Time and Notification Assignment Solution

<https://github.com/rodleyva/iOS_Solutions>

Here is the Link to the endpoints you can user for the Bucketlist Assignments:

<https://www.getpostman.com/collections/a4b76f4a553a18b8eb44>

Small API with SQLite database that saves notes

<https://github.com/rodleyva/BucketServer>

API Back End can be create

Create a Back-end Server using Codeigniter, Pylot/Django, Express, or Rails

**Faster UI**

Great! We've completed the views to view all of the people in star wars as well as all of the films. The only thing missing is a nice user experience. **Right now it takes too long for our data to load in the table view!**

To understand why we need to understand a little bit about the Grand Central Dispatch.

**Grand Central Dispatch**

GCD is a C API that interfaces with the threads on your computer to manage processes. The GCD manages queues of tasks to run where each of the queues runs concurrently (at the same time) and each of the tasks within a queue runs synchronously. In iOS by default, the GCD has 4 queues that it manages with one completely dedicated to UI related tasks.

Behind the scenes, iOS does everything it can to make the user experience of any application the best possible. Because of this, it keeps all UI related tasks on one queue and switches to the other queues for UI unrelated tasks such as HTTP requests!

In order to make our application faster, all we need to do is make sure that the actual tableView.reloadData() function is run on the main queue that runs the UI. Conveniently enough iOS calls this queue the "main queue".

Let's see this in action:

**peopleViewController.swift**

override func viewDidLoad() {

super.viewDidLoad()

// Specify the url that we will be sending the GET Request to

let url = URL(string: "https://swapi.dev/api/people/?format=json")

// Create a URLSession to handle the request tasks

let session = URLSession.shared

// Create a "data task" which will request some data from a URL and then run a completion handler after it is done

let task = session.dataTask(with: url!, completionHandler: {

data, response, error in

// We get data, response, and error back. Data contains the JSON data, Response contains the headers and other information about the response, and Error contains an error if one occured

// A "Do-Try-Catch" block involves a try statement with some catch block for catching any errors thrown by the try statement.

do {

// Try converting the JSON object to "Foundation Types" (NSDictionary, NSArray, NSString, etc.)

if let jsonResult = try JSONSerialization.jsonObject(with: data!, options: JSONSerialization.ReadingOptions.mutableContainers) as? NSDictionary {

if let results = jsonResult["results"] as? NSArray {

for person in results {

// cast to dictionary for data extraction

let personDict = person as! NSDictionary

self.people.append(personDict["name"]! as String)

}

}

DispatchQueue.main.async {

self.tableView.reloadData()

}

}

} catch {

print("Something went wrong")

}

})

// Actually "execute" the task. This is the line that actually makes the request that we set up above

task.resume()

print("I happen before the response!")

}

Now if we run our application the UI loads much faster than before! **Go ahead and make the same change to your Film Controller to fix it as well.**

**Remember that all UI related tasks *and only UI related tasks* belong in the main queue. typically your requests will look like:**

let url = URL(string: "http://www.some-api-url.com")

let session = URLSession.shared

let task = session.dataTask(with: url!, completionHandler: {

data, response, error in

// Do something here with the data from the response

DispatchQueue.main.async {

// Do something here to update the UI

}

})

task.resume()