

**Univerzitet u Nišu**

**Elektronski fakultet**

**Katedra za računarstvo**

**Seminarski rad**

**Supabase baza podataka kao primer Database-as-a-Service rešenja**

**Sistemi za upravljanje bazama podataka**

|  |  |  |
| --- | --- | --- |
| **Mentor**: |  | **Student**: |
| Dr Aleksandar Stanimirović |  | Miloš Veljanovski 1559 |

**Niš, jun 2023.**

**Sadržaj**

[1. Koncept Database-as-a-Service rešenja 3](#_Toc138797337)

[2. Supabase 5](#_Toc138797338)

[2.1. Uvod 5](#_Toc138797339)

[2.2. Poređenje sa Firebase-om 5](#_Toc138797340)

[2.3. Arhitektura i osnovni tehnološki stek 6](#_Toc138797341)

[2.4. Supabase-ovi proizvodni principi 9](#_Toc138797342)

[3. Rad sa Supabase-om 11](#_Toc138797343)

[3.1. Demo projekat 11](#_Toc138797344)

[3.2. CRUD operacije korišćenjem C# klijentske biblioteke 14](#_Toc138797345)

[3.2.1. Dodavanje podataka 15](#_Toc138797346)

[3.2.2. Pribavljanje podataka 15](#_Toc138797347)

[3.2.3. Ažuriranje podataka 16](#_Toc138797348)

[3.2.4. Brisanje podataka 17](#_Toc138797349)

[3.2.5. „Upsert“ podataka 17](#_Toc138797350)

[3.3. Primeri korišćenja filtera u upitima 18](#_Toc138797351)

[3.3.1. Filtriranje kolona koje se podudaraju sa šablonom 18](#_Toc138797352)

[3.3.2. Filtriranje kolona koje se nalaze u nizu 18](#_Toc138797353)

[3.3.3. Filtriranje kolona koje sadrže svaki element iz niza 18](#_Toc138797354)

[3.4. Primeri korišćenja modifikatora 19](#_Toc138797355)

[3.4.1. Sortiranje rezultata 19](#_Toc138797356)

[3.4.2. Ograničavanje broja vraćenih redova 19](#_Toc138797357)

[3.5. Bezbednost na nivou reda (RLS) 20](#_Toc138797358)

[3.5.1. RLS polise 20](#_Toc138797359)

[3.5.2. Primeri RLS polisa 21](#_Toc138797360)

[4. Supabase cenovnik 23](#_Toc138797361)

[5. Literatura 25](#_Toc138797362)

[6. Listing 26](#_Toc138797363)

# 1. Koncept Database-as-a-Service rešenja

Baza podataka kao usluga (DBaaS) je usluga računarstva u oblaku koja omogućava korisnicima pristup i upravljanje bazama podataka bez potrebe za postavljanjem hardvera ili instaliranjem softvera. Pružaoci DBaaS usluga brinu se o svemu što se odnosi na bazu podataka, kao što su obezbeđivanje, licenciranje, podrška i održavanje. Korisnici mogu pristupati i manipulisati podacima programski, putem API-ja, ili alata za administraciju baza podataka.

DBaaS se može koristiti za različite vrste baza podataka, kao što su SQL ili NoSQL. SQL baze podataka su relacione baze podataka koje skladište podatke u strukturiranim tabelama i koriste upitni jezik za izvršavanje operacija. NoSQL baze podataka su nerelacione baze podataka koje skladište podatke u fleksibilnim formatima, kao što su dokumenti, grafovi ili parovi ključ-vrednost. NoSQL baze podataka su pogodnije za nestrukturirane ili kompleksne podatke, kao što su video snimci, audio zapisi ili društvene mreže.

Glavni primeri SQL DBaaS rešenja su:

* Amazon Relational Database Service (RDS): Usluga koja nudi šest popularnih SQL baza podataka, kao što su MySQL, PostgreSQL i Oracle.
* Azure SQL Database: Usluga koja pruža potpuno upravljenu SQL Server bazu podataka sa ugrađenim funkcionalnostima i sigurnosnim funkcijama.
* Google Cloud SQL: Usluga koja omogućava korisnicima da pokreću MySQL, PostgreSQL ili SQL Server baze podataka na Google Cloud Platformi.

Neki primeri NoSQL DBaaS rešenja su:

* MongoDB Atlas: Usluga koja nudi potpuno upravljenu MongoDB bazu podataka sa globalnom skalabilnošću, sigurnošću i optimizacijom performansi.
* Amazon DynamoDB: Usluga koja pruža brzu i fleksibilnu ključ-vrednost i dokument bazu podataka sa visokom dostupnošću i doslednošću.
* Azure Cosmos DB: Usluga koja omogućava korisnicima da grade aplikacije sa više NoSQL modela, kao što su MongoDB, Cassandra ili Gremlin.

Benefiti DBaaS usluga jesu:

* Nema dodatnog hardvera na lokaciji: Korisnici mogu uštedeti troškove i prostor izbegavajući potrebu za kupovinom i održavanjem fizičkih servera i uređaja za skladištenje.
* Brzo implementiranje baza podataka: Korisnici mogu brzo i jednostavno obezbediti baze podataka na oblak platformi po svom izboru, bez čekanja na pomoć ili odobrenje IT sektora.
* Priprema za buduće operacije sa podacima: Korisnici mogu iskoristiti moć i fleksibilnost oblaka za upravljanje velikim i kompleksnim skupovima podataka, izgradnju aplikacija u stvarnom vremenu i skaliranje baza podataka prema potrebi.

Neki nedostaci DBaaS usluga su:

* Sigurnost i privatnost podataka: Korisnici mogu imati brige u vezi sa bezbednošću i poverljivošću svojih podataka na oblaku, posebno ako moraju da se pridržavaju određenih propisa ili standarda.
* Zaključavanje za dobavljača usluge: Korisnici mogu naići na teškoće ili troškove pri prebacivanju svojih podataka ili aplikacija sa jednog pružaoca usluga na drugog ili nazad na lokalno okruženje.
* Performanse i dostupnost: Korisnici mogu imati problema sa kašnjenjem ili prekidima usled problema sa mrežom, prestanka usluge ili aktivnostima održavanja.

A picture containing text, font, diagram, screenshot

Description automatically generated

Slika 1. Šema DBaaS arhitekture u oblaku

# 2. Supabase

## **2.1. Uvod**

Supabase je bez-serverska, open-source alternativa za Firebase, izgrađena na osnovu PostgreSQL baze podataka. Pruža sve backend usluge potrebne za kreiranje full-stack aplikacija. Kao korisnik, možete upravljati svojom bazom podataka putem Supabase interfejsa, što uključuje kreiranje tabela i veza, pisanje SQL upita i real-time engine na vrhu PostgreSQL-a.

Supabase dolazi sa zaista sjajnim funkcionalnostima koje olakšavaju razvoj full-stack aplikacija. Neke od tih funkcionalnosti su:

* Sigurnost na nivou reda (RLS) - Supabase dolazi sa PostgreSQL RLS funkcionalnošću koja vam omogućava da ograničite redove u tabelama baze podataka. Kada kreirate ograničenja, to činite direktno putem SQL-a.
* Real-time baza podataka - Supabase ima mogućnost ažuriranja na PostgreSQL bazi podataka koja se može koristiti za praćenje promena u realnom vremenu.
* Supabase UI - Supabase ima open-source biblioteku korisničkog interfejsa za brzo i efikasno kreiranje aplikacija.
* Autentifikacija korisnika - Supabase automatski kreira tabelu auth.users čim kreirate svoju bazu podataka. Kada kreirate aplikaciju, Supabase će takođe dodeliti korisnika i ID čim se registrujete na aplikaciji, što može biti referencirano unutar baze podataka. Za metode prijave, postoje različiti načini autentifikacije korisnika, kao što su email, lozinka, magični linkovi, Google, GitHub i drugi.
* Edge funkcije - Edge funkcije su TypeScript funkcije distribuirane globalno na edge serverima, blizu korisnika. Mogu se koristiti za obavljanje funkcija kao što su integracija sa trećim stranama ili osluškivanje WebHook-ova.

## **2.2. Poređenje sa Firebase-om**

Supabase je platforma bazirana na klaudu koja ima za cilj da pruži kompletno rešenje za razvoj veb aplikacija, slično kao Firebase. Međutim, za razliku od Firebase-a koji se oslanja na NoSQL baze podataka i proprietarne usluge, Supabase je izgrađen na otvorenim tehnologijama, kao što su PostgreSQL, PostgREST i Realtime.

Dakle, Firebase koristi NoSQL bazu podataka koja se naziva Firestore, a koja podatke čuva kao kolekcije dokumenata. Dokumenti su fleksibilni i nemaju unapred definisane šeme, što znači da mogu imati različita polja i strukture. To čini Firestore pogodnim za čuvanje nestrukturiranih ili dinamičnih podataka, kao što su sadržaj koji generišu korisnici ili ažuriranja u realnom vremenu. Sa druge strane, Supabase koristi relacionu bazu podataka koja se naziva PostgreSQL, a koja podatke čuva kao tabele sa redovima i kolonama. Tabele imaju unapred definisane šeme, što znači da imaju fiksna polja i tipove. To čini PostgreSQL pogodnim za čuvanje strukturiranih ili konzistentnih podataka, kao što su transakcije ili spisi.

Jedna od glavnih karakteristika Supabase-a je da nudi potpuno upravljenu PostgreSQL bazu podataka, koja je popularna i moćna relaciona baza podataka. PostgreSQL podržava napredne funkcionalnosti, kao što su transakcije, okidači, pogledi, strani ključevi i JSON tipovi podataka. Supabase takođe pruža API sloj iznad PostgreSQL-a, koristeći PostgREST, što omogućava programerima izvršavanje CRUD operacija, autentifikaciju, autorizaciju i prilagođene funkcije putem jednostavnih HTTP zahteva. Osim toga, Supabase omogućava pretplate u realnom vremenu na promene u bazi podataka, koristeći Realtime, koji je server koji transformiše PostgreSQL-ov WAL tok u JSON poruke.

Još jedna karakteristika Supabase-a je da nudi uslugu skladištenja za velike datoteke, kao što su slike i video zapisi. Usluga skladištenja koristi S3-kompatibilno skladište objekata i pruža API za otpremanje, preuzimanje i brisanje datoteka. Pored toga, Supabase podržava edge funkcije, koje su serverless funkcije koje se izvršavaju na mreži Cloudflare Workers. Edge funkcije se mogu koristiti za implementaciju prilagođene logike ili integracija sa drugim uslugama, bez potrebe za implementacijom ili skaliranjem servera.

Treća karakteristika Supabase-a je da ima živahnu i brižnu zajednicu programera koji doprinose njenom otvorenom kodu i pružaju povratne informacije i sugestije. Supabase takođe ima korisnički prijateljski veb interfejs koji omogućava programerima da kreiraju projekte, upravljaju bazama podataka, konfigurišu autentifikaciju, prate upotrebu i pristupaju dokumentaciji. Štaviše, Supabase se vrlo lako integriše sa raznim veb okvirima i bibliotekama, kao što su React, Next.js, Vue.js, Angular, Flutter i drugi.

U poređenju sa Firebase-om, Supabase ima neke prednosti i mane. Sa jedne strane, Supabase nudi više fleksibilnosti i kontrole nad šemom baze podataka i upitima, kao i bolju kompatibilnost sa postojećim SQL alatima i standardima. Supabase takođe ima prednost otvorenosti izvornog koda i transparentnosti u vezi sa planom razvoja. Sa druge strane, Firebase ima zrelije i stabilnije usluge, kao što su Firestore, Cloud Functions, Hosting, Analytics i ML Kit. Firebase takođe ima veću i uspostavljeniju korisničku bazu i ekosistem.

A black text on a white background

Description automatically generated with medium confidence

Slika 2. Supabase – Firebase-ova alternativa otvorenog koda

## **2.3. Arhitektura i osnovni tehnološki stek**

Supabase usvaja skalabilne alate otvorenog koda, ističe korisnički prijateljska iskustva i naglašava vizuelno prijatan dizajn.

Supabase se razlikuje od Firebase okvira u pogledu pristupa i karakteristika, jer donosi specifične tehnološke izbore. Prvo, Supabase se isključivo oslanja na rešenja otvorenog koda. Takođe, daje prednost korišćenju i podršci postojećih alata kad god je to moguće, umesto da se počinje od nule.

Zapažen odstup od Firebase-a je korišćenje Postgresa umesto NoSQL skladišta. Ova svesna odluka proističe iz verovanja da nijedna druga baza podataka nema potrebne funkcionalnosti za takmičenje sa Firebase-om, istovremeno održavajući potrebnu skalabilnost za prevazilaženje konkurencije.

Sledeća slika prikazuje glavne tehničke gradivne blokove iz high level perspektive:

A picture containing text, screenshot, rectangle, font

Description automatically generated

Slika 3. High level pogled na arhitekturu

Prema ovome, glavni delovi platforme su:

* Supabase – Glavni deo. To je kompletan backend koji se bazira na Postgres bazi podataka.
* Supabase Studio – Administrativni korisnički interfejs koji vidite kada posetite app.supabase.com.
* Supabase klijentske biblioteke – Supabase pruža pristup niskog nivoa putem klijentskih biblioteka u različitim tehnologijama.
* Supabase CLI – Supabase CLI pruža interfejs komandne linije za Supabase API.
* S3 provajder za skladištenje – Supabase koristi Amazon S3 skladištenje za čuvanje svih velikih fajlova. Međutim, za varijantu koja se samostalno hostuje, morate odabrati svog S3-kompatibilnog provajdera za skladištenje.

Na sledećoj slici su prikazani tehnički gradivni blokovi Supabase-a:

A picture containing diagram, text, screenshot, rectangle

Description automatically generated

Slika 4. Arhitektura Supabase-a

Supabase obuhvata obiman set komponenti i usluga, kombinujući sledeće gradivne blokove:

* PostgreSQL (Baza podataka): PostgreSQL služi kao temelj Supabase-a, pružajući direktni pristup i puna ovlašćenja korisnicima. Za razliku od apstrakovanih baza podataka, Supabase koristi PostgreSQL i pruža praktične alate koji olakšavaju njegovu upotrebu, slično Firebase-u.
* Supabase Studio (Kontrolna tabla): Supabase Studio je open-source kontrolna tabla dizajnirana za upravljanje bazama podataka i pripadajućim uslugama.
* GoTrue (Autentifikacija): GoTrue pruža JWT-based API koji je odgovoran za upravljanje korisnicima i izdavanje tokena. On se besprekorno integriše sa PostgreSQL-ovim Row Level Security i API serverima.
* PostgREST (API): PostgREST funkcioniše kao nezavisan web server, direktno pretvarajući PostgreSQL bazu podataka u RESTful API. Ovo se postiže korišćenjem pg\_graphql ekstenzije koja omogućava pružanje GraphQL API-ja.
* Realtime (API & multiplayer): Realtime obuhvata skalabilni web socket engine koji upravlja prisustvom korisnika, emitovanjem poruka i emitovanjem promena u bazi podataka u realnom vremenu. On osigurava efikasno upravljanje interakcijama u realnom vremenu i funkcionalnostima za više igrača.
* Storage API (skladištenje velikih fajlova): Supabase koristi S3-kompatibilnu uslugu za skladištenje velikih fajlova. Metapodaci povezani sa skladištenim fajlovima se čuvaju u PostgreSQL-u.
* Deno (Edge funkcije): Deno predstavlja moderno izvršno okruženje za izvršavanje JavaScript i TypeScript koda.
* postgres-meta (Upravljanje bazom podataka): postgres-meta pruža RESTful API namenjen upravljanju PostgreSQL bazama podataka, uključujući operacije poput dobijanja tabela, dodavanja uloga i izvršavanja upita.

Pored toga, Supabase integriše sledeće komponente radi unapređenja svoje funkcionalnosti:

* PgBouncer: Lagani pooler konekcija za PostgreSQL, posebno koristan pri povezivanju sa Postgresom u Serverless funkcijama.
* Kong (API Gateway): Cloud-native API gateway zasnovan na Nginx-u.

Takođe, Supabase koristi različite provajdere skladištenja za skladištenje velikih fajlova, poput Amazon S3, sa planovima da podrži dodatne provajdere poput Wasabi i Backblaze u budućnosti.

## **2.4. Supabase-ovi proizvodni principi**

Supabase se pridržava određenog skupa načela koji usmeravaju njegov razvoj i osiguravaju bezproblemnu integraciju skalabilnih i korisnički prijateljskih funkcionalnosti:

* Sve funkcioniše izolovano: Svaka komponenta u okviru Supabase funkcioniše kao samostalan alat sa minimalnim zavisnostima. Cilj je pružiti korisnicima jednostavno korisničko iskustvo gde svaki alat može efikasno raditi samostalno. Ovo se procenjuje prema kriterijumu da li korisnici mogu koristiti proizvod isključivo sa Postgres bazom podataka.
* Sve je integrisano: Supabase ima kompozitni pristup, gde svaki proizvod unutar platforme funkcioniše nezavisno dok istovremeno sinergijski poboljšava mogućnosti drugih proizvoda. Integracija se postiže izlaganjem API-ja i Webhook-ova radi besprekorne komunikacije i interoperabilnosti između alata.
* Sve je proširivo: Supabase prioritet daje proširivosti preferirajući proširenje postojećih alata umesto uvodjenja novih. Ovaj pristup osigurava fokusiranost platforme i izbegava nepotrebnu složenost. Pružanjem osnovnih alata za razvoj, Supabase omogućava korisnicima da postignu svoje ciljeve lako i efektivno.
* Sve je prenosivo:Da bi se izbeglo zadržavanje kod određenog dobavljača i osigurala fleksibilnost, Supabase stavlja snažan naglasak na prenosivost. Korisnici mogu lako migrirati u i iz platforme, a održava se kompatibilnost između ponude u oblaku i samohostovane verzije proizvoda. Postojeći standardi, poput pg\_dump i CSV fajlova, se koriste radi poboljšanja prenosivosti podataka. U slučaju pojave novog standarda koji se takmiči sa Supabase pristupom, kompanija će preći na usvajanje novog standarda radi održavanja kontinuirane prenosivosti i konkurentnosti.
* Igranje dugoročne igre: Supabase pristupa dugoročnom pristupu, prioritet dajući održivom rastu u odnosu na kratkoročne dobitke. Umesto održavanja prilagođenih funkcionalnosti isključivo za svoje korisnike, kompanija aktivno podržava napore za uključivanje nedostajućih funkcionalnosti u glavnu verziju koda. Ova saradnja koristi celokupnoj zajednici i obezbeđuje dugovečnost Supabase-a kao pouzdanog i prilagodljivog rešenja.
* Građeno za programere: Programeri su u centru ciljne grupe Supabase-a. Platforma je dizajnirana da odgovara njihovim potrebama kao graditeljima i kreatorima. Supabase prepoznaje efikasnost i uticaj koje programeri mogu postići i ima za cilj da im pruži potrebne alate i resurse da ostvare svoj puni potencijal. Kako se pejzaž programera menja, Supabase neprestano razvija svoje ponude proizvoda da se uskladi sa novim trendovima i zahtevima.
* Podrška postojećim alatima: Supabase podstiče saradnju i aktivno podržava postojeće alate i zajednice. Funkcioniše kao "zajednica zajednica", priznajući vrednost i ekspertizu koje različiti alati i zajednice donose ekosistemu. Supabase se angažuje u saradničkim open-source naporima zapošljavanjem održavaoca, sponzorisanjem projekata, investiranjem u preduzeća i razvojem sopstvenih open-source alata. Ovaj pristup neguje dinamično i saradničko okruženje koje koristi kako korisnicima Supabase-a, tako i široj programerskoj zajednici.

# 3. Rad sa Supabase-om

## **3.1. Demo projekat**

Ono što je najpre potrebno uraditi, nakon prijavljivanja na Supabase platformu jeste kreiranje novog projekta. Koristeći hostovani Supabase *Dashboard* korisnički interfejs, u sekciji *Projects*, odabirom opcije *New Project* dobijamo modal za kreiranje i postavljanje opcija za novi projekat. U mom slučaju, demo projekat će se zvati Blog API.

A screenshot of a project

Description automatically generated with medium confidence

Slika 5. Kreiranje novog Supabase projekta

Nakon uspešno kreiranog projekta, dobijamo klaud instancu PostgreSQL baze podataka u kojoj možemo krenuti sa kreiranjem tabela. Ovo je moguće postići na dva načina, kroz vizuelni *Table editor* ili odabirom opcije *SQL editor* koja podrazumeva pisanje SQL skripti za kreiranje novih tabela. Za ovu priliku, odabraću opciju vizuelnog kreiranja tabele koristeći korisnički interfejs nakon čega će biti prikazan sledeći modal:

A screenshot of a chat

Description automatically generated with medium confidence

Slika 6. Kreiranje tabele pomoću *Table editor*-a

Na datom modalu ponuđene su nam opcije konfigurisanja kolona u *blogs* tabeli. Nove kolone u tabeli se mogu dodavati odabirom opcije *Add column* i popunjavanjem odgovarajućeg imena kolone, odabirom tipa podataka koji će data kolona predstavljati, opciono se može postaviti i podrazumevana vrednost, kao i checkbox opcija koja služi da datu kolonu postavi kao primarni ključ tabele. Takođe, „settings“ ikonica sa desne strane nudi dodatne opcije za podešavanje ponašanja kolone, a to su specifikacija da li je data kolona *nullable* (odnosno da li može prihvatati i null vrednosti), zatim specifikacija da li data kolona mora poštovati unique ograničenje (odnosno da li vrednost u svakom redu kolone mora biti jedinstvena ili ne), kao i opcija koja nam omogućava da datu kolonu definišemo kao višedimenzionalni niz promenljive dužine.

Odabirom *Save* opcije na kraju modala, tabela će biti kreirana. Zatim, kroz korisnički interfejs, odabirom opcije Insert možemo pristupiti unošenju određenih vrednosti u tabelu. Vrednosti se unose kroz modal koji izgleda ovako:

A screenshot of a computer

Description automatically generated with medium confidence

Slika 7. Unos podataka u tabelu pomoću korisničkog interfejsa

## **3.2. CRUD operacije korišćenjem C# klijentske biblioteke**

Generalno, u svrhu demonstacije rada sa Supabase bazom podataka, kreiraću mali REST API koristeći .NET 7 razvojni okvir pomoću koga ću kontaktirati udaljenu bazu podataka i izvršiti određene CRUD operacije nad njom.

Supabase C# klijentska biblioteka je open-source (kreirana je i održavana od strane zajednice), podržava integracije sa svim Supabase servisima i ima sjajnu prateću dokumentaciju.

Nakon kreiranja .NET 7 Web Api projekta, možemo dodati Supabase klijentsku biblioteku kroz *NuGet Package Manager* koji je podrazumevani Microsoft-ov menadžer paketa i dolazi integrisan uz Visual Studio okruženje. Biblioteka koju ćemo koristiti zove se ***supabase-csharp***. Prva stvar koju treba uraditi nakon dodavanja Supabase klijenta jeste konfigurisati ga. Za ovo će nam trebati jedinstveni identifikator i ključ našeg prethodno kreiranog *blogs* projekta koji se mogu naći u okviru podešavanja projekta.

Najzad, Supabase klijent ćemo u .NET projekat dodati kao *Scoped* servis sa postavljenim opcijama za automatsko osvežavanje tokena i automatskim povezivanjem u realnom vremenu na sledeći način:

builder.Services.AddScoped<Supabase.Client>(\_ =>

new Supabase.Client(

builder.Configuration["SupabaseUrl"]!,

builder.Configuration["SupabaseKey"],

new SupabaseOptions

{

AutoRefreshToken = true,

AutoConnectRealtime = true

}));

Primer 1. Dodavanje Supabase klijentske biblioteke kao servis

Ovime je konfiguracija Supabase klijenta kompletirana i može se koristiti u okviru našeg API-a. Sledeća stvar koja će nam biti potrebna jeste *Model* klasa koja će odgovarati podacima iz naše *blogs* tabele. Ona će izgledati ovako:

[Table("blogs")]

public class Blog : BaseModel

{

[PrimaryKey("id")]

public long Id { get; set; }

[Column("name")]

public string Name { get; set; }

[Column("read\_time")]

public int ReadTime { get; set; }

[Column("description")]

public string Description { get; set; }

[Column("created\_at")]

public DateTime CreatedAt { get; set; }

}

Primer 2. *Blogs* model klasa

U datoj klasi smo koristili atribute uz dekoratore iznad svakog od svojstava kako bismo precizno uparili imena svojstava iz klase sa kolonoma iz *blogs* tabele.

Dalje, radi demonstracije CRUD operacija nad bazom, možemo pristupiti slanju odgovarajućih HTTP metoda.

### 3.2.1. Dodavanje podataka

Krenimo sa mapiranjem naše *Post* metode koja će prvo kreirati *Contract* objekat zvani *CreateNewBlogRequest* a zatim, koristeći Supabase klijent, ga upisati u našu *blogs* bazu. Za ovu operaciju je bilo neophodno specificirati klijentu na koju tabelu se zahtev odnosi korišćenjem *From<>* metode, a zatim pozivom *Insert* metode koja ne samo da upisuje dati objekat u bazu već vraća *ModeledResponse* objekat iz koga možemo izvući upravo kreirani red u bazi i na taj način dobiti, recimo, *Id* novokreiranog zapisa:

app.MapPost("/blogs", async (

CreateNewBlogRequest request,

Supabase.Client client) =>

{

var blog = new Blog

{

Name = request.Name,

Description = request.Description,

ReadTime = request.ReadTime,

CreatedAt = DateTime.Now

};

var response = await client.From<Blog>().Insert(blog);

var newBlog = response.Models.First();

return Results.Ok(newBlog.Id);

});

Primer 3. *Insert* operacija

### 3.2.2. Pribavljanje podataka

Slično, podatke sa udaljenje Supabase baze možemo pribaviti u naš API korišćenjem *Get* metode Supabase klijenta. U nastavku je dat primer:

app.MapGet("/blogs/{id}", async (long id, Supabase.Client client) =>

{

var response = await client

.From<Blog>()

.Where(n => n.Id == id)

.Get();

var blog = response.Models.FirstOrDefault();

if (blog is null)

{

return Results.NotFound();

}

var blogResponse = new BlogResponse

{

Id = blog.Id,

Name = blog.Name,

Description = blog.Description,

ReadTime = blog.ReadTime,

CreatedAt = blog.CreatedAt

};

return Results.Ok(blogResponse);

});

Primer 4. *Get* operacija

U datom primeru smo na isti način Supabase klijentu precizirali sa kojom tabelom radimo korišćenjem *From<>* metode, zatim smo u nastavku iskoristili *Where* metodu za postavljanje uslova za traženim *Id*-em bloga i na kraju prosledili zahtev pozivom *Get* metode. Ova metoda takođe vraća response objekat za koji možemo vršiti proveru validnosti i iz koga uzimamo tražene podatke.

### 3.2.3. Ažuriranje podataka

Ažuriranje reda podataka u Supabase tabeli se postiže pomoću *Update* metode. U primeru ispod smo pomoću Supabase C# klijenta najpre pribavili torku sa zadatim *Id*-em čiji opis (description) želimo da promenimo, zatim smo nakon provere postojanja datog objekta pristupili lokalnom menjanju vrednosti nakon čega smo tu promenu komunicirali udaljenoj bazi pozivom *Update<>* metode.

app.MapPut("/blogs/{id}", async (long id, string description, Supabase.Client client) =>

{

var blog = await client

.From<Blog>()

.Where(n => n.Id == id)

.Single();

if (blog is null)

{

return Results.NotFound();

}

blog.Description = description;

await blog.Update<Blog>();

return Results.Ok();

});

Primer 5. *Update* operacija

### 3.2.4. Brisanje podataka

Brisanje podataka (reda) iz baze vršimo krajnje jednostavno, pomoću već spomenute *Where* metode filtriramo traženi red po *Id*-u i pozivamo *Delete* nativnu metodu Supabase klijenta*.*

app.MapDelete("/blogs/{id}", async (long id, Supabase.Client client) =>

{

await client

.From<Blog>()

.Where(n => n.Id == id)

.Delete();

return Results.NoContent();

});

Primer 6. *Delete* operacija

### 3.2.5. „Upsert“ podataka

Još jedna interesantna funkcionalnost Supabase baze podataka jeste *Upsert* operacija. Upsert predstavlja kombinaciju *Insert* i *Update* operacija. Odnosi se na operaciju koja ili ubacuje novi zapis u tabelu baze podataka ili ažurira postojeći zapis ako već postoji, na osnovu specificiranog uslova.

Jedno od ograničenja korišćenja ove operacije jeste da primarni ključevi moraju biti uključeni u sadržaj (*payload*) podataka da bi ažuriranje ispravno funkcionisalo. Ovo ujedno vuče i ograničenje da primarni ključevi moraju biti prirodni, a ne surogat.

## **3.3. Primeri korišćenja filtera u upitima**

Kao što smo videli u prethodnim primerima, podatke u Supabase bazi možemo filtrirati korišćenjem *Where* metode u kombinaciji sa operatorima poređenja vrednosti (npr. =, <, > i sl.). Međutim, za komplikovanije izraze, Supabase C# klijentska biblioteka je obezbedila *Filter* metodu pomoću koje možemo raditi sva ona filtriranja koja su podržana u relacionim bazama podataka (kao što je i PostgreSQL).

### 3.3.1. Filtriranje kolona koje se podudaraju sa šablonom

Sledeći primer pronalazi sve redove čija se vrednost u navedenoj koloni poklapa sa datim šablonom.

var result = await supabase.From<Blog>()

.Filter(x => x.Name, Operator.Like, "%la%")

.Get();

Primer 7. Filtriranje kolona po šablonu

Dati primer vrši proveru tako što uzima u obzir velika i mala slova. *Case-insensitve* varijanta bi bila sa operatorom *ILike* (Operator.ILike).

### 3.3.2. Filtriranje kolona koje se nalaze u nizu

Sledeći primer pronalazi sve redove čija se vrednost u navedenoj koloni nalazi u datom nizu vrednosti.

var result = await supabase.From<Blog>()

.Filter(x => x.Name, Operator.In, new List<object> { "Name 1", "Name 2" })

.Get();

Primer 8. Filtriranje kolona čije se vrednosti nalaze u nizu

### 3.3.3. Filtriranje kolona koje sadrže svaki element iz niza

Primer pronalazi sve redove *Blog* tabele čija *Description* kolona sadrži date vrednosti.

var result = await supabase.From<Blog>()

.Filter(x => x.Description, Operator.Contains, new List<object> { "description 1", "description 2" })

.Get();

Primer 9. Filtriranje kolona koje sadrže svaki element

## **3.4. Primeri korišćenja modifikatora**

Filteri rade na nivou reda i omogućavaju nam da vratimo redove koji odgovaraju samo određenim uslovima bez promene oblika redova.

Modifikatori, sa druge strane, su sve ono što se ne uklapa u ovu definiciju omogućavajući nam da promenimo format rezultata tako što ćemo postaviti ograničenja ili pomeraj.

### 3.4.1. Sortiranje rezultata

Prost primer korišćenja modifikatora sa Supabase klijentom jeste sledeći pomoću koga ćemo sortirati rezultujuću tabelu prema nekoj koloni u odgovarajućem redosledu.

var result = await supabase.From<Blog>()

.Select(x => new object[] { x.Name, x.Description })

.Order(x => x.Id, Ordering.Descending)

.Get();

Primer 10. Sortiranje kolona

### 3.4.2. Ograničavanje broja vraćenih redova

Predstojeći primer predstavlja *Limit* modifikator koji služi za ograničavanje broja vraćenih redova u rezultatu.

var result = await supabase.From<Blog>()

.Select(x => new object[] { x.Name, x.CountryId })

.Limit(10)

.Get();

Primer 11. Ograničavanje broja vraćenih redova

Slično datom modifikatoru, može se koristiti i *Range* modifikator koji ograničava rezultat na redove unutar navedenog inkluzivnog opsega, npr. *Range(0, 3)*.

## **3.5. Bezbednost na nivou reda (RLS)**

Nešto ranije u ovom radu, prilikom kreiranja nove tabele u bazi (na slici broj 6), mogli smo primetiti da je opcija *Enable Row Level Security (RLS)* podešena tako da bude aktivna (uključena).

Bezbednost na nivou reda (RLS) je moćna funkcionalnost podržana od strane Supabase-a koja omogućava preciznu kontrolu pristupa za tabele u bazi podataka. Pomoću RLS-a, mogu se postaviti ograničenja na određene redove na osnovu korisničkih uloga ili atributa, osiguravajući da samo ovlašćeni korisnici mogu pristupiti relevantnim podacima. Ova mogućnost pruža siguran i fleksibilan način upravljanju pristupom podacima u bazama.

Implementacija RLS-a u Supabase tabelama donosi sledeće prednosti:

* Bezbednost: RLS štiti osetljive podatke omogućavajući pristup samo ovlašćenim korisnicima, čime se poboljšava zaštita podataka.
* Fleksibilnost: Mogu se definisati prilagođene politike kontrole pristupa korišćenjem RLS-a, uzimajući u obzir korisničke uloge, atribute ili druge kriterijume. Ova precizna kontrola omogućava pristup podacima prilagođen specifičnim zahtevima.
* Skalabilnost: Politike RLS-a se primenjuju na nivou baze podataka, što rezultira efikasnom i skalabilnom kontrolom pristupa podacima kako aplikacija raste.

RLS u Supabase-u koristi inherentne mogućnosti PostgreSQL-a. Prema podrazumevanim podešavanjima, PostgreSQL odbija sve zahteve za čitanje i pisanje, ali se mogu kreirati specifične politike kako bi se omogućile željene radnje za određene tabele. Ova integracija provere autorizacije na nivou baze podataka (koja je praktično veoma slična postavljanju *WHERE* klauzule na svaki upit) poboljšava performanse i pouzdanost eliminacijom potrebe za posebnim proverama autorizacije putem API-ja prilikom dohvatanja podataka.

### 3.5.1. RLS polise

Polise su mehanizam pravila kod PostgreSQL-a koji su izuzetno moćni i fleksibilni, omogućavajući nam da pišemo kompleksna SQL pravila koja se uklapaju u naše jedinstvene poslovne potrebe. Kada jednom savladamo njihovu upotrebu, polise su jednostavne za razumevanje. Svaka polisa je vezana za tabelu i izvršava se svaki put kada se pristupa toj tabeli. Možemo zamisliti da dodajemo *WHERE* klauzulu svakom upitu. Supabase nam takođe pruža nekoliko jednostavnih funkcija koje možemo koristiti sa polisama. Na primer, sledeća polisa:

create policy "Individuals can view their own todos."

on todos for select

using ( auth.uid() = user\_id );

bi se prevodila na sledeći način kad god korisnik pokuša da izvrši *SELECT* naredbu nad todos tabelom:

select \*

from todos

where auth.uid() = todos.user\_id; -- Polisa se implicitno dodaje.

Supabase takođe pruža nekoliko jednostavnih funkcija koje možemo koristiti u okviru polisa:

* auth.uid() – Vraća ID korisnika koji pravi zahtev.
* auth.jwt() – Vraća JWT korisnika koji pravi zahtev.

### 3.5.2. Primeri RLS polisa

Evo nekoliko primera koji će nam pokazati snagu PostgreSQL-ove bezbednosti na nivou reda:

-- 1. Create table

create table profiles (

id uuid references auth.users,

avatar\_url text

);

-- 2. Enable RLS

alter table profiles

enable row level security;

-- 3. Create Policy

create policy "Public profiles are viewable by everyone."

on profiles for select using (

true

);

Primer 12. RLS – dozvola pristupa za čitanje

U datom primeru, prva sekcija upita (*Create table*) kreira tabelu *profiles* u javnoj (podrazumevanoj) šemi. Zatim, drugi deo upita (*Enable RLS*) omogućava opciju bezbednosti na nivou reda za datu tabelu, dok poslednji deo upita (*Create Policy*) kreira polisu koja omogućava pokretanje svih *SELECT* upita nad datom tabelom.

Razmotrimo sada primer polise koja nad tabelom *profiles* ograničava mogućnost ažuriranja reda samo ako on pripada datom korisniku.

-- 1. Create table

create table profiles (

id uuid references auth.users,

avatar\_url text

);

-- 2. Enable RLS

alter table profiles

enable row level security;

-- 3. Create Policy

create policy "Users can update their own profiles."

on profiles for update using (

auth.uid() = id

);

Primer 13. RLS – ograničenje ažuriranja

U datom primeru, prva i druga sekcija upita rade isto kao u prethodnom primeru (opet kreiraju tabelu *profiles* u javnoj šemi i omogućavaju opciju bezbednosti na nivou reda za datu tabelu), dok poslednji deo upita (*Create Policy*) kreira polisu koja omogućava prijavljenim korisnicima da ažuriraju sopstvene podatke.

# 4. Supabase cenovnik

Supabase nudi velikodušan besplatan plan za one koji tek počinju svoje projekte i "Pay As You Go" (plaćaj po upotrebi) za produkciona okruženja koja planiraju svoje širenje.

Supabase ima tri cenovna plana:

* Besplatan plan - za do četiri amaterska, odnosno hobi projekta;
* Pro plan – koji iznosi 25 dolara mesečno po projektu, namenjen ozbiljnijim projektima;
* Plan zasnovan na korišćenju - za one kojima treba više resursa od onoga što je dostupno na Pro planu.

Sledeća slika bolje ilustruje dostupne planove:

A picture containing text, screenshot, document, design

Description automatically generated

Slika 8. Supabase cenovni planovi

Besplatni plan je savršen za hobi projekte i jednostavne web stranice. Njime su podržani neograničen broj API poziva, OAuth provajderi za socijalne mreže, 500MB prostora za skladištenje u PostgreSQL bazi podataka, kao i 1GB prostora za neorganizovano (blob) skladištenje. Osim toga, besplatni plan može podržati i do 50 hiljada aktivnih korisnika mesečno, 500 hiljada poziva *Edge* funkcija kao i politiku jednodnevnog čuvanja logova. Međutim, treba imati na umu da ovaj plan podrazumeva to da se besplatni projekti „pauziraju“ nakon 1 nedelje neaktivnosti.

Za proizvodne aplikacije s mogućnošću skaliranja, preporučuje se Pro plan. Ovaj plan podrazumeva da nema pauziranja projekata, a osim svega navedenog u besplatnom planu nudi i: 8GB prostora u bazi podataka kao, 100GB prostora za datoteke, 50GB propusnog opsega, kao i mogućnost otpremanja 5GB datoteka. Osim toga, podržani broj aktivnih mesečnih korisnika iznosi 100 hiljada i njima je na raspolaganju dozvoljeni broj poziva *Edge* funkcija od 2 miliona. Ovaj plan podrazumeva politiku čuvanja logova 7 dana. Pro plan takođe uključuje „kvotu“ korišćenja i podrazumevano je uključeno ograničenje potrošnje. Ako želite preći preko uključenih granica, jednostavno isključite ograničenje potrošnje kako biste platili dodatno korišćenje i nesmetano se skalirali. Imajte na umu da će vaš projekat biti ograničen ako imate uključeno ograničenje potrošnje i iscrpite kvotu. Vaše postavke kontrole troškova određuju da li dozvoljavate prekoračenje. Dodatne naknade se primjenjuju za korišćenje izvan previđenog plana.

# 5. Literatura

1. *MongoDB Atlas docs. "Database as a Service (DbaaS) Explained". [*[*Online*](https://www.mongodb.com/database-as-a-service)*], [Accessed: 17-Jun-2023]*
2. *Woke, Goodness (2023). "Firebase and Supabase: Key differences you need to know". Redgate Hub, [*[*Online*](https://www.red-gate.com/simple-talk/development/other-development/firebase-and-supabase-key-differences/)*], [Accessed: 18-Jun-2023]*
3. *Aiyelabegan, Taofiq (2022). "Building a full-stack app with Next.js and Supabase". Log Rocket, [*[*Online*](https://blog.logrocket.com/build-full-stack-app-next-js-supabase/)*], [Accessed: 18-Jun-2023]*
4. *Roos, Patrick (2022). "Under the hood: Architecture and Technology Stack of Supabase". workingsoftware.dev, [*[*Online*](https://www.workingsoftware.dev/tech-stack-and-architecture-of-supabase/)*], [Accessed: 18-Jun-2023]*
5. *Supabase docs. "Architecture". [*[*Online*](https://supabase.com/docs/guides/getting-started/architecture)*], [Accessed: 19-Jun-2023]*
6. *Boudro, Dylan (2023). "Row Level Security in Supabase Tables: An Overview and Getting Started Guide for Next.js Web Applications". Starmorph Blog, [*[*Online*](https://blog.starmorph.com/blog/row-level-security-supabase-tables-nextjs)*], [Accessed: 20-Jun-2023]*
7. *Supabase docs. "Row Level Security". [*[*Online*](https://supabase.com/docs/guides/auth/row-level-security)*], [Accessed: 20-Jun-2023]*
8. *Supabase docs. “C# Client Library”. [*[*Online*](https://supabase.com/docs/reference/csharp/installing)*], [Accessed: 22-Jun-2023]*
9. *Supabase docs. “Using filters”. [*[*Online*](https://supabase.com/docs/reference/csharp/eq)*], [Accessed: 23-Jun-2023]*
10. *Supabase docs. “Using modifiers”. [*[*Online*](https://supabase.com/docs/reference/csharp/order)*], [Accessed: 23-Jun-2023]*

# 6. Listing

Slika 1. Šema DBaaS arhitekture u oblaku 4

Slika 2. Supabase – Firebase-ova alternativa otvorenog koda 6

Slika 3. High level pogled na arhitekturu 7

Slika 4. Arhitektura Supabase-a 8

Slika 5. Kreiranje novog Supabase projekta 11

Slika 6. Kreiranje tabele pomoću *Table editor*-a 12

Slika 7. Unos podataka u tabelu pomoću korisničkog interfejsa 13

Slika 8. Supabase cenovni planovi 23

Primer 1. Dodavanje Supabase klijentske biblioteke kao servis 14

Primer 2. *Blogs* model klasa 15

Primer 3. *Insert* operacija 15

Primer 4. *Get* operacija 16

Primer 5. *Update* operacija 17

Primer 6. *Delete* operacija 17

Primer 7. Filtriranje kolona po šablonu 18

Primer 8. Filtriranje kolona čije se vrednosti nalaze u nizu 18

Primer 9. Filtriranje kolona koje sadrže svaki element 19

Primer 10. Sortiranje kolona 19

Primer 11. Ograničavanje broja vraćenih redova 19

Primer 12. RLS – dozvola pristupa za čitanje 21

Primer 13. RLS – ograničenje ažuriranja 22