IvanCaldwell / **ios-firebase-chat**
forked from LambdaSchool/ios-firebase-chat

No description, website, or topics provided.

Edit

Manage topics

4 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with LambdaSchool:master.

SpencerCurtis Update grammar and clarify instructions in README.md

README.md Update grammar and clarify instructions in README.md

Clone with HTTPS Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/IvanCaldwell/ios-fire

Open in Desktop | Download ZIP

README.md

Firebase Chat

Introduction

The goal of this project is to create a working chat application that uses the Firebase SDK and MessageKit.

Instructions

Please fork and clone this repository. There is no starter project, so go ahead and create your own to use.

Part 1 - Learn about the Firebase SDK

As you will be using Firebase's SDK as a library in your app, you will gain the full functionality of Firebase as opposed to simply using it as a REST API. In order to come up with a good plan of how to structure your data, refer to Firebase's documentation. A few good places to start are:

- https://firebase.google.com/docs/database/ios/structure-data.
- https://firebase.google.com/docs/database/ios/read-and-write.

For explanations of different objects, refer to the iOS section of their reference guide here. Firebase has many different parts, though the only ones you will need to interact with are Firebase Core and Firebase Database. The documentation for the classes, enums, etc. of each section are on the left hand side of that page.

Part 2 - Adding the frameworks

1. Refer to this page. It will help you add the Firebase framework to your application. Pay attention to step 5 of the "Add the SDK" section, as you must add a `plist` to your project. This isn't something that you always have to do, but it is a requirement for Firebase. You will need to add the `pod 'Firebase/Core'` and `pod 'Firebase/Database'` pods to your Podfile.

2. Add MessageKit to your project using Carthage, CocoaPods, or as a submodule.

Part 3 - Storyboard Setup

1. In the "Main.storyboard", add a table view controller scene that will allow you to create new "chat rooms" and view one. Embed it in a navigation controller and set the navigation controller as the initial view controller. Create a Cocoa Touch subclass for this view controller and set the scene's class to it.
2. Add a view controller scene that will act as the `MessageViewController` subclass to view the messages using the UI

that the `MessageViewController` will provide (input bar, chat bubbles, etc.)
3. Create a segue from the table view controller scene to the detail view controller.

Part 4 - Model Setup

1. Create a model object to represent a message, and one for a chat room. Give them the appropriate values as you see fit.
    - Remember that the data in Firebase is stored as JSON, but note that the SDK's methods to save and load the data (here, for example) to/from Firebase use dictionaries. Because of this, you will need to write code to convert your model objects to and from a dictionary in order for you to use these methods.
2. Create a model controller. Again referring to the Firebase guides and documentation, add the following functionality:
    - Create a chat room in Firebase
    - Fetch chat rooms from Firebase
    - Create a message in a chat room in Firebase
    - Fetch messages in a chat room from Firebase

Feel free to create as many helper methods as you need to avoid repeating yourself.

Part 5 - View Controller Implementation

Finally, wire up your view controllers.

1. In your table view controller subclass, fetch the chat rooms, then implement the necessary `UITableViewDataSource` methods to list them.
2. In the detail view controller, implement the `MessageDataSource` methods to get your messages displaying on the `messagesCollectionView`.
3. Implement the `MessageInputBarDelegate didPressSendButtonWith` method to allow you to call your create message function.
4. Implement some (or all) of the `MessageDisplayDelegate` and `MessagesLayoutDelegate` methods to customize the look of the messages.

Go Further

- Add the MessageKit framework using one (or even better, both) of the different methods than the one you used.
- Look at (by downloading the entire project from the repo here or navigating to the file in GitHub) the "ConversationViewController.swift" file in the example project from the repo, and look at the Keyboard Style mark. The functions in that section customize the look of the input bar to mimic other apps. Try it out yourself and try to make your own style.