POLITÉCNICA

ETSIT UPM

dit UPM

Desarrollo de Apps para iOS

Demo Game Top 3

IWEB 2017-2018

Santiago Pavón

ver: 2017.11.15

Índice

- Desarrollo de un servidor web para usar en los ejemplos de este tema.
  - Guarda los puntos obtenidos por los usuarios en un juego.
  - El servidor tiene un API REST.
  - Desarrollo del servidor con nodejs.
- Desarrollo de una aplicación que baje las tres mejores puntuaciones.
  - Paso 1: Desarrollar el GUI de la aplicación usando una TableView.
  - Paso 2: Implementar una descarga de datos JSON usando:
    - Opción 1: NSData(contentsOfURL:).
    - Opción 2: NSURLSession y DataTask.
- Realizar una aplicación que suba nuevas puntuaciones al servidor.

Desarrollar el Servidor Web Game Top 3

Desarrollo con Nodejs

El Servicio Web

- Servicio Web
  - Almacena nombres y puntuaciones obtenidas en las partidas de un juego
  - Consultar las 3 mejores puntuaciones.
- Servicio Web RESTful implementado con Nodejs.

Pistas para el Desarrollo

- Instalar Nodejs
- Instalar el paquete express-generator:

```
$ npm install express-generator
```

- Crear esqueleto de la aplicación, e instalar dependencias:

```
$ ./node_modules/.bin/express --ejs gametop3
$ cd gametop3
$ npm install
```

- Limpieza: eliminar rutas y recurso user
  - Borrar el fichero `routes/user.js`
  - Borrar de `app.js` las sentencias:

```
var users = require('./routes/users');
app.use('/users', users);
```

- Crear marco de aplicación:
  - Instalar el paquete express-partials

```
$ npm install --save express-partials
```

    - Añadir en `app.js` la carga e inicialización de este paquete:

```
var partials = require('express-partials');

app.use(partials());
```

  - Crear el fichero con el marco `views/layout.ejs`
  - Eliminar de `views/index.ejs` todo lo que debe ir en el marco.

- Instalar el paquete method-override

```
$ npm install --save method-override
```

  - Añadir en app.js la carga e inicialización de este paquete:

```
var methodOverride = require('method-override');
app.use(methodOverride('_method', {methods: ["POST", "GET"]}));
```

- Instalar los paquetes sequelize, sequelize-cli, sqlite3

```
$ npm install --save sequelize
$ npm install --save sequelize-cli
$ npm install --save sqlite3
```

- Crear los ficheros `models/index.js` y `models/score.js` para la definición de los modelos de datos.
- Crear una migración para la tabla Scores

```
$ ./node_modules/.bin/sequelize init:migrations
$ ./node_modules/.bin/sequelize migration:create
      --name CreateScoresTable
      --url sqlite://$(pwd)/scores.sqlite
```

  - Editar `migrations/#######-CreateScoresTable.js` y rellenar los métodos up y down.
  - Aplicar las migraciones:

```
$ ./node_modules/.bin/sequelize db:migrate
            --url sqlite://$(pwd)/scores.sqlite
```

  - Nota: para que la opción --url usada en los comandos anteriores funcione, la ruta al fichero con la base de datos no puede tener espacios en blanco (ni otros caracteres *raros*.)

- Crear las rutas
  - Añadirlas a `routes/index.js`
- Crear el controlador `controllers/score_controller.js`
  - Añadir los middlewares de las rutas.
- Crear las vistas en `views/scores`
  - Crear los ficheros:
    - `index.ejs`, `show.ejs`, `edit.ejs`, `new.ejs`, y `_form.ejs.ejs`
- Los fuentes del proyecto están disponibles en el mismo directorio que estos apuntes.
  - Consultarlos para el contenido de todos los ficheros mencionados anteriormente.

- Lanzar el servidor:

```
$ npm start
```

  - No olvidar instalar antes las dependencias de paquetes y aplicar la migración.

```
$ npm install
```

```
$ ./node_modules/.bin/sequelize db:migrate
          --url sqlite://$(pwd)/scores.sqlite
```

- Recuperar datos de las 3 mejores puntuaciones:
  - Desde un navegador conectarse a:

```
http://localhost:3000/scores/top3
```

  - Para ver la versión JSON:

```
http://localhost:3000/scores/top3.json
```

```
[{"id":13,
  "name":"Manolo",
  "total":555,
  "createdAt":"2016-11-15T17:52:05.199Z",
  "updatedAt":"2016-11-16T16:53:59.425Z"},
 {"id":14,
  "name":"Gerardo",
  "total":1,
  "createdAt":"2016-11-15T17:52:13.593Z",
  "updatedAt":"2016-11-16T16:53:49.695Z"},
 {"id":15,
  "name":"Carla",
  "total":0,
  "createdAt":"2016-11-15T17:52:19.489Z",
  "updatedAt":"2016-11-16T16:53:40.358Z"},
 {"id":16,
  "name":"Luis",
  "total":70,
  "createdAt":"2016-11-15T17:54:30.794Z",
  "updatedAt":"2016-11-16T16:53:31.482Z"}
]
```

- **Crear un registro nuevo:**
  - Desde un formulario:

```
$ curl -d '_method=POST&name=Nuria&total=53'
       http://localhost:3000/scores
```

  - Con datos JSON:

```
$ curl -d '{"name":"Heliodora","total": 51}'
       -H 'Content-type: application/json'
       http://localhost:3000/scores.json
```

- **Editar un registro:**
  - Desde un formulario:

```
$ curl -d 'total=153'
       http://localhost:3000/scores/13?_method=PUT
```

  - Con datos JSON:

```
$ curl -d '{"total": 21}'
       -H 'Content-type: application/json'
       http://localhost:3000/scores/13?_method=PUT
```

- **Borrar un registro:**

```
$ curl http://localhost:3000/scores/13?_method=DELETE
```

App para Mostrar el Top3

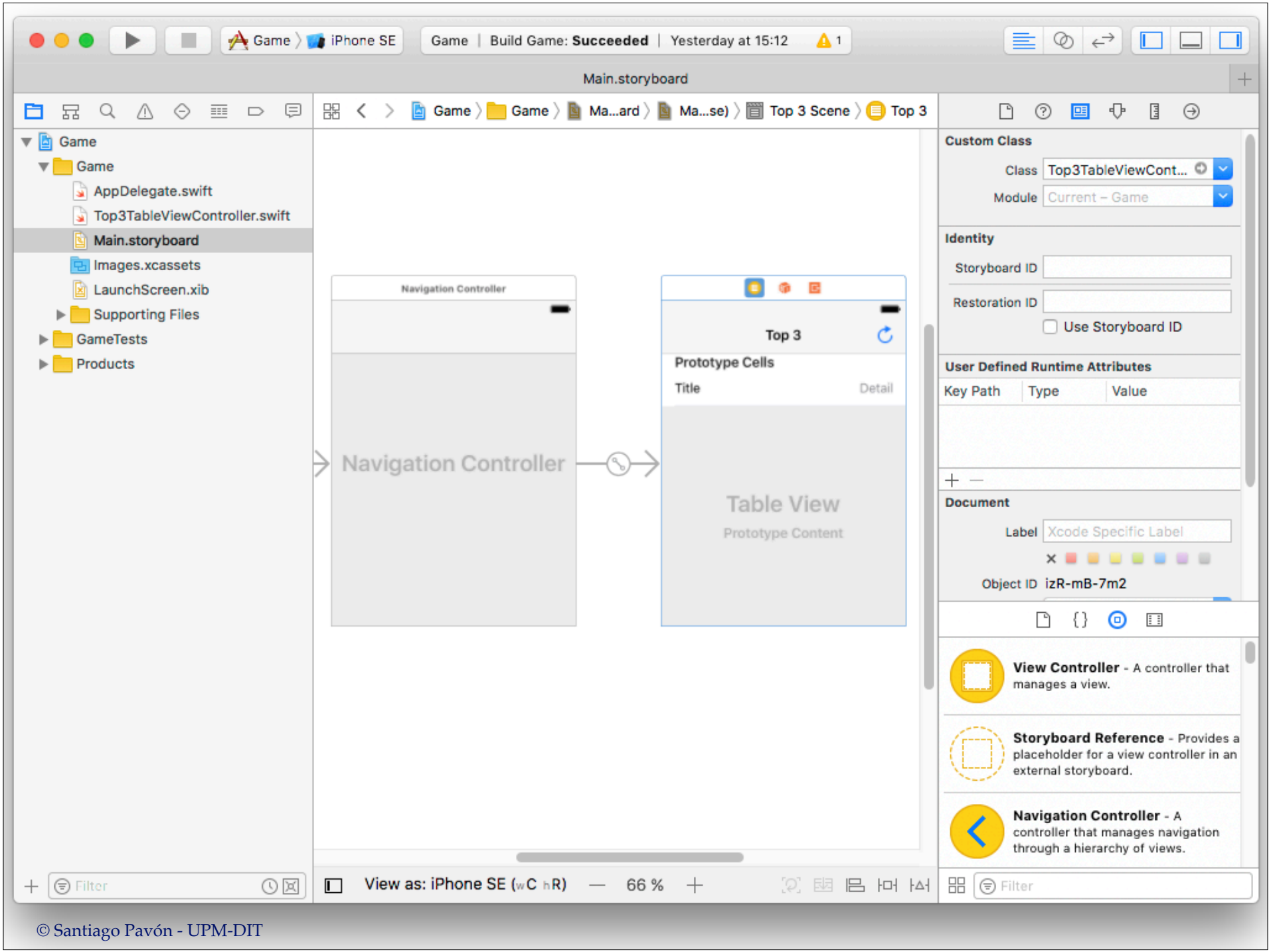
Crear Aplicación: Game

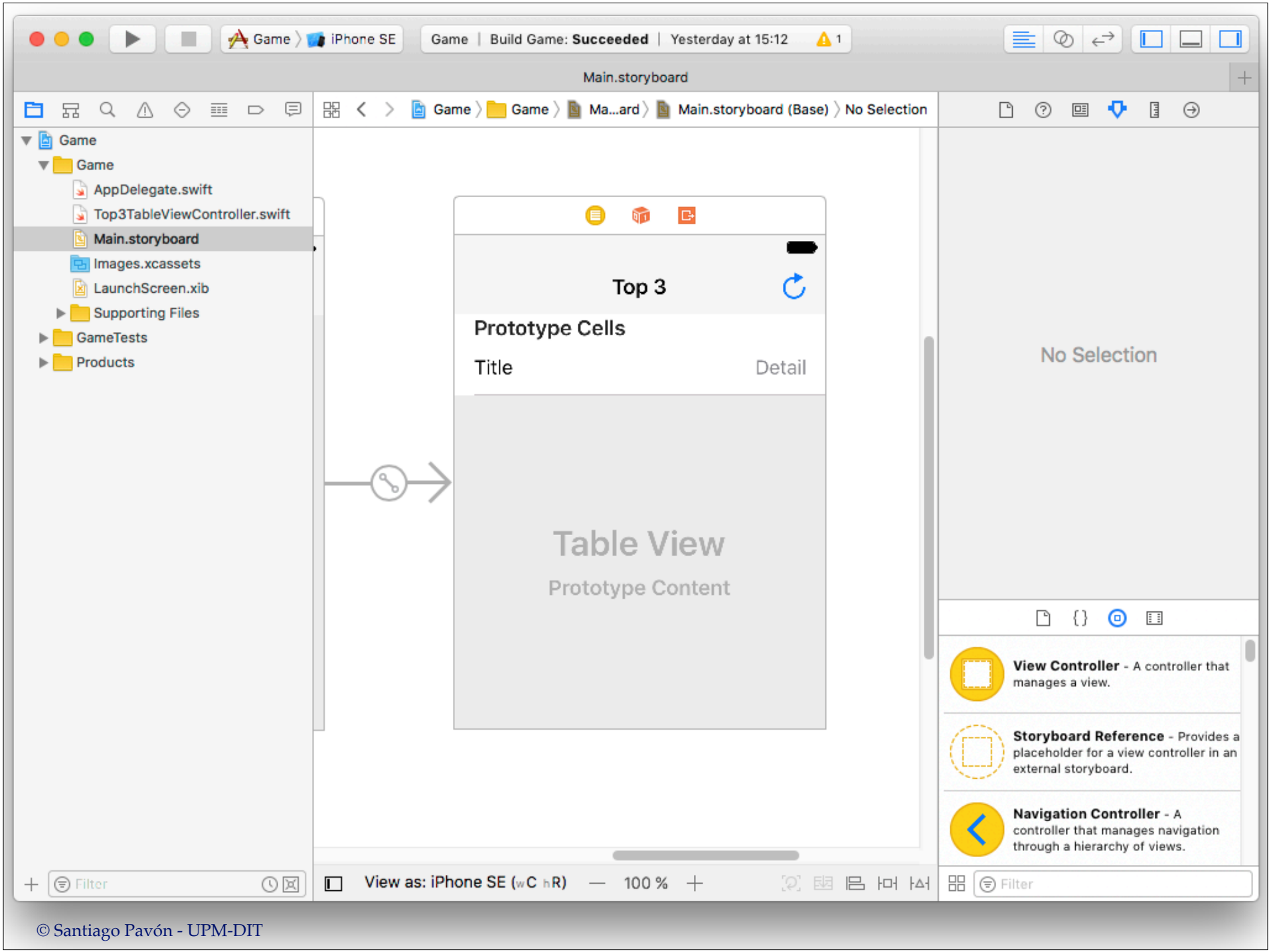
- En esta aplicación mostraremos los datos obtenidos al acceder al recurso **`scores/top3`** del servicio web.

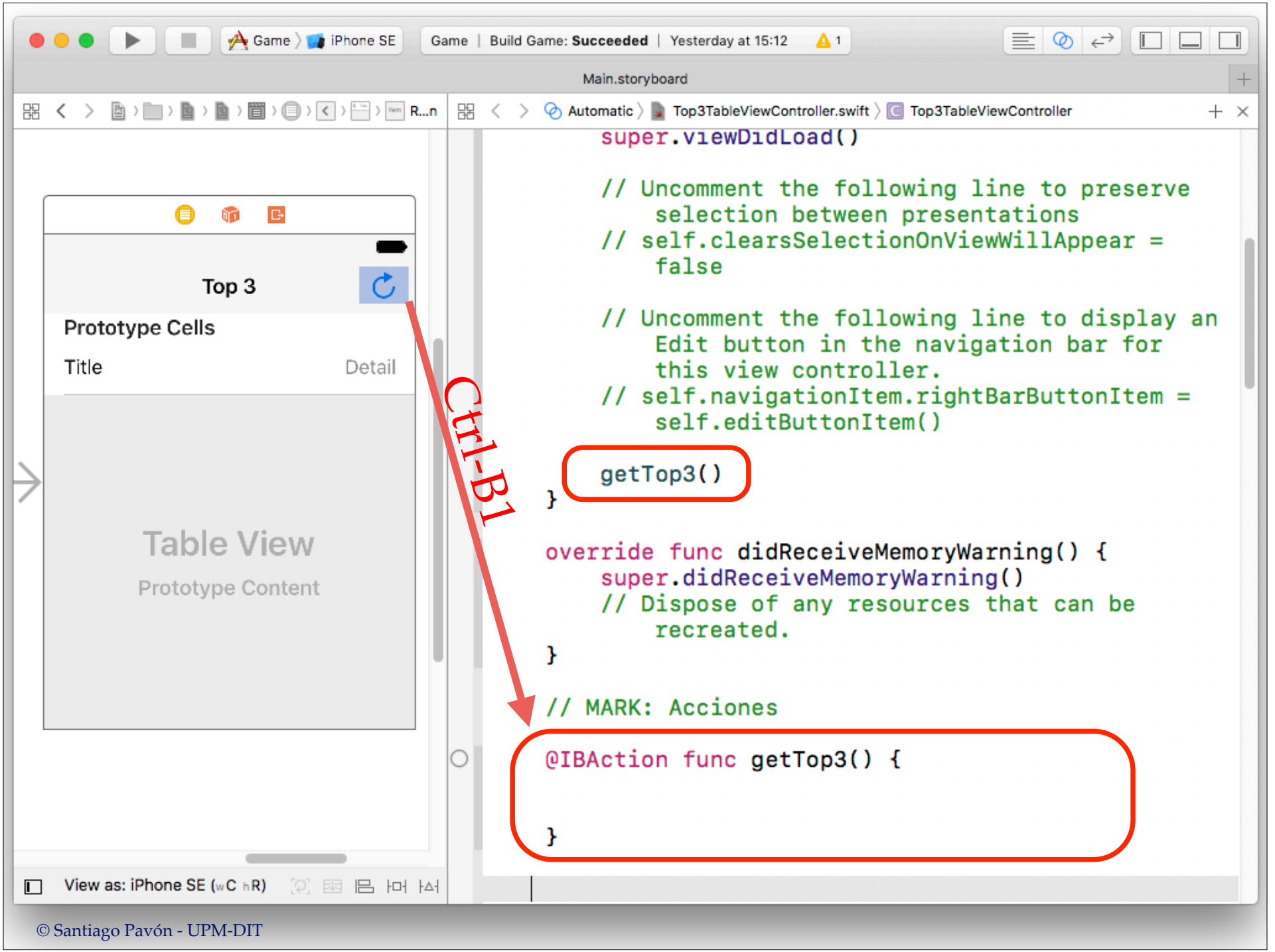
- Crear un nuevo proyecto iOS.
  - Usar plantilla Single View Application
    - Nombre de la aplicación: Game
    - La plantilla crea una clase ViewController
      - Borrar esta clase y su escena del storyboard.
    - Crear una clase nueva llamada Top3TableViewController
      - que derive de UITableViewController
    - Añadir al storyboard una escena TableViewController para la clase Top3TableViewController.
    - Meter esta escena en un Navigation Controller.

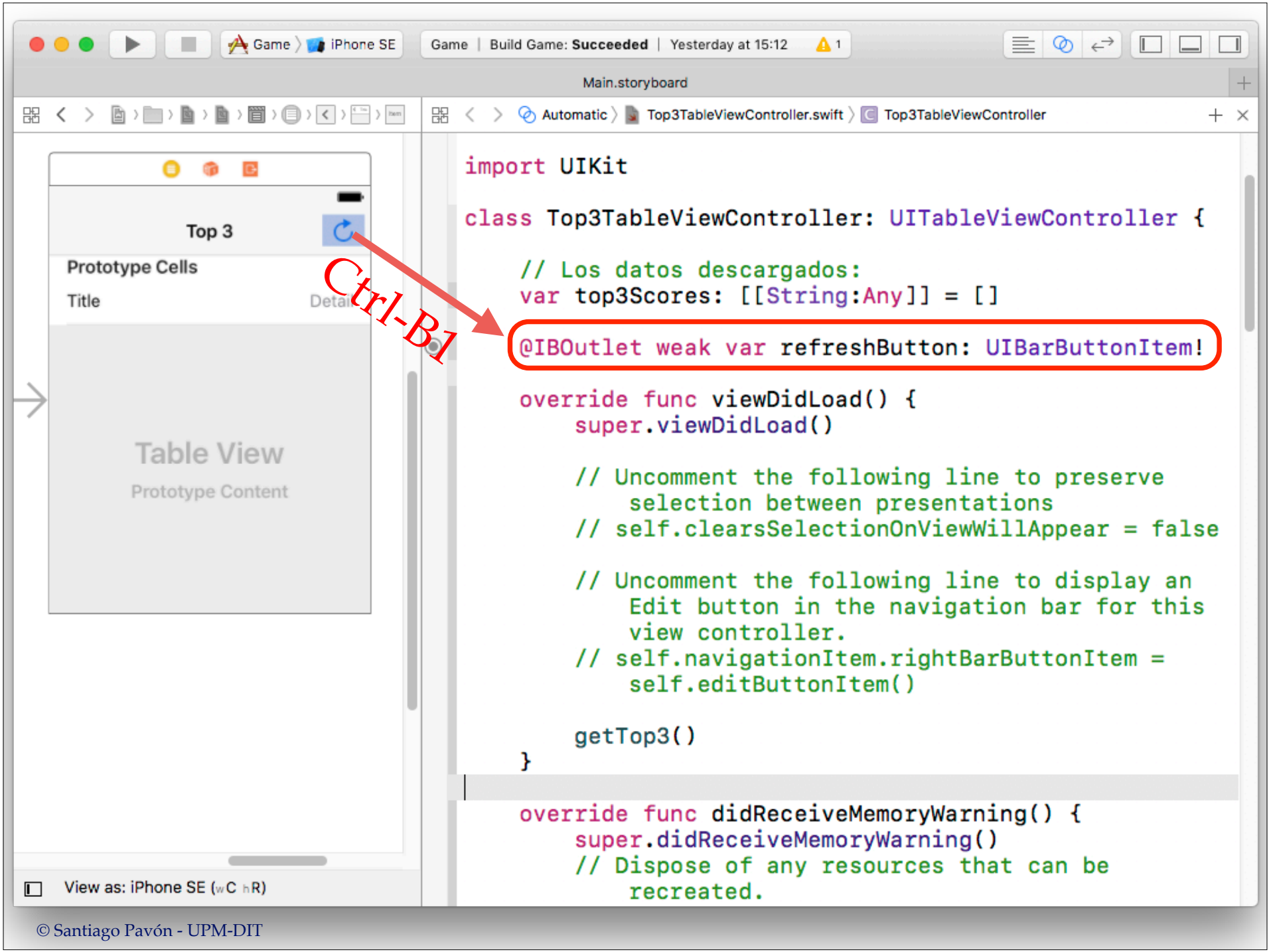
- Editar el storyboard:
  - Editar los atributos del prototipo de celda:
    - Cambiar el estilo de celda a **Right Detail**.
    - Cambiar el identificador de celda a "**Score Cell**".
  - Poner **Top 3** como título de la barra de navegación.
  - Añadir un Bar Button Item (con identificador **Refresh**) para recargar las puntuaciones.
    - Crear una IBAction para este botón llamada **getTop3**.
    - Llamar al método **getTop3** desde **viewDidLoad** para cargar los datos inicialmente.
    - Crear un outlet a ese botón para poder deshabilitarlo programáticamente.
- Editar Info.plist para deshabilitar ATS para todos los dominios:

```
▼ NSAppTransportSecurity
      NSAllowsArbitraryLoads = YES
```









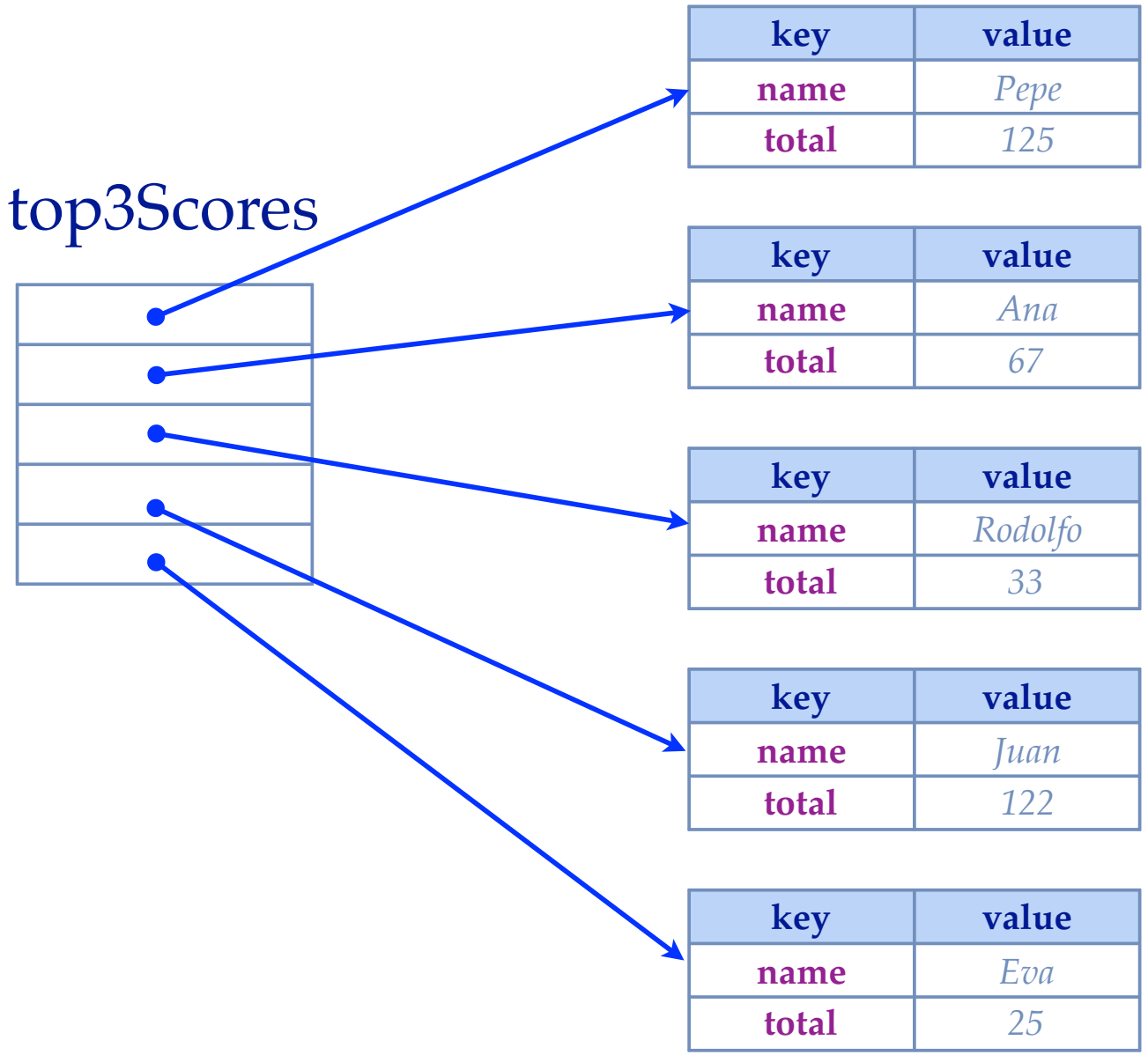
La propiedad top3Scores

- Añadir una propiedad en **Top3TableViewController** para almacenar los datos descargados del servidor web.

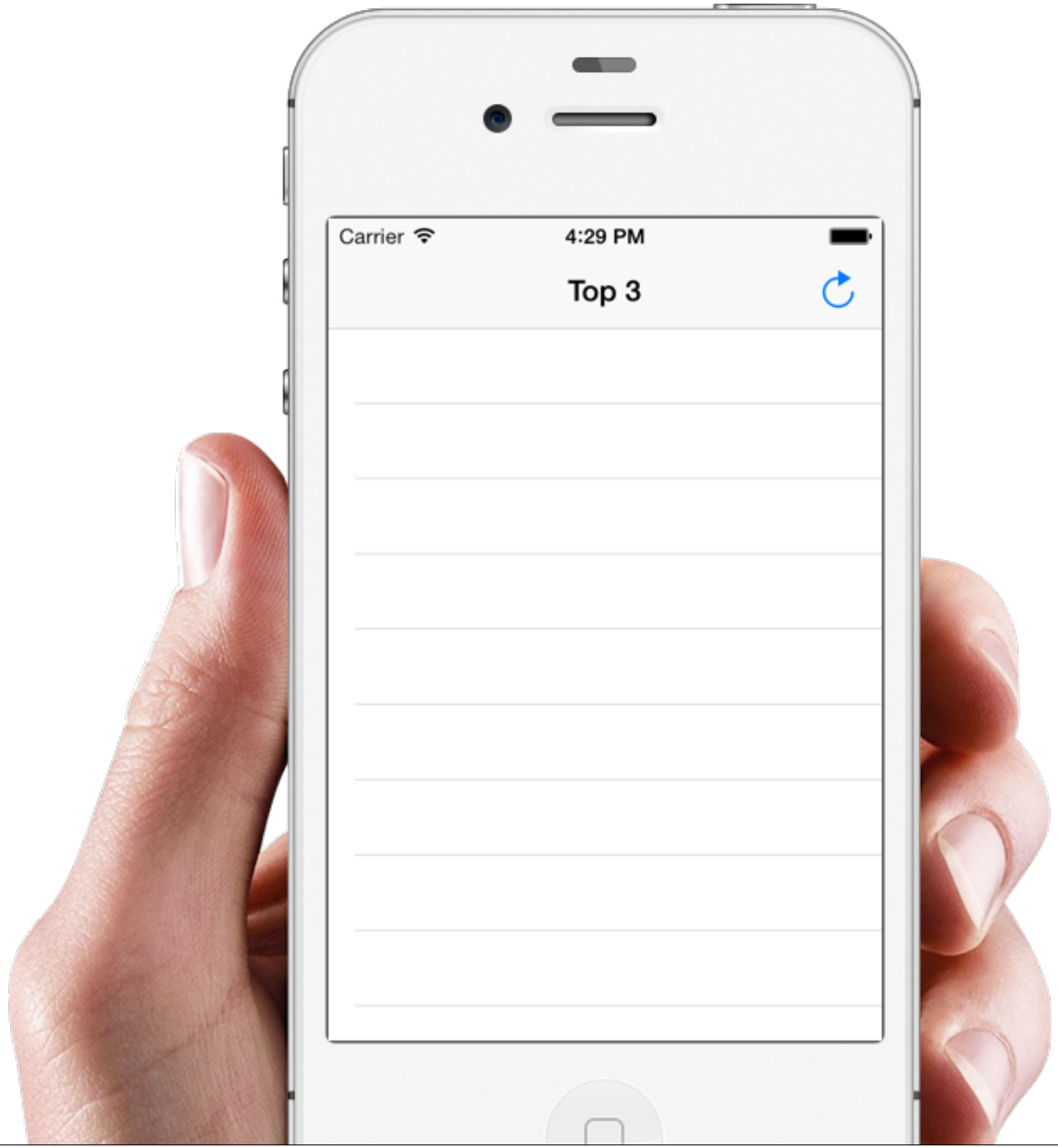
```
var top3Scores: [[String:Any]] = []
```

  - Es un array de diccionarios.
  - Usado por el data source de la tabla.

- Los datos descargados son diccionarios.
  - Un diccionario para cada puntuación.
  - Cada diccionario almacena un nombre y una puntuación
    - Usar la clave "**name**" para acceder al nombre del jugador.
    - Usar la clave "**total**" para acceder a la puntuación.



```
override func numberOfSections(in tableView: UITableView) -> Int {
    return 1
}

override func tableView(_ tableView: UITableView,
                        numberOfRowsInSection section: Int) -> Int {
    return top3Scores.count
}

override func tableView(_ tableView: UITableView,
                        cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "Score Cell",
                                             for: indexPath)

    let dic = top3Scores[indexPath.row]

    cell.textLabel?.text = dic["name"] as? String

    let total = dic["total"] as! Int
    cell.detailTextLabel?.text = "\(total)"

    return cell
}
```



Descargar Datos JSON con `Data(contentsOf url:)`

Descargar JSON

- Para descargar el contenido JSON de una URL:

```
let GAME_URL = "http://localhost:3000/scores/top3.json"
let url = URL(string: GAME_URL)!
let jsonData = try Data(contentsOf: url)
```

- **jsonData** es un buffer con los datos JSON.
- Extraemos los datos:

```
let newTop3 = try JSONSerialization.jsonObject(with: jsonData)
                                as? [[String:Any]]
```

La acción getTop3

```
let GAME_URL = "http://localhost:3000/scores/top3.json"

@IBAction func getTop3() {
    title = "Descargando..."
    refreshButton.isEnabled = false
    UIApplication.shared.isNetworkActivityIndicatorVisible = true

    let url = URL(string: GAME_URL)!

    if let jsonData = try? Data(contentsOf: url),
       let newTop3 = (try? JSONSerialization.jsonObject(with: jsonData))
                                                 as? [[String:Any]] {
        top3Scores = newTop3
        tableView.reloadData()
        title = "Top 3"
    } else {
       title = "Desactualizado"
    }

    self.refreshButton.isEnabled = true
    UIApplication.shared.isNetworkActivityIndicatorVisible = false
}
```

La acción getTop3 (GCD)

```
@IBAction func getTop3() {
    title = "Descargando..."
    refreshButton.isEnabled = false
    UIApplication.shared.isNetworkActivityIndicatorVisible = true

    let queue = DispatchQueue(label: "download queue")
    queue.async {
        let url = URL(string: GAME_URL)!

        if let jsonData = try? Data(contentsOf: url)
            let newTop3 = (try? JSONSerialization.jsonObject(with: jsonData))
                                                              as? [[String:Any]] {
            DispatchQueue.main.async {
                self.top3Scores = newTop3!
                self.tableView.reloadData()
                self.title = "Top 3"
            }
        } else {
            DispatchQueue.main.async {
               self.title = "Desactualizado"
            }
        }
        DispatchQueue.main.async {
            self.refreshButton.isEnabled = true
            UIApplication.shared.isNetworkActivityIndicatorVisible = false
        }
    }
}
```

getTop3 + Codable

Swift 4

```
struct Score : Codable {
    let name: String
    let total: Int
}

var top3Scores = [Score]()

func getTop3() {
   let newTop3 = try? JSONDecoder().decode([Score].self, from: jsonData)
}

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

  let score = top3Scores[indexPath.row]

  cell.textLabel?.text = score.name
  cell.detailTextLabel?.text = "\(score.total)"
}
```

Nuevo tipo para los valores a decodificar

Array de Score

Decodificar un Data que contiene un JSON

Datos para rellenar la celda

URLSession
Descargar Datos con un DataTask

Realizar una petición GET

- Objetivo:
  - Hacer una petición HTTP de tipo GET para descargar unos datos de una URL.
- Los pasos a seguir son:
  - Crear y configurar una sesión URLSession para todas la tareas que se creen en un futuro.
  - Crear la URL.
    - No hay caracteres conflictivos que sea necesario escapar.
  - Cada vez que se quieran descargar los datos hay que crear una tarea URLSessionDataTask.
    - Al terminar, ejecutará el completionHandler pasado como parámetro.
    - La tarea empieza suspendida; hay que llamar a resume().

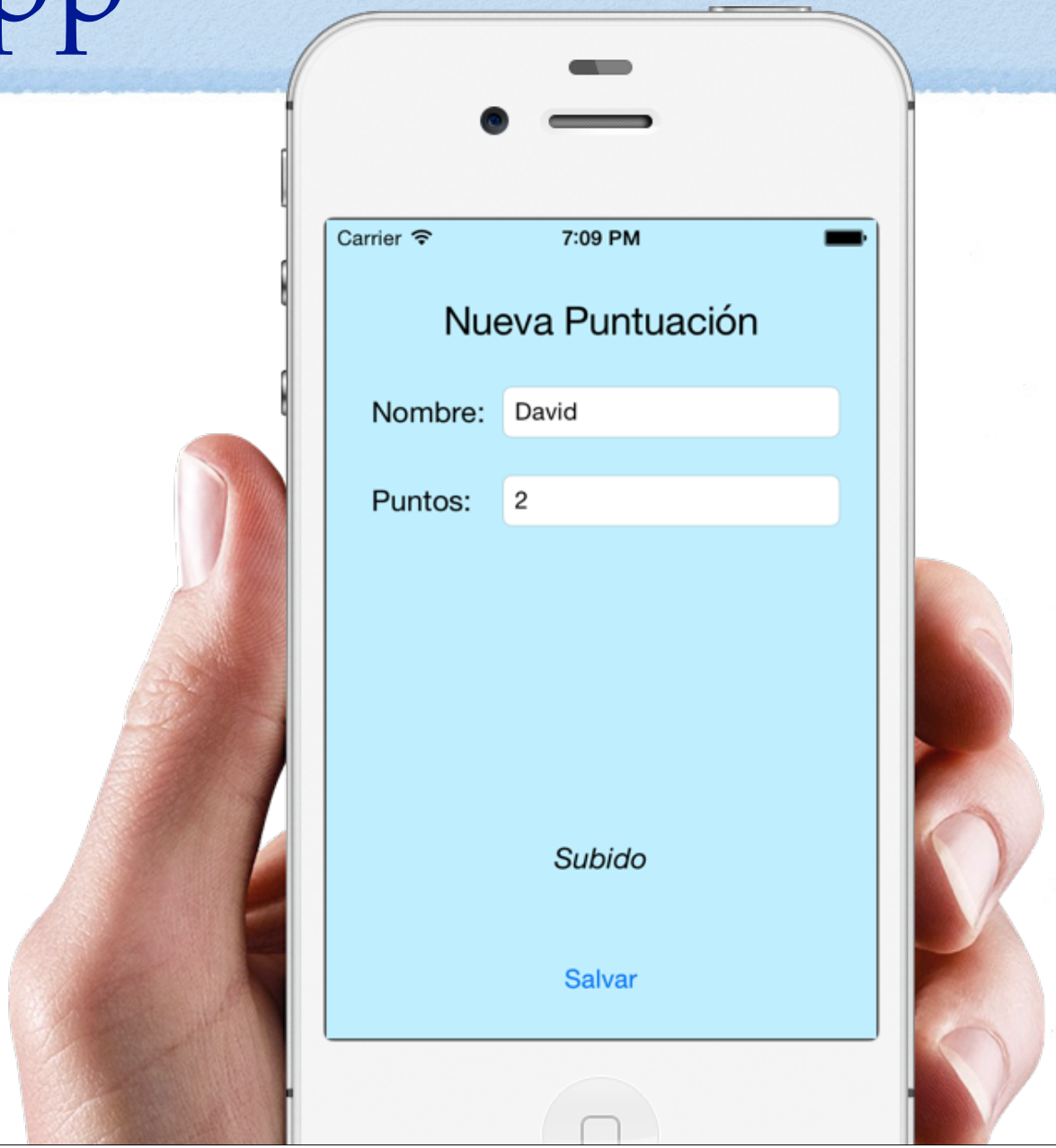
```swift
let GAME_URL = "http://localhost:3000/scores/top3.json"

// La session
var session: URLSession!

override func viewDidLoad() {
    super.viewDidLoad()

    // Crear la session
    let config = URLSessionConfiguration.default
    session = URLSession(configuration: config)

    // Descarga inicial de datos
    getTop3()
}

@IBAction func getTop3() {

    title = "Descargando..."
    refreshButton.isEnabled = false
    UIApplication.shared.isNetworkActivityIndicatorVisible = true

    // Construir la URL
    let url = URL(string: GAME_URL)!

   // Continua la funcion getTop3 ->
```

```
    // Crear Data Task
    let dataTask = session.dataTask(with: url) { (data: Data?,
                                                  response: URLResponse?,
                                                  error: Error?) in
        var newTop3: [[String:Any]]?

        if error == nil,
           (response as? HTTPURLResponse)?.statusCode == 200 {
            newTop3 = (try? JSONSerialization.jsonObject(with: data!))
                                                    as? [[String:Any]]
        }

        // El completionHandler no corre en el Main Thread
        DispatchQueue.main.async {
            if newTop3 == nil {
                self.title = "ERROR"
            } else {
                self.top3Scores = newTop3!
                self.tableView.reloadData()
                self.title = "Top 3"
            }
            self.refreshButton.isEnabled = true
            UIApplication.shared.isNetworkActivityIndicatorVisible = false
        }
    }
    dataTask.resume()
}
```

URLSession

Subir Nueva Puntuación JSON

Nueva App



Creamos un **URLSessionUploadTask**

- La **URL** donde subimos es **`http://localhost:3000/scores.json`**
- La petición HTTP:
  - Usa el **método** es **POST**
  - y pone la siguiente cabecera **Content-type**:

```
application/json; charset=utf-8
```

- El dato a subir es un Data obtenido convirtiendo en JSON un objeto swift:

```
[ "name": "Pedro",
  "total": 125 ]
```

- Creamos una **URLSession** y creamos una tarea **URLSessionUploadTask** para subir el Data anterior.
  - Usamos un completionHandler que se invoca cuando la tarea ha terminado.
  - Inicialmente la tarea está suspendida y tenemos que reanudarla (resume).
  - Devuelve un JSON con todos los campos del registro creado.

```
@IBAction func save() {

    statusLabel.text = "Salvando Puntuación"
    UIApplication.shared.isNetworkActivityIndicatorVisible = true

    //--- Datos a subir:

    if nameTextField.text == "" {
        nameTextField.text = "Anónimo"
    }
    if totalTextField.text == "" {
        totalTextField.text = "0"
    }

    let score = ["name": nameTextField.text!,
                 "total": totalTextField.text!]

    guard let dataScore = try?
              JSONSerialization.data(withJSONObject: score) else {
        return
    }
```

```
//--- URL destino:
let SCORES_URL = "http://localhost:3000/scores.json"
let url = URL(string: SCORES_URL)!


//--- La peticion HTTP:
var request = URLRequest(url: url)
request.httpMethod = "POST"
request.addValue("application/json; charset=utf-8",
                 forHTTPHeaderField: "Content-Type")


//--- La sesion:
let sessionConf = URLSessionConfiguration.ephemeral
let session = URLSession(configuration: sessionConf)
```

```
    //--- La tarea para subir los datos:
    let uploadTask = session.uploadTask(with: request, from: dataScore) {
        (data: Data?, response: URLResponse?, error: Error?) -> Void in

        // El completionHandler no corre en el Main Thread
        DispatchQueue.main.async {
            if error != nil {
                self.statusLabel.text = error!.localizedDescription
            } else if let res = response as? HTTPURLResponse,
                res.statusCode != 200 && res.statusCode != 201 {
                let msg = HTTPURLResponse.localizedString(forStatusCode:
                                                          res.statusCode)
                self.statusLabel.text = msg
            } else {
                self.statusLabel.text = "Subido"
            }
            UIApplication.shared.isNetworkActivityIndicatorVisible = false
        }
    }
    uploadTask.resume()
}
```