Base de Datos: Firebase

Estructura

La base de datos tiene una estructura en forma de árbol, cuando se quiere leer un nivel, se tendrán que descargar todos los niveles inferiores a él. Por ello, el diseño de la base de datos se tiene que hacer lo más plano posible.

Del *root* del árbol cuelga:

- Users: representa a un usuario, se identifica con el *ID* de *Google Authentification*. Aporta el nombre del usuario y una URL a su foto.
- Groups: se identifica con el *ID* de *Google Authentification* del usuario. Contiene una lista con los nombres del grupo y dentro de ellos los identificadores de los miembros.
- Friends: se identifica con el *ID* de *Google Authentification*. Contiene una lista con los identificadores de sus amigos.
- Events: contiene una lista con los eventos. Cada evento contiene: nombre, descripción, coordenadas del lugar, fechas, asistentes (y en un futuro puede que más cosas).

Harán falta más tablas: saber que miembros están en un evento. Y creo que ya estarían todas. Ya se irán haciendo conforme hagan falta.

Ejemplo:

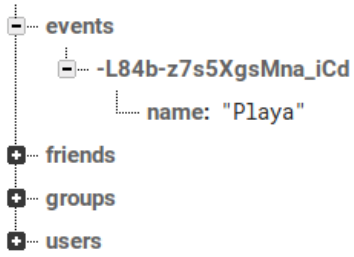

Figure 1: Ejemplo de los eventos, faltarían muchos mas atributos

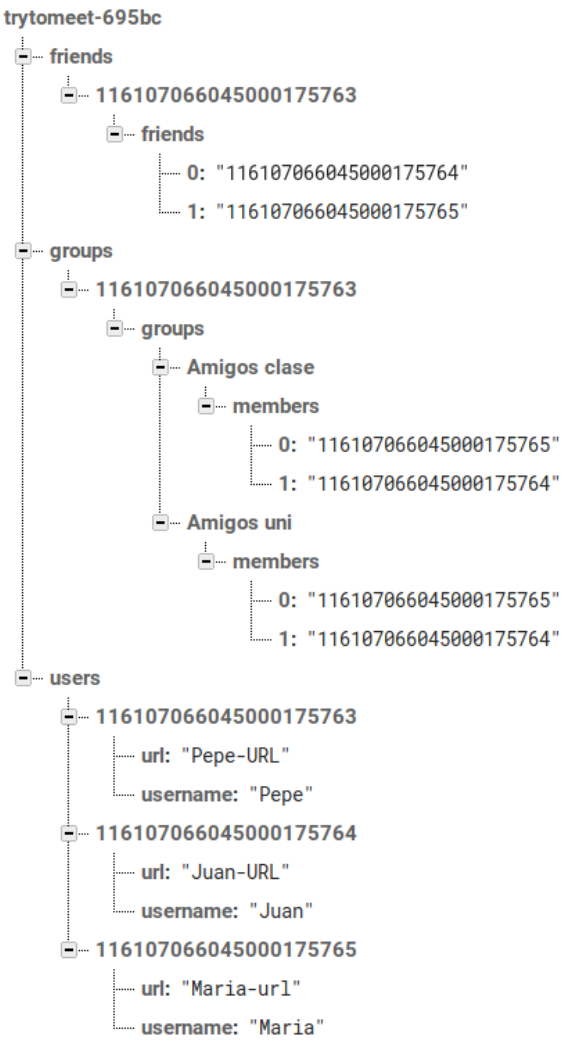

Figure 2: Ejemplo de *friends*, *groups* y *users*

Leer

Para leer se puede hacer de 2 formas:

- Leer cada vez que haya un cambio en el nodo (o niveles inferiores). Método *addListenerValueEvent*
- Leer una única vez. Método *addListenerForSingleValueEvent*

Creo que nunca usaremos el primer caso, pero el uso es igual, solo que se ejecutaría muchas veces.

```
//Obtenemos la referencia a la base de datos
private DatabaseReference mDatabase;
// ...
mDatabase = FirebaseDatabase.getInstance().getReference();
// Accedemos al nodo que pertenece al propio usuario
mDatabase.child("users").child(account.getId()).
        addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                User u = dataSnapshot.getValue(User.class);
                // ..
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                Log.e("Error", "Something bad");
            }
        });
```

Escribir

A la hora de escribir se tiene que tener en cuenta si la modificación que estamos haciendo puede colisionar o no con otra que haga otro usuario.

- Agregar un nodo al árbol (si había algo antes lo destruye): le especificamos la ruta en el árbol (usando el método *child()*) y luego con *setValue()* se coloca.

  ```
  //Obtenemos la referencia a la base de datos
  private DatabaseReference mDatabase;
  // ...
  mDatabase = FirebaseDatabase.getInstance().getReference();
  // Accedemos al nodo que pertenece a un usuario y lo
  // creamos/reemplazamos
  mDatabase.child("users").child(userId).setValue(user);
  // Se puede modificar uno de sus atributos unicamente
  mDatabase.child("users").child(userId).child("username").setValue(name);

  // Nota: userId == account.getId()
  ```

- Agregar datos que pueden colisionar, como el caso de grupos. Cuando se quiere insertar un nuevo grupo para un usuario, hay que descargar los grupos ya existentes, actualizar con el cambio y volver a subir. Pero si durante el procedimiento, hay algún cambio en los grupos, habrá que volver a empezar.

```
//Obtenemos la referencia a la base de datos
private DatabaseReference mDatabase;
// ...
mDatabase = FirebaseDatabase.getInstance().getReference();
mDatabase.child("groups").child(account.getId())
        .runTransaction(new Transaction.Handler() {
            @Override
            public Transaction.Result doTransaction
                (MutableData mutableData) {
                Groups current_groups = mutableData
                        .getValue(Groups.class);
                //HACER SIEMPRE ESTA MODIFICACION
                if(current_groups != null){
                    // Ponemos el nuevo valor
                    current_groups.groups.put("Amigos uni", g);
                }
                else{
                    // Crear el map e insertarlo
                    current_groups = new Groups(....);
                }

                // Hacemos set
                mutableData.setValue(current_groups);
                // Notificamos el cambio
                return Transaction.success(mutableData);
            }

            @Override
            public void onComplete(DatabaseError databaseError,
                boolean b, DataSnapshot dataSnapshot) {
                // Codigo que se ejecutara cuando la BD
                // se haya modificado correntamente
            }
```

- Agregar elementos a una lista donde el identificador tenga que ser único. Para ello, se puede pedir a *Firebase* que genere uno y de esta manera nos aseguramos que no puedan existir colisiones. Es el caso de los eventos, queremos almacenarlos y tener un id con el que identificar que miembros están en un evento.

```
// push() es para crear un nuevo nivel en el nodo actual.
String key = mDatabase.child("events").push().getKey();
Event event = new Event("Playa"); //No relevante para el ejemplo
// ahora ya sabemos en que subnivel guardar la info
mDatabase.child("events").child(key).setValue(event);
```