

**I.- OBJETIVOS:**

* Crear una aplicación que haga uso de la cámara de fotos de un dispositivo.

**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 trabará con un equipo con MAC OS.

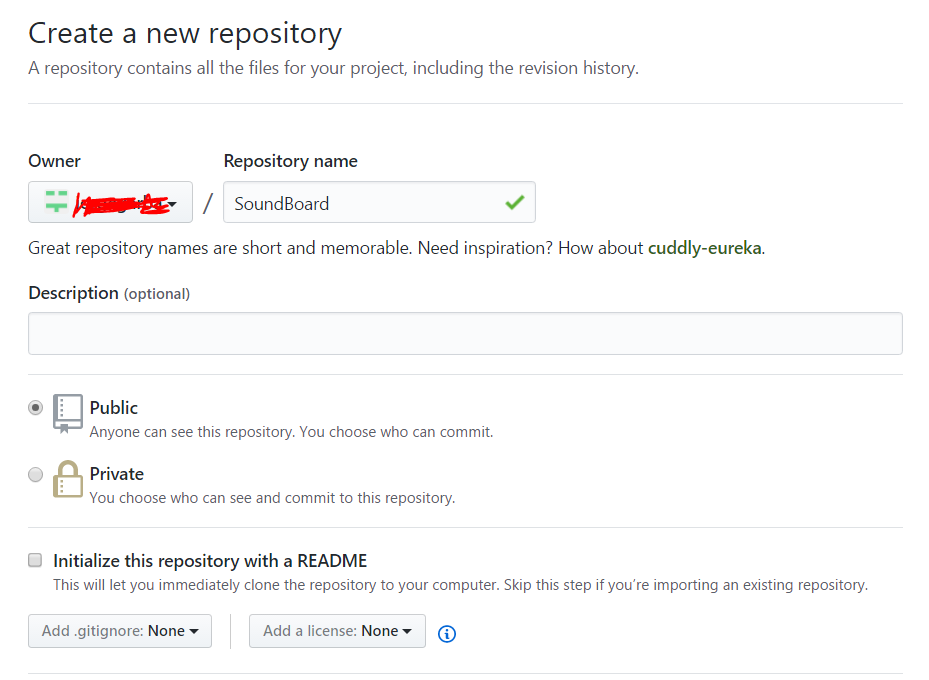
**VI.- METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:**

* El desarrollo del laboratorio es individual.

**VII.- PROCEDIMIENTO:**

**ACTIVIDADES:**

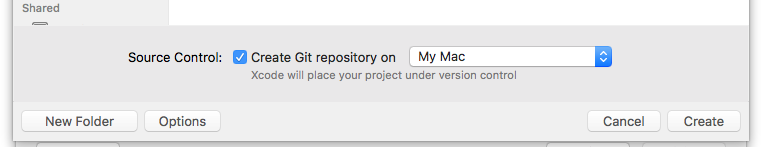
1. Ahora, abra safari e ingrese a su cuenta en **github,** si no tiene una, créela.
2. Cree un nuevo repositorio, asegúrese de colocar el mismo nombre del proyecto que crearemos en esta sesión. “**SoundBoard**” .
3. Clic en el botón verde de abajo “**Create Repository**”



1. Copie la dirección generada del repositorio.

**CREANDO LA APLICACIÓN**

1. Cree un nuevo proyecto en Xcode de tipo Single View Application
2. El nombre del proyecto será “SoundBoard”. No olvide activar la casilla de CoreData
3. En el siguiente menú contextual, elija la opción **Create Git Repository on My Mac.**



1. Termine la creación del proyecto.

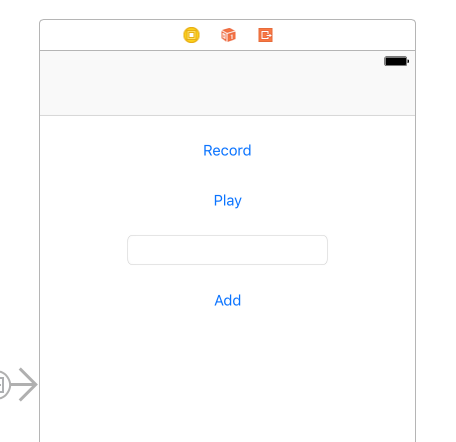
**SINCRONIZANDO EL REPOSITORIO**

1. Abra la aplicación sourceTree y sincronice su repositorio local con GitHub.

**DISEÑANDO LA INTERFAZ DE USUARIO**

1. Agregue un Table View en el ViewController inicial y modifique sus constraints para que ocupe toda la pantalla.
2. Clic en el mismo ViewController y agregue un NavigationController desde el menú **Editor** -> **Embed** **In** -> **NavigationController**.
3. Seleccione el ViewController y en sus propiedades, en la categoría Layout, clic (para quitar el check) en **Adjust** **Scroll** **View** **Insets**.
4. Agregue un bar button con el símbolo de “+” en la esquina superior derecha.
5. Añada un segundo ViewController al main.storyboard.
6. Cree un segue de tipo show del botón “+” del primer ViewController al segundo.
7. En el segundo ViewController creado agregue:
   1. 1 Botón centrado y a 20 puntos de la parte superior del ViewController con el texto: **Record**.
   2. 1 Botón centrado dependiente vertical del primer botón a 20 puntos con el texto: **Play**.
   3. 1 TextField centrado dependiente vertical del segundo botón a 20 puntos y ancho(width) estático de 200 puntos. Además Placeholder con texto: “**Sound Name**” centrado en el TextField.
   4. 1 Botón centrado dependiente vertical del campo de texto a 20 puntos con el texto: “**Add**”.

El diseño debería ser similar al siguiente:



**AGREGANDO UN VIEW CONTROLLER**

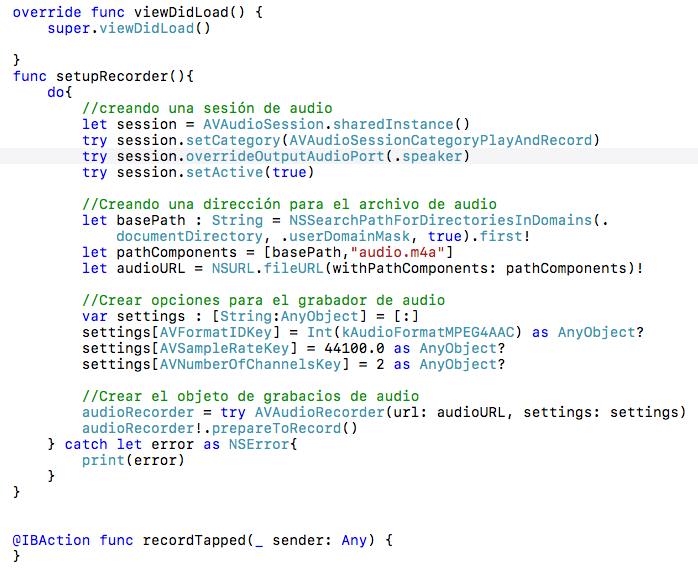
1. Agregue un nuevo archivo de tipo CocoaTouch con el nombre de **SoundViewController,** subclase de UIVIewController en lenguaje Swift.
2. En el main.Storyboard, asocie la clase ViewController recién creada con la segunda vista.
3. Divida su pantalla y cree 2 conexiones de tipo Outlet en soundViewController y su respectiva vista
   1. Del botón Record con el nombre **recordButton**
   2. Del TextField con el nombre **nameTextField.**
4. Cree 3 conexiones de tipo Action de los tres botones en el **soundViewController** con los nombres de: **recordTapped, playTapped y addTapped,**

**TRABAJANDO CON EL MICRO**

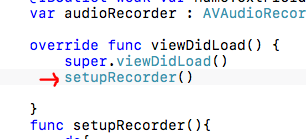
1. En el archivo **soundViewController** importe la librería **AVFoundation.**
2. Como variable de clase, justo luego de la creación de los outlets, cree una variablde de tipo AVFoundation



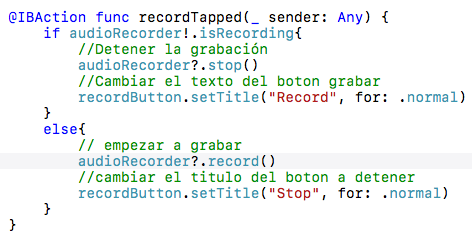
1. Implemente la función setupRecorder justo luego de la función viewDidLoad() como muestra la figura a continuación:



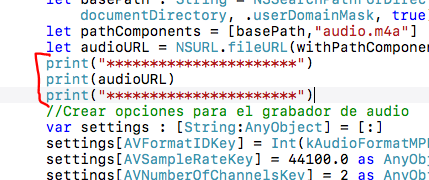
1. Asegúrese de llamar a la función creada en la sección anterior en la función viewDidLoad



1. Implemente la función recordTapped con el siguiente código:



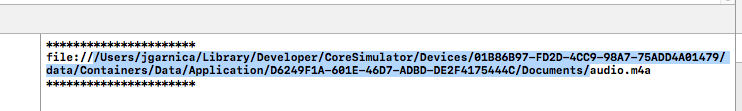
1. Añada 3 líneas de código a la función setuRecorder() al final de la sección “**Creando una dirección para el archivo de audio**” para imprimir ( temporalmente), el lugar en el que se guardarán nuestros archivos de audio.



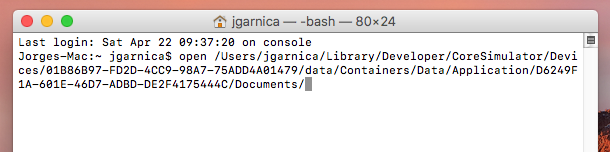
1. Ejecute la aplicación:
   1. Presiones **Record**. El botón habrá cambiado su texto a **“Stop”.**
   2. Presione **Stop.**
   3. En la consola de Xcode, verifique que se haya impreso la ruta de dónde se haya guardado el archivo de audio:



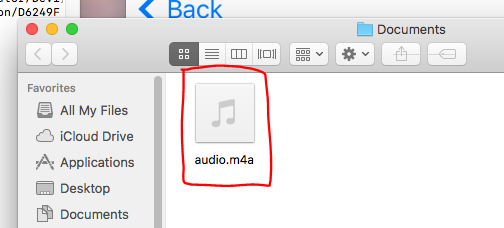
1. Copie la dirección marcada a continuación:



1. Abra el terminal (ventana de comandos) en y coloque el comando **open** y peque la dirección copiada:



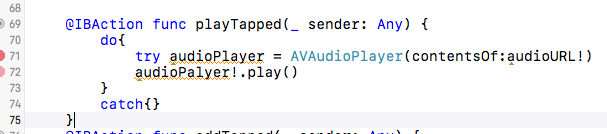
1. Presione **ENTER.**
2. A continuación, se abrirá el directorio en el que se encuentra almacenado nuestro nuevo archivo de audio



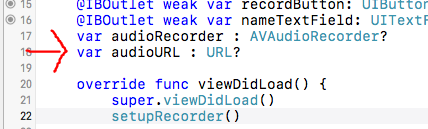
1. Observará que incluso usando el simulador de Xcode, se usa el micrófono de la MACBOOK, de modo que a este punto, será posible reproducir el archivo.

**REPRODUCIENDO EL AUDIO EN iOS**

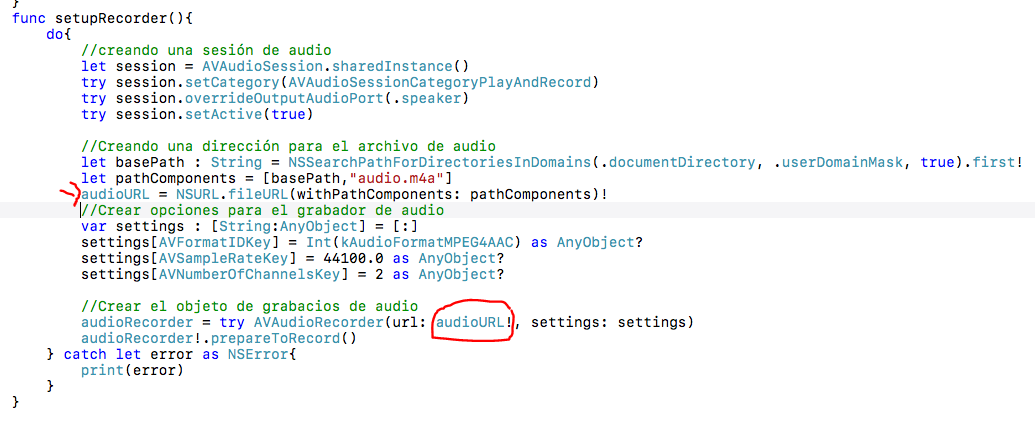
1. Cree una variable de clase justo debajo de la variable **audioRecorder,** pero esta vez con el nombre de **audioPlayer** objeto de la clase **AVAudioPlayer?.**
2. Ahora implemente la función **playTapped,** función encargada de reproducir el archivo de audio con el siguiente código:



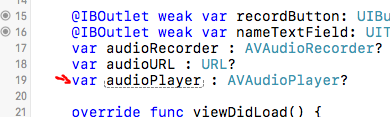
1. Extraiga la variable audioURL de la función **setupRecorder,** ahora será una variable global.



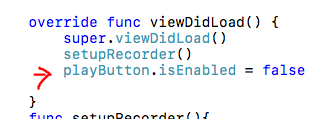




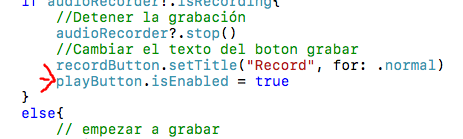
1. Agregue una variable de clase de nombre audioPlayer como se muestra en la imagen a **continuación**:



1. Intente grabar un nuevo audio y reproducirlo desde el simulador.
2. Anote los detalles más importantes:
3. Cree una conexión de tipo Outlet del botón **play,** en el ViewController el nombre de la variable será **playButton.**
4. Cree una conexión de tipo Outlet del botón **add,** en el ViewController el nombre de la variable será **addButton.**
5. Inyecte el siguiente código en la función **ViewDidLoad** para que el botón play esté deshabilitado por defecto.



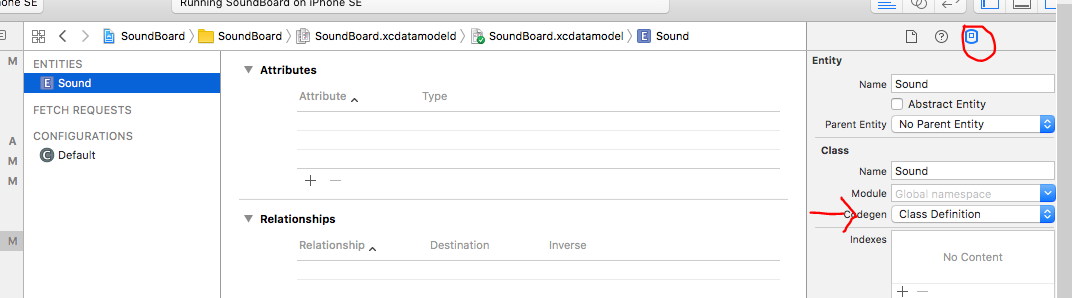
1. Inyecte el siguiente al final de la función recordTapped() para que el botón play sehabilite luego de hacer una grabación.



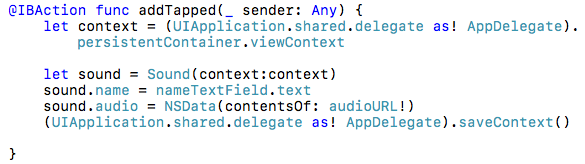
1. Pruebe que la aplicación funcione para grabar y reproducir un único audio.
2. Anote los detalles más importantes.

**GUARDANDO LOS SONIDOS CON COREDATA**

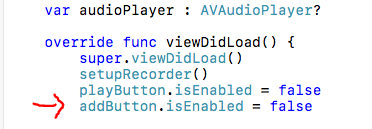
1. Ingrese a el archivo SoundBoard.xdatamodeld
2. Cree una entidad de nombre **Sound.**
3. Asegúrese que el código para dicha entidad sea una clase. En caso de que no, cámbielo como se muestra en la figura.



1. Cree ahora dos entidades para la clase Sound:
   1. Attribute: name Tipo: String
   2. Attribute: audio Tipo: Binary Data
2. Regrese al archivo SoundViewController.swift e inyecte el siguiente código en la función addTapped para guardar el sonido con core data.



1. Agregue las siguientes líneas para bloquear el botón de add sin haber grabado algún audio.



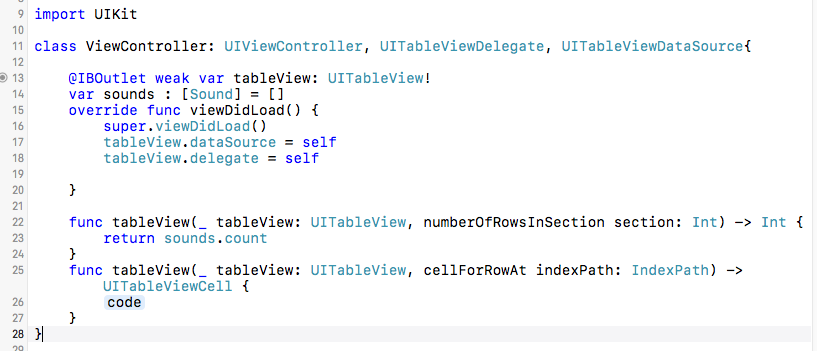


1. Finalmente, la línea de código al final de la función addTapped para que una vez guardada la grabación, se regrese a la vista anterior.

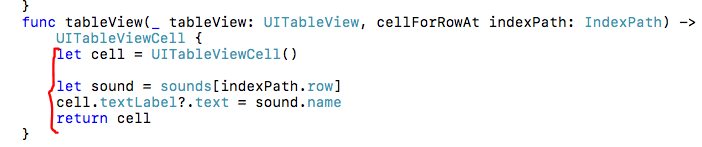


**MOSTRANDO LA LISTA DE GRABACIONES**

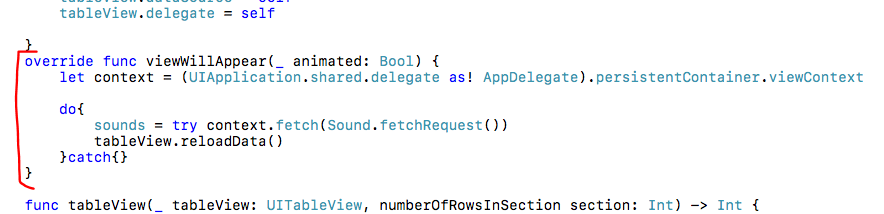
1. Diríjase al ViewController principal (el que tiene el TableView).
2. Cree una conexión de tipo Outlet del TableView hacia su clase con el nombre de “**tableView**”.
3. A continuación, en la misma clase, implemente los protocolos mínimos que obedezcan al control de un **tableView.**
4. En la función viewDidLoad, asigne al tableView creado el dataSource y delegado respectivo.
5. Cree un arreglo vacío de objetos de tipo **Sound** de nombre **sounds** como arreglo de clase luego del Outlet del **tableView**
6. Implemente las funciones numberOfRowsInSection y cellForRowAt.
7. En la función numberOfRowsInSection, retorne la cantidad de elementos que tiene el arreglo **sounds**
8. Todo ello debería quedar como en la siguiente imagen.



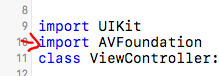
1. Implemente la función cellForRowAt para llenar el table View con la lista de grabaciones que tengamos en CoreData.



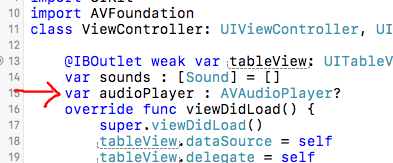
1. Implemente la función **viewWillAppear** luego de la función **viewDidLoad,** para que cada vez que la vista vaya a ser lanzada, se llene y recargue el tableView con las grabaciones disponibles en **CoreData.**



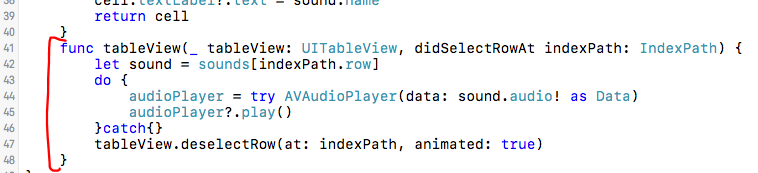
1. Ejecute la aplicación y verifique que la lista de grabaciones se muestre en la primera vista además de poder grabar nuevas pistas.
2. Importe la librería para poder reproducir y configurar pistas de audio en la clase actual.



1. Cree una variable de clase de nombre **audioPlayer().**



1. Para finalizar, implemente la función **didSelectRowAt** para reproducir el audio seleccionado de la lista en el **TableView.**



1. Pruebe la aplicación. A este punto, la aplicación podrá reproducir los audios de la lista de grabaciones, además de grabar un nuevo audio en cualquier momento.
2. Anote los detalles más importantes.

**ELIMINANDO LAS GRABACIONES DESDE EL TABLEVIEW**

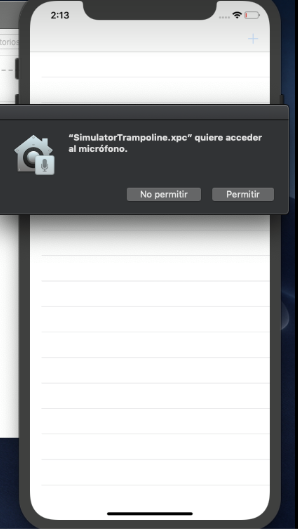
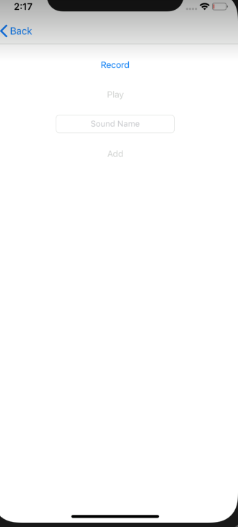
1. Implemente la función **commit** **editingStyle.** Esta función analizará si hacen un deslizamiento en la pantalla “**SWIPE**” sobre un ítem de la lista para posteriormente eliminarlo.

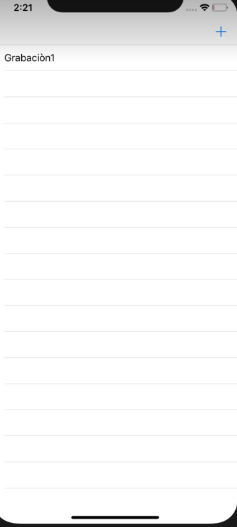


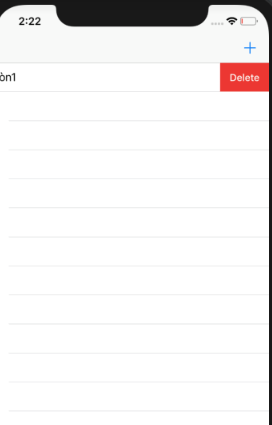
LISTO! Tenemos nuestra aplicación terminada.

1. Ejecute la aplicación y compruebe que una grabación se puede eliminar de la lista de grabaciones deslizando con el dedo a hacia la izquierda.
2. Anote los detalles más importantes.
3. GUARDE SU PROYECTO EN SOURCETREE PARA SER SINCRONIZADO CON GITHUB

CAPTURAS:



**TAREA:**

**Realice un trabajo de investigación breve sobre FIREBASE, definiciones, conceptos, usos, ejemplos (sin tener que demostrar ningún código), etc.**

Firebase es una plataforma móvil desarrollada por Google, que nos ofrece un gran número de características que puedes escoger y mezclar entre ellas. Son, sobre todo, servicios en la nube que permiten al usuario guardar o recuperar datos desde cualquier dispositivo o navegador.

Puede ser muy útil para aplicaciones de mensajería (WhatsApp, Telegram, Hangouts…), notificaciones para dispositivos o navegadores, analítica web e incluso te permite ganar dinero a través de AdMob (introduciendo banners o native addons).

En resumen, Firebase se encarga de los elementos del backend para centrarnos en el front-end y la funcionalidad.

En muchas ocasiones nos planteamos cómo poder acceder a un servicio web para poder tener nuestra aplicación trabajando con datos en la nube. Por ello surgió Firebase, para proveer una API para guardar y sincronizar datos en la nube en tiempo real.

Sus características fundamentales están divididas en varios grupos, las cuales podemos agrupar en:

* Analíticas: Provee una solución gratuita para tener todo tipo de medidas (hasta 500 tipos de eventos), para gestionarlo todo desde un único panel.
* Desarrollo: Permite construir mejores apps, permitiendo delegar determinadas operaciones en Firebase, para poder ahorrar tiempo, evitar bugs y obtener un aceptable nivel de calidad. Entre sus características destacan el almacenamiento, testeo, configuración remota, mensajería en la nube o autenticación, entre otras.
* Crecimiento: Permite gestionar los usuarios de las aplicaciones, pudiendo además captar nuevos. Para ello dispondremos de funcionalidades como las de invitaciones, indexación o notificaciones.
* Monetización: Permite ganar dinero gracias a AdMob.

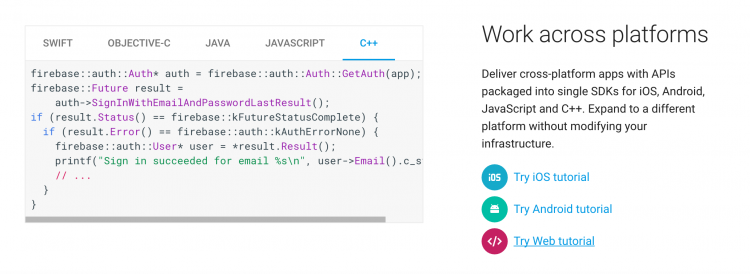
Google no ha obviado ningún detalle, y a la hora de gestionar cualquiera de los módulos anteriores, tenemos disponible una gran cantidad de información, así como de ejemplos o incluso tutoriales.



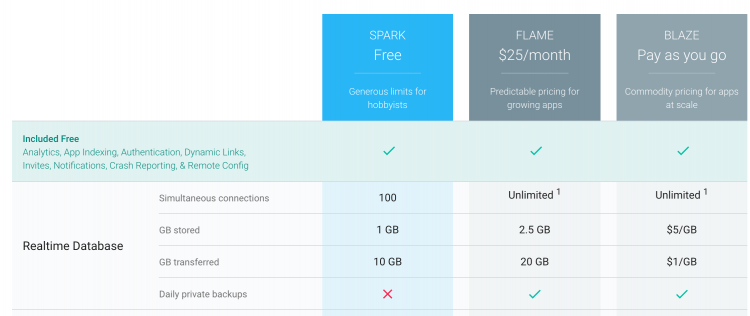
Todo ello, tanto para iOS como para Android como para web:

[Documentación de Firebase](https://firebase.google.com/docs/)

En esta página podremos encontrar toda la información necesaria, clasificada por las categorías que anteriormente os citábamos, incluyendo cada una de las plataformas, de modo que todo será muy intuitivo para leer, y nos resultará muy sencillo su uso gracias a los ejemplos con código real.

[](https://elandroidelibre.elespanol.com/wp-content/uploads/2016/05/Screen-Shot-2016-05-19-at-00.32.57.png)

Pero la parte negativa de Firebase es que no es totalmente gratis. Es decir, disponemos de 3 planes según nuestras necesidades:

[](https://elandroidelibre.elespanol.com/wp-content/uploads/2016/05/Screen-Shot-2016-05-19-at-00.14.13.png)

Como observareis, **un plan es gratuito**, pero presenta una serie de limitaciones en espacio almacenado y conexiones simultáneas, que quizá no en todos los casos será suficiente para el desarrollador. En estos casos tendremos que pasarnos a otra de las tarifas, entre las que tenemos una tarifa plana de 25 dólares al mes (con límite también), u otra que ya va en función del consumo que tengamos.

Pero a pesar de ello, para todos los que necesitéis hacer uso de la nube es una de las mejores opciones, pues además conseguiréis delegar en Firebase una gran cantidad del desarrollo necesario, ahorrando tiempo (y por tanto, dinero).

**CONCLUSIONES (5 COMO MÍNIMO)**

\_ Se creó una aplicación que haga uso del audio de un dispositivo.

\_Se utilizó SourceTree para facilitarnos el trabajo con GitHub

\_Se guardaron los datos y grabaciones en el Coredata

\_Se realizó una función para eliminar las grabaciones

\_Se elaboró una función para agregar grabaciones