Lifting Log Developers Guide

Joe Melito and Neil Kalanish

Overview 3
Technology 3
Views 3
Structures 4
Services 4
Extensions 5
Architecture 5
Views 5
    ViewController 5
    CreateAccountController 6
    WelcomeDashboardController 6
    SettingsController 6
    NewWorkoutView 7
    PastWorkoutsController 7
    DetailViewController 8
    WilksCalculatorController 8
    OneRMCalculatorViewController 8
Services 8
    CalculationService 8
    DatabaseService 9
    AuthenticationService 10
Database 10

Overview

Technology

The goal of the lifting log app was to provide serious lifters an application they can use to track their workouts and finally replace pen and paper logs. All other lifting applications on the market try to cater to different types of activities and workouts so Joe and I set out to create an application for one activity and do it the best without all of the other fluff.

- Language: Swift version 5.7
- IDE: Xcode
- Source Control: Github (https://github.com/Neilkal867/LiftingLog)
- Database: Firebase Firestore Database
- Authentication: Firebase Authentication
- Creators: Neil Kalanish and Joe Melito

Views

The app has nine different controllers, located under in the “Views” folder, that control the UI and the functionality of each object.

- ViewController (The Login screen)
- CreateAccountController (Allows the user to create an account)
- WelcomeDashBoardController (The main screen of the app with all the options)
- SettingsController (The settings screen)
- NewWorkoutView (Allows the user to log workouts)

- PastWorkoutsController (Displays the dates of past workouts)
- DetailViewController (Displays the details of a selected workout)
- WilksCalculatorController (Allows the user to calculate their wilks number)
- OneRmCalculatorViewController (The view for handling the one rep max calculator)

Structures

We created three different structures to help pass data around and keep it clean. Each structure is in it's own file located in the "Structures" folder.

- WorkoutStruct - Contains the type "Workout" which consist of a date, workoutType, weight, reps, sets and any comments.
- WorkoutsStruct - Contains the type "Workouts" which has one variable "results" which is an array of type Workout.
- AuthResponseStruct - Contains the type Authresponse and this is what's returned from the authentication service when a user logs in. It contains a bool if the user logging in successfully or not and if there is an error present that message will be passed along too.

Services

The controllers rely on services to perform all of the work they are asked to do. Services make all database calls, authenticate users and perform all calculations which allows the controllers to just display the data. There are three services located in the "Services" folder.

- CalculationService - Handles all of the math for the wilks calculator and one rep max calculator.

- AuthenticationService - Handles all of the calls and responses to and from the Firebase authentication service. It returns the Authresponse with a bool if the user logged in and if not what the error was.
- DatabaseService - This service is what makes all of the calls to the Firebase db and handles formatting the data properly for saving or sending it back to the view to be displayed.

Extensions

We created an UIViewControllerExtension in the Extensions folder to help with different functionality across the views. The UIViewController helps with displaying and interacting with alerts, calling different views to help with navigation and hiding / unhiding the keyboard in different instances.

Architecture

Views

ViewController

The view control controls the main login screen which allows a user to log in or create an account. The create account button automatically takes you to the CreateAccountController but clicking Log in calls the logInClicked() function. That function verifies that the username and password fields aren't empty then calls the LogUserIn() function in the Authentication service. If the user is successfully logged in it navigates to the WelcomeDashBoardController. On the chance the user credentials aren't values the Authentication Service will return false and the error message to be displayed to the user.

CreateAccountController

The create account controller handles the creation of new users. Currently the app only uses email and password to log in so the first name, last name and date of birthday aren't used for anything. There are a few functions to handle the date and formate it correctly but the function createUser() is the big function that takes in the email address and password then calls the Authentication service to create the account. If there are any errors like the email isn't formatted correctly, password is too short or account already exists, the authentication service will return an error with that information.

WelcomeDashboardController

This is the main dashboard and handles the creation of all the menu options. It uses a UITableViewController to handle switching to the different views. This controller doesn't provide any functionality itself it just shows the different menus available and calls the appropriate controller when a user selects ones.

SettingsController

Currently the only two options available in the settings controller is log out which takes the user back to the login screen and delete account which actually deletes the account. When the user clicks the delete account button it calls the deleteAccount function which calls into the UIViewController to display a warning that deleting is permanent. If the user clicks cancel nothing happens if they click ok their account is deleted and cannot be recovered.

NewWorkoutView

This is the view that allows the user to save their workouts. It consists over several input boxes for Workout Type, Weight, Number of Reps, Number of Sets and Comments. All of the fields are required to save a workout except the comments. Weight, Number of Reps and Number of Sets must be integers and those fields force a number only keyboard to prevent any letters from being entered. The workout Type and Comments can be alphanumeric and the user will have full access to either keyboard. There are no minimum or max length set on anything.

The view also has two buttons, cancel and submit. When the user clicks submit the submitNewWorkout() function is called which immediately calls the SubmitWorkout() function that pulls all of the information from the text fields, formats correctly then calls the database service to save the workout. If any field is empty, except the comments, the user is given an error to finish filling the form out. After the information is saved clearTextFields() is called and all of the values are cleared.

PastWorkoutsController

The PastWorkoutsController is based off of the welcomedashboard because it is just a table view that gets populated based off the listofworkouts array. The listofworkouts array is just a list of all of the dates of previous workouts. When the user clicks a date it takes them to the DetailViewController which then displays all of the details of the workout on that specific day. The

array is initialized with all workouts as soon as the user logs in so the view loads immediately without any lag.

DetailViewController

WilksCalculatorController

The wilks calculator controller is a simple view that has a toggle for Male or Female then a view text fields that take in the users bodyweight, max bench, max squat and max deadlift then calls the calculation service with those values to get the wilks number and display it. This controller inherits from the UIViewController class to display alerts and when the calculate button is clicked it calls the calculateWilksButton() function which verifies all values are filled out and then calls the calculator service to get the value. The values must all be integers and if the view only allows access to the numeric keyboard to enforce that.

OneRMCalculatorViewController

The one rep max calculator is similar to the wilks calculator, it consists of a few input fields and when the user clicks calculate it called the CalculateRM() function which verifies all of the information is present and then calls the calculation service to calculate the one rep max and display it.

Services

CalculationService

The calculation service is the only service that doesn't do any database calls. It has three functions to perform simple calculations.

CalculateOneRepMax() takes in two parameters the weight and reps both as doubles and returns an Int, the calculated one rep max. The calculations are pretty straight forward. The next two functions are similar except one is for male and one is for female. calculateMensWilksNumberInPoints() and calculateWomensWilksNumberInPounds() both functions take the same parameters, maxbench, maxsquat, maxdeadlift and bodyweight, all as doubles then returns a double, the calculated wilks number.

DatabaseService

The database service is the big service in the entire program, it handles all of the database calls. There is a saveWorkout function with takes a workout type and saves it to the database. It also stores the today of the workout in another place on the database so that we can access the name of all of the collections, the function that does that is called addDateToWorkoutCollection and gets todays date.

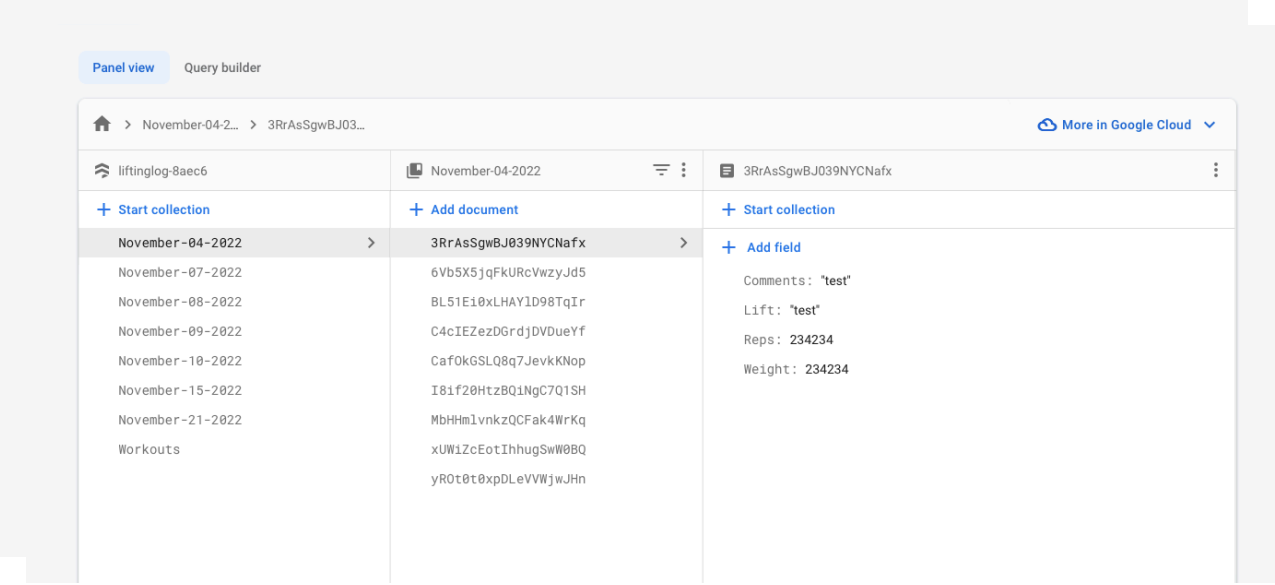
There are a few helper functions createWorkoutObject that transforms the raw data into a workout object that can be used anywhere and getCurrentMonthDayYear() that gets the get date in MMMM-DD-YYYY format and returns it as as string.

The DatabaseService also contains the arrayOfWorkouts array that the PastWorkoutsController uses to display all of the dates. When the user logs in the intalizeWorkoutsArray() is called and that is what calls into the database and gets all of the workout dates. This is essential so the app can load quickly when the user is navigating between workouts.

AuthenticationService

The last service is the AutenthicationService that contains two methods createUser() and logUserIn(), both take and email address and password as a string and have a completion handle that returns the AuthResponseStruct. If the user logs in or account is created successfully it will return true but if either fails it will return false and an error message will be displayed to the user through the completion handler.

Database



The Firestore Database is setup that “collections” contain the date of the workout. Each collection has a “document” and the documents contain each workout in a day and inside each “document” contains fields which are the works comments, lift, reps and weight. Firebase doesn’t offer a way for you to get all of the collections back so we put all of the dates in a collection then pull back all the documents from that collection which mirrors all the collections. Then when a user selects a collection we can retrieve all of the documents and fields to display the workouts to the users.