

Exerciseopgave11-12 – Login & Server kald

Da det er svært at forklare større kodelapper med tekst, forventes der, at I forsøger at kigge på slidesne samtidigt med at I laver opgaverne.

Opsætning af database

1. Åben linket <https://console.firebase.google.com/u/0/> og naviger til jeres Firebase projekt.
2. I venstre menu klikker I Develop > Database.
3. Rul ned og tryk på "Create database"
4. I pop-up vinduet trykker I "Start in **test mode**" og derefter "Enable"
5. Opbyg jeres database med en liste (tabel) der kaldes *albums* og *artists*. Brug slide 7 til at se eksemplet.
6. Sørg for at få en JSON struktur som set på slide 7:
 - a. Vi har *albums* som eksempelvis indeholder *fearless* og *speakNow* (data fundet fra vores Taylor Swift album API: http://rallycoding.herokuapp.com/api/music_albums).
 - b. Et album indeholder artist, image og title.
 - c. Sørg for at artist (her Taylor Swift) skrives med camelCase (taylorSwift). Dette er blot for at give et eksempel senere, hvordan I bruger én værdi fra et sted i databasen til at søge et andet sted i databasen. Teknisk set kan I bare have artisten direkte angivet i albummet, og så spare den anden liste *artists*. Dog kan det være, at man skal bruge flere informationer om artisten, og derfor vil en liste med *artists* måske være brugbar.
7. Sørg for at jeres read og write rules er sat til *true*. Tryk på "Rules" ved siden af "Data".

Opsætning af projektet fra sidst

8. Kopier SignUpForm og kald kopien *LoginForm*.

9. Åben App.js

- a. Importer nu i stedet for SignUpForm, den nye LoginForm og ændre de steder der står SignUpForm med LoginForm. (Det er blot importen og anvendelsen af componenten i render()).
- b. Tilføj en boolean state variabel til App.js kaldet loggedIn og starte med at sætte den til null.
- c. Inde i componentWillMount() funktionen, under firebase.initializeApp, implementerer du metoden *firebase.auth().onAuthStateChanged(user)* funktionen.
 - i. Denne funktion er en listener, der holder øje om Firebase's authentication state ændre sig (dvs. at denne funktion bliver aktiveret idet der prøves at logges ind).
 - ii. Implementer en if-statement ligesom i slide **x**, som tjekker om den user, der forsøges at logges ind som, er authorized (og dermed findes) eller ej. Findes user, sæt state variabelen loggedIn til true, ellers false.
- d. Inde i render() funktionen
 - i. Opretter du en switch der tjekker på this.state.loggedIn.
 1. Hvis case false: returner et View der indeholder din LoginForm component
 2. Hvis case true: returner et View der indeholder et Text component, der skriver "Logged in – this is Home" (dette gøres blot for at du kan køre appen og teste, mens du ikke har udviklet de skærme man kommer ind på, når man logger ind. Når vi gør dét, skal vi herind og ændre Text-componenten).
 3. Hvis case default: returner en ActivityIndicator med size='large'
 4. Du kan benytte dig af lidt styling så din LoginForm vises i midten. Sæt dine View components style til containertagget:

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'stretch',
    justifyContent: 'center',
  },
});
```

a.

LoginForm.js

10. Ændr class definitionen til LoginForm i stedet for SignUpForm.
11. Importer SignUpForm herind. Husk at LoginForm og SignUpForm ligger i samme mappe, begge inde i 'components'-mappen, og derfor skal du blot importere fra './SignUpForm'
12. Ændr navnet på onPress funktionen til signIn
13. I stedet for at bruge funktionen *firebase.auth().createUserWithEmailAndPassword* som vi gør i SignUpForm, skal du ændre funktionen til *firebase.auth().signInWithEmailAndPassword*. Logikken inde i funktionen forbliver det samme
14. I LoginForm's state, tilføjer du boolean variabelen "hasLogin" og sætter den til true
15. I render() funktionen skal du:
 - a. Wrappe en switch rundt om den logik der allerede står i render(). Switchen skal tjekke på this.state.hasLogin, og den tilstedeværende logik skal returneres ved case true.
 - b. Hvis case false: returner et View, der indeholder din SignUpForm-component, men også en Button nedenunder med titlen "Go back" og en onPress funktion, der sætter state hasLogin til true. Denne knap skal bruges til at navigere tilbage til LoginForm, hvis man efter at have trykket "Sign up" alligevel vil tilbage igen.
 - c. Hvis vi går tilbage til case true: Opret et Text component der skriver "Sign in" over det første TextInput.
 - i. I bunden af case true: Efter this.renderButton() lav en Button med titlen "Sign up" og en onPress funktion, der sætter state hasLogin til false. Dette vil få appen til at vise din SignUpForm.

- d. Inde i `renderButton()` funktionen, skal du ændre `Button`-componenten til at have en `title` "Log in" og `onPress` funktionen skal referere til din `signIn` funktion vi lavede i opgave 12.
16. Ændre navnene til funktionerne `onSignUpSuccess` og `onSignUpFail` til `onLoginSuccess` og `onLoginFailed`. Husk at ændre både navnene og kaldet af funktionerne i `signIn()` funktionen.
17. Kør appen og se om du både kan navigere fra `LoginForm` til `SignUpForm` og tilbage. Tjek også og se om du kan logge ind med en eksisterende bruger (Bliver teksten vi skrev i opgave 9.d.2 ("Logged in – this is Home") vist?). Gør den dét, så har du startet en session, hvilket vil sige, at hvis du lukker og åbner appen igen, vil den automatisk være logget ind igen (ligesom du ser på rigtige apps, hvor du ikke er logget ud). Dette er godt! Dog kan vi ikke teste `SignUpForm`, da vi ikke har en logud knap endnu. Derfor skal du gå ind på Firebase under Authentication og slette den bruger du har logget ind med og køre appen igen. Forsøg nu at oprette en bruger. **Note:** Når du opretter en bruger, læser Firebase det som, at du også automatisk bliver authenticated. Derfor vil du automatisk blive logget ind her også. Dette forventes, og hvis det sker, så er alt vel.

Integrering af screens *efter* man er logget ind.

Overordnet set kommer denne del af øvelserne til at handle om, hvordan I bygger hele jeres tab-navigation app med listen af Albums fra tidligere øvelse ind efter man er logget ind. `FlatList` skal hente fra jeres database i stedet for tidligere API link + I skal lave en ny knap i tab-baren, hvor I får adgang til at oprette et nyt album.

Vi har givet jer en pakke for, at I ikke selv skal bruge tid på kopiere de forskellige screens fra forskellige projekter og sætte det op. Pakken indeholder blot de Screens I har lavet i tidligere øvelser, hvor vi har vores `AlbumsScreen` (kaldet `HomeScreen` her), `DetailsScreen` og `SettingsScreen`. Vi har også tilføjet endnu et screen kaldet `AddNewScreen`, som er den skærm vi skal bruge til at oprette nye Albums. `Home.js` er det component vi nu vil bruge som skal agere som "App.js", blot når man er logget ind. Åben den og I vil se, at den indeholder præcist det samme som jeres `App.js` fra tidligere øvelser.

18. Opret en 'pages' mappe inde i 'components' mappen. Smid alle filer fra pakken derind.

19. Gå ind i App.js og importer Home.js. Slet Text-componenten der blev oprettet i opgave 9.d.2 og lav i stedet en Home-component <Home/>. På dette tidspunkt burde din app nu navigere ind og vise listen med albums fra tidligere øvelser, så snart man logger ind.
20. Husk at installere følgende libraries:
- react-navigation
 - react-native-elements

HomeScreen – hent data fra Firebase

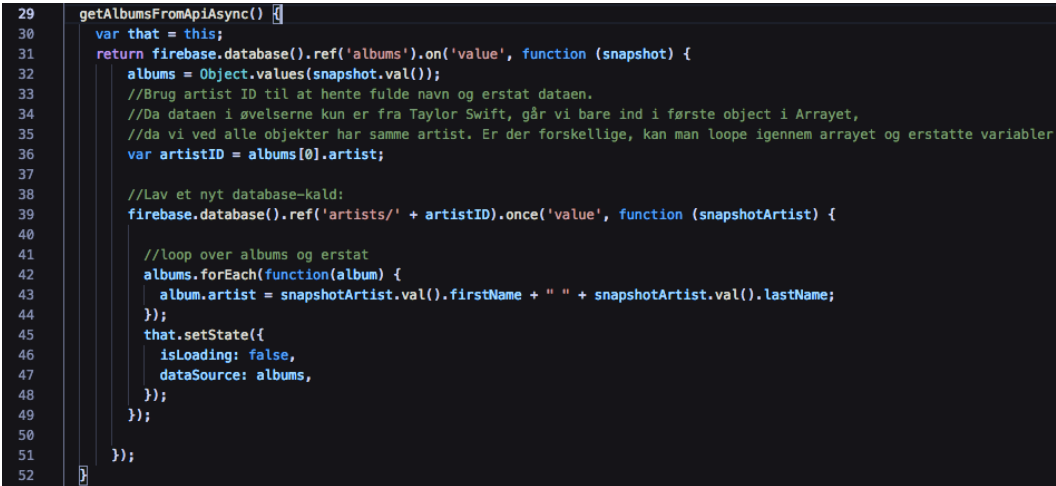
21. Importer firebase
22. Inde i getAlbumsFromAsync() funktionen skal I slette den eksisterende logik og benytte jer af firebase's .on() function i stedet for:

```
getAlbumsFromApiAsync() {  
  var that = this;  
  return firebase.database().ref('albums').on('value', function (snapshot) {  
    albums = Object.values(snapshot.val());  
    that.setState({  
      isLoading: false,  
      dataSource: albums,  
    });  
  });  
}
```

-
- .on() funktionen er en listener, der bliver kaldt *hver gang* der sker ændringer i databasen. Dvs. hvis nogen opretter et nyt album, mens du kigger på albumlisten, så vil du med det samme se det nye album. "snapshot" variabelen er for at vise et øjebliksbillede af databasen på det tidspunkt det bliver kaldt. Dvs. at der altså er et nyt snapshot så snart en ændring forekommer i databasen. .once() funktionen henter et snapshot én gang, og bliver ikke opdateret før funktionen den står i kaldes igen.
- Vi skriver "var that = this;" for at kunne få fat i "this" (altså HomeScreen componenten) gennem en anden variabel, da når vi er inde i en function (function(snapshot) {}) og kalder "this", så får I fat i kalderen af denne function, hvilket altså er firebase databasen. Det er svært at forklare. Spørg gerne om en uddybende forklaring.

23. Din data i databasen burde nu blive vist, hvis du kører appen.¹

24. Benyttelsen af `.once()` funktionen kan implementeres således for at erstatte `taylorSwift` med Taylor Swift fra `artists` listen i databasen:

a. 

```
29 getAlbumsFromApiAsync() {
30   var that = this;
31   return firebase.database().ref('albums').on('value', function (snapshot) {
32     albums = Object.values(snapshot.val());
33     //Brug artist ID til at hente fulde navn og erstat dataen.
34     //Da dataen i øvelserne kun er fra Taylor Swift, går vi bare ind i første object i Arrayet,
35     //da vi ved alle objekter har samme artist. Er der forskellige, kan man loope igennem arrayet og erstatte variabler
36     var artistID = albums[0].artist;
37
38     //Lav et nyt database-kald:
39     firebase.database().ref('artists/' + artistID).once('value', function (snapshotArtist) {
40
41       //Loop over albums og erstat
42       albums.forEach(function(album) {
43         album.artist = snapshotArtist.val().firstName + " " + snapshotArtist.val().lastName;
44       });
45       that.setState({
46         isLoading: false,
47         dataSource: albums,
48       });
49     });
50   });
51 }
52
```

AddNewScreen – skriv data til Firebase

25. Åben `AddNewScreen`. Her vil du se en state, hvor vi kan definere artist, title og image. Vi hardcoder image-linket, da vi ikke gider at bruge ressourcer på at skrive et link ind ved oprettelse af et nyt album. Derfor vil alle albums der oprettes igennem her få det samme album image.

26. Inde i `render()` funktionen skal du oprette to `TextInput`, som tager imod artist og title og sætter dem lig statens artist og title. Kig eksempelvis i din `LoginForm.js` for at få inspiration til hvordan dette kan gøres.

- Opret et tredje `TextInput`, hvor du sætter den til at være `editable={false}` og dets value til `this.state.image`, blot for at vise linket i UI-delen, men hvor den ikke kan ændres.
- Opret en `Button` med en title og en `onPress` funktion der refererer til `writeAlbum()`.

27. Inde i `writeAlbum()` skal I benytte jer af Firebase's `.push()` funktion for at skubbe et objekt op i databasen.

¹ Vi oplevede en konstant gul warning på Android, som vi og andre på nettet ikke har kunnet finde en løsning på. Et hotfix folk fandt frem til var at lave `console.ignoredYellowBox = ['Setting a timer'];` Får du advarslen, kan du sætte kodelinjen ind i constructoren i `HomeScreen`.

```

23 writeAlbum(){
24     const artist = this.state.artist;
25     const title = this.state.title;
26     const image = this.state.image;
27
28     firebase.database().ref('albums/').push({
29         artist,
30         title,
31         image
32     }).then((data)=>{
33         alert("Album created successfully");
34     }).catch((error)=>{
35         //error callback
36         console.log('error ', error)
37     })
38 }

```

a.

28. Test og se om det virker. Tjek databasen og se om dit album er blevet oprettet. Bemærk at den kommer til at få et autogenerated ID.

SettingsScreen – signOut

29. Åben SettingsScreen og importer firebase.

30. Opret en tredje Button i render() funktionen, hvor onPress kalder funktionen
 firebase.auth().signOut()

31. Appen vil automatisk logge dig ud og sende dig tilbage til login-skærmen.