



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

Desarrollo en React JS

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

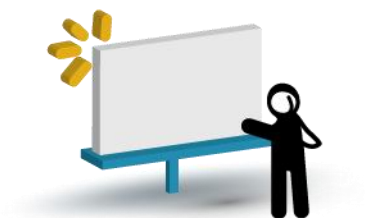
www.sceu.frba.utn.edu.ar/e-learning



Módulo III

Eventos, Firebase

Unidad 2: Firebase parte 2



Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Presentación:

En esta unidad aprenderemos como insertar, leer, actualizar y eliminar datos en nuestra base de datos de firebase.



Objetivos:

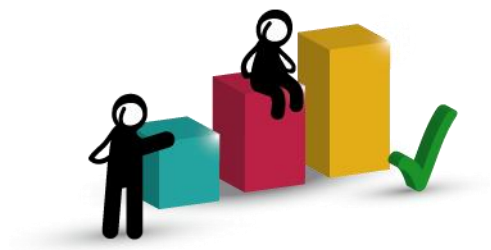
Que los participantes*:

- Aprendan a insertar, leer, eliminar y actualizar datos en firebase
- Analicen cambios a realizar en la estructura de la aplicación



Bloques temáticos*:

- Organicemos nuestro código
- Insertar datos en la base de firebase
- Consultar datos almacenados
- Actualizar o eliminar datos



Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

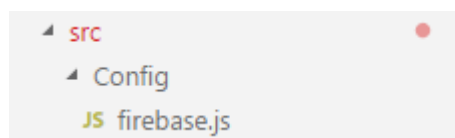
Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.



Organicemos nuestro código

Crear una carpeta **Config** dentro de **src** y dentro de esa carpeta crear un archivo **firebase.js**



En el archivo **firebase.js** incluimos el código que teníamos en **app.js**, quedando de la siguiente manera:

```
import * as firebase from 'firebase'

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

export default firebase;
```

Borrar el contenido de **App.js** (solo lo relativo a **firebase**)



De esta manera nuestra conexión a firebase queda disponible en cualquier parte de nuestro código, por ejemplo utilicemoslo en el componente Productos:

```
import firebase from '../Config/firebase';

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



Insertar datos en la base de firebase

Armamos un formulario de registro

Vamos a crear un componente **Registro**, en el colocaremos un formulario cuyos datos estén bindeados con el componente. Quedará de la siguiente manera:

```
import firebase from '../Config/firebase';

class Registro extends Component {
  constructor () {
    super();
    this.state = {
      nombre: ''
    }
    console.log(firebase.database());
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleSubmit(event) {
    alert('A name was submitted: ' + this.state.nombre);
    event.preventDefault();
  }
  handleChange(event) {
    this.setState({nombre: event.target.value})
    console.log(this.state.nombre);
  }
  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <div>
            <label>Nombre</label>
            <input type="text" placeholder="Introduzca nombre" value={this.state.nombre} onChange={this.handleChange.bind(this)}>
          </div>
        </form>
      </div>
    )
  }
}
```

Como vemos importamos la clase **firebase** creada anteriormente.

Luego creamos 2 métodos, uno para manejar las modificaciones en los input y otro para el submit del formulario.

En el método del submit podemos ver como acceder al contenido del input, el cual se almacena en un estado y es actualizado por el **onChange**



Guardar datos en firebase

Para guardar datos en firebase utilizaremos el método **set**

```
firebase.database().ref('users/' + userId).set({
  username: name,
  email: email,
  profile_picture : imageUrl
});
```

En nuestro ejemplo quedaría de la siguiente manera

```
handleSubmit(event) {
  //alert('A name was submitted: ' + this.state.nombre);
  console.log(this.state);
  firebase.database().ref('usuarios/').set({
    nombre: this.state.nombre,
    apellido: this.state.apellido,
    email: this.state.email,
    password: this.state.password
  });
  event.preventDefault();
}

handleChange(event) {
  const target = event.target;
  const value = target.type === 'checkbox' ? target.checked : target.value;
  const name = target.name;

  this.setState({
    [name]: value
  });
}

render() {
  return (
    <div>
      <form onSubmit={this.handleSubmit}>
        <div>
          <label>Nombre</label>
          <input type="text" placeholder="Introduzca nombre" name="nombre" value={this.state.nombre} onChange={this.handleChange}/>
        </div>
      </form>
    </div>
  );
}
```

La url especificada en el método ref es el nombre de la colección de datos que deseamos almacenar.


Vemos que en el método handleChange realizamos una modificación para que pueda actualizarse de forma genérica para cualquier input

Es importante tener el atributo **name** en los inputs

Desde la consola de firebase podemos ver que los datos fueron almacenados correctamente



 <https://react-74ab8.firebaseio.com/>

 **Tus reglas de seguridad están definidas como públicas
datos**

react-74ab8

 **usuarios**

— **apellido:** "Gil"

— **email:** "leangilutn@gmail.co

— **nombre:** "Leandro

— **password:** "dasasdasdas



Consultar datos almacenados

Para consultar los datos almacenados desde nuestra aplicación debemos realizar lo siguiente

```
componentDidMount(){  
  var this_class = this;  
  return firebase.database().ref('usuarios/').once('value').then(function(snapshot) {  
    //console.log(snapshot.val())  
    this_class.setState({  
      nombre: snapshot.val().nombre  
    })  
  });  
}
```

Lo haremos dentro de **componentDidMount** para poder actualizar la vista

Luego haremos la asignación de una variable, a la cual le asignamos **this** ya que sino dentro del promise **this** hará no hará referencia a la clase.

En el objeto **snapshot** que recibimos en el promise tenemos los datos devueltos por firebase, para acceder a los mismos debemos hacerlo con el método **val()**



Actualizar o eliminar datos

Para actualizar datos debemos utilizar el método **update**

```
var updates = {};  
updates['/posts/' + newPostKey] = postData;  
updates['/user-posts/' + uid + '/' + newPostKey] = postData;  
  
return firebase.database().ref().update(updates);
```

De manera similar al utilizado para agregar o leer datos.

En caso de querer borrar datos debemos utilizar **delete()** en lugar de **update()**



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/forms.html>

<https://firebase.google.com/docs/database/web/read-and-write?hl=es-419>

<https://www.sitepoint.com/work-with-forms-in-react/>



Lo que vimos:

En esta unidad aprendimos cómo insertar, leer, actualizar y eliminar datos en firebase



Lo que viene:

En la próxima unidad aprenderemos cómo realizar nuestra aplicación con todos los temas vistos de firebase.

