REALTIME DATABASE

rortegag

Reglas de seguridad

.read
Escritura
.write
Lectura
.validate
Validar y controlará el formato correcto de los datos en cuanto a atributos y tipo.
.indexOn
obtiene el índice, indicará el hijo en donde se hará la consulta.

Variables

auth
controlará si el usuarios está autentificado con valor del UID del usuario.
Si el valor es nulo no está autentificado.
now
marca de tiempo ahora, con milisegundos EPOCH.
data
hace referencia a los datos relacionados con la petición de lectura o escritura.
Suministra un RuleDataSnapshot con diferentes métodos para encontrar el contenido de un
dato en concreto.
newData
es un *RuleDataSnapshot creado después de que una petición de escritura sea aprobada.
root
*RuleDataSnapshot con el árbol de la base de datos.

Métodos

child()
devuelve un RuleDataSnapshot de una ruta especificada.
parent()
devuelve el nodo padre del actual nodo.
hasChild(childpath)
este método devolverá true o false dependiendo si existe el hijo especificado.
hasChildren([children])
true o false, dependiendo si existe el array de hijos enviados como argumento.
exists()
true o false si el RuleDataSnapShot tiene data.
getPriority()
devuelve la prioridad de la data del snapshot.
isNumber()
true o false dependiendo de si el snapshot es un valor numérico.
isString()
true o false dependiendo de si el snapshot es un string.
isBoolean()
true o false dependiendo de si el snapshot es un boleano.
val()
es un método del método child() y extrae el valor asociado al nodo hijo.

Reglas

Reglas proporcionadas por https://fjmduran.com/

Sin acceso a los datos

```
{
  "rules": {
   ".read": false,
   ".write": false,
  }
}
```

Nadie puede leer ni escribir en la base de datos.

Sin seguridad

```
{
  "rules": {
   ".read": true,
   ".write": true,
  }
}
```

Todos pueden leer y escribir, incluso sin autenticación.

Con autenticación

```
{
  "rules": {
   ".read": "auth.uid!=null",
   ".write": "auth.uid!=null",
  }
}
```

Pueden leer y escribir los usuarios autentificados en Firebase.

Crear una nueva entrada de usuarios para cada "(librería)"

```
{
  "rules": {
   "$libraryId":{
   ".read": "data.child('users').hasChildren([auth.uid])"",
   ".write": "data.child('users').hasChildren([auth.uid])",
   }
  }
}
```

Cada usuario sólo podrá leer y escribir sobre los datos de la librería en la que esté registrado.

Acceso sólo para usuarios autenticados de un dominio

```
{
  "rules": {
  ".read": “auth.token.email.endsWith(‘@domain.com’)”,
  ".write": “auth.token.email.endsWith(‘@domain.com’)”
  }
}
```

Cada usuario sólo podrá leer y escribir si su email tiene un dominio específico.

Para más información: https://firebase.google.com/docs/database

REALTIME DATABASE

rortegag

Reglas de seguridad

Acceso a datos de usuario sólo por el propio usuario

```
{
  "rules": {
   "users": {
    "$uid": {
     ".read": "$uid === auth.uid",
     ".write": "$uid === auth.uid"
    }
   }
  }
}
```

Acceso a los datos del usuario que estarán dentro del nodo con su uid y a su vez dentro del nodo users.

Escritura sólo permitida para el rol admin

```
{
"rules": {
"posts": {
"$uid": {
  ".write": "root.child('users').child('admin').val() === true"
}
}
}
}
```

Validar datos

```
{
  "rules": {
    "posts": {
      "$uid": {
        ".validate": "newData.isString()
        && newData.val().length > 0
        && newData.val().length <= 140"
      }
    }
  }
}
```

Validar si existe atributos

```
{
  "rules": {
    "posts": {
      "$uid": {
        ".validate":
        "newData.hasChildren(['username', 'timestamp'])"
      }
    }
  }
}
```

Validad que la estampa de tiempo no tenga un valor futuro

```
{
  "rules": {
    "posts": {
      "$uid": {
        "timestamp": {
          ".validate": "newData.val() <= now"
        }
      }
    }
  }
}
```

Evita que se pueda eliminar o actualizar

```
{
  "rules": {
    "posts": {
      "$uid": {
        ".write": "!data.exists()"
      }
    }
  }
}
```

Evita que se pueda eliminar

```
{
  "rules": {
    "posts": {
      "$uid": {
        ".write": "newData.exists()"
      }
    }
  }
}
```

o

```
{
 "rules": {
 "posts": {
 "$uid": {
 ".write": "!data.exists() || !newData.exists()"
 }
 }
 }
}
```

Evita crear y borrar

```
{
  "rules": {
    "posts": {
      "$uid": {
        ".write": "data.exists() && newData.exists()"
      }
    }
  }
}
```

Para más información: https://firebase.google.com/docs/database