SVEUČILIŠTE ALGEBRA

Primijenjeno računarstvo

ZAVRŠNI RAD

RAZVOJ PROGRAMSKOG RJEŠENJA S NAPREDNIM SUČELJEM ZA POHRANU, ORGANIZACIJU I UPRAVLJANJE DATOTEKAMA I MAPAMA U OBLAKU

Kristijan Rosandić

Zagreb, kolovoz 2024.

SVEUČILIŠTE ALGEBRA

Primijenjeno računarstvo

ZAVRŠNI RAD

RAZVOJ PROGRAMSKOG RJEŠENJA S NAPREDNIM SUČELJEM ZA POHRANU, ORGANIZACIJU I UPRAVLJANJE DATOTEKAMA I MAPAMA U OBLAKU

Kristijan Rosandić

Mentor: Daniel Bele

Zagreb, kolovoz 2024.

Predgovor

Zahvaljujem svom mentoru, Danielu Beleu, na ukazanom povjerenju, pomoći i savjetima prilikom izrade rada. Daniel Bele pokazao je nenadmašiv entuzijazam na vježbama i predavanjima tijekom svih triju godina, bez obzira na to je li bilo osam sati ujutro ili osam sati navečer, s iskrenom željom da nas nadahne i potakne našu znatiželju za dubljim razumijevanjem materije.

Sažetak

Ovim završnim radom prikazan je razvoj programskoga rješenja za pohranu, organizaciju i upravljanje datotekama i mapama u oblaku. U središtu je pozornosti napredno korisničko sučelje koje rješava nedostatke postojećih rješenja, kao što su Dropbox i Microsoft OneDrive. Pri razvoju rješenja obuhvaćen je niz najnovijih tehnologija i web standarda. Struktura rješenja ima tri sloja. Na klijentskom sloju je web aplikacija u Vue.js programskom okviru u TypeScript jeziku i Android aplikacija u Kotlin jeziku. Na poslužiteljskom sloju jest Node.js aplikacija u TypeScript jeziku. Na podatkovnom sloju su tri različite vrste baze podataka: baza binarnih objekata, nerelacijska baza JSON objekata i nerelacijska baza *Document* objekata. U svim slojevima koristi se autentikacija korisnika. Glavni je fokus rada na implementaciji klijentskog sloja web aplikacije. Iako je web aplikacija u potpunosti prilagođena za korištenje na mobilnim uređajima (svih veličina, orijentacija i na dodir), izrađena je i znatno jednostavnija Android mobilna aplikacija kako bi se usporedile tehnologije. Važan dio programskog rješenja čini funkcionalnost PWA (*Progressive Web App*). Ono omogućuje instalaciju aplikacije na svim platformama s modernim preglednicima, pružajući napredne funkcionalnosti kao što je pristup datotečnom sustavu korisnika za lakše učitavanje i preuzimanje strukturiranih mapa.

Ključne riječi: Web, Vue.js, Firebase, PWA, Android, pohrana u oblaku

Abstract

This thesis presents the development of a software solution for storing, organizing and managing files and folders in the cloud. The focus is on an advanced user interface that addresses the shortcomings of existing solutions, such as Dropbox and Microsoft OneDrive. The development of the solution encompasses a range of the latest technologies and web standards. The structure of the solution consists of three layers. The client layer includes a web application built using the Vue.js framework in TypeScript and an Android application developed in Kotlin. The server layer features a Node.js application written in TypeScript. The data layer includes three different types of databases: a binary object database, a non-relational JSON object database, and a non-relational *document* database. User authentication is implemented across all layers. The main focus of the thesis is on the implementation of the client layer for the web application. Although the web application is fully optimized for use on mobile devices (of all sizes, orientations, and touch-enabled), a significantly simpler Android mobile application was also developed to compare the technologies. An important part of the software solution is the PWA (*Progressive Web App*) functionality. This allows the application to be installed on all platforms with modern browsers, offering advanced features such as access to the user's file system for easier uploading and downloading of structured folders.

Key words: Web**,** Vue.js, Firebase, PWA, Android, Cloud Storage

Sadržaj

1. Uvod

S obzirom na sveprisutnost digitalnih podataka i rastuće zahtjeve za dostupnošću, učinkovita pohrana i upravljanje datotekama u oblaku postali su neizbježan dio svakodnevnoga života i poslovanja. Iako su servisi poput Dropbox-a i Microsoft OneDrive-a široko prihvaćeni, njihova sučelja ne prate najbolje dinamičnost modernih korisničkih potreba.

Unatoč njihovoj popularnosti, sučelja trenutnih servisa na webu ne uspijevaju ponuditi dovoljno fleksibilnosti, što otežava navigaciju i smanjuje produktivnost, osobito pri radu s velikim količinama podataka. Nedostatak opcija za brzo i intuitivno prebacivanje između različitih dijelova strukture mapa te ograničena funkcionalnost stabla mapa dodatno kompliciraju svakodnevne zadatke korisnika.

Ovaj rad opisuje razvoj inovativnoga rješenja koje će poboljšati korisničko iskustvo u oblaku kroz napredno sučelje dizajnirano za brzo i jednostavno organiziranje te upravljanje datotekama i mapama. Detaljnom analizom problema postojećih sučelja, ovaj će rad identificirati ključne izazove i ponuditi konkretna rješenja.

Cilj je stvoriti rješenje koje će ne samo prevladati ograničenja postojećih rješenja, već i redefinirati način na koji korisnici pristupaju organizaciji podataka.

Rad će obuhvatiti detaljnu analizu arhitekture sustava, s posebnim naglaskom na web aplikaciju klijentskog sloja. Bit će istražene tehnologije poput Vue.js programskog okvira, TypeScript jezika i Firebase servisa, a primarni izvori rada bit će službene dokumentacije Google-a i Vue.js tima. S druge strane, Android aplikacija pokazat će kako se mogu postići slične funkcionalnosti na mobilnoj platformi koristeći se programskim jezikom Kotlin, čak i uz integraciju naprednih značajki, poput razgovora s umjetnom inteligencijom. Također će biti razmotreni aspekti skalabilnosti, sigurnosti i performansi, osiguravajući da sustav može podržati rastući broj korisnika i količinu podataka.

Nakon opisa arhitekture, slijedi implementacija klijentskog, poslužiteljskog i podatkovnog sloja sustava. Primjenom progresivnih web aplikacija (PWA) rješenje će postići brže učitavanje, ujednačeno iskustvo na svim uređajima te omogućiti napredne funkcionalnosti. Zatim će biti uspostavljen sustav kontinuirane integracije i isporuke, a rad će završiti testiranjem i analizom korisničkog iskustva putem ankete.

2. Problem postojećih sučelja pohrane u oblaku

U ovom poglavlju opisuje se identifikacija izazova postojećih rješenja, rješenje problema savladavanjem izazova i usporedba s postojećim rješenjima uz pomoć tablice prednosti i nedostataka. Poglavlje završava isticanjem prednosti PWA rješenja u odnosu na tradicionalne pristupe.

2.1. Identifikacija izazova

Ključni izazov postojećih rješenja (Dropbox, Google Drive, Microsoft OneDrive, Mega i sl.) jest nedostatak kartičnoga prikaza s jasno definiranom putanjom na web klijentu, što je značajna prepreka za korisnike koji trebaju brzu i jednostavnu navigaciju kroz složene strukture mapa. Stablo mapa, kao jedan od osnovnih elemenata organizacije datoteka, kod postojećih rješenja često je ograničeno u svojim mogućnostima, što dodatno otežava rad s velikim količinama podataka.

Nadalje, web aplikacije postojećih servisa često pate od sporog učitavanja, osobito kod prvog učitavanja. Razlog tomu jest što servisi postaju preopterećeni zbog složenosti i količine opcija koje nude ili nisu dovoljno optimizirani. Dodatno, često nije podržana sinkronizacija podataka u stvarnom vremenu pa je potrebno osvježavati aplikaciju iznova.

Nedostatak podrške progresivnih web aplikacija (PWA) također predstavlja ograničenje, jer PWA omogućuje veću fleksibilnost i pristupačnost, posebno na različitim platformama i uređajima.

Velik problem predstavlja i vrlo ograničena mogućnost personalizacije izgleda i funkcionalnosti sučelja. Iako se cijeni konzistentnost u dizajnu, nedostatak opcija za prilagodbu može rezultirati iskustvom koje ne odgovara individualnim potrebama i preferencijama korisnika.

2.2. Rješenje problema

Kako bi se prevladali izazovi postojećih servisa za pohranu u oblaku, rješenje ovoga rada uvest će kartični prikaz s putanjama i napredno stablo mapa, omogućujući korisnicima bržu i intuitivniju navigaciju kroz datoteke i mape. Sučelje će biti optimizirano za brzinu, eliminirajući dugo učitavanje stranica, te će podržavati sinkronizaciju svih podataka i

postavki u stvarnom vremenu i najnoviju PWA tehnologiju, osiguravajući dosljedno korisničko iskustvo na svim platformama.

Dodatno, rješenje će ponuditi opsežne mogućnosti personalizacije izgleda, omogućujući korisnicima da prilagode sučelje svojim potrebama i željama, čime će se značajno povećati zadovoljstvo i produktivnost u radu s podacima u oblaku.

2.3. Usporedba s postojećim rješenjima

Istraživanjem i testiranjem postojećih servisa za pohranu u oblaku identificirani su ključni nedostaci i ograničenja koja utječu na korisničko iskustvo. Ta će se usporedba fokusirati na ključne aspekte web aplikacija na web pregledniku stolnog računala i na mobilnom web pregledniku, pružajući jasan pregled konkurentnosti rješenja. U tablici koja slijedi (Slika 2.1) kvačica označava postojanje značajke, dok znak X označava da je nema.

Značajka	**Predloženo rješenje**		Google Drive		Microsoft OneDrive		Dropbox		Mega	
	Desktop	***Mobile***	*Desktop*	*Mobile*	*Desktop*	*Mobile*	*Desktop*	*Mobile*	*Desktop*	*Mobile*
Kartični prikaz	✓		✗		✗		✗		✗	
Stablo mapa	**Vrlo napredno**		Napredno	✗	✗		Osnovno	✗	Napredno	✗
Navigacijska traka	**Vrlo napredna**		Napredna	Loša	Napredna	Osnovna	Napredna	Osnovna	Osnovna	Loša
Pretraga	**Napredna**		Vrlo napredna	Loša	Napredna		Napredna		Napredna	✗
Personalizacija	**Vrlo visoka**		Visoka	✗	Niska		Visoka		Srednja	
Učitavanje aplikacije	**Vrlo brzo**		Brzo	Vrlo brzo	Brzo		Sporo		Vrlo sporo	
Sinkronizacija uživo	✓		✗		✗		✓		✓	
PWA podrška	✓		✓		✓		✗		✗	

Slika 2.1 Usporedba s postojećim rješenjima

Iz tablice je vidljivo da predloženo rješenje nudi niz ključnih prednosti u odnosu na postojeća rješenja. Kartični prikaz, stablo mapa i navigacijska traka omogućuju bolju navigaciju, organizaciju i upravljanje podacima. Pretraga podataka konkurira drugim servisima, a razina personalizacije znatno je opširnija. Sinkronizacija podataka uživo (u stvarnom vremenu) također nadmašuje druga rješenja jer su svi dijelovi aplikacije sinkronizirani, čak i rezultati pretrage i sve postavke.

2.4. Prednosti PWA rješenja u odnosu na tradicionalne pristupe

Progresivne web aplikacije (engl. *Progressive Web App*, skraćeno PWA) nude brojne prednosti u odnosu na tradicionalne web i *desktop* aplikacije, poboljšavajući korisničko iskustvo:

1. **Brže učitavanje i pristup bez mreže.** PWA koristi predmemoriranje (engl. *caching*) za brzo učitavanje sadržaja i omogućuje pristup aplikaciji čak i bez internetske povezanosti. Time se smanjuje vrijeme čekanja i omogućuje neprekidan rad, što je korisno u uvjetima s nestabilnom mrežom.
2. **Ujednačeno iskustvo na svim uređajima.** PWA omogućuje dosljedno korisničko iskustvo preko različitih platformi – mobilnih uređaja, tableta, laptopa i stolnih računala – zahvaljujući standardiziranim web tehnologijama. Ta dosljednost osigurava da aplikacija funkcionira kao nativna aplikacija bez obzira na vrstu uređaja.
3. **Instalacija i pristup.** PWA aplikacije mogu se instalirati izravno s preglednika na stolno računala ili mobilne uređaje, bez potrebe za distribucijom putem aplikacijskih trgovina. Na taj način eliminiraju se troškovi i komplikacije povezane s objavljivanjem na platformama poput Apple App Store-a i Google Play Store-a, što je osobito korisno za manje tvrtke i neovisne programere.
4. **Jedinstvena baza kôda.** Za programere, jedna od ključnih prednosti PWA jest mogućnost korištenja istoga koda za različite platforme. To pojednostavljuje razvoj i održavanje aplikacije, smanjujući vrijeme i troškove povezane s razvojem različitih verzija za mobilne i *desktop* platforme.

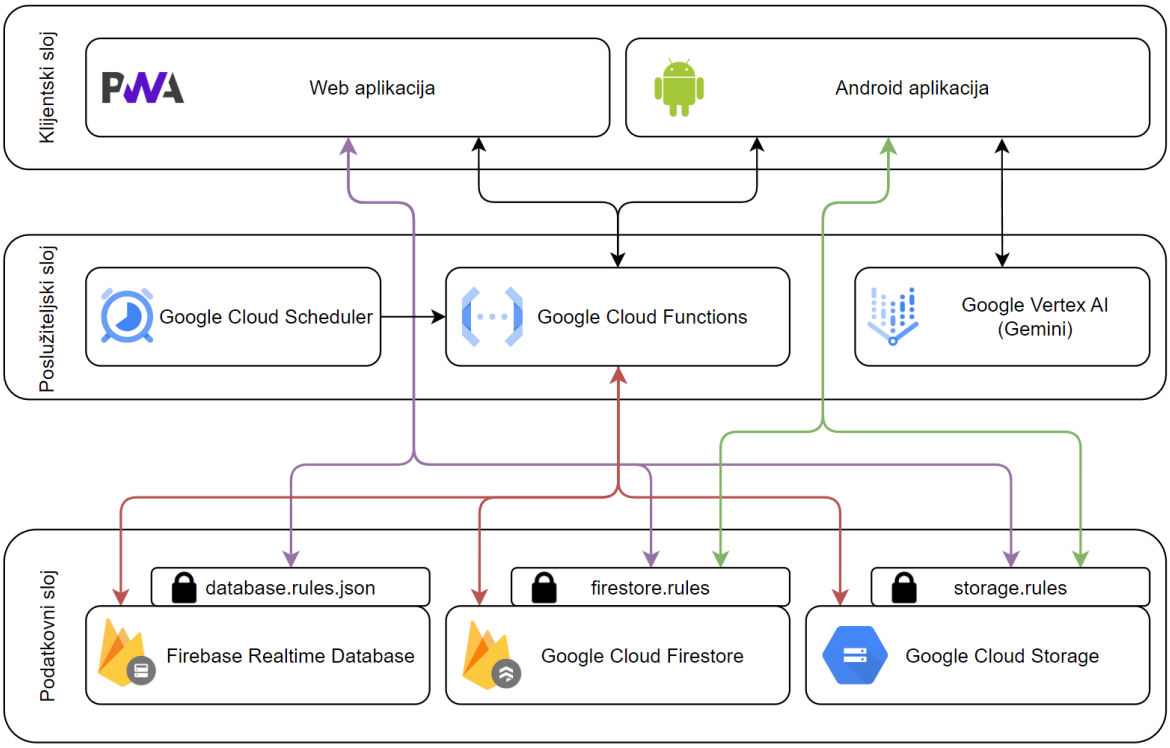
PWA rješenja ne samo da nude poboljšanu brzinu i dostupnost već i veću fleksibilnost i pristupačnost za razvoj i distribuciju aplikacija, čime značajno unapređuju iskustvo korisnika i olakšavaju rad programerima. [1]

3. Arhitektura sustava

Ovo poglavlje sadrži shemu, opis sustava i pojedinačnih slojeva te proučava aspekte skalabilnosti, sigurnosti i performansi.

3.1. Shema i opis sustava

Sustav se sastoji od triju slojeva: klijentski, poslužiteljski i podatkovni (Slika 3.1).



Slika 3.1 Shema sustava

Klijentski sloj sadrži web aplikaciju, koja je primaran fokus ovoga rada, ali i nativnu mobilnu aplikaciju za Android uređaje. Svi dijelovi poslužiteljskog i podatkovnog sloja dio su Google Cloud platforme (GCP). Poslužiteljski sloj sadrži Cloud Functions servis za važne operacije nad bazama podataka, Cloud Scheduler za periodično izvršavanje funkcija i Vertex AI za razgovor s umjetnom inteligencijom. Podatkovni sloj sadrži tri različite vrste baze podataka, ovisno o vrsti podataka koji se spremaju, a to su: Firebase Realtime Database, Cloud Firestore i Cloud Storage. U svim slojevima koristi se Firebase Authentication servis za autentikaciju korisnika.

3.1.1. Klijentski sloj

Klijentski sloj sadrži web aplikaciju i nativnu Android aplikaciju.

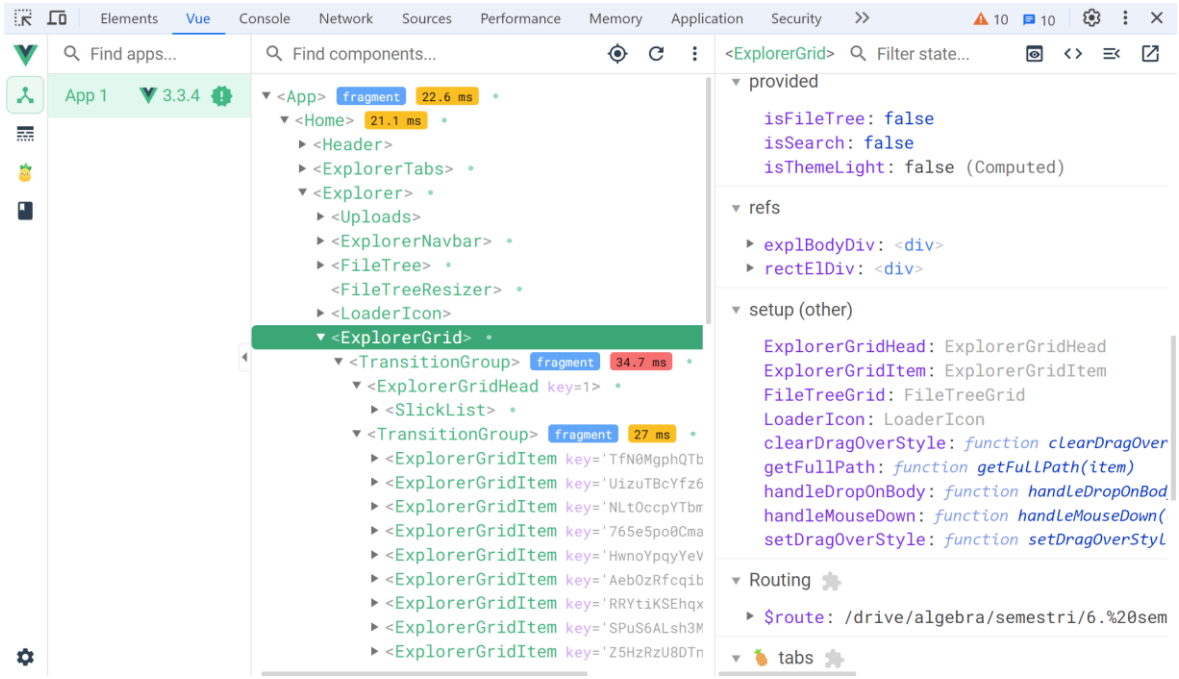
Web aplikacija izrađena je u Vue.js programskom okviru (engl. *framework*) za JavaScript jezik (u daljnjem tekstu: Vue). Umjesto JavaScript-a koristi se TypeScript jezik – nadskup (engl. *superset*) JavaScript-a, za koji Vue ima odličnu podršku. [2]

Arhitektura web aplikacije jest SPA (S*ingle Page Application*). SPA omogućuje korištenje aplikacije bez učitavanja potpuno novoga HTML DOM-a (*Document Object Model*) s poslužitelja, što rezultira poboljšanjem performansi i dinamičnijim iskustvom, uz neke nedostatke kompromisa kao što je potencijalno lošiji SEO (*Search Engine Optimization*), više truda potrebna za održavanje stanja, implementaciju navigacije i praćenje putanja korisnika za analitiku. [3]

Renderiranje web aplikacije jest CSR (*Client Side Rendering*), a budući da nema asinkronih Vue komponenata, preglednik preuzima samo jedan izgrađeni JavaScript paket (engl. *bundle*) pri prvom otvaranju. To znači da se HTML generira na strani klijenta izvršavanjem JavaScript kôda u pregledniku. JavaScript kôd izvršava motor (engl. *engine*) JavaScript-a (npr. Chrome V8, Nitro, SpiderMonkey itd.). To renderiranje ne treba miješati s renderiranjem web stranice, koje izvodi motor preglednika (npr. Webkit, Blink, Gecko itd.) kad primi HTML datoteku i izgradi "stablo renderiranja" iz DOM-a i CSS OM-a, a zatim "iscrtava" web stranicu. [4] CSR je odabran zbog maksimalne brzine nakon prvog učitavanja, zato što pristup zahtjeva autentikaciju i zato što u ovome slučaju nije bitan SEO. No u slučaju komercijalizacije, lako je izraditi prezentacijsku stranicu koja bi predstavljala ovu aplikaciju na poddomeni i tada bi SEO bio bitan te se preporučuje korištenje SSR-a (*Server Side Rendering*) ili statičnoga generatora stranica kao što je Nuxt.

Arhitekturni obrazac web aplikacije jest MVVM (*Model View ViewModel*), što je standard za aplikacije s Vue programskim okvirom i opisan je u implementaciji kroz primjer Vue komponente (Poglavlje 4.1.4.3).

Alati za pomoć u izradi web aplikacije su: uređivač kôda Visual Studio Code (uključujući niz dodatno instaliranih proširenja za uređivač), Chrome DevTools, Chrome proširenje Vue.js DevTools (Slika 3.2**Error! Reference source not found.**), Vue SFC Playground web aplikacija, Git, Firebase CLI (*Command Line Interface*) i Google Cloud CLI.



Slika 3.2 Chrome proširenje Vue.js DevTools kao dio Chrome DevTools alata

Android aplikacija izrađena je u jeziku Kotlin u Android Studio IDE-u (*Integrated Development Environment*).

Arhitekturni obrazac Android aplikacije jest MVC (*Model View Controller*). Obrazac razdvaja aplikaciju na tri komponente: *Model* (upravljanje podacima), *View* (pogled, prikaz korisničkog sučelja) i *Controller* (posrednik između modela i prikaza, upravlja ulazima korisnika). Važno je napomenuti da zbog jednostavnosti aplikacije ulogu pogleda i kontrolera imaju aktivnosti i fragmenti na način da pozivaju ručno izrađene globalne Firebase funkcije. U slučaju rasta aplikacije, kako bi se izbjeglo čvrsto spajanje (engl. *tight coupling*), lako je izraditi *ViewModel* klase, čime bi arhitekturni obrazac prešao na MVVM. *ViewModel* upravlja i održava podatke potrebne za prikaz. Time se bolje odvajaju odgovornosti, pojednostavljuje poslovna logika kontrolera, omogućuje jednostavnije testiranje i povećava skalabilnost aplikacije.

3.1.2. Poslužiteljski sloj

Poslužiteljski sloj sadrži Cloud Functions, Cloud Scheduler i Vertex AI.

Cloud Functions je *serverless* okvir koji omogućuje automatsko pokretanje *back-end* kôda kao odgovor na događaje pokrenute pozadinskim događajima, HTTPS (*Hypertext transfer protocol secure*) zahtjevima ili zadacima Cloud Scheduler servisa. [5] *Serverless* je način

pružanja pozadinskih usluga (*back-end* funkcija), gdje se naplaćuje korištenje prema stvarnoj potrošnji.

Cloud Scheduler je servis za zakazivanje poslova koji omogućuje automatizaciju različitih zadataka, od *batch* i *big data* poslova do operacija u *cloud* infrastrukturi. Nudi pouzdano izvršavanje, ponovne pokušaje u slučaju grešaka i centralizirano upravljanje svim automatiziranim zadacima. [6]

Glavna svrha ovoga sloja jest provedba dugotrajnih i važnih događaja koji se moraju izvršiti do kraja (u potpunosti), kao što je brisanje korisničkog računa. Dodatno, ovaj sloj treba izvoditi periodične funkcije za brisanje korisničkih podataka nakon što su bili u smeću 30 dana, za što je Cloud Functions i Cloud Scheduler idealno rješenje.

Vertex AI je platforma za strojno učenje koja ima API (*Application Programming Interface*) za korištenje već treniranih generativnih multimodalnih AI (*Artificial Intelligence*) modela [7]. Koristi ga Android aplikacija za razgovor između korisnika i umjetne inteligencije s tekstualnim odgovorima.

3.1.3. Podatkovni sloj

Podatkovni sloj sadrži tri različite vrste baze podataka: Firebase Realtime Database, Cloud Firestore i Cloud Storage. Svaka od njih ima specifične karakteristike i namjene, ovisno o vrsti podataka koji se spremaju:

1. Firebase Realtime Database (u daljnjem tekstu: RTDB) jest NoSQL baza podataka koja pohranjuje podatke u obliku JSON strukture. Njezina je ključna značajka mogućnost sinkronizacije podataka u stvarnom vremenu, što omogućuje trenutnu distribuciju promjena podataka među svim povezanim klijentima. Ta je baza idealna za aplikacije gdje je potrebno kontinuirano praćenje promjena ili gdje je važna brza reakcija na promjene podataka. Podaci se spremaju na jednom globalnom stablu, što može biti prednost za jednostavnije strukture, ali može otežati upravljanje složenijim podacima. [8] Zbog navedenih razloga, ova baza upotrebljava se za postavke korisnika.
2. Cloud Firestore također je NoSQL baza podataka, ali je razvijena kao naprednija i skalabilnija verzija RTDB-a. Podaci se pohranjuju u obliku kolekcija i dokumenata, što omogućuje fleksibilnije strukturiranje podataka. Firestore podržava naprednije upite, indeksiranje i filtriranje podataka te je dizajniran da bolje podrži složene

aplikacije s većim opsegom podataka. Poput RTDB-a, Firestore nudi sinkronizaciju podataka u stvarnom vremenu, ali uz veću pouzdanost i skalabilnost. Ta je baza podataka posebno prikladna za aplikacije koje zahtijevaju strukturirano i hijerarhijski organizirano spremanje podataka, s naglaskom na fleksibilnost i prilagodljivost upita. [9] Zbog navedenih razloga, ova baza rabi se za strukturu datoteka i mapa korisnika.

3. Cloud Storage služi za pohranu binarnih podataka, odnosno bilo kojih datoteka. Namijenjen je za skalabilnu i sigurnu pohranu te dohvaćanje datoteka. S obzirom na to da pruža visoku dostupnost i brzinu pristupa te se odlično integrira s drugim bazama i klijentskim ovisnostima [10], ova baza rabi se za sve što korisnici učitaju sa svojih uređaja, odnosno podatke korisnika.

Termin NoSQL, skraćenica od "*not only SQL*" (ne samo SQL (*Structured Query Language*)), odnosi se na nerelacijske baze podataka koje za pohranu podataka upotrebljavaju netabularni format. NoSQL baze podataka rabe fleksibilni model sheme koji podržava širok raspon nestrukturiranih podataka. [11]

Web aplikacija koristi se svim bazama, a Android aplikacija svima osim RTDB-a jer nema mogućnost odabira postavki.

3.1.4. Autentikacija

Za autentikaciju se upotrebljava servis Firebase Authentication.

Firebase Authentication čvrsto se integrira s drugim Firebase uslugama i upotrebljava industrijske standarde kao što su OAuth 2.0 i OpenID Connect, tako da se može jednostavno integrirati s bilo kojim *backend-om*. [12]

Koriste se tri različite vrste autentikacije:

1. Autentikacija putem *e-maila* i lozinke. Firebase Authentication SDK (*Software Development Kit*) pruža metode za stvaranje i upravljanje korisnicima koji rabe svoje adrese e-pošte i lozinke za prijavu. Firebase Authentication također upravlja slanjem e-poruka za ponovno postavljanje lozinke.
2. Autentikacija putem integracije federalnog pružatelja identiteta. Firebase Authentication SDK pruža metode koje korisnicima omogućuju prijavu sa svojim Google, Twitter/X, GitHub, Microsoft i Yahoo računima. Za svaku od navedenih platformi potrebno je imati otvoren *developer* račun kako bi se registrirala aplikacija

u njihovu sustavu zajedno uz dopuštenja koja će tražiti. Na primjer, za Twitter/X dovoljno je tražiti korisničko ime i *email*, a ne i objave.

3. Autentikacija putem telefonskog broja na način da Firebase šalje SMS poruku na korisnikov uređaj.

Korisnik se nakon odjave ili na drugom uređaju može prijaviti na isti račun drugom metodom, ako je korišten isti *e-mail*. To je omogućeno kroz Firebase *Account Linking*.

3.2. Skalabilnost, sigurnost i performanse

Budući da je rad na GCP infrastrukturi, aplikacije se mogu lako prilagoditi rastućem broju korisnika i količini podataka. Sve baze podataka automatski se skaliraju, osiguravajući besprijekorno korisničko iskustvo čak i pri velikom opterećenju.

Firebase Authentication pruža sigurnu prijavu i registraciju korisnika, dok sigurnosna pravila svih baza podataka omogućuju preciznu kontrolu pristupa podacima.

Firebase Hosting, u kombinaciji s globalnom CDN (*Content Delivery Network)* mrežom, omogućuje brzo učitavanje web aplikacije diljem svijeta i automatski osigurava SSL (*Secure Sockets Layer*) certifikate. [13] Kada se uspostavlja HTTPS veza, web preglednik provjerava SSL certifikat poslužitelja kako bi se uvjerio da je legitiman. Ako je certifikat valjan, preglednik i poslužitelj uspostavljaju sigurnu, šifriranu vezu koristeći TLS (*Transport Layer Security*) protokol. To znači da su svi podaci koji se razmjenjuju između preglednika i poslužitelja zaštićeni od prisluškivanja i neovlaštenog pristupa.

S klijentske strane, performanse su jedna od prednosti ovoga projekta u usporedbi s postojećim rješenjima. Potrebno je paziti na veličinu konačnog JavaScript *bundle-a* upotrebljavajući *Tree-shaking, Code splitting* te odabir *bundler-a,* minifikatora i njihovih opcija koje omogućuju optimizaciju i minimizaciju. Nakon što je aplikacija učitana, radit će brzo, s ugodnim tranzicijama i animacijama, na način da sve "teče glatko". Vue je jedan od najbržih okvira, brži od poznatijih React i Angular okvira. Nadalje, rabe se najbolje Vue prakse, kao što su *Props stability* i *Computed stability* [14], ali i generalne optimizacije kao što je čuvanje DOM čvorova (engl. *nodova*) u što manjem broju te odabir brzih CSS prikaza (engl. *display*) i iskorištavanje CSS optimizacijskih svojstava. [15]

S obzirom na to da su funkcionalnosti Android aplikacije trivijalne u usporedbi s web aplikacijom, može se očekivati da neće imati problem s performansama.

4. Implementacija sustava

U ovom poglavlju objašnjena je implementacija slojeva sustava, s glavnim fokusom na web aplikaciju klijentskog sloja.

4.1. Klijentski sloj

U ovome poglavlju sva potpoglavlja započinju opisom web aplikacije. Ako postoji Android implementacija, zasebno je opisana na kraju.

Web aplikacija koristi NPM (*Node Package Manager*) za upravljanje paketima (ili ovisnostima, engl. *dependencies*) i izgradnju projekta, dok Android aplikacija koristi Gradle za istu svrhu.

NPM koristi *package.json* datoteku za upravljanje paketima, dok Gradle koristi *build.gradle* datoteke za istu svrhu. Na temelju analize dokumentacije NPM-a [16] i Gradle-a [17], može se zaključiti da Gradle nudi više mogućnosti za prilagodbu procesa izgradnje aplikacija, dok je NPM fokusiran na jednostavno rukovanje JavaScript/TypeScript paketima.

Budući da su to glavne datoteke koje opisuju projekte u različitim okruženjima, opisani su njihovi ključni dijelovi, s naglaskom na web aplikaciju jer je ona fokus ovoga rada.

```
...
"dependencies": {
  "@vueuse/core": "^10.7.0",
  "file-icon-vectors": "^1.0.0",
  "firebase": "^10.5.2",
  "firebaseui": "^6.1.0",
  "pinia": "^2.1.7",
  "uuid": "^9.0.1",
  "v-wave": "^1.5.1",
  "vue": "^3.3.4",
  "vue-router": "^4.2.5",
  "vue-slicksort": "^2.0.5",
  "vuefire": "^3.1.17"
},
"devDependencies": {
  "@playwright/test": "^1.42.1",
  "@types/node": "^20.8.7",
```

```
    "@types/uuid": "^9.0.7",
    "@vitejs/plugin-vue": "^4.2.3",
    "autoprefixer": "^10.4.16",
    "daisyui": "^4.4.7",
    "postcss": "^8.4.31",
    "prettier": "^3.0.3",
    "prettier-plugin-tailwindcss": "^0.5.7",
    "rollup-plugin-visualizer": "^5.12.0",
    "tailwindcss": "^3.3.3",
    "terser": "^5.26.0",
    "typescript": "^5.0.2",
    "vite": "^4.4.5",
    "vue-tsc": "^1.8.5"
  }
}
```

Kôd 4.1 Ovisnosti u *package.json* datoteci web aplikacije

Ovisnosti (*dependencies*) sadrže pakete koje ulaze u produkcijski kôd, dok razvojne ovisnosti (*devDependencies*) sadrže pakete za razvoj produkcijskog kôda (Kôd 4.1). Na primjer, preglednici razumiju samo JavaScript i zato TypeScript ne ulazi u ovisnosti, nego pomaže u procesu razvoja. U kontekstu NPM paketa i semantičkog verzioniranja, znak ^ (engl. *caret*) ispred verzija označava fleksibilni raspon verzija, dopuštajući NPM-u da instalira ili ažurira paket na najnoviju kompatibilnu verziju, koja je veća ili jednaka specificiranoj verziji, ali manja od sljedeće glavne verzije. [16]

Opisi ovisnosti web aplikacije su:

- Paket "@vueuse/core" jest kolekcija uslužnih (engl. *utility*) funkcija temeljena na Vue *Composition* API-ju. [18] Koristi se za, na primjer, geste prijelaza (engl. *swipe*).
- Paket "file-icon-vectors" sadrži SVG (*Scalable Vector Graphics*) ikone za tip/ekstenziju datoteke. [19] SVG ikone iz toga paketa kopirane su u Android aplikaciju putem menadžera za resurse (engl. *Resource Manager*) Android Studio platforme, koji je pretvorio ikone u XML (*Extensible Markup Language*) format i spremio u *drawable* mapu.
- Paket "firebase" jest JavaScript SDK za servise koje Firebase nudi, kao što su baze podataka, dok paket "firebaseui" pruža jednostavno i prilagodljivo korisničko sučelje za autentikaciju povrh Firebase SDK-a kako bi se uklonio standardni, ponavljajući kôd i promovirale najbolje prakse. [20]

- Paket "pinia" služi za upravljanje stanjem u Vue aplikacijama i olakšava dijeljenje podataka (engl. *state*) između komponenti i stranica, nudi podršku za testiranje, dodatke (engl. *plugins*), odličnu TypeScript podršku i alate za razvoj. [21]
- Paket "uuid" upotrebljava se za izradu UUID-a (*Universally Unique Identifier*), a potreban je za spremanje jedinstvenih ID-eva na entitetima RTDB servisa.
- Paket "v-wave" je direktiva za Vue koja omogućuje vizualni efekt valovitosti (engl. *ripple*), na primjer, klikom na gumb kako bi korisnik uočio trenutak klika.
- Paket "vue" je progresivni JavaScript okvir za izradu korisničkih sučelja. Njegova fleksibilnost i modularnost omogućuju izgradnju svega, od jednostavnih komponenti do složenih aplikacija. Vue se temelji na dva ključna koncepta: deklarativno renderiranje i reaktivnost. Deklarativno renderiranje omogućuje definiranje HTML izlaza na temelju JavaScript stanja, koristeći proširenu sintaksu koja pojednostavljuje upravljanje DOM-om. Vue-ova reaktivnost automatski prati promjene u JavaScript stanju i ažurira DOM prema potrebi. [22] Virtualni DOM (VDOM) je programski koncept u kojem se idealna, ili "virtualna", reprezentacija korisničkog sučelja čuva u memoriji i sinkronizira sa "stvarnim" DOM-om. Ovaj koncept je prvi put predstavljen od strane React programskog okvira, a kasnije su ga prihvatili mnogi drugi okviri s različitim implementacijama, uključujući Vue. [23] U srži Vue-a je koncept komponenti. Komponenta je samostalni dio korisničkog sučelja, poput gumba, liste ili obrasca za prijavu. Vue SFC (*Single File Component*), datotečnog nastavka "vue", omogućuje enkapsulaciju predložka (engl. *template*), logike i stiliziranja Vue komponente u jednoj datoteci. Vue SFC je prirodno proširenje klasičnog trija HTML-a, CSS-a i JavaScript-a. Drugim riječima, blokovi `<template>`, `<script>` i `<style>` enkapsuliraju i smještaju prikaz, logiku i stiliziranje komponente u istu datoteku. [24] Vue komponente mogu se pisati u dva različita API stila: *Options* API i *Composition* API. S obzirom na veliku složenost i opseg funkcionalnosti (engl. *feature-rich*) web aplikacije, Vue *Composition* API idealan je izbor zbog nekoliko ključnih razloga, koji se temelje na često postavljenim pitanjima u službenoj dokumentaciji [25]:
    1. Omogućuje daleko bolju organizaciju kôda u odnosu na tradicionalni *Options* API, osobito kad je riječ o velikim komponentama. Mogućnost grupiranja povezane logike, umjesto da je razbacana po različitim opcijama

komponente, znatno olakšava razumijevanje, navigaciju i refaktoriranje koda, što je ključno za održivost projekta na duge staze.

2. Vrlo je koristan u ponovnoj upotrebi logike kroz stvaranje sastavljivih (engl. *composable*) funkcija. Time se rješavaju nedostaci *mixin*-a, tradicionalnog mehanizma za ponovnu upotrebu u *Options* API-ju, i omogućuje izradu čistih, višekratno upotrebljivih dijelova logike. Također, ovo otvara vrata integraciji vanjskih paketa i servisa na elegantan i reaktivan način.
3. Slaže se izvrsno s TypeScript-om, pružajući vrhunsku podršku za tipiziranje koja je znatno bolja od one koju nudi *Options* API. To rezultira robusnijim kodom, većim povjerenjem prilikom izmjena i puno ugodnijem iskustvu razvoja.

Ukratko, Vue *Composition* API pruža fleksibilnost, skalabilnost i održivost potrebne za izgradnju i održavanje ove web aplikacije, osiguravajući da kod ostane čist, organiziran i lako razumljiv s rastom složenosti.

- Paket "vue-router" službeni je usmjerivač (engl. *router*) za Vue, namijenjen izradi SPA aplikacija s višestrukim prikazima. Omogućuje povezivanje URL-ova (*Uniform Resource Locator*) s komponentama Vue aplikacije, omogućujući korisnicima kretanje kroz aplikaciju bez ponovnog učitavanja stranice. Ključne funkcionalnosti uključuju navigaciju putem definiranih ruta, podršku za dinamičke rute s parametrima, te programsku navigaciju kroz JavaScript. [26]
- Paket "vue-slicksort" sadrži komponente za ručno sortiranje lista. [27]
- Paket "vuefire" rješenje je za stvaranje veza u stvarnom vremenu između Firebase baza i Vue aplikacije. Iako Firebase SDK pruža API za sinkronizaciju lokalnih podataka s promjenama u udaljenoj bazi podataka, taj proces može biti puno zamorniji i uključuje brojne rubne slučajeve (engl. *edge cases*). VueFire olakšava održavanje lokalnih podataka uvijek sinkroniziranim s udaljenim bazama podataka. [28]

Neki NPM paketi započinju s "@" jer su to *scoped* paketi, a simbol označava imenični prostor (engl. *namespace*) za paket, što pomaže u izbjegavanju konflikata imenovanja između različitih paketa na NPM registru. Paketi u "@types" prostoru posebno su rezervirani na NPM-u za pakete koji pružaju TypeScript definicije tipova za JavaScript pakete. Ovaj prostor je kontroliran od strane DefinitelyTyped projekta i tima koji održavaju kvalitetu i konzistentnost definicija tipova.

Opisi razvojnih ovisnosti web aplikacije su:

- Paketi u "@types" prostoru dodaju TypeScript podršku za pakete koji ga nemaju. U idealnom slučaju, svi paketi bi trebali imati ugrađene definicije tipova, ali to nije uvijek slučaj.
- Paket "autoprefixer" automatski dodaje potrebne *vendor* prefikse u CSS kako bi se osigurala kompatibilnost sa što više različitih preglednika. [29]
- Paket "daisyui" je knjižnica komponenata za Tailwind CSS. DaisyUI proširuje Tailwind CSS s kolekcijom gotovih, stiliziranih komponenti kao što su gumbi, prostori za unos teksta, padajući izbornici i mnogi drugi. [30]
- Paket "postcss" transformira i obrađuje CSS pomoću JavaScript dodataka (engl. *plugins*). Daje veliku fleksibilnost i kontrolu nad CSS kodom, omogućujući korištenje naprednih CSS značajki i automatiziranje zadataka. Jedan od najpopularnijih PostCSS dodataka je upravo Autoprefixer, koji se ovdje koristi. [31]
- Paket "prettier" automatski formatira programski kôd na način da bude dosljedan i čitljiv kroz cijeli projekt te podržava brojne programske jezike. Slično PostCSS-u, omogućuje korištenje dodataka, kao što je "prettier-plugin-tailwindcss". [32]
- Paket "prettier-plugin-tailwindcss" je dodatak za Tailwind CSS koji automatski sortira klase prema najboljim Tailwind CSS praksama. [33]
- Paket "rollup-plugin-visualizer" vizualizira konačni JavaScript *bundle*, kojega izgrađuje Rollup, za analizu modula koji zauzimaju memorijsku prostor. Cilj je imati što manji *bundle* kako bi se aplikacija što brže preuzela i zato je ovaj paket odličan za optimizaciju. [34]
- Paket "tailwindcss" skenira sve zadane datoteke i generira odgovarajuće stilove koje upisuje u statičku CSS datoteku. Za razliku od Bootstrap-a, koji koristi unaprijed definirane komponente, Tailwind CSS nudi potpunu kontrolu nad izgledom putem *utility* klasa, omogućujući daleko veću fleksibilnost i prilagodljivost dizajna. Nudi i dodatnu sintaksu, funkcije i direktive, odnosno prilagođena at-pravila (@). [35]
- Paket "terser" jedan je od najjačih JavaScript minifikatora, s boljom kompresijom i dodatnim optimizacijskim opcijama, ali treba više vremena za izgradnju *bundle-a*. Korišten je eksperimentalno te je zaključak da je smanjenje veličine konačnog *bundle-a* zanemarivo i zato se koristi zadani *esbuild*, koji dolazi s Vite paketom.
- Paket "typescript" je programski jezik koji proširuje mogućnosti JavaScript-a dodavanjem statičkog tipiziranja, čime omogućuje provjeru tipova podataka tijekom

kompilacije. Kao *superset* JavaScript-a, TypeScript prihvaća sav ispravan JavaScript kôd, ali uvodi dodatna pravila koja pružaju veću kontrolu nad načinom korištenja različitih vrsta podataka. [36]

- Paket "vite" je munjevito brz skupljač modula (engl. *module bundler*) bogat značajkama. Sastoji se od dva ključna dijela:
    1. Razvojnog poslužitelja koji pruža brojna poboljšanja u odnosu na nativne ES (ECMAScript) module, kao što je iznimno brza zamjena modula uživo (*Hot Module Replacement*, skraćeno HMR).
    2. Naredbe za izradu koja objedinjuje programski kôd pomoću Rollup-a, unaprijed konfiguriranog za generiranje visoko optimiziranih statičkih resursa za produkcijsko okruženje.

  Vite je alat s jasnim stavom (engl. *opinionated*) i dolazi s razumnim zadanim postavkama. Podrška za različite okvire ili integracija s drugim alatima moguća je putem *plugin-ova*. Vite je također visoko proširiv putem svojeg API-ja za dodatke i JavaScript API-ja s potpunom podrškom za TypeScript. [37]
- Paket "vue-tsc" je omotač za TypeScript koji dodatno provjerava projekt prije izrade produkcijskog kôda.

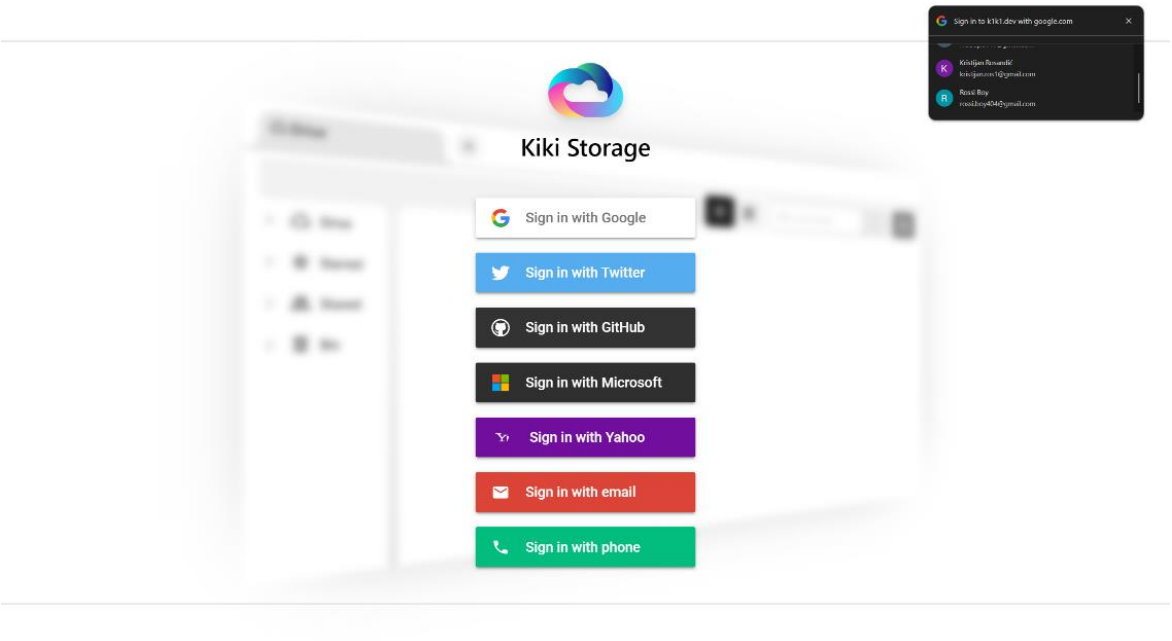
Uz zadane ovisnosti koje dodaje Android Studio pri stvaranju projekta, ovo su kratki opisi ručno dodanih ovisnosti Android aplikacije:

- Ovisnost "com.google.android.gms:play-services-auth" omogućuje autentifikaciju putem Google-ovih usluga unutar Android aplikacija, uključujući prijavu putem Google računa.
- Ovisnosti "firebase-bom", "firebase-auth", "firebase-firestore", " firebase-storage", "firebase-functions" i "firebase-vertexai" grupnog identifikatora "com.google.firebase" koriste se za upravljanje autentikacijom korisnika, pohranom podataka, pozivanjem poslužitelja i primjenom naprednih AI modela putem Firebase platforme, dok BoM (*Bill of Materials*) osigurava kompatibilnost među svim Firebase ovisnostima.
- Ovisnosti "firebase-ui-auth" i "firebase-ui-firestore" grupnog identifikatora "com.firebaseui" proširuju funkcionalnosti Firebase-a, pružajući gotove UI (*User Interface*) komponente za autentifikaciju korisnika te prikaz i upravljanje podacima iz Firestore baze podataka.

- Ovisnost "androidx.preference:preference-ktx" omogućuje lakšu i intuitivniju implementaciju postavki unutar Android aplikacija koristeći Kotlin ekstenzije.
- Ovisnost "androidx.work:work-runtime-ktx" upravlja zadacima koji moraju biti obavljeni čak i kada aplikacija nije aktivna, dok "com.google.code.gson:gson" omogućuje serijalizaciju i deserializaciju Java objekata u JSON format, olakšavajući rad sa složenim podacima unutar tih zadataka.
- Ovisnosti "navigation-fragment-ktx" i "navigation-ui-ktx" grupnog identifikatora "androidx.navigation" pružaju podršku za navigaciju između različitih fragmenata i integraciju navigacijskih elemenata poput izbornika za navigaciju, olakšavajući upravljanje prijelazima između ekrana unutar aplikacije.

4.1.1. Registracija i prijava

Učitavanjem web aplikacije prvi put, zaslon se nalazi na "/login" putanji. Korisnik se može registrirati ili prijaviti bilo kojom od ponuđenih metoda (Slika 4.1) i time ostaje trajno prijavljen dok se ne odjavi ili ne očisti kolačiće preglednika.



Slika 4.1 Opcije prijave na početnom zaslonu web aplikacije

U slučaju bitnih aktivnosti kao što su promjena lozinke ili brisanje korisničkog računa, koje korisnik može pokrenuti, mora se ponovo prijaviti.

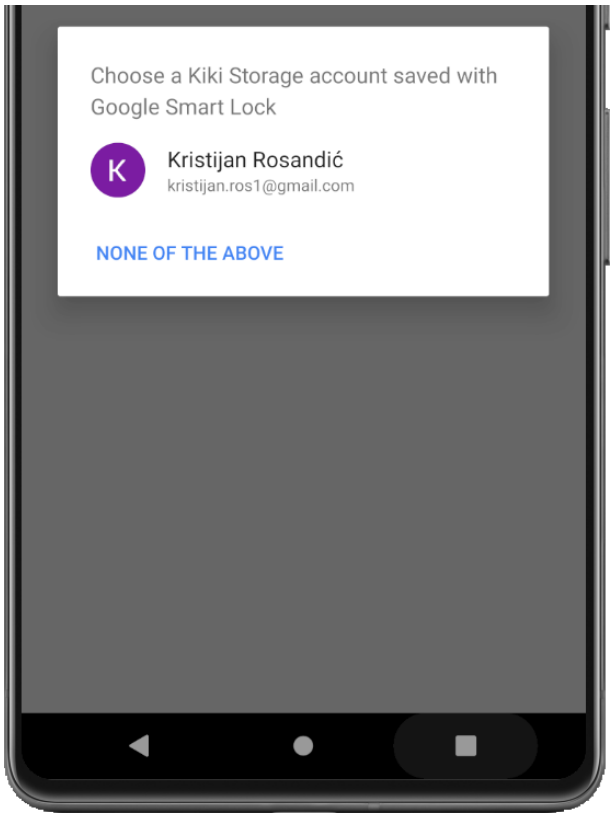
Registracijom putem *e-maila* i lozinke potrebno je unijeti korisničko ime.

Registracijom putem pružatelja identiteta postupak je isti kao i za prijave te je potrebno slijediti upute u skočnom prozoru.

Registracijom putem telefonskog broja postupak je isti kao i za prijave te je potrebno potvrditi da korisnik nije robot.

Ako je korisnik prijavljen u preglednik Google računom, u gornjem desnom kutu postoji *One Tap* prijava, što znači da se korisnik može još brže prijaviti – samo jednim klikom, bez otvaranja skočnog prozora. [38] U slučaju registracije prikazuje se poruka: "*Za nastavak, google.com će podijeliti vaše ime, adresu e-pošte i profilnu sliku s ovom web lokacijom*.".

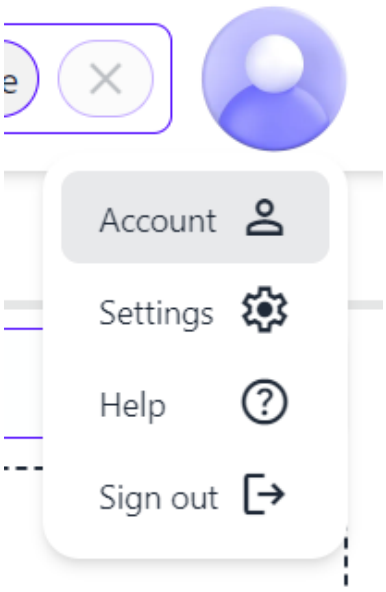
Uz sve ostale metode prijave, *One Tap* prijava implementirana je i na Android aplikaciji (Slika 4.2). Firebase zna za korisničke račune jer komunicira s Google Identity servisima koji upravljaju Google računima na uređaju.



Slika 4.2 *One Tap* prijava na Android aplikaciji

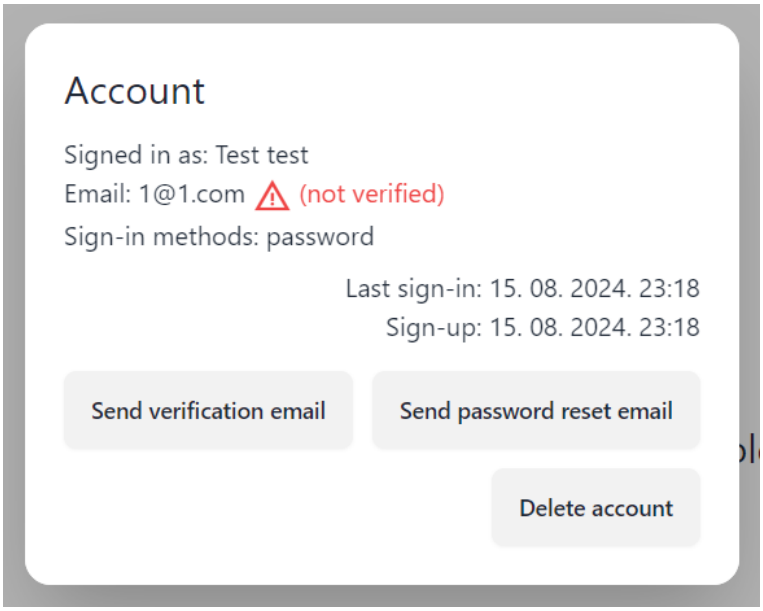
4.1.2. Korisnički račun i postavke

U gornjem desnom kutu je profilna slika korisnika, učitana putem Firebase Authentication servisa, a ako ne postoji, koristi se zadana (engl. *default*) slika (Slika 4.3). Prelaskom kursora preko slike otvara se padajući izbornik s opcijama: račun, postavke, pomoć i odjava.



Slika 4.3 Padajući izbornik iz profilne slike na web aplikaciji

Klikom na korisnički račun otvara se prozor koji dohvaća korisničko ime, *email*, metode prijave koje su izvršene, datum i vrijeme zadnje prijave te datum i vrijeme stvaranja korisničkog računa (Slika 4.4).



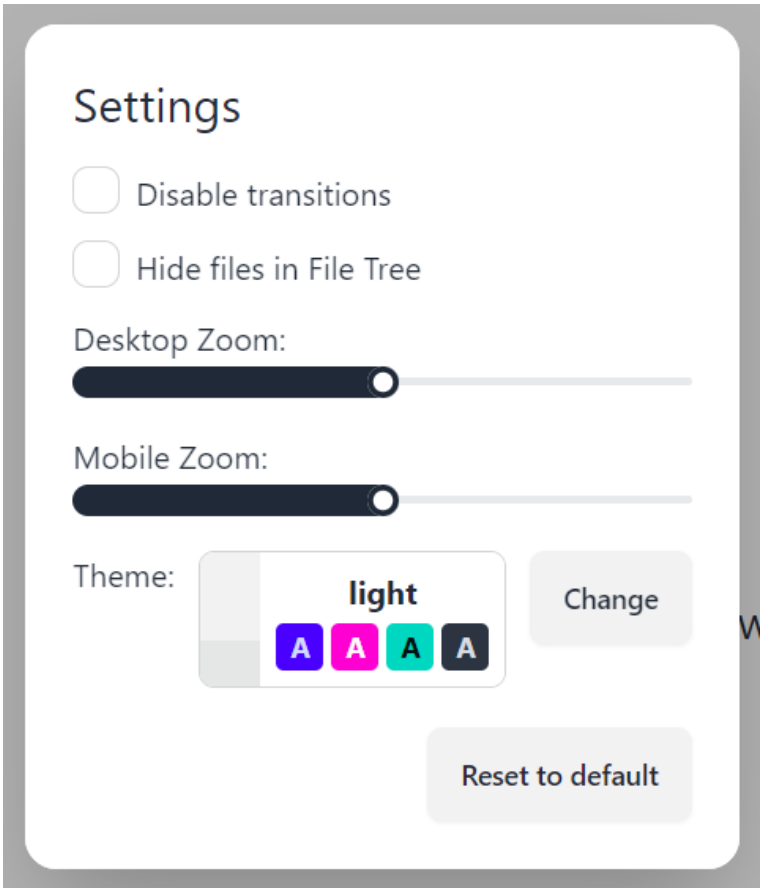
Slika 4.4 Prozor korisničkog računa na web aplikaciji

Ispod navedenih informacija ponuđene su najviše tri opcije:

1. Slanje *e-maila* za potvrdu korisničkog računa. Prikazuje se u slučaju da *e-mail* nije potvrđen. To može biti važno u budućnosti ako se aplikacija komercijalizira i korisnici žele kupiti dodatan prostor za pohranu podataka. Nakon slanja, prikazuje se poruka da je poveznica poslana i korisnik ju mora otvoriti u dobivenoj poruci.

2. Slanje *e-maila* za resetiranje korisničke lozinke. U slučaju da se korisnik nije registrirao s lozinkom, na ovaj način ju može dobiti. Nakon slanja, prikazuje se poruka da je poveznica poslana, putem koje korisnik mora unijeti novu lozinku.
3. Brisanje korisničkog računa. Klikom na gumb otvara se dijaloški skočni prozor s upozorenjem (prev. s eng.): "Ova nepovratna radnja dovest će do trajnog gubitka svih vaših podataka i imajte na umu da neće biti sačuvane sigurnosne kopije". Klikom na potvrdu prolazi jedna sekunda (kako bi se spriječio slučajni dvostruki klik) i opet se otvara dijaloški prozor s porukom (prev. s eng.): "Jeste li potpuno sigurni da želite nastaviti s brisanjem svog računa?". Klikom na drugu potvrdu, pojavljuju se poruke brisanja baza podataka u stvarnom vremenu:
    a. *"Deleting items..."* – brisanje *Item-a* iz Firestore baze.
    b. *"Deleting files..."* – brisanje datoteka iz Cloud Storage baze.
    c. *"Deleting settings..."* – brisanje postavki iz RTDB baze.
    d. *"Deleting account..."* – brisanje korisničkog računa.

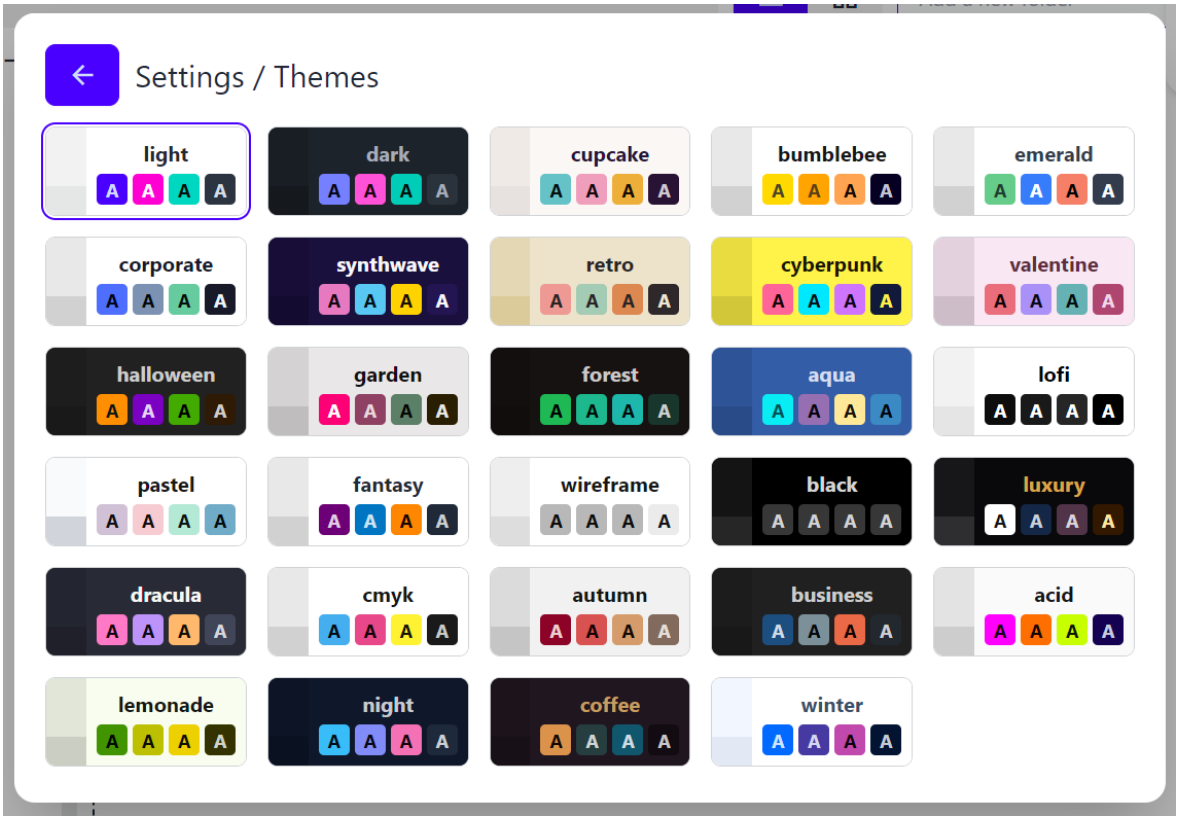
    Konačna poruka glasi da je korisnički račun obrisan.



Slika 4.5 Prozor korisničkih postavki na web aplikaciji

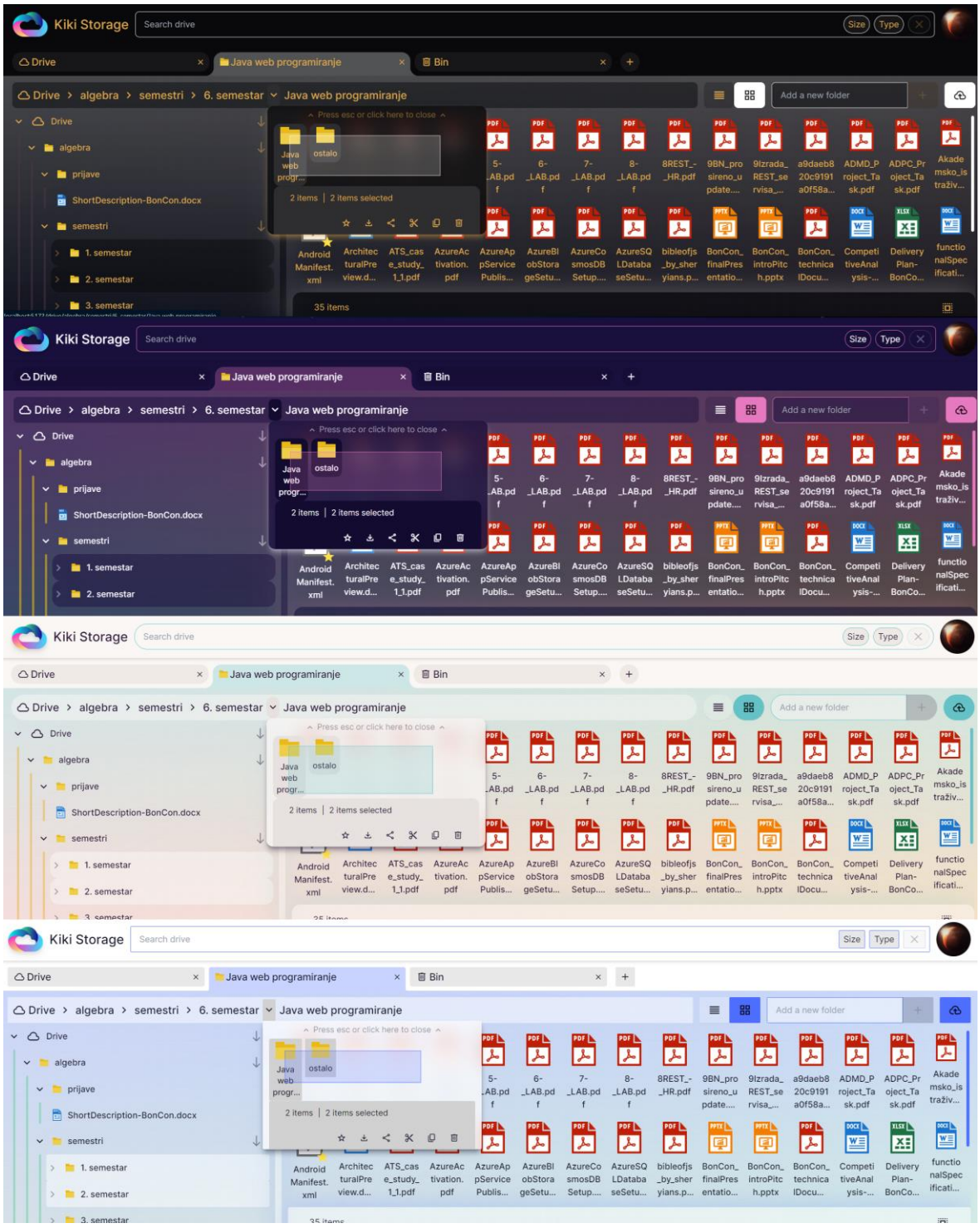
Klikom na gumb postavki otvara se prozor (Slika 4.5) koji sadrži opcije koje se uživo spremaju i primjenjuju:

1. Onemogućavanje tranzicija. Klikom na *checkbox* onemogućuju se tranzicije na način da `<html>` odnosno *root* element dokumenta dobiva CSS klasu koja zabranjuje sve tranzicije.
2. Skrivanje dokumenata u stablu mapa. Klikom na *checkbox* skrivaju se svi entiteti u stablu koji nisu tip mape, odnosno prikazuju se samo mape na način da se filtrira lista koja je već učitana na klijentu.
3. Promjena zuma na računalu ili mobitelu. Povlačenjem preko klizača zum se povećava ili smanjuje na način da *root* dobiva *style* atribut sa veličinom fonta. Ovo nije isto što i zum preglednika jer preglednik samo skalira sve piksele. Cijela aplikacija izrađena je uzimajući u obzir REM jedinicu veličine. Korištenje REM jedinica osigurava pristupačnost, skalabilnost i održivost dizajna prilagođavajući se zadanim postavkama korisnika i različitim uređajima te je zadana jedinica za Tailwind CSS. Veličina fonta može biti vrlo precizna s obzirom na to da podržava decimalna mjesta, dok zadano povećanje (*zoom*) preglednika skače sa 100% na 110% pa na 125% pa na 150% itd., a nije ni sinkroniziran između uređaja. [39] PX jedinica koristi se samo na nekoliko mjesta u aplikaciji, kao što su selekcijski okvir ili širina stabla mapa. Nakon postavljanja širine stabla mapa povlačenjem granice okvira, najlakše je uočiti razliku između *zoom-a* preglednika i *zoom-a* aplikacije. Preglednik će skalirati sve, dok će *zoom* aplikacije zadržati točan položaj granice. Ova opcija osobito je korisna za mobilne korisnike jer nema lakog pristupa *zoom-u* na mobilnom pregledniku.
4. Promjena teme, tj. ugođaja. Klikom na gumb "*Change*" otvara se lista svih tema i svaka se može odabrati klikom (Slika 4.6).



Slika 4.6 Prozor odabira tema (ugođaja) na web aplikaciji

To je moguće jer je kroz cijelu aplikaciju korišten DaisyUI za što više dijelova (Slika 4.7), uključujući brojne varijable koje DaisyUI isporučuje. [40] Zadana, svijetla (engl. *light*) tema posebno je prilagođena jer se za sve druge teme koristi prozirnost kako bi se uočili razni efekti. Budući da je glavna pozadinska boja svijetle teme bijela, umjesto prozirnosti ručno su upisane bazne boje. Stoga, često se koristi varijabla `isThemeLight` (Kôd 4.2).



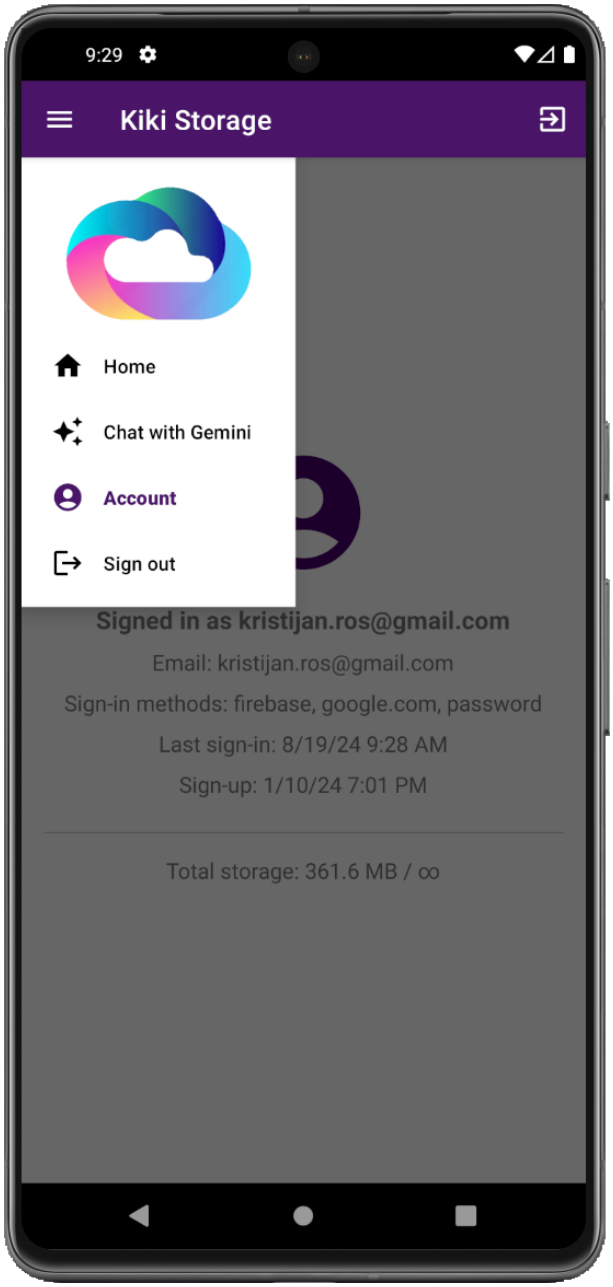
Slika 4.7 Primjeri raznih tema (ugođaja) web aplikacije

```
:class="{
  'hover:bg-base-200': isThemeLight,
  'hover:bg-base-100/25': !isThemeLight,
}"
```

Kôd 4.2 Korištenje svijetle teme u Tailwind-u s uvjetnim CSS klasama

5. Resetiranje postavki na zadano. Klikom na gumb otvara se upozorenje koje navodi što će sve biti resetirano. Potvrdom se poziva funkcija koja u potpunosti briše sve na putanji korisnika u RTDB bazi podataka.

Na Android aplikaciji korisničkom računu pristupa se putem izbornika koji se otvara klikom na ikonu hamburgera u gornjem lijevom kutu (Slika 4.8). Osim informacija o korisniku koje prikazuje i web aplikacija, ovaj zaslon prikazuje i ukupnu količinu iskorištenog prostora.



Slika 4.8 Korisnički račun na Android aplikaciji

4.1.3. Datoteke i mape

Glavni entitet projekta je stavka (engl. *Item*), koja predstavlja datoteku ili mapu.

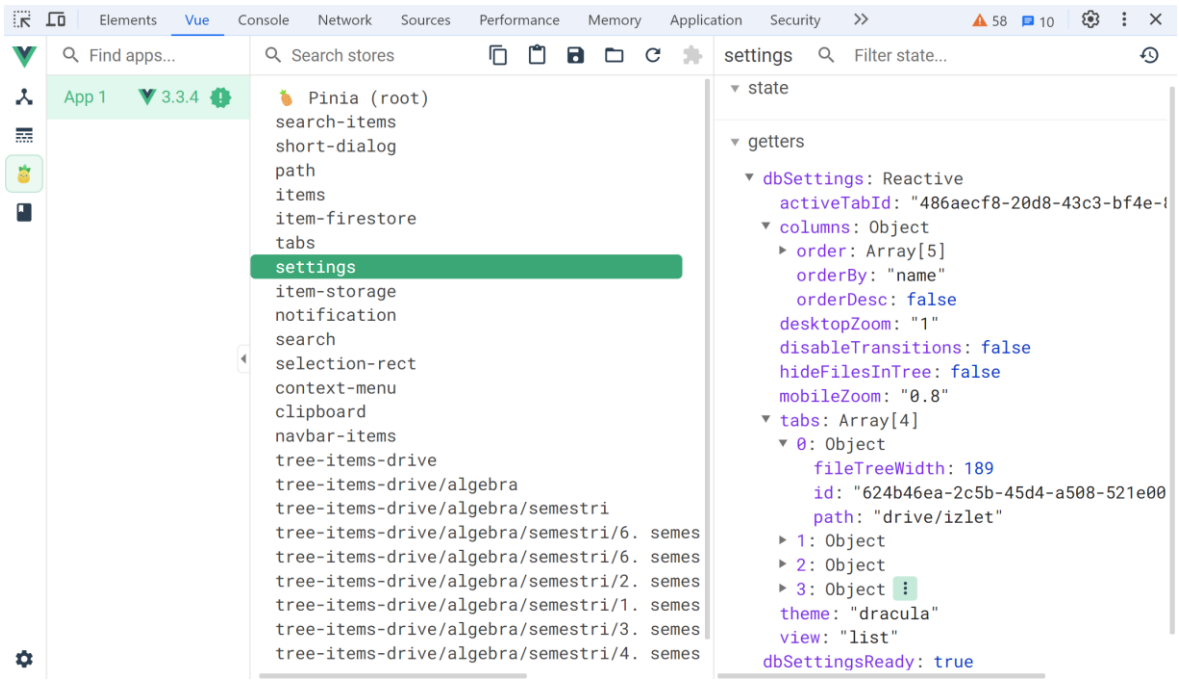
```
interface ItemCore {
  id?: string;
  name: string;
  type: string;
  dateAdded: Date;
  dateModified: Date;
  dateDeleted?: Date;
  path: string;
  isFolder?: boolean;
  isStarred?: boolean;
  size?: number;
}
interface Item extends ItemCore {
  isSelected?: boolean;
  isRenaming?: boolean;
  isCopied?: boolean;
  isCut?: boolean;
  newName?: string;
  storageFile?: any; // No type for ReturnType<typeof
useStorageFile>
}
type Overwrite<T, U> = Pick<T, Exclude<keyof T, keyof U>> &
U;
...
export type DbItem = Overwrite<
  ItemCore,
  {
    dateAdded: Timestamp;
    dateModified: Timestamp;
    dateDeleted?: Timestamp;
  }
>;
```

Kôd 4.3 TypeScript definicije stavke (*Item-a*)

Kao što je prikazano iz isječaka (Kôd 4.3), tri su glavna tipa *Item-a*:

1. **`ItemCore`** je tip koji sadrži jezgru informacija o *Item-u*. Budući da datoteka ne mora nužno imati ekstenziju (tip), potrebno je imati i `isFolder` svojstvo koje označava da je *Item* mapa. `dateModified` označava kada je zadnji put *Item* izmijenjen, a to je za mape specifično onda kad se direktni *Item-i* u mapi preimenuju, izbrišu ili dodaju. Ostala svojstva *Item-a* su samorazumljiva.
2. **`Item`** je tip koji uz jezgru *Item-a* sadrži opcionalna svojstva za rad u aplikaciji. Budući da VueFire nema ispravno definiran tip Vue *composable-a* za rad sa Cloud Storage servisom, koristi se tip *any*, koji dopušta bilo koji tip za svojstvo `storageFile`. Iako se uvijek treba izbjegavati *any*, ovdje bi tip bio previše složen za ručnu implementaciju.
3. **`DbItem`** je tip *Item-a* kojeg koristi Firebase Firestore. Slično kao i u Android aplikaciji, Firebase mora pretvoriti svoj tip u tip definiran u aplikaciji i obrnuto. Zbog toga, u web aplikaciji izvozi se vlastita funkcija `firestoreItemConverter` tipa `FirestoreDataConverter<ItemCore, DbItem>`.

Umjesto *composable* funkcija primarno se koriste Pinia trgovine (engl. *stores*) zbog naprednih mogućnosti za dijeljenje podataka i lakše otklanjanje pogrešaka (engl. *debugging*) (Slika 4.9).



Slika 4.9 Pinia kao dio Vue.js Devtools alata

Ključne trgovine koje rukuju s *Item-ima* su:

1. Mapa "firebase"
    a. `firestore.ts` sadrži poslovnu logiku za rad s Firebase Firestore-om i izlaže (engl. *export*s) API za druge trgovine.
    b. `storage.ts` sadrži poslovnu logiku za rad s Firebase Storage-om i izlaže API za druge trgovine, uz reference na *Item-e* koji su u procesu podizanja.
2. `cleanup.ts` izlaže funkcije koje se pozivaju nakon preimenovanja, premještanja ili brisanja *Item-a*. Na primjer, ako se preimenuje mapa, stabla mapa na svim karticama moraju se ažurirati i, radi performansa, moraju se očistiti sve *Item* trgovine s putanjama koje su uključivale preimenovanu mapu.
3. `clipboard.ts` izlaže funkcije za rad s međuspremnikom, koji sadrži *Item* reference.
4. `index.ts` sadrži definiciju *Item* trgovine. Jednom kada postane aktivna trgovina, sluša promjene u bazama podataka preko drugih trgovina i za svaki novi *Item* provjerava postoji li u "susjednim" *Item* trgovinama. To znači da sve trgovine uvijek referenciraju jedan te isti *Item*. Na primjer, ako se selektira dokument u radnom okviru, isti će biti selektiran u stablu mapa, pretrazi i svim drugim mjestima. Uz informacije o trenutnoj putanji, trgovina izlaže rukovatelje (engl. *handler*) događaja (engl. *event*). Na primjer, rukovatelj `handleDrop` poziva se iz pet različitih Vue komponenti.
5. `manager.ts` je tvornica *Item* trgovina i upravlja listom istih. Kad je trgovina u stablu zatražena sa zadanom putanjom i ne postoji, onda se stvara i sprema nova.

Sve trgovine koje pohranjuju informacije o *Item-ima* na klijentu imaju ugrađenu sigurnosnu mjeru: funkciju `$reset` koja se automatski pokreće prilikom odjave korisnika. Ova funkcija osigurava privatnost brisanjem svih osjetljivih informacija.

Svaka trgovina mora imati ID. Razlikuju se četiri *Item* trgovine:

1. Trgovina "*items*" koristi se za radni okvir.
2. Trgovina "*search-items*" koristi se za okvir u pretrazi.
3. Trgovina "*navbar-items*" koristi se za okvir u navigacijskoj traci.
4. Trgovine "*tree-items-${path}*" koristi se za neograničen broj trgovina u stablu mapa.

Dakle, koristi se pristup dinamičnih trgovina s obrascem tvornica (engl. *factory pattern*). Ovaj pristup s dinamičnim Pinia trgovinama je rijedak, ali koristan u ovoj situaciji. Ovisno

o ID-u i drugim parametrima u uvezivanju (`ItemStoreBindings`), trgovina se ponaša drugačije. No zato je potrebno i čistiti trgovine. [41]

Za *Item-e* se koriste *utility* funkcije kako bi pomogle pri stvaranju mapa, konverziji prenesenih datoteka, provjeri ispravnosti i slično. Provjera ispravnosti može rezultirati nekom od idućih poruka:

- *"A folder name can't be blank."*
- *"This destination already contains a ${type} named '${fullName}'."*
- *"A ${type} name can't contain any of the following characters: \\ / : * ? " < > |"*

Korijenske (engl. *root*) mape su predefinirane mape:

- "*Drive*" – glavna mapa sa svim podacima.
- "*Starred*" – ravna mapa (bez podmapa) sa podacima koji su označeni zvjezdicom, predstavlja omiljene podatke / favorite.
- "*Shared*" – mapa sa dijeljenim podacima. Za buduću implementaciju.
- "*Bin*" – mapa s obrisanim podacima.

Ovisno o stanju *Item-a*, mijenja se njegov izgled:

- Ako je *Item* u favoritima, ima žutu zvjezdicu u donjem desnom kutu ikone.
- Ako je *Item* fokusiran, ima tanak okvir.
- Ako je *Item* selektiran, mijenja pozadinsku boju.
- Ako je *Item* izrezan, potamnjuje 40%.

Android aplikacija isto dijeli naziv *Item* kao glavni entitet projekta. Kotlin definicija ekvivalent je TypeScript tipu `ItemCore` (Kôd 4.4).

```
data class Item(
    @DocumentId var id: String? = null,
    var name: String = "",
    var type: String = "",
    var dateAdded: Timestamp? = null,
    var dateModified: Timestamp? = null,
    var dateDeleted: Timestamp? = null,
    var path: String = "",
    @get:PropertyName("isFolder")
@set:PropertyName("isFolder") var isFolder: Boolean = false,
```

```
    @get:PropertyName("isStarred")
@set:PropertyName("isStarred") var isStarred: Boolean =
false,
    var size: Long? = null
)
```

Kôd 4.4 Kotlin definicija stavke (*Item-a*)

FirebaseUI Android knjižnjica pruža puno više alata od web verzije koja pruža samo sučelje za autentikaciju. Jedna od ključnih funkcija u Android aplikaciji gdje FirebaseUI pomaže je povezivanje podataka u stvarnom vremenu iz Cloud Firestore-a s korisničkim sučeljem [42], što se u određenoj mjeri može usporediti s VueFire web knjižnicom. Budući da se prati JavaBean obrazac imenovanja (engl. *naming pattern*) za rad s Firestore-om, potrebno je dodati eksplicitni naziv svojstva (`PropertyName`) zasebno za dobavljanje i postavljanje polja modela čiji naziv započinje sa "*is*", kao što je prikazano kôdom (Kôd 4.4). U suprotnom bi proces serijalizacije izbrisao "*is*", ne bi se pojavila greška ni upozorenje i veza s bazom ne bi radila.

Za razliku od weba, *Item* nema posebnih vizualnih stilova, osim kad je u favoritima i onda ima zvjezdicu na desnom rubu jer je prikaz uvijek lista.

4.1.3.1 Učitavanje, brisanje i preuzimanje

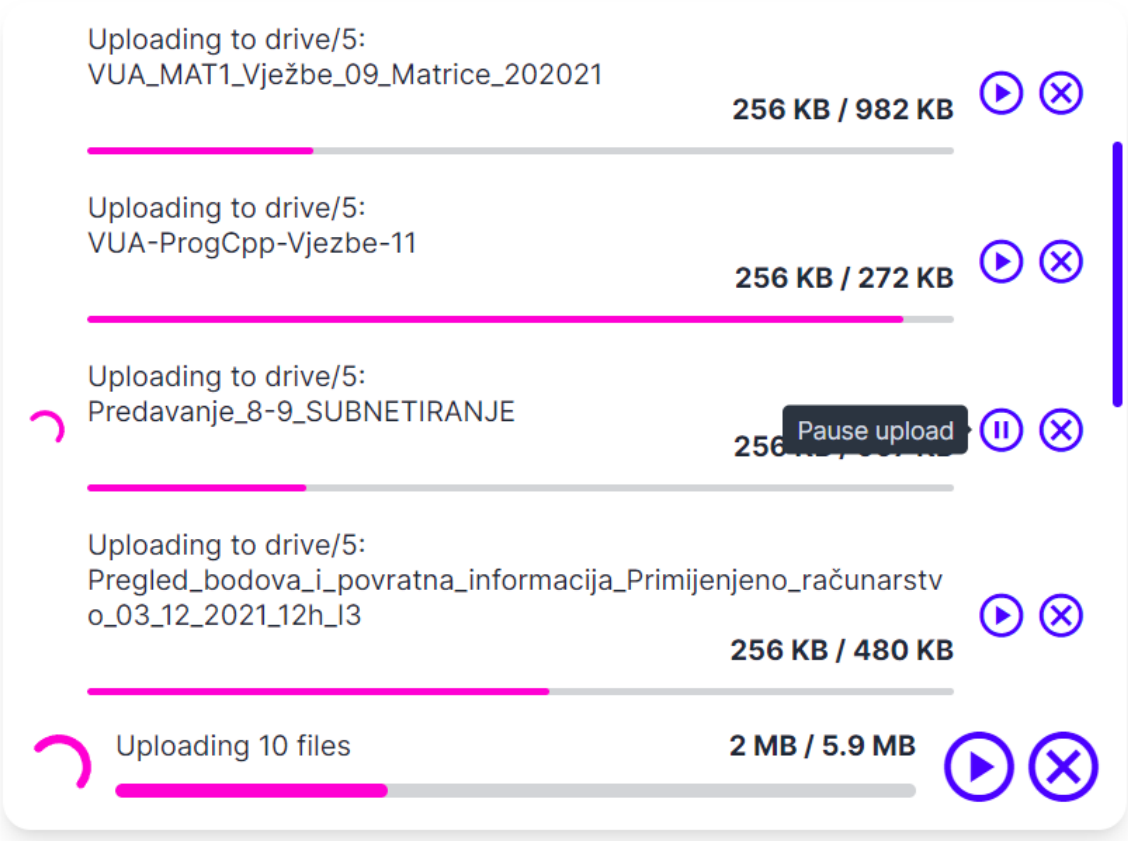
Učitavanje mapa direktno s uređaja nije moguće bez instalacije PWA, ali se mape mogu stvarati. U polju za stvaranje nove mape, pored navigacijske trake, potrebno je unijeti naziv te kliknuti na gumb s ikonom plusa (Slika 4.10).

Slika 4.10 Učitavanje datoteka i mapa na web aplikaciji

Učitavanje datoteka moguće je na dva načina:

1. Klikom na gumb s ikonom oblaka pored polja za stvaranje nove mape.
2. Povlačenjem i ispuštanjem datoteka iznad mjesta za ispuštanje (engl. *drag and drop*). Područje za ispuštanje datoteka vizualno se ističe pomoću efekta svijetljenja, odnosno radijalnim gradijentom koji prati poziciju kursora dok se povlači sadržaj.

Ispuštanjem ili odabirom iz preglednika datoteka OS-a otvara se prozor centriran pri dnu ekrana (Slika 4.11).



Slika 4.11 Učitavanje datoteka na web aplikaciji

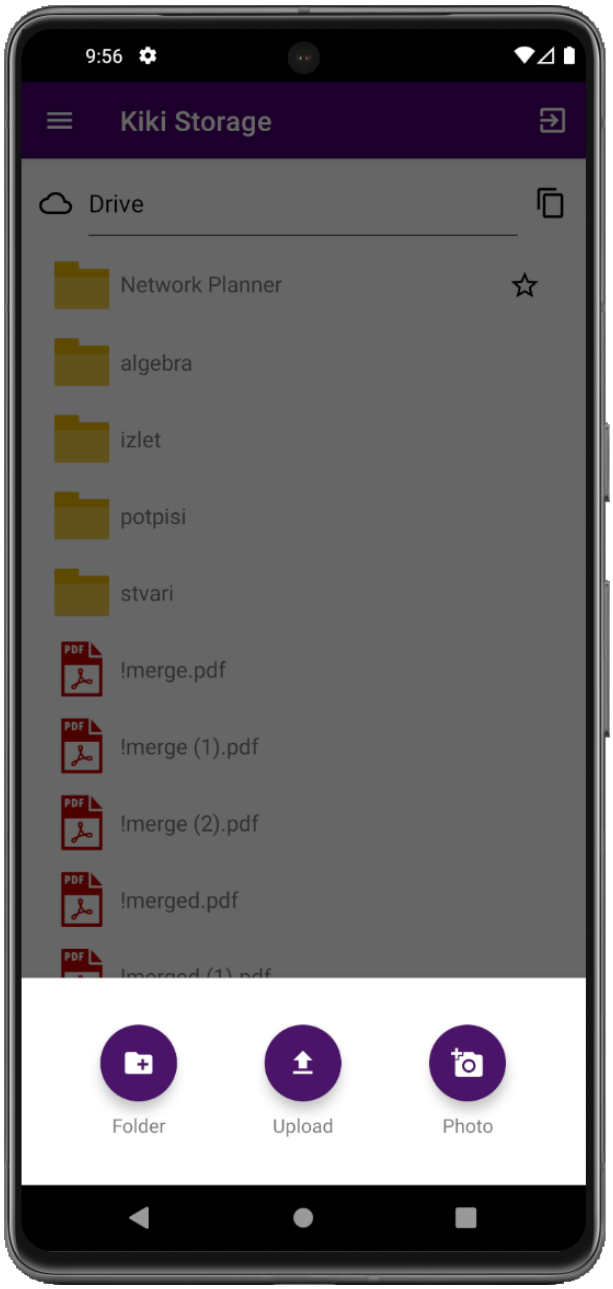
U prozoru se prikazuje suma svih datoteka uz statusnu traku i količinu bajtova koji se podižu u oblak (poslano bajtova / ukupno bajtova). Prelaskom kursora preko prozora prikazuju se isti podaci za sve datoteke individualno i svaku je moguće pauzirati, nastaviti ili otkazati. Moguće je pauzirati, nastavljati, otkazivati i sve datoteke odjednom klikom na velike kontrole.

Brisanje mapa i datoteka je jednako. Desnim klikom, u alatnoj traci ili tipkom "*delete*" na *Item(e)* otvara se prozor za potvrdu brisanja (N) *Item-a*. Ako je *Item* već u *Bin* mapi, briše se trajno (iz Firestore-a i Cloud Storage-a), a ako nije onda mijenja lokaciju u "*bin*", odnosno izvršava se premještaj. Isto bi se dogodilo povlačenjem i ispuštanjem *Item-a* iznad *Bin-a.*

Preuzimanje datoteka moguće je odabirom jedne ili više i klikom na ikonu preuzimanja u alatnoj traci. Datoteke se potom preuzimaju na klijentu te se prozor za odabir lokacije spremanja otvara tek nakon preuzimanja.

Ako je instalirana web aplikacija (koristi se PWA), ona ima pristup File System Access API-ju. To omogućuje učitavanje i preuzimanje bilo koje strukture mapa i datoteka. Dakle, ako se učita mapa s nizom podmapa, na isti način će biti prenesena u oblak i obrnuto na datotečni sustav uređaja (bez komprimiranja i raspakiravanja).

Na Android aplikaciji učitavanje datoteka i mapa izvršava se na zaslonu "Home", klikom na lebdeći kružić u donjem desnom kutu, nakon čega se otvara dijaloški okvir pri dnu zaslona (Slika 4.12). Dodatno, podržano je i direktno učitavanje fotografije koja se snima kamerom iz aplikacije.

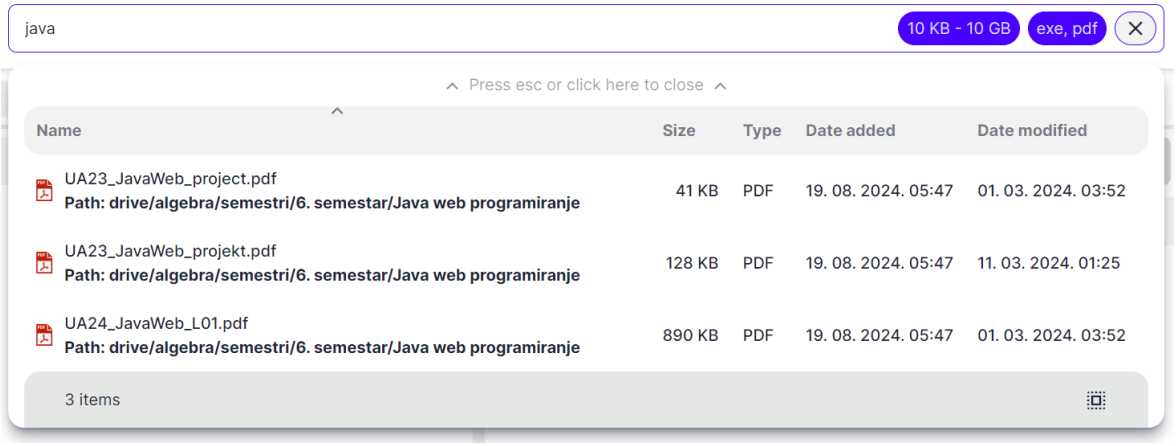


Slika 4.12 Učitavanje datoteka i mapa na Android aplikaciji

Početak i kraj učitavanja prikazan je kratkom plutajućom obavijesti (*toast notification*). Budući da se učitavanje izvršava putem Android *worker-a*, korisnik može izaći iz aplikacije te će se i dalje pojaviti ista obavijest završetkom učitavanja u pozadini. Na sličan način je implementirano preuzimanje koristeći plutajuće obavijesti i *worker*.

4.1.3.2 Pretraga

Na vrhu zaslona, unosom karaktera u polje za pretragu odmah se pojavljuju rezultati u padajućem prozoru, a taj prozor je radni okvir (Slika 4.13).



Slika 4.13 Pretraga datoteka i mapa na web aplikaciji

Rezultati su svi *Item-i* koji u nazivu sadrže traženi unos i zadovoljavaju filtere. Dodatno, svaki rezultat pokazuje svoju putanju. Kao i kod svih drugih radnih okvira, rezultati pretrage funkcioniraju u stvarnom vremenu. Pretraga maksimalno koristi Firestore-ove brze složene upite (engl. *compound queries*), a zatim za napredne filtere koje Firestore ne podržava pretraga filtrira rezultate na klijentu.

Opcije pretrage nalaze se na desnom kraju polja i to su:

1. Veličina (engl. *size*) – Klikom se otvara prozor za unos raspona veličine traženih dokumenata u odabranoj mjernoj jedinici (B, KB, MB, GB, TB).
2. Tip (engl. *type*) – Klikom se otvara prozor za odabir pretrage dokumenata ili mapa. Ako se pretražuju dokumenti, u predviđeno polje može se unijeti niz traženih tipova odvojenih zarezom. Na primjer, "pdf, docx, txt".

U bilo kojem od prozora opcija pretrage, klikom na očisti (engl. *clear*) resetira se filter, a klikom na pretragu (engl. *search*) primjenjuje se filter, što je dodatno istaknuto promjenom boje gumba kojim se otvara prozor. Sve opcije, uključujući i traženi naziv, mogu se resetirati klikom na gumb s ikonom X na kraju polja za pretragu.

Pretraga se primjenjuje samo na trenutno otvorenu korijensku mapu i moguće ju je mijenjati dok su rezultati pretrage otvoreni. Kad se u rezultatima pretrage otvori mapa, zatvara se prozor pretrage i u trenutnoj kartici se učitava putanja mape.

4.1.3.3 Alatna traka

Alatna traka (Slika 4.14) dio je svakog radnog okvira osim stabla mapa, a za brži pristup opcijama trake koristi se desni klik na *Item*.

Slika 4.14 Alatna traka web aplikacije

Ovo su opcije koje nudi dok su jedan ili više *Item-a* selektirani:

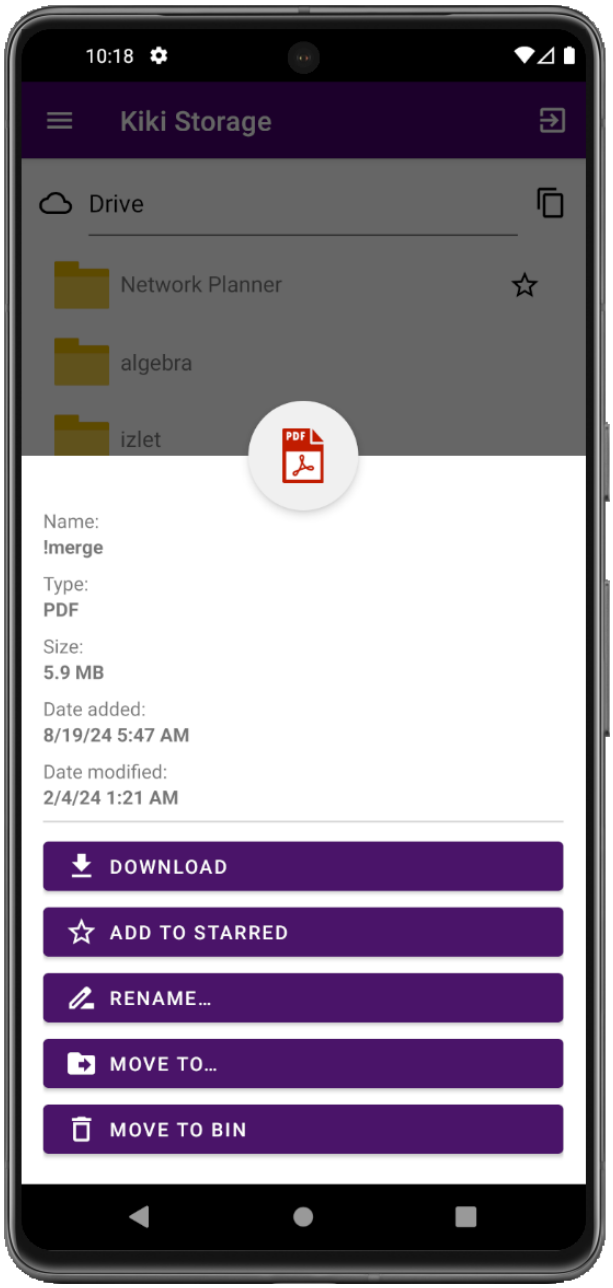
1. Dodavanje/uklanjanje zvjezdice – dodaje ili uklanja *Item* iz *Starred* mape.
2. Otvaranje u novoj kartici – pojavljuje se samo ako je odabrana jedna mapa.
3. Preuzimanje – moguće je preuzeti više dokumenata odjednom, a u PWA cijele strukture mapa.
4. Dijeljenje – za buduću implementaciju.
5. Preimenovanje – pojavljuje se samo ako je odabran jedan *Item*, a klikom se otvara polje za unos teksta preko naziva *Item-a* i klikom na kvačicu se potvrđuje.
6. Izrezivanje – stavlja *Item-e* u međuspremnik i nakon lijepljenja ih briše.
7. Kopiranje – za buduću implementaciju.

Ovo su opcije koje nudi dok ni jedan *Item* nije selektiran:

1. Lijepljenje – premješta izrezane ili kopira kopirane *Item-e*.
2. Selekcija svega – selektira sve *Item-e*.

Alatna traka uvijek prikazuje ukupan broj *Item-a*, a kad su neki selektirani prikazan je njihov broj i ukupna memorijska veličina. Veličina se ne prikazuje ako je u selekciji mapa.

Na Android aplikaciji opcije alatna trake iz weba implementirane su ispod svojstava *Item-a*, koji se nalaze u dijaloškom okviru pri dnu zaslona (Slika 4.15). Okvir se otvara dodirom na dokument ili dugim dodirom na mapu.



Slika 4.15 Svojstva i opcije *Item-a* na Android aplikaciji

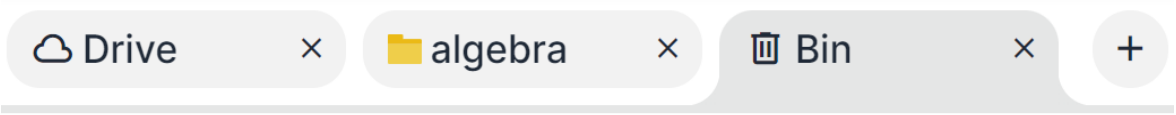
4.1.4. Navigacija, organizacija i upravljanje

U ovom poglavlju objašnjena je implementacija navigacije putem kartica, navigacijske trake, radnog okvira, stabla mapa, zatim organizacija kartica i stabla mapa te na kraju upravljanje putem kontekstnih izbornika, kontrolama, prečacima i gestama.

4.1.4.1 Kartice

Kartice su ključan dio web aplikacije i jedna su od najvećih prednosti u odnosu na postojeća rješenja (Slika 4.16). Stoga, nakon registracije odmah se otvara prva kartica s nazivom

"Drive". Naziv kartice je mapa koja je otvorena. Funkcionalnost je slična karticama preglednika ili File Explorer-a OS-a Windows 11 od verzije 22H2. Nemoguće je imati nula kartica i nema ograničenja broja kartica.



Slika 4.16 Kartice web aplikacije

Zadana (engl. *default*) putanja za svaku novu karticu je "/drive". Otvara se klikom na plus ikonu, srednjim klikom na bilo koju nekorijensku mapu ili putem kontekstnog izbornika, a zatvara srednjim klikom ili klikom na X ikonu. Duplim klikom na karticu aplikacija prelazi u cijeli zaslon (engl. *fullscreen*) te se na isti način izlazi iz cijelog zaslona.

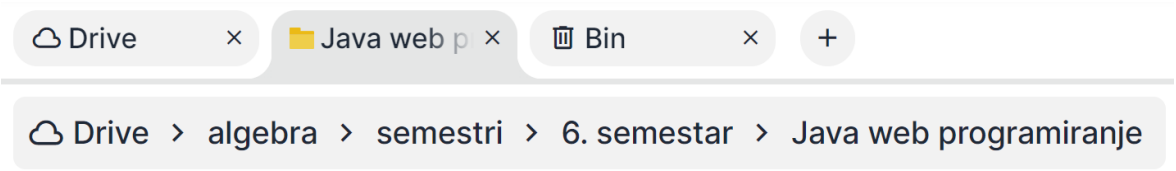
Kad se aplikacija otvara, učitava se putanja koja se nalazi u navigacijskoj traci preglednika. Ako ta putanja ne postoji ili je nema, otvara se kartica sa zadanom putanjom, a ako postoji otvara se kartica s tom putanjom. U oba slučaja, ako već postoji kartica s traženom putanjom otvara se ta, a u suprotnom se stvara i otvara nova kartica.

Kartice je moguće organizirati povlačenjem i ispuštanjem, što je implementirano s Vue *Slicksort* paketom. Redoslijed i aktivna kartica sinkroniziraju se na svim uređajima.

Svaka kartica vizualno sadrži navigacijsku traku, radni okvir i stablo mapa. Navedeni elementi ne dupliciraju se za svaku karticu, već mijenjaju stanje promjenom kartica. Drugim riječima, sve komponente ispod kartica su susjedni DOM elementi. Komponente slušaju RTDB promjene i ažuriraju stanje ovisno o putanji, proširenim putanjama i širini stabla mape koje čuva kartica.

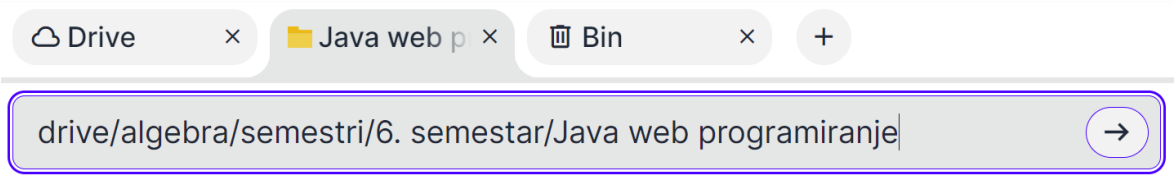
4.1.4.2 Navigacijska traka

Navigacijska traka nalazi se odmah ispod kartica i sadrži trenutnu putanju kartice (Slika 4.17). Dok nije aktivna, putanja je uvijek sinkronizirana s putanjom u navigacijskoj traci preglednika.



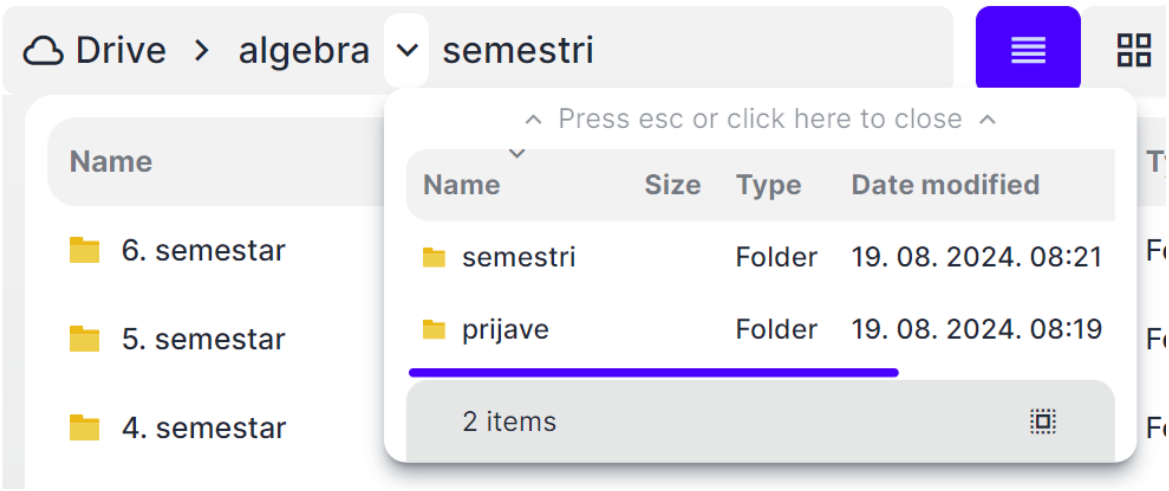
Slika 4.17 Navigacijska traka web aplikacije

Navigacijska traka je aktivna dok ima fokus nakon što je kliknuta u prazno polje (desno od gumbova) te se mijenja u tekstualno polje i može se upisati nova putanja. (Slika 4.18).



Slika 4.18 Aktivna navigacijska traka web aplikacije

Nakon što je upisana nova putanja, ona se primjenjuje enterom ili klikom na strelicu. Ako je unesena putanja neispravna, pojavljuje se odgovarajuće upozorenje i traka se resetira. Dok nije aktivna, navigacijska traka sastoji se od niza gumbova. Klikom na bilo koji gumb u traci koji je dio putanje, učitava se putanja do toga gumba. Između dijelova putanje nalaze se strelice (>) na koje se može kliknuti kako bi se otvorio radni okvir (Slika 4.19).

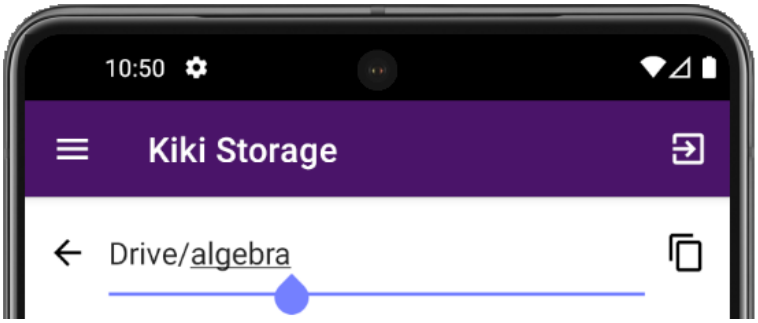


Slika 4.19 Radni okvir navigacijske trake na web aplikaciji

Radni okvir navigacijske trake prikazuje sadržaj putanje do gumba lijevo od strelice koja kliknuta i ostaje otvoren dok se ručno ne zatvori. Dakle, ako je radni okvir otvoren, promjeni se kartica i ta kartica ima putanju okvira, okvir ostaje otvoren. Ako ta druga kartica nema istu putanju, radni okvir će se skriti, a povratkom na karticu koja ima tu putanju, opet će se prikazati.

Prvi gumb u traci je korijenska putanja i osim što se može kliknuti kako bi se učitala, prelaskom kursora prikazuje se padajući izbornik s drugim korijenskim putanjama prema kojima je moguće navigirati. To je vrlo korisno jer su korijenske putanje u stablu mapa uglavnom proširene i tada nisu brzo dohvatljive.

Na Android aplikaciji navigacijska traka je tekstualno polje koje se može uređivati (Slika 4.20).



Slika 4.20 Navigacijska traka Android aplikacije

Unosom karaktera, ako tražena mapa postoji odmah se pojavljuje sadržaj. Ako trenutna lokacija nije korijenska, lijevo od polja nalazi se ikona strelice. Dodirom na strelicu odlazi se u roditeljsku mapu. Desno od polja je ikona za kopiranje putanje u međuspremnik uređaja, što može biti korisno ako će se premještati sadržaj.

Putem *PreferenceManager* servisa, aplikacija lokalno sprema zadnje učitanu putanju kako bi se novim otvaranjem aplikacije na istom mjestu mogao nastaviti rad.

Budući da fragment sprema stog (engl. *stack*) putanja koje je korisnik posjetio od pokretanja aplikacije, vraćanjem nazad učitavaju se prethodne putanje. Tipkanjem u navigacijsko polje stog se ne puni, nego se putanja sprema na stog tek nakon zatvaranja tipkovnice pokretanjem (`android:imeOptions="actionGo"`). Implementirano je i brisanje cirkularnih putanja na vrhu stoga kako se korisnik ne bi vrtio u krug.

4.1.4.3 Radni okvir

Radni okvir je okvir za sve *Item-e* i svi su uvijek sinkronizirani u stvarnom vremenu. Predstavlja ga Vue komponenta `<ExplorerGrid>`. Budući da je to jedna od najvažnijih SFC komponenti, u ovom je poglavlju opisana njena struktura, redoslijedom koji je konzistentan za sve komponente.

Na početak komponenti dolazi `<script setup lang="ts">`, ako je potrebna poslovna logika, a gotovo uvijek jest. Ovaj dio poznat je kao *ViewModel* u MVVM arhitekturi i tu se piše TypeScript, što je označeno parametrom.

Obično, kad trebamo proslijediti podatke od roditeljske do podređene komponente, koristimo ulazne parametre ili svojstva (Vue *props*). Svi Vue *prop-ovi* tvore jednosmjerno vezanje prema dolje između svojstva djeteta i nadređenog: kad se nadređeno svojstvo

ažurira, ono će se primijeniti prema dolje do djeteta, ali ne obrnuto. Time se sprječava da podređene komponente slučajno ne mutiraju nadređeno stanje, što može otežati razumijevanje protoka podataka aplikacije. Međutim, ako imamo veliko stablo komponenti, a duboko ugniježđena komponenta treba nešto od udaljene komponente pretka, samo s *prop-ovima* morali bismo ih proslijediti kroz cijeli roditeljski lanac. To može biti jako naporno za pratiti kad neki roditelji ni ne trebaju taj *prop* i zato je poznat termin "*props drilling*", a primjenjuje se i na druge JavaScript okvire kao što je React. Vue ima rješenje: *provide* i *inject*. Roditeljska komponenta može poslužiti kao pružatelj ovisnosti (engl. *dependency provider*) za sve svoje potomke. Bilo koja komponenta u stablu potomaka, bez obzira na to koliko je duboko ugniježđena, može ubrizgati (engl. *inject*) ovisnosti koje pružaju komponente u njenom roditeljskom lancu. Ovaj *pattern* poznat je kao *Dependency Injection*. [43]

U ovom slučaju, `<ExplorerGrid>` komponenta ubrizgava logičke varijable (engl. *bool*) koje govore komponenti nalazi li se ona u stablu podataka ili u pretrazi.

Zatim se redom definiraju:

- Prethodno spomenuti *prop-ovi*. U ovom slučaju to je samo jedan *prop*: aktivna Pinia *Item* trgovina (`ReturnType<StoreDefinition>`).
- Trgovina kartica, selekcijskog okvira, postavki i kontekstnog izbornika.
- Redom, DOM elementi iz predloška iste komponente i jednostavne strukture u tip *Ref*. Vue *Ref* uzima unutarnju vrijednost i vraća reaktivni i promjenjivi objekt koji ima jedno svojstvo *"value*" koje ukazuje na unutarnju vrijednost. [44]
- *Computed* varijable sa jednostavnim i složenim strukturama. Vue *computed* uzima *getter* funkciju i vraća reaktivni *ref* objekt samo za čitanje vrijednost koju vraća *getter*. Također može uzeti objekt s funkcijama *get* i *set* za stvaranje *ref* objekta koji se može pisati. [44]
- Promatrači (engl. *watchers*). Promatrači promatraju jedan ili više reaktivnih izvora podataka i pozivaju *callback* funkciju kada se bilo koji izvor promijeni. Treći argument je opcionalan objekt s opcijama promatrača. [44]
- Rukovatelji, tj. obične funkcije koje rukuju događajima iz *template-a*.

Zatim započinje `<template>`, dio svake Vue SFC komponente poznat kao *View* u MVVM arhitekturi.

Ovdje je otkriven nova, vrlo specifična Vue Core greška (engl. *bug*) izazvana **komentarom**. Uočena je razlikom u ponašanju aplikacije u razvojnom modu u odnosu na produkciju. Riječ je o vizualnoj grešci jer se animacije ne izvršavaju ispravno. Preciznije, dodavanje komentara u *root* `<template>` elementa stvara animaciju `<TransitionGroup>` elementa gdje ne bi trebala biti. Stvorena je minimalna reprodukcija na Vue SFC Playground aplikaciji, a u nastavku je dio njezina kôda.

```
<script setup>
import { ref } from 'vue'
const showSpace = ref(false);
</script>
<template>
  <!-- comment -->
  <div>
    <button @click="showSpace = !showSpace">Show
space</button>
    <div v-if="showSpace"><br><br>Some space</div>
    <TransitionGroup>
      <div :key="1">text</div>
    </TransitionGroup>
  </div>
</template>
<style>
.v-move {
  transition: transform 1s;
}
</style>
```

Kôd 4.5 Reprodukcija Vue Core greške (engl. *bug*) u Vue verziji @7a6c665

Još prije je otvoren *Issue* na GitHub-u gdje je komentar izazvao problem s grupnim tranzicijama, ali je izlazna greška drugačija. Budući da je *Issue* i dalje otvoren, nije otvoren dodatni jer je ovo vrlo vjerojatno varijacija istog. Greška (Kôd 4.5) je predstavljena Vue Core programerima u Discord grupi. Odgovor Vue Core programera Skirtle je:

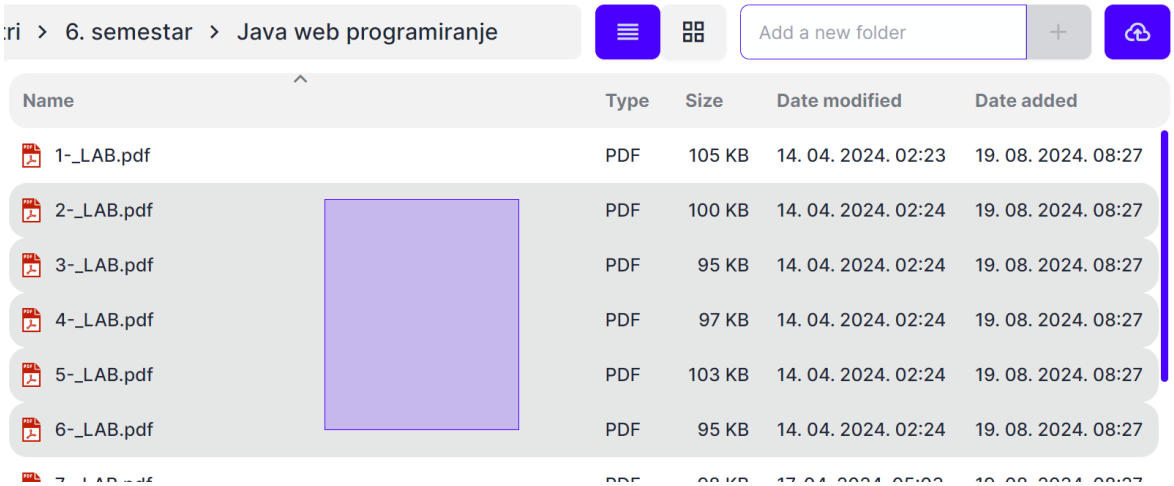
"The comment is adding in an extra fragment and tweaking the patch flags, and it seems that is inadvertently triggering the TransitionGroup to handle a move when its children haven't changed. It shouldn't be animating just because a sibling v-if has changed. It's a bug in Vue core and it needs fixing."

Zanimljivost je što je taj komentar (greška) stvorio prikaz kakav je bio željen. Još veća zanimljivost je da proces izgradnje onemogućuje repliciranje greške u produkciji čak i kada se komentari zadrže promjenom postavki Vue *compiler-a*.

Pronađeno je rješenje za dobivanje željenog prikaza: komentar je premješten, a umjesto njega dodan je još jedan `<TransitionGroup>` omotač.

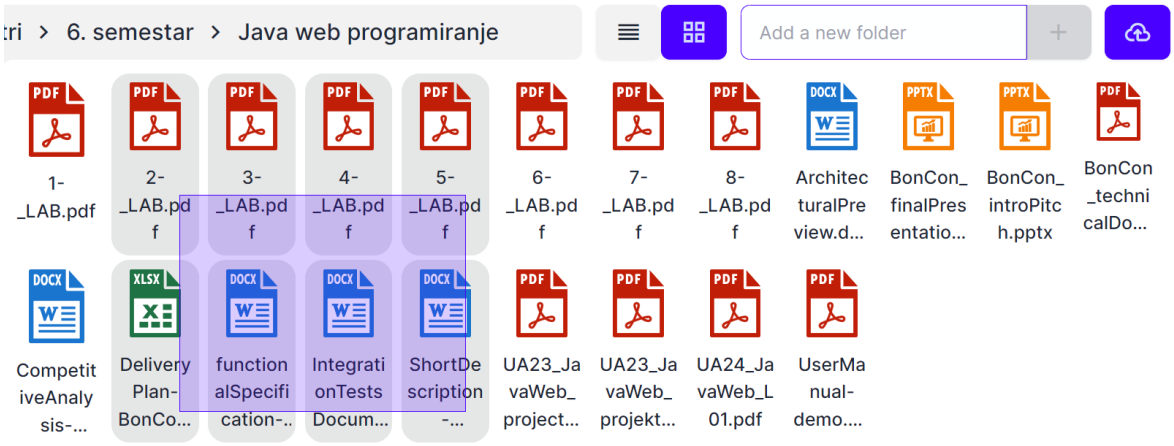
Animacija o kojoj je ovdje riječ događa se pri promjeni pogleda na radni okvir.

Pogled radnog okvira primjenjuje se na sve radne okvire, a može biti:



Slika 4.21 *List* pogled radnog okvira web aplikacije

1. "*List view*" (Slika 4.21) – *Item-i* zauzimaju cijeli redak i nižu se jedan ispod drugoga. Prvi redak je fiksirani `<ExplorerGridHead>` koji predstavlja zaglavlje i sadrži stupce koji su svojstva *Item-a*: naziv, tip, veličina, datum izmjene i datum dodavanja. Klikom na stupac u zaglavlju *Item-i* se sortiraju uzlazno ili silazno ovisno o svojstvu, a ponovnim klikom na isti stupac sortiranje je suprotno. Prelaskom kursora preko stupca u zaglavlju pojavljuje se ikona pomoću koje se povlačenjem i ispuštanjem može promijeniti redoslijed prikaza svojstava.



Slika 4.22 *Grid* pogled radnog okvira web aplikacije

2. "*Grid view*" (Slika 4.22) – *Item-i* su posloženi u fleksibilnu mrežu u kojoj se automatski raspoređuju u redove, uz određena ograničenja.

Ono što omogućuje prijelazne animacije jest zadržavanje istih elemenata i stila prikazivanja (CSS *display*). Bez obzira koji je prikaz odabran, svi elementi ostaju postojani u DOM-u, osim zaglavlja koje se prikazuje ili ne prikazuje. Ovo je bio najveći CSS izazov u aplikaciji i riješen je korištenjem CSS prikaza koji je tek u rujnu 2023. postao *Baseline* (osnovna linija kompatibilnosti) za moderne preglednike, a to je CSS *Subgrid*.

Kad se na spremnik mreže (engl. *grid container*) primijeni svojstvo `display: grid`, samo njegova izravna djeca postaju elementi mreže i mogu se postaviti na stvorenu mrežu. Djeca tih elemenata prikazuju se u normalnom toku. Moguće je "ugnijezditi" mreže tako da element mreže postane spremnik mreže. Međutim, te ugniježđene mreže su neovisne o roditeljskoj mreži i jedna o drugoj, što znači da ne preuzimaju dimenzije traka od roditeljske mreže. To otežava poravnavanje ugniježđenih elemenata mreže s glavnom mrežom. Ako se vrijednost `subgrid` postavi na `grid-template-columns`, `grid-template-rows` ili oboje, umjesto stvaranja novog popisa traka, ugniježđena mreža koristi trake definirane na roditeljskoj mreži. [45]

U radnom okviru `grid-template-columns: subgrid;` koristi se na čak dvije razine dubine u istom mrežnom spremniku: prvo na tijelu okvira, a zatim na svakom *Item-u* okvira koji je u tome tijelu. Susjed tijelu je zaglavlje pa i ono koristi *subgrid* prikaz.

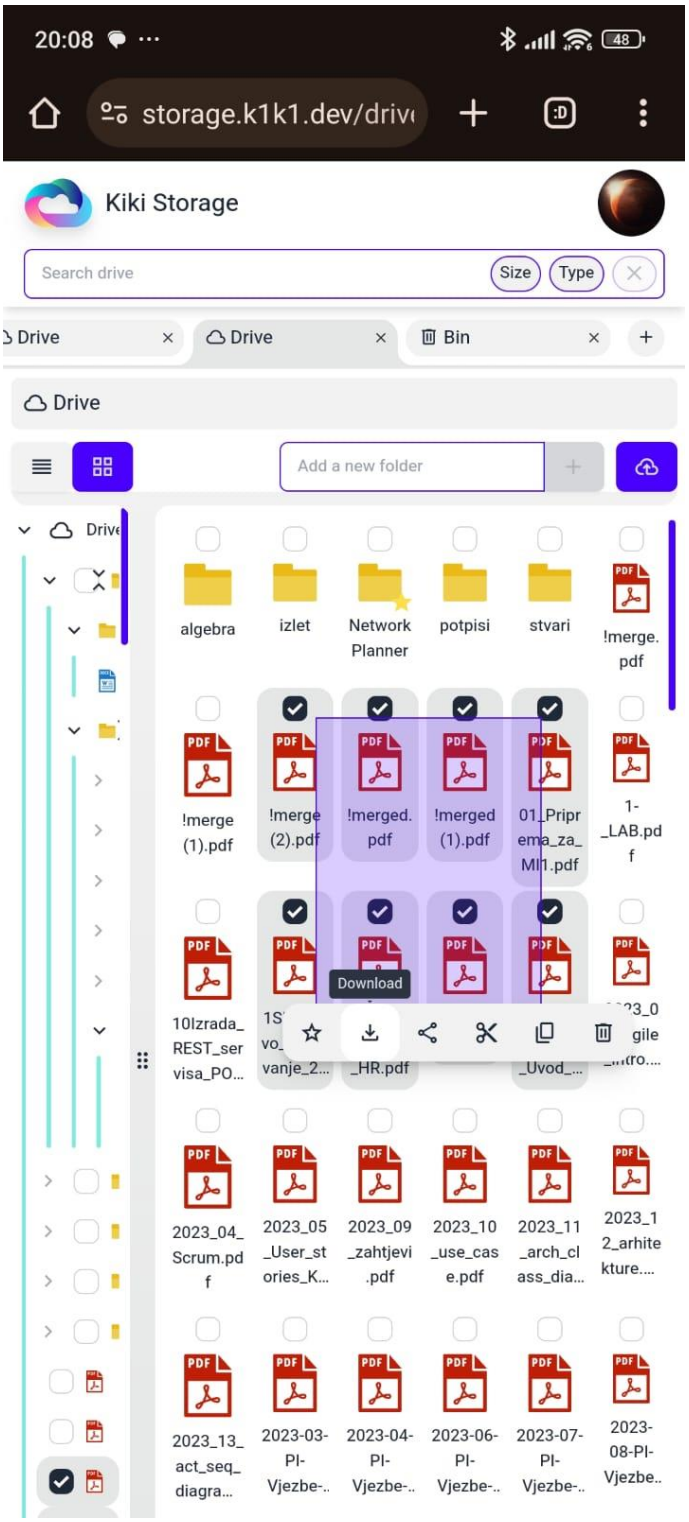
Subgrid se pokazao vrlo korisnim, ali nakon implementacije stabla mapa pojavio se velik problem performansi. Budući da stabla rekurzivno koriste radni okvir, puno je veća složenost ako je na primjer deset proširenih ugniježđenih radnih okvira koji koriste *subgrid-e* i svaki

radni okvir ima mnoštvo *Item-a*. Broj *Item-a* nije ni približno utjecao na performanse onoliko koliko je utjecala dubina proširenosti. Rad sa takvim stablom je moguć bez ikakvog zastajkivanja, ali je problem kad se promijeni kartica s jednostavnog stabla na složeno ili proširi mapa s već puno proširenih podmapa. Tada bi se sve na kartici preglednika "zamrznulo" na nekoliko sekundi na konkurentom Intel 13700K procesoru.

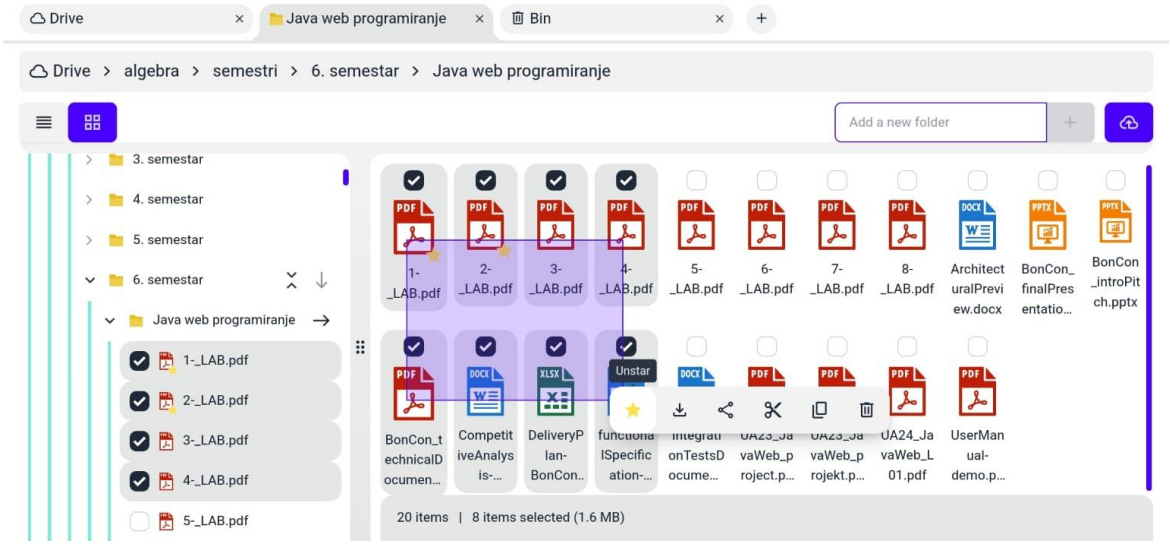
Rješenje je koristiti optimizirani *flex* prikaz umjesto svih *grid* i *subgrid* prikaza. Budući da je ova komponenta zajedno s podkomponentama bila već jako dobro razrađena, nije bilo praktično izrađivati nove komponente za stablo podataka. Stoga, uvedene su uvjetne CSS klase.

Dakle, glavni radni okvir nalazi se desno od stabla mapa, a osim njega i svih radnih okvira u stablu mapa, radni okviri koriste se još u rezultatima pretrage i u navigacijskoj traci te su poznati kao "lebdeći radni okviri".

Vrlo važan element radnih okvira je selekcijski okvir (Slika 4.23, Slika 4.24).



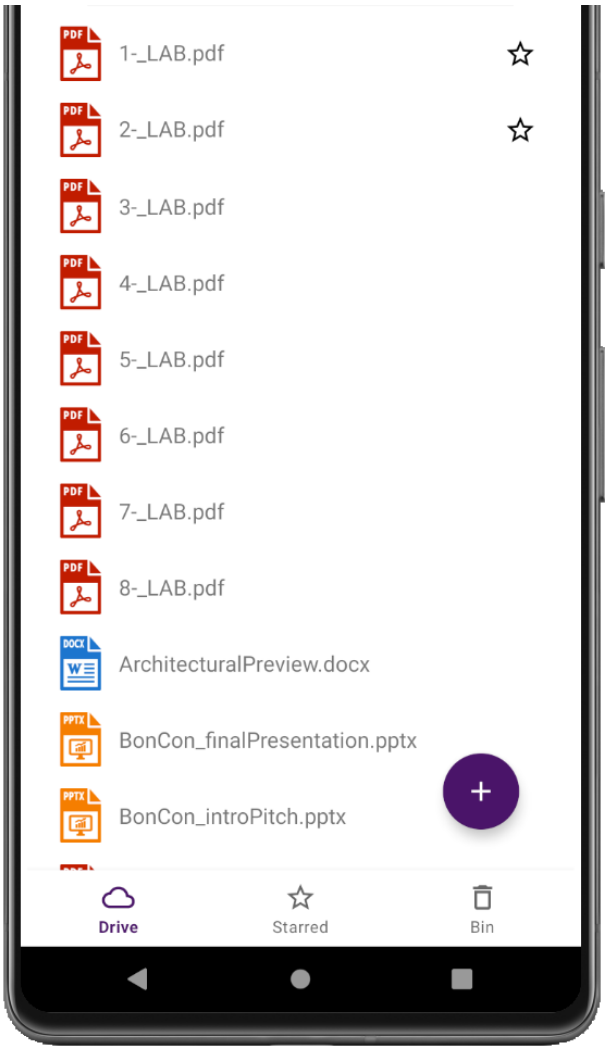
Slika 4.23 Selekcijski okvir web aplikacije na dodirnom uređaju u portretnoj orijentaciji



Slika 4.24 Selekcijski okvir web aplikacije na dodirnom uređaju u punom zaslonu s pejzažnom orijentacijom

Selekcijski okvir služi za brz i jednostavan izbor više *Item-a* i dodatno ističe posebnost ove aplikacije u odnosu na postojeća rješenja jer je primjenjiv na svim radnim okvirima. Trgovina selekcijskog okvira sadrži najviše matematike kako bi zadovoljila različita okruženja u kojima treba funkcionirati, uključujući dodirne uređaje. Najveći izazov bio je napraviti okvir funkcionalnim tijekom listanja (engl. *scrolling*), kad se dođe do rubova radnih okvira ili se od njih udaljava. Riješen je vremenskim intervalima, zbrajanjem, oduzimanjem i dijeljenjem relativnih duljina i koordinata događaja.

Fokus je bitan pojam kod radnih okvira jer korisnik mora znati koji okvir ga u jednom trenutku "sluša". Lebdeći radni okviri imaju pojačanu sjenu kad su fokusirani, a radni okviri u stablu mapa imaju podebljanu liniju u drugoj boji. Bitan je i *stacking* prozora (`z-index`) koji govori koji prozori su iznad kojih. Na primjer, imamo otvorene lebdeće radne okvire i nasumično selektirane *Item-e* u oba. Započinjemo s fokusom na pretragu. Prvim pritiskom na tipku Esc sve se deselektira u tome okviru rezultata. Drugim pritiskom radni okvir se zatvara i fokus se prebacuje na radni okvir u navigacijskoj traci. Sada on prisluškuje sve kontrole tipkovnice. Trećim i četvrtim pritiskom deselektira se i zatvara taj prozor. Fokus se premješta na glavni radni okvir.

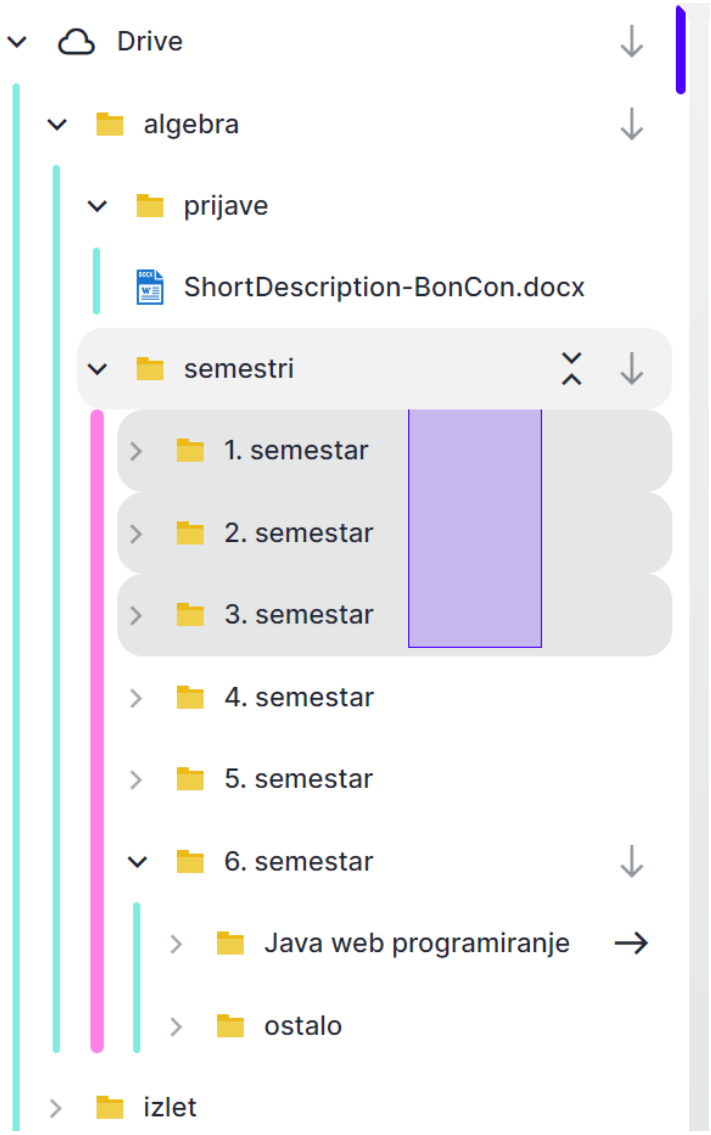


Slika 4.25 Radni okvir Android aplikacije

U Android aplikaciji, radnim okvirom (Slika 4.25) može se nazvati `RecyclerView` komponenta. Adapter komponente je vlastita klasa `ItemAdapter` koja nasljeđuje `FirestoreRecyclerAdapter`, uvezen iz FirebaseUI knjižnjice. *Item-i* su također uvijek sinkronizirani u stvarnom vremenu.

4.1.4.4 Stablo mapa

Stablo mapa (Slika 4.26) sadrži rekurzivne Vue komponente radnih okvira.



Slika 4.26 Stablo mapa web aplikacije

Stablo uvijek sadrži korijenske mape te se one ne mogu selektirati, nego samo otvoriti ili proširiti (engl. *expand*). Pored svake mape u stablu mapa, osim ravnih mapa (*Starred*), nalazi se strelica (>) kojom se proširuje sadržaj mape u stablu. Ponovnim klikom na strelicu ili na liniju koja se produžuje iz strelice prema dolje, mapa se sažima (engl. *collapses*). Ako mapa u sebi ima jednu ili više mapa koje su proširene, prelaskom kursora preko nje pojavljuje se ikona za rekurzivno sažimanje svih mapa u toj mapi.

Ako je bilo koja od mapa u trenutno otvorenoj putanji onda ima strelicu prema dolje na desnom rubu, a ako je potpuno jednaka onda je smjer strelice prema desno.

Zadana širina okvira stabla mapa je automatska, a može se postaviti povlačenjem desnog ruba okvira. Duplim klikom na taj okvir širina postaje ponovo automatska.

4.1.4.5 Kontekstni izbornici i premještaj podataka

Kontekstni izbornici su mali prozori koji se otvaraju desnim klikom ili dugim dodirom na dodirnim uređajima. Mjesta na kojima se aktiviraju su:

- *Item-i* – otvara se konteksti izbornik s alatnom trakom.
- Stupci u zaglavlju radnog okvira – otvara se konteksti izbornik s odabirom prikaza stupaca.
- Radni okviri – otvara se konteksti izbornik s opcijama sortiranja.

Item-i se mogu premještati samo ako su prije toga selektirani. To je najlakše učiniti povlačenjem i ispuštanjem. Druga opcija premještanja je putem kontekstnog izbornika, izrezivanjem u međuspremnik i lijepljenjem, a to je i jedina opcija za korisnike na dodirnim uređajima. Mjesta za ispuštanje su svi radni okviri i mape, a povlačenjem je moguće promijeniti karticu, otvoriti radni okvir u navigacijskoj mapi i proširiti mape u stablu mapa. Premještanje je podržano i između prozora preglednika, što je posebno korisno pri rukovanju s dva rezultata pretrage u dva različita prozora ili kartice preglednika. No ako se pokušaju premjestiti *Item-i* iz različitih korisničkih računa pojavljuje se upozorenje da to još nije moguće te je ostavljeno za buduću implementaciju.

Na Android aplikaciji, *Item-i* se premještaju jedan po jedan pomoću opcije "MOVE TO…" i unosom apsolutne putanje.

4.1.4.6 Kontrole, prečaci i geste

Klikom na *pomoć* u padajućem izborniku korisničkog računa otvara se prozor koji sadrži pomoćne instrukcije za lakše i brže korištenje aplikacije. Redom, prevedene s engleskog jezika, upute su:

1. Prečaci na tipkovnici.

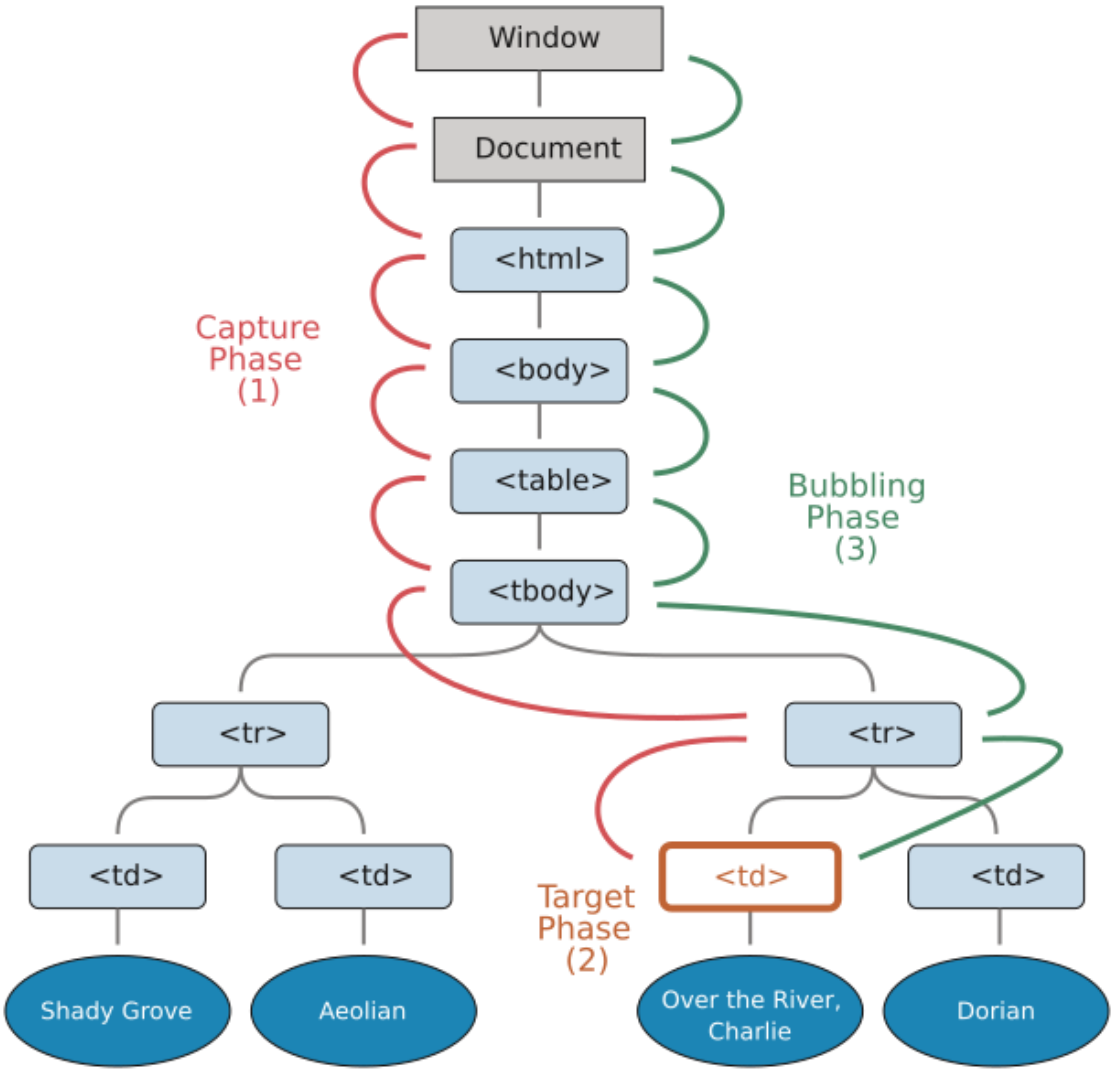
   Svi prečaci na tipkovnici primjenjuju se na trenutno odabranu mapu (trenutno fokusiran radni okvir). Odabrana mapa označena je jačom sjenom u lebdećim prozorima ili drugačije obojenom linijom u stablu mapa. Ako nijedna mapa nije odabrana, prečaci se primjenjuju na trenutno otvorenu mapu.

   - Esc:
     - Poništi odabir svih Item-a i isprazni međuspremnik aplikacije.
     - Zatvori lebdeću mapu.
     - Poništi odabir mape u stablu datoteka.

        - Zaustavi preimenovanje *Item-a*/ukloni fokus s polja.
    - Del: Izbriši odabrane *Item-e*.
    - F2: Preimenuj odabrani *Item*.
    - Ctrl + A: Odaberi sve *Item-e*.
    - Ctrl + I: Obrni odabir.
    - Ctrl + C: Kopiraj odabrane *Item-e*.
    - Ctrl + X: Izreži odabrane *Item-e*.
    - Ctrl + V: Zalijepi *Item-e* iz međuspremnika.
2. Upravljanje tipkovnicom i mišem.
    - Drži Ctrl: Za odabir više *Item-a*.
    - Drži Shift: Za odabir raspona *Item-a* od posljednje odabranog.
    - Drži Ctrl prije ispuštanja: Za kopiranje *Item-a* umjesto premještanja.
3. Upravljanje klikom.
    - Dvaput klikni dokument: Za otvaranje pretpregleda dokumenta.
    - Dvaput klikni mapu: Za otvaranje mape.
    - Dvaput klikni karticu: Za prebacivanje prozora na cijeli zaslon.
    - Srednji klik na karticu: Za zatvaranje kartice.
    - Srednji klik na mapu: Za otvaranje u novoj kartici.
    - Dvaput klikni na desni rubni okvir stabla datoteka: Za automatsko prilagođavanje širine.
4. Upravljanje dodirom.
    - Povuci prstom ulijevo na putanji navigacijske trake: Za navigaciju do roditeljske mape.
    - Povuci prstom udesno na mapi u stablu mapa: Za navigaciju do nje.
    - Dugo pritisni karticu: Za početak povlačenja.
    - Dugo pritisni *Item*, a zatim pomakni u bilo kojem smjeru: Za odabir više *Item-a*.

Ako se aplikacija otvara na mobitelu, redoslijed uputa je obrnut.

Najveći izazov u izradi slušača, što uključuje kontrole, prečace, geste, povlačenja, ispuštanja i dr. je propagacija događaja u DOM-u, odnosno dvije faze: *bubbling* i *capturing* (Slika 4.27).



Slika 4.27 Propagacija događaja u DOM-u web aplikacije
(izvor: I. Kantor, 2024 [46])

Kod *bubbling-a*, događaj se prvo obrađuje na elementu na kojem se dogodio, a zatim se "mjehuri" (kreće) prema gore kroz hijerarhiju roditeljskih elemenata. Kod *capturing-a*, događaj se prvo obrađuje na najvišem elementu u hijerarhiji, a zatim se "spušta" prema dolje do ciljanog elementa. [46]

Većina događaja se obrađuje *bubbling-om*, ali se koristi i *capturing* za hvatanje događaja prije no što dođu do ciljanog elementa. Vrlo često se ti događaji u aplikaciji sprječavaju kako ne bi izvršili nešto što je zadano za preglednik. Na primjer, kartice su sve `<a>` oznake, odnosno *anchor* elementi, što je vidljivo jer se prelaskom kursora preko kartica pojavljuje

puni URL u donjem lijevom kutu preglednika. Klikom na karticu, zadano ponašanje preglednika učitalo bi URL iznova i zato to treba biti spriječeno. U tome pomažu vrlo korisni Vue modifikatori događaja, čijom uporabom se znatno smanjuje količina kôda i metode se mogu baviti isključivo logikom podataka, a ne i detaljima DOM događaja. [47] U nastavku je primjer kôda (Kôd 4.6) na elementu kartice u kojem se koriste modifikatori *middle* i *prevent*.

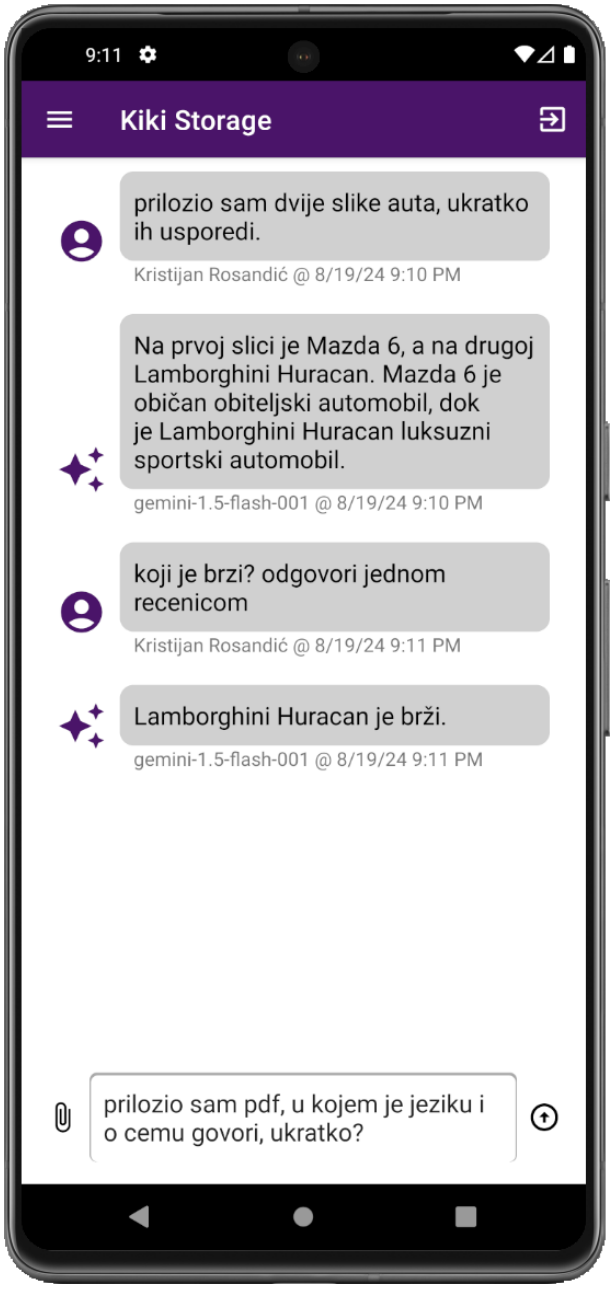
```
@auxclick.middle.prevent="tabsStore.deleteTab(tab)"
@click.prevent="tabsStore.switchTab(tab)"
@dragover="tabsStore.switchTab(tab)"
@dblclick="handleTabDblClick"
```

Kôd 4.6 Vue modifikatori događaja

Znak "@" je kratica za *v-on* direktivu nakon čega dolazi događaj koji se sluša, a zatim idu modifikatori: *middle* označava srednji klik, *prevent* sprječava zadano ponašanje. Gumb X za zatvaranje kartice je DOM potomak (engl. *descendant*) te kartice i na sebi koristi hvatač događaja `@click.stop.prevent="tabsStore.deleteTab(tab)"`. U ovom slučaju se koristi *stop* kako bi se zaustavila propagacija događaja prema gore i zato se ne poziva *click* slušač.

4.1.5. Razgovor s umjetnom inteligencijom

Razgovor s umjetnom inteligencijom dodatna je značajka Android aplikacije (Slika 4.28).



Slika 4.28 Razgovor s umjetnom inteligencijom o priloženim datotekama na Android aplikaciji

Implementirana verzija Vertex AI servisa je Gemini 1.5 Flash. U ovome kao i u "Home" fragmentu koristi se `RecyclerView` za prikaz liste. Lista predstavlja razgovorne "oblačiće". Otvaranjem fragmenta stvara se nova razgovorna sesija. Lijevo od unosa za tekst poruke je ikona za prilaganje jedne ili više datoteka, a implementirana je ponovnom upotrebom `BottomSheetDialogFragment` komponente. Budući da onda *Dialog* fragment ne prima parametar putanje, ne prikazuje opciju stvaranja mape i nakon učitavanja

dokumenata otprema događaj "files_ready". Roditeljski fragment zatim dodaje BLOB datoteke u graditelj sadržaja za razgovor. Slanjem sadržaja generiraju se prvi tokeni u stvarnom vremenu i započinje skupljanje komada (engl. *chunk*) odgovora. Tok (engl. *stream*) odgovora odmah se ispisuje na zaslon. [48]

4.2. Poslužiteljski sloj

Na poslužiteljskom sloju, Cloud Functions servis izrađen je u Node.js okruženju u TypeScript jeziku. Servis koristi Firebase Admin SDK kako bi funkcije imale neograničen pristup svim bazama podataka, odnosno kako bi "zaobišle" definirana pravila svake.

Klijentski SDK-ovi za Cloud Functions servis omogućuju izravno pozivanje funkcija iz Firebase aplikacije. Za to su implementirane HTTP *Callable* funkcije na poslužitelju. Važno je napomenuti da HTTP *Callable* funkcije nisu identične HTTP funkcijama. Za korištenje HTTP *Callable* funkcija nužno je koristiti klijentski SDK za određenu platformu zajedno s *backend* API-jem (ili implementirati protokol). Ključne razlike *Callable* funkcija u odnosu na HTTP funkcije su automatsko uključivanje Firebase Authentication tokena, FCM (*Firebase Cloud Messaging*) tokena i App Check tokena u zahtjeve te automatska deserializacija tijela zahtjeva i validacija autentikacijskih tokena. [49]

Osim što su implementirane *Callable* funkcije specifične za Firebase, implementirane su i regularne HTTP funkcije (REST API funkcije) koje poziva Cloud Scheduler.

Cloud Scheduler implementiran je putem GCP UI-a definiranjem poslova prema zadanoj frekvenciji. Vertex AI API nije potrebno implementirati ni konfigurirati s poslužiteljske strane osim aktivacije Firebase *Blaze* pretplatničkog plana kako bi se dobio pristup.

4.3. Baze podataka

O ovom poglavlju objašnjena je implementacija spremanja podataka u različitim bazama podataka. Sve baze podataka zaštićene su pravilima, a svaka su definirana u zasebnim datotekama za pravila ovisno o bazi podataka koju štite:

- "database.rules.json" štiti Firebase RTDB
- "firestore.rules" štiti Firebase Firestore
- "storage.rules" štiti Cloud Storage

Na primjer, za RTDB pravila su definirana kôdom u nastavku (Kôd 4.7).

```
{
  "rules": {
    "settings": {
      "$uid": {
        ".read": "auth !== null && auth.uid === $uid",
        ".write": "auth !== null && auth.uid === $uid"
      }
    }
  }
}
```

Kôd 4.7 RTDB pravila

Dakle, korisnik mora biti autenticiran te može čitati i pisati u bazu podataka samo na lokaciji "settings/$uid", gdje je UID jedinstveni identifikator korisnika. Inače, UID smije biti javan jer Firebase u pozadini provjerava pripada li UID autentikacijskom objektu na siguran način. Ovo je preporučeni pristup za pisanje Firebase RTDB pravila.

4.3.1. Postavke korisnika

Za postavke korisnika koristi se Firebase RTDB. Postavke korisnika definirane su TypeScript tipom *Settings*, a poslovna logika nalazi se u Pinia trgovini "settings". Na klijentu su definirane zadane postavke. Način na koji se RTDB popunjava je dinamičan, odnosno tek kad korisnik promijeni neku postavku, ona se sprema. Za dobavljanje podataka koristi se *Deep Merge* algoritam jer je potrebno spojiti sve iz baze podataka sa zadanim postavkama na klijentu, čime se štedi prostor i resursi (Kôd 4.8).

```
const settings = computed<Settings>(() => {
  return mergeDeep(getDefaultSettings(), dbSettings.value);
});
```

Kôd 4.8 Učitavanje RTDB postavki

Resetiranjem postavki brišu se svi podaci na korisničkoj putanji u bazi podataka.

4.3.2. Struktura datoteka i mapa korisnika

Datoteke i mape, odnosno svi *Item-i* spremaju se u Firebase Firestore. Podaci se pohranjuju u obliku kolekcija i dokumenata putem izmjeničnog obrasca: kolekcija > dokument > kolekcija > dokument itd. Dakle, nije moguće referencirati kolekciju u kolekciji ili dokument u dokumentu. Moguće je stvoriti hijerarhijsku strukturu koja bi pratila strukturu mapa, ali

tada bi upiti bili puno složeniji i sporiji. Zbog toga se upotrebljava ravna struktura: kolekcije su korisnički identifikatori koji sadrže dokumente (*Item-e*). Dokumenti osim podkolekcija mogu imati polja, koja su također svojstva *Item-a*. Budući da je svojstvo svakog *Item-a* njegova apsolutna putanja, lako je imati brze upite za dohvaćanje i slušanje promjena u stvarnom vremenu. Nedostatak takvog načina rada je spremanje veće količine podataka, a time i veći izlaz iz mreže (engl. *network egress*). Ipak, prednosti su puno veće. Način na koji su svi radni okviri uvijek ažurirani u stvarnom vremenu je putem VueFire kolekcija, koje su definirane u Pinia Firestore trgovini. Dio API-ja kojega ta trgovina izlaže je za dohvaćanje kolekcija *Item-a* prema zadanoj putanji i drugim opcijama kao što je prikazano u kôdu.

```
getItems(
  path: string,
  nestedOnly?: boolean,
  options?: UseCollectionOptions,
) {
  if (path in PATH_ITEM_COLLECTIONS && !nestedOnly)
    return PATH_ITEM_COLLECTIONS[path];
  const coll = useCollection(
    query(
      collection(db, dbPath.value),
      ...(nestedOnly
        ? startsWithConstraints(path)
        : [where("path", "==", path)]),
    ).withConverter(firestoreItemConverter),
    { ...options },
  );
  if (!nestedOnly && !options) PATH_ITEM_COLLECTIONS[path] =
coll;
  return coll;
},
...
function startsWithConstraints(string: string) {
  return [
    where("path", ">=", `${string}`),
    where("path", "<=", `${string}\uf8ff`),
  ];
}
```

Kôd 4.9 Dohvaćanje Item kolekcija iz Firestore baze podataka

Dakle, trgovina sprema *Item* kolekcije te ih dohvaća ili stvara kad je pozvana. U kôdu (Kôd 4.9) navedena je i pomoćna funkcija `startsWithConstraints` koja se koristi za dohvaćanje ograničenja svih *Item-a* koji započinju sa zadanom putanjom. Budući da Firestore nema funkciju za to, trik je u definiraju raspona. Znak `\uf8ff` vrlo je visoka kodna točka u Unicode rasponu, nakon većine uobičajenih znakova. To znači da će dva ograničenja raspona uključiti sve dokumente koji počinju sa zadanim nizom, ali isključiti one koji imaju isti početak i nastavljaju s drugim znakovima. Kod premještaja mapa `nestedOnly` je `true` jer se tada trebaju dohvatiti svi *Item-i* koji započinju sa zadanom putanjom. U tom slučaju se ubacuje i opcionalan VueFire `options` parametar koji postavlja dohvat podataka na jedanput, što znači da će VueFire automatski uništiti Firestore pretplatu čim se kolekcija potpuno dohvati.

4.3.3. Podaci korisnika

Podaci korisnika (binarni veliki objekti, engl. *Binary Large Object*, skraćeno BLOB) spremaju se u Cloud Storage. Ovdje se ne koristi posebna tehnika za organizaciju jer upravo tome služi Firestore. Dakle, binarni podaci svih *Item-a* spremaju se u korisničku listu u Cloud Storage-u s nazivom koji je jednak identifikatoru *Item-a*, generiranom od Firestore servisa.

4.4. Kontinuirana integracija i isporuka

Za kontinuiranu integraciju i isporuku koristi se Github Actions i Firebase Hosting.

GitHub Actions predstavlja platformu za kontinuiranu integraciju i isporuku (*Continuous integration / Continuous deployment*, skraćeno CI/CD) koja omogućuje automatizaciju procesa izgradnje, testiranja i implementacije rješenja. Platforma nudi virtualna računala s Linux, Windows i macOS operativnim sustavima za izvršavanje tijekova rada, a postoji i mogućnost korištenja vlastitih pokretača u podatkovnom centru ili *cloud* infrastrukturi. Komponente GitHub Actions uključuju tijekove rada (engl. *workflows*), koji se sastoje od jednog ili više poslova (engl. *jobs*). Poslovi se izvršavaju sekvencijalno ili paralelno unutar virtualnih računala ili kontejnera te sadrže korake (engl. *steps*) koji izvršavaju skripte ili akcije (engl. *actions*). Tijekovi rada definiraju se YAML datotekama pohranjenim u repozitoriju, a pokreću se događajima u repozitoriju, ručno ili prema unaprijed definiranom

rasporedu. Moguće je i referenciranje jednog tijeka rada unutar drugog, što omogućuje njihovu ponovnu upotrebu. [50]

Web aplikacija upotrebljava GitHub Actions s dva definirana *workflow-a*:

1. "firebase-hosting-merge.yml" – Isporuka na Firebase Hosting nakon što je napravljeno spajanje (engl. *merge*). *Merge* je proces u kojem se promjene iz jedne grane (engl. *branch*) uključuju u drugu granu, obično glavnu granu razvoja.
2. "firebase-hosting-pull-request.yml" – Isporuka na Firebase Hosting nakon što je napravljen zahtjev za spajanje (engl. *pull request*). *Pull request* je zahtjev da se promjene iz jedne grane pregledaju i spoje u drugu granu, što omogućuje suradnju i kontrolu kvalitete koda prije spajanja.

U oba *workflow-a* koraci *job-ova* su isti:

1. Pokretanje akcije koja preuzima GitHub repozitorij (ažurirani izvorni kôd).
2. Postavljanje radne mape na "./web-client".
3. Pokretanje naredbe "npm ci && npm run build", što su zapravo dvije naredbe u nizu kojima se instaliraju ovisnosti i izgrađuje produkcijski kôd.
4. Pokretanje akcije koja isporučuje produkcijski kôd na Firebase Hosting prema zadanim ulaznim parametrima.

Firebase Hosting je produkcijsko web *hosting* rješenje koje stavlja ovu web aplikaciju na CDN, optimizirano za statičke i SPA web aplikacije. [13]

5. Testiranje i analiza rješenja

Za testiranje rješenja upotrebljava se Playwright.

Playwright je alat za pouzdano *end-to-end* (E2E) testiranje modernih web aplikacija. E2E testiranje provjerava cijelu aplikaciju od početka do kraja, oponašajući stvarne korisničke interakcije i podatke. Playwright podržava sve moderne preglednike (Chromium, WebKit, Firefox) i omogućuje testiranje na različitim platformama (Windows, Linux, macOS) i jezicima (TypeScript, JavaScript, Python, .NET). Nadalje, omogućuje izolirano testiranje s brzim izvršavanjem, podržava složene scenarije s višestrukim korisnicima i kontekstima te pruža moćne alate poput generiranja testova, inspektora stranica i pregleda tragova za analizu grešaka. [51]

Za analizu rješenja proveden je anketni upitnik kojim su prikupljena iskustva petnaest korisnika pri korištenju web aplikacije. Korišten je Google Forms, a korisnicima je bilo zadano da ocijene glavne značajke ovoga web rješenja u odnosu na bilo koje drugo postojeće web rješenje. Svako pitanje imalo je ponuđeno pet ocjena:

- Ocjena 1: Znatno lošije
- Ocjena 2: Lošije
- Ocjena 3: Jednako
- Ocjena 4: Bolje
- Ocjena 5: Znatno bolje

Analizom odgovora uočeno je visoko zadovoljstvo korisnika, kako na računalima tako i na mobilnim uređajima. Od ukupno 195 ocjena, prosjek je **4,65**, a ovo su pitanja i prosjeci odgovora:

1. Kako biste ocijenili kartični prikaz na *desktop* verziji web aplikacije? **4,93**
2. Kako biste ocijenili kartični prikaz na mobilnoj verziji web aplikacije? **4,40**
3. Kako biste ocijenili stablo mapa na *desktop* verziji web aplikacije? **4,73**
4. Kako biste ocijenili stablo mapa na mobilnoj verziji web aplikacije? **4,40**
5. Kako biste ocijenili navigacijsku traku na *desktop* verziji web aplikacije? **4,93**
6. Kako biste ocijenili navigacijsku traku na mobilnoj verziji web aplikacije? **4,60**
7. Kako biste ocijenili pretragu na *desktop* verziji web aplikacije? **4,33**
8. Kako biste ocijenili pretragu na mobilnoj verziji web aplikacije? **4,2**0

9. Kako biste ocijenili personalizaciju na *desktop* verziji web aplikacije? **4,93**
10. Kako biste ocijenili personalizaciju na mobilnoj verziji web aplikacije? **5,00**
11. Kako biste ocijenili brzinu učitavanja cijele aplikacije na *desktop* verziji web aplikacije? **4,60**
12. Kako biste ocijenili brzinu učitavanja cijele aplikacije na mobilnoj verziji web aplikacije? **4,46**
13. Kako biste ocijenili sinkronizaciju podataka i postavki između *desktop* i mobilne verzije web aplikacije? **4,93**

U pitanjima se "verzija" odnosi na istu web aplikaciju: *desktop* verzija jest ona otvorena na web pregledniku računala, a mobilna na pregledniku mobitela. Preporučeni web preglednik za oba uređaja jest Chrome. Prosjek ocjena na računalu iznosi **4,74**, dok je na mobitelu **4,51**.

Na otvoreno pitanje najpoželjnije značajke koja nedostaje, najviše korisnika odgovorilo je pretpregled (engl. *preview*) datoteka i dijeljenje. No zbog složenosti implementacije tih značajki, osobito pretpregleda, koji bi idealno zahtijevao podršku za gotovo beskonačan broj formata datoteka, te uzimajući u obzir već značajan opseg ovog rada, njihova potpuna realizacija ostavljena je za buduće napore. Ipak, činjenica da su temelji za ove funkcionalnosti već postavljeni u aplikaciji svjedoči o predviđanju i planiranju njezina budućeg razvoja, otvarajući mogućnosti za daljnja istraživanja i implementaciju u budućim verzijama.

Sveukupno, rezultati ankete pokazuju da su korisnici dobro prihvatili razvijenu web aplikaciju. Iako postoji prostor za daljnja unaprjeđenja, visoke ocjene potvrđuju da je aplikacija uspjela riješiti neke od glavnih nedostataka postojećih servisa.

Zaključak

Ovaj rad predstavlja sveobuhvatan pristup razvoju inovativnoga rješenja za pohranu u oblaku, usmjerena na poboljšanje korisničkoga iskustva i prevladavanje ograničenja postojećih servisa. Detaljnom analizom problema i implementacijom ključnih funkcionalnosti rad je uspješno postigao svoje ciljeve, istovremeno demonstrirajući snagu i fleksibilnost modernih tehnologija.

U uvodnom dijelu rada istaknuta je važnost učinkovite pohrane i upravljanja datotekama u oblaku u današnjem digitalnom dobu. Unatoč širokoj upotrebi popularnih alata poput Dropbox-a i Microsoft OneDrive-a, njihova sučelja često ne uspijevaju pružiti dovoljnu fleksibilnost i intuitivnost, osobito pri radu s velikim količinama podataka.

Identifikacijom ključnih izazova, poput nedostatka kartičnoga prikaza i ograničene funkcionalnosti stabla mapa, rad je postavio temelje za razvoj rješenja koje će redefinirati način na koji korisnici pristupaju organizaciji podataka u oblaku. Uvođenjem kartičnoga prikaza s putanjama, naprednoga stabla mapa, sinkronizacije podataka u stvarnom vremenu i opsežnih mogućnosti personalizacije, korisnicima je omogućena brža i intuitivnija navigacija kroz datoteke i mape, prilagođena njihovim individualnim potrebama.

Arhitektura sustava, temeljena na Google Cloud Platformi, osigurava skalabilnost, sigurnost i performanse potrebne za podršku rastućem broju korisnika i količini podataka. Korištenje Firebase servisa za autentikaciju, pohranu podataka i *serverless* funkcije omogućuje učinkovito upravljanje korisničkim računima, podacima i važnim operacijama.

Implementacija sustava istaknula je snagu Vue.js programskog okvira i TypeScript jezika u izgradnji naprednoga i responzivnoga korisničkog sučelja za web aplikaciju. S druge strane, Android aplikacija, izgrađena u Kotlinu, pokazala je kako se mogu postići osnovne funkcionalnosti na mobilnoj platformi. Korištenje Pinia trgovina, Vue Router-a, VueFire-a, ali i naprednih CSS tehnologija poput CSS *Grid* i *Subgrid* prikaza omogućilo je stvaranje dinamičnoga i responzivnoga sučelja, prilagođenog različitim uređajima i veličinama ekrana. Upravo je istraživanje i primjena CSS *Subgrid-a*, relativno nove tehnologije, ukazalo na važnost praćenja i primjene najnovijih dostignuća u web razvoju kako bi se postiglo optimalno korisničko iskustvo i performanse. Također, tijekom razvoja, naišlo se na specifičnu Vue Core grešku koja je ukazala na dinamičnost razvoja modernih radnih okvira i važnost aktivne zajednice u otkrivanju i rješavanju takvih problema.

Primjena progresivnih web aplikacija (PWA) omogućila je ovom rješenju brže učitavanje i ujednačeno iskustvo na svim uređajima, uz napredne funkcionalnosti poput izravnoga pristupa datotečnom sustavu. Ta implementacija PWA rješenja demonstrira prednosti modernih web tehnologija u izgradnji aplikacija, koje pružaju iskustvo slično nativnim aplikacijama.

Integracija naprednih značajki, poput razgovora s umjetnom inteligencijom na Android aplikaciji, dodatno je obogatila funkcionalnost te ukazuje na mogućnost integracije s podacima i na web aplikaciji. Također, ostavljena je mogućnost za buduće nadogradnje poput dijeljenja, kopiranja i pretpregleda datoteka, ukazujući na potencijal za daljnji razvoj i prilagodbu rješenja specifičnim potrebama korisnika.

U konačnici, ovaj rad je uspješno razvio inovativno rješenje za pohranu u oblaku koje će poboljšati način na koji korisnici organiziraju i upravljaju svojim podacima. Kombinacijom naprednih tehnologija, promišljenim dizajnom i implementacijom novih funkcionalnosti, rad je postavio nove standarde u području pohrane u oblaku. Istovremeno, rad je pokazao kako se korištenjem najnovijih tehnologija i aktivnim sudjelovanjem u zajednici razvojnih programera mogu prevladati izazovi u razvoju i stvoriti rješenje koje će se nastaviti razvijati i prilagođavati potrebama korisnika u budućnosti.

Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja, niti generiranjem od strane AI alata. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključujući i poništenje javne isprave stečene dijelom i na temelju ovoga rada.

U Zagrebu, 25. 8. 2024.

Kristijan Rosandić

Popis kratica

API	*Application Programming Interface*	Programsko sučelje aplikacije
AI	*Artificial Intelligence*	Umjetna inteligencija
BLOB	*Binary Large Object*	Binarni veliki objekt
CD	*Continuous deployment*	Kontinuirano postavljanje
CDN	*Content Delivery Network*	Mreža za isporuku sadržaja
CI	*Continuous integration*	Kontinuirana integracija
CLI	*Command Line Interface*	Sučelje komandne linije
CSR	*Client Side Rendering*	Renderiranje na strani klijenta
CSS	*Cascading Style Sheets*	Kaskadni stilski listovi
DOM	*Document Object Model*	Model objekta dokumenta
E2E	*End-to-End*	Od kraja do kraja
FCM	*Firebase Cloud Messaging*	Firebase cloud poruke
GCP	*Google Cloud Platform*	Google cloud platforma
HMR	*Hot Module Replacement*	Zamjena vrućeg modula
HTTP	*HyperText Transfer Protocol*	HyperText protokol za prijenos
HTTPS	*HyperText Transfer Protocol Secure*	Siguran HyperText protokol za prijenos
IDE	*Integrated Development Environment*	Integrirano razvojno okruženje
JSON	*JavaScript Object Notation*	JavaScript objektna notacija
MVC	*Model View Controller*	Model Prikaz Kontroler
MVVM	*Modal View ViewModel*	Model prikaz prikazModel
NPM	*Node Package Manager*	Upravitelj Node paketa
PWA	*Progressive Web Application*	Progresivna web aplikacija
REST	*Representational State Transfer*	Representativni prijenos stanja
RTDB	*Firebase Realtime Database*	Firebase baza podataka u stvarnom vremenu
SDK	*Software Development Kit*	Komplet za razvoj softvera
SEO	*Search Engine Optimization*	Optimizacija za tražilice
SFC	*Single File Component*	Komponenta jedne datoteke
SPA	*Single Page Application*	Aplikacija jedne stranice
SQL	*Structured Query Language*	Strukturirani jezik upita
SSL	*Secure Sockets Layer*	Sloj sigurnih priključaka
SVG	*Scalable Vector Graphics*	Skalabilna vektorska grafika
TLS	*Transport Layer Security*	Sigurnost transportnog sloja

UID	*Unique Identifier*	Jedinstveni identifikator
UI	*User Interface*	Korisničko sučelje
URL	*Uniform Resource Locator*	Jedinstveni lokator resursa
UUID	*Universally Unique Identifier*	Univerzalni jedinstveni identifikator
XML	*Extensible Markup Language*	Proširivi jezik za označavanje

Popis slika

Popis kôdova

Literatura

[1] Google, »Welcome to Learn Progressive Web Apps!,« [Mrežno]. Dostupno: https://web.dev/learn/pwa/welcome. [Pokušaj pristupa August 2024].

[2] Vue.js Team, »Using Vue with TypeScript,« [Mrežno]. Dostupno: https://vuejs.org/guide/typescript/overview. [Pokušaj pristupa August 2024].

[3] Mozilla, »SPA (Single-page application),« [Mrežno]. Dostupno: https://developer.mozilla.org/en-US/docs/Glossary/SPA. [Pokušaj pristupa August 2024].

[4] P. Ram, »Server Side Rendering (SSR) vs. Client Side Rendering (CSR) vs. Pre-Rendering using Static Site Generators (SSG) and client-side hydration.,« October 2021. [Mrežno]. Dostupno: https://medium.com/@prashantramnyc/server-side-rendering-ssr-vs-client-side-rendering-csr-vs-pre-rendering-using-static-site-89f2d05182ef. [Pokušaj pristupa August 2024].

[5] Google, »Cloud Functions for Firebase,« [Mrežno]. Dostupno: https://firebase.google.com/docs/functions. [Pokušaj pristupa August 2024].

[6] Google, »Google Cloud Scheduler documentation,« [Mrežno]. Dostupno: https://cloud.google.com/scheduler/docs. [Pokušaj pristupa August 2024].

[7] Google, »Introduction to Vertex AI,« [Mrežno]. Dostupno: https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform. [Pokušaj pristupa August 2024].

[8] Google, »Firebase Realtime Database,« [Mrežno]. Dostupno: https://firebase.google.com/docs/database. [Pokušaj pristupa August 2024].

[9] Google, »Cloud Firestore,« [Mrežno]. Dostupno: https://firebase.google.com/docs/firestore. [Pokušaj pristupa August 2024].

[10] Google, »Cloud Storage for Firebase,« [Mrežno]. Dostupno: https://firebase.google.com/docs/storage. [Pokušaj pristupa August 2024].

[11] Google, »What is a NoSQL database?,« [Mrežno]. Dostupno: https://cloud.google.com/discover/what-is-nosql. [Pokušaj pristupa August 2024].

[12] Google, »Firebase Authentication,« [Mrežno]. Dostupno: https://firebase.google.com/docs/auth. [Pokušaj pristupa August 2024].

[13] Google, »Firebase Hosting,« [Mrežno]. Dostupno: https://firebase.google.com/docs/hosting. [Pokušaj pristupa August 2024].

[14] Vue.js Team, »Performance,« [Mrežno]. Dostupno: https://vuejs.org/guide/best-practices/performance. [Pokušaj pristupa August 2024].

[15] Mozilla, »CSS performance optimization,« [Mrežno]. Dostupno: https://developer.mozilla.org/en-US/docs/Learn/Performance/CSS. [Pokušaj pristupa August 2024].

[16] npm, »About npm,« [Mrežno]. Dostupno: https://docs.npmjs.com/about-npm. [Pokušaj pristupa August 2024].

[17] Gradle, »Gradle Basics,« [Mrežno]. Dostupno: https://docs.gradle.org/current/userguide/gradle_basics.html. [Pokušaj pristupa August 2024].

[18] VueUse Team, »Get Started,« [Mrežno]. Dostupno: https://vueuse.org/guide/. [Pokušaj pristupa August 2024].

[19] D. M. Hendricks, »File Icon Vectors,« [Mrežno]. Dostupno: https://github.com/dmhendricks/file-icon-vectors. [Pokušaj pristupa August 2024].

[20] Google, »firebaseui-web,« [Mrežno]. Dostupno: https://github.com/firebase/firebaseui-web. [Pokušaj pristupa August 2024].

[21] Vue.js Team, »Pinia Introduction,« [Mrežno]. Dostupno: https://pinia.vuejs.org/introduction.html. [Pokušaj pristupa August 2024].

[22] Vue.js Team, »Introduction,« [Mrežno]. Dostupno: https://vuejs.org/guide/introduction.html. [Pokušaj pristupa August 2024].

[23] Vue.js Team, »Rendering Mechanism,« [Mrežno]. Dostupno: https://vuejs.org/guide/extras/rendering-mechanism. [Pokušaj pristupa August 2024].

[24] Vue.js Team, »Single-File Components,« [Mrežno]. Dostupno: https://vuejs.org/guide/scaling-up/sfc.html. [Pokušaj pristupa August 2024].

[25] Vue.js Team, »Composition API FAQ,« [Mrežno]. Dostupno: https://vuejs.org/guide/extras/composition-api-faq.html. [Pokušaj pristupa August 2024].

[26] Vue.js Team, »Vue Router Introduction,« [Mrežno]. Dostupno: https://router.vuejs.org/introduction.html. [Pokušaj pristupa August 2024].

[27] J. Simonds, »Vue Slicksort Introduction,« [Mrežno]. Dostupno: https://vue-slicksort.netlify.app/introduction. [Pokušaj pristupa August 2024].

[28] Vue.js Team, »VueFire Introduction,« [Mrežno]. Dostupno: https://vuefire.vuejs.org/guide/. [Pokušaj pristupa August 2024].

[29] PostCSS Team, »autoprefixer,« [Mrežno]. Dostupno: https://github.com/postcss/autoprefixer. [Pokušaj pristupa August 2024].

[30] DaisyUI Team, »daisyUI — Tailwind CSS Components,« [Mrežno]. Dostupno: https://daisyui.com/. [Pokušaj pristupa August 2024].

[31] PostCSS Team, »PostCSS - a tool for transforming CSS with JavaScript,« [Mrežno]. Dostupno: https://postcss.org/. [Pokušaj pristupa August 2024].

[32] Prettier Team, »What is Prettier?,« [Mrežno]. Dostupno: https://prettier.io/docs/en/. [Pokušaj pristupa August 2024].

[33] Tailwind Labs, »prettier-plugin-tailwindcss,« [Mrežno]. Dostupno: https://github.com/tailwindlabs/prettier-plugin-tailwindcss. [Pokušaj pristupa August 2024].

[34] D. Bardadym, »rollup-plugin-visualizer,« [Mrežno]. Dostupno: https://github.com/btd/rollup-plugin-visualizer. [Pokušaj pristupa August 2024].

[35] Tailwind Labs, »Get started with Tailwind CSS,« [Mrežno]. Dostupno: https://tailwindcss.com/docs. [Pokušaj pristupa August 2024].

[36] Microsoft, »TypeScript Documentation,« [Mrežno]. Dostupno: https://www.typescriptlang.org/docs/. [Pokušaj pristupa August 2024].

[37] Vite Team, »Getting Started | Vite,« [Mrežno]. Dostupno: https://vitejs.dev/guide/. [Pokušaj pristupa August 2024].

[38] Google, »Understand the One Tap user experience,« [Mrežno]. Dostupno: https://developers.google.com/identity/gsi/web/guides/features. [Pokušaj pristupa August 2024].

[39] Mozilla, »CSS values and units,« [Mrežno]. Dostupno: https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units. [Pokušaj pristupa August 2024].

[40] DaisyUI Team, »Colors,« [Mrežno]. Dostupno: https://daisyui.com/docs/colors/. [Pokušaj pristupa August 2024].

[41] wujekbogdan, »Add ability to destroy stores,« [Mrežno]. Dostupno: https://github.com/vuejs/pinia/issues/557#issuecomment-870615336. [Pokušaj pristupa August 2024].

[42] Google, »FirebaseUI for Cloud Firestore,« [Mrežno]. Dostupno: https://firebaseopensource.com/projects/firebase/firebaseui-android/firestore/readme/. [Pokušaj pristupa August 2024].

[43] Vue.js Team, »Provide / Inject,« [Mrežno]. Dostupno: https://vuejs.org/guide/components/provide-inject. [Pokušaj pristupa August 2024].

[44] Vue.js Team, »Reactivity API: Core,« [Mrežno]. Dostupno: https://vuejs.org/api/reactivity-core.html. [Pokušaj pristupa August 2024].

[45] Mozilla, »Subgrid - CSS,« [Mrežno]. Dostupno: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout/Subgrid. [Pokušaj pristupa August 2024].

[46] I. Kantor, »Bubbling and capturing,« [Mrežno]. Dostupno: https://javascript.info/. [Pokušaj pristupa August 2024].

[47] Vue.js Team, »Event Modifiers,« [Mrežno]. Dostupno: https://vuejs.org/guide/essentials/event-handling#event-modifiers. [Pokušaj pristupa August 2024].

[48] Google, »Gemini API using Vertex AI in Firebase,« [Mrežno]. Dostupno: https://firebase.google.com/docs/vertex-ai. [Pokušaj pristupa August 2024].

[49] Google, »Call functions from your app | Cloud Functions for Firebase,« [Mrežno]. Dostupno: https://firebase.google.com/docs/functions/callable. [Pokušaj pristupa August 2024].

[50] GitHub, »Understanding GitHub Actions,« [Mrežno]. Dostupno: https://docs.github.com/en/actions/about-github-actions/understanding-github-actions. [Pokušaj pristupa August 2024].

[51] Microsoft, »Playwright - Fast and reliable end-to-end testing for modern web apps,« [Mrežno]. Dostupno: https://playwright.dev/. [Pokušaj pristupa August 2024].