**INTRODUCCION**

React es una librería. A diferencia de un framework puede integrarse desde un cdn. Fue desarrollada y es mantenida hasta el dia de hoy por la misma empresa, Facebook.

Está destinada únicamente al frontend, por tanto no se realiza backend con esta. Puede implementarse tanto como para SPA (simple page application) como para grandes proyectos

**SINGLE PAGE APPLICATION**

Este tipo de web se iniciaron para reemplazar a las MPA (multiple page application). La principal características es que las SPA no recargan constantemente toda la pagina para realizar cambios tanto en el back como en el front. Simplemente recargan cierta parte de la página en una “ventana” o en una porción con una animación amigable. Con esto se intenta no perder la atención del usuario y no generar la sensación de que “las cosas” están cambiando tan radicalmente.

En las SPA, las url van cambiando constantemente.

Estas aplicaciones interactúan principalmente con dos tipos de servicios: REST y Serverless.

En los rest, no importa el lenguaje o la arquitectura del backend, ya que “devuelve” el contenido en formato JSON, lo que lo hace muy fácil de interpretar y operar al frontend.

En los serverless, como firebase (db no relacional), se conecta a estos servicios y proveen el backend y la db. Es más “simple” que utilizar o crear un servicio REST propios. En proyectos muy grandes se recomienda utilizar un framework o crear el servicio REST.

**REST**

(Representational State Transfer- Transferencia de Estado Representacional)

Buscando una definición sencilla, REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST.

Características de REST

● Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.

● Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

● Los objetos en REST siempre se manipulan a partir de la URI. Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

● Interfaz uniforme: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

● Sistema de capas: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

● Uso de hipermedios: hipermedia es un término acuñado por Ted Nelson en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Para cualquier API REST es obligatorio disponer del principio HATEOAS (Hypermedia As The Engine Of Application State – Hipermedia Como Motor del Estado de la Aplicación) para ser una verdadera API REST. Este principio es el que define que cada vez que se hace una petición al servidor y éste devuelve una respuesta, parte de la información que contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente.

**Qué es el Serverless**

Aunque significa «sin servidor» esto no es así, lo que ocurre es que dejas de usar un servidor físico o uno en la nube claramente identificados por unos contenedores temporales y sin estado donde se ejecutan los códigos de las aplicaciones. Estos contenedores se crean en el momento que ejecutas la aplicación y luego desaparecen, por lo que el servidor pasa a ser una parte menos visible del sistema.  
Ya no tienes que hacer mantenimiento de los servidores donde tienes instalados todos los programas y aplicaciones. El código se ejecuta en un contenedor temporal, ya no vas a necesitar instalar software, gestionar puertos de acceso o estar pendiente de las actualizaciones.

El sistema lo puedes escalar de manera horizontal todo lo que lo necesites. Puedes añadir todos los clusters, balanceo de cargas… conforme lo vayas necesitando.

Solamente vas a pagar por el tiempo que estés usando el proceso.

Las funciones que utilizas las puedes integrar con el resto de servicios que ofrece la plataforma, como son logging, virtualización o crear endpoints.

**JAVASCRIPT O TYPESCRIPT**

La principal ventaja de React sobre Angular es que angular trabaja sobre typescript. En cambio, sabiendo javascript,se puede comenzar a operar con react. Sin embargo es importante saber que React no trabaja 100% sobre JS sino que interpreta JSX, que es “una forma distinta de interpretar JS”.

**JSX**

Es una “mezcla” de JS y XML. Con esto es posible crear “nuestras propias etiquetas” (componentes).  
¨Para interpretar los archivos, React usa otra librería llamada Babel.  
Create-react-app es un soft que autogenera una base para hacer react.

**INSTALACIONES**

Para comprobar la versión de node se puede utilizar en el símbolo de sistema el comando: **node -v**

Para comprobar npm, **npm -v**

YARN – es un gestor de dependencias que puede ayudar con estas instalaciones

**EXTENSIONES DE VS**

* CSS snippets
* ES7 React/Redux/GraphQL/React-Native snippets
* gruvbox ish
* HTML CSS SUPPORT
* Para implementar algunos símbolos: en **settings.json**, **“editor.fontLigatures”:true,**  despues de editor.wordWap y antes de workbench.startupEditor