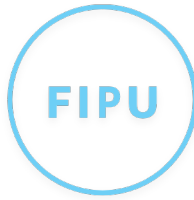


Programsko inženjerstvo

Nositelj: doc. dr. sc. Nikola Tanković

Asistent: mag. inf. Alesandro Žužić

Ustanova: Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli



Fakultet informatike u Puli

[7] Firebase

Firebase je platforma za razvoj aplikacija koja nudi brojne usluge poput baze podataka, autentifikacije, hostinga i analitike. U kombinaciji s Vue.js, Firebase omogućuje brz i jednostavan razvoj modernih web aplikacija s real-time funkcionalnostima.



- [Programsko inženjerstvo](#)
- [\[7\] Firebase](#)
 - [Uvod u Firebase](#)
 - [Postavljanje Firebase projekta](#)
 - [Konfiguriranje Vue projekta](#)
 - [Firebase Autentifikacija](#)
 - [Registracija/prijava korisnika](#)
 - [Email/Password registracija/prijava](#)
 - [Google registracija/prijava](#)
 - [Praćenje stanja autentifikacije](#)
 - [Odjava korisnika](#)
 - [Slanje verifikacijskog emaila](#)
 - [Ponovna autentifikacija korisnika](#)
 - [Upravljanje korisničkim računom](#)
 - [Promjena lozinke korisnika](#)
 - [Slanje emaila za resetiranje lozinke](#)
 - [Promjena email adrese korisnika](#)
 - [Brisanje korisničkog računa](#)

Uvod u Firebase

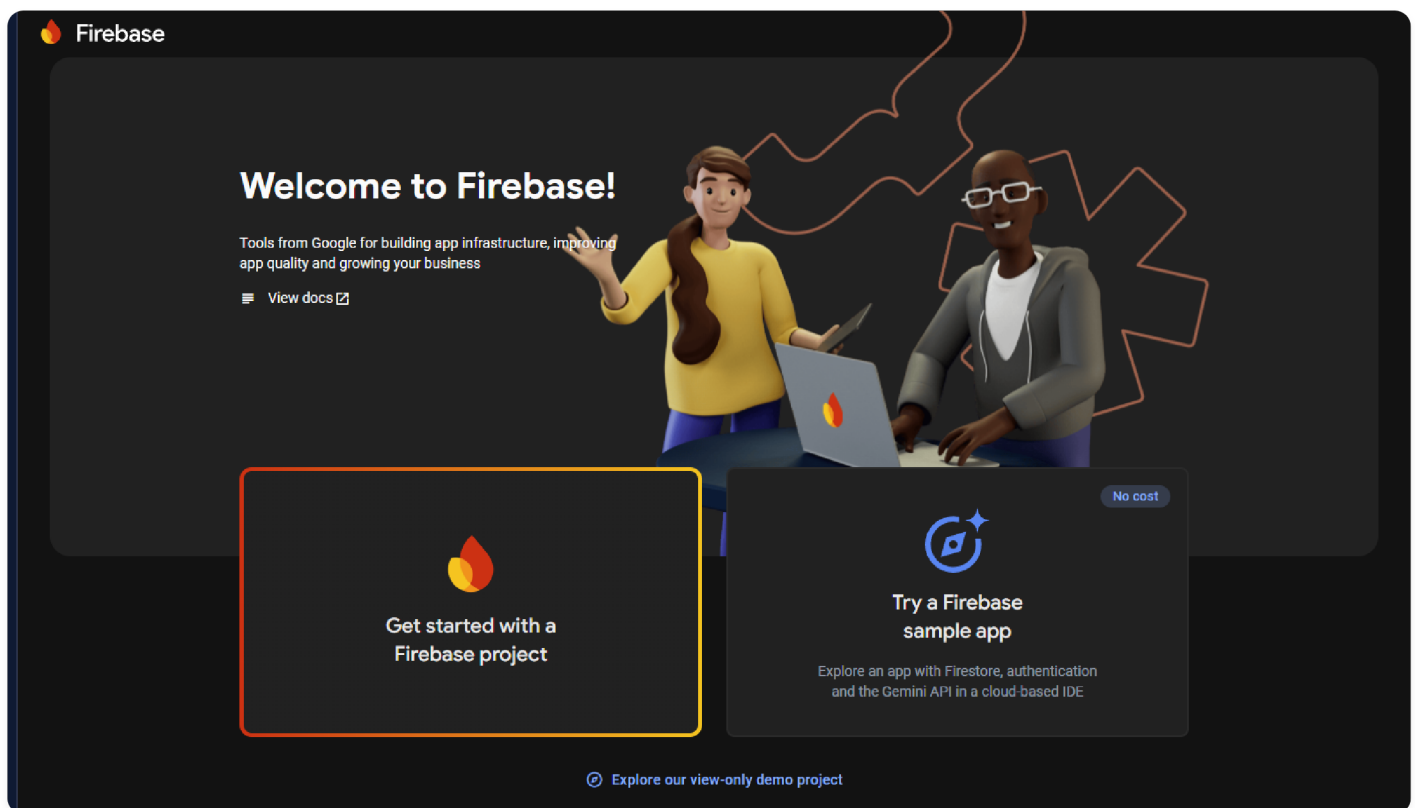
Firebase je sveobuhvatna platforma koju je razvio Google s ciljem da programerima omogući jednostavniji i brži razvoj mobilnih i web aplikacija. Umjesto da moramo brinuti o izgradnji i održavanju složene infrastrukture, Firebase nudi skup integriranih alata i servisa koji pokrivaju najvažnije potrebe tijekom cijelog ciklusa aplikacije – od autentifikacije korisnika, preko pohrane podataka, do praćenja performansi i analitike.



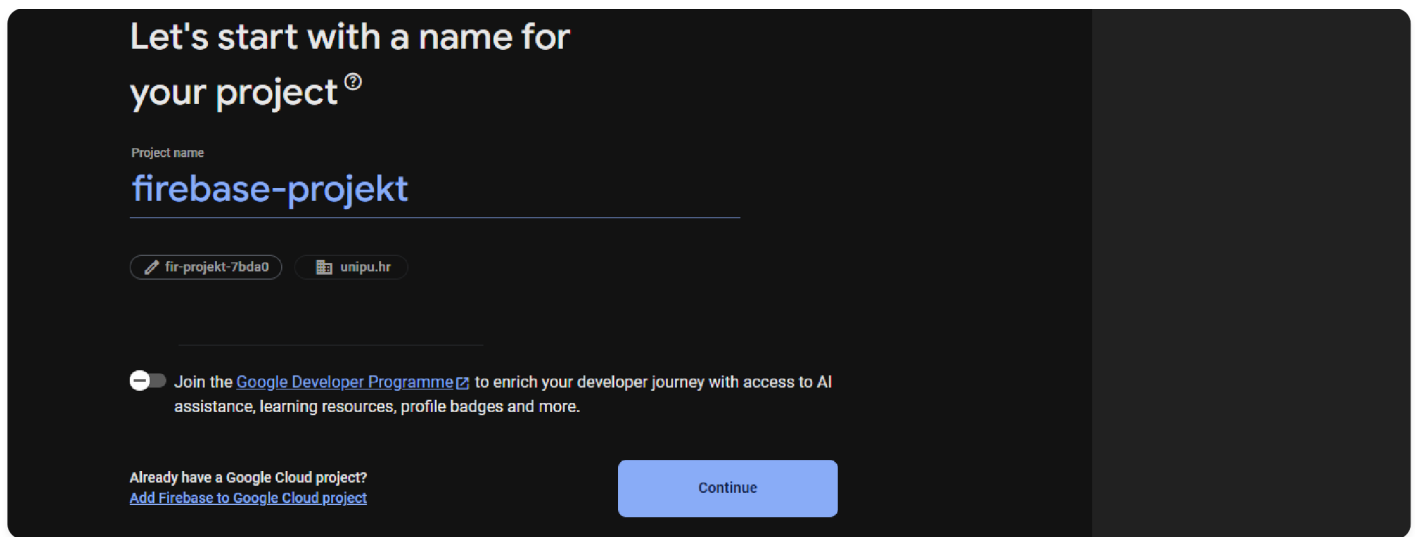
Postavljanje Firebase projekta

Prije nego uopće ubacimo Firebase u Vue aplikaciju trebamo napraviti i postaviti **Firebase projekt**. Da bi to napravili trebamo ići **Firebase konzolu**:

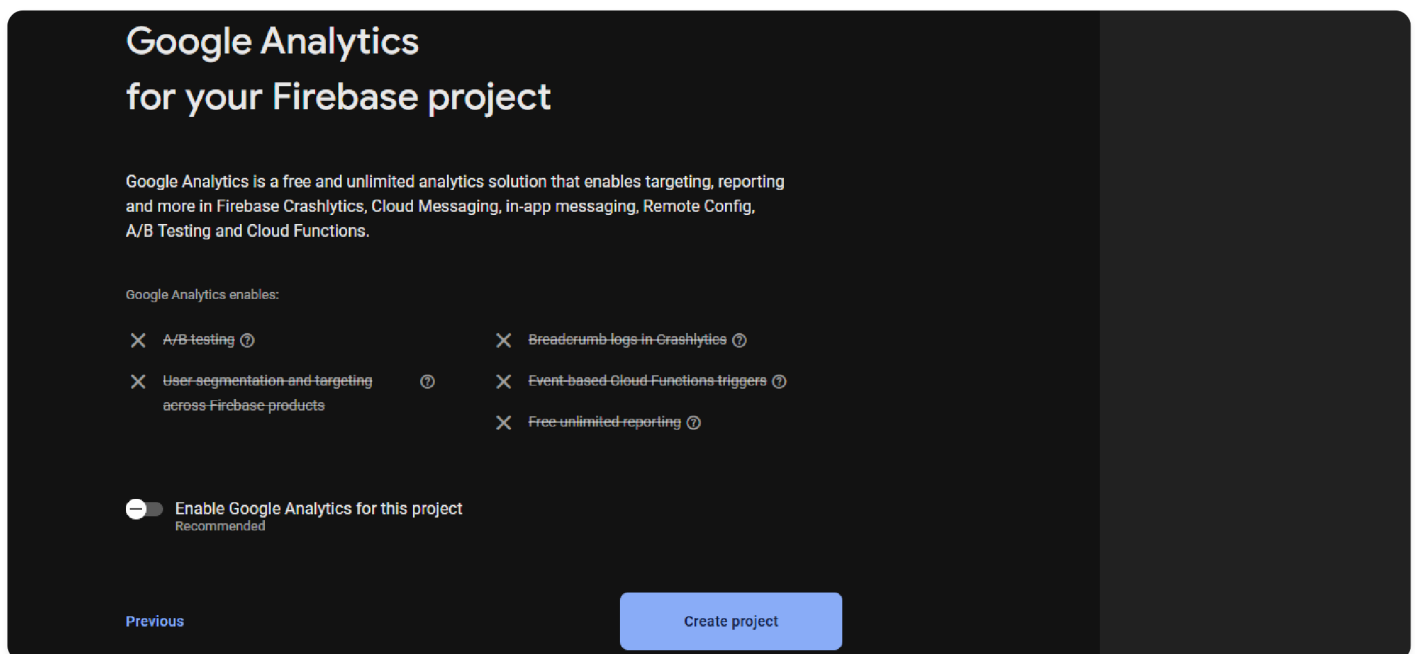
<https://console.firebase.google.com>



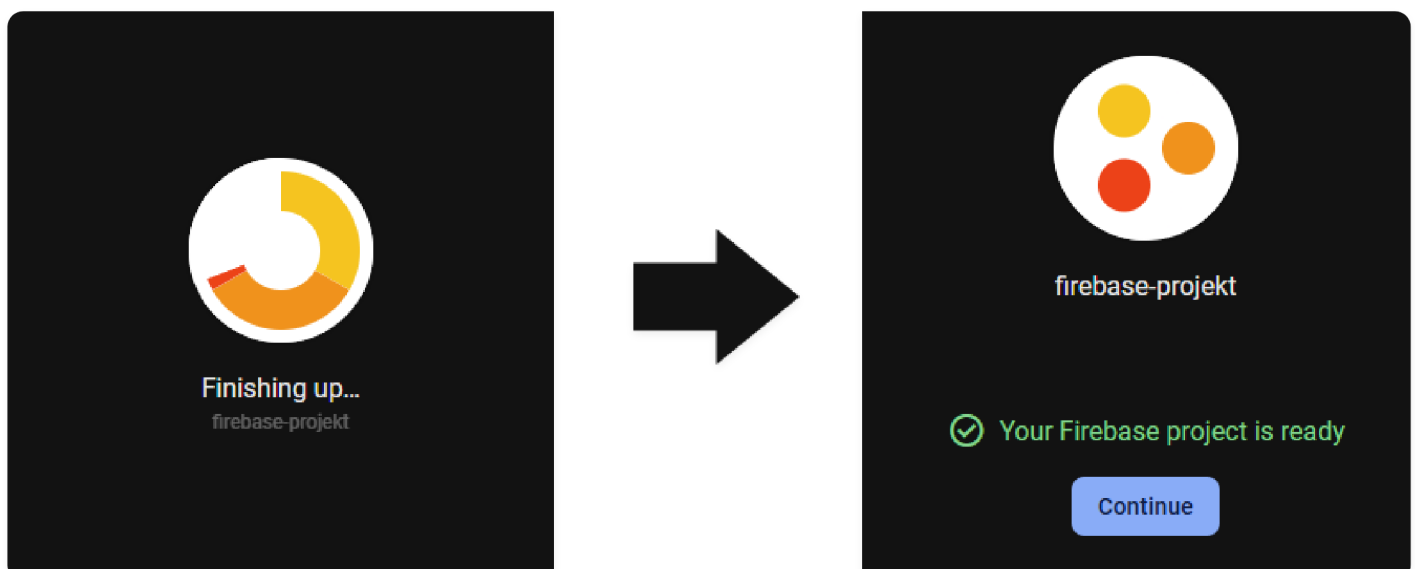
- Nakon što kliknemo na *Get started with a Firebase project* trebamo napisati **ime projekta**:



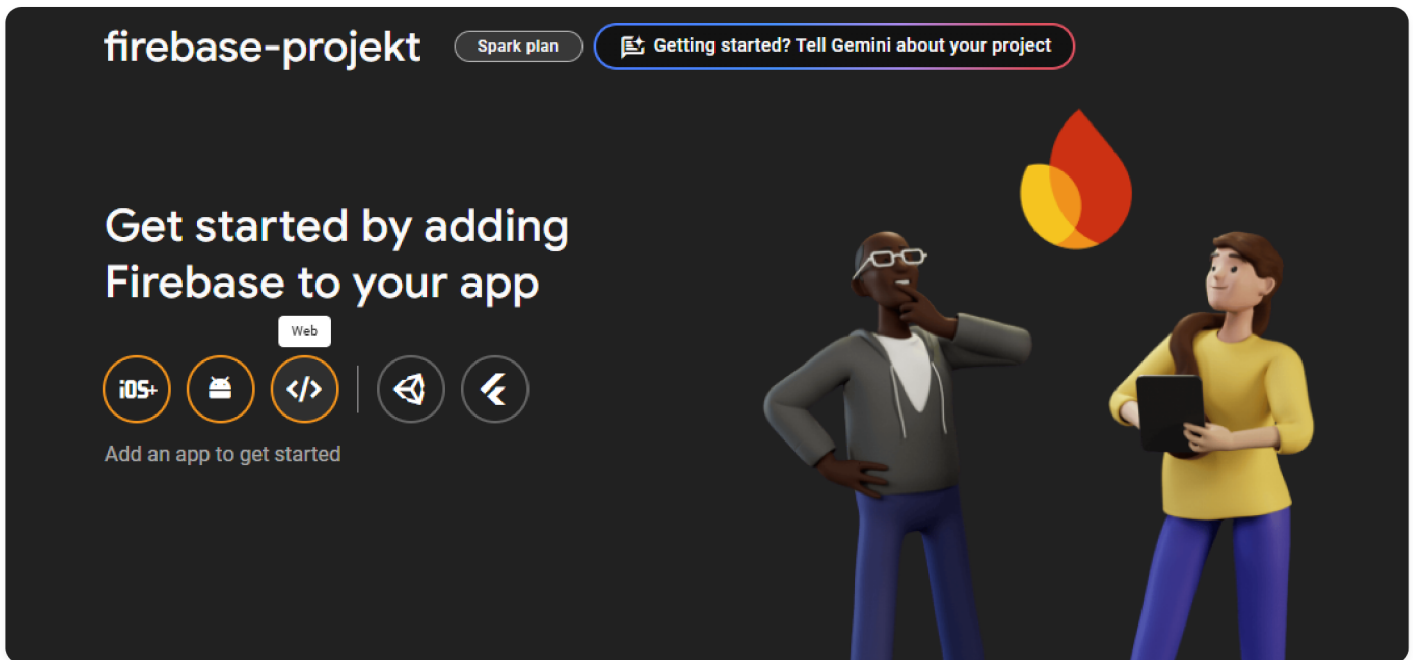
- Zatim ćemo **isključiti** *Google Analytics* s obzirom da nam to za sada ne treba, uvijek se može naknadno poslije uključiti:



- Kada smo isključili, možemo **kreirati projekt**:



- Sada ima novi svježi Firebase Projekt, gdje nam je sljedeći cilj dodati ga u Vue aplikaciju. S obzirom da je Vue **web aplikacija** odabrati ćemo **Web** app:



Konfiguriranje Vue projekta

Sljedeći korak nam je da sada registramo tu web aplikaciju tako da odaberemo **naziv**, *Firebase Hosting* nam nije potreban trenutno:

- I za kraj nam ostaje da instaliramo **Firebase paket** u Vue i kopiramo dani kôd u Vue projekt:

```
npm install firebase
```

✓ Register app

2 Add Firebase SDK

☒ Use npm ☐ Use a <script> tag

If you're already using [NPM](#) and a module bundler such as [webpack](#) or [Rollup](#), you can run the following command to install the latest SDK ([learn more](#)):

```
$ npm install firebase
```

Then, initialise Firebase and begin using the SDKs for the products that you'd like to use.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
const firebaseConfig = {
  apiKey: "AIzaSyB1e1--PL4p274X23L3nF20016I",
  authDomain: "fir-project-3ad60.firebaseapp.com",
  projectId: "fir-project-3ad60",
  storageBucket: "fir-project-3ad60.appspot.com",
  messagingSenderId: "73016228225",
  appId: "1:73016228225:web:3:20200404:4:0172d1f"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
```

Note: This option uses the [modular JavaScript SDK](#), which provides reduced SDK size.

Learn more about Firebase for web: [Get started](#), [Web SDK API Reference](#), [Samples](#)

Continue to the console

- Dani kôd je najbolje smjestiti u zasebnu `firebase.js` datoteku unutar `src` mape i učitati ju unutar `main.js` datoteke:

```
src
├── assets
├── components
├── App.vue
├── firebase.js
└── main.js
```

```
import './assets/main.css'
import './firebase.js'

import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

S obzirom da ćemo koristiti samo Firebase **autentifikaciju** i **firestore**, onda ćemo urediti `firebase.js` datoteku tako da učitamo i namjestimo ta dva servisa:

```
// Učitavanje potrebnih funkcija
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";

// Firebase konfiguracija
const firebaseConfig = {
  apiKey: "API_KEY",
  authDomain: "PROJECT_ID.firebaseapp.com",
  projectId: "PROJECT_ID",
  storageBucket: "PROJECT_ID.appspot.com",
  messagingSenderId: "SENDER_ID",
  appId: "APP_ID"
};

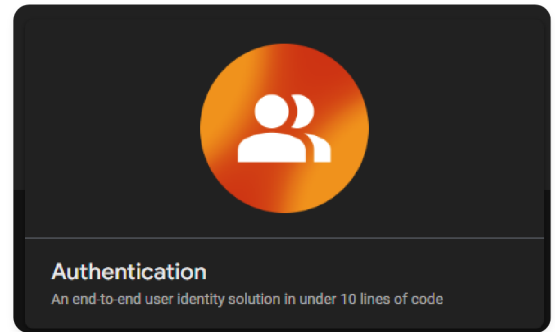
// Firebase inicijalizacija
const app = initializeApp(firebaseConfig);

// Inicijalizacija servisa
const auth = getAuth(app); // auth instanca
const db = getFirestore(app); // database instanca

// Izvoz servisa
export { auth, db };
```

Firebase Autentifikacija

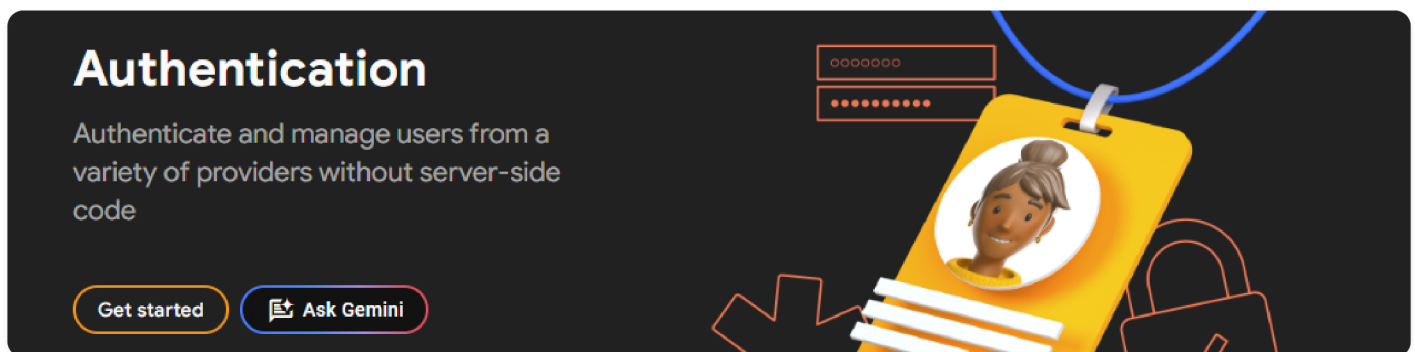
Jedna od ključnih komponenti Firebasea je **autentifikacija**, koja omogućuje jednostavnu integraciju različitih metoda prijave korisnika, uključujući prijavu putem emaila i lozinke, kao i prijavu putem trećih strana kao što su Google, Facebook ili Apple. Ova usluga značajno smanjuje vrijeme potrebno za implementaciju sigurnog sustava za upravljanje korisničkim računima.



Firestore Docs: <https://firebase.google.com/docs/auth>

Registracija <input type="text" value="Email..."/> <input type="text" value="Lozinka..."/> <button>Registrij se</button>	Prijava <input type="text" value="Email..."/> <input type="text" value="Lozinka..."/> <button>Prijavi se</button>	Google registracija <button>Sign in with Google</button>	Stanje autentifikacije <p>Nema prijavljenog korisnika!</p>
Odjava <button>Odjavi se</button>	Ponovna autentifikacija <input type="text" value="Email..."/> <input type="text" value="Lozinka..."/> <button>Ponovno se autentificiraj</button>	Ponovna Google autentifikacija <button>Google reauthentication</button>	Slanje email potvrde <button>Pošalji email potvrdu</button>
Promjena lozinke <input type="text" value="Nova lozinka"/> <button>Promijeni lozinku</button>	Resetiranje lozinke <input type="text" value="Email..."/> <button>Resetiraj lozinku</button>	Promjena emaila <input type="text" value="Email..."/> <button>Promijeni email</button>	Brisanje korisničkog računa <button>Obrisi korisnički račun</button>


Da bi uopće koristili **autentifikaciju**, trebamo je uključiti u **Firestore konzoli**.





- Kada smo je uključili, možemo odabrati **vrstu autentifikacije**, gdje ćemo za početak odabrati `Email/Password`:

Get started with Firebase Auth by adding your first sign-in method


Native providers


 Email/Password


 Phone


 Anonymous


Additional providers


 Google

 Facebook


 Play Games


 Game Center

 Apple

 GitHub

 Microsoft

 Twitter

 Yahoo

Custom providers

 OpenID Connect

 SAML

- I uključiti autentifikaciju putem Email/Password, registracija bez email-a nam ne treba, pa neka ostane isključeno:


 Email/Password

☒ Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery and email address change primitives. [Learn more](#)

Email link (passwordless sign-in)

☐ Enable

 Delete provider

Cancel

Save


- Sada možemo vidjeti da smo uspješno uključili Email/Password autentifikaciju, uvijek možemo dodati druge vrste autentifikacije.


Sign-in providers

Add new provider

Provider

Status

 Email/Password

 Enabled

Registracija/prijava korisnika

Prvi korak u većini aplikacija koja zahtjeva rad s korisnicima je njihova registracija. Trebamo korisnicima omogućiti način da kreiraju novi korisnički račun ako ga već nemaju ili da se prijave u postojeći ako ga imaju. Firebase nam nudi različite metode registracije/prijave:

Vrsta	Funkcija	Opis
Email/Password	<code>createUserWithEmailAndPassword()</code>	Kreira novog korisnika koristeći email adresu i lozinku.
Email/Password	<code>signInWithEmailAndPassword()</code>	Prijava korisnika pomoću email adrese i lozinke.
Socijalna mreža (OAuth)	<code>signInWithPopup(provider)</code>	Registracija putem vanjskih servisa kao što su Google , Facebook, Twitter/X, GitHub, Apple, Microsoft i Yahoo
Anonimna autentikacija	<code>signInAnonymously()</code>	Omogućava privremeni anonimni pristup bez stvarnog korisničkog računa.
Telefonski broj	<code>signInWithPhoneNumber()</code>	Registracija pomoću broja mobitela i SMS verifikacije.

Email/Password registracija/prijava

Najjednostavnija registracija koju Firebase nudi jest putem email adrese i lozinke, koristeći funkciju `createUserWithEmailAndPassword()` koju učitavamo iz paketa `firebase/auth`.

Firestore Docs: <https://firebase.google.com/docs/auth/web/password-auth>

Ova funkcija prima tri argumenta:

- `@param auth` — **instanca autentikacije** (*Auth*)
- `@param email` — korisnička **email adresa**
- `@param password` — **lozinka** koju je korisnik odabrao

Auth instancu učitavamo iz `firebase.js` datoteke, gdje smo je prethodno definirali. Ostala dva argumenta email i lozinku sami dodjeljujemo, obično dohvaćamo iz HTML elemenata, primjerice iz `<input>` polja forme.

Budući da se *Firestore funkcije* izvršavaju **asinhrono**, koristit ćemo `async/await` kako bismo pričekali njihov završetak. Također, za hvatanje eventualnih grešaka pri registraciji koristimo `try-catch` blok. Na taj način osiguravamo da aplikacija pravilno reagira i u slučaju da dođe do greške, primjerice ako email već postoji ili je lozinka prekratka.

```

<script setup>
import { ref } from 'vue'
import { createUserWithEmailAndPassword } from 'firebase/auth'
import { auth } from '@/firebase.js'

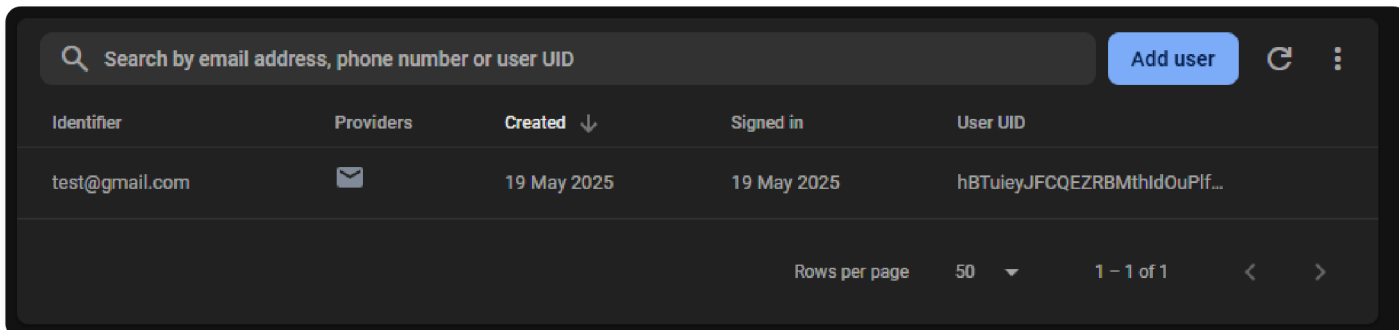
const email = ref('')
const password = ref('')
const response = ref({ error: false, message: '' })


const register = async () => {
  try {
    const userCredential = await createUserWithEmailAndPassword(auth, email.value,
password.value);
    response.value.error = false;
    response.value.message = 'Korisnik registriran: ' +
JSON.stringify(userCredential.user);
  } catch (error) {
    response.value.error = true;
    response.value.message = 'Greška pri registraciji: ' + error.message;
  }
};
</script>

<template>
  <form @submit.prevent="register">
    <h2>Registracija</h2> <hr>
    <input v-model="email" type="email" placeholder="Email...">
    <input v-model="password" type="password" placeholder="Lozinka...">
    <button type="submit">Registriraj se</button>
    <span :class="response.error ? 'text-rose-600' : 'text-emerald-600'">{{
response.message }}</span>
  </form>
</template>

```

- Nakon uspješnog kreiranja korisničkog računa, korisnik će automatski biti prijavljen u aplikaciji i spremljen među korisnike Firebase projekta:



Identifier	Providers	Created ↓	Signed in	User UID
test@gmail.com		19 May 2025	19 May 2025	hBTuieyJFCQEZRBMthIdOuPlf...

Rows per page 50 1 – 1 of 1

- Kreiranje računa može biti neuspješno ako korisnički račun već postoji ili ako je lozinka neispravna

Registracija

Registriraj se

```

{
  "uid": "hBTuieyJFCQEZRBMthIdOuPlfsj1",
  "email": "test@gmail.com",
  "emailVerified": false,
  "isAnonymous": false,
  "providerData": [
    {
      "providerId": "password",
      "uid": "test@gmail.com",
      "displayName": null,
      "email": "test@gmail.com",
      "phoneNumber": null,
      "photoURL": null
    }
  ],
  "stsTokenManager": {

```

Registracija

Registriraj se

Firestore: Error (auth/email-already-in-use).

Registracija

Registriraj se

Greška pri registraciji: Firestore: Password should be at least 6 characters (auth/weak-password).

Registracija

Registriraj se

Greška pri registraciji: Firestore: Error (auth/missing-email).

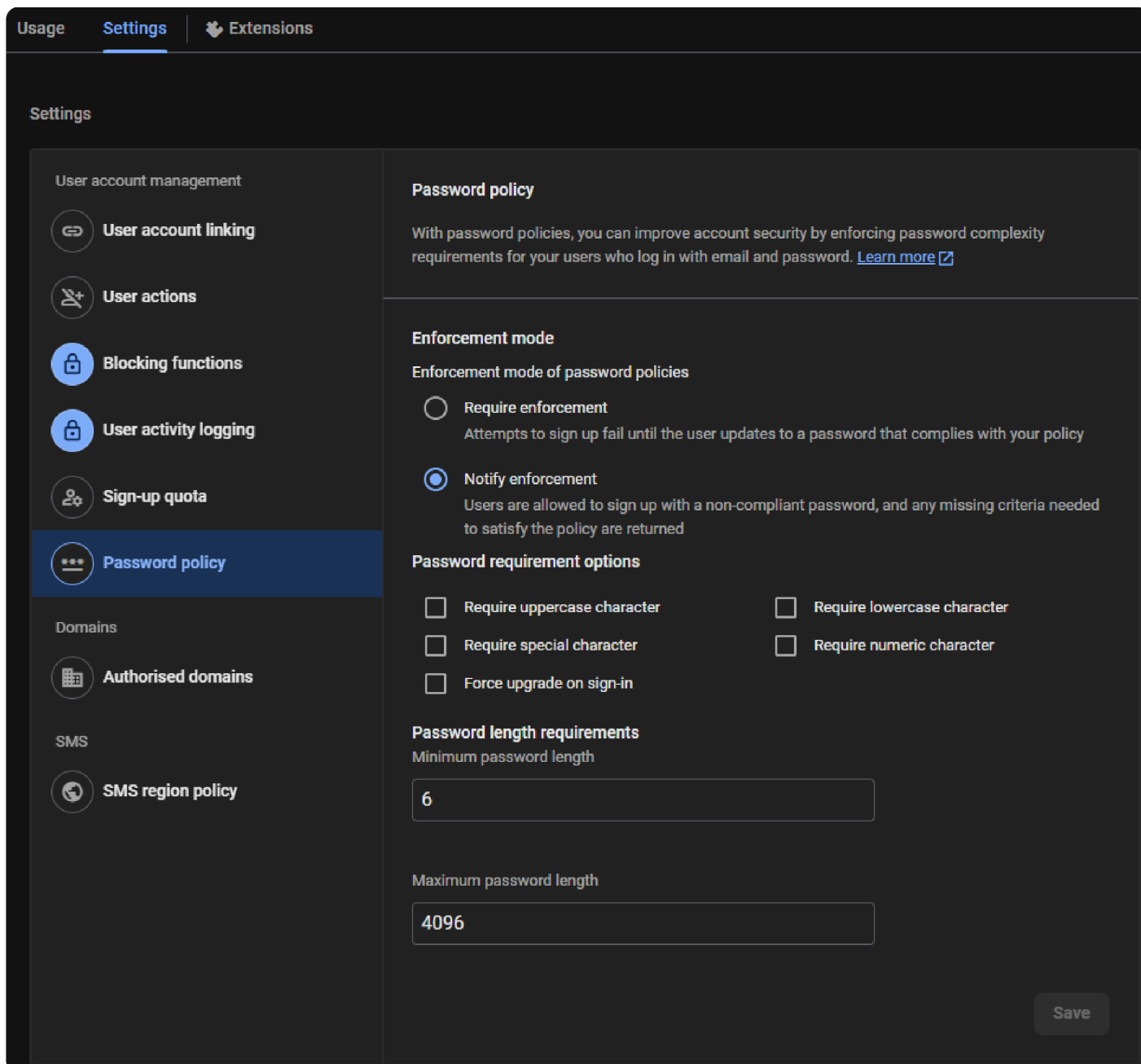
Registracija

Registriraj se

Greška pri registraciji: Firestore: Error (auth/invalid-email).

- Možemo poboljšati sigurnost korisničkih računa primjenom složenijim lozinki

Za konfiguraciju pravila lozinki u projektu, otvorit ćemo karticu **Password policy** unutar stranice **Authentication Settings** u Firebase konzoli



Napomena: Email adresa služi kao jedinstveni identifikator korisnika i omogućuje resetiranje lozinke putem emaila. Ova funkcija kreira novi korisnički račun i postavlja početnu lozinku za korisnika.

Kada je korisnik već registriran i ponovno želi koristiti aplikaciju, ne treba izrađivati novi korisnički račun — umjesto toga, omogućujemo mu da se jednostavno prijavi u svoj postojeći račun, koristeći funkciju `signInWithEmailAndPassword()` koju učitavamo iz paketa `firebase/auth`.

Firebase Docs: <https://firebase.google.com/docs/auth/web/password-auth>

Ova funkcija prima tri argumenta:

- `@param auth` — **instanca autentikacije** (*Auth*)
- `@param email` — korisnička **email adresa**
- `@param password` — **lozinka** koju je korisnik odabrao

```

<script setup>
import { ref } from 'vue'
import { signInWithEmailAndPassword } from 'firebase/auth'
import { auth } from '@/firebase.js'

const email = ref('')
const password = ref('')
const response = ref({ error: false, message: '' })

const login = async () => {
  try {
    const userCredential = await signInWithEmailAndPassword(auth, email.value,
password.value);
    response.value.error = false;
    response.value.message = 'Korisnik prijavljen: ' +
JSON.stringify(userCredential.user);
  } catch (error) {
    response.value.error = true;
    response.value.message = 'Greška pri prijavi: ' + error.message;
  }
};
</script>

<template>
  <form @submit.prevent="login">
    <h2>Prijava</h2> <hr>
    <input v-model="email" type="email" placeholder="Email...">
    <input v-model="password" type="password" placeholder="Lozinka...">
    <button type="submit">Prijavi se</button>
    <span :class="response.error ? 'text-rose-600' : 'text-emerald-600'">{{
response.message }}</span>
  </form>
</template>

```

- Nakon uspješe prijave, korisnik će automatski biti prijavljen u aplikaciji u suprotnom prijava može biti neuspješna ako korisnički račun ne postoji ili ako su podaci neispravni:

Prijava

Prijavi se

```
{
  "uid": "hBTuieyJFCQEZRBMthIdOuPlfsj1",
  "email": "test@gmail.com",
  "emailVerified": false,
  "isAnonymous": false,
  "providerData": [
    {
      "providerId": "password",
      "uid": "test@gmail.com",
      "displayName": null,
      "email": "test@gmail.com",
      "phoneNumber": null,
      "photoURL": null
    }
  ],
  "stsTokenManager": {
```

Prijava

Prijavi se

Firestore: Error (auth/missing-password).

Prijava

Prijavi se

Firestore: Error (auth/invalid-credential).

Napomena: Firebase iz sigurnosnih razloga nikada neće precizno naznačiti je li prilikom prijave pogrešan email ili lozinka. Umjesto toga, prikazuje općenitu poruku o neuspješnoj autentikaciji. Na taj način se sprječava mogućnost otkrivanja postojećih korisničkih računa zlonamjernim pokušajima pogađanja email adresa. Ovo je standardna sigurnosna praksa kojom se štite korisnički podaci.

Google registracija/prijava

Druga najjednostavnija vrsta registracije/prijave koju Firebase nudi jest putem **Google računa**, gdje iz paketa `firebase/auth` učitavamo funkciju `signInWithPopup()` i konstruktor `GoogleAuthProvider()`.

Firestore Docs: <https://firebase.google.com/docs/auth/web/google-signin>

Funkcija `signInWithPopup()` prima dva argumenta:

- `@param auth` — **instanca autentikacije** (*Auth*)
- `@param provider` — Autentikacijski *provider* mora biti **OAuthProvider** (u ovom slučaju `GoogleAuthProvider()`)

Auth instancu učitavamo iz `firebase.js` datoteke, gdje smo je prethodno definirali. Definiramo novi **Google provider** koristeći konstruktor `GoogleAuthProvider()`:

```

<script setup>
import { ref } from 'vue'
import { GoogleAuthProvider, signInWithPopup } from 'firebase/auth'
import { auth } from '@firebase.js'
import GoogleButton from './GoogleButton.vue'

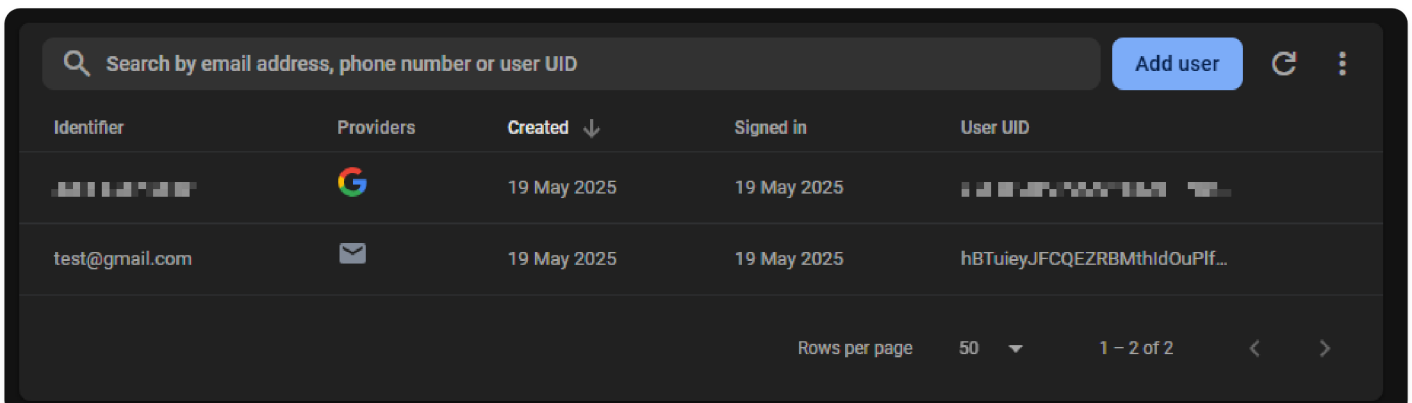
const response = ref({ error: false, message: '' })





const register = async () => {
  try {
    const provider = new GoogleAuthProvider();
    const userCredential = await signInWithPopup(auth, provider);
    response.value.error = false;
    response.value.message = 'Korisnik registriran: ' +
JSON.stringify(userCredential.user);
  } catch (error) {
    response.value.error = true;
    response.value.message = 'Greška pri registraciji: ' + error.message;
  }
};
</script>

<template>
  <form @submit.prevent="register">
    <h2>Registracija</h2> <hr>
    <GoogleButton/>
    <span :class="response.error ? 'text-rose-600' : 'text-emerald-600'">{{
response.message }}</span>
  </form>
</template>

```

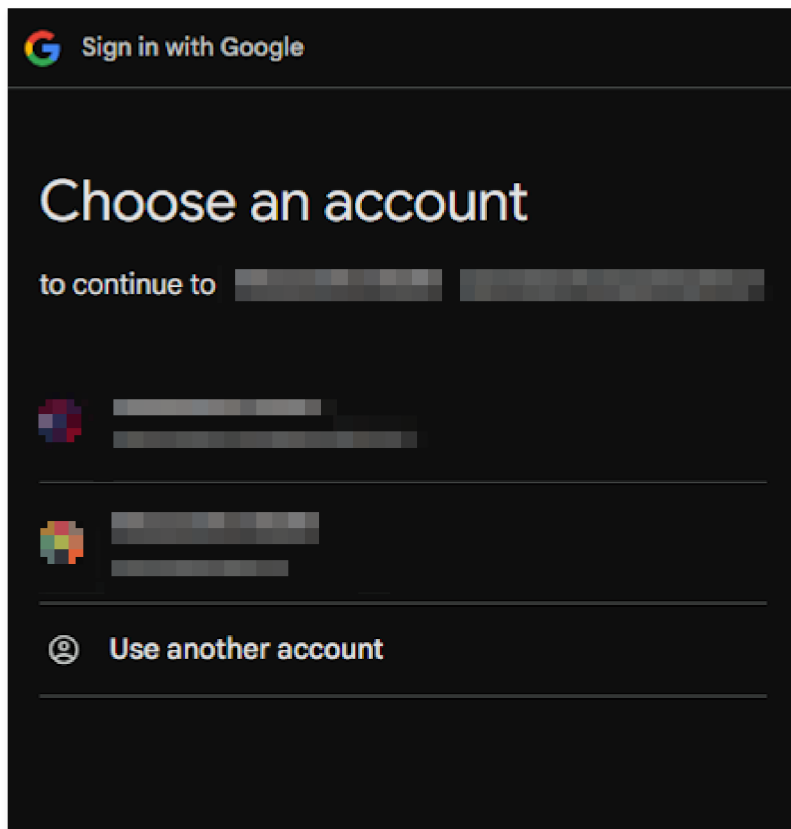
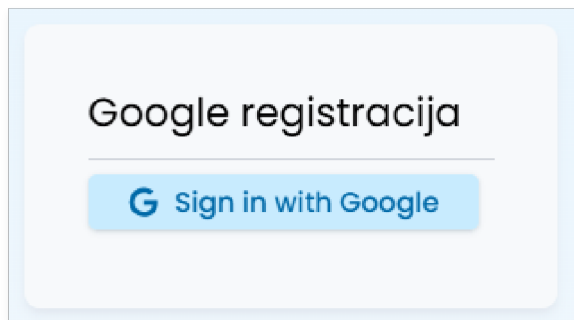
- Nakon uspješnog kreiranja korisničkog računa, korisnik će automatski biti prijavljen u aplikaciji i spremljen među korisnike Firebase projekta:



Identifier	Providers	Created ↓	Signed in	User UID
		19 May 2025	19 May 2025	
test@gmail.com		19 May 2025	19 May 2025	hBTuieyJFCQEZRBMthldOuPlf...

Rows per page 50 1 – 2 of 2

- Pri korištenju `signInWithPopup()` funkcije, otvara se novi skočni prozor za biranje Google računa



Napomena: Ako korisnik s istim računom već postoji, Firebase će ga automatski samo prijaviti u aplikaciju.

Praćenje stanja autentifikacije

Nakon uspješne registracije ili prijave, važno je znati tko je trenutno aktivan korisnik u aplikaciji. Da korisnik ne bi morao iznova unositi podatke prilikom svakog otvaranja aplikacije, Firebase omogućuje **praćenje stanja autentifikacije** putem funkcije `onAuthStateChanged()`.

- Ova funkcija dolazi iz paketa `firebase/auth`, i služi za slušanje (*praćenje*) promjena u autentifikacijskom stanju — npr. kad se korisnik prijavi, odjavi ili kada se aplikacija ponovno pokrene, a korisnik je i dalje prijavljen.

Funkcija `onAuthStateChanged()` radi na sličan način kao Vue `watch()` funkcija — promatra promjene nad **Auth instancom** u stvarnom vremenu. Kada se status korisnika promijeni (*npr. prijava, odjava ili automatska obnova sesije*), poziva se funkcija povratnog poziva (*callback*) s informacijom o trenutnom korisniku.

- Ovo je posebno korisno za aplikacije koje žele prikazati različito sučelje ovisno o tome je li korisnik prijavljen (*npr. prikaz korisničkog profila ili forme za prijavu*).


```

<script setup>
import { ref } from 'vue';
import { onAuthStateChanged } from 'firebase/auth';
import { auth } from '@/firebase.js'

const user = ref(null)

onAuthStateChanged(auth, (currentUser) => {
  if (currentUser) {
    user.value = currentUser;
  } else {
    user.value = null;
  }
});
</script>

<template>
  <form>
    <h2>Stanje autentifikacije</h2>
    <hr>
    <span v-if="!user" class="text-rose-600"> Nema prijavljenog korisnika! </span>
    <span v-else class="text-emerald-600"> Prijavljen korisnik: <b>{{ user.email
  }}</b> </span>
  </form>
</template>

```

- Nakon svake uspješe prijave, registracije ili odjave, pozvat će se `onAuthStateChanged()` funkcija:

Stanje autentifikacije

Prijavljen korisnik: **azuzic@unipu.hr**

Stanje autentifikacije

Nema prijavljenog korisnika!

Napomena: `onAuthStateChanged()` neće automatski reagirati kada korisnik potvrdi svoju email adresu putem verifikacijske poruke. Kako bi se dohvatilo ažurirano stanje korisnika (*npr. verified email*), korisnik se mora ponovno prijaviti ili ručno osvježiti stranicu. Alternativno, moguće je dohvatiti najnovije podatke pozivom metode `reload()` na `auth.currentUser`.

Odjava korisnika

Kada korisnik završi s korištenjem aplikacije ili želi prijeći na drugi korisnički račun, potrebno mu je omogućiti jednostavnu i sigurnu odjavu. Firebase za tu svrhu nudi funkciju `signOut()`, koju također učitavamo iz paketa `firebase/auth`. Ova funkcija odjavljuje trenutno prijavljenog korisnika iz aplikacije.

```
<script setup>
import { signOut } from 'firebase/auth';
import { auth } from '@/firebase.js'

const logout = () => {
  signOut(auth);
};
</script>

<template>
  <form @submit.prevent="logout">
    <h2>Odjava</h2>
    <hr>
    <button type="submit">Odjavi se</button>
  </form>
</template>
```

Odjava

Odjavi se

Korisnik je odjavljen!

Slanje verifikacijskog emaila

Nakon što se korisnik registrira pomoću emaila i lozinke, preporučuje se da **potvrdi** svoju **email adresu**. Time se dodatno osigurava autentičnost korisnika i omogućuje bolja zaštita računa (*npr. kod resetiranja lozinke*).

Firebase nudi funkciju `sendEmailVerification()` kojom možemo poslati korisniku **verifikacijski email** na adresu s kojom je registriran. Funkcija prima kao argument trenutno prijavljenog korisnika (`auth.currentUser`) i šalje automatski generiranu poruku na njegov email.

```
<script setup>
import { sendEmailVerification } from 'firebase/auth';
import { auth } from '@/firebase.js'

const sendVerification = async () => {
  await sendEmailVerification(auth.currentUser);
};
</script>

<template>
  <form @submit.prevent="sendVerification">
    <h2>Slanje email potvrde</h2>
    <hr>
    <button type="submit">Pošalji email potvrdu</button>
  </form>
</template>
```

- Nakon što korisniku pošaljemo verifikacijski email, on mora otvoriti svoju elektroničku poštu i kliknuti na poveznicu za potvrdu. Time uspješno potvrđuje svoju email adresu. Nakon potvrde, može se ponovno prijaviti u aplikaciju ili jednostavno osvježiti stranicu kako bi sustav prepoznao ažurirani status verifikacije.

Stanje autentifikacije

Prijavljen korisnik: `user@example.com`

Email potvrđen: `false`


Slanje email potvrde

Pošalji email potvrdu

Email potvrda je poslana!

Verify your email for project-XXXXXX

External



noreply@XXXXXX.firebaseio.com

to me ▾

Hello,

Follow this link to verify your email address.

[XXXXXX](#)

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your project-XXXXXX team

Your email has been verified

You can now sign in with your new account

Stanje autentifikacije

Prijavljen korisnik: `user@example.com`

Email potvrđen: `true`

Ponovna autentifikacija korisnika

Pojedine sigurnosno osjetljive radnje — poput brisanja korisničkog računa, postavljanja primarne email adrese ili promjene lozinke — zahtijevaju da je korisnik nedavno prijavljen. Ako pokušavamo izvršiti neku od tih radnji, a korisnik se prijavio prije predugog vremena, operacija će završiti s pogreškom.

U takvim slučajevima potrebno je **ponovno autentificirati** korisnika tako da od njega zatražimo nove podatke za prijavu.

- Ako korisnik koristi način prijave putem email adrese i lozinke, tada za ponovno autentificiranje koristimo funkciju `reauthenticateWithCredential()` zajedno s **providerom** `EmailAuthProvider`.

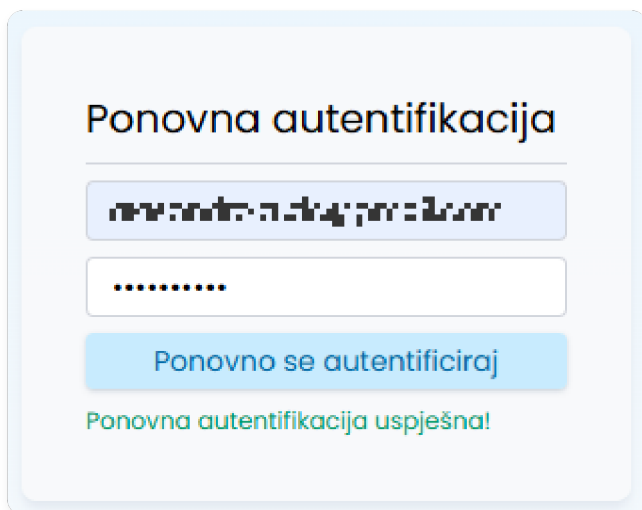
U tom slučaju, od korisnika tražimo da ponovno unese svoje podatke:

```
<script setup>
import { EmailAuthProvider, reauthenticateWithCredential } from 'firebase/auth';
import { ref } from 'vue';
import { auth } from '@/firebase';

const email = ref('');
const password = ref('');

const reauthenticate = async () => {
  const credential = EmailAuthProvider.credential(email.value, password.value);
  await reauthenticateWithCredential(auth.currentUser, credential);
};
</script>

<template>
  <form @submit.prevent="reauthenticate">
    <h2>Ponovna autentifikacija</h2>
    <hr>
    <input v-model="email" type="email" placeholder="Email...">
    <input v-model="password" type="password" placeholder="Lozinka...">
    <button type="submit">Ponovno se autentificiraj</button>
  </form>
</template>
```



Ponovna autentifikacija

Ponovno se autentificiraj

Ponovna autentifikacija uspješna!

- Ako se korisnik prijavio putem Google računa, tada za ponovno autentificiranje koristimo funkciju `reauthenticateWithPopup()` zajedno s **providerom** `GoogleAuthProvider`.


U tom slučaju, potrebno je inicirati novi popup prozor za Google prijavu, kojim korisnik ponovno potvrđuje svoj identitet:

```
<script setup>
import { GoogleAuthProvider, reauthenticateWithPopup } from 'firebase/auth';
import { auth } from '@firebase.js'
import GoogleButton from './GoogleButton.vue';

const reauthenticateWithGoogle = async () => {
  const provider = new GoogleAuthProvider();
  await reauthenticateWithPopup(auth.currentUser, provider);
};
</script>

<template>
  <form @submit.prevent="reauthenticateWithGoogle">
    <h2>Ponovna Google autentifikacija</h2>
    <hr>
    <GoogleButton>Google reauthentication</GoogleButton>
  </form>
</template>
```

Ponovna Google autentifikacija

 Google reauthentication

Ponovna autentifikacija uspješna!

Upravljanje korisničkim računom

Nakon što korisnik uspješno izvrši prijavu u aplikaciju, često je potrebno omogućiti mu dodatne funkcionalnosti vezane za upravljanje vlastitim računom. Firebase pruža sve potrebne alate za sigurno i jednostavno izvršavanje ovih operacija.

Promjena lozinke korisnika

Firebase omogućuje jednostavnu **promjenu lozinke** za trenutno prijavljenog korisnika pomoću funkcije `updatePassword()`. Prije izvršavanja ove operacije, potrebna je ponovna autentifikacija korisnika.

```
<script setup>
import { ref } from 'vue';
import { updatePassword } from 'firebase/auth';
import { auth } from '@/firebase';

const novaLozinka = ref('');

const promijeniLozinku = async () => {
  await updatePassword(auth.currentUser, novaLozinka.value);
};
</script>

<template>
  <form @submit.prevent="promijeniLozinku">
    <h2>Promjena lozinke</h2>
    <hr>
    <input v-model="novaLozinka" type="password" placeholder="Nova lozinka">
    <button type="submit">Promijeni lozinku</button>
  </form>
</template>
```

Promjena lozinke

Promijeni lozinku

Lozinka je uspješno promijenjena!

Slanje emaila za resetiranje lozinke

Za korisnike koji su zaboravili lozinku, Firebase nudi funkciju `sendPasswordResetEmail()` koja šalje posebno generiranu poveznicu na korisnikov email za **resetiranje lozinke**.

```

<script setup>
import { ref } from 'vue';
import { sendPasswordResetEmail } from 'firebase/auth';
import { auth } from '@/firebase';

const emailZaReset = ref('');

const posaljiResetLink = async () => {
  try {
    await sendPasswordResetEmail(auth, emailZaReset.value);
    alert('Email za resetiranje lozinke poslan! Provjerite svoj inbox.');
```

```

  } catch (error) {
    console.error('Greška pri slanju emaila:', error.message);
  }
};
</script>

<template>
  <input v-model="emailZaReset" type="email" placeholder="Vaša email adresa">
  <button @click="posaljiResetLink">Pošalji link za reset</button>
</template>

```


- Nakon što korisniku pošaljemo email za resetiranje lozinke, on mora otvoriti svoju elektroničku poštu i kliknuti na poveznicu za resetiranje. Time uspješno resetira svoju lozinku. Nakon potvrde, može se ponovno prijaviti u aplikaciju ili jednostavno osvježiti stranicu.


Resetiranje lozinke

Resetiraj lozinku

Email za resetiranje lozinke uspješno poslan!



Reset your password for project-




noreply@.io

to me ▾


Hello,

Follow this link to reset your project- password for your  account.




If you didn't ask to reset your password, you can ignore this email.

Thanks,

Your project-  team

Reset your password

for 

New password

.....

SAVE

Password changed

You can now sign in with your new password

Promjena email adrese korisnika

Za ažuriranje email adrese korisnika koristi se funkcija `verifyBeforeUpdateEmail()`. Kao i kod promjene lozinke, preporučuje se ponovna autentifikacija prije izvršavanja ove operacije.


```

<script setup>
import { ref } from 'vue';
import { verifyBeforeUpdateEmail } from 'firebase/auth';
import { auth } from '@/firebase';

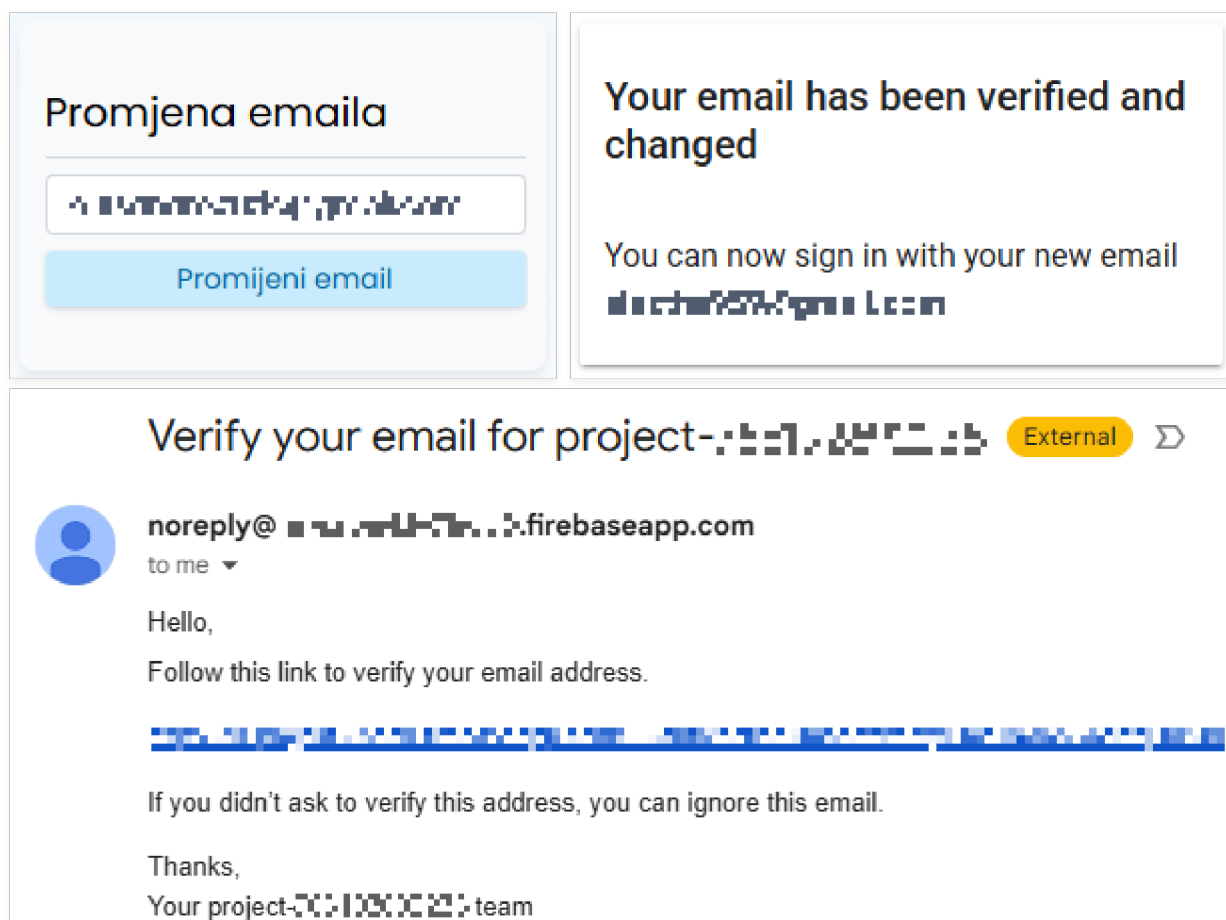
const noviEmail = ref('');

const promijeniEmail = async () => {
  await verifyBeforeUpdateEmail(auth.currentUser, noviEmail.value);
};
</script>

<template>
  <form @submit.prevent="promijeniEmail">
    <h2>Promjena emaila</h2>
    <hr>
    <input v-model="noviEmail" type="email" placeholder="Email...">
    <button type="submit">Promijeni email</button>
  </form>
</template>

```

- Nakon što korisniku pošaljemo verifikacijski email, on mora otvoriti svoju elektroničku poštu i kliknuti na poveznicu za potvrdu. Time uspješno potvrđuje svoju novu email adresu. Nakon potvrde, može se ponovno prijaviti u aplikaciju ili jednostavno osvježiti stranicu.



Brisanje korisničkog računa

Korisnici mogu trajno izbrisati svoj račun pomoću funkcije `deleteUser()`. Ova operacija zahtijeva nedavnu autentifikaciju radi sigurnosti.

```
<script setup>
import { deleteUser } from 'firebase/auth';
import { auth } from '@/firebase.js'

const obrisiKorisnika = async () => {
  await deleteUser(auth.currentUser);
};
</script>

<template>
  <form @submit.prevent="obrisiKorisnika">
    <h2>Brisanje korisničkog računa</h2>
    <hr>
    <button type="submit">Obriši korisnički račun</button>
  </form>
</template>
```

Brisanje korisničkog računa

Obriši korisnički račun

Korisnički račun je obrisani!