

Collisae

# **Manual Técnico Sitio Web ASODEC-SJ**

07/06/2017  
San José, Costa Rica

# Contenido

<b>Consola Firebase</b>	<b>3</b>
Acceder a la Consola Firebase	3
Administrar Usuarios	5
Administrar Base de Datos	10
Administrar Almacenamiento en la Nube	14
<b>Código Javascript/jQuery</b>	<b>18</b>
Acceder a elementos DOM	18
Validación de Formularios	19
<b>Código Firebase</b>	<b>22</b>
Aspectos Básicos	22
Inicio de Sesión y LogOut	23
Lectura de Datos	25
Lectura por Id	26
Ejemplo de “Join”	27
Escritura de Datos	28
Push	28
Update	29
Set	29
Eliminación de Datos	30
<b>Otros Recursos</b>	<b>30</b>

# Consola Firebase

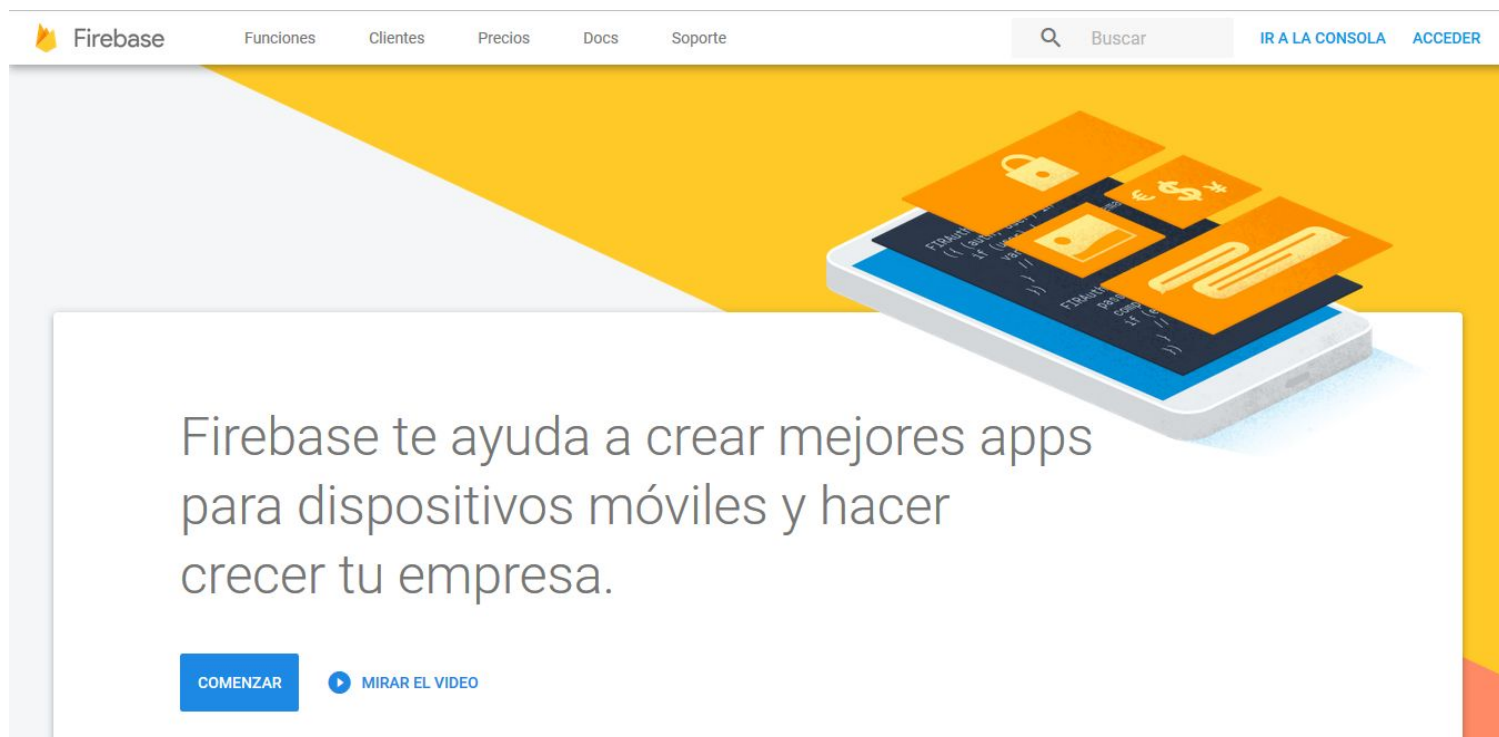
Firebase es una plataforma de desarrollo web y móvil creada en el 2011 por Andrew Lee y James Tamplin. Posteriormente fue adquirida por Google en el 2014. Esta plataforma brinda servicios como base de datos en tiempo real, autenticación, almacenamiento en la nube, test lab para Android, reporte de fallos, hosting, AdMob, entre otros.

Para este proyecto se utilizaron principalmente las funciones de base de datos en tiempo real, autenticación y almacenamiento en la nube. A continuación se presenta como acceder y utilizar la consola de firebase para utilizar estas tres funcionalidades con el fin de mantener el proyecto.

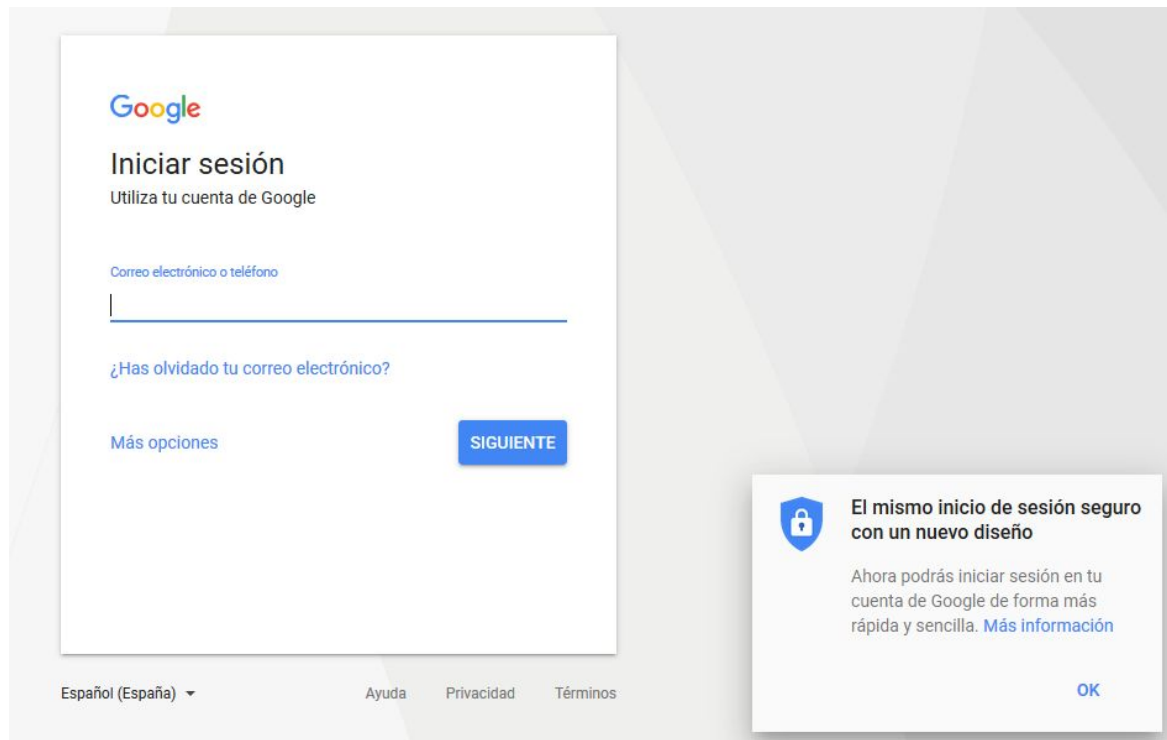
## Acceder a la Consola Firebase

Ingresa a la siguiente dirección en su navegador de preferencia:  
<https://firebase.google.com/>

Posteriormente, haga click en “IR A LA CONSOLA”.

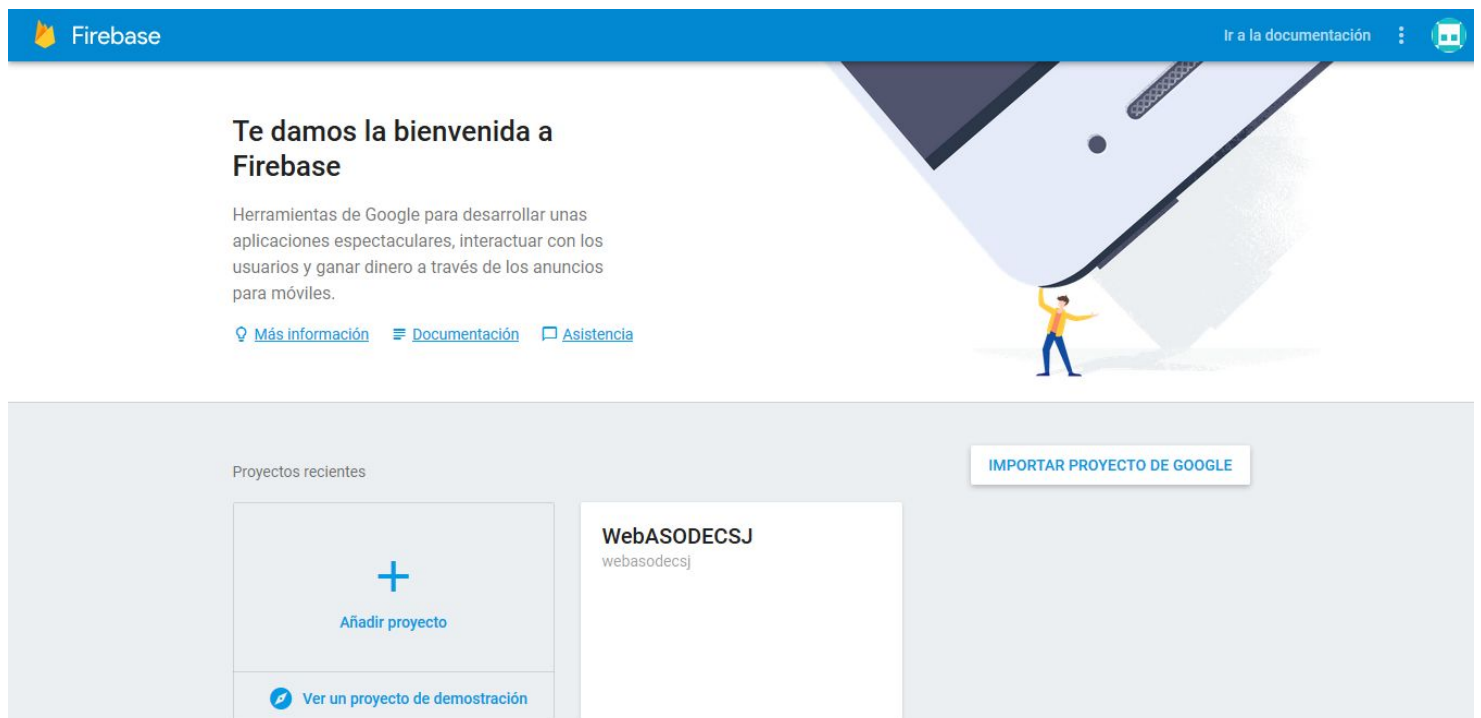


Se le solicitará iniciar sesión con su cuenta Google relacionada con la base de datos que desea gestionar. En el caso de este proyecto la cuenta es **asodecsj.web@gmail.com**.

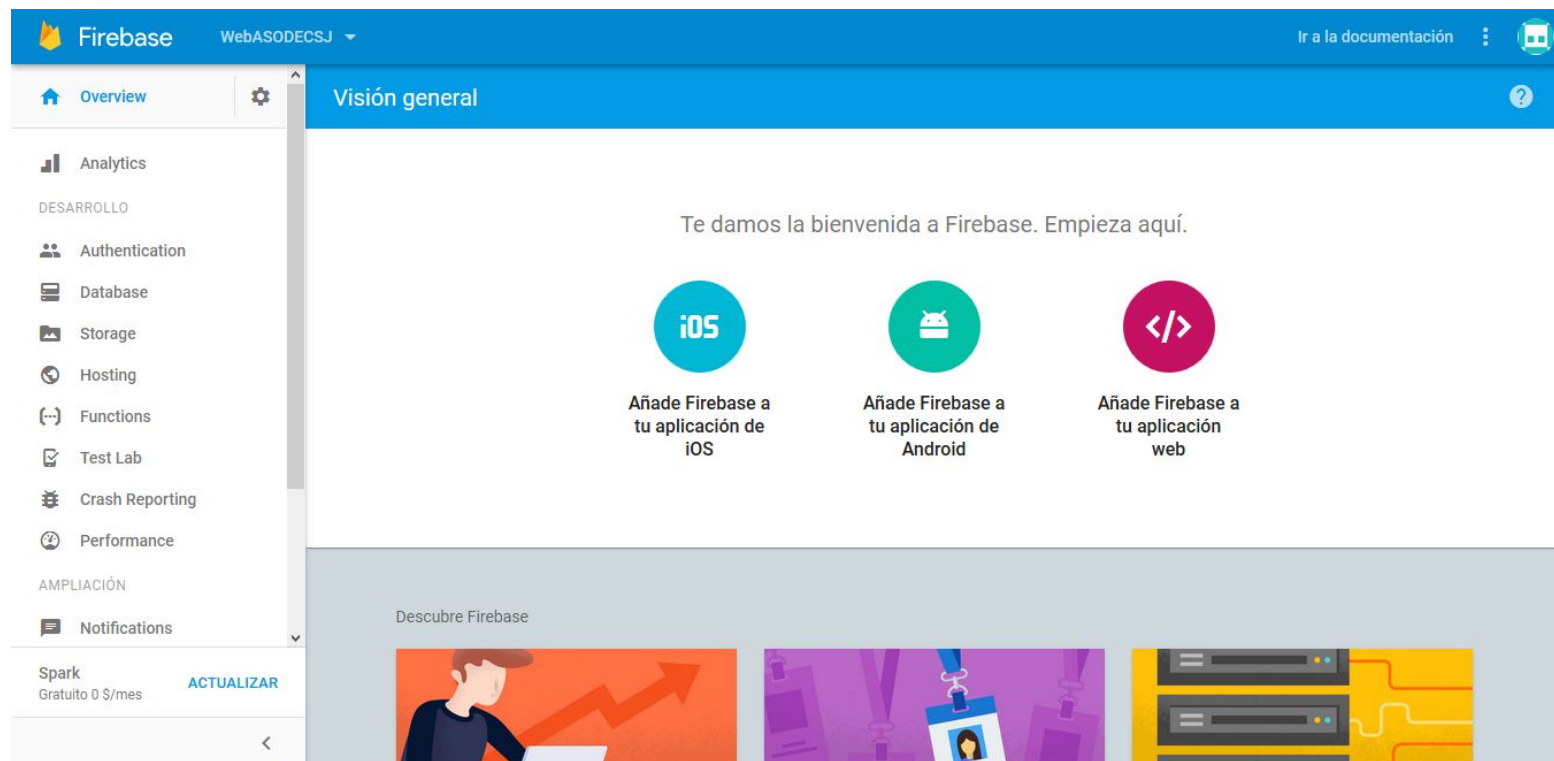


The image shows the Google login interface. At the top is the Google logo. Below it is the heading "Iniciar sesión" (Sign in) with the subtext "Utiliza tu cuenta de Google" (Use your Google account). There is a text input field labeled "Correo electrónico o teléfono" (Email or phone). Below the field is a link "¿Has olvidado tu correo electrónico?" (Forgot your email?). To the left of the "SIGUIENTE" (Next) button is a link "Más opciones" (More options). At the bottom of the form are links for "Español (España)" (Spanish (Spain)), "Ayuda" (Help), "Privacidad" (Privacy), and "Términos" (Terms). To the right of the form is a message box with a shield icon, stating "El mismo inicio de sesión seguro con un nuevo diseño" (The same secure sign-in with a new design). It continues: "Ahora podrás iniciar sesión en tu cuenta de Google de forma más rápida y sencilla. Más información" (Now you can sign in to your Google account faster and easier. More information). An "OK" button is at the bottom right of the message box.

Una vez haya ingresado los datos solicitados, correo y contraseña, podrá acceder a la consola firebase.

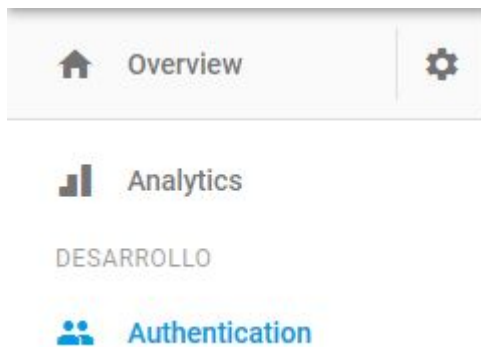


En la pantalla que se le presentó, seleccione el proyecto al que desea acceder, en este caso es **WebASODECSJ**. Al seleccionarlo, se le mostrará el administrador del proyecto, en el que puede revisar los datos de la base de datos, las imágenes que ha almacenado y los usuarios que pueden realizar modificaciones en su base de datos.



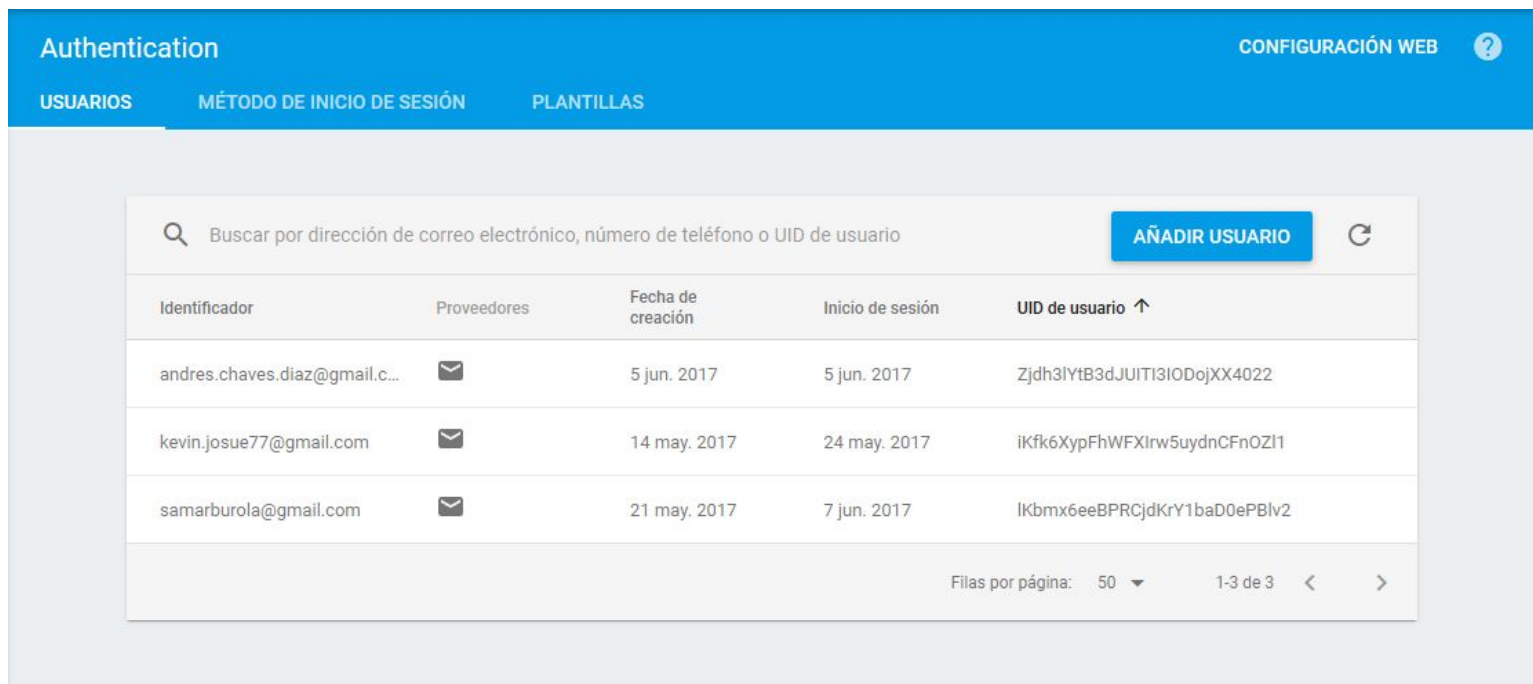
## Administrar Usuarios

Firebase le permite mantener control de los usuarios que pueden realizar modificaciones en la base de datos y en el almacenamiento. En este proyecto, esto se utiliza para el inicio de sesión en el Sistema de Gestión de Contenidos (CMS), en el cual los administradores de la página web pública realizan modificaciones en los datos que se muestran en esta.



Para agregar, editar o eliminar usuarios acceda a la opción Authentication en la barra lateral de la consola firebase.

Se le mostrará la siguiente pantalla:

The image shows the 'Authentication' page in the Firebase console, specifically the 'USUARIOS' (Users) tab. At the top, there's a search bar and an 'AÑADIR USUARIO' (Add User) button. Below is a table with columns: 'Identificador', 'Proveedores', 'Fecha de creación', 'Inicio de sesión', and 'UID de usuario'. There are three users listed. At the bottom right, it says 'Filas por página: 50' and '1-3 de 3'.

Como se puede observar, se enlistan los usuarios que pueden modificar los datos de la base de datos. Al pasar el mouse sobre alguno de ellos se mostrará la opción

de copiar UID, y los tres puntos de más opciones. Al hacer click en estos tres puntos se muestran las siguientes opciones:

Cambiar contraseña

Inhabilitar cuenta

Eliminar cuenta

Observe que puede cambiar la contraseña del usuario. Inhabilitar su cuenta para bloquear el acceso, o bien, eliminar la cuenta del todo.

Para agregar un nuevo usuario haga click en “AGREGAR USUARIO”. Se le mostrará la siguiente pantalla:

Añade un usuario de correo electrónico/contraseña

Correo electrónico

Contraseña

CANCELAR

AÑADIR USUARIO

Ingresa los datos solicitados y presione el botón “AÑADIR USUARIO”. De esta manera le ha dado acceso a un nuevo usuario al CMS para realizar cambios en el sitio web público.

Como se observa en una captura anterior, hay una pestaña llamada “MÉTODO DE INICIO DE SESIÓN” al acceder a ella se muestra la siguiente pantalla:

USUARIOS		MÉTODO DE INICIO DE SESIÓN	PLANTILLAS
Proveedores de inicio de sesión			
Proveedor	Estado		
✉ Correo electrónico/contraseña	Habilitada		
☎ Teléfono	Inhabilitado		
🌐 Google	Habilitada		
📘 Facebook	Inhabilitado		
🐦 Twitter	Inhabilitado		
🐙 GitHub	Inhabilitado		
👤 Anónimo	Inhabilitado		

En esta sección se le permite seleccionar los proveedores de inicio de sesión, esto le permite utilizar dichos proveedores para iniciar sesión en el CMS y realizar modificaciones.

Al bajar se presentan más opciones relacionadas con el inicio de sesión:

Authorized Domains ⓘ

AÑADIR DOMINIO

Dominio autorizado	Tipo
localhost	Predeterminado
webasodecsj.firebaseio.com	Predeterminado

Opciones avanzadas

**Una cuenta por dirección de correo electrónico**  
Se impide a los usuarios crear varias cuentas con la misma dirección de correo electrónico y con distintos proveedores de autenticación. [Más información](#) ⓘ

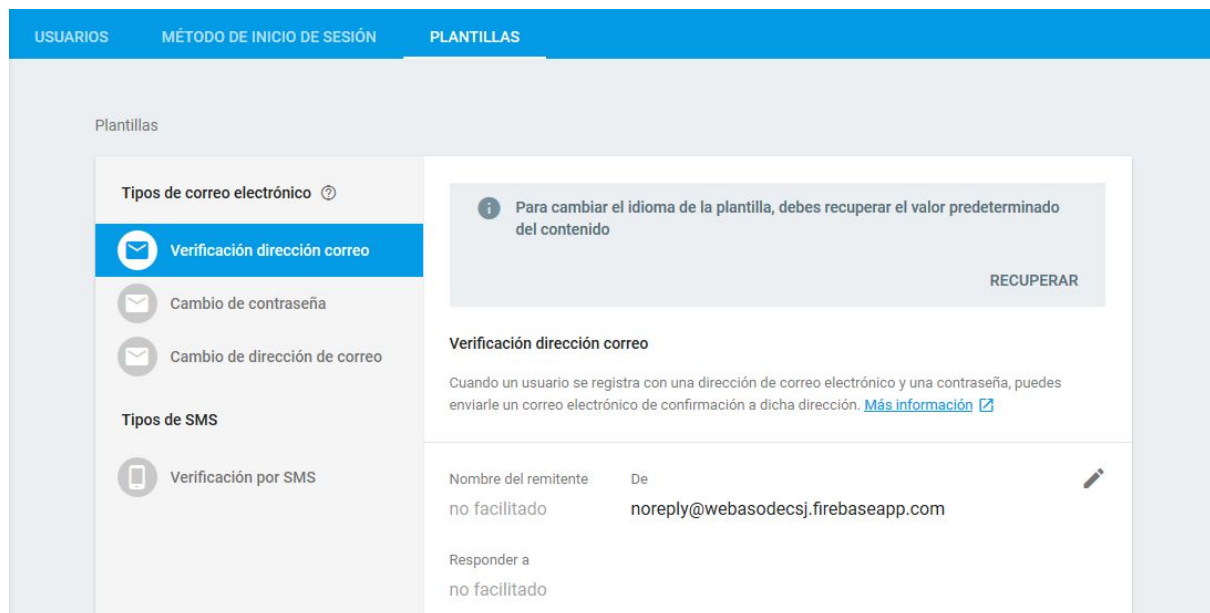
CAMBIAR

**Administrar la cuota de registro**  
Para proteger tu proyecto de cualquier uso inadecuado, limitamos el número de registros (tanto los anónimos como los que utilizan correo electrónico y contraseña) que pueden realizarse en tu aplicación desde una misma dirección IP. En esta sección puedes solicitar y programar cambios temporales para esta cuota.  
Cuota por hora actual: 15

CAMBIAR

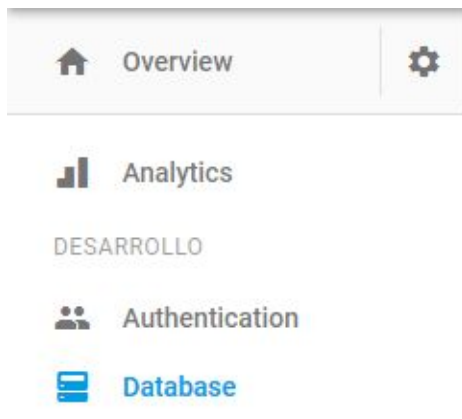


Finalmente, al acceder a “PLANTILLAS” en el menú, se le mostrarán posibles formatos de correos electrónicos y mensajes de texto que puede alterar y utilizar para verificar correos al registrar usuarios, cambiar contraseña, cambiar correo electrónico o iniciar sesión mediante el servicio de mensajería instantánea.



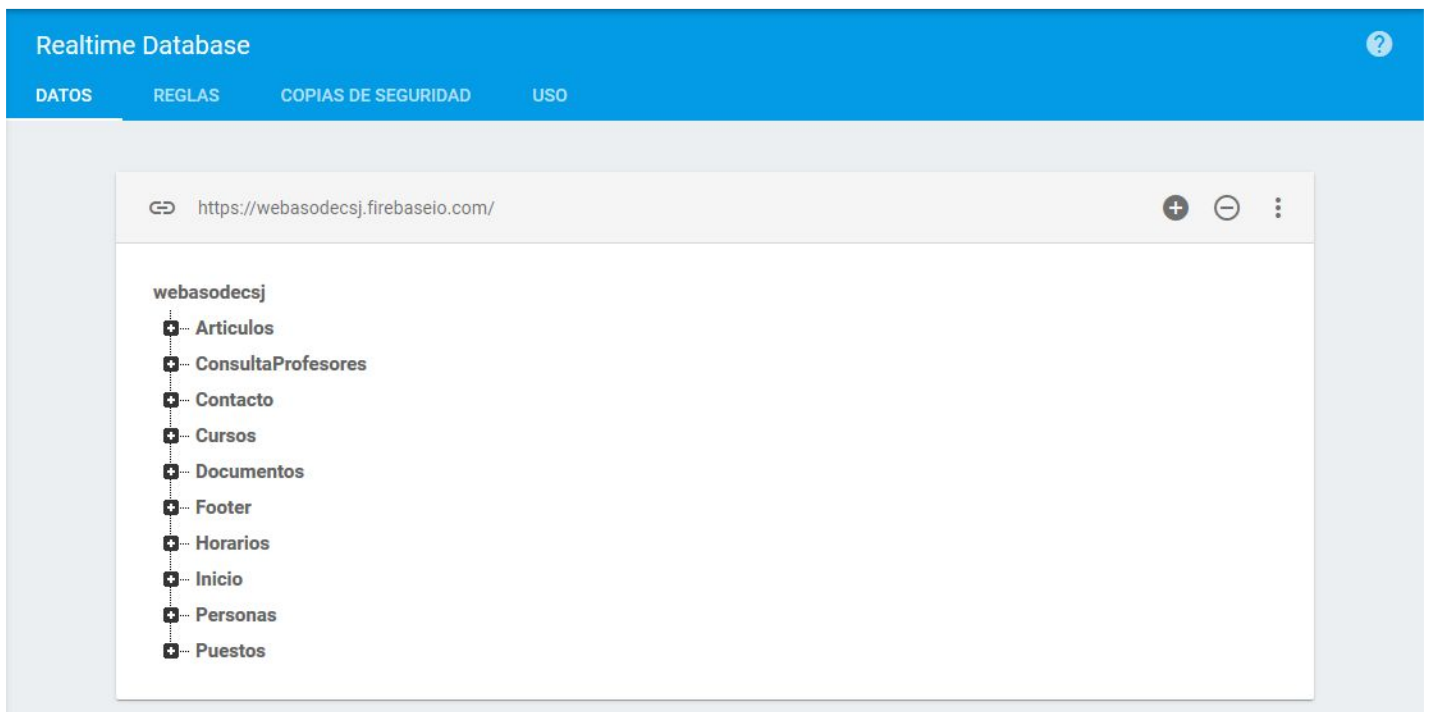
## Administrar Base de Datos

Firebase desde su consola permite ver los datos actuales en la base de datos, así como definir una serie de reglas en ella, realizar copia de seguridad y ver las estadísticas de uso.



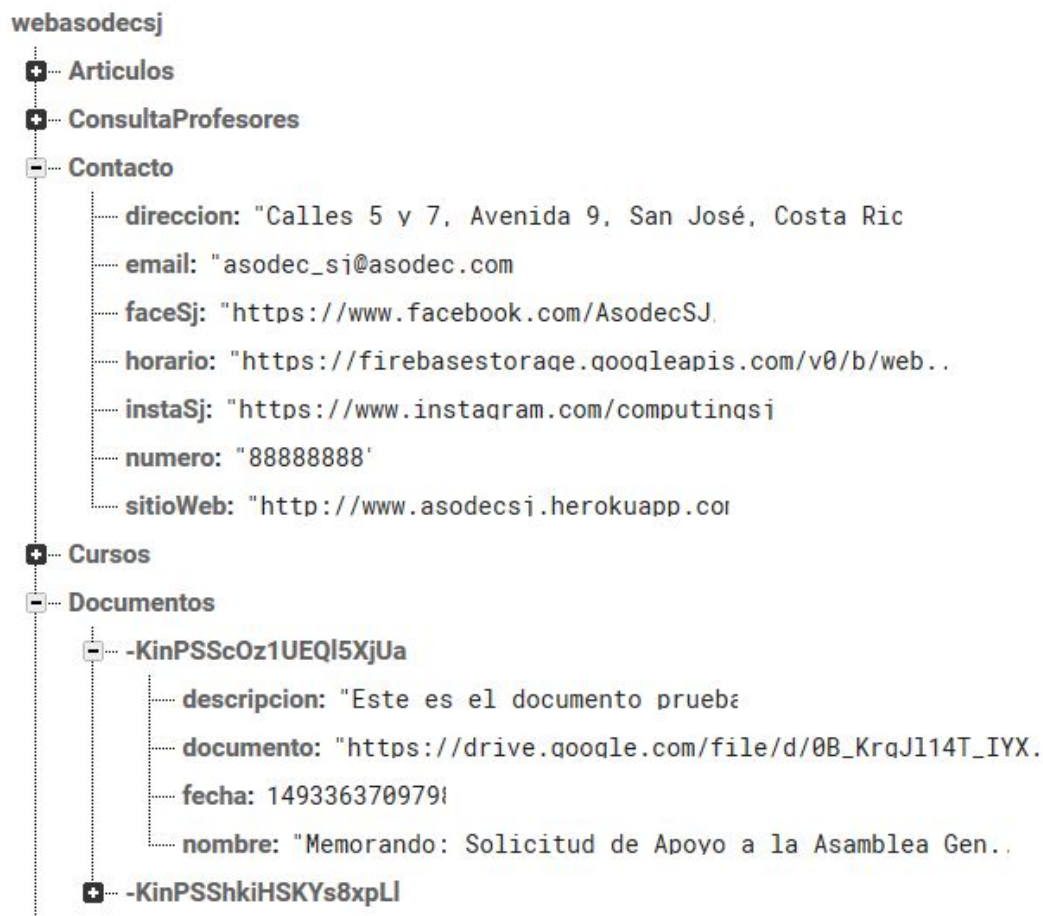
Para acceder a esta sección, ingrese a “Database” en la barra lateral de la consola de firebase.

Se le mostrará la siguiente pantalla:



Aquí se debe notar que la base de datos en cuestión es noSQL del tipo llave-valor, por lo que los datos no se almacenan en tablas, si no en formato JSON, por lo que para su mantenimiento se requiere de conocimiento de modelado en este tipo de base de datos.

Así mismo, en esta pantalla se muestran los datos que utiliza el sitio público para funcionar de manera adecuada, para observarlos, presione cualquiera de los “+” en los datos, estos se muestran así:

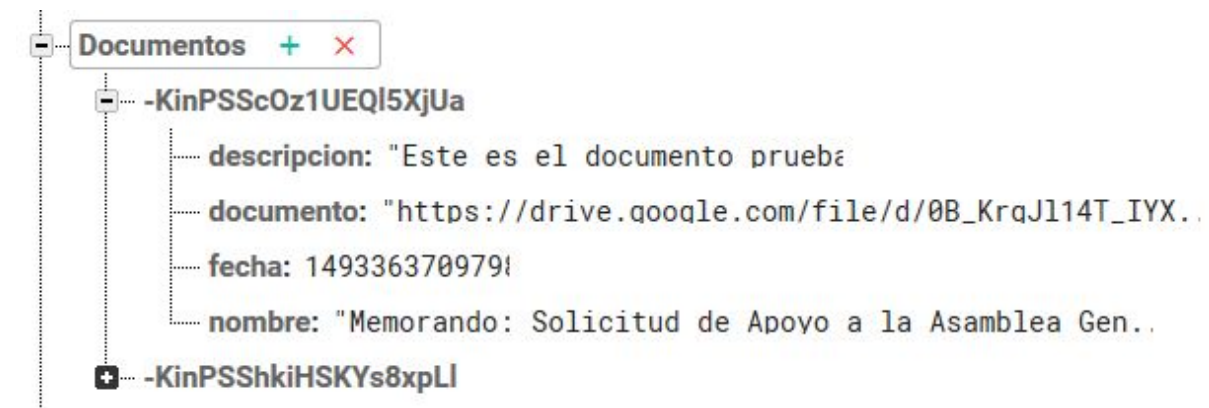


Cabe resaltar que las llaves de los datos son generadas automáticamente por Firebase a la hora de agregar datos a la base de datos. Además, en caso de modificaciones en la base firebase nos las muestra de la siguiente manera:

- **Datos en color verde:** Son nuevos datos que se acaban de agregar a la base de datos. Se muestran en color verde por algunos segundos.
- **Datos en color amarillo:** Son datos que fueron modificados. Aquellos nodos que fueron alterados de alguna manera se mostrarán de color amarillo por algunos segundos.
- **Datos en color rojo:** Son datos que fueron eliminados. Aquellos nodos que fueron eliminados se mostrarán de color rojo por algunos segundos y posteriormente desaparecerán.

**Se recomienda no** alterar datos desde esta parte de la consola para evitar problemas en el sitio web público.

Al pasar el mouse sobre algunos de los datos, se mostrará una cruz verde (+) o una equis roja (X). La cruz verde permite agregar un nuevo par llave-valor como hijo del nodo seleccionado. La equis roja elimina el nodo seleccionado. **CUIDADO** en caso de ser un nodo padre se eliminarán **todos** los datos.



**+:**

Documentos X

Nombre  Valor  + X

CANCELAR AÑADIR

**X:**

Eliminar datos X

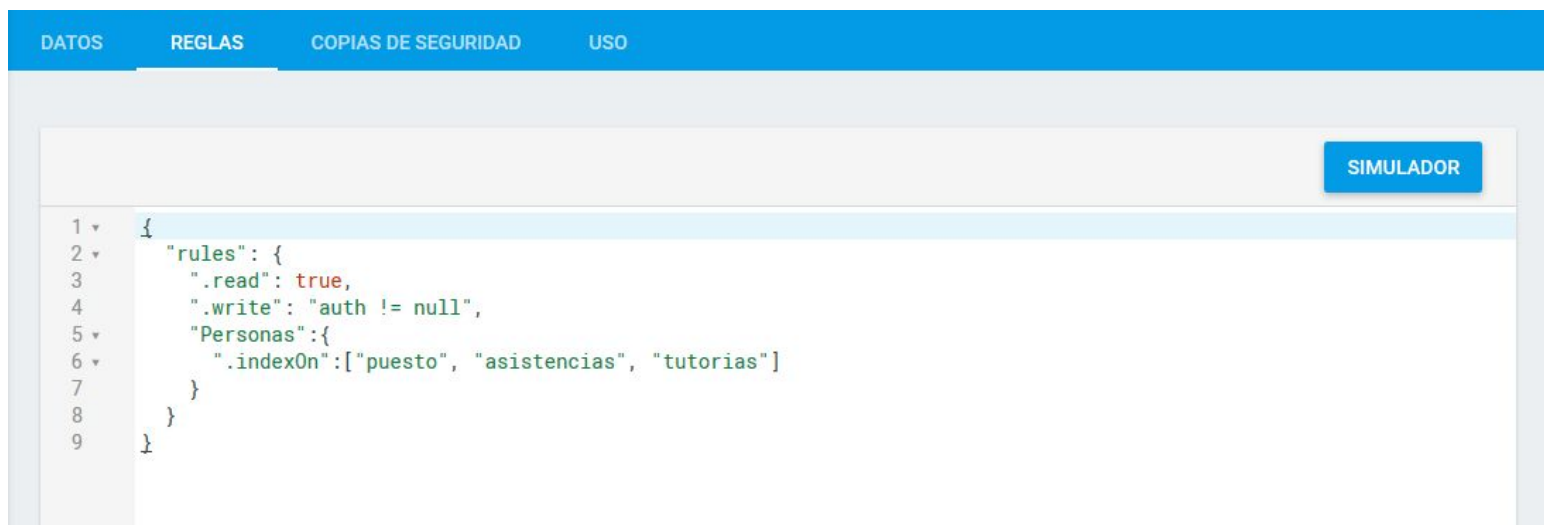
⚠ Todos los datos de esta ubicación (incluidos los datos anidados) se eliminarán de forma permanente

Ubicación de los datos  
/Documentos

Para eliminar datos sin que aparezca este mensaje, mantén pulsada la tecla Mayús y haz clic al eliminar un nodo.

CANCELAR ELIMINAR

Ingresa a “REGLAS” y se mostrará la siguiente pantalla:



Las reglas en firebase se utilizan para definir quienes pueden leer y escribir a la base de datos, así como definir índices en datos con el fin de mejorar el rendimiento de los queries. Note que “write” tiene como necesario que la autenticación sea diferente a null, es decir, que solamente usuarios autenticados pueden modificar la base de datos. Así mismo, se definieron índices en puestos, asistencias y tutorías, ya que las personas se filtran por esas propiedades a la hora de obtenerlas en el sitio público.

Para más información sobre las reglas de firebase puede acceder a la siguiente página: <https://firebase.google.com/docs/database/security/quickstart?hl=es-419>.

Por otro lado, la opción “SIMULADOR” que se observa en la pantalla permite simular la lectura y escritura a la base de datos con el fin de verificar que las reglas funcionen correctamente.


Ingresa a la pantalla “COPIAS DE SEGURIDAD” y se mostrará la siguiente pantalla:

DATOS

REGLAS

COPIAS DE SEGURIDAD

USO



Guarda copias de seguridad diarias de tus datos y reglas de seguridad de forma automática

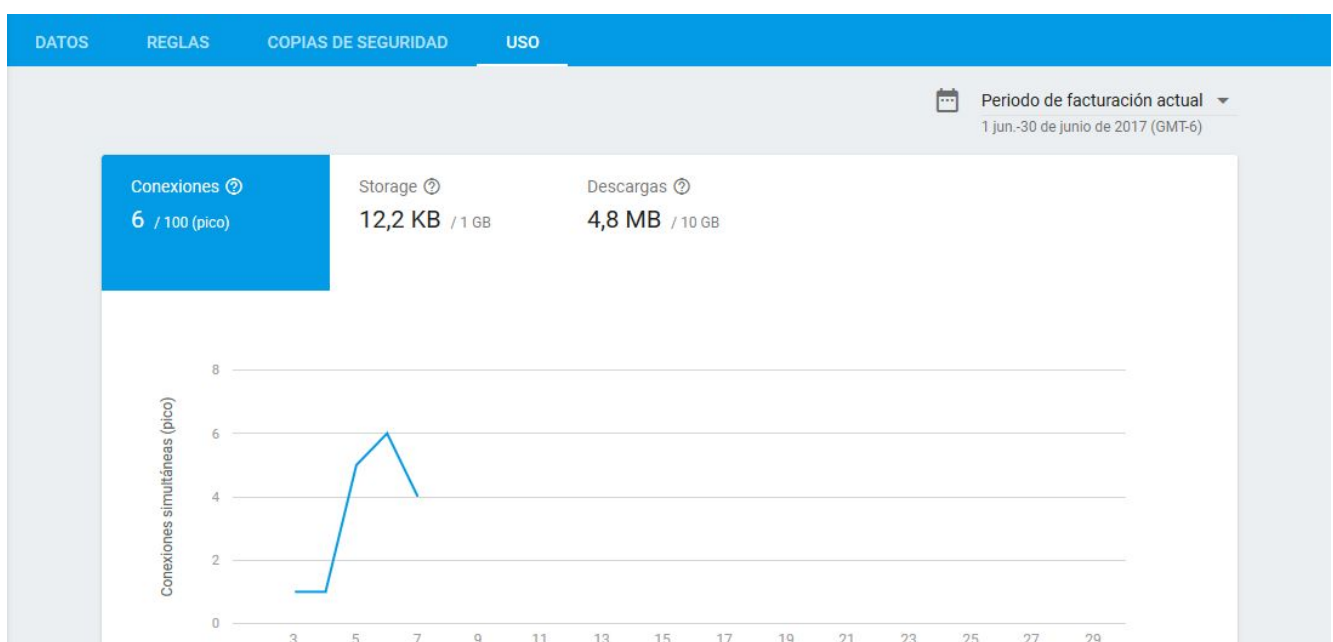
[Más información](#)

**ACTUALIZAR AHORA**

Disponible con una actualización al plan Blaze

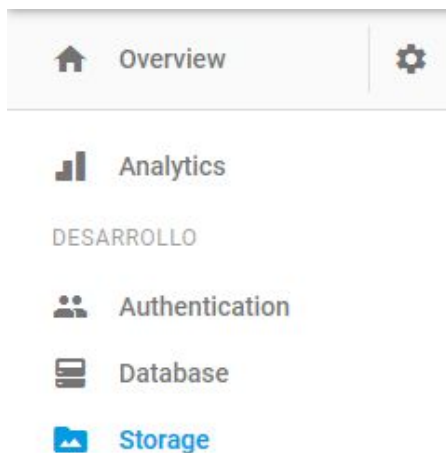
Dado que en el proyecto el almacenamiento utilizado es el gratuito, actualmente no se puede realizar copia de seguridad. Para más información de los planes de firebase puede ingresar al siguiente sitio: <https://firebase.google.com/pricing/>

Finalmente al ingresar a la pantalla “USO” se pueden ver estadísticas del uso de la base de datos como conexiones a ella, cantidad de datos almacenados y cantidad de datos descargados (leídos).



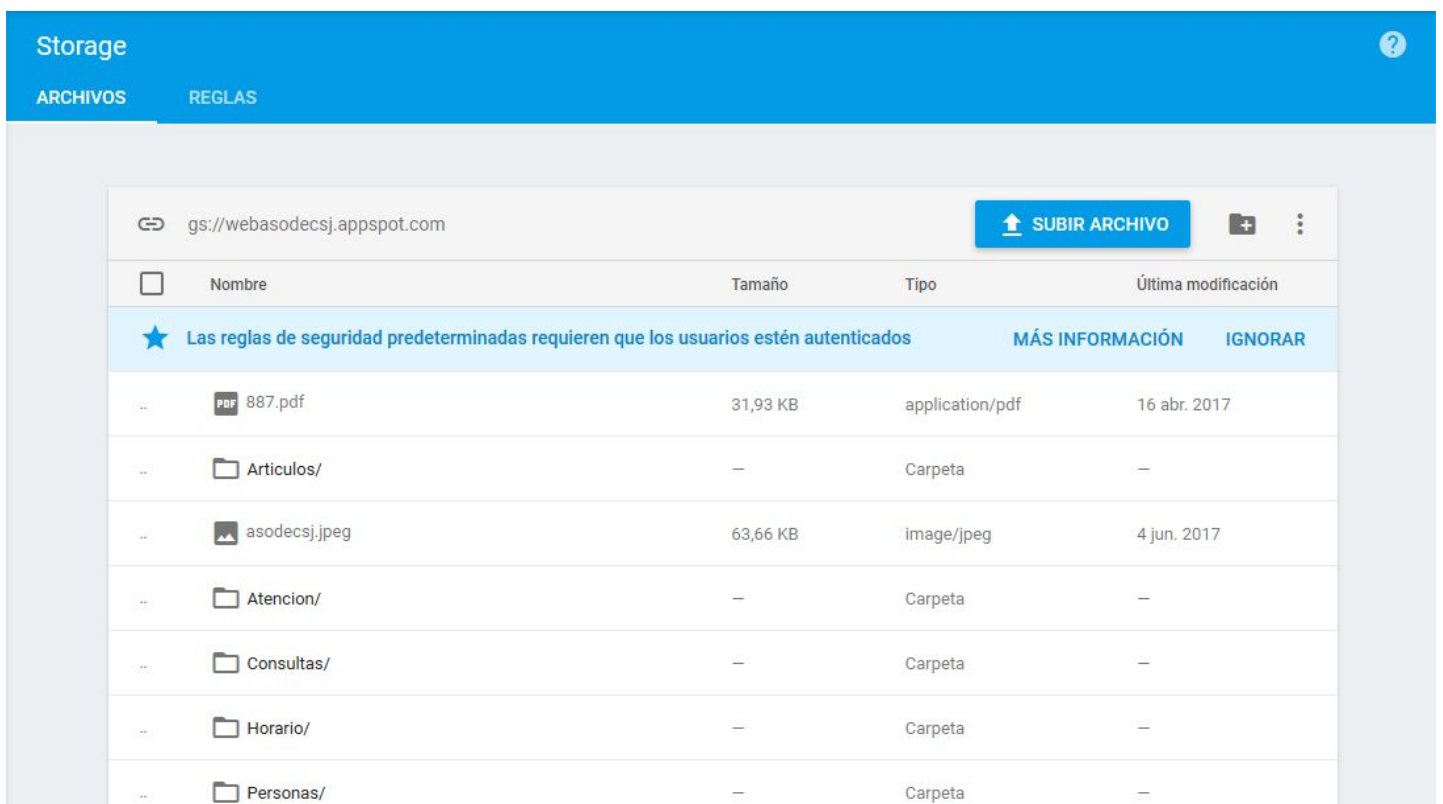
## Administrar Almacenamiento en la Nube

Firebase permite administrar el almacenamiento de archivos desde su consola. Esto demuestra también que firebase es una base de datos no solo llave-valor si no también orientada a documentos.

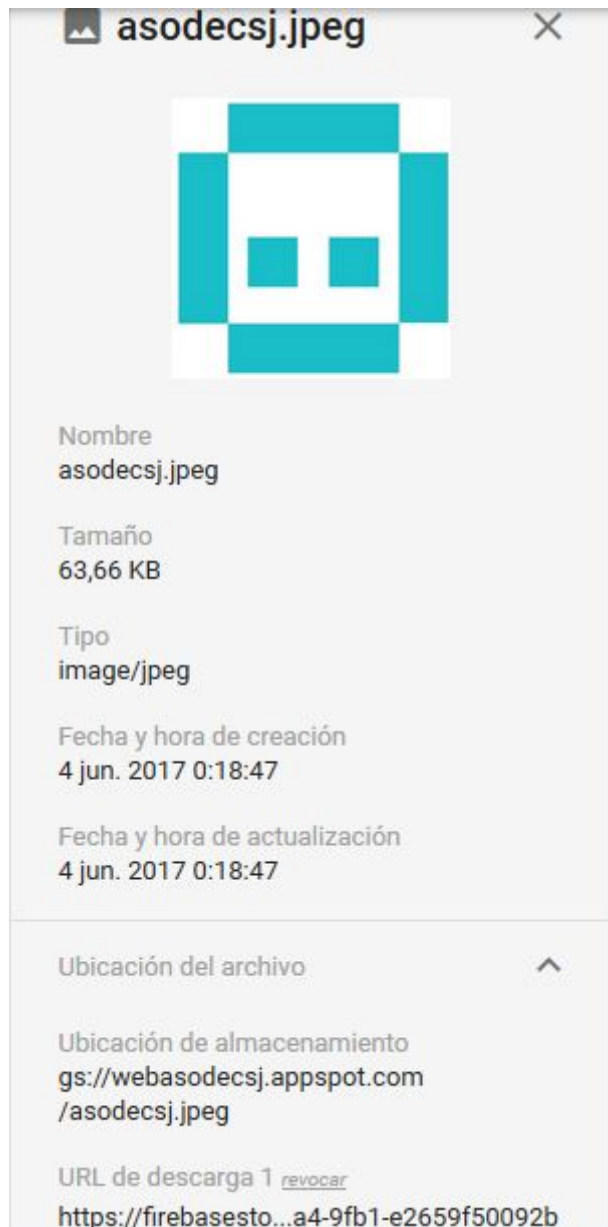


Para acceder al almacenamiento en la Nube ingrese a “Storage” en la barra lateral de la consola de Firebase.

Se le mostrará la siguiente pantalla:



En ella se muestran los archivos y carpetas que se encuentran almacenadas en la base de datos.

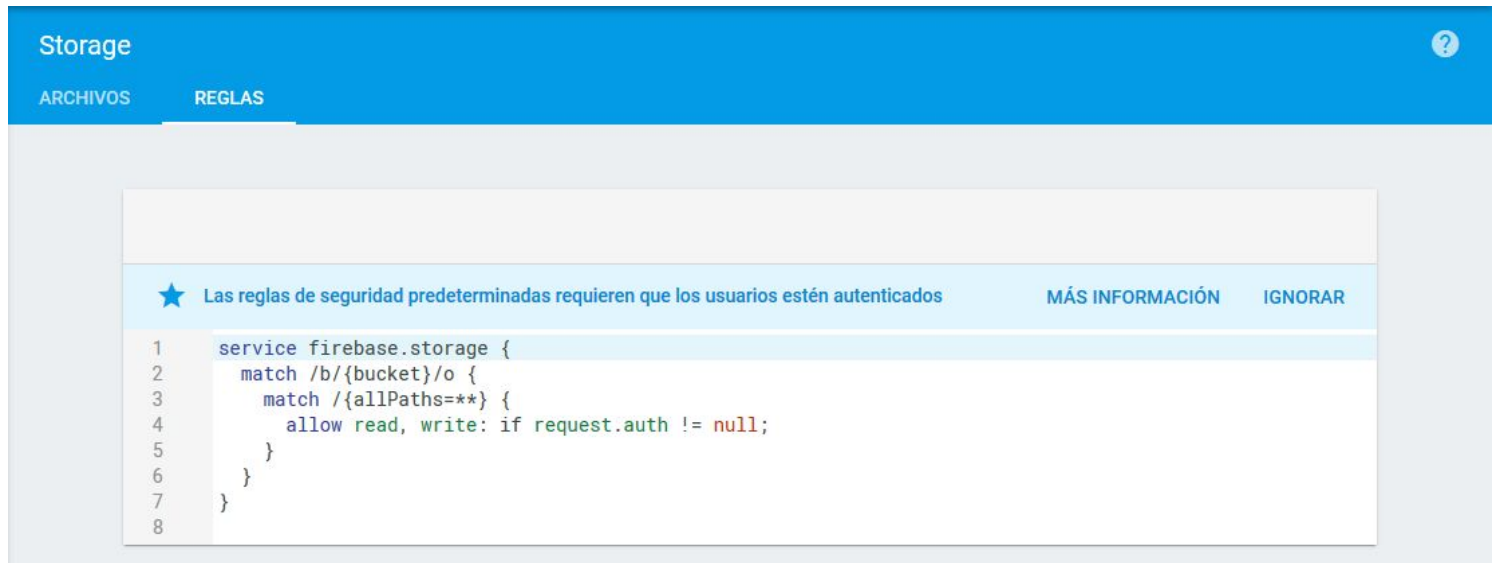


Al dar click sobre algún archivo, se le mostrará un preview del mismo, así como su información, el nombre, el tamaño, el tipo de archivo, la fecha y hora de creación y la fecha y hora de actualización. Así mismo, se brindan las direcciones desde las que se puede acceder al archivo en particular.

Si ingresa a una carpeta, se le mostrarán los archivos en esta con las mismas opciones.



Ingresa a “REGLAS” y se le mostrará lo siguiente:



Al igual que en la base de datos, son reglas para limitar el acceso a los archivos. Para más información acerca de las reglas de firebase puede acceder a la siguiente página: <https://firebase.google.com/docs/database/security/quickstart?hl=es-419>.

# Código Javascript/jQuery

Con el fin de entender de mejor manera el uso de firebase y la modificación de la interfaz gráfica con los datos de la base de datos se explicarán aspectos básicos y puntuales de jQuery. Se requiere conocimiento básico de javascript.

## Acceder a elementos DOM

Para acceder a elementos DOM jQuery facilita el trabajo:

```
//Acceder a elementos DOM por Id
$("#idElemento")
$("#idElemento").val();

//Hacer binding de un evento al elemento.
//Cambiar evento por click, change, etc.
$("#idElemento").on('evento', function(){

});

//Acceder a elementos DOM por clase
$(".claseElemento")
```

En este proyecto se accedió principalmente mediante el uso de Ids. Los eventos es de suma importancia para saber cuando se seleccionó alguna opción de un select, o se marcó una casilla o se presionó un botón. Por ejemplo para llevar a cabo alguna acción en caso de marcar una casilla:

```
$("#checkEncargoEdit").on('change', function(){
    if ($("#checkEncargoEdit").is(":checked")){
        $("#encargoEdit").show()
    } else {
        $("#encargoEdit").hide()
    }
})
```

Note además el método `.is()` para determinar si la casilla está marcada o desmarcada.

## Validación de Formularios

Gran parte de lo que se llevó a cabo mediante Javascript fue la validación de formularios para evitar formatos inesperados de los datos esperados. Esta validación se llevo a cabo mediante la biblioteca jQueryValidaton. Puede obtener más información de esta biblioteca en el enlace <https://jqueryvalidation.org/>.

A continuación, se presenta un breve ejemplo del funcionamiento de la validación de formularios para facilitar el mantenimiento de esta sección.

```
var Validation = function () {  
    return {  
        //Validation  
        initValidation: function () {  
            $("#formAgregarDocumentos").validate({  
                // Rules for form validation  
                rules:  
                {  
                    nombreDocumento:  
                    {  
                        required: true  
                    },  
                    descripcionDocumento:  
                    {  
                        required: true  
                    },  
                    linkDocumento:  
                    {  
                        required: true,  
                        url: true  
                    }  
                },  
                // Messages for form validation  
                messages:  
                {  
                    nombreDocumento:  
                    {  
                        required: 'Ingrese el nombre del documento.'  
                    },  
                    descripcionDocumento:  
                    {  
                        required: 'Ingrese la descripción del documento.'  
                    },  
                    linkDocumento:  
                    {  
                        required: 'Ingrese el link de la ubicación del documento.'  
                    }  
                }  
            });  
        }  
    }  
};
```

En esta primera parte se observa la inicialización de la validación. Así mismo hay dos secciones principales:

- Reglas: El nombre de cada regla es el mismo que el Id del elemento DOM que se va a validar, por ejemplo un input o un select. En formato JSON, se agregan las validaciones que deben aprobarse para que los datos de dicho elemento estén correctos. Para ver cuáles reglas hay disponibles, se puede ingresar al sitio de la biblioteca.
- Mensajes: Es el mensaje que el usuario observará cuando un dato que haya ingresado no cumpla con la regla. Por cada elemento y regla debe haber un mensaje, de lo contrario la biblioteca mostrará uno por defecto en inglés.

```
// Do not change code below
errorPlacement: function(error, element)
{
    error.insertAfter(element.parent());
},
submitHandler: function(form)
{
    var formData = $(form).serializeArray()
    var nombreDoc = formData[0]["value"];
    var descDoc = formData[1]["value"];
    var linkDoc = formData[2]["value"];

    firebase.database().ref('Documentos/').push({
        documento: linkDoc,
        fecha: moment().valueOf(),
        nombre: nombreDoc,
        descripcion: descDoc
    });

    alert("Documento agregado correctamente");
    cargarSelect()
    $("#nombreDocAgregar").val('');
    $("#descDocAgregar").val('');
    $("#linkDocAgregar").val('');
    return false;
}
});

$.validator.addMethod("checkSelect", function(value) {
    return $("#docSelect").val() != ""
});
}
}();
```

En esta segunda parte se muestra la función errorPlacement, que indica en que parte de la página se colocará el error. En este caso se coloca debajo del campo del

formulario que presentó el error. Posteriormente se muestra la función submitHandler, que recibe el formulario como parámetro y abarca el código que se ejecuta en caso de que el formulario pase la validación. Note que por esta razón es ahí donde se escribe a la base de datos ya que se cumple el formato de los datos.

Finalmente, debajo de cada formulario por validar se pueden colocar métodos o reglas propias de validación, en este caso se podría poner en una regla "checkSelect" en caso de required, y la validación se hará con dicho método. Note que debe devolver true o false dependiendo de lo que se desee validar.

# Código Firebase

A continuación, se presentan las operaciones básicas que se llevaron a cabo con Firebase mediante código, como recuperar, guardar, actualizar y eliminar datos. Se presentan ejemplos concretos con el fin de dar una introducción básica a Firebase y facilitar el mantenimiento de la página web.

## Aspectos Básicos

Para conectar la página web con firebase podemos ir a Overview en la consola de Firebase e ingresar a “Añade Firebase a tu aplicación web”.

Visión general

Te damos la bienvenida a Firebase. Empieza aquí.

iOS

Añade Firebase a tu aplicación de iOS

Añade Firebase a tu aplicación de Android

</>

Añade Firebase a tu aplicación web

Añade Firebase a tu aplicación web

Copia y pega el fragmento que se indica a continuación en la parte inferior de tu código HTML, delante de otras etiquetas de secuencia de comandos.

```
<script src="https://www.gstatic.com/firebasejs/4.1.2/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyD03G77hqsICi7lKAMzzvyOkfxDmBg0xw4",
    authDomain: "webasodecsj.firebaseio.com",
    databaseURL: "https://webasodecsj.firebaseio.com",
    projectId: "webasodecsj",
    storageBucket: "webasodecsj.appspot.com",
    messagingSenderId: "595397153077"
  };
  firebase.initializeApp(config);
</script>
```

COPIAR

Consulta estos recursos para obtener más información acerca de Firebase para aplicaciones web:

[Get Started with Firebase for Web Apps](#) [Firebase Web SDK API Reference](#) [Firebase Web Samples](#)

Este código debe colocarse en el .php o .html que hará uso de firebase. La primera línea de <script> es obligatoria. La segunda línea de <script> donde se inicializa firebase puede colocar dentro del mismo .php o .html, o bien, en un .js que se incluya.

## Inicio de Sesión y LogOut

Como se vio anteriormente, Firebase puede manejar sus propios usuarios y a su vez brinda código que permite iniciar sesión con esas credenciales.

Para iniciar sesión basta con agregar el siguiente código:

```
firebase.auth().signInWithEmailAndPassword($("#form :input")[0].value, $("#form :input")[1].value).catch(function(error) {  
    // Handle Errors here.  
    var errorCode = error.code;  
    var errorMessage = error.message;  
    console.log(errorCode);  
    console.log(errorMessage)  
    // ...  
});
```

Simplemente hay que pasarle como parámetro el correo y contraseña que ingresó el usuario. Firebase se encarga de validar estos datos, así como de crear los cookies necesarios para mantener la sesión activa.

Note que en caso de error, este estará almacenado en dos variables, una con el código de error y otro con el mensaje de error. El bloque de código donde se muestran dichas variables, así como los console.log() se ejecutará únicamente si el inicio de sesión falla por alguna razón como usuario y contraseña incorrectos.

Para verificar la sesión se debe agregar el siguiente código:

```
firebase.auth().onAuthStateChanged(function(user) {  
    if (user) {  
        window.location.replace("index.php");  
    }  
});
```

Verifica si existe un usuario. En este caso corresponde al módulo de LogIn, por lo que si un usuario logra iniciar sesión, se redirige al inicio del CMS.

Cabe resaltar que el siguiente código aparece en el .js correspondiente de cada página con el fin de verificar la sesión y evitar modificaciones no permitidas. Además, agrega al navbar el correo del usuario activo.

```
firebase.auth().onAuthStateChanged(function(user) {  
  if (user) {  
    $("#account").html(user.email);  
  } else {  
    window.location.replace("login.html");  
  }  
});
```

A pesar de que permite que las páginas se carguen por un segundo, no hay problema en cuanto a seguridad debido a que si se intenta modificar un dato la base de datos bloqueará la modificación ya que no hay usuario activo. Si un usuario no está activo, este código redirige al inicio de sesión.

Finalmente, para cerrar sesión basta con agregar el siguiente código al archivo .js encargado del cierre de sesión, en este caso es el mismo de login, ya que cuando un usuario solicita cerrar sesión, se redirige a la pantalla de login.

```
firebase.auth().signOut().then(function() {  
  // Sign-out successful.  
}).catch(function(error) {  
  // An error happened.  
});
```



## Lectura de Datos

La lectura de datos en Firebase se realiza de forma muy sencilla. Para este proyecto se utilizó únicamente el método `.once()` ya que no se requiere un listener de cambios en la base de datos. Para más información acerca de los métodos de lectura puede acceder a:

<https://firebase.google.com/docs/database/web/retrieve-data?hl=es-419>

El siguiente código muestra la lectura en firebase de los datos de contacto:

```
function cargarDatosContacto(){  
  return firebase.database().ref('/Contacto/').once('value').then(function(snapshot) {  
    var direccion = snapshot.val().direccion;  
    var email = snapshot.val().email;  
    var facebook = snapshot.val().faceSj;  
    var instagram = snapshot.val().instaSj;  
    var tel = snapshot.val().numero;  
    var sitioWeb = snapshot.val().sitioWeb;  
    var horario = snapshot.val().horario;  
    $("#direccion").html('<i class="fa fa-home"></i>'+direccion);  
    $("#correo").html('<a href="mailto:'+email+'"><i class="fa fa-envelope"></i>'+email+'</a>');  
    $("#tel").html('<a href="#"><i class="fa fa-phone"></i>'+tel+'</a>');  
    $("#sitioWeb").html('<a href="'+sitioWeb+'"><i class="fa fa-globe"></i>'+sitioWeb+'</a>');  
    $("#facebook").html('<a href="'+facebook+'"><i class="fa fa-facebook"></i>'+facebook+'</a>');  
    $("#instagram").html('<a href="'+instagram+'"><i class="fa fa-instagram"></i>'+instagram+'</a>');  
    $("#imgHorarioAsocia").attr("src", horario);  
  });  
}
```

El argumento de `.ref()` debe ser el nodo padre de la información, en este casos queremos la información que se encuentra en el nodo Contacto en la base de datos. Luego la variable `snapshot`, que se define como argumento de la función javascript contiene la información solicitada. Para obtener el dato de un nodo hijo, basta con `snapshot.val().nodoHijo` y se puede almacenar en una variable. Finalmente, se agregan los datos como código HTML para mostrar en la interfaz gráfica.

## Lectura por Id

```
//Escuchamos cambios en el select
$("#docSelect").on('change', function(){
    docId = $(this).val();
    if (docId != 0){
        cargarDatosDocumento(docId);
    }
})

function cargarDatosDocumento(docId){
    return firebase.database().ref('Documentos/' + docId).once('value').then(function(snapshot) {
        var nombreDoc = snapshot.val().nombre;
        var descripcionDoc = snapshot.val().descripcion;
        var linkDoc = snapshot.val().documento;
        $("#nombreDocEdit").val(nombreDoc);
        $("#descDocEdit").val(descripcionDoc);
        $("#linkDocEdit").val(linkDoc);
    });
}
```

El código anterior muestra la lectura de un documento específico mediante su Id. Este Id se obtiene mediante el documento seleccionado en el CMS para cargar sus datos por editar. Se envía como parámetro a `.ref()` `Documentos/docId`, y `firebase` nos devuelve únicamente el documento en particular. Esto es aplicable para cualquier datos que queramos obtener por Id.

## Ejemplo de “Join”

Firebase por su naturaleza noSQL, no permite hacer Joins en datos, por lo que a continuación se explica una forma de imitar un Join en la base de datos. Note que las personas que pertenecen a la asocia tienen como atributo único “puesto” que hace reference a un puesto en la base de datos mediante su Id.

```
function obtenerAsocia(){
  var personaCount = 1;
  var query = firebase.database().ref("Personas").orderByChild("puesto").startAt("");
  query.once("value").then(function(snapshot) {
    snapshot.forEach(function(childSnapshot) {
      //Join con Puestos
      database.ref('Puestos/'+childSnapshot.val().puesto).once('value', function(puestoSnap) {
        var nombre = childSnapshot.val().nombre;
        var foto = childSnapshot.val().foto;
        var puesto = puestoSnap.val().puesto;
        var telefono = childSnapshot.val().telefono;
        var correo = childSnapshot.val().correo;
        var facebook = childSnapshot.val().facebook;
        var descripcion = childSnapshot.val().descripcion;
        var descripcionAsocia = childSnapshot.val().descripcionAsocia;

        console.log(nombre, foto, puesto, telefono, correo, facebook, descripcion, descripcionAsocia, "html", html);
        personaCount += 1;
      });
    });
  });
}
```

Aquí se introduce la función `orderByChild()` de Firebase, que recibe como parámetro el nombre del hijo por el cual se desean ordenar los resultados. En este caso se hace por puesto, y luego `.startAt()` que permite recibir únicamente aquellos resultados que inicien con el string parámetro de `startAt()`, en este ejemplo se envía un string vacío, ya que los que no tienen el atributo “puesto”, retornan null y no coincidirá con que inicia con “”, por lo que esto nos devuelve únicamente personas con el atributo “puesto”.

Así mismo, se hace un `forEach` al resultado de Firebase (snapshot) que es un array de personas, y `childSnapshot` representa a cada una de ellas. Dentro del `ForEach`, se hace “Join” con Puestos al solicitarle a la base de datos que retorne el puesto de la persona, mediante el atributo “puesto” de cada una que es en realidad el id del puesto correspondiente (ver Lectura por Id).

## Escritura de Datos

Firebase facilita la escritura de datos mediante tres métodos: Push, Update y Set.

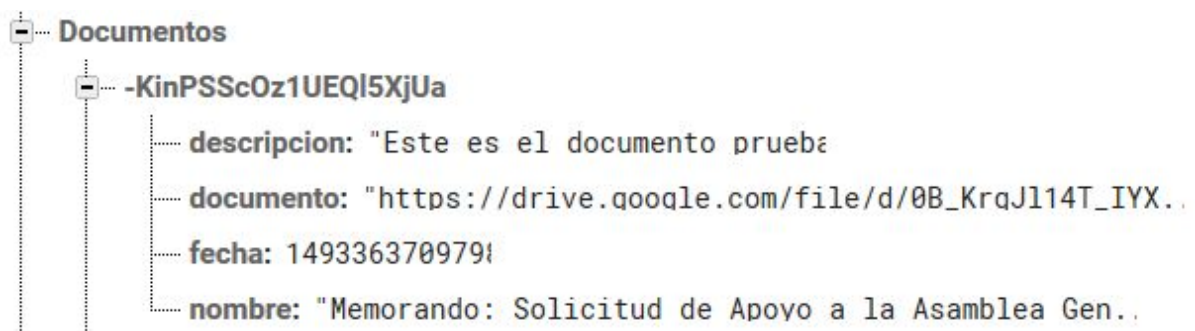
### Push

Este método permite escribir en la base de datos a un nodo en específico, y en caso de que el nodo no exista, se crea. Además, la información ingresada mediante push genera un id automático, por lo que no hay necesidad de estar pendiente del Id de los datos de la base de datos.

```
firebase.database().ref('Documentos/').push({
  documento: linkDoc,
  fecha: moment().valueOf(),
  nombre: nombreDoc,
  descripcion: descDoc
});
```

Este código se puede utilizar para agregar datos a la base de datos. En esta caso particular, lo que lleva a cabo es agregar un nuevo documento. Note que ref() recibe como parámetro el padre del nuevo nodo por agregar, en este caso Documentos. En caso de que dicho nodo padre no exista, firebase lo crea y le agrega el nuevo nodo hijo. Dentro de push, se envía una serie de llave:valor, que corresponde al atributo y valor.

Ese código genera un nuevo documento que se ve de la siguiente manera en la base de datos:



Al realizar esta operación, y observar la base de datos desde la consola de Firebase, se podrá ver resaltado con un color verde por algunos segundos, el documento nuevo agregado.

## Update

Este método se utiliza de manera similar a push. A diferencia de Push, Update no crea un nodo hijo, tampoco un id nuevo. Lo que hace es actualizar los atributos que reciba como parámetro. Les cambia su valor en la base de datos.

Siguiendo con el ejemplo de documentos, al ejecutar el siguiente update se cambia el nombre del documento, la descripción y el link. Note que no se modifica de ninguna manera el atributo fecha.

```
database.ref('Documentos/'+docId).update({
  nombre: nombreDoc,
  descripcion: descDoc,
  documento: linkDoc
});
```

Al realizar esta operación, y si se observa la base de datos desde la consola firebase, se podrá notar un color amarillo resaltando por algunos segundos el documento editado.

Nota: Update no permite almacenar campos null. Para esto en algunos casos se utilizó Set, en lugar de Update.

## Set

La funcionalidad de Set es muy similar a Update, con la diferencia de que set modifica completamente al nodo que se desea alterar. Por ejemplo, en el caso anterior, en documentos se hizo update de tres atributos, si se hubiera utilizado set, el nodo a modificar quedaría solo con esos tres atributos, el otro, como la fecha, se eliminaría por la utilización de set. Además, set permite guardar campos null, a diferencia de update.

```
database.ref('Articulos/'+idArticulo).set({
  nombre: nombre,
  foto: foto,
  precio: precio,
  precioOferta: precioOferta,
  descripcion: descripcion,
  linkEncargo: linkEncargo,
  fechaLimiteEncargo: fechaLimiteEncargo
});
```

En este ejemplo, si el atributo foto se hubiera omitido en set, se eliminaría por completo, a diferencia de haber usado update, que simplemente no lo modifica.

## Eliminación de Datos

La eliminación de datos en Firebase se hace mediante el método `remove()` y se utiliza de manera muy similar a los métodos anteriores. Se tiene una referencia de la base de datos y se le aplica el método. Observe el código para eliminar un artículo.

```
//Borrar Artículo
$("#delArticulo").on('click', function(){
    var selectValue = $("#selectArticulos").val()
    if (selectValue != 0){
        var nombreArticulo = $("#nombreEdit").val();
        if(confirm("Seguro que desea borrar "+nombreArticulo)){
            database.ref('Articulos/').child(selectValue).remove();
            $("#formEditarArticulo")[0].reset();
            $("#descuentoEdit").hide();
            $("#encargoEdit").hide();
            $("#checkOfertaEdit").attr('checked', false);
            $("#checkEncargoEdit").attr('checked', false);
            $("#verFotoArticulo").attr("src", "assets/img/art.jpg");
            cargarArticulos();
        }
    }
});
```

Observe que se hace referencia a Artículos, y a un artículo en específico mediante el ID, y simplemente se le aplica el método `remove()`. De esta manera borramos el artículo seleccionado. Cabe destacar que en caso de que se aplique el `remove` y se observe la base de datos desde la consola Firebase, se resaltará con color rojo el elemento, en este caso un artículo, y posteriormente desaparecerá de la base de datos.

## Otros Recursos

- Documentación oficial de Firebase: <https://firebase.google.com/docs/?hl=es-419>
- Acceso a consola firebase: <https://console.firebase.google.com/>
- Documentación oficial de javascript: <https://www.javascript.com/learn/javascript/strings>
- Documentación oficial de jQuery: <http://api.jquery.com/>
- Documentación oficial de biblioteca jQuery Validation: <https://jqueryvalidation.org/>