremos controlar cuando el usuario llegue con una URL que contenga una etiqueta o elemento de hash.

La solución es bastante sencilla y es que en **JS** existe el objeto **location**, el cual hace referencia a la URL que maneja el navegador, este objeto facilita acceder al hash de la URL con Javascript.

En concreto la propiedad del objeto location llamada .hash devuelve el hash de la URL.

```
console.log(location.hash)
```

Para poder controlar los cambios del elemento hash de la URL, Javascript ofrece la posibilidad de manejar un evento de ventana llamado **hashchange**. Para registrar este evento se utiliza:

```
window.addEventListener("hashchange", funcion_manejador)
```

Combinando el elemento **hash** de la URL (más el paso de parámetros) con el evento **hashchange** del objeto window es posible definir una política de rutas (o de ruteo) de las SPAs.

¿En que lenguaje de programación se hacen las SPA?

Esto es fácil, las SPA siempre están hechas con **JavaScript**. No hay otro lenguaje, ya que este tipo de webs se ejecutan en el lado del cliente, es decir, en el navegador, y ahí solo se ejecuta JavaScript.

Pero si el código de una SPA es 100% frontend ¿qué pasa con el código backend que proporciona los datos?

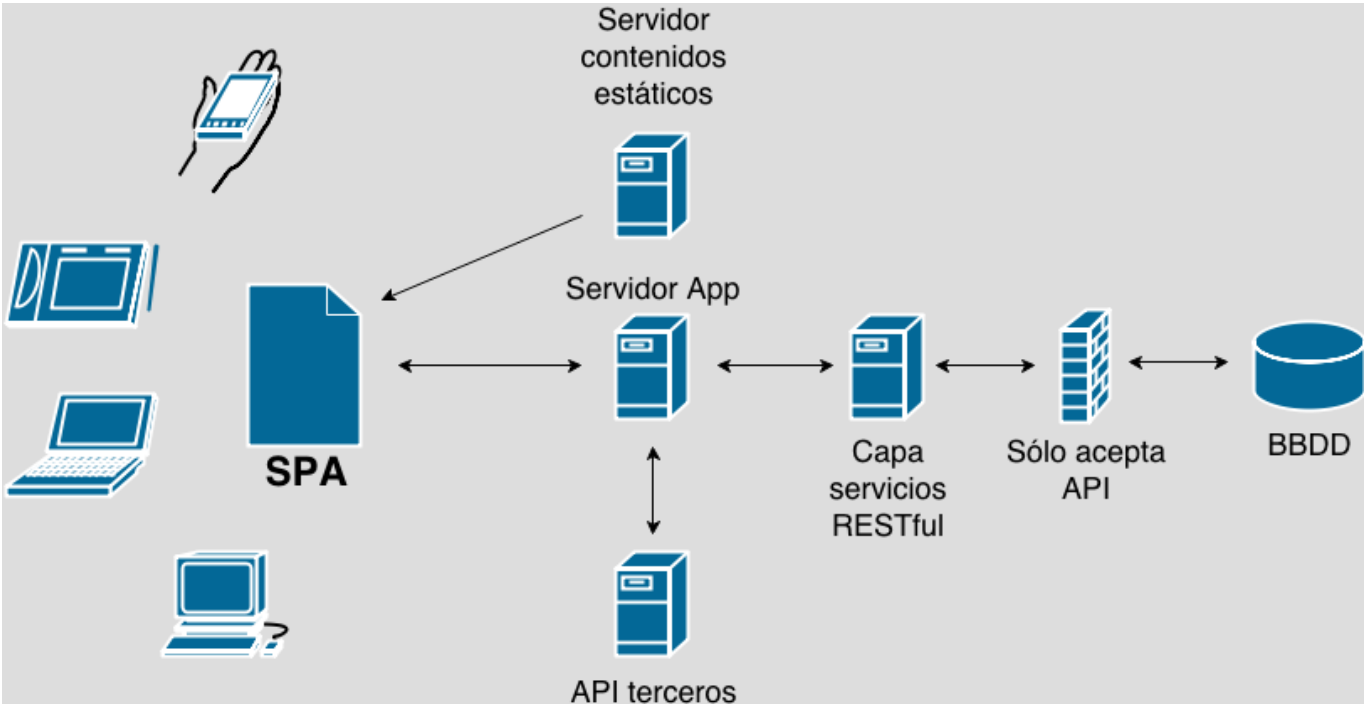
El código backend que provee los datos a una SPA, es 100% independiente de esta. De hecho a la SPA le importa poco en qué lenguaje este programado su backend o que reglas, patrones, estructuras o técnicas use internamente, mientras este le proporcione los datos en una API expuesta generalmente en formato JSON es todo lo que la SPA necesita de un backend.

Con lo anterior expuesto, significa que desarrollar un proyecto bajo el paradigma de las SPAs puedes programar de **forma independiente** el **frontend** del **backend** y sólo conectarlo a través de la API que los comuniqué entre sí.

De esta forma es posible mantener mayor control e independencia de desarrollo entre cada una de las partes de código (frontend y backend) de todo tu proyecto.

De hecho en los últimos años el paradigma de desarrollo de las SPAs le ha ido ganando terreno al patrón de desarrollo MVC (Modelo - Vista - Controlador) que fue muy popular en los inicios de la web, en este paradigma el frontend y backend estan mezclados y con mucha dependencia entre sí, además quien llevaba la mayor responsabilidad era el backend y generalmente con cada nueva acción en la aplicación implicaba volver a solicitar los datos al servidor lo que implicaba una recarga al navegador, y por ende una nueva petición por cada acción ejecutada.

Con la llegada de AJAX el tema de las recargas al servidor se solucionó, sin embargo la codependencia y mezcla entre código frontend y backend sigue ocurriendo en el paradigma MVC.



Actualmente con el avance que ha tenido JavaScript como lenguaje y la proliferación de librerías y frameworks que se basan en él, sumado a las nuevas formas de almacenar y consumir información en la web como el cómputo en la nube y el desarrollo de microservicios en la misma; el paradigma de las SPAs cada vez gana mayor aceptación frente al MVC, gracias a que permite desarrollar de forma autónoma e independiente cada parte de código de nuestra aplicación: frontend, backend y persistencia de datos desacoplados y descentralizados de sí mismos.

Angular, **React**, Ember.js, Polymer, etc, son ejemplos de frameworks y librerías que nos permiten construir SPAs de manera eficiente. En las semanas siguientes puntualmente abordaremos el uso de **React** para construir los prácticos de la asignatura.

¿Y el SEO en una SPA?

Algo importante a considerar es que la mayoria del contenido de una SPA se carga de forma dinámica, por lo que los mecanismos que tienen los buscadores para analizar el código HTML de las urls prácticamente son inservibles, pues en su mayoría, las SPAs tienen un sólo tag HTML vacío con un nombre de identificador y mediante JavaScript, el contenido es cargado asíncronamente a dicho tag.

Por lo anterior el código HTML de nuestra SPA estaría vacío ante los rastreadores de los buscadores, encargados de posicionar el contenido en la web. Esto último es lo que se conoce como SEO (Search Engine Optimization); se trata del conjunto de estrategias y técnicas de optimización que se hacen en una página web para que aparezca orgánicamente en buscadores de Internet como Google, Yahoo o Youtube. La correcta aplicación del SEO puede causar incrementos expresivos en el tráfico y la visibilidad de las marcas en Internet.

Si bien las SPAs no son tan amigables con el SEO, lo cierto es que en los últimos años los mismos buscadores han implementado mecanismos en sus rastreadores para detectar el código HTML generado por una SPA.

Adicionalmente existen 2 técnicas para volver más **SEO friendly** las SPAs:

- Los Generadores de Sitios Estáticos (SSG - Static Site Generators) y
- El Renderizado del Lado del Servidor (SSR - Server Side Rendering).

Dependerá del uso de la SPA para priorizar estas técnicas de renderizado.

Bibliografía o Referencia

- Aprende Javascript