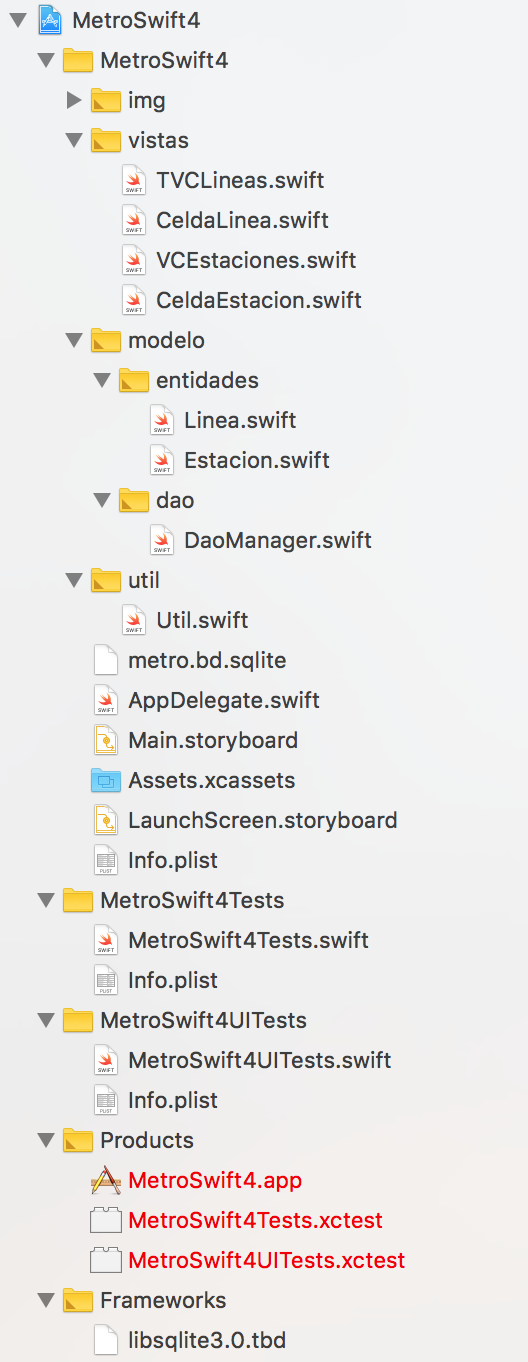
Metro Swift4 sqlite:

**Only sqlite:**

Pasos para preparar todo lo necesario para que la base de datos funcione. El resto es como cualquier otra app:

Navegador de Proyectos:

1. Añadimos la librería libsqlite3.0.tbd desde las propiedades del proyecto linked frameworks and librarys pulsando sobre el símbolo + buscándola y seleccionandola.
2. Sobre la carpeta principal del proyecto añadir la base de datos metro.bd.sqlite
3. Crear la siguiente estructura de carpetas:



1. Eliminar el ViewController.swift y el ViewCotroller del stoyboard
2. Crear un UITableViewController llamado TVCLineas en la carpeta vistas. Sacar un NavigationController desde el Storyboard y asignarle la clase TVCLineas. Indicadle que es el **initial** el TableViewController creado con el NavigationController.
3. Crear la clase Linea en la carpeta entidades como sigue:

class Linea: NSObject {

var id:Int32 = Int32()

var nombre: String = String()

var color: String = String()

var inicioFin: String = String()

func toString() -> String {

return "id:\(id), nombre:\(nombre), color: \(color), inifin: \(inicioFin)"

}

}

1. Añadir la clase Util.swift del profe a la carpeta útil
2. Añadir la clase DaoManager del profe a la carpeta dao
3. Añadir la siguiente línea en negrita en el archivo AppDelegate.swift

func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

// Override point for customization after application launch.

**Util.copyFile("metro.bd.sqlite")**

return true

}

1. Sobre el TVCLineas escribir lo siguiente en en viewDidLoad

let lineas:[Linea] = DaoManager.getInstance().getLineas();

let linea = lineas[0]

print(linea.toString())

Así podremos comprobar si todo ha ido bien.

En la consola debe salir:

id:1, nombre:Línea 1, color: #FF00CCFF, inifin: Optional("Pinar de Chamartín") - Optional("Valdecarros")

\*\*\*\*\*\*\*\*\*\* FIN Only Sqlite \*\*\*\*\*\*\*\*\*\*

Ahora toca hacer la App sabiendo que la Base de Datos es ya accesible…

Storyboard:

1. Diseñar el Prototipo de celda.
   1. Ponerle Reuse Identifier
   2. Sacar las view que necesitamos (ImageView línea y los labels para el numero de línea y para las estaciones iniciales y finales de la línea).

Código:

1. Crear las clase CeldaLinea que herece de UITableViewCell (CeldaLinea.swift).
2. En la carpeta entidades crea la clase Estacion con el siguiente código:

class Estacion: NSObject {

var id:Int = Int()

var nombre:String = String()

var correspondencias : [String] = [String]()

}

Storyboard:

1. Poner a la celda la clase CeldaLinea.swift en su propiedad **class** de la categoría de custom class de la tercera pestaña.
2. Sacar un viewcontroller para la segunda pantalla de estaciones de una línea.
3. Conectar la celda con este viewcontroler que acabamos de sacar pulsando la tecla control desde la celda y soltándolo en el viewcontroller. Nos saldrá un menú contestual y responderemos con la opción, selection segue **show**
4. Seleccionando la flechita segue que nos ha creado, le pondremos un identificador al mismo, desde la 4 pestaña, en la propiedad **identifier** de la categoría Storyboard Segue (segueVerEstaciones).
5. Conectamos los tres elementos de la celda ivLinea, lblNLinea, lblInicioFin con la clase CeldaLinea.swift.

Código:

1. Implementar los 2 métodos para poder visualizar el TVCLineas.

override func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {

// #warning Incomplete implementation, return the number of rows

return lineas.count

}

override func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

let cell = tableView.dequeueReusableCell(withIdentifier: "celdaLineaID", for: indexPath) as! CeldaLinea

let linea = lineas[indexPath.row]

cell.foto.image = UIImage(named: "icono\_linea\_\(linea.id)")

cell.lbllinea.text = linea.nombre

cell.lblinifin.text = linea.inicioFin

return cell

}

1. Eliminar el código que metimos antes para hacer la prueba en el viewDidLoad y poner el siguiente (solo negrita):

class TVCLineas: UITableViewController {

**var lineas:[Linea]=[Linea]()**

override func viewDidLoad() {

super.viewDidLoad()

**lineas = DaoManager.getInstance().getLineas();**

}

1. Probar que la primera pantalla ya funciona….
2. A por la segunda …..
3. En TVCLineas implementar el método prepare(for segue

Para pasar información de un controller a otro, lo hacemos a través del segue. Con este accedemos al controller destino y le anotamos los datos que necesita desde el origen accediendo a variables que previamente hemos preparado en el destino. Como cuando salta el segue es cuando pulsamos sobre un elemento del TableView, para saber que cenda pulsaron, accedemos a la misma mediante self.tableView.indexPathForSelectedRow! que es el índice de la pulsada.

override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

if segue.identifier == "segueVerEstaciones" {

// Accedemos al objeto donde nos han pulsado

let indexPath = self.tableView.indexPathForSelectedRow!

let linea = lineas[indexPath.row]

// accedemos al controller detalle

let controllerEstaciones = segue.destination as! TVEstaciones

// Le anotamos desde aquí la línea en una variable que allí preparamos

controllerEstaciones.linea = linea

}

}

Nota: Os regalo una función que convierte el color hexadecimal que viene en la BD en un UIColor que es lo que necesitáis para pintar el background:

// Funcion que convierte color en Hex a UIColor.

func hextoUIColor (\_ hex:String) -> UIColor {

var cString:String = hex

if (cString.hasPrefix("#")) {

cString = ((cString as NSString).substring(from: 3) as NSString).substring(to: 6)

}

if (cString.characters.count != 6) {

return UIColor.gray

}

let rString = (cString as NSString).substring(to: 2)

let gString = ((cString as NSString).substring(from: 2) as NSString).substring(to: 2)

let bString = ((cString as NSString).substring(from: 4) as NSString).substring(to: 2)

var r:CUnsignedInt = 0, g:CUnsignedInt = 0, b:CUnsignedInt = 0;

Scanner(string: rString).scanHexInt32(&r)

Scanner(string: gString).scanHexInt32(&g)

Scanner(string: bString).scanHexInt32(&b)

return UIColor(red: CGFloat(r) / 255.0, green: CGFloat(g) / 255.0, blue: CGFloat(b) / 255.0, alpha: CGFloat(1))

}

Notas:

Para poner color de fondo a una pantalla:

self.view.backgroundColor = ...

Para poner el fondo a una celda:

cell.backgroundColor = …

Para añadir dinámicamente una uiimagenview a una celda:

let imageView = UIImageView(frame: CGRect(x: 200+(40\*i), y: 0, width: 40, height: 40))

let image = UIImage(named: "icono\_linea\_\(corr)")

imageView.image = image

imageView.tag = 1

cell.contentView.addSubview(imageView)

Para eliminar las imagenes de las celdas:

// El siguiente if es necesario para que cuando se haga scroll

// en el tableview las fotos no se mezclen

// Si quereis ver el efecto, poner entre comentario este if

// y hacer scrool con el dedo, vereis como salen fotos

// de correspondencias inventadas

if (cell.contentView.subviews.count>1) {

for subview in cell.contentView.subviews {

if subview.tag == 1 { // solo las fotos tienen tag = 1

subview.removeFromSuperview()

}

}

}