

## Programación en Móviles Avanzado

### LABORATORIO 13

## Firestore – Creando una Red Social III

CODIGO DEL CURSO:



Alumno(s):	Andrade Chura Mary Carmen					Nota	
Grupo:	C-24 B			Ciclo: V			
Criterio de Evaluación	Excelente (4pts)	Bueno (3pts)	Regular (2pts)	Requiere mejora (1pts)	No accept. (0pts)	Puntaje Logrado	
Crea y elimina objetos almacenados en Storage y RealTimeDatabase						3	
Instala y utiliza componente SDWebImage para carga de imágenes desde URL						3	
Desarrolla adecuadamente los ejercicios propuestos						6	
Realiza observaciones y conclusiones que aporten un opinión crítica y técnica						3	
Es puntual y redacta el informe adecuadamente sin copias de otros autores						2	
Evidencia avance en laboratorio						3	

**I.- OBJETIVOS:**

- Crear una aplicación que haga uso de Storage y RealTimeDataBase
- Eliminar objetos de Storage y RealTimeDataBase

**II.- SEGURIDAD:****Advertencia:**

**En este laboratorio está prohibida la manipulación del hardware, conexiones eléctricas o de red; así como la ingestión de alimentos o bebidas.**

**III.- FUNDAMENTO TEÓRICO:**

Revise sus diapositivas del tema antes del desarrollo del laboratorio.

**IV.- NORMAS EMPLEADAS:**

No aplica

**V.- RECURSOS:**

- En este laboratorio cada alumno trabajará con un equipo con MAC OS.

**VI.- METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:**

- El desarrollo del laboratorio es individual.

**VII.- PROCEDIMIENTO:****ACTIVIDADES:**

1. Abra el proyecto creado en la sesión anterior denominado: **Snapchat**
2. No olvide de ir guardando los cambios a su repositorio Github

**CREANDO LOS SNAPS EN FIREBASE**

3. La estructura que tendrá una cuenta para ver la **lista de snaps** que le han llegado, será la siguiente en el árbol **JSON** de **FIREBASE**. (ESTE SÓLO ES UN EJEMPLO)

snapchat-afd1b

usuarios

hpSjfNDU6VMCwhyBOWiZf9XJm5A2

Snaps

Snap1

Descripcion: "Descripcion 1"

From: "correo1@gmail.com"

ImagenURL: "imagen.jpg"

Snap2

Descripcion: "Descripcion 2"

From: "correo2@gmail.com"

ImagenURL: "imagen2.jpg"

email: "nuevo1@gmail.com"

Lista de Snaps recibidos

Datos de Snaps

4. Cree 2 variables de clase en **elegirUsuarioViewController.swift**, para representar la dirección de la **imagen** y la **descripción** o mensaje del **snap**.

```
class ElegirUsuarioViewController:
    UIViewController, UITableViewDataSource, UITableViewDelegate {

    @IBOutlet weak var listaUsuarios: UITableView!
    var usuarios:[Usuario] = []
    var imagenURL = ""
    var descrip = ""
```

5. Estos datos serán enviados desde **ImagenViewController.swift**, el controlador que en su diseño tiene la **imagen** y la **descripción**.
6. Diríjase a dicho controlador y modifique la función **elegirContactoTapped** y modifique como se muestra tal como se muestra en las siguientes líneas  
 Línea 38 y 39: crea variable cargarImagen en dos líneas  
 Línea 46 a 52: función **downloadURL** permite ver si imagen subió a Storage  
 Línea 53: se configura segue para enviar la ruta de la imagen subida a Storage con **url.absoluteString**  
 Línea 57 a 76: Comente estas líneas, ya que se controlará con la función **downloadURL**

```

    @IBAction func elegirContactoTapped(_ sender: Any) {
35         self.elegirContactoBoton.isEnabled = false
36         let imagenesFolder = Storage.storage().reference().child("imagenes")
37         let imagenData = imageView.image?.jpegData(compressionQuality: 0.50)
38         let cargarImagen = imagenesFolder.child("\(NSUUID().uuidString).jpg")
39         cargarImagen.putData(imagenData!, metadata: nil) { (metadata, error) in
40             if error != nil {
41                 self.mostrarAlerta(titulo: "Error", mensaje: "Se produjo un error al subir la
                    imagen. Verifique su conexion a internet y vuelva a intentarlo.", accion:
                        "Aceptar")
42                 self.elegirContactoBoton.isEnabled = true
43                 print("Ocurrio un error al subir imagen: \(error) ")
44                 return
45             }else{
46                 cargarImagen.downloadURL(completion: { (url, error) in
47                     guard let enlaceURL = url else{
48                         self.mostrarAlerta(titulo: "Error", mensaje: "Se produjo un error al
                            obtener informacion de imagen.", accion: "Cancelar")
49                         self.elegirContactoBoton.isEnabled = true
50                         print("Ocurrio un error al obtener informacion de imagen \
                            (error)")
51                         return
52                     }
53                     self.performSegue(withIdentifier: "seleccionarContactoSegue", sender:
                        url?.absoluteString )
54                 })
55             }
56         }
    }

```

```

57      /*
58      let alertaCarga = UIAlertController(title: "Cargando Imagen ...", message: "0%",
        preferredStyle: .alert)
59      let progresoCarga : UIProgressView = UIProgressView(progressViewStyle: .default)
60
61      cargarImagen.observe(.progress) { (snapshot) in
62          let porcentaje = Double(snapshot.progress!.completedUnitCount)
63              / Double(snapshot.progress!.totalUnitCount)
64          print(porcentaje)
65          progresoCarga.setProgress(Float(porcentaje), animated: true)
66          progresoCarga.frame = CGRect(x: 10, y: 70, width: 250, height: 0)
67          alertaCarga.message = String(round(porcentaje*100.0)) + " %"
68          if porcentaje>=1.0 {
69              alertaCarga.dismiss(animated: true, completion: nil)
70          }
71      }
72      let btnOK = UIAlertAction(title: "Aceptar", style: .default, handler: nil)
73      alertaCarga.addAction(btnOK)
74      alertaCarga.view.addSubview(progresoCarga)
75      present(alertaCarga, animated: true, completion: nil)
76      */
77  }

```

7. Modifique la función **prepare for segue** de la siguiente manera(envió de variables)

```

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let siguienteVC = segue.destination as! ElegirUsuarioViewController
    siguienteVC.imagenURL = sender as! String
    siguienteVC.descrip = descriptionTextField.text!
}

```

8. En **ElegirUsuarioViewController.swift**, implemente la función **didSelectRowAt** para crear los **snap**s en la base de datos de **FIREBASE**.

```

func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    let usuario = usuarios[indexPath.row]
    let snap = ["from" : usuario.email, "descripcion" : descrip, "imagenURL" : imagenURL]

    Database.database().reference().child("usuarios").child(usuario.uid).child("snaps")
        .childByAutoId().setValue(snap)
}

```

9. Cree un usuario nuevo denominado [usuario2@empresa.com](mailto:usuario2@empresa.com) con su respectiva contraseña.
10. Ejecute la aplicación e intente enviar un **snap** del [usuario2@empresa.com](mailto:usuario2@empresa.com) a [usuario1@empresa.com](mailto:usuario1@empresa.com) , verifique los cambios en su base de datos en **FIREBASE**.

#### 10.1. Iniciando Sesión:



Usuario:

usuario2@empresa.com

Contraseña:

.....

Iniciar Sesión



4:09

INICIO DE SESION

Usuario:

usuario2@empresa.com

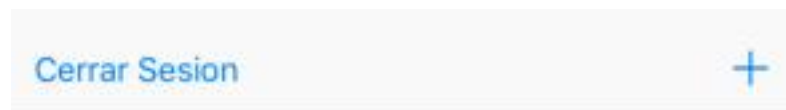
Contraseña:

.....

Iniciar Sesión

Iniciar Google

## 10.2. Creando un nuevo Snap



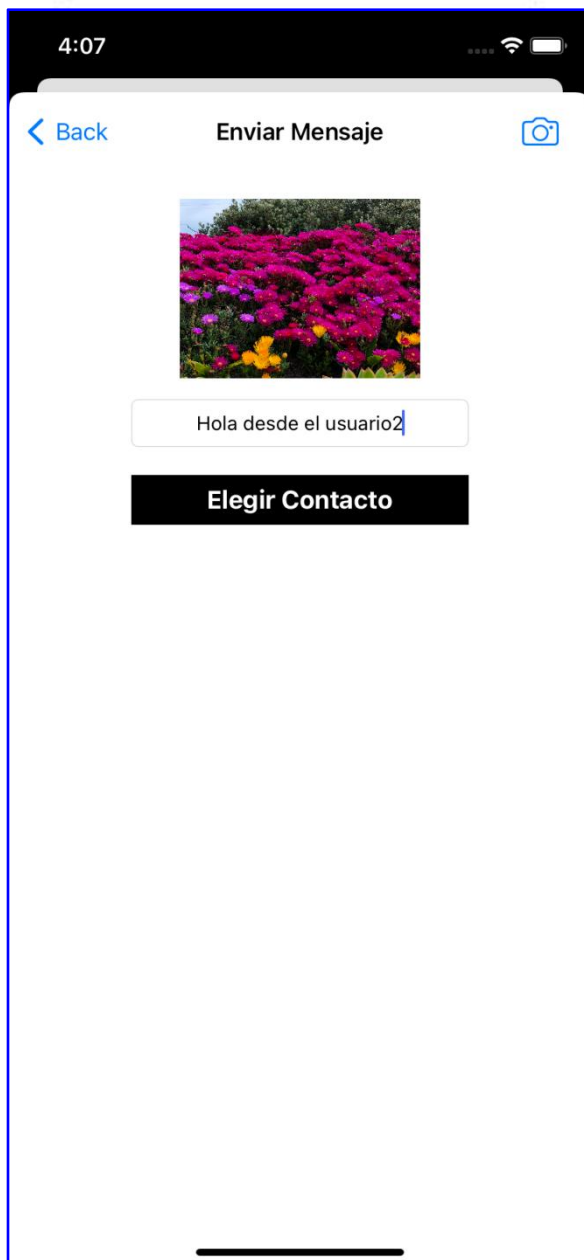
Cerrar Sesion

## 10.3. Seleccionando la imagen y la descripción del snap



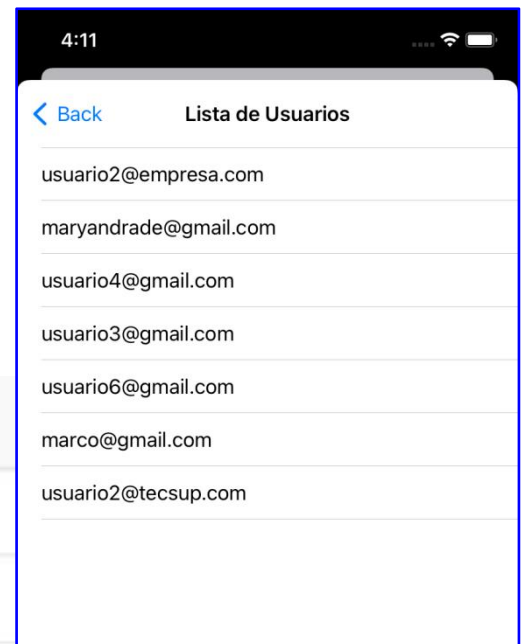
Primer mensaje

Elegir Contacto

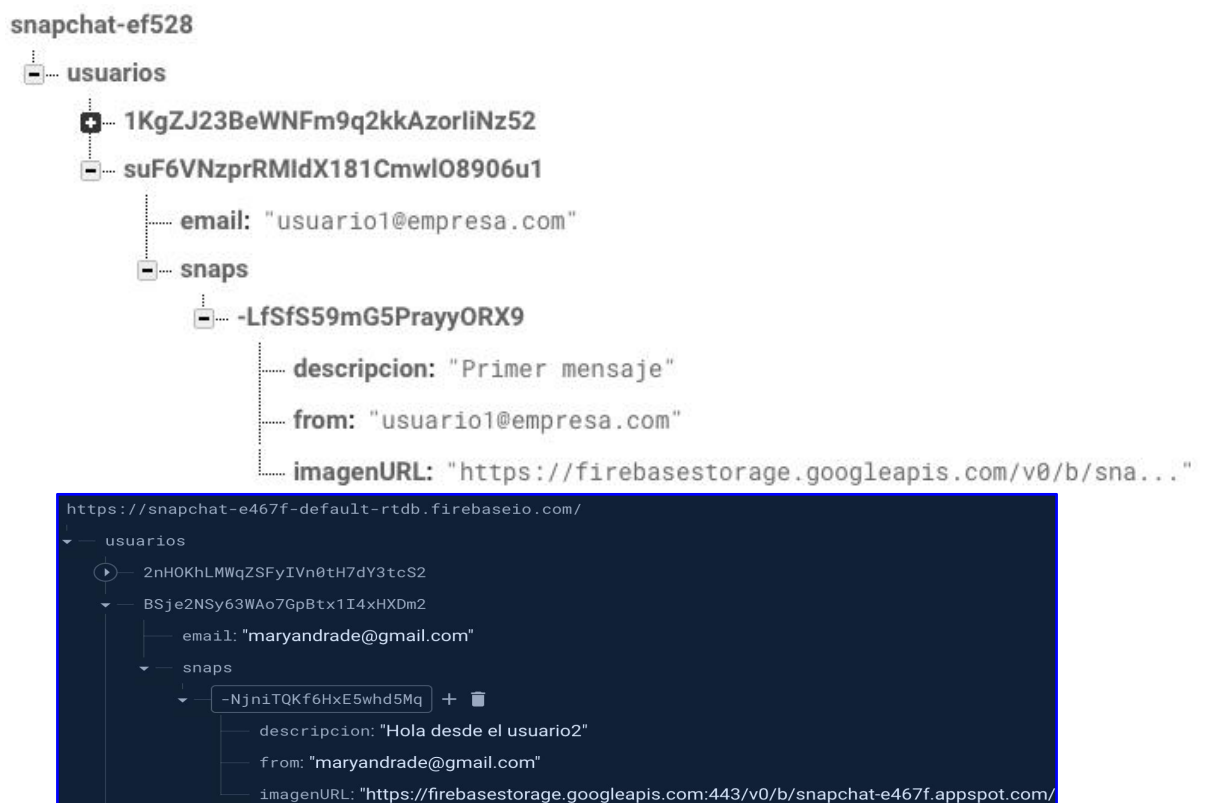




- 10.4. Eligiendo el usuario destino: Seleccionaré a [usuario1@empresa.com](mailto:usuario1@empresa.com) (En su caso seleccione el otro usuario con el cual no inicio sesión)



- 10.5. Verificando los cambios en la base de datos de **FIREBASE**.



11. Indique los detalles más importantes del código implementado

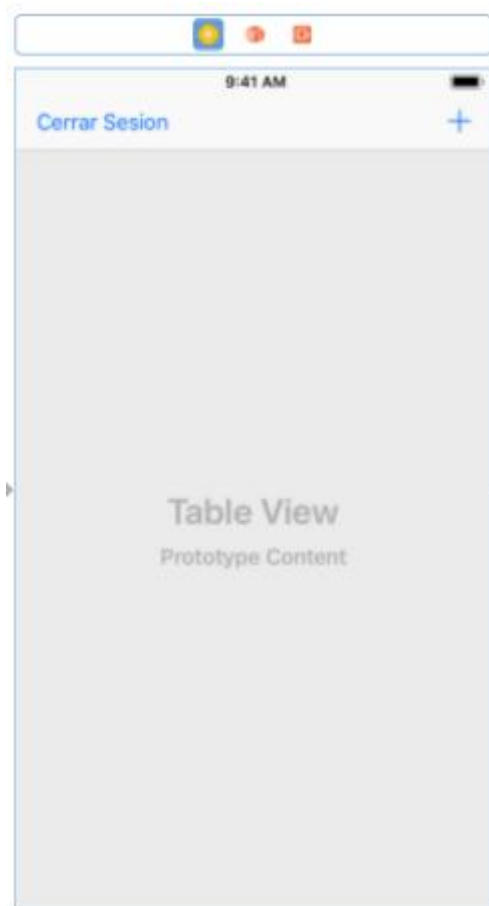
Nos permite al usuario seleccionar una imagen, cargarla a Firebase Storage y luego continuar hacia otra pantalla, presumiblemente para seleccionar un contacto al que se enviará esta imagen.

## LISTANDO LOS SNAPS ENTRANTES

12. Cree un nuevo **modelo** (análogo a **Usuario.swift** en el folder **Modelos**) esta vez, con el nombre de **Snap.swift**, será el modelo que representará a un **snap**.
13. En dicho modelo, inyecte el siguiente código.



14. Diríjase al **main.storyboard**. En la vista controlada por **SnapsViewController** arrastre un elemento **TableView**.
  - 14.1. Añada **Constraints** para:
    - 14.1.1. Que el tableView tenga margen 0(cero) en todos los lados
  - 14.2. En la propiedad "**Adjust Scroll View Insets**" del ViewController, quite el Check.



15. Divida la pantalla y cree una conexión de tipo **Outlet** del **TableView** con su clase controladora con nombre de variable de clase "**tablaSnaps**". Y agregue los protocolos para manejar el **tableView** y las funciones propias sugeridas por el IDE(numberOfRowsInSection, cellForAt). Cree además una instancia de clase **Snap.swift** para poder utilizarla e importe el paquete **Firestore**  
Línea 10: Importar Firestore  
Línea 12: Agregar protocolos UITableViewDelegate y UITableViewDataSource  
Línea 14: instancia de clase Snap  
Línea 16 a 22: Funciones sugeridas por los protocolos implementados



```

9  import UIKit
10 import Firebase
11
12 class SnapsViewController: UIViewController,UITableViewDelegate,UITableViewDataSource {
13     @IBOutlet weak var tablaSnaps: UITableView!
14     var snaps:[Snap] = []
15
16     func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
17         code
18     }
19
20     func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
        UITableViewCell {
21         code
22     }

```

16. Implemente la función **viewDidLoad** para delegar funciones al **tableView** y visualizar los **snaps** de entrada al usuario que ha iniciado sesión en la tabla **tablaSnaps** mencionado.

```

29     override func viewDidLoad() {
30         super.viewDidLoad()
31         tablaSnaps.delegate = self
32         tablaSnaps.dataSource = self
33
34         Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid)!).child("snaps").observe(DataEventType.childAdded, with: { (snapshot) in
35             let snap = Snap()
36             snap.imagenURL = (snapshot.value as! NSDictionary)["imagenURL"] as! String
37             snap.from = (snapshot.value as! NSDictionary)["from"] as! String
38             snap.descrip = (snapshot.value as! NSDictionary)["descripcion"] as! String
39             self.snaps.append(snap)
40             self.tablaSnaps.reloadData()
41         })

```

17. Modifique la función **numberOfRowsInSection** para retornar el número de ítems que tenga nuestro arreglo de **Snaps**.
18. Modifique la función **cellForRowAt** para imprimir en cada celda, el usuario que envía el **snap**

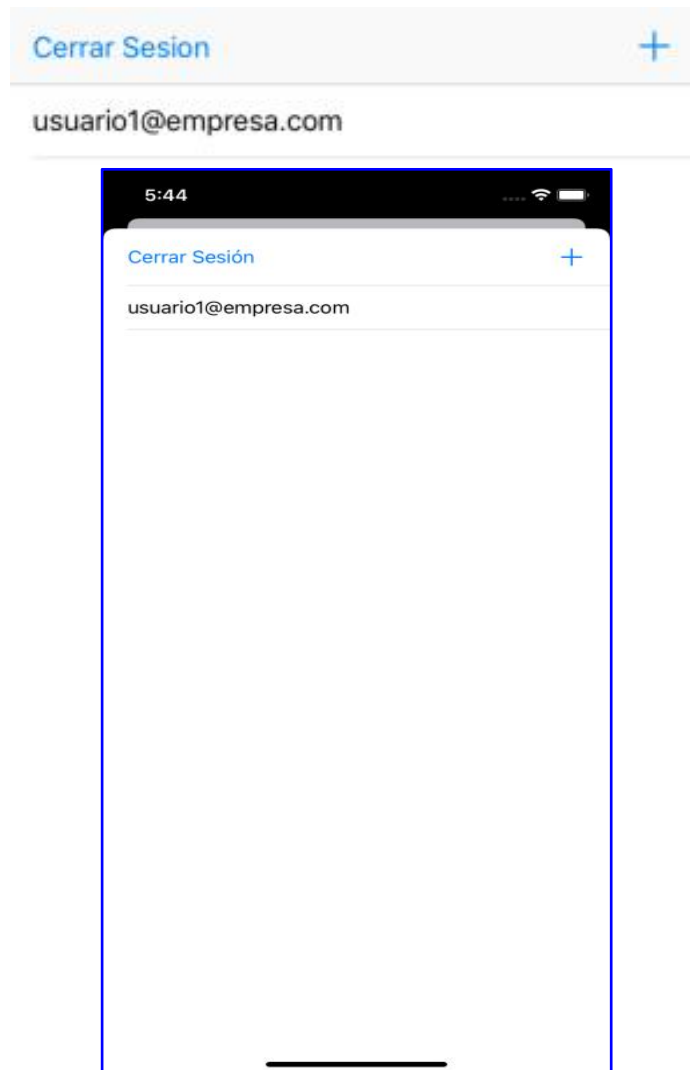
```

func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    return snaps.count
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell {
    let cell = UITableViewCell()
    let snap = snaps[indexPath.row]
    cell.textLabel?.text = snap.from
    return cell
}

```

19. Ejecute la aplicación e intente iniciar sesión con algún **usuario** que tenga **snaps** de entrada ([usuario1@empresa.com](mailto:usuario1@empresa.com)). Verifique los resultados (A ESTE PUNTO, NO SE PODRÁ VISUALIZAR LA INFORMACIÓN DE UN SNAP, pero debe listarse los destinatarios).

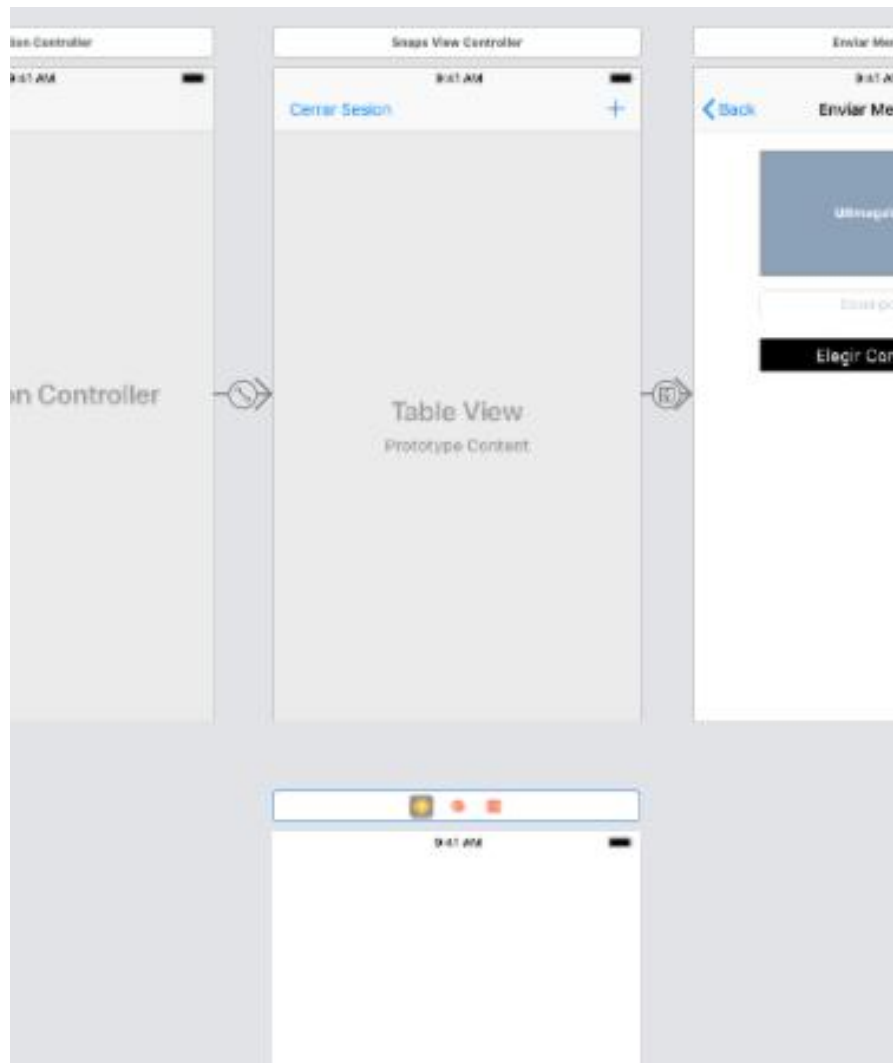


20. Indique los detalles más importantes del código implementado

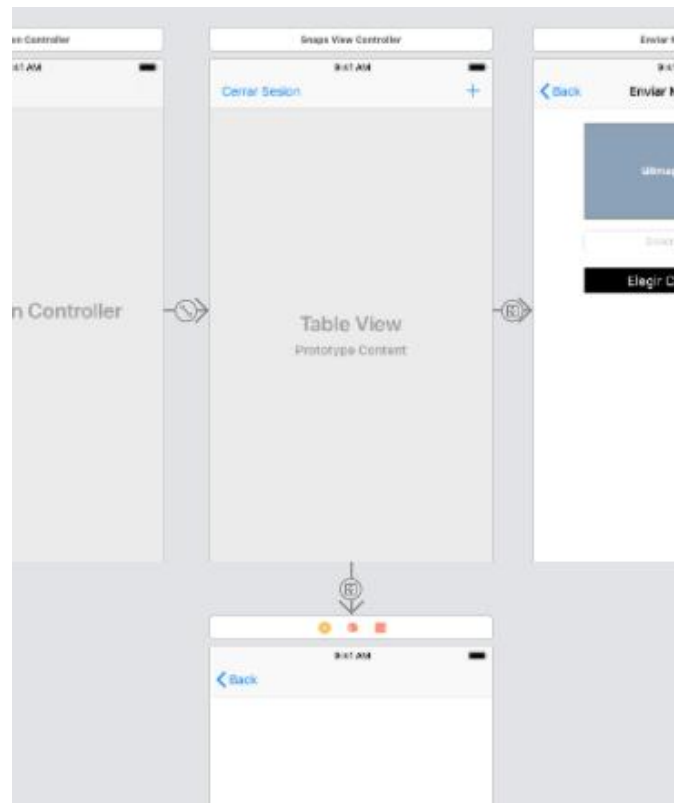
Lo que estamos haciendo es que el controlador maneje la visualización de snaps recibidos por el usuario actual y se encarga de cargarlos en una tabla para su visualización, en este caso hemos enviado un span desde el usuario a otro y este se puede ver cuando iniciamos sesión con el otro usuario, nos muestra el nombre de usuario del que nos envió el snap.

## VIENDO UN SNAP

21. Diríjase al **Main.storyboard** y arrastre un **ViewController** nuevo.

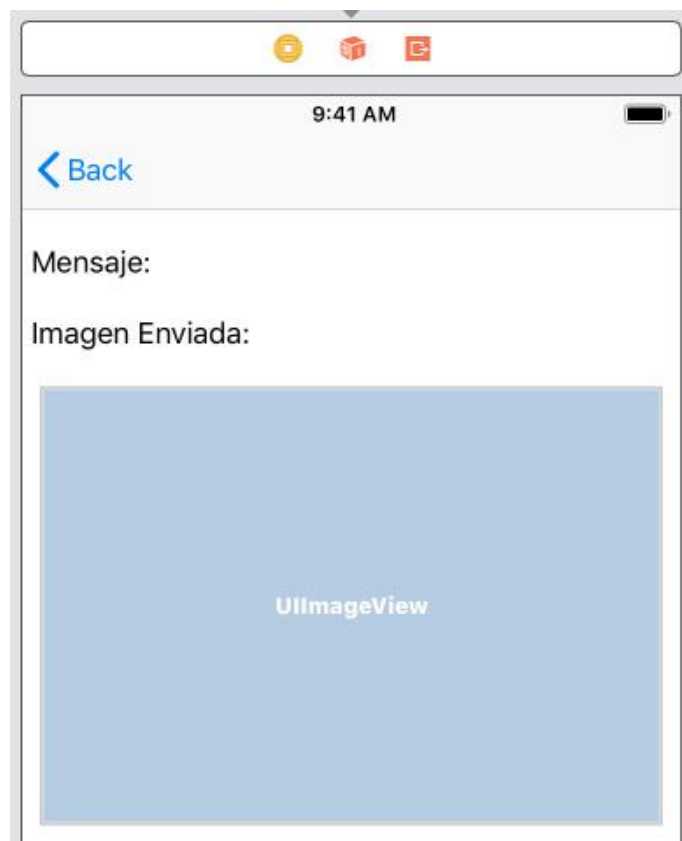


22. Cree una nueva conexión (**Segue**) de tipo “**show**” del **SnapsViewController** (ícono amarillo) al nuevo **ViewController**.



23. Al **Segue** creado, asígnele como nombre de identificador: “**versnapsegue**”

24. Cree el siguiente diseño en el nuevo **ViewController** insertado:



25. Agregue un nuevo archivo de tipo **Cocoa Touch** dentro de la carpeta **ViewControllers**, de nombre **VerSnapViewController.swift** (sub-clase de **UIViewController**) y asícielo al **ViewController** creado en el paso anterior.

26. Cree 2 conexiones de tipo **outlet** del **VerSnapViewController** (vista) a su clase.

26.1. **ImageView** Nombre: **imageView**

26.2. **Label** Nombre: **lblMensaje** (del label Mensaje:)

27. Adicionalmente, cree un objeto del modelo **Snap** con el nombre **snap** como objeto de clase. En el método **viewDidLoad** agregue código para mostrar el mensaje enviado en el **label** previamente referenciado

28. Elimine el código que no se utiliza en **VerSnapViewController**.

29. El código resultante debe ser el siguiente

```
class VerSnapViewController: UIViewController {
    @IBOutlet weak var lblMensaje: UILabel!
    @IBOutlet weak var imageView: UIImageView!
    var snap = Snap()

    override func viewDidLoad() {
        super.viewDidLoad()
        lblMensaje.text = "Mensaje: " + snap.descrip
    }
}
```

30. Regrese a **SnapsViewController.swift** (código) implemente la función **didSelectRowAt**,

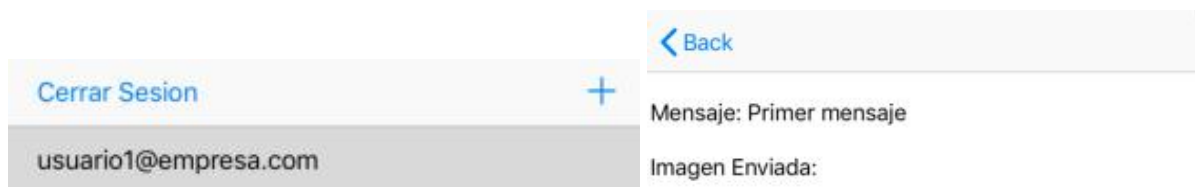
```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    let snap = snaps[indexPath.row]
    performSegue(withIdentifier: "versnapsegue", sender: snap)
}
```

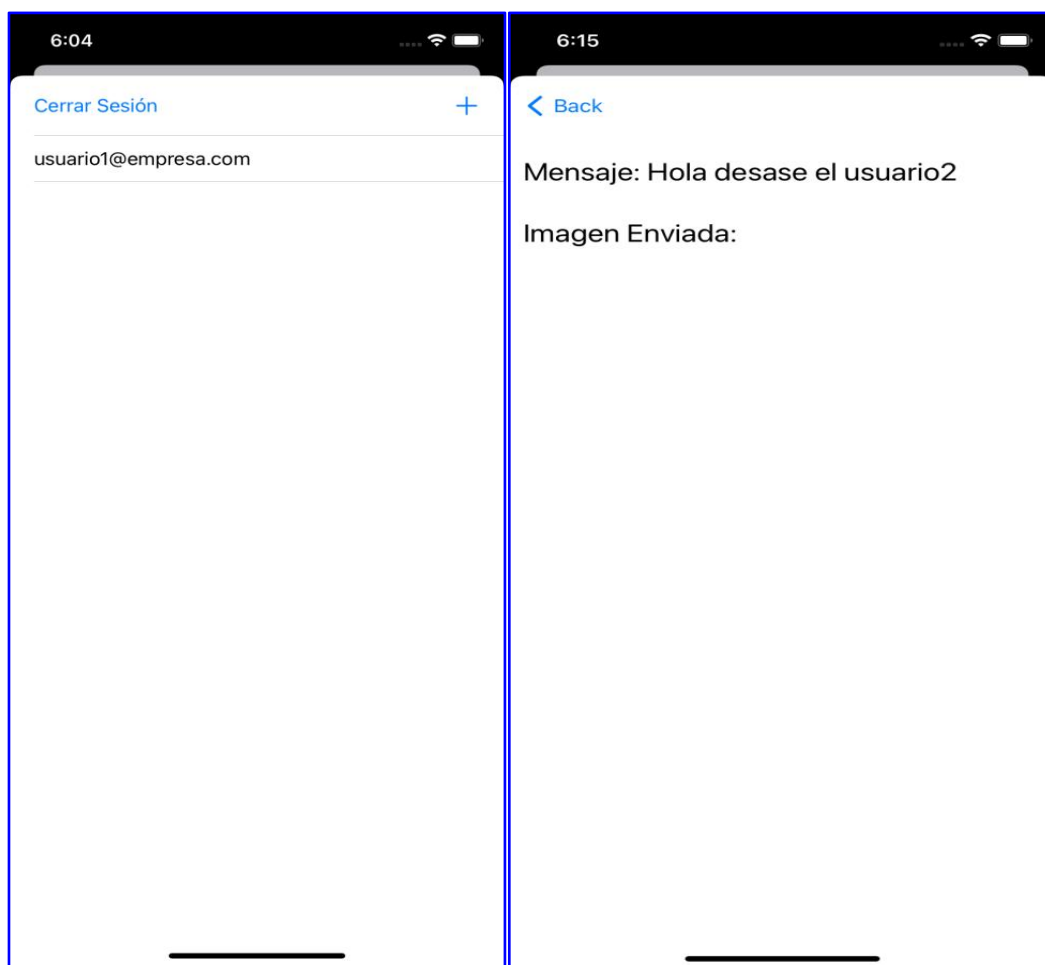
31. A continuación, implemente la función **prepare** y escriba el código que se muestra en la siguiente figura.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "versnapsegue" {
        let siguienteVC = segue.destination as! VerSnapViewController
        siguienteVC.snap = sender as! Snap
    }
}
```

32. A este punto la aplicación debería **mostrar la descripción de un snap** de entrada seleccionado.

33. Ejecute la aplicación y compruebe que puede ver el texto(**descripción**) de un **snap**.





34. Anote los detalles más importantes del código implementado.

Se utiliza para manejar la transición entre vistas cuando se selecciona una fila en la tabla de `SnapsViewController`. La función `didSelectRowAt` se activa cuando el usuario toca una celda en la tabla, y esta función, a su vez, desencadena la ejecución del método `prepare(for segue:)`. Aquí se comprueba si el identificador de la transición es "versnapsegue", y si es así, se obtiene la instancia del `VerSnapViewController` y se le asigna el objeto `Snap` correspondiente para mostrar los detalles del snap seleccionado.

### VISUALIZANDO LA IMAGEN CON `SDWebImage`

35. Para visualizar la imagen, necesitaremos una librería de nombre "**`SDWebImage`**" que será descargada por **Cocoapods**.
36. Si gusta ver la documentación de la librería solo diríjase a la página oficial de **Cococapods** y busque la librería con su nombre.

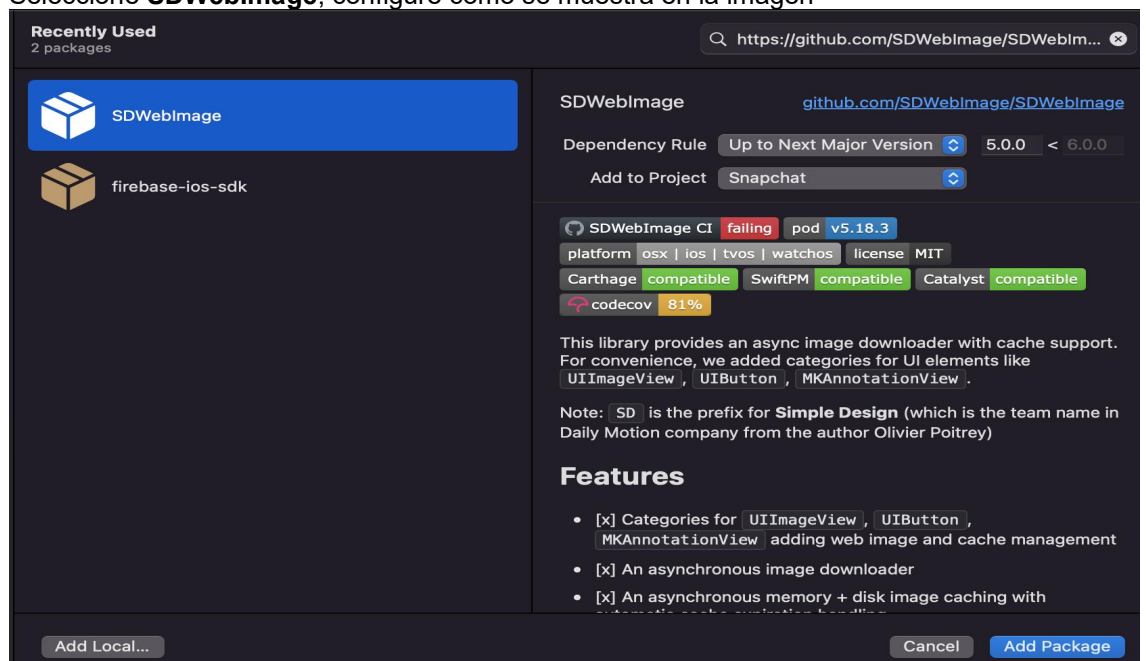


# SDWebImage 4.3.3

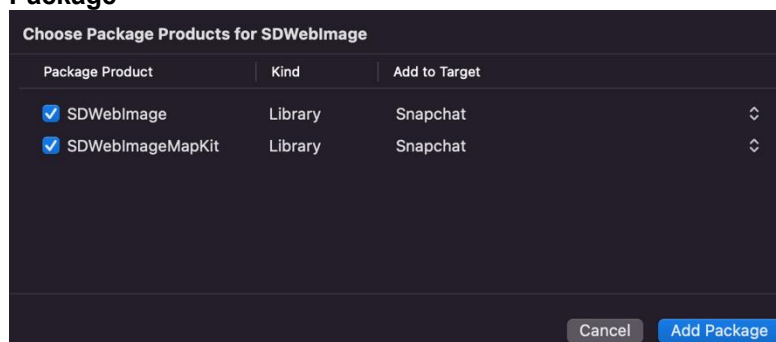


37. En Xcode vaya al menú **File Add Packages**

- Busque el repositorio: <https://github.com/SDWebImage/SDWebImage>
- Seleccione **SDWebImage**, configure como se muestra en la imagen



- Haga clic en **Add Package**
- Espere a que muestre la lista de paquetes a agregar , seleccione los que sugiera y clic en **Add Package**



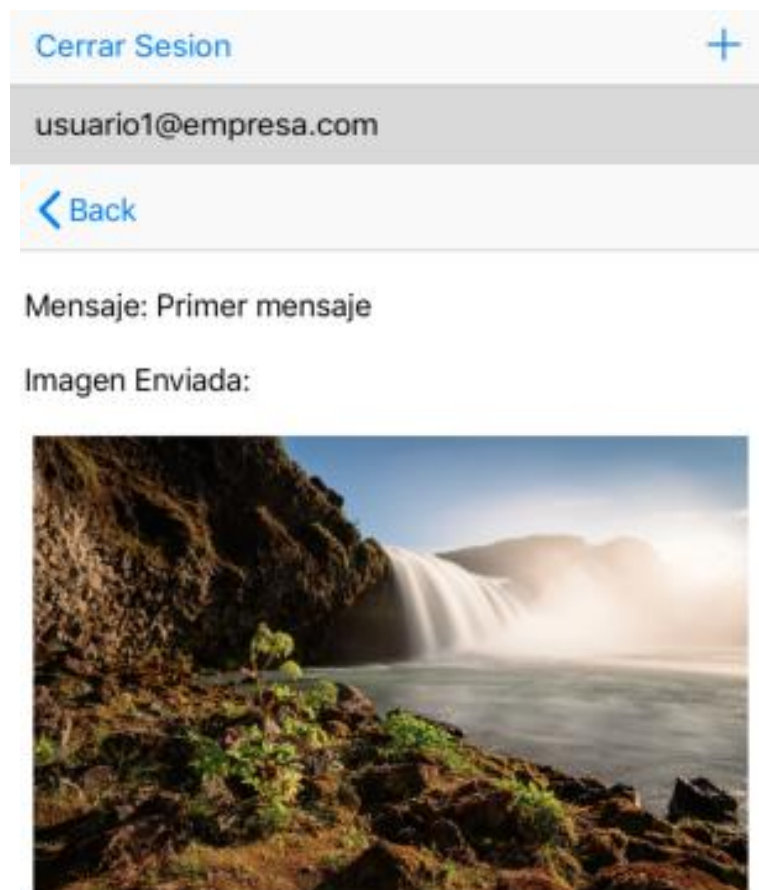
- Espere a que se reconstruya el proyecto

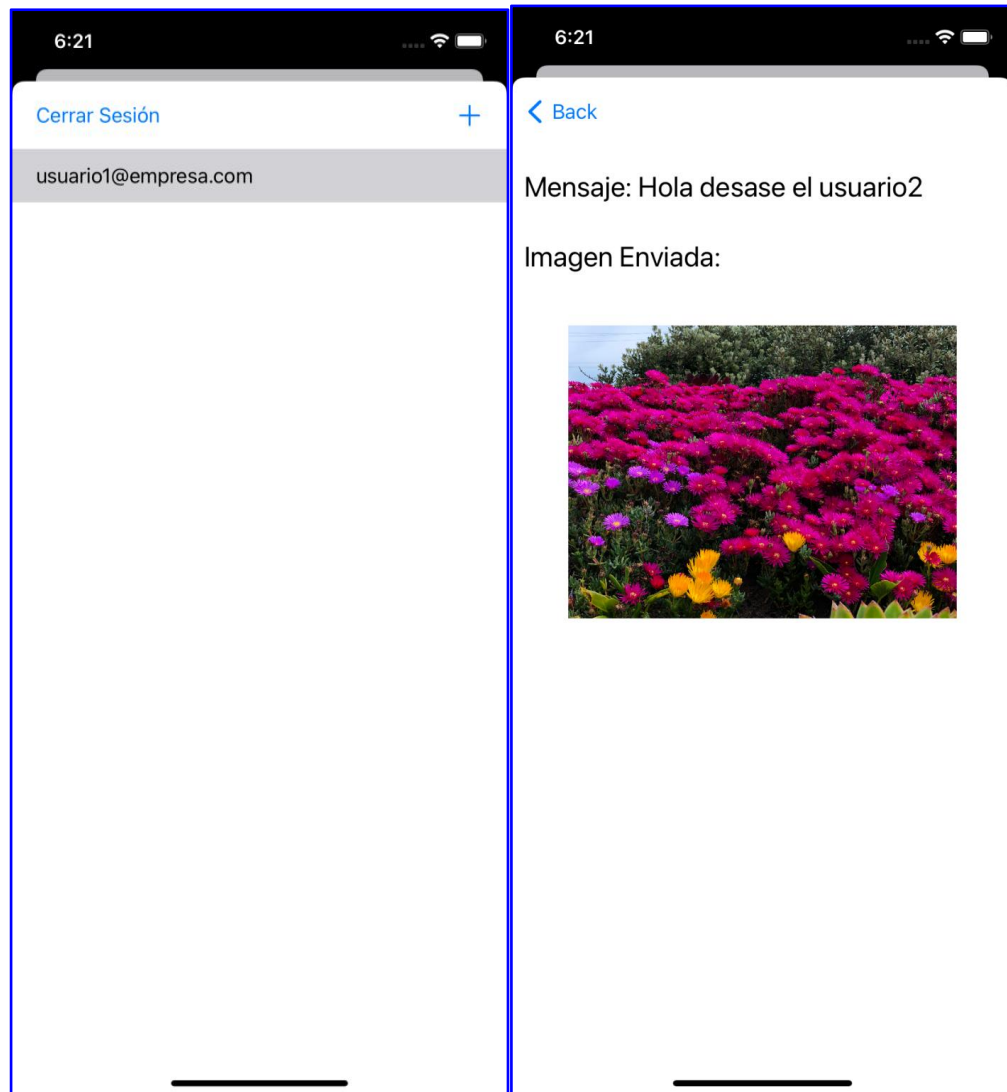
38. Ahora, diríjase a **VerSnapViewController.swift**, importe la librería **SDWebImage** y escriba el código para descargar la imagen y colocarla en el **UIImageView**  
 Línea 10: Importación de clase SDWebImage

Línea 20: Uso de método para carga de imagen alojada en un enlace web

```
9  import UIKit
10 import SDWebImage
11
12 class VerSnapViewController: UIViewController {
13     @IBOutlet weak var lblMensaje: UILabel!
14     @IBOutlet weak var imageView: UIImageView!
15     var snap = Snap()
16
17     override func viewDidLoad() {
18         super.viewDidLoad()
19         lblMensaje.text = "Mensaje: " + snap.descrip
20         imageView.sd_setImage(with: URL(string: snap.imagenURL), completed: nil)
21     }
```

39. Ejecute la aplicación y verifique que al visualizar el **snap**, se muestre la imagen y la descripción





### ELIMINANDO EL SNAP CUANDO ES VISTO

43. Necesitamos el **ID** del **snap** para eliminarlo.
44. Cree el atributo necesario en el modelo **Snap** para hacer referencia a su identificador.

```
import Foundation
class Snap{
    var imagenURL = ""
    var descrip = ""
    var from = ""
    var id = ""
}
```

45. Diríjase al archivo **SnapsViewController.swift (código)**, y agregue el código para asignar el identificador del **snap** a su objeto respectivo.  
Línea 41: Obtener **key** generado para cada snap enviado

```

32     override func viewDidLoad() {
33         super.viewDidLoad()
34         tablaSnaps.delegate = self
35         tablaSnaps.dataSource = self
36
37         Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid)!).child("snaps")
38         observe(DataEventType.childAdded, with: { (snapshot) in
39             let snap = Snap()
40             snap.imagenURL = (snapshot.value as! NSDictionary)["imagenURL"] as! String
41             snap.from = (snapshot.value as! NSDictionary)["from"] as! String
42             snap.descrip = (snapshot.value as! NSDictionary)["descripcion"] as! String
43             snap.id = snapshot.key
44             self.snaps.append(snap)
45             self.tablaSnaps.reloadData()
46         })
47     }
48 }

```

46. En el archivo **"VerSnapViewController.swift"**, importe la librería **Firestore** e implemente la función **viewWillDisappear** para eliminar el **snap** una vez leído.

```

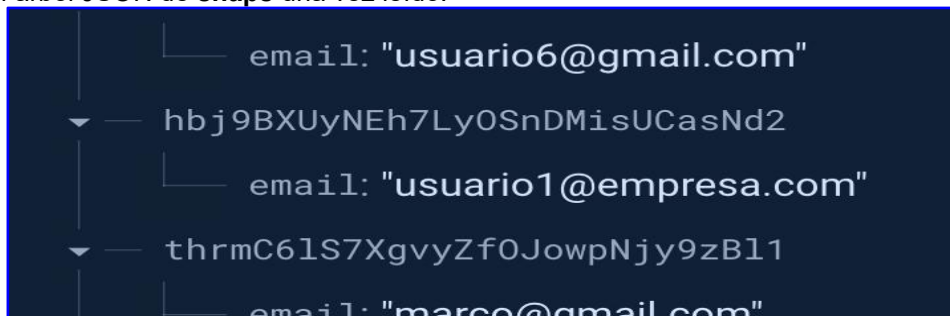
override func viewWillDisappear(_ animated: Bool) {

    Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid)!).child("snaps").child(snap.id).removeValue()

}

```

47. Ejecute la aplicación y verifique en su base de datos en la página de **FIREBASE** que un **snap** es eliminado del árbol JSON de **snaps** una vez leído.



48. Anote los detalles más importantes del código implementado

El método, **viewWillAppear**, se activa justo antes de que la vista del **VerSnapViewController** aparezca en la pantalla. En este caso, realiza una operación para eliminar un **snap** específico de la base de datos de **Firestore**.

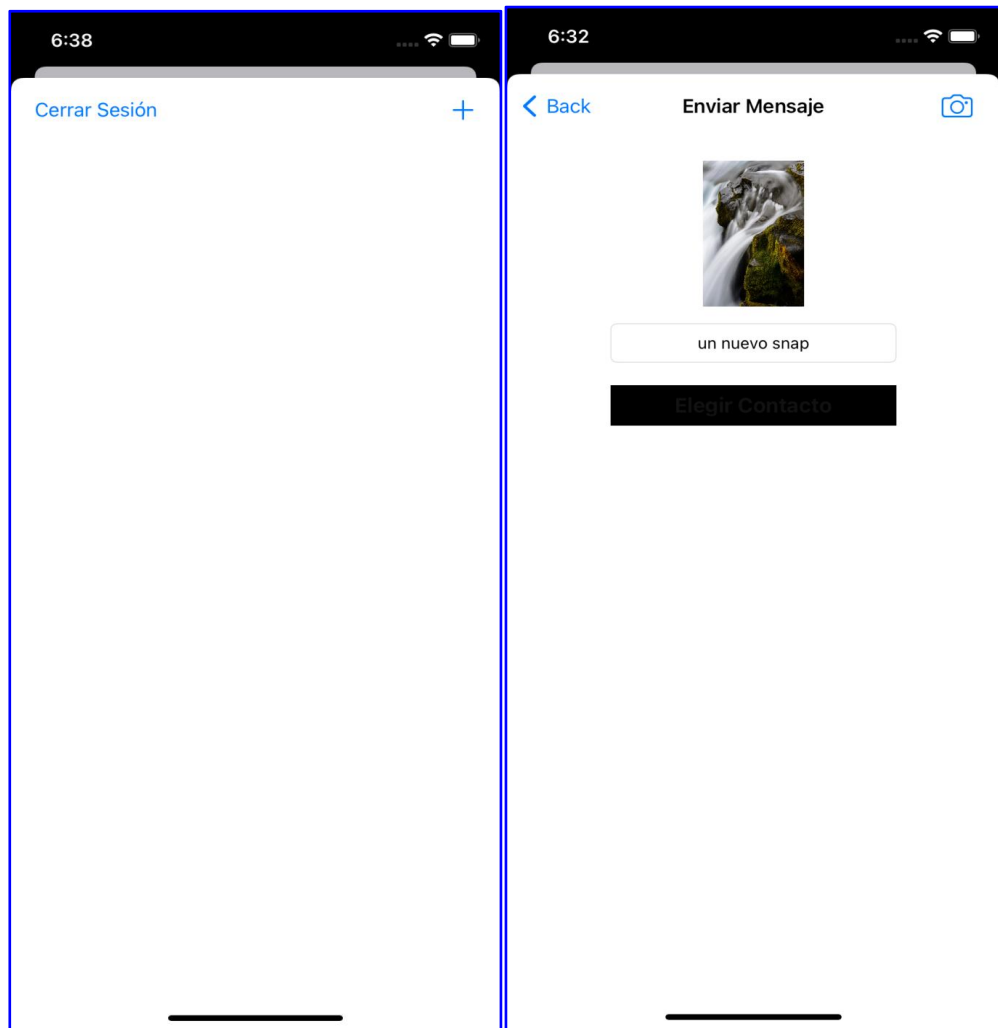
49. Para actualizar el **UITableView** donde se encuentra la lista de **snaps**, no basta con hacer un **ReloadData**. Tenemos que hacer escucha del evento **childRemoved** propio de **Firestore**.

50. Diríjase al archivo **SnapsViewController**, agregue el siguiente código en la función **viewDidLoad** para actualizar el **TableView** una vez se haya borrado el **snap**.

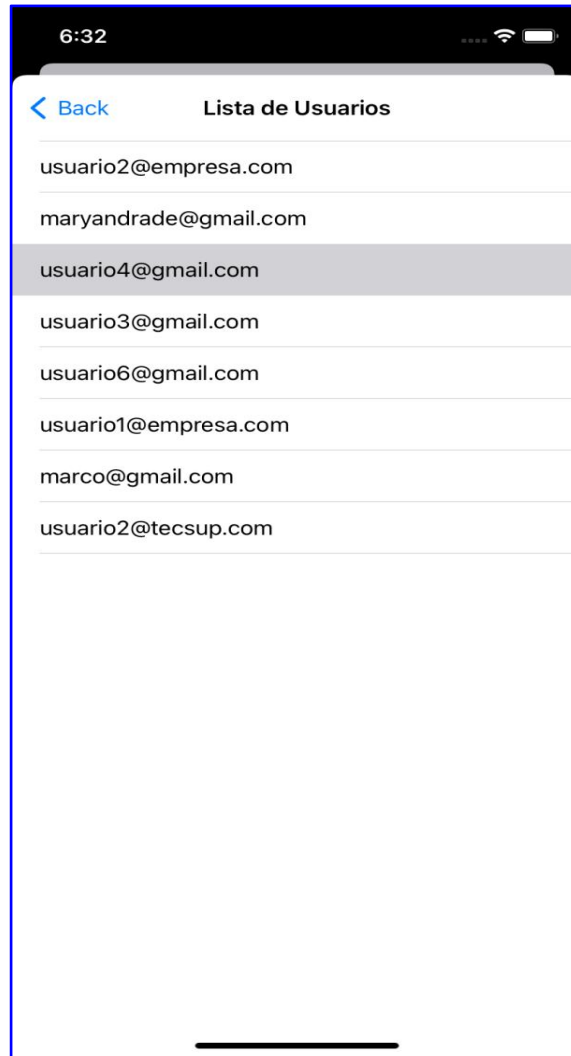
Líneas 45 a 54: Agregue líneas para eliminar el dato de snap ya leído

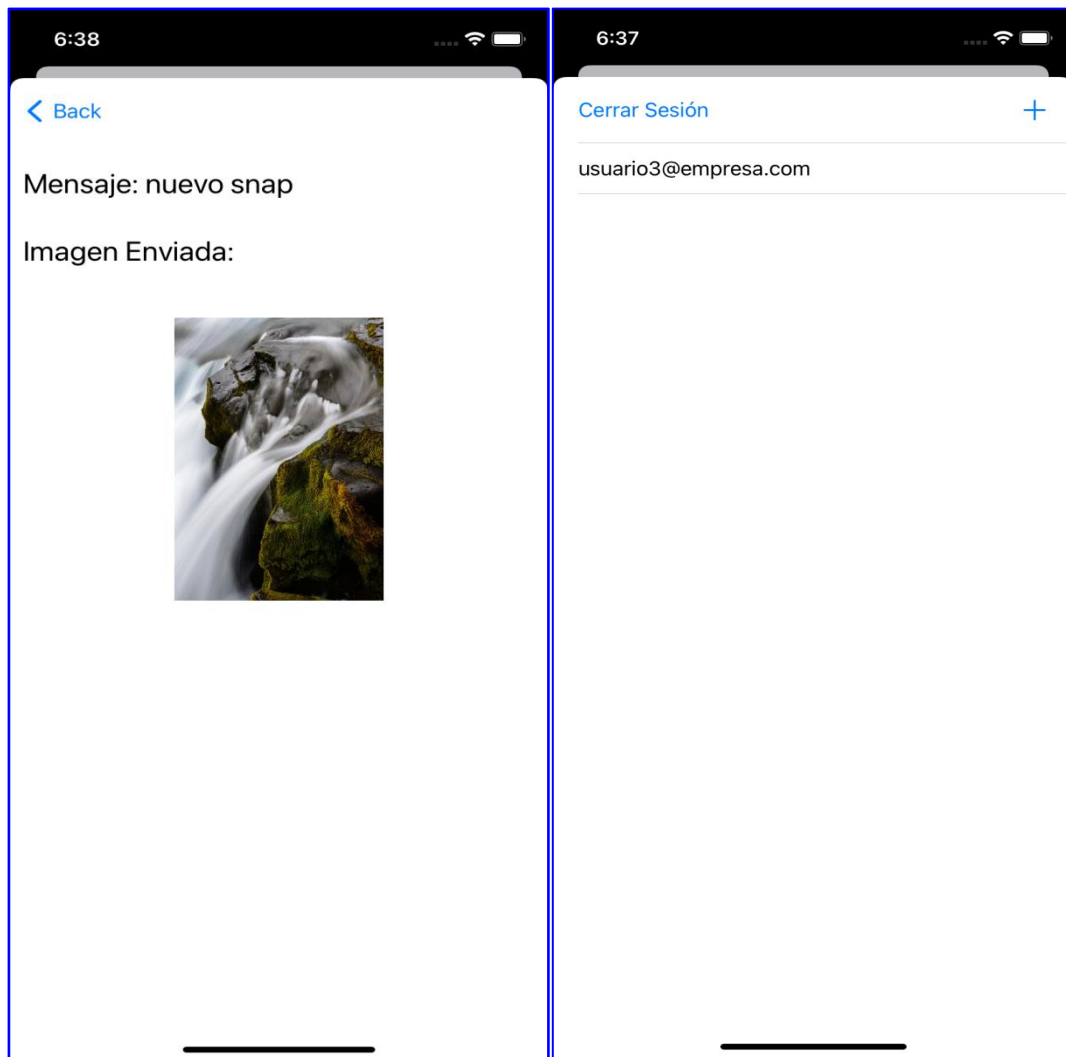
```
32     override func viewDidLoad() {
33         super.viewDidLoad()
34         tablaSnaps.delegate = self
35         tablaSnaps.dataSource = self
36
37         Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid)!).child("snaps").observe(DataEventType.childAdded, with: { (snapshot) in
38             let snap = Snap()
39             snap.imagenURL = (snapshot.value as! NSDictionary)["imagenURL"] as! String
40             snap.from = (snapshot.value as! NSDictionary)["from"] as! String
41             snap.descrip = (snapshot.value as! NSDictionary)["descripcion"] as! String
42             snap.id = snapshot.key
43             self.snaps.append(snap)
44             self.tablaSnaps.reloadData()
45         })
46
47         Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid)!).child("snaps").observe(DataEventType.childRemoved, with: { (snapshot) in
48             var iterator = 0
49             for snap in self.snaps{
50                 if snap.id == snapshot.key{
51                     self.snaps.remove(at: iterator)
52                 }
53                 iterator += 1
54             }
55             self.tablaSnaps.reloadData()
56         })
57     }
```

51. Ejecute la aplicación e ingrese con un usuario que no tenga **snap**, y cree un **snap** dirigido a otro usuario. Vuelva a la pantalla de **loggeo** y verifique que cuando se visualiza un **snap**, se borra de la base de datos de **FIREBASE** y también del **tableView** de la lista de **snaps**.









52. Anote los detalles más importantes del código implementado

Se utiliza el método observe para escuchar eventos de eliminación (childRemoved) en la base de datos de Firebase. Cuando un elemento es eliminado del nodo "snaps" del usuario actual, se activa esta función.

## ELIMINANDO LA IMAGEN DEL STORAGE DE FIREBASE CUANDO UN SNAP ES VISTO

Se ha eliminado el **Snap**, pero no la imagen que se aloja en Storage. Por ello, debemos eliminarlo para no crear basura en **FIREBASE**

53. Diríjase al archivo “**imagenViewController.swift**” (código).
54. Cree una variable de clase de nombre “**imagenID**” (representará el nombre o ID de la imagen que se guarda en el storage).

```
@IBOutlet weak var imageView: UIImageView!
@IBOutlet weak var descripcionTextField: UITextField!
@IBOutlet weak var elegirContactoBoton: UIButton!

var imagePicker = UIImagePickerController()
var imagenID = NSUUID().uuidString
```

55. En el mismo archivo, en la función **elegirContactoTapped()**, reemplace la llamada de la variable **imagenesFolder** como se muestra a continuación

Línea 39: Se elimina en base al ID generado para cada imagen subida a Storage

```
36     self.elegirContactoBoton.isEnabled = false
37     let imagenesFolder = Storage.storage().reference().child("imagenes")
38     let imagenData = imageView.image?.jpegData(compressionQuality: 0.50)
39     let cargarImagen = imagenesFolder.child("\(imagenID).jpg")
40     cargarImagen.putData(imagenData!, metadata: nil) { (metadata, error) in
41         if error != nil {
```

56. Diríjase al archivo **ElegirUsuarioViewController.swift** (código) y cree la variable de clase **imagenID**.

```
class ElegirUsuarioViewController: UIViewController,

@IBOutlet weak var tableView: UITableView!
var usuarios:[Usuario] = []
var imagenURL = ""
var descrip = ""
var imagenID = ""
```

57. En el archivo “**imagenViewController.swift**”, modifique la función **prepare** para pase al siguiente **ViewController** el valor del **ID** de la imagen.

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let siguienteVC = segue.destination as! ElegirUsuarioViewController
    siguienteVC.imagenURL = sender as! String
    siguienteVC.descrip = descripcionTextField.text!
    siguienteVC.imagenID = imagenID
}
```

58. En el archivo **ElegirUsuarioViewController.swift** En la función **didSelectRowAt**, modifique el código del diccionario del **snap** para añadir un campo **imagenID**, adicionalmente, agregue una línea para que cuando se seleccione un usuario destino para enviarle el **snap**, se regrese automáticamente a la pantalla inicial de nuestro **NavigationController**.

```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    let usuario = usuarios[indexPath.row]
    let snap = ["from" : usuario.email, "descripcion" : descrip, "imagenURL" : imagenURL,
                "imagenID" : imagenID]

    Database.database().reference().child("usuarios").child(usuario.uid).child("snaps").
        childByAutoId().setValue(snap)
    navigationController?.pushViewController(animated: true)
}
```

59. Agregue dicho campo al modelo **Snap**.

```
import Foundation
class Snap{
    var imagenURL = ""
    var descrip = ""
    var from = ""
    var id = ""
    var imagenID = ""
}
```

60. En la clase **SnapsViewController.swift**, agregue la línea para **settear** el valor de **imagenID** en el objeto **snap** que será agregado al array.

```
override func viewDidLoad() {
    super.viewDidLoad()
    tablaSnaps.delegate = self
    tablaSnaps.dataSource = self

    Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid!)).child("snaps").observe(DataEventType.childAdded, with: { (snapshot) in
        let snap = Snap()
        snap.imagenURL = (snapshot.value as! NSDictionary)["imagenURL"] as! String
        snap.from = (snapshot.value as! NSDictionary)["from"] as! String
        snap.descrip = (snapshot.value as! NSDictionary)["descripcion"] as! String
        snap.id = snapshot.key
        snap.imagenID = (snapshot.value as! NSDictionary)["imagenID"] as! String
        self.snaps.append(snap)
        self.tablaSnaps.reloadData()
    })

    Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid!)).child("snaps").child(snap.id).removeValue()
```

61. Ahora que tenemos la información del **ID de la imagen**, regrese al archivo **VerSnapViewController.swift**, (código) y coloque el siguiente código en la función **viewWillDisappear**. Esta función permitirá borrar la imagen de Storage

```
override func viewWillDisappear(_ animated: Bool) {

    Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid!)).child("snaps").child(snap.id).removeValue()




    Storage.storage().reference().child("imagenes").child("\(snap.imagenID).jpg").delete
    { (error) in
        print("Se elimino la imagen correctamente")
    }
}
```

62. Ejecute la aplicación y verifique que luego de visualizar un **snap**, se borre el **snap** y la imagen.

## Foto asociada al snap

<input type="checkbox"/>		0E714808-8866-417C-8A49-F6A3773...	313,34 ...	application/...
<input checked="" type="checkbox"/>		4DC0ED0C-2860-4A52-9E9E-153E36...	389,45 ...	application/...

Una vez visualizada el **snap** se elimina la imagen

<input type="checkbox"/>		D097AEB4-E358-4B35-87E9-3AD1840FB5AE.jpg	1.96 MB	application/octet-stream	21 nov 2023
<input type="checkbox"/>		D3C76E69-081C-4919-A77D-20457F8A11B7.jpg	816.13 KB	application/octet-stream	18 nov 2023
<input type="checkbox"/>		ED5C4F3F-A3D5-48EE-BD7D-47B8E01F58E0.jpg	488.82 KB	application/octet-stream	21 nov 2023

63. Anote los detalles más importantes del código implementado

Se esta borrando el span y la imagen una vez que el usuario ve el span, esto tambien se borra de la base de datos def Firebase.

## MEJORANDO LA APLICACIÓN

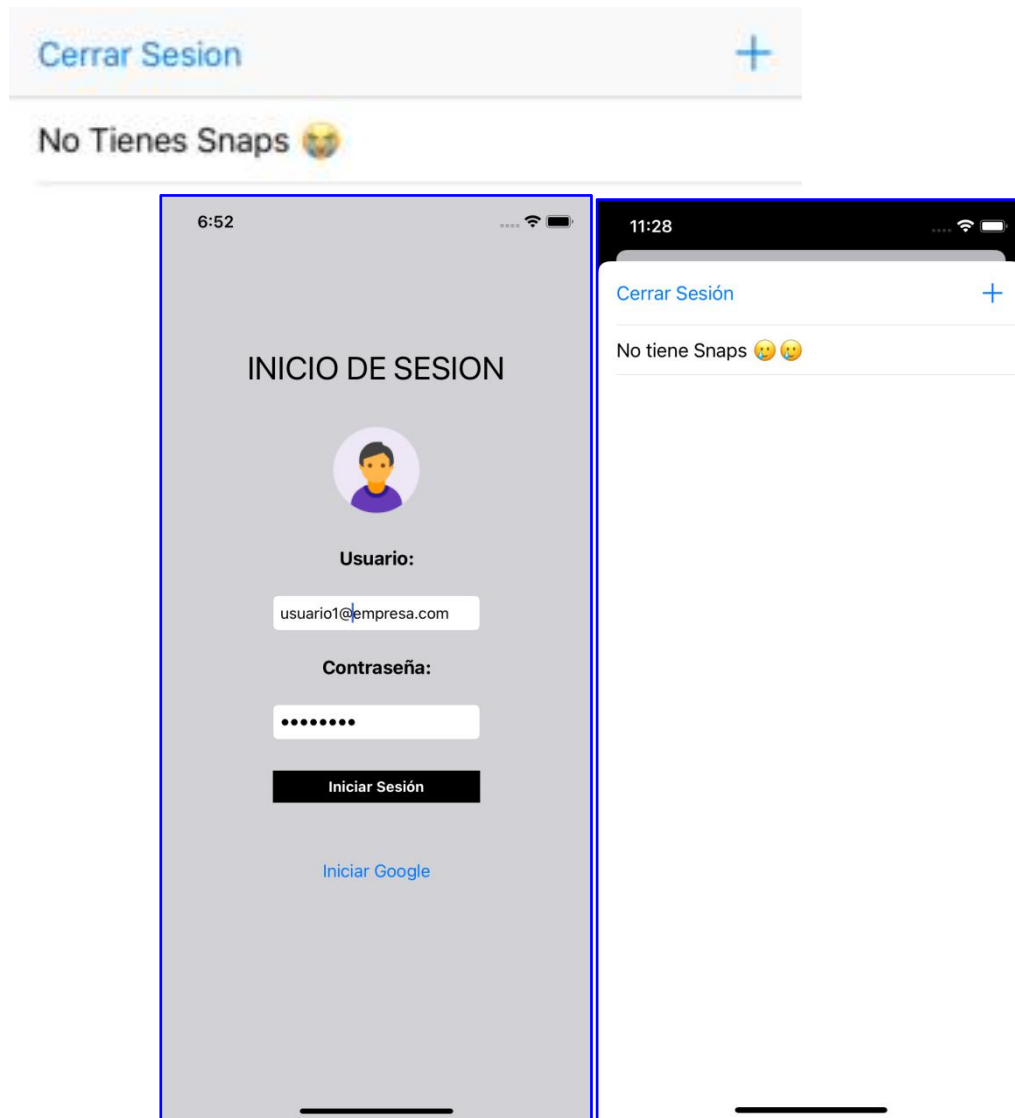
64. Diríjase al archivo **SnapsViewController.swift** para mostrar **NO SNAPS** en caso no haya ninguno. Modifique el código para que quede de la siguiente manera.

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    if snaps.count == 0{
        return 1
    }else{
        return snaps.count
    }
}
```

65. Realice el mismo trabajo en la función **cellForRowAt**

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
    let cell = UITableViewCell()
    if snaps.count == 0{
        cell.textLabel?.text = "No Tienes Snaps 😊"
    }else{
        let snap = snaps[indexPath.row]
        cell.textLabel?.text = snap.from
    }
    return cell
}
```

66. Pruebe la aplicación y verifique que el mensaje de **NO SNAPS** aparece en caso de no tener ninguno.



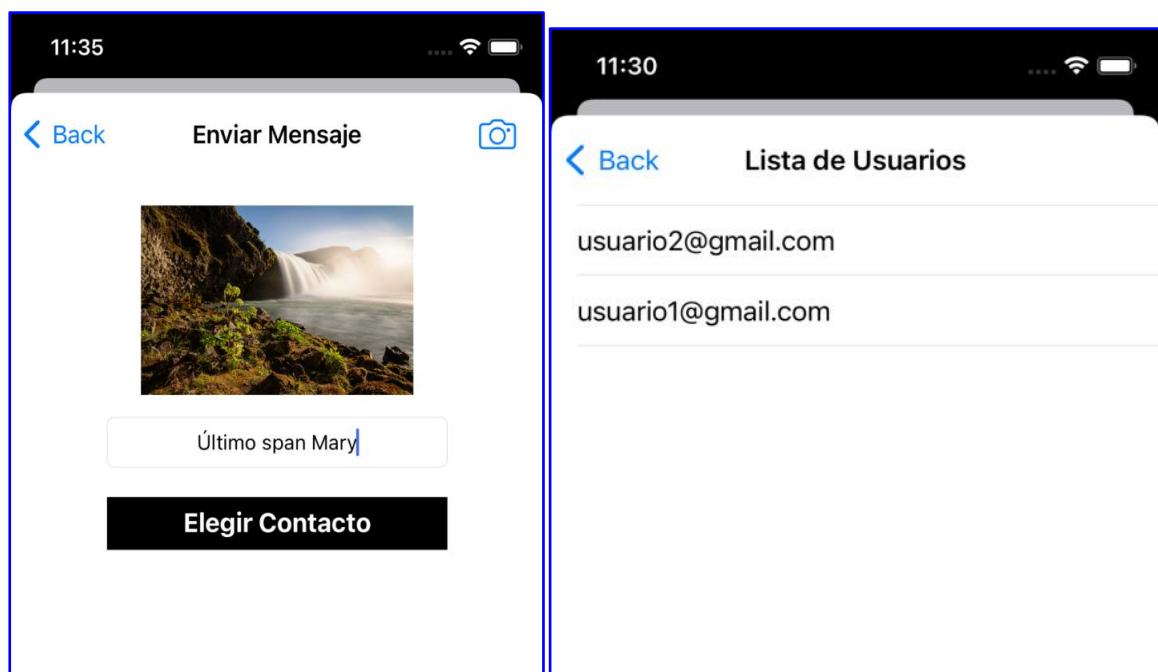
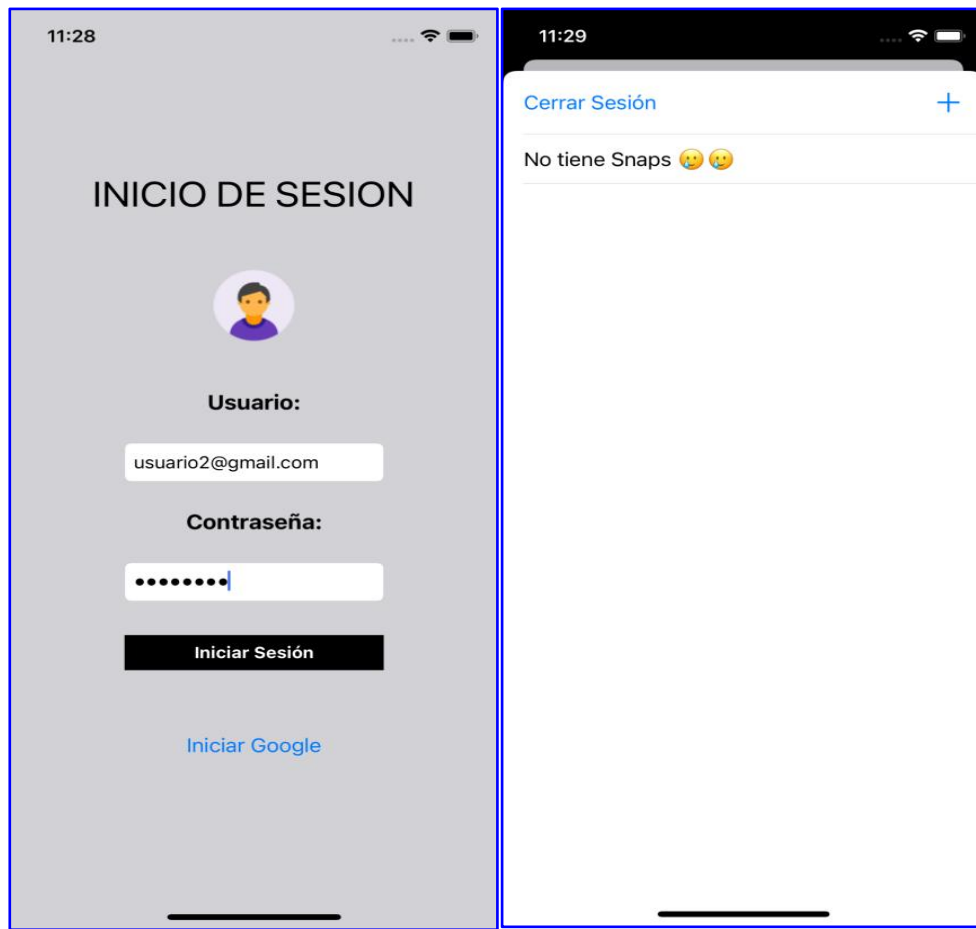
67. Un error en el programa es que el **from** (origen del **snap**), está siendo el mismo usuario de destino, lo cual es errado, el usuario que tiene que ir en el campo **from** es el **usuario que tiene iniciada la sesión actual**.
68. Diríjase al archivo **ElegirUsuarioViewController.swift** y modifique la función **didSelectRowAt** y coloque el siguiente código:

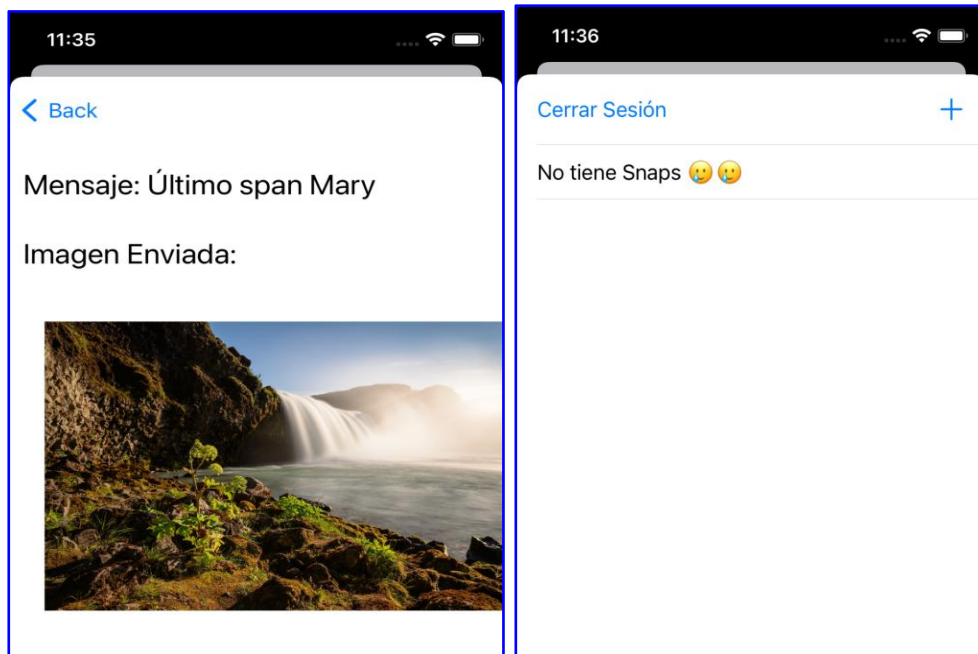
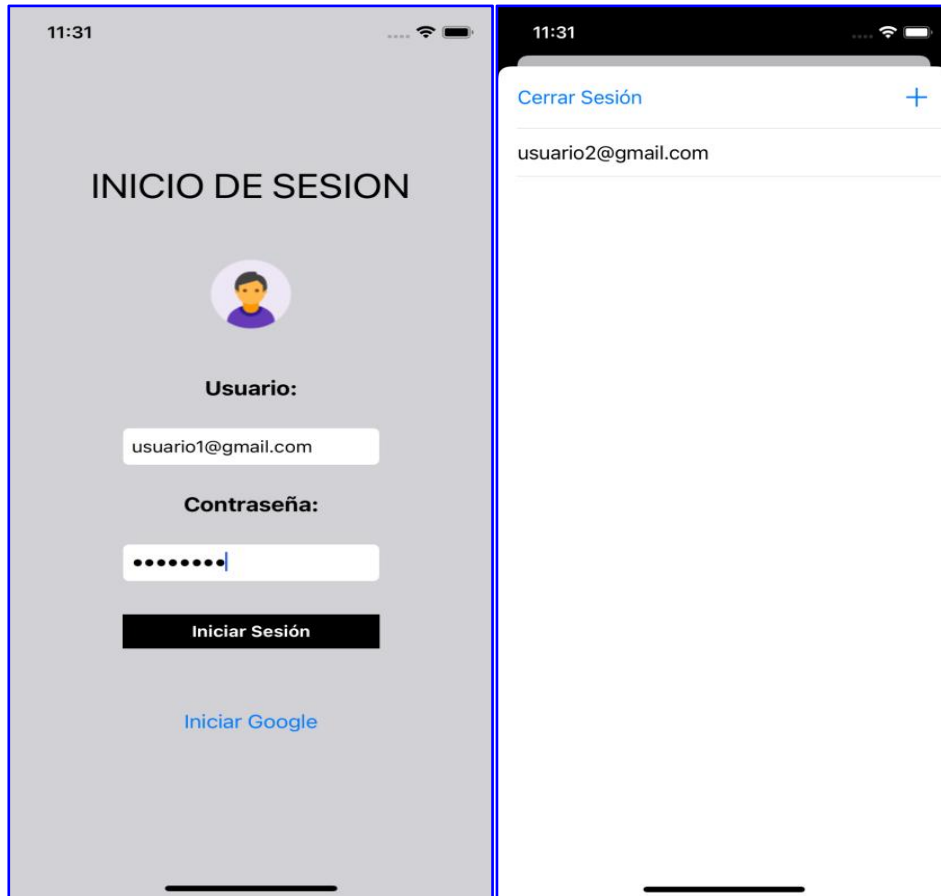
```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    let usuario = usuarios[indexPath.row]
    let snap = ["from" : Auth.auth().currentUser?.email, "descripcion" : descrip,
               "imagenURL" : imagenURL, "imagenID" : imagenID]

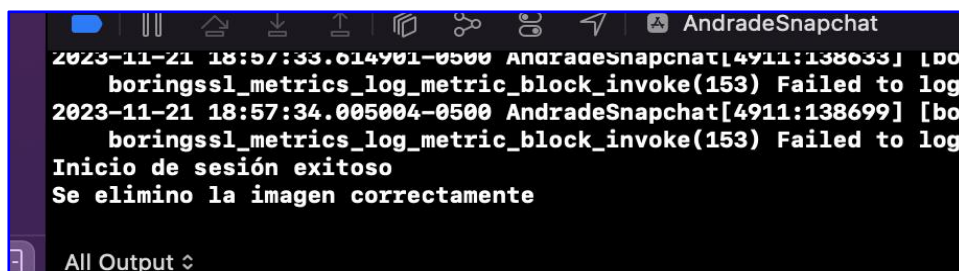
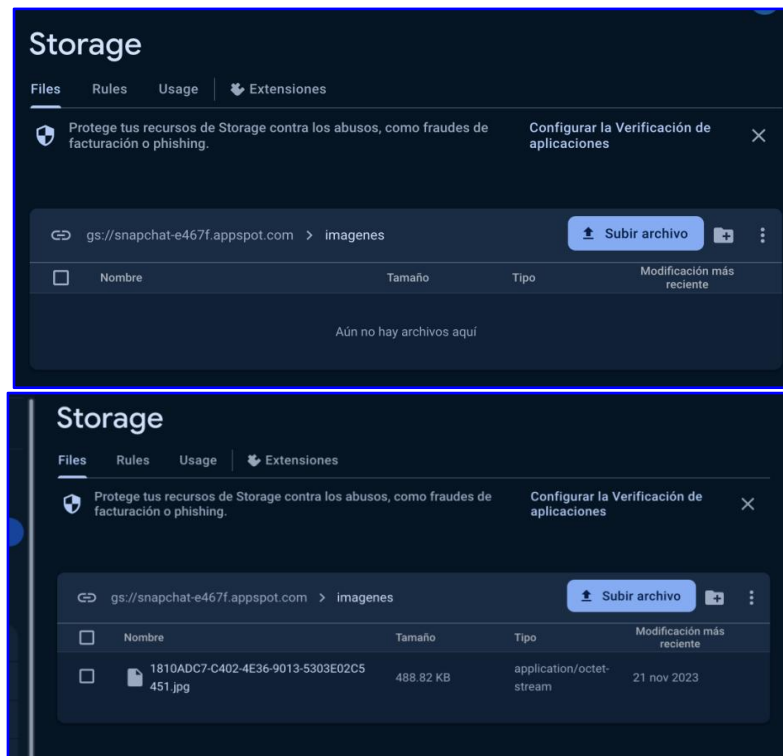
    Database.database().reference().child("usuarios").child(usuario.uid).child("snaps").
    childByAutoId().setValue(snap)
    navigationController?.pushViewController(animated: true)
}
```

69. Borre todos sus usuarios, imágenes, snaps, etc.
70. ¡Ejecute la aplicación con 2 usuarios distintos, y pruebe el envío de SNAPS!!!!!!
71. Adjunte capturas del proceso de envío y recepción de **snaps**



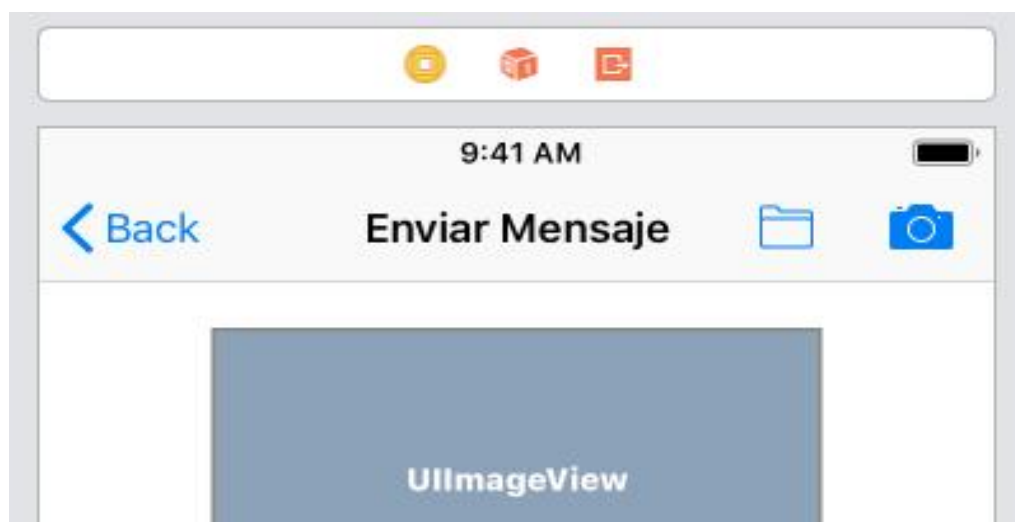






#### PASO ADICIONAL:

75. Para cuando la app esté renderizada en un dispositivo real, active la funcionalidad y permisos para que las imágenes sean fotografías tomadas con la cámara del dispositivo.
76. Diríjase al **main.storyboard** y en el **viewController** asociado al archivo **ImagenViewController.swift** agregue un **Bar Button Item** y modifique su propiedad **System Item** a **Organize**.



77. Cree un **Action** para el botón agregado con el nombre **mediaTapped**

78. Modifique el código de **camaraTapped** y **mediaTapped** como se muestra

```
@IBAction func mediaTapped(_ sender: Any) {
    imagePicker.sourceType = .savedPhotosAlbum
    imagePicker.allowsEditing = false
    present(imagePicker, animated: true, completion: nil)
}

@IBAction func camaraTapped(_ sender: Any) {
    imagePicker.sourceType = .camera
    imagePicker.allowsEditing = false
    present(imagePicker, animated: true, completion: nil)
}
```

79. Diríjase al archivo **Info.plist** para gestionar permisos del uso de la cámara.

Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
Privacy - Camera Usage Des...	String	Necesitamos permiso para usar la camara de tu dispositivo

80. Pruebe la aplicación y su funcionamiento

#### TAREA:

- Mejore la aplicación para enviar audios además de imágenes a través de los snaps. (se envía imagen, descripción y audio)
- El audio debe poder reproducirse remotamente cuando se vea el mensaje enviado. (se visualiza la imagen, descripción y audio enviado)
- Controle los posibles errores que se puedan generar en la aplicación

#### Snap.swift

```
7
8 import Foundation
9 class Snap{
10     var imagenURL = ""
11     var descrip = ""
12     var from = ""
13     var id = ""
14     var imagenID = ""
15     var audioID = ""
16     var audioURL = ""
17
18 }
19
```

## ElegirUsuarioViewController.swift

```

8  import UIKit
9
10 import Firebase
11 import FirebaseDatabase
12
13 class ElegirUsuarioViewController: UIViewController,UITableViewDataSource,UITableViewDelegate {
14
15     @IBOutlet weak var listaUsuarios: UITableView!
16     var usuarios:[Usuario] = []
17     var imagenURL = ""
18     var descrip = ""
19     var imagenID = ""
20     var audioURL: URL?
21
22     override func viewDidLoad() {
23         super.viewDidLoad()
24         listaUsuarios.delegate = self
25         listaUsuarios.dataSource = self
26         Database.database().reference().child("usuarios").observe(DataEventType.childAdded, with: {(snapshot) in
27             print(snapshot)
28
29             let usuario = Usuario()
30             usuario.email = (snapshot.value as! NSDictionary)["email"] as! String
31             usuario.uid = snapshot.key
32             self.usuarios.append(usuario)
33             self.listaUsuarios.reloadData()
34         })
35     }
36
37     func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
38         return usuarios.count
39     }
40     func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
41         let cell = UITableViewCell()
42         let usuario = usuarios[indexPath.row]
43         cell.textLabel?.text = usuario.email
44         return cell
45     }
46
47     func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
48         let usuario = usuarios[indexPath.row]
49         var snapData: [String: Any] = [
50             "from": Auth.auth().currentUser?.email,
51             "descripcion": descrip,
52             "imagenURL": imagenURL,
53             "imagenID": imagenID
54         ]
55
56         if let audioURL = audioURL {
57             snapData["audioID"] = NSUUID().uuidString
58             let audiosFolder = Storage.storage().reference().child("audios")
59             let cargarAudio = audiosFolder.child("\(snapData["audioID"] as! String).m4a")
60             cargarAudio.putFile(from: audioURL, metadata: nil) { (metadata, error) in
61                 if let error = error {
62                     print("Ocurrió un error al subir el audio: \(error)")
63                 } else {
64                     cargarAudio.downloadURL { (url, error) in
65                         if let audioURL = url {
66                             snapData["audioURL"] = audioURL.absoluteString
67                             self.guardarSnapEnFirebase(usuario: usuario, snapData: snapData)
68                         }
69                     }
70                 }
71             }
72         } else {
73             guardarSnapEnFirebase(usuario: usuario, snapData: snapData)
74         }
75     }
76
77     func guardarSnapEnFirebase(usuario: Usuario, snapData: [String: Any]) {
78         Database.database().reference().child("usuarios")
79             .child(usuario.uid)
80             .child("snaps")
81             .childByAutoId()
82             .setValue(snapData)
83         navigationController?.pushViewController(animated: true)
84     }
85

```



## ImagenViewController.swift

```

7
8 import UIKit
9 import FirebaseStorage
10 import AVFoundation
11 class ImagenViewController:
    UIViewController, UIImagePickerControllerDelegate, UINavigationControllerDelegate, AVAudioRecorderDelegate {
12
13     var imagePicker = UIImagePickerController()
14     var imagenID = NSUUID().uuidString
15     var audioRecorder: AVAudioRecorder?
16     var audioURL: URL?
17     @IBAction func camaraTapped(_ sender: Any) {
18         imagePicker.sourceType = .savedPhotosAlbum
19         imagePicker.allowsEditing = false
20         present(imagePicker, animated: true, completion: nil)
21     }
22     @IBOutlet weak var imageView: UIImageView!
23     @IBOutlet weak var descripcionTextField: UITextField!
24     @IBOutlet weak var elegirContactoBoton: UIButton!
25     @IBAction func elegirContactoTapped(_ sender: Any) {
26         self.elegirContactoBoton.isEnabled = false
27         let imagenesfolder = Storage.storage().reference().child("imagenes")
28         let imageData = imageView.image?.jpegData(compressionQuality: 0.50)
29         let cargarImagen = imagenesfolder.child("\(imagenID).jpg")
30         cargarImagen.putData(imageData!, metadata: nil) { (metadata, error) in
31             if error != nil {
32                 self.mostrarAlerta(titulo: "Error", mensaje: "Se produjo un error un error al subir la imagen.
33                     Verifique su conexion a internet y vuelva a intentarlo.", accion: "Aceptar")
34                 self.elegirContactoBoton.isEnabled = true
35                 print("Ocurrio un error al subir la imagen: \(error)")
36             } else {
37                 cargarImagen.downloadURL(completion: {(url, error) in
38                     guard let enlaceURL = url else {
39                         self.mostrarAlerta(titulo: "ERROR", mensaje: "Se produjo un error al obtener la
40                             informacion de la imagen", accion: "Cancelar")
41                         self.elegirContactoBoton.isEnabled = true
42                     }
43                 })
44             }
45         }
46     }
47 }

```

Line: 113 Col: 36

```

@IBOutlet weak var grabarAudioButton: UIButton!
@IBAction func grabarAudioAction(_ sender: Any) {
    if audioRecorder?.isRecording == true {
        detenerGrabacion()
    } else {
        empezarGrabacion()
    }
}

override func viewDidLoad() {
    super.viewDidLoad()
    imagePicker.delegate = self
    elegirContactoBoton.isEnabled = false
    setupGrabarAudio()
}

// Do any additional setup after loading the view.

func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info:
    [UIImagePickerController.InfoKey : Any]) {
    let image = info[UIImagePickerController.InfoKey.originalImage] as? UIImage
    imageView.image = image
    imageView.backgroundColor = UIColor.clear
    elegirContactoBoton.isEnabled = true
    imagePicker.dismiss(animated: true, completion: nil)
}

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    let siguienteVC = segue.destination as! ElegirUsuarioViewController
    siguienteVC.imagenURL = sender as! String
    siguienteVC.descripcion = descripcionTextField.text!
    siguienteVC.imagenID = imagenID
    siguienteVC.audioURL = audioURL
}

```



```

let directorioPath = getDirectorioDocumentos().appendingPathComponent("grabacionAudio.m4a")

let configuracionGrabacion: [String : Any] = [
    AVFormatIDKey: kAudioFormatAppleLossless,
    AVEncoderAudioQualityKey: AVAudioQuality.max.rawValue,
    AVEncoderBitRateKey: 320000,
    AVNumberOfChannelsKey: 2,
    AVSampleRateKey: 44100.0
]

do {
    audioRecorder = try AVAudioRecorder(url: directorioPath, settings: configuracionGrabacion)
    audioRecorder?.delegate = self
    audioRecorder?.prepareToRecord()
} catch {
    print(error.localizedDescription)
}

func empezarGrabacion() {
    audioURL = getDirectorioDocumentos().appendingPathComponent("grabacionAudio.m4a")
    audioRecorder?.record()
    grabarAudioButton.setTitle("Detener Grabación", for: .normal)
}

func detenerGrabacion() {
    audioRecorder?.stop()
    grabarAudioButton.setTitle("Grabar Audio", for: .normal)
}

func getDirectorioDocumentos() -> URL {
    let paths = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask)
    return paths[0]
}

func audioRecorderDidFinishRecording(_ recorder: AVAudioRecorder, successfully flag: Bool) {
    if flag {
        print("Grabación de audio exitosa")
    } else {
        print("Error en la grabación de audio")
    }
}

```

## SnapsViewController.swift

```

35 }
36 override func viewDidLoad() {
37     super.viewDidLoad()
38     tablaSnaps.delegate = self
39     tablaSnaps.dataSource = self
40
41     Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid!).child("snaps")
42         .observe(DataEventType.childAdded, with: { (snapshot) in
43             let snap = Snap()
44             snap.imagenURL = (snapshot.value as! NSDictionary)["imagenURL"] as! String
45             snap.from = (snapshot.value as! NSDictionary)["from"] as! String
46             snap.descrip = (snapshot.value as! NSDictionary)["descripcion"] as! String
47             snap.id = snapshot.key
48             snap.imagenID = (snapshot.value as! NSDictionary)["imagenID"] as! String
49             snap.audioID = (snapshot.value as! NSDictionary)["audioID"] as! String
50             self.snaps.append(snap)
51             self.tablaSnaps.reloadData()
52         })
53     Database.database().reference().child("usuarios").child((Auth.auth().currentUser?.uid!).child("snaps")
54         .observe(DataEventType.childRemoved, with: { (snapshot) in
55             var iterator = 0
56             for snap in self.snaps {
57                 if snap.id == snapshot.key {
58                     self.snaps.remove(at: iterator)
59                 }
60                 iterator += 1
61             }
62             self.tablaSnaps.reloadData()
63         })
64     })
65 }
66 func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
67     let snap = snaps[indexPath.row]
68     performSegue(withIdentifier: "versnapsegue", sender: snap)
69 }

```

## VerSnapViewController.swift

```
@IBOutlet weak var imageView: UIImageView!
@IBOutlet weak var lblMensaje: UILabel!
var snap = Snap()
var audioPlayer: AVAudioPlayer?

@IBOutlet weak var reproducirAudioButton: UIButton!

@IBAction func reproducirAudioAction(_ sender: Any) {
    if let audioURL = URL(string: snap.audioURL) {
        do {
            audioPlayer = try AVAudioPlayer(contentsOf: audioURL)
            audioPlayer?.play()
            reproducirAudioButton.setTitle("Reproducir Audio", for: .normal)
        } catch {
            print("Error al reproducir audio: \(error.localizedDescription)")
        }
    }
    print("Reproduciendo audio...")
}

override func viewDidLoad() {
    super.viewDidLoad()
    lblMensaje.text = "Mensaje: " + snap.descrip
    imageView.sd_setImage(with: URL(string: snap.imagenURL), completed: nil)
}

override func viewWillAppear(_ animated: Bool) {
    if let currentUserID = Auth.auth().currentUser?.uid {
        let imagenRef = Storage.storage().reference().child("imagenes").child("\(snap.imagenID).jpg")

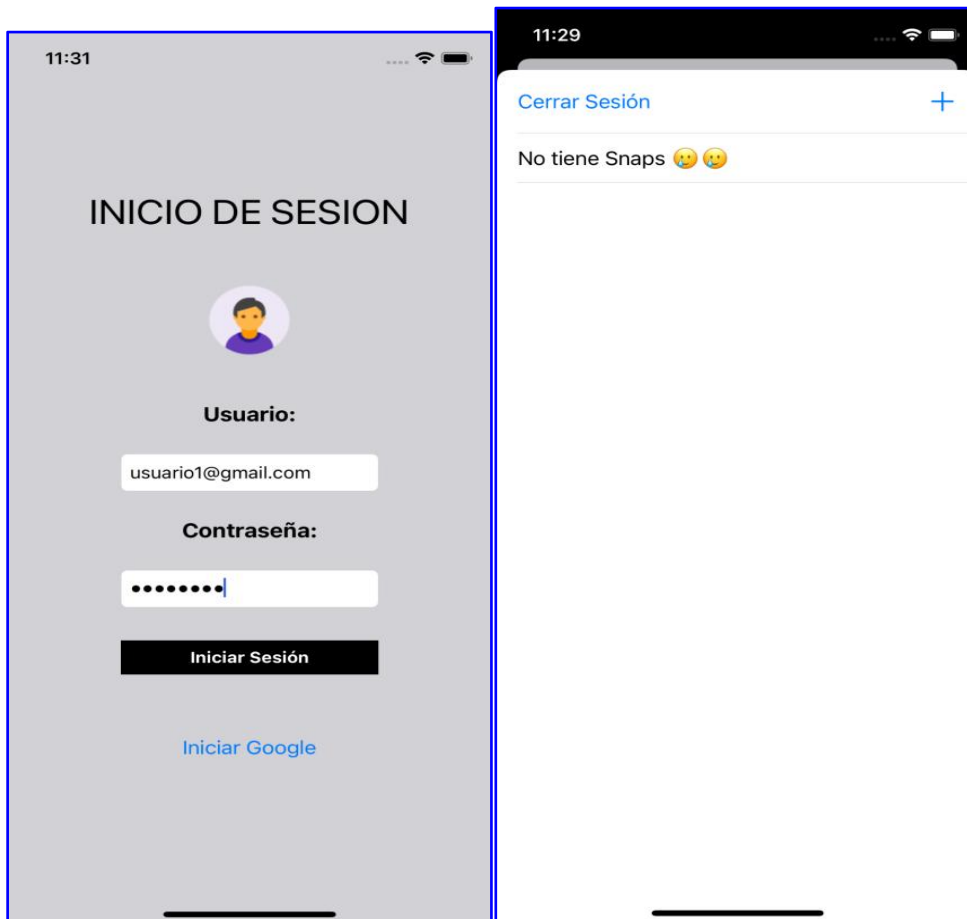
        imagenRef.delete { error in
            if let error = error {
                print("Error al eliminar la imagen: \(error.localizedDescription)")
            } else {
                print("Imagen eliminada correctamente")
            }
        }

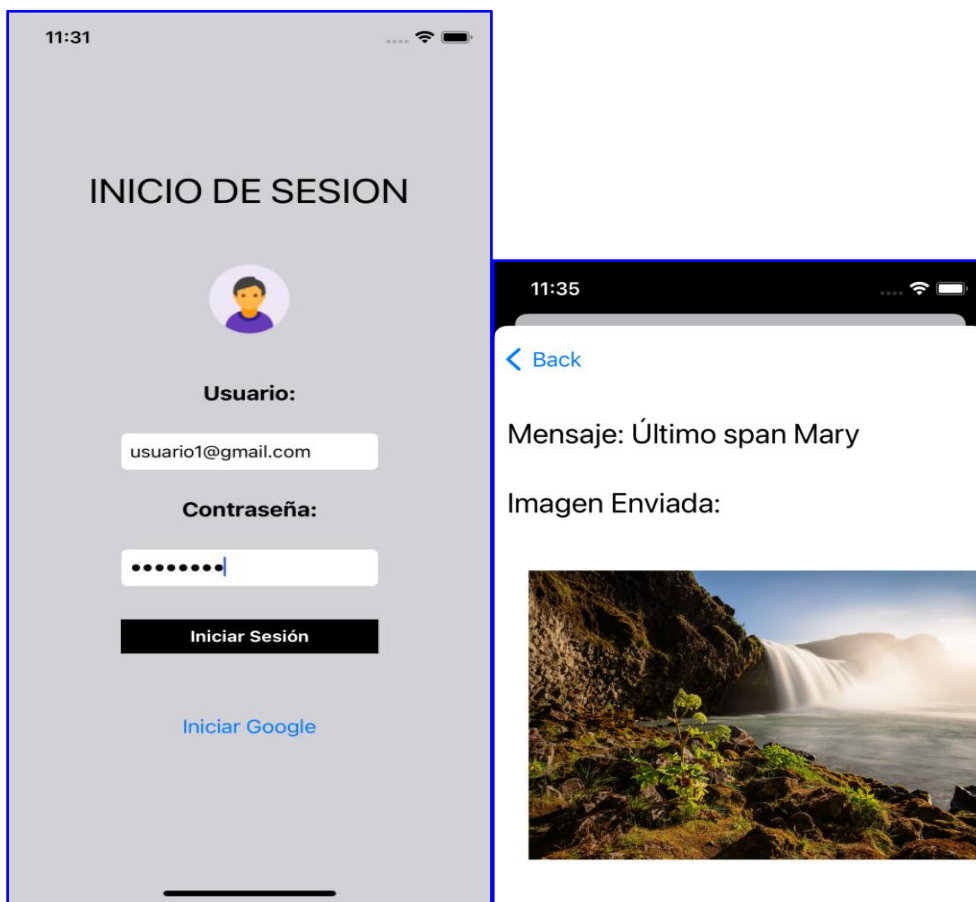
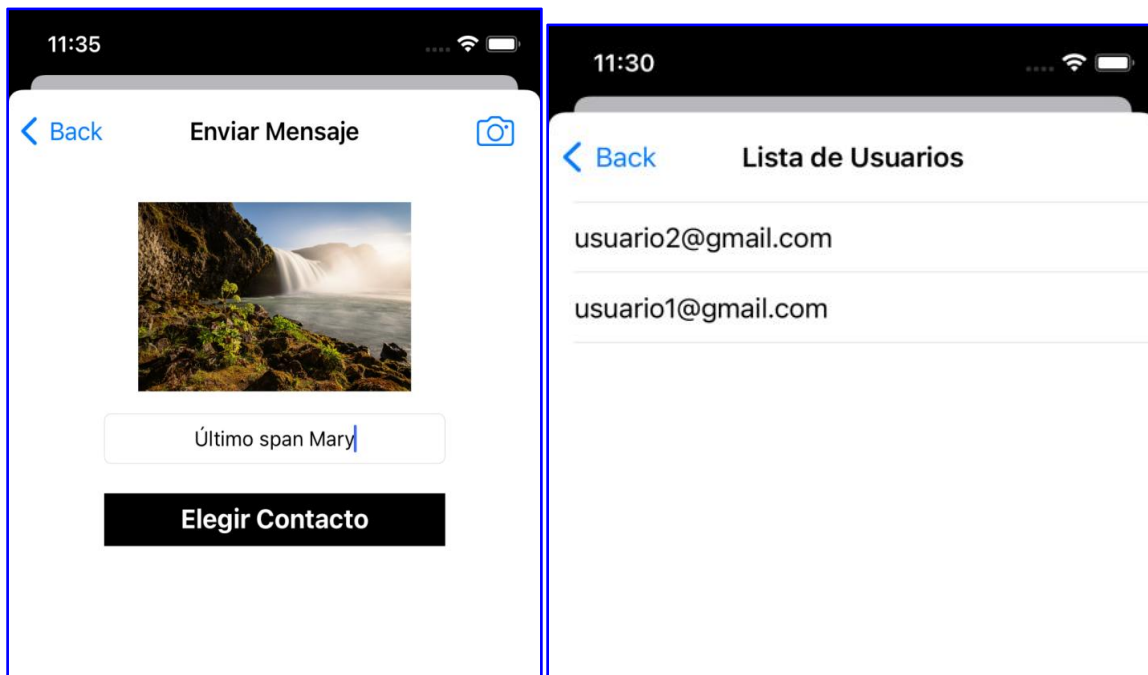
        let audioRef = Storage.storage().reference().child("audios").child("\(snap.audioID).m4a")
        audioRef.delete { error in
            if let error = error {
                print("Error al eliminar el audio: \(error.localizedDescription)")
            } else {
                print("Audio eliminado correctamente")
            }
        }

        Database.database().reference().child("usuarios").child(currentUserID).child("snaps")
            .child(snap.id).removeValue()
    }
}
```

Como podemos ver en esta vista se agrego un boton para grabar el audio, despues se realizo una conexion de tipo Outlet del boton hacia ImagenViewController.swift.

Aqui podemos ver que se agrego otro boton para poder reproducir el audio creado anteriormente, para esto tambien se creo una conexion de tipo Outlet del boton hacia VerSnapViewController.swift





**OBSERVACIONES (5 mínimo):**

(Las observaciones son las notas aclaratorias, objeciones y problemas que se pudo presentar en el desarrollo del laboratorio)

- Implementar medidas de seguridad en Firebase es crucial para proteger los datos sensibles. Se deben establecer reglas de seguridad para el Realtime Database y el Storage que limiten el acceso no autorizado a los datos.
- Manipulación de imágenes y audios: Al trabajar con imágenes y audios en Firebase Storage, se deben considerar las optimizaciones de tamaño y formatos de archivos para reducir el tiempo de carga y mejorar la experiencia del usuario.
- Asegurarse de comprender completamente cómo funciona la sincronización en tiempo real en Firebase Realtime Database. Los datos se actualizan en tiempo real, lo que puede impactar en el rendimiento y la lógica de la aplicación.
- Al eliminar elementos de la base de datos o el almacenamiento, se debe tener precaución y validar adecuadamente para evitar pérdida de datos no deseada.
- Es importante monitorear el rendimiento de la aplicación al interactuar con Firebase. Los tiempos de carga, la latencia de la red y otros indicadores clave deben ser evaluados para optimizar la experiencia del usuario.

**CONCLUSIONES (5 mínimo):**

(Las conclusiones son una opinión personal sobre tu trabajo, explicar como resolviste las dudas o problemas presentados en el laboratorio. Además de aportar una opinión crítica de lo realizado)

- Enfrentar problemas técnicos en la integración con Firebase fue una oportunidad para mejorar habilidades de resolución de problemas. La documentación de Firebase y las comunidades en línea fueron recursos valiosos para superar estos desafíos.
- El laboratorio resaltó la importancia del aprendizaje continuo en entornos de desarrollo de aplicaciones. Cada problema resuelto amplió mi comprensión de Firebase y su aplicación en proyectos futuros.
- La seguridad de los datos es fundamental, y la configuración adecuada de reglas de seguridad en Firebase es esencial para proteger la información del usuario.
- La retroalimentación constante y las pruebas exhaustivas son claves para garantizar un funcionamiento óptimo de la aplicación. Identificar y corregir errores en etapas tempranas minimiza problemas futuros.
- Se reconoce la importancia de la planificación y el análisis crítico en el desarrollo de aplicaciones utilizando Firebase. Evaluar constantemente las decisiones de diseño y la arquitectura de datos asegura un producto final sólido y confiable.