Clase 20: Material Complementario

Sitio:	Centro de E-Learning - UTN.BA	Imprimido por:	Nilo Crespi
Curso:	Curso de Backend Developer - Turno Noche	Día:	Friday, 23 de January de 2026, 10:36
Libro:	Clase 20: Material Complementario		

Tabla de contenidos

1. Firebase Parte 1

Bloques temáticos:

- ¿Qué es Firebase?
- Características
- Ventajas de utilizar firebase
- Crear un proyecto en firebase
- Crear base de datos realtime
- Conectar nuestra app react con firebase
- React Bootstrap

1.1. Firebase Parte 1

¿Qué es Firebase?

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2011 y adquirida por Google en 2014

En 2011, James Tamplin y Andew Lee fundaron una startup llamada Envolve. Esta plataforma proporcionó a los desarrolladores una API que permitía la integración de sistemas de chat en sus páginas web.

Tras lanzar el servicio chat, Tampin y Lee observaron que dicho chat estaba siendo utilizado en gran escala por los desarrolladores para pasar paquetes de información de sus aplicaciones, como el estado de las partidas en el caso de los juegos. Fue entonces cuando decidieron separar ambas funcionalidades, el sistema de chat y el sistema de arquitectura en tiempo real que lo propulsaba, dando como resultado la fundación de Firebase en Abril de 2012.

En octubre de 2014, Firebase fue comprado por Google. En octubre de 2015, Google adquirió Divshot para fusionar su equipo con el equipo de Firebase. Desde su adquisición, Firebase ha crecido dentro de Google y ha expandido sus servicios para convertirse en una plataforma unificada para desarrolladores móviles. Firebase se integra actualmente con otros servicios de Google para poder ofrecer productos a mayor escala para los desarrolladores. En enero 2017, Google adquirió Tejido y Crashlytics de Twitter para unir sus servicios al equipo de Firebase. En octubre de 2017 Firebase lanzó Cloud Firestore, una base de datos NoSQL en la nube.

Características

Firebase nos brinda las siguientes caracteristicas:

- **Analíticas:** Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.
- **Desarrollo:** Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firebase, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.
- **Crecimiento:** Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.

- **Monetización:** Permite ganar dinero gracias a AdMob.

Base de datos realtime

Con la base en tiempo real de Firebase podrás guardar todos los datos que requiera tu aplicación. Se lleva muy bien con React y su patrón reactivo que permite actualizar los datos en los componentes automáticamente. Los datos se almacenan en formato JSON y se pueden agregar reglas para permitir requests con token o solo desde una URL por ejemplo.

Autenticación

Es un servicio que nos simplifica el inicio de sesión y la gestión de la misma en nuestra aplicación. Si la usamos en aplicaciones web es extremadamente fácil de configurar, sobretodo si usamos el proveedor de Google, aún así si usamos otros de los disponibles (Correo/Contraseña, Teléfono, Facebook, Twitter, GitHub, Anónimo) también es muy fácil, sólo es un paso más en el caso de las redes.

Almacenamiento

Este servicio es muy bueno para aplicaciones que requieran guardar archivos del usuario. También nos sirve si queremos subir estáticos ya que existe un botón desde la interfaz o podemos programar algo. En mi caso lo he usado para subir imágenes desde un formulario y no he tenido ningún tipo de problemas. Como la base de datos, tiene reglas que podemos configurar.

Hosting

Este servicio es uno de mis favoritos. Con una colección de estáticos (o de archivos que han pasado ya el proceso de build) podemos subir una aplicación y esta automáticamente contará con SSL y HTTP2. Si tenemos una app con Angular o Firebase podemos hacer un build desde nuestros ordenadores y subir estos archivos generados y nos funcionarán sin problemas. Si necesitamos un Backend tendríamos que subirlo a otro lugar o hacer uso de las Cloud Functions.

Notificaciones

Este servicio, como su nombre nos indica, nos permitirá gestionar el envío de notificaciones a nuestros usuarios con la diferencia de que estas podrán ser programadas acorde a diferentes parámetros.

Planes

Firebase no es un entorno de desarrollo gratuito, pero si brinda sus servicios de forma gratuita con ciertos límites.

Para ver los mismos podes visitar https://firebase.google.com/pricing/?hl=es-419

Ventajas de utilizar firebase

1. **Rapidez**

Algunas de las funcionalidades de esta plataforma que nos ayudan a optimizar el tiempo dedicado al desarrollo y optimización de nuestra app son, por ejemplo, la función para detectar errores. Podemos, además, almacenar todo en la Nube, testear la app o configurarla de manera remota.

2. **Olvídate de la infraestructura**

El desarrollar una app puede convertirse en una tediosa tarea debido a la complejidad de su estructura. Otro de los motivos que tenemos, por tanto, para utilizar Firebase es que nos podemos olvidar de eso. Implementar Firebase es rápido y fácil. Gracias a API intuitivas contenidas en un solo SDK, puedes concentrarte en resolver los problemas de tus clientes y evitar perder tiempo en crear una infraestructura compleja.

3. **Analítica**

Firebase nos ayuda a tomar decisiones inteligentes basadas en los datos. Firebase Analytics es la solución de análisis gratuita e ilimitada directamente integrada a Firebase. Puedes obtener gran cantidad de información sobre tus usuarios, desde los clics en anuncios hasta el uso de la app. Firebase Analytics funciona con otras características de Firebase para que puedas controlarlo todo, desde la tasa de clics hasta los fallos de la app.

4. **Multiplataforma**

Ya lo avisamos anteriormente. Con Firebase puedes ofrecer apps multiplataforma con API integradas a SDK individuales para Android, iOS y JavaScript.

5. **Notificaciones**

Con esta plataforma también podrás gestionar la programación y envío de notificaciones push a los usuarios de tus apps, algo realmente útil para mantenerlos al día de las últimas novedades… y que no se olviden de ella.

6. **Inicio gratuito y proyecto escalable**

La mayoría de las funciones de Firebase son de uso gratuito siempre, para cualquier tamaño. Además, cuando nuestra app sea un éxito, no tendremos que preocuparnos por el escalamiento del código de nuestro servidor ni por ofrecer capacidad adicional ya que Firebase se encargará de eso.

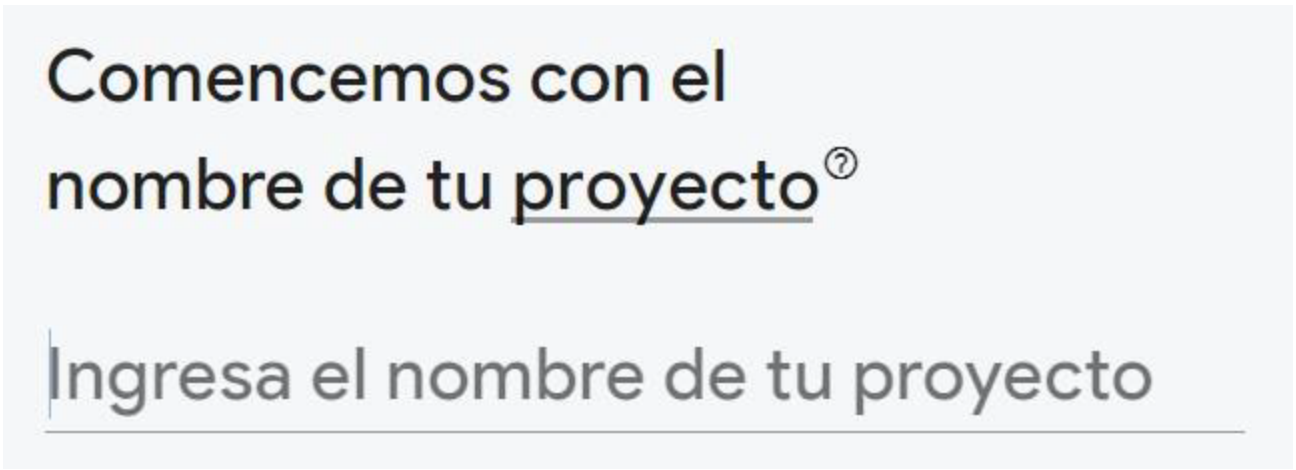
7. **Soporte gratuito**

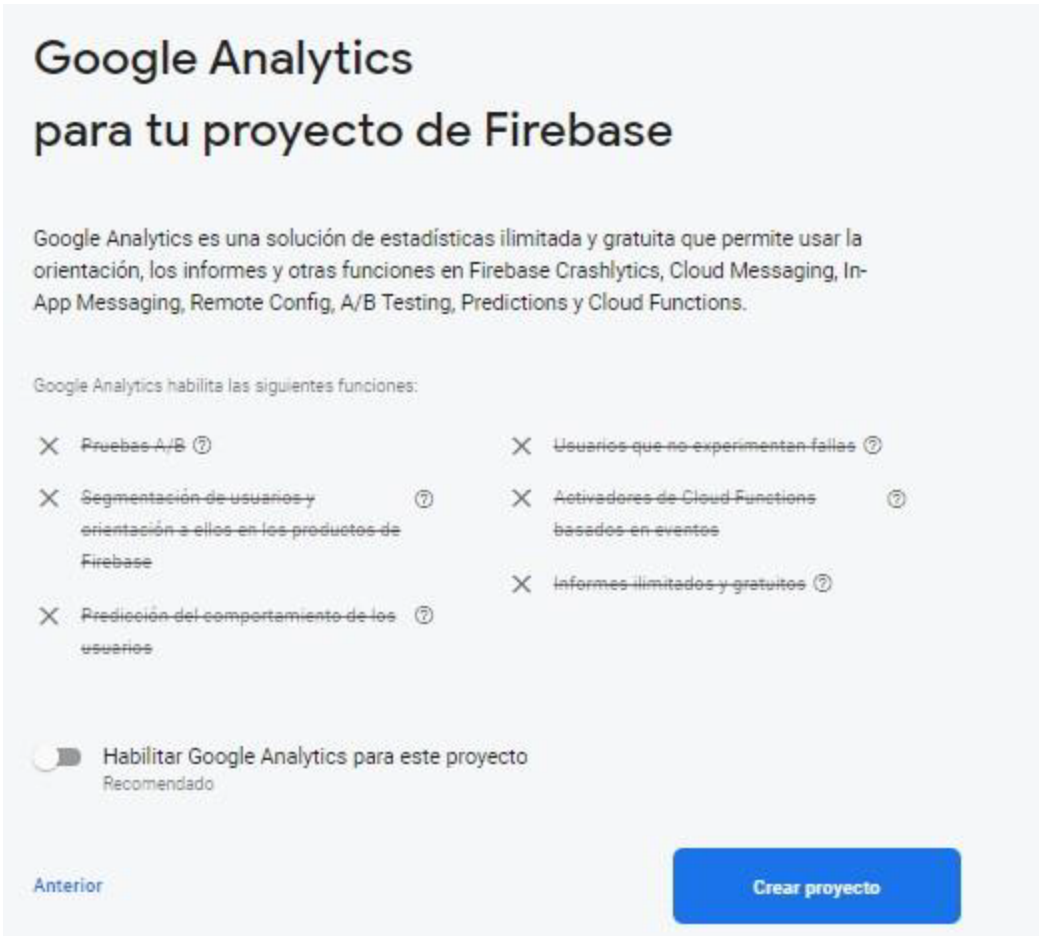
¡Cómo no! Google también nos ofrece soporte gratuito por correo electrónico a todos los desarrolladores mientras que el equipo de Firebase y expertos en desarrollo de Google participan activamente en comunidades en línea como Stack Overflow y GitHub.

Crear un proyecto en firebase

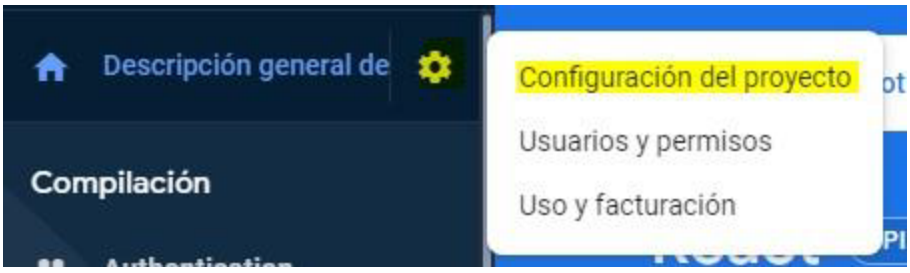
Debes ir a la nueva consola de firebase (https://console.firebase.google.com/) y crear un proyecto nuevo:

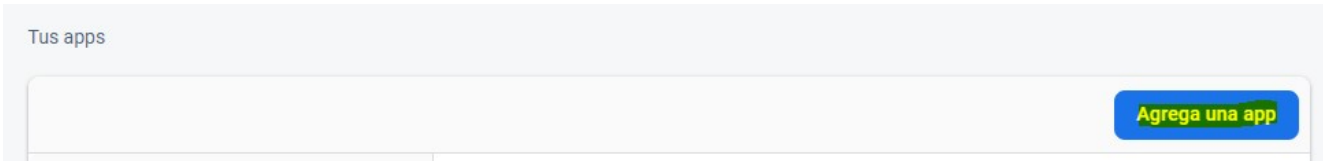
Nos solicitara los siguientes datos:

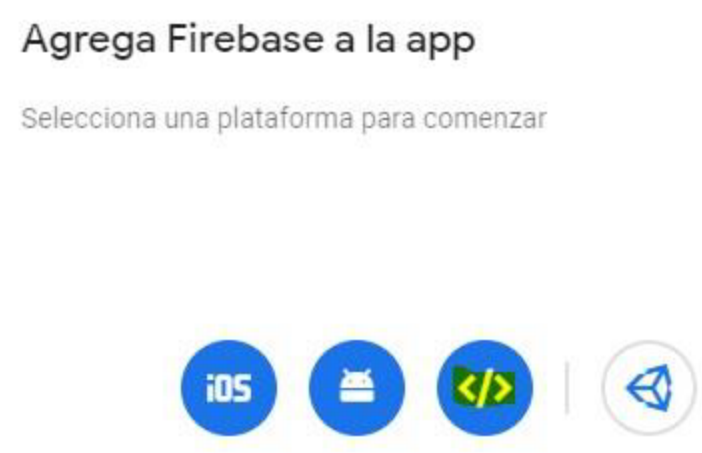




Agregar firebase como aplicación web:







Nos brinda un código el cual deberemos guardar para integrarlo con nuestra aplicación:

Crear base de datos firestore

Vamos al menú **firestore database**

Seleccionamos **comenzar en modo prueba**

Se nos abrirá el siguiente menú de base de datos:

En dicho menú podemos ver los datos, es decir los datos que están almacenados en la misma.

También podemos ver las reglas, es decir podremos limitar el acceso a nuestros datos.

Conectar nuestra app react con firebase

Debemos ejecutar el comando

npm install --save firebase

Copiamos el código provisto por Firebase (guardado en pasos anteriores) en nuestro componente app.

En el constructor de nuestro componente haremos un

console.log(firebase.database()),

el código quedaría de la siguiente manera:

```
import * as firebase from 'firebase'
// Initialize Firebase
var config = {
  apiKey: "AIzaSyDTBifG9hGKRqTfnY9TWx0iJv_9jHpop_0",
  authDomain: "react-74ab8.firebaseapp.com",
  databaseURL: "https://react-74ab8.firebaseio.com",
  projectId: "react-74ab8",
  storageBucket: "react-74ab8.appspot.com",
  messagingSenderId: "306351405488"
};
firebase.initializeApp(config);


class App extends Component {
  constructor () {
    super()
    console.log(firebase.database());
  }
```

React Bootstrap

React Bootstrap es una biblioteca UI basada en Bootstrap para react. La documentación de la misma se encuentra en:

https://react-bootstrap.github.io/

- Desde la consola, parada en la raíz del proyecto debemos ejecutar

```
npm install react-bootstrap@next bootstrap@5.1.0
```

- Agregar la hoja de estilo en index.html

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.
css"
integrity="sha384BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+Pm
STsz/K68vbdEjh4u" crossorigin="anonymous">
```

- Luego se debe importar el componente que se quiere utilizar, por ejemplo si en nuestro componente “Registro” queremos utilizar el componente Button debemos importar el mismo

```
import Button from 'react-bootstrap/Button'
```

Se puede ver el listado completo de componentes que provee Bootstrap en el siguiente enlace:

https://react-bootstrap.github.io/components/alerts

Bibliografía utilizada y sugerida

Fedoseyev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt. https://openwebinars.net/blog/que-es-firebase-de-google/ https://marketing4ecommerce.net/ecommerce/ https://carlosazaustre.com/conectando-firebase-a-react/ https://console.firebase.google.com

2. FIREBASE 2 Desarrollar nuestra app en React JS

Bloques temáticos:

- Renderizado condicional
- Listas
- Key

2.1. FIREBASE 2 Desarrollar nuestra app en React JS

FIREBASE 2 Desarrollar nuestra app en React JS

Renderizado condicional

En react podemos colocar condicionales al momento de realizar el renderizado.

Una forma de hacerlo es la siguiente:

```
render() {
    if (true) {
        return (
            <div>
                <h1>{this.state.nombre}</h1>
            </div>
        );
    } else {
        return (
            <div>
                <h1>{this.state.nombre} false</h1>
            </div>
        );
    }
}
```

Como podemos observar, dentro del método render colocamos el condicional en el cual colocamos un return dentro de cada posible camino a tomar.

Otra opción es manejar la condición a través de un estado, por ejemplo:

```
render() {
  const isLoggedIn = this.state.isLoggedIn;
  let button;
  if (isLoggedIn) {
    button = <LogoutButton onClick={this.handleLogoutClick} />;
  } else {
    button = <LoginButton onClick={this.handleLoginClick} />
  }
  return (
    <div>
      <Greeting isLoggedIn={isLoggedIn} />
```

```
        {button}
      </div>
    );
}
```

En este caso vemos el estado isLoggedIn y la condición de acuerdo a ese valor.

Como vemos podemos asignar código JSX a una variable Javascript y luego renderizar el contenido de esa variable.

React nos permite anidar la condición con el operador &&, veamos el siguiente ejemplo:

```
render() {
    const isLoggedIn = this.state.isLoggedIn;
    let button;
    if (isLoggedIn) {
        button = <LogoutButton onClick={this.handleLogoutClick} />;
    } else {
        button = <LoginButton onClick={this.handleLoginClick} />
    }
    return (
        <div>
            <Greeting isLoggedIn={isLoggedIn} />
            {button}
        </div>
    );
}
```

Vemos que la primer parte (**unreadMessages.length** > 0) es la condición, si la misma es true se renderiza el contenido JSX que es lo que vemos a la derecha del &&

Por último también podemos utilizar el operador ternario:

```
return (
    <div>
        <h1>Hello!</h1>
        {unreadMessages.length > 0 &&
            <h2>
                You have {unreadMessages.length} unread messages.
            </h2>
        }
    </div>
);
```

Listas

Las listas o colecciones (arrays) las podemos iterar utilizando la función map (vista anteriormente)

Por ejemplo:

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
    <li>{number}</li>
);
```

Como vemos la función map itera la lista numbers y mediante esta iteración renderiza el contenido de cada componente del array.

Keys

La key es un helper de react que nos permite saber que item ha cambiado, se ha eliminado o se ha agregado.

Siguiendo el ejemplo anterior podemos ver la combinación de este helper con el método map

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
    <li key={number.toString()}>
        {number}
    </li >
);
```

La mejor práctica es utilizar como key un atributo unívoco, por ejemplo un id:

```
const todoltems = todos.map((todo) =>
    <li key={todo.id}>
        {todo.text}
    </li>);
```

En caso de no tener un id unívoco para conformar la key, podemos utilizar el index (aunque no es la mejor práctica o recomendada)

```
const todoItems = todos.map((todo, index) =>
    // Only do this if items have no stable IDs
    <li key={index}>
        {todo.text}
    </li>
);
```

En caso de recorrer 2 arrays (que tienen el mismo contenido pero se renderizan 2 veces) podemos utilizar la misma key, por ejemplo:

```
function Blog(props) {
    const sidebar = (
        <ul>
            {props.posts.map((post) =>
                <li key={post.id}>
                    {post.title}
                </li>
            )}
        </ul>
    );
    const content = props.posts.map((post) =>
        <div key={post.id}>
            <h3>{post.title}</h3>
            <p>{post.content}</p>
        </div>
    );
    return (
        <div>
            {sidebar}
            <hr />
            {content}
        </div>);
}
```

Como vemos el array tiene el mismo contenido, pero primero se renderiza el sidebar y

luego el content

Bibliografía utilizada y sugerida

Fedosejev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt.

https://reactjs.org/docs/conditional-rendering.html https://reactjs.org/docs/forms.html https://reactjs.org/docs/lifting-state-up.html https://reactjs.org/docs/lists-and-keys.html