



INSTITUTO DE EDUCACIÓN SECUNDARIA CAMPANILLAS
TÉCNICO SUPERIOR DE DESARROLLO DE APLICACIONES WEB

TRABAJO FINAL DE GRADO

SAMERY

Realizado por
Mohamed Chahdi Dkikar

Tutorizado por
Equipo Educativo 2DAW

DEPARTAMENTO DE FAMILIA PROFESIONAL DE
INFORMÁTICA Y COMUNICACIONES



MÁLAGA, JUNIO DE 2021

ÍNDICE

[ÍNDICE](#)

[ACERCA DE SAMERY](#)

[TECNOLOGÍAS UTILIZADAS](#)

[ReactJS](#)

[Firebase](#)

[HTML](#)

[CSS](#)

[JavaScript](#)

[SASS](#)

[NodeJS](#)

[HOSTING](#)

ACERCA DE SAMERY

Samery es una biblioteca que me permite gestionar entre un amplio catálogo de películas a tiempo real cuales son las que tengo planeadas para ver y cuales he visto. Los usuarios además pueden modificar sus usuarios.

TECNOLOGÍAS UTILIZADAS

He desarrollado Samery haciendo uso de ReactJS y Firebase y utilizando una API llamada TMDB.

También he utilizado HTML5, SASS (CSS), JavaScript, NodeJS...

ReactJS

React te ayuda a crear interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en tu aplicación, y React se encargará de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien.

Las vistas declarativas hacen que tu código sea más predecible, por lo tanto, fácil de depurar.

Crea componentes encapsulados que manejen su propio estado, y conviértelos en interfaces de usuario complejas.

Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, puedes pasar datos de forma sencilla a través de tu aplicación y mantener el estado fuera del DOM.

React puede también renderizar desde el servidor usando Node, así como potencializar aplicaciones móviles usando [React Native](#).

Los componentes de React implementan un método llamado `render()` que recibe datos de entrada y retorna qué mostrar.

Además de obtener datos de entrada (a los que accedes a través de `this.props`), un componente puede tener datos en su estado interno (a los que accedes a través de

this.state). Cuando los datos del estado de un componente cambian, se vuelve a invocar render con los nuevos valores en this.state.

Firestore

¿Qué es Firestore?

Firestore de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móvil. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo.

Aunque fue creada en 2011 pasó a ser parte de Google en 2014, comenzando como una base de datos en tiempo real. Sin embargo, se añadieron más y más funciones que, en parte, permitieron agrupar los SDK de productos de Google con distintos fines, facilitando su uso.

¿Para qué sirve Firestore?

Su función esencial es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.

Sus herramientas son variadas y de fácil uso, considerando que su agrupación simplifica las tareas de gestión a una misma plataforma. Las finalidades de las mismas se pueden dividir en cuatro grupos: desarrollo, crecimiento, monetización y análisis. Es especialmente interesante para que los desarrolladores no necesiten dedicarle tanto tiempo al backend, tanto en cuestiones de desarrollo como de mantenimiento.

Tal vez estés buscando en internet: tutorial Firestore, manual firestore o dev firestore, pero la mejor forma de entender en qué consiste Firestore es hacernos una idea de las herramientas que ofrece y a qué están destinadas, algo que veremos a continuación.


Funciones de Firestore

Firestore dispone de diferentes funcionalidades, que se pueden dividir básicamente en 3 grupos: Desarrollo (Develop), Crecimiento (Grow) y Monetización (Earn), a los que hay que sumar la Analítica (Analytics).

DESARROLLO

El primer grupo de funciones es conocido como Desarrollo o Develop en Firestore. Como su nombre indica, incluye los servicios necesarios para el desarrollo de un proyecto de aplicación móvil o web. Estos contribuyen a que el proceso sea más rápido, puesto que se dejan determinadas actividades a mano de Firestore, mientras que otras permiten optimizar diversos aspectos para conseguir la calidad deseada.

Realtime database



Una de las herramientas más destacadas y esenciales de Firebase son las bases de datos en tiempo real. Estas se alojan en la nube, son No SQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real, manteniéndolos actualizados aunque el usuario no realice ninguna acción.

Firebase envía automáticamente eventos a las aplicaciones cuando los datos cambian, almacenando los datos nuevos en el disco. Aunque no hubiera conexión por parte de un usuario, sus datos estarían disponibles para el resto y los cambios realizados se sincronizarían una vez restablecida la conexión.

Autenticación de usuarios

La identificación de los usuarios de una app es necesaria en la mayoría de los casos si estos quieren acceder a todas sus características.

Firebase ofrece un sistema de autenticación que permite tanto el registro propiamente dicho (mediante email y contraseña) como el acceso utilizando perfiles de otras plataformas externas (por ejemplo, de Facebook, Google o Twitter), una alternativa muy cómoda para usuarios reacios a completar el proceso.

Así, este tipo de tareas se ven simplificadas, considerando también que desde aquí se gestionan los accesos y se consigue una mayor seguridad y protección de los datos. Se debe mencionar que Firebase puede guardar en la nube los datos de inicio de sesión con total seguridad, evitando que una persona tenga que identificarse cada vez que abra la aplicación.

Almacenamiento en la nube

Firebase cuenta con un sistema de almacenamiento, donde los desarrolladores pueden guardar los ficheros de sus aplicaciones (y vinculándolos con referencias a un árbol de ficheros para mejorar el rendimiento de la app) y sincronizarlos. Al igual que la mayoría de herramientas de Firebase, es personalizable mediante determinadas reglas.

Este almacenamiento es de gran ayuda para tratar archivos de los usuarios (por ejemplo, fotografías que hayan subido), que se pueden servir de forma más rápida y fácil. También hace la descarga de referencias a ficheros más segura.

Hosting

Firebase también ofrece un servidor para alojar las apps de manera rápida y sencilla, esto es, un hosting estático y seguro. Proporciona certificados de seguridad SSL y HTTP2 de forma automática y gratuita para cada dominio, reafirmando la seguridad en la navegación. Funciona situándolas en el CDN (Content Delivery Network) de Firebase, una red que recibe los archivos subidos y permite entregar el contenido.

HTML

HTML es el lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, etc.

El HTML se creó en un principio con objetivos divulgativos de información con texto y algunas imágenes. No se pensó que llegara a ser utilizado para crear área de ocio y consulta con carácter multimedia (lo que es actualmente la web), de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, si que se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML. Numerosos estándares se han presentado ya. El HTML 4.01 es el último estándar a febrero de 2001. Actualización a mayo de 2005, en estos momentos está apunto de presentarse la versión 5 de HTML, de la que ya se tiene un borrador casi definitivo.


El HTML es un lenguaje de marcación de elementos para la creación de documentos hipertexto, muy fácil de aprender, lo que permite que cualquier persona, aunque no haya programado en la vida, pueda enfrentarse a la tarea de crear una web. HTML es fácil y pronto podremos dominar el lenguaje. Más adelante se conseguirán los resultados profesionales gracias a nuestras capacidades para el diseño y nuestra vena artista, así como a la incorporación de otros lenguajes para definir el formato con el que se tienen que presentar las webs, como CSS.

CSS

Antes de comenzar, debes tener claro un concepto clave: una página web es realmente un documento de texto. En dicho documento se escribe código HTML, con el que se crea el contenido de una web. Por otro lado, existe el código CSS, que unido al código HTML permite darle forma, color, posición (y otras características visuales) a una página.

En resumen, se trata de un idioma como podría ser el inglés o el alemán, que los navegadores web como Chrome o Firefox conocen y pueden entender. Nuestro objetivo como diseñadores y programadores web es precisamente ese: aprender el idioma.

Las siglas CSS (Cascading Style Sheets) significan «Hojas de estilo en cascada» y parten de un concepto simple pero muy potente: aplicar estilos (colores, formas, márgenes, etc...) a uno o varios documentos (generalmente documentos HTML, páginas webs) de forma masiva.



Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia que trataremos más adelante) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan sólo información y datos, relativos al significado de la información a transmitir (el contenido), y todos los aspectos relacionados con el estilo (diseño, colores, formas, etc...) se encuentren en un documento CSS independiente (la presentación):

De esta forma, se puede unificar todo lo relativo al diseño visual en un solo documento CSS, y con ello, varias ventajas:

Si necesitamos hacer modificaciones visuales lo hacemos en un sólo lugar. No necesitamos editar todo el HTML en cuestión por separado.

Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil de organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).


Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, etc...

JavaScript

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje.

SASS

Los preprocesadores CSS son herramientas para los desarrolladores de sitios web, que permiten traducir un código de hojas de estilo no estándar, específico del preprocesador en cuestión, a un código CSS estándar, entendible por los navegadores.



Los preprocesadores básicamente nos ofrecen diversas utilidades que a día de hoy no se encuentran en el lenguaje CSS, o bien no son compatibles con todos los navegadores. Ejemplos pueden ser variables, anidación de selectores, funciones (denominadas mixins), etc.

Al desarrollar con un preprocesador se consigue principalmente un ahorro de tiempo, ya que tenemos que escribir menos código para hacer las cosas. Pero también conseguimos una mayor facilidad de mantenimiento del código, gracias a una mejor organización del código y la posibilidad de editar una vez ciertos valores y que afecten a decenas, o cientos, de lugares del código CSS generado.

NodeJS

Node.js es un entorno JavaScript que nos permite ejecutar en el servidor, de manera asíncrona, con una arquitectura orientada a eventos y basado en el motor V8 de Google. Es una plataforma que avanza muy rápidamente y cada vez está más presente en el mercado.

HOSTING

1

Instala Firebase CLI

Con [Firebase CLI](#), es fácil configurar un proyecto nuevo de Hosting, ejecutar un servidor de desarrollo local y también implementar contenido.

2

Configura un directorio de proyecto

Agrega los elementos estáticos al directorio de un proyecto local y, luego, ejecuta **firebase init** para conectar el directorio a un proyecto de Firebase.

En el directorio de tu proyecto local, también puedes configurar Cloud Functions o Cloud Run para tu [contenido dinámico y microservicios](#).

3

Visualiza, prueba y comparte los cambios antes de publicarlos (*opcional*).

Ejecuta **firebase emulators:start** para emular Hosting y los recursos de tu proyecto de backend en una URL alojada localmente.

Para ver y compartir los cambios en una URL de vista previa temporal, ejecuta **firebase hosting:channel:deploy** a fin de crear e implementar en un canal de vista previa. Configura la [integración en GitHub](#) para realizar iteraciones sencillas de tu contenido de vista previa.

4

Implementa tu sitio

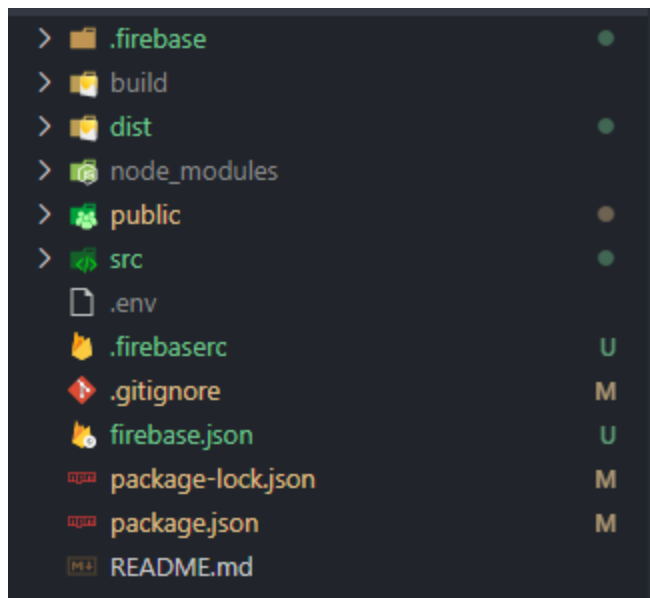
Cuando estés satisfecho con la configuración, ejecuta **firebase deploy** para subir la instantánea más reciente a nuestros servidores. Si necesitas deshacer la implementación, puedes revertirla con un solo clic en Firebase console.

5

Vincula el sitio a una aplicación web de Firebase (*opcional*)

Cuando vinculas tu sitio a una [aplicación web de Firebase](#), puedes usar [Google Analytics](#) para recopilar datos de uso y comportamiento de tu app y [Firebase Performance Monitoring](#) para obtener estadísticas sobre las características de su rendimiento.

ESTRUCTURA DEL PROYECTO



En la carpeta de .firebase se guardan las keys para el hosting.

Las carpetas de build y dist son las carpetas utilizadas a la hora de desplegar la aplicación y que necesita el servidor de hosting.

La carpeta de node_modules contiene los paquetes y dependencias necesarias de npm.

En public tenemos el index.html, favicon, logo,...

En src es la carpeta en la cual tendremos todo nuestro código fuente.

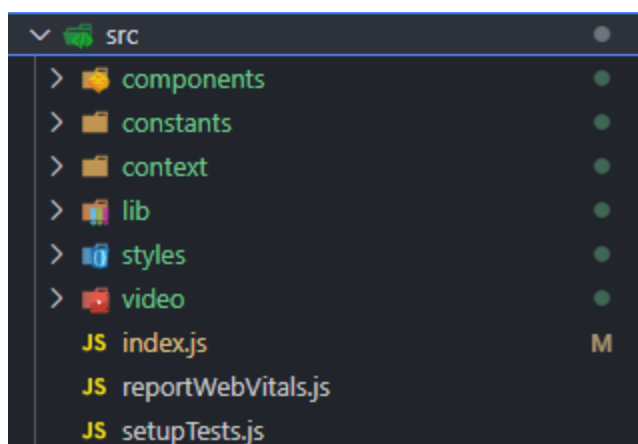
En el archivo .env tendremos las credenciales de las API utilizadas en la aplicación.

El archivo .firebaserc y firebase.json son archivos autogenerados por firebase para guardar los parámetros necesarios relacionados con hosting y rutas.

Los package.json contienen información acerca de las dependencias usadas en el proyecto.

Y por último el archivo README, contiene la información de la aplicación que se visualizará en Github.

Ahora me centraré más a fondo en la carpeta src, la cual contiene todo el código que he programado.



En components guardo todos los componentes de los que dispondrá la aplicación.

En constants guardo todas las constantes de las que dependerá la aplicación.

En context creo las funciones para el manejo de las películas que podrán ser usadas de manera global en la aplicación gracias a la ayuda de React Context.

En la carpeta lib guardo los archivos de font-awesome con los iconos que usaré.

En la carpeta de styles guardo todos los estilos de la aplicación.

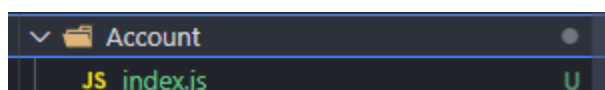
En la carpeta de video guardo todos los videos de la aplicación.

El archivo index.js es el corazón de la aplicación, es el archivo que será ejecutado y del cual se irán llamando al resto.

reportWebVitals.js te permite analizar y medir la performance de la aplicación.

setupTest.js te permite hacer test.

Dentro de components tenemos las siguientes carpetas con sus respectivos archivos:



```
import React from 'react'

import { PasswordForgetForm } from '../PasswordForget'
import PasswordChangeForm from '../PasswordChange'
import { AuthUserContext, withAuthorization } from '../Session'

const AccountPage = () => (
  <AuthUserContext.Consumer>
    {authUser => (
      <div>
        <h1>Account: {authUser.email.split('@', 1)}</h1>
        <PasswordForgetForm />
        <PasswordChangeForm />
      </div>
    )}
  </AuthUserContext.Consumer>
)

const condition = authUser => !!authUser

export default withAuthorization(condition)(AccountPage)
```