Einkaufslisten Applikation

Inhalt

Ziele

Das Ziel dieser Übung ist eine Anbindung einer mobilen Applikation an ein Webservices zur gleichzeitigen Bearbeitung von bereitgestellten Informationen. ### Voraussetzungen * Grundlagen einer höheren Programmiersprache * Grundlagen über Synchronisation und Replikation * Grundlegendes Verständnis über Entwicklungs- und Simulationsumgebungen * Verständnis von Webservices ### Aufgabenstellung Es ist eine mobile Anwendung zu implementieren, die einen Informationsabgleich von verschiedenen Clients ermöglicht. Dabei ist ein synchronisierter Zugriff zu realisieren. Als Beispielimplementierung soll eine "Einkaufsliste" gewählt werden. Dabei soll sichergestellt werden, dass die Information auch im Offline-Modus abgerufen werden kann, zum Beispiel durch eine lokale Client-Datenbank.

Es ist freigestellt, welche mobile Implementierungsumgebung dafür gewählt wird. Wichtig ist dabei die Dokumentation der Vorgehensweise und des Designs. Es empfiehlt sich, die im Unterricht vorgestellten Methoden sowie Argumente (pros/cons) für das Design zu dokumentieren. ### Bewertung * Gruppengrösse: 1 Person * Anforderungen "Grundkompetenz überwiegend erfüllt" * Beschreibung des Synchronisationsansatzes und Design der gewählten Architektur (Interaktion, Datenhaltung) * Recherche möglicher Systeme bzw. Frameworks zur Synchronisation und Replikation der Daten * Dokumentation der gewählten Schnittstellen * Anforderungen "Grundkompetenz zur Gänze erfüllt" * Implementierung der gewählten Umgebung auf lokalem System * Überprüfung der funktionalen Anforderungen zur Erstellung und Synchronisation der Datensätze * Anforderungen "Erweiterte-Kompetenz überwiegend erfüllt" * CRUD Implementierung * Implementierung eines Replikationsansatzes zur Konsistenzwahrung * Anforderungen "Erweiterte-Kompetenz zur Gänze erfüllt" * Offline-Verfügbarkeit * System global erreichbar ## Umsetzung Anfangs gab es die Überlegung die Applikation

mit einem eigenen Server, welcher Daten von Clients in einer Mlab-Datenbank speichert, umzusetzen. Nach einer Empfehlung von Schulkollegen und einer daraus resultierenden Recherche wurde die Applikation mittels React Native + Firebase umgesetzt. Es wurde eine Tutorial benutzt, welches eine ToDo Liste erstellt. Das Tutorial wurde auf die Einkaufliste umgeschrieben und gestylet. ## Verbindung mit Firebase `react-native-firebase` bietet dem Entwickler ein vorgegebenes Projekt mit einer Installationsanleitung. Hierbei werden alle Schritte zum Aufsetzen der Applikation beschrieben. ## Datenformat Das Datenformat eines Einkaufslisten-Items schaut folgendermaßen aus:

```
item = {
    itemname    : string,
    bought      : boolean,
}
```

`itemname` beinhaltet den Namen des Items, welches in der Einkaufsliste angezeigt wird. `bought```` beinhaltet einen Booleanwert, welcher angegibt, ob das Item schon gekauft wurde oder nicht. ## Synchronisation der Datensätze Folgender Codeabschnitt beschreibt die Synchronisation der Datensätze:JavaScript constructor() { // Referenz zur Kollektion this.ref = firebase.firestore().collection('items'); // Attribut, welche für die Synchronisation zuständig ist this.unsubscribe = null; ... }

// Diese Methode wird aufgerufen, wenn der Komponent gerendet wird // Mit dieser Zeile wird bei dem Event Snapshot die Methode this.onCollectionUpdate ausgeführt componentDidMount() { this.unsubscribe = this.ref.onSnapshot(this.onCollectionUpdate) }

// Diese Methode beendet das Listen, falls ein Snapshot passiert. componentWillUnmount() { this.unsubscribe(); }

// Hierbei handelt es sich um eine Funktion, welche wie ein Attribut definiert wird // Als Parameter wird querySnapshot weitergegeben. Es beinhaltet den Snapshot. onCollectionUpdate = (querySnapshot) => { // Neue Liste wird initialisiert const newList = []; // Der Snapshot wird durchiteriert und bei jedem Datensatz wird eine Funktion aufgerufen // welche sich die Daten holt und in newList reingibt querySnapshot.forEach((doc) => { // Holt sich die Daten des Dokumentes const { itemname, bought } = doc.data(); // Gibt alle Daten in die newList newList.push({ key: doc.id, // ID des Dokumentes doc, // Datenreferenz itemname, // Name des Items bought, // Gekauft-Attribut des Items }); }); // Setzt den State der jetzigen Listemit der neuen this.setState({ list : newList }); } `## CRUD-Funktionialität ### Hinzufügen von Datensätzen Das Hinzufügen von Datensätzen wird in der App-Klasse durchgeführt.`JavaScript construcot() { // Ist die Referenz auf die Kollektion mit den Dokumenten der Einkaufsliste this.ref = firebase.firestore().collection('items'); }

addItem() { // Mit this.ref.add wird ein Datensatz hinzugefügt this.ref.add({ itemname: this.state.textInput, bought: false, }); // Hier wird das Textfeld

wieder zurückgesetzt this.setState({ textInput: “”, }); }

render() { // Hier wird bei einem Klick (onPress) die addItem()-Methode aufgerufen return (... <Button style={styles.button} title={‘Add shop item’} disabled={!this.state.textInput.length} onPress={() => this.addItem()} />); } ### Veränderung von Datensätzen Das Verändern von Datensätzen ist in der ShopItem-Klasse durchgeführt worden.JavaScript ___toggleItem() { // doc ist ein Property von der Klasse ShopItem // Mit dem doc kann auf das ref zugegriffen werden // Damit kann man update machen und Datensätze verändern this.props.doc.ref.update({ bought: !this.props.bought, }); } ### Löschen von Datensätzen Das Löschen von Datensätzen ist auch in der ShopItem-Klasse durchgeführt worden.JavaScript ___delItem() { // Es wird wieder mit dem doc-Property auf das ref zugegriffen // Dann wird nur die delete()-Methode aufgerufen this.props.doc.ref.delete(); } ### Zuweisung von Veränderung und Löschen von Datensätzen Eine TouchableOpacity wurde verwendet um Datensätze zu verändern beziehungsweise zu löschen. Hierbei wird beim kurzen Tippen die Veränderung durchgeführt. Bei einem langen Draufdrücken wird der Datensatz gelöscht.JavaScript render() { ... return (<TouchableOpacity onPress={() => this.___toggleItem()} onLongPress={() => this.___delItem()} style={styles.listItem && styling} > ...); } ## Offline-Verfügbarkeit und globale Erreichbarkeit Firebase bietet eine Offline-Verfügbarkeit. Die Daten werden lokal gespeichert und dann bei Internetzugriff mit dem Server aktualisiert. Firebase bietet zudem auch eine globale Erreichbarkeit. ## Probleme mit dem Layout Das Layout der Items in der Liste konnte nicht wie gewollt gestaltet werden, weshalb die Items in der Liste nicht gut aussehen. Dies wirkt sich jedoch nicht auf die Funktionalität der Applikation auf. ## Installation und Ausführung ### Voraussetzung für die Installation und Ausführung * [React Native](https://facebook.github.io/react-native/) * [Node Package Manager](https://www.npmjs.com) * [Android Studio](https://developer.android.c für einen Emulator ### Ausführung 1. git clone https://github.com/fscopulovic-tgm/Einkaufsliste_App2.cd Einkaufsliste_App3.npm install4. Emulator starten 5.react-native run-android### Ausführung auf einem Android Handy 1. Das Handy muss sich im [developer mode](https://www.greenbot.com/article/2457986/ar befinden 2. Handy mit dem Computer via USB verbinden 3.react-native run-android### Probleme mit iOS Die Implementation auf iOS-Smartphones konnte nicht umgesetzt werden. Es gab Probleme bei der Ausführung mitreact-native run-ios“‘. Die Applikation konnte nicht gebuildet werden.