UNIVERZITET U SARAJEVU
PRIRODNO-MATEMATIČKI FAKULTET
INFORMACIONE TEHNOLOGIJE

MEMORY GAME

(PROJEKAT IZ REACT JS-A)

PREDMET: Web programiranje II
PROFESOR: prof. dr. Adis Alihodžić
STUDENT: Nejra Pribinja

Sarajevo, 2022

Sadržaj

O projektu

React

React je JavaScript biblioteka kreirana sa ciljem izgradnje korisničkih interfejsa.

React je postao jedan od najpopularnijih front-end framework-a za izradu web aplikacija.

U većini slučajeva, glavna uloga framework-a jeste prenošenje podataka DOM-u (ili modelu objekta dokumenta), i kao takve, aplikacije koje se zasnivaju na React-u često zahtijevaju dodatne biblioteke za održavanje stanja i rutiranje. Zbog toga često se u procesu developmenta uključuju i Redux (za održavanje stanja) i React Router (za ruting).

Osnove i primjena

React je nadaleko poznata i popularna biblioteka otvorenog koda za JavaScript izgrađena s namjerom da se kreiraju korisnički interfejsi (UI) za jednostranične aplikacije. Kao takav, React upravlja komandnom linijom za web aplikacije (i za mobilne uređaje, uz pomoć React-Native).

Glavne karakteristike

Zahvaljujući svojim karakteristikama, React omogućava programerima da razviju velike web aplikacije koje omogućavaju promjenu podataka bez potrebe za ponovnim osvježavanjem same stranice.

1. JSX (JavaScript XML)
2. Virtual DOM
3. React Native
4. Jednosmjerni protok podataka

Firebase

Firbase je razvojna platforma za mobilne i web aplikacije. Platforma uključuje nekoliko dobro integriranih značajki koje možete kombinirati, uključujući analitiku te pozadinske programe za mobilne uređaje.

Implementacija Firebase-a znači uključivanje gotove pozadine u vaš klijentski kod kako biste ga učinili dinamičnim.

Firebase platforma nudi za pravljenje i API-je za upravljanje bazom podataka, autentifikaciju, push obavijesti, hosting u oblaku i još mnogo toga.

Neke od ključnih karakteristika:

1. Skladištenje
2. Hosting
3. Autentifikacija
4. Firebase ML
5. Ugrađena push obavještenja

Skladištenje

Google Firebase koristi namjenske NoSQL baze podataka zasnovane na oblaku, Firestore i bazu podataka u realnom vremenu za pohranjivanje informacija. Kao i druge NoSQL baze podataka, one spremaju informacije u zbirke i dokumente.

Hosting

Također možete lako hostirati svoju web aplikaciju na Firebaseu. Nudeći mikrousluge, Firebase vam omogućava da hostujete i brzo implementirate svoju web aplikaciju uz nekoliko naredbi. Kada to učinite, vaša aplikacija se nalazi na globalno distribuiranim mrežama za isporuku sadržaja (CDN). Ovo osigurava da korisnici mogu čitati i pisati u vašu aplikaciju bez zastoja.

Autentifikacija

Jedna od funkcija za uštedu vremena koju želite istražiti u Firebaseu je njegova usluga provjere autentičnosti. Kada svoju aplikaciju povežete s Firebaseom, možda nećete morati kreirati zasebno sučelje za prijavu. Da uštedite vrijeme, možete koristiti njegovo ugrađeno korisničko sučelje za prijavu da prijavite korisnike u svoju aplikaciju.

Firebase ML

Firebase nudi mogućnosti mašinskog učenja za modele obuke. Stoga vam omogućava da integrirate prilagođene modele u svoju aplikaciju i hostirate ih u oblaku.

Ugrađena push obavještenja

Kodiranje i implementacija push obavijesti može biti naporno. Firebase-ovo ugrađeno push obavještenje omogućava vam da svojoj aplikaciji dodate personalizirane mogućnosti upozorenja u stvarnom vremenu bez pisanja posebne skripte od nule.

Opis projekta

Igrica se sastoji od SignUp i LogIn formi. Cilj igrice je pronaći svih 6 parova u što manje koraka, pri čemu se u bazu spremaju najbolji rezultati.

Rad sa Firebase-om

Konfiguracija

Prije svega instaliramo Firebase pomoću komande *npm install firebase*.

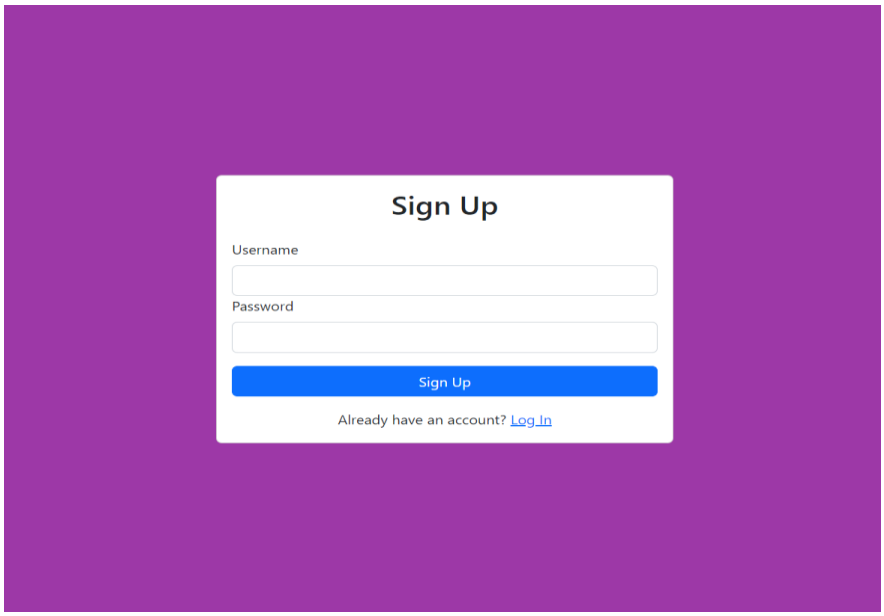
Zatim potrebno je uključiti Firbase u projekat i izvršiti konfiguraciju pomoću podataka iz Firebase Console kada kreiramo projekat.

Inicijaliziramo Firebase u projektu i kreiramo objekat app.

Da bi kasnije pristupali Firebase-u u aplikaciji koristimo objekat app iz ove konfiguracije, u komandi *const db = getFirestore(app)*

Sign Up

Firebase autentifikacija omogućava svojim korisnicima da se autentifikuju sa Firebase-om koristeći svoje adrese e-pošte i lozinke.

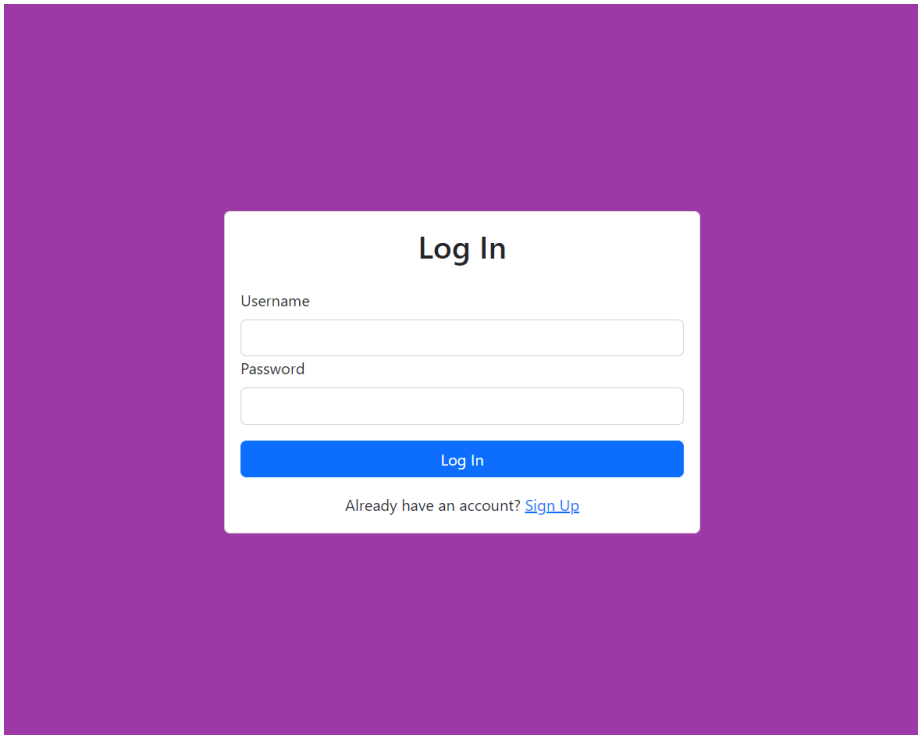


Kreiranje novog profila se vrši pomoću funkcije *createUserWithEmailAndPassword* iz biblioteke "firebase/auth", kojoj prosljeđujemo objekat auth iz Firebase-konfiguracije, email i lozinku.

Ukoliko je profil već kreiran, korisnik je automatski logovan.

```
13    const register = async (e) => {
14        e.preventDefault()
15        try {
16            const user = await createUserWithEmailAndPassword(auth, email, password);
17            console.log(user);
18            navigate('/');
19        } catch (err) {
20            console.log(err);
21            setError(err.code)
22        }
23    }
```

Log In



Prijavljivanje korisnika je slično kreiranju novog profila, samo se koristi funkcija *signInWithEmailAndPassword* kojoj prosljeđujemo objekat auth iz Firebase-konfiguracije, email i lozinku.

```
13    const login = async (e) => {
14        e.preventDefault()
15        try {
16            const user = await signInWithEmailAndPassword(auth, email, password);
17            console.log(user);
18            navigate('/game')
19        } catch (err) {
20            console.log(err.message);
21            if (err.code === "auth/wrong-password") {
22                setError("Wrong password")
23            }
24            else if (err.code === "auth/user-not-found") {
25                setError("User not found")
26            }
27            else {
28                setError(err.code)
29            }
30        }
31    }
```

Najbolji rezultat

Prilikom učitavanja stranice, iz kolekcije se pomoću funkcije getDocs učitava najbolji rezultat trenutno prijavljenog igrača.

```
56      const getScore = async () => {
57          const querySnapshot = await getDocs(collection(db, "players"));
58              querySnapshot.forEach((doc) => {
59                  if (doc.id === user.uid) {
60                      setBestSc(doc.data().bestScore)
61                  }
62              });
63      }
```

Spremanje najboljeg rezultata

Koristi se setDoc, i ukoliko u bazi već postoji igrač s istim id-em kao kod trenutnog igrača, podatak se mijenja ako je bolji rezultat, u suprotnom se ne mijenja, a ako igrač prvi put igra dodaje se novi dokument u kolekciju.

```
84          if (foundPairs === 6) {
85              if (bestSc > turns || bestSc === 0) {
86                  setBestSc(turns)
87                  setDoc(doc(db, "players", user.uid), {bestScore:turns});
88              }
89
90              setModalShow(true)
91              setFoundPairs(0)
92          }
```

Struktura igrice

Nakon logovanja prikaže se 12 kartica, gdje trebamo naći 6 parova. Također se prikaže mail trenutnog korisinika kao I njegov najbolji rezultat ukoliko je već igrao.

Dugme New Game za novu igru, kao i dugme Sign Out za odjavljivanje korisnika.

Shuffle cards funkcija

Glavna funkcija, gdje prvo naš niz od 6 slika dupliramo, zatim pravimo random niz i svakoj karti(slici) dodajemo random broj kao id. Na kraju kao niz postavljamo taj random niz.

```
33    const shuffleCards = () => {
34        const shuffledCards = [...cardImages, ...cardImages]
35            .sort(() => Math.random() - 0.5)
36            .map((card) => ({ ...card, id: Math.random() }))
37
38        setChoiceOne(null)
39        setChoiceTwo(null)
40        setCards(shuffledCards)
41        setTurns(0)
42    }
```

Provjera parova

Prvo provjeravamo da li su obe karte izabrane, a zatim da li je source te dvije slike jednak. Ukoliko jeste, prolazimo kroz naš niz karata i source svake karte upoređujemo sa izabranim, te ako se podudaraju povećavamo broj nađenih parova i vraćamo sve atribute te karte a atribut matched koji na početku bude false, mijenjamo u true.

```
65      useEffect(() => {
66          if (choiceOne && choiceTwo) {
67              setDisabled(true)
68              if (choiceOne.src === choiceTwo.src) {
69                  setCards(prevCards => {
70                      return prevCards.map(card => {
71                          if (card.src === choiceOne.src) {
72                              setFoundPairs(foundPairs + 1);
73                              return { ...card, matched: true }
74                          } else {
75                              return card
76                          }
77                      })
78                  })
79                  resetTurn()
80              } else {
81                  setTimeout(() => resetTurn(), 1000)
82              }
83          }
```

Provjera kraja igrice

Pri svakoj promjeni izbora karata, provjerava se da li broj nađenih parova jednak 6, ukoliko jeste provjerava se da li je rezultat (broj koraka) bolji od prethodnog u bazi ilia ko igrač prvi put igra, tea ko je nešto od to dvoje tačno, u bazi se dodaje (mijenja) najbolji rezultat.

```
84          if (foundPairs === 6) {
85              if (bestSc > turns || bestSc === 0) {
86                  setBestSc(turns)
87                  setDoc(doc(db, "players", user.uid), {bestScore:turns});
88              }
89
90              setModalShow(true)
91              setFoundPairs(0)
92          }
```

Kraj igrice

Kada pronađemo svih 6 parova, prikazuje se komponenta Score na kojoj je prikazan trenutni rezultat (broj koraka) kao i dotadašnji najbolji rezultat.

Također se prikazuju dva dugmeta New Game za početak nove igre, kao i Close za zatvaranje modala.