Bachelor Degree Project

Automatic synchronization and data merging between mobile and server

- Solution for easy setup of synchronized CRUD functionality between server and application

Author: Mikael Melander
Semester: ST 2018

Abstract

Keywords: iOS, crud, Linux, offline synchronization

Preface

Contents

1 Introduction

1.1 Background

The world of mobile applications has exploded in the last couple of years. Companies increasingly adopt the functionality of mobile devices to streamline the daily workflow, what used to be done by pen and paper or only by sitting down at a computer is now possible with mobile applications.

Because a mobile phone can be taken anywhere, your work should be able to follow. But most applications demand an external database to store data across users, this intern relies upon a mobile data connection or WIFI by the device itself.
This can become a problem if you find yourself in places where your cellular reception is limited. If this happens then you won't be able to work.
The solution to this is to be able to add, see and use the data that already exists within the application regardless of your cellular reception.

1.2 Related work

Research around similar solutions have been conducted and there is some big and famous companies that have created some type solutions to the problem.

Firebase is googles database service that allows you to create an mobile/web application connected to the database within a couple of steps. They also have a set of tools to verify data, create security, analyse usage, send push notifications and many more[1].

Microsoft Azure also supports the ability for offline client sync, but aswell as Firebase it relies on you using their cloud services[2].

1.3 Problem formulation

The goal is to find a suitable solution that can handle objects structures on both the iOS locally and the backend database the same way.
Creating the offline functionality by saving cached data on the device and the synchronising it to the backend when the connections allows it.

Because the solution should be angled towards multi tenant usage, the system itself needs to be able to determine the correct way to merge different versions of the same object trying to be saved to the database[3].

In order for the system to be able to track this, the data model needs to be implemented in such a way as to allow this. For example, adding version

1.4 Motivation

Today it already exists different solutions and platforms for offline data synchronisations, for example firebase. Firebase is owned by google and using it locks you down to their backend and their systems and you have little to no control over your data.
Many companies today that want a solution for mobile applications associated to their work flow would not allow that the data is stored anywhere other that in their own control.
So the creation of a ready to go backend solution that runs on a Linux environment and can be easily connected to and iOS device would create an opportunity for these companies to be in control of the complete backend.

1.5 Objectives

O1	Research and determine server framework
O2	Create server database and iOS database model
O3	Implement connection between iOS and server
O4	Implement local storage iOS
O5	Implement methods for querying data given specific arguments that handles local/server storage
O6	Implement functionality that automatically tracks what data has been sent to the server and what data only exists locally
O7	Determine and perform experiments for merge rules for data

Because of the need to be able to implement querying and rules for merging of data, the database needs to be run behind a server.
The intent is to research and determine a suitable and reliable server framework as well as a database model that will suit both the local storage and the server database.

For the offline data to work the device need a local storage, the storage should work as a middle layer to the server database query. When new server data is returned it should be saved locally, then returned. The implementation needs to keep track of what data is already stored locally and what data is new and needs to be saved.

The server itself needs to be able to handle the requests and execute merge rules.

1.6 Scope/Limitation

Within the scope of this project, the solution should allow for a relatively easy setup that gives you a reusable solution within iOS/Xcode that would allow you to query data and return results, either from locally saved data or from the online database depending on your internet connection.

You should be able to save new data and edit previously added data that when reconnected to a internet connection should be synced with the database.

The solution that is created and the dependencies used for it has to be free.
Because Xcode and iOS changes over time the solution should be set to work and support at least the latest 3 versions of iOS (9-11).

The project will not support real time database.

1.7 Target group

This project can be of interest to companies, organization or persons wanting to host their own solution that integrates with iOS that is cost efficient.
The solution could as well be used as a starting point to keep building upon, but already has the important support for offline synchronization.

1.8 Outline

The next chapter will present the **methods** that is used to execute the different objectives that is presented above.

The **implementation** chapter will be a more detailed explanation of how the project will be implemented and how the solution itself works, how the merge rules will be executed and how the automatic syncing is handled.

The **result** chapter will cover what came of the project, what the resulting structure of the solution became.

The **analysis** will cover an overall analyze of the concluded results.

The **discussion** will deeper discuss the analysis and results.

Chapter seven will include **conclusions** that are based on the results aswell as present future work and recommendations.

2 Method

2.3 Reliability and Validity

2.4 Ethical Considerations

3 Implementation

4 Results

5 Analysis

6 Discussion

7 Conclusion

7.1 Future work

References

[1] https://firebase.google.com/docs/database/ios/offline-capabilities
2018-02-29

[2] https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-ios-get-started-offline-data
2018-02-29

[3] http://opensourceforu.com/2017/01/mobile-app-without-internet/
2018-03-01