1. Firebase Realtime Database integracijos aprašymas

Projekte „Užrašų aplikacija“ (4užduotis) naudojama „Firebase Realtime Database“ technologija duomenų saugojimui ir sinchronizavimui debesyje. Firebase pasirinkta dėl paprastos integracijos su „Angular“ karkasu, greito duomenų apdorojimo bei automatinio sinchronizavimo tarp kliento ir serverio be atskiros „backend“ infrastruktūros.

1. **Firebase integracija:** realizuota per REST API sąsają naudojant HttpClient.
2. **Duomenų bazės adresas:** <https://fir-uzduotis-default-rtdb.firebaseio.com/>
3. **Operacijos:**

* GET – duomenų nuskaitymas (loadNotes())
* POST – naujo įrašo kūrimas (addNote())
* DELETE – įrašo šalinimas (deleteNote())

1. **Atnaujinimas:** duomenys rodomi tiesiogiai iš notes masyvo, kuris sinchronizuojamas su Firebase po kiekvienos operacijos.
2. Integracijos principas

Šiame projekte Firebase prijungta **tiesiogiai per REST API sąsają**, naudojant HttpClient iš @angular/common/http bibliotekos. Tai reiškia, kad vietoje papildomų paketų (@angular/fire) naudojamas lengvas, rankiniu būdu parašytas API ryšys tarp Angular programos ir Firebase duomenų bazės.

Firebase duomenų bazės pagrindinis adresas (endpoint) nurodytas AppComponent klasėje:

|  |
| --- |
| private readonly BASE = 'https://fir-uzduotis-default rtdb.firebaseio.com/'; |

1. Duomenų struktūra

Firebase Realtime Database duomenų struktūra paremta JSON formatu. Šiame projekte visi duomenys saugomi po šaknimi „notes“, kur kiekvienas įrašas turi unikalų identifikatorių, automatiškai sukuriamą Firebase sistemos.

|  |
| --- |
| {  "notes": {  "-NslA8d73kF4": {  "title": "Pirmasis užrašas",  "text": "Tai testinis įrašas",  "createdAt": "2025-10-22 15:32:10"  },  "-NslA9Bg3xT1": {  "title": "Antrasis užrašas",  "text": "Dar vienas pavyzdys",  "createdAt": "2025-10-22 15:35:48"  }  }  } |

Paveikslėlis, kuriame yra tekstas, ekrano kopija, programinė įranga

Dirbtinio intelekto sugeneruotas turinys gali būti neteisingas.

1 pav. Realtime Database

1. Funkcionalumo įgyvendinimas
   1. Duomenų nuskaitymas (GET)

Duomenų atvaizdavimui naudojama loadNotes() funkcija, kuri siunčia GET užklausą į Firebase REST API ir grąžina visus įrašus iš lentelės „notes“. Gauti duomenys transformuojami į lokalią notes masyvo struktūrą, kuri vėliau naudojama duomenų atvaizdavimui naudotojo sąsajoje.

|  |
| --- |
| loadNotes(): void {      this.http.get<FirebaseMap | null>(`${this.BASE}/notes.json`).subscribe({        next: (data) => {          this.notes = data            ? Object.entries(data).map(([id, note]) => ({ id, ...note }))            : [];        },        error: (err) => console.error('GET notes failed:', err),      });    } |

Ši funkcija siunčia GET užklausą į Firebase REST API ir grąžina visus įrašus iš lentelės „notes“. Gauti duomenys transformuojami į lokalią notes masyvo struktūrą, kuri vėliau naudojama duomenų atvaizdavimui naudotojo sąsajoje.

* 1. Naujo įrašo pridėjimas (POST)

Naujo užrašo sukūrimas atliekamas per addNote() funkciją:

|  |
| --- |
| this.http.post<{ name: string }>(`${this.BASE}/notes.json`, payload).subscribe({        next: (res) => {          const id = res.name;          this.notes.unshift({ id, ...payload });          this.title = '';          this.text = '';        },        error: (err) => console.error('POST note failed:', err),      }); |

* Duomenys siunčiami POST užklausa į /notes.json adresą.
* Firebase automatiškai sukuria įrašo identifikatorių (name).
* Ką tik pridėtas įrašas iš karto įtraukiamas į notes masyvą, todėl sąrašas naršyklėje atsinaujina be papildomo perkrovimo.
  1. Įrašo ištrynimas (DELETE)

Esamų duomenų pašalinimas atliekamas funkcijoje deleteNote():

|  |
| --- |
| deleteNote(id: string): void {      this.http.delete<void>(`${this.BASE}/notes/${id}.json`).subscribe({        next: () => {          this.notes = this.notes.filter((n) => n.id !== id);        },        error: (err) => console.error('DELETE note failed:', err),      });    } |

Pašalinimas vykdomas siunčiant DELETE užklausą į konkretų Firebase įrašo adresą (/notes/{id}.json). Sėkmingai ištrynus įrašą iš Firebase, jis taip pat pašalinamas iš lokalaus masyvo, todėl duomenys vartotojo ekrane sinchronizuojami iš karto.

1. Lentelės atnaujinimas realiu laiku

Kadangi šioje implementacijoje naudojamas tiesioginis REST API būdas, duomenų atnaujinimas vyksta rankiniu būdu – iškviečiant loadNotes() funkciją po duomenų įkėlimo, trynimo ar atnaujinimo. Jeigu reikėtų pilno realaus laiko sinchronizavimo, būtų galima panaudoti Firebase WebSocket kanalą arba oficialų @angular/fire modulį su onValue() stebėjimu.

1. Duomenų koregavimas

Jeigu reikėtų įgyvendinti įrašų redagavimą (duomenų atnaujinimą), būtų galima pridėti papildomą funkciją:

|  |
| --- |
| updateNote(id: string, updatedNote: NewNote): void {  this.http.put(`${this.BASE}/notes/${id}.json`, updatedNote).subscribe({  next: () => this.loadNotes(),  error: (err) => console.error('UPDATE note failed:', err),  });  } |

Ši funkcija naudotų PUT užklausą ir leistų keisti esamus Firebase įrašus pagal id.  
Tokiu būdu vartotojas galėtų redaguoti užrašus lentelėje, o pakeitimai būtų išsaugomi debesyje.