Övningsuppgift 12 - Pokéfire

Detta är en större övningsuppgift. I denna skall du bygga en Pokemon-applikation där du kan spara ett register över alla din Pokemons.

Du kommer att

- Använda Router för att navigera mellan olika komponenter som ansvarar för olika delar.
- Använda Firebase för att lagra din data online utan att behöva implementera ett eget backend.

1. Skapa en firestore database

1. Skapa ett Google-konto om du inte redan har ett. (Gmail).
2. Gå till console.firebase.google.com för att skapa ett nytt firebase projekt. Namnet spelar ingen roll - men t.ex. pokefire kan fungera.
3. Skapa en ny "firestore"-databas. Denna skall vara i test-mode nu (vi kommer att ändra detta i nästa övning då vi implementerar autentisering).
4. Skapa en collection som heter "pokemons". Skapa några olika pokemons (dokument). Dem behöver några olika fält - namn (string), hp (number), price (number), forSale (boolean). När du skapar dina pokemons kan du låta firestore skapa dina dokument id (auto-id).
5. Testa query builder för att lista olika del-mängder av dina pokemons (t.ex. dem pokemons som har hp över 100).
6. Skapa en ny "Web" applikation. Du kan kalla den vad som helst. Du kommer se en konfigurations-kod - denna kommer du att behöva senare. Du behöver inte spara den någon annanstans eftersom du kan alltid komma tillbaka till din firebase console och hitta den.

2. Sätt upp din utvecklingsmiljö

1. Skapa ett nytt angular projekt (ng new Pokefire). Du vill använda routing i denna applikationen.
2. Uppdatera din tsconfig.json — sätt strict mode till "false".
3. Installera firebase "npm install firebase"
4. Installera firebase-tools globalt (om du inte redan har det)
    1. Windows: npm install -g firebase-tools
    2. Mac: sudo npm install -g firebase-tools
5. Logga in till firebase "firebase login" - här skall du använda samma Google-konto som du använde i deluppgift 1.
6. Installera firebase "ng add @angular/fire" - du skall välja att installera firestore i listan över val. (Notera: Eftersom det finns en del problem med angular/firebase verktygen kommer du att få ett felmeddelande om att den inte kunde skapa din environments - vi gör detta strax manuellt - 3.1).
7. Installera bootstrap "ng add @ng-bootstrap/ng-bootstrap"

8. Städa upp i app.component.ts — gör om den till inline om du föredrar det. Tabort all boilerplate-kod från html-mallarna.
9. Testa att ditt projekt startar (ng serve —open)
10. Skapa två komponenter
    1. pokemon-list
    2. pokemon-edit
    3. Du kan skapa dem med ”ng g c -t -s” om du vill ha dem som inline komponenter. Du kan radera alla spec-filer eftersom vi inte använder dem i denna kursen.
11. Skapa en ny datatyp i en egen fil - du kan kalla datatypen för ”Pokemon”. Den skall ha samma attribut som dina pokemons du skapade i din firestore database (se 1.4)

3. Förbered ditt angular-projekt för firebase

1. Skapa mappen ”environments” i src/ med en fil som heter environments.ts. Lägg följande kod i filen:

```
export const environment = {
production: false,
    firebaseConfig: {
        // firebase config goes here
    }
}
```

2. Gå till console.firebase.google.com och kopiera firebase configurationen som du skapade i 1.6 och klistra in den i din environment.ts.
3. Gå till din app modul och importera
    1. AngularFireModule
    2. AngularFirestoreModule
    3. environment (från filen som du skapade i 3.1).
    4. Lägg till AngularFireModule.initilizeApp(environment.firebaseConfig) i din imports-sektion.
    5. Lägg till AngularForestoreModule till din imports-sektion.
4. Ditt projekt är nu redo att börja använda firebase

4. Grundläggande routing

1. Placera din router-outlet i app-components - HTML/markup.
2. Skapa en ””-route, den skall ta dig till din pokemon-list komponent.
3. Skapa en "edit”-route, den skall ta dig till den pokemon-edit komponent.
4. Du behöver inte skapa någon ”default”-route i denna övningen.

5. Testa att din routing-komponent fungerar

5. Skapa en Pokemon-lista (R)

(Läs igenom hela deluppgiften innan du börjar)

1. Skapa HTML/Markup för en tabell som skall lista alla din pokemon.
2. Skapa HTML/Markup fär en knapp som skall ha etiketten: ”Add Pokemon”. Den skall köra en metod som heter addPokemon().
3. Din komponent skall använda OnInit. Importera och implementera detta interface (ngOnInit() kan vara tom för tillfället).
4. Skapa ett nytt klass-attribut som du kan kalla ”pokemonList” den ska vara av typen AngularFirestoreCollection - och den ska vara av din typ ”<Pokemon>” från 2.11.
5. Skapa ett attribut som du kallar för pokemons av typen any.
6. I din ngOnInit() - tilldela din pokemonList en AngularFirestoreCollection som är kopplad till din Pokemon-collection(som du skapade i 1.4). För att göra detta behöver du injecta AngularFirestore i din constructor. Du kan använda detta objektet för att hämta objektet.
7. Du skall sedan använda snapshotChanges() från din pokemonList för att prenumerera på alla förändringar på denna samlingen. Du kan tilldela resultatet av detta till pokemons (5.5):

```
this.pokemons = this.pokemonList.snapshotChanges().pipe(
    map(actions => {
        return actions.map( a => {
            const data = a.payload.doc.data() as Pokemon;
            const id = a.payload.doc.id;
            return { id, data };
        })
    })
);
```

8. Uppdatera din HTML/Markup för att loopa över pokemons med en ngFor - den skall vara ”async”
9. Implementera kod i din ”addPokemon()” metod som använder ett ”Router” objekt för att navigera till din ”edit” komponent. Du kommer att behöva skapa ett router-objekt med hjälp av dependency injection.
10. Du ska nu ha en lista med alla dina pokemons synlig på din webbsida. Din knapp skall också ta dig till din nya komponent.

6. Skapa en ny Pokemon (C)

1. Dags att gå till din edit-pokemon-komponent.
2. Skapa ett klass-attribut för en Pokemon - du kan kalla det för newPokemon.

3. Skapa Markup/HTML för ett formulär som har alla input-element som behövs för att skapa en ny pokemon. Implementera FormsModule så att du kan läsa och skriva data i formuläret. Din ”newPokemon” skall vara det attribut i klassen som håller datan för formen. Se tidigare övningar för hur man implementerar FormsModule om du är osäker.
4. Skapa en knapp för submit och en knapp för cancel. Submit-knappen skall bara vara klickbar bara om formen är korrekt ifylld (Pokemon har fått ett namn).
5. I din submit()-metod: Använd AngularFirestore och en ”collection” för ”Pokemon” - lägg till ditt nya pokemonkort (add()). Navigera sedan tillbaka till list-komponenten. Du behöver skapa både AngularFirestore och Router medhjälp av Dependency injection för att åstadkomma detta.
6. Skapa en metod för din cancel-knapp. Denna skall bara navigera tillbaka till din list-komponent.
7. Du ska nu kunna lägga till ett nytt kort och komma tillbaka till din listvy. Du kan i firebase konsollen kontrollera att ditt kort skapades korrekt.

7. Radera en befintlig Pokemon (D)

1. Vi kommer att radera pokemons från listvyn. Gå tillbaka till din list-komponent och gör förändringarna för denna underuppgift i den komponenten.
2. Skapa en länk eller en knapp till höger om varje Pokemon i din lista. Denna skall köra en klassmetod (deletePokemon()). Du behöver skicka med dokument id för Pokemon-kortet till denna metoden.
3. I din deletePokemon-metod - fråga användaren om hen verkligen vill radera korten och om så radera kortet från din firestore databas. Du kan göra detta genom att använda ditt firestore objekt som du tidigare injectade i constructorn. Denna har en doc() metod som returnerar ett FirestoreDocument - detta i sin tur har en delete()-metod.
4. Du skall nu kunna radera pokemon-kort från din databas.

8. Uppdatera en befintlig Pokemon (R, U)

1. I din list-komponent; skapa en länk runt namnet för varje pokemon (i HTML/markup mallen). Om länken blir nedtryckt skall du navigera till din edit-komponent och du skall skicka med dokumentets id som en parameter.
2. Skapa en ny route i (app-routing.module.ts) - detta skall vara ytterligare en ”edit”-route som dessutom tar :id som parameter. Den skall visa edit-komponeten.
3. Vi avslutar denna övningen i edit-komponenten.
4. Importera och Implementera OnInit-interface:et.
5. Lägg till ett nytt klass-attribut som du kallar id. I din ngOnInit() ska du prenumerera på id-parametern. Stoppa in värde för denna i ditt nya id-attribut.
6. I din ngOnInit(); kontrollera om du har fått ett id - har du det skall du läsa in dess dokument från firestore databasen och spara det till ett nytt klass-attribut. Du kan sedan använda valueChange() som ger dig en observable och prenumerera på förändringar (subscribe()). Koppla förändringar till Dokumentet till newPokemon-attributet i klassen.

7. Nu skall du kunna få data för din Pokemon i din edit-pokemon form.
8. Det sista du behöver göra är att uppdatera submit()-metoden. Du behöver kontrollera om id-attributet är tomt - är det sant ska du köra add()-metoden som du tidigare implementerade, annars ska du med hjälp av firestore hämta en referens till ditt dokuent (doc()) och sedan uppdatera det med data från formuläret ((update()).
9. Du skall nu kunna uppdatera dina pokemons med hjälp av formuläret.