

**SVEUČILIŠTE U SPLITU**  
**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Stručni diplomski studij Primijenjeno računarstvo

**ANAMARIJA PAPIĆ**

**D I P L O M S K I   R A D**

**RAZVOJ WEB APLIKACIJE ZA REPETICIJE**

Split, veljača 2026.

**SVEUČILIŠTE U SPLITU**  
**SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE**

Stručni diplomski studij Primijenjeno računarstvo

**Predmet:** Kriptovalute

**D I P L O M S K I   R A D**

**Kandidat:** Anamarija Papić

**Naslov rada:** Razvoj web aplikacije za repeticije

**Mentor:** Nikola Grgić, viši predavač

Split, veljača 2026.

# Sadržaj

|   |           |
|---|-----------|
| <b>Sažetak</b>                                      | <b>1</b>  |
| <b>1 Uvod</b>                                       | <b>2</b>  |
| <b>2 Tehnologije i teoretska podloga</b>            | <b>4</b>  |
| 2.1    Prezentacijski i aplikacijski sloj . . . . . | 4         |
| 2.2    Sloj korisničkog sučelja . . . . .           | 6         |
| 2.3    Podatkovni sloj . . . . .                    | 8         |
| 2.4    Komunikacijski sloj . . . . .                | 9         |
| 2.4.1    WebRTC . . . . .                           | 9         |
| 2.4.1.1    Topologije sustava WebRTC . . . . .      | 11        |
| 2.4.1.2    Faze uspostave WebRTC veze . . . . .     | 12        |
| 2.5    Ostali alati . . . . .                       | 16        |
| <b>3 Prikaz praktičnog dijela</b>                   | <b>17</b> |
| 3.1    Potpoglavlje . . . . .                       | 17        |
| 3.1.1    ABC . . . . .                              | 17        |
| 3.1.1.1    123 . . . . .                            | 17        |
| 3.1.1.2    456 . . . . .                            | 17        |
| 3.1.2    DEF . . . . .                              | 17        |
| 3.2    Još jedno potpoglavlje . . . . .             | 17        |
| <b>4 Zaključak</b>                                  | <b>18</b> |
| <b>Literatura</b>                                   | <b>19</b> |
| <b>Dodatci</b>                                      | <b>21</b> |

# Sažetak

Cilj ovog diplomskog rada je razviti intuitivnu web aplikaciju koja će služiti kao platforma za povezivanje studenata i tutora s ciljem olakšavanja organizacije i održavanja repeticija, pojednostavljujući proces pružanja i pohađanja repeticija uživo ili online. Aplikacija omogućuje studentima jednostavno pronalaženje i rezervaciju repeticija, dok tutori mogu učinkovito upravljati vlastitim rasporedom, ponudama i rezervacijama. Sudjelovanje u online repeticijama podržano je integracijom *peer-to-peer* tehnologije WebRTC za video pozive. *Full-stack* web aplikacija razvijena je u razvojnom okviru za React – Next.js-u, dok se za pohranu podataka koristi relacijska baza podataka PostgreSQL upravljana putem Prisma ORM-a i smještena na platformi Supabase. U ovom pisanom radu predstavljene su korištene tehnologije, arhitektura sustava, poslovna logika te implementacija pojedinih funkcionalnosti aplikacije.

**Ključne riječi:** *sustav za upravljanje rezervacijama, Next.js, WebRTC, audio-video pozivi*

## Summary

### Development of a Web Application for Tutoring

The aim of this thesis is to develop an intuitive web application that serves as a platform for connecting students and tutors, with the goal of facilitating the organization and conduct of tutoring sessions and simplifying the process of providing and attending lessons, both in-person and online. The application enables students to easily find and book tutoring sessions, while tutors can efficiently manage their schedules, offers, and reservations. Participation in online tutoring sessions is supported through integrated WebRTC peer-to-peer technology for video calls. The full-stack web application is developed using the Next.js framework for React, while data storage is handled by a relational PostgreSQL database through Prisma ORM and managed on the Supabase platform. This thesis presents the technologies used, the system architecture, business logic, and the implementation of individual application functionalities.

**Keywords:** *booking management system, Next.js, WebRTC, audio-video calls*

# 1. Uvod

U hrvatskom se obrazovanju posljednjih dvadesetak godina jasno potvrđuje snažan rast „sjene obrazovanja” – razgranat obrazovni biznis plaćenih privatnih instrukcija i organiziranih priprema za državnu maturu i prijemne ispite, koji funkcionira paralelno s formalnim obrazovnim sustavom. Dodatne poduke postale su uobičajen dio školovanja i svakodnevnica mnogih hrvatskih đaka svih uzrasta – od osnovne škole do fakulteta, a ne iznimka, unatoč tome što ovakav oblik obrazovanja otvara niz pitanja jednakosti pristupa znanju (navedene usluge dostupnije su učenicima iz socioekonomski povoljnijih obitelji te iz većih geografskih centara) te stvara pritisak na učenike i komercijalizira obrazovanje. Istraživanja Borisa Jokića i Zrinke Ristić Dedić pokazuju da je gotovo 40,0% učenika u svakoj ispitanoj generaciji (8. razred osnovne te 2. i završni razred srednje škole) koristilo privatne instrukcije u školskoj godini 2020./2021., a među maturantima ih je 38,0%. Svaki drugi maturant gimnazije pohađa instrukcije (i to 17,1% njih redovito), dok 39,2% učenika završnih razreda strukovnih i umjetničkih škola također koristi takvu pomoć, najčešće iz matematike i ključnih predmeta za upis na studij. Većina učenika i dalje preferira pohađanje repeticija uživo, međutim 20,6% maturanata i 15,9% osmaša pohađalo je privatne instrukcije u šk. god. 2020./2021. u online obliku [1]. 56,0% maturanata pohađalo je pripremne tečajeve za državnu maturu, a njih 36,1% pohađalo ih je u nekom obliku kod privatnih tvrtki. 62,9% gimnazijalaca i 50,3% učenika strukovnih škola koji planiraju upisati studij pohađalo je pripremne tečajeve u šk. god. 2020./2021., pri čemu 48,6% gimnazijalaca i 26,4% *strukovnjaka* kod privatnih tvrtki [2].

Trenutno se ponuda privatnih repeticija u Hrvatskoj najčešće pronalazi putem usmenih preporuka, online oglasnika ili grupa na društvenim mrežama, kao i na zalijepljenim oglasima na javnim površinama, gdje svoje usluge podučavanja najčešće nude uspješniji studenti i (umirovljeni) nastavnici kako bi zaradili dodatni prihod. U domaćem se medijskom prostoru povremeno izvještava da su tutori stoga i česta meta Državnog inspektorata zbog rada „na crno” jer mnogi od njih nisu registrirani kao obrtnici, a s obzirom na popularnost ovakvog oblika dodatne zarade, nadzori su učestali. Razvijena aplikacija zasad ipak ne zalazi u ovu pravnu i poreznu problematiku, već samo ističe cijene usluga te plaćanje usluga ostaje između tutora i učenika izvan same aplikacije.

Praktični dio ovog rada, aplikacija *repeticije-hr*, nastaje upravo unutar prethodno pred-

stavljenog konteksta, kao pokušaj da se postojeće, često netransparentno i nejednako tržište privatnih instrukcija učini pravednijim, preglednijim i dostupnijim većem broju učenika i studenata. Temeljna ideja aplikacije je digitalno povezati učenike i tutore na jednom mjestu, uz jasan uvid u kvalifikacije, ponudu predmeta, cijene, termine, lokaciju (uživo/online) i povratne informacije drugih korisnika, čime se smanjuje ovisnost o „vezi“ i uskim krugovima preporuka. Time se barem djelomično adresiraju problemi koje ističu Jokić i Ristić Dedić – neravnomjerna dostupnost instrukcija, netransparentnost ponude i snažna socijalna selektivnost – jer učenici iz različitih dijelova Hrvatske, neovisno o tome žive li u većim centrima ili manjim sredinama, dobivaju strukturiran digitalni prostor u kojem mogu lakše pronaći podršku koja im je potrebna.

U poglavljima koja slijede prvo će kratko biti predstavljene korištene tehnologije pri izradi aplikacije i objašnjena teoretska podloga, a zatim će se detaljno i pregledno opisati poslovna logika i način na koji je sama aplikacija implementirana.

## 2. Tehnologije i teoretska podloga

U ovom poglavlju detaljno su opisane tehnologije i alati korišteni u razvoju, kao i teoretska podloga koja je oblikovala arhitekturu i dizajn sustava. Polazi se od klasične troslojne arhitekture (prezentacijski, aplikacijski i podatkovni sloj), ali su za potrebe jasnijeg opisa korištenih tehnologija ti slojevi dodatno razrađeni u zajednički prezentacijski i aplikacijski sloj, sloj korisničkog sučelja, podatkovni sloj te komunikacijski sloj. Na kraju poglavlja kratko su opisani i ostali alati koji su korišteni u procesu razvoja.

### 2.1. Prezentacijski i aplikacijski sloj

U klasičnoj troslojnoj arhitekturi prezentacijski sloj zadužen je za prikaz podataka i interakciju s korisnikom, dok aplikacijski sloj implementira poslovnu logiku i orkestrira pristup podatkovnom sloju. U ovoj aplikaciji Next.js, temeljen na Reactu, istovremeno preuzima ulogu prezentacijskog sloja (renderiranje korisničkog sučelja u pregledniku) i dijela aplikacijskog sloja (renderiranje na strani poslužitelja, API rute, obrada zahtjeva), dok TypeScript, Node.js i npm čine temeljni tehnološki ekosustav u kojem se taj sloj razvija i izvršava.

**Next.js** je razvojni okvir (engl. *framework*) otvorenog kôda za React koji omogućava objedinjeni (engl. *full-stack*) razvoj web aplikacija, što znači da obuhvaća razvoj i klijentskog (engl. *front-end*) i poslužiteljskog (engl. *back-end*) dijela aplikacije. Razvijen je od strane tvrtke Vercel, a izvorni kôd dostupan je na platformi GitHub [3] gdje je prva verzija objavljena 2016. godine.

Temeljen je na JavaScriptu i izvršava se u okruženju **Node.js**, pri čemu Node.js omogućava pokretanje JavaScript kôda izvan preglednika, na poslužitelju, što je preduvjet za poslužiteljsko renderiranje i izgradnju aplikacije [4]. U praksi se Next.js projekti gotovo uvijek razvijaju uz **TypeScript**, nadskup JavaScripta razvijen od strane Microsofta, koji uvodi statičku tipizaciju i provjeru kôda u vrijeme prevođenja, čime se smanjuje broj pogrešaka u složenijim aplikacijama [5].

Upravljanje ovisnostima obavlja se putem **npm**-a (engl. *Node Package Manager*) ili alternativno **pnpm**/**yarn**/**bun** upravitelja paketa, koji omogućuju jednostavnu instalaciju, ažuriranje i skriptiranje razvojnih zadataka u okruženju Node.js. Paketi (biblioteke) se preuzimaju iz javnog repozitorija **npmjs.com**, a definiraju se u datoteci `package.json` unutar

projekta, a pri instalaciji se kreira i datoteka `package-lock.json` koja zaključava točne verzije ovisnosti radi konzistentnog okruženja. U direktoriju `node_modules` pohranjuju se sve instalirane ovisnosti projekta [6].

**React** (React.js) je slobodan softver otvorenog kôda tzv. FOSS (engl. *Free and Open Source Software*). Riječ je o JavaScript biblioteci za izgradnju korisničkih sučelja, razvijenoj od strane Facebooka (sada Meta). Programski kôd javno je objavljen 2013. godine, a od tada je prigrljen od strane zajednice i pozicionirao se kao jedna od najpopularnijih biblioteka za razvoj web aplikacija [7].

React se često opisuje kroz tri osnovne značajke: *declarative*, *component-based* i sloganom „*learn once, write anywhere*”. Deklarativan pristup znači da razvojni programer opisuje kakvo korisničko sučelje želi u određenom stanju aplikacije, a React se brine za učinkovit prikaz i osvježavanje DOM-a (engl. *Document Object Model*), što olakšava razumijevanje i održavanje kôda. Komponentni pristup podrazumijeva da se sučelje sastoji od malih, ponovno iskoristivih komponenti koje enkapsuliraju logiku i prikaz pa se iste komponente mogu koristiti na više mjesta i u različitim projektima. Načelo „*learn once, write anywhere*” odnosi se na činjenicu da se ista znanja i koncepti mogu primijeniti u web aplikacijama, ali i u izradi mobilnih (React Native) i desktop aplikacija, bez ponovnog učenja potpuno novog programskog modela [8].

React se temelji na konceptu komponenti, pri čemu se u povijesnom razvoju razlikuju klasne (engl. *class*) komponente i funkcijske (engl. *function*) komponente, koje su trenutno preporučeni način pisanja komponenti. Uvođenjem kuka (engl. *hooks*) preko Hook API-ja (engl. *Application Programming Interface*) funkcijske komponente dobile su mogućnost upravljanja stanjem i *side* efektima bez potrebe za klasama, čime se dodatno pojednostavilo strukturiranje logike i ponovna iskoristivost kôda. React koristi virtualni DOM – laganu, u memoriji pohranjenu reprezentaciju strukture korisničkog sučelja kako bi učinkovito uspoređivao prethodno i novo stanje te primjenjivao samo nužne promjene u stvarnom DOM-u, što značajno poboljšava performanse složenijih aplikacija. JSX (engl. *JavaScript XML*) je sintaksno proširenje za JavaScript koji omogućuje zapis komponenti u obliku HTML-u slične sintakse, pri čemu se JSX u pozadini prevodi u pozive funkcije `React.createElement`, a razvojnem programeru olakšava čitanje i strukturiranje kôda [9]. U kombinaciji s TypeScriptom koristi se proširenje TSX (engl. *TypeScript JSX*). U novijim verzijama React uvodi

i *Server Components* – komponente koje se izvršavaju isključivo na poslužitelju, generiraju HTML bez slanja JavaScript kôda na klijent i tako smanjuju veličinu paketa te povećavaju sigurnost i performanse, osobito u kombinaciji s razvojnim okvirima poput Next.js-a.

Next.js nadograđuje React dodajući brojne značajke i optimizacije koje olakšavaju izgradnju skalabilnih i visokoučinkovitih web aplikacija. Uvodi višestruke načine renderiranja stranica i napredne optimizacije koje su posebno važne za performanse i SEO. Osim klasičnog renderiranja na strani klijenta – CSR (engl. *Client-Side Rendering*), Next.js podržava renderiranje na strani poslužitelja – SSR (engl. *Server-Side Rendering*), statičko generiranje stranica – SSG (engl. *Static Site Generation*) te inkrementalno statičko obnavljanje – ISR (engl. *Incremental Static Regeneration*), pri čemu se većina sadržaja isporučuje kao statički HTML, a pojedine stranice se povremeno regeneriraju kada se podaci promijene. Next.js također automatski dijeli kôd u manje dijelove (engl. *code splitting*) i koristi usmjeravanje (engl. *routing*) bazirano na strukturi direktorija, poznato kao *file-based routing*. To znači da se samo nužni dijelovi kôda učitavaju za svaku stranicu, čime se smanjuje vrijeme učitavanja i poboljšava korisničko iskustvo. U novijim verzijama uveden je App Router s podrškom za React Server Components, strujanje (engl. *streaming*) sadržaja i Server Actions, što omogućuje da se dio logike obrade podataka izvršava na poslužitelju uz manju količinu potrebnog JavaScript kôda na klijentu, što pruža brži prikaz sadržaja korisniku [10].

Osim spomenutih značajki, Next.js sadrži integrirane alate za optimizaciju slika automatski generirajući različite veličine, koristi lijeno učitavanje (engl. *lazy loading*) i moderne formate slika, čime se poboljšavaju Core Web Vitals metrike i ukupne performanse web aplikacija. U kombinaciji s podrškom za TypeScript „*out-of-the-box*”, integracijom s Vercelom za jednostavnu isporuku te bogatim ekosustavom dodataka, Next.js se pozicionira kao cjelovit razvojni okvir za izradu skalabilnih, produkcijski spremnih aplikacija [10].

U trenutku izrade rada najnovija stabilna verzija Next.js-a je 16, dok je React dosegaio verziju 19 te su iste korištene u izradi praktičnog rada, uz TypeScript verzije 5.x. Korištena verzija Node.js-a je 24.x LTS (engl. *Long Term Support*) uz npm verzije 11.x.

## 2.2. Sloj korisničkog sučelja

Sloj korisničkog sučelja (engl. UI – *User Interface*) aplikacije temelji se na kombinaciji biblioteka *shadcn/ui*, *Radix UI* i *Tailwind CSS*, koje su usko povezane i nadograđuju se

jedna na drugu.

**shadcn/ui** je zbirka unaprijed pripremljenih React komponenti (npr. avatar, botuni, bedževi, kartice za prikaz sadržaja, dijaloški okviri, padajući izbornici, polja i grupe za unos, navigacijski izbornici, tablice itd.) koje su izgrađene na Radix UI „primitivima” i stilizirane pomoću Tailwind CSS-a. Umjesto da se isporučuje kao klasična biblioteka ovisnosti, **shadcn/ui** generira stvarne izvorišne datoteke komponenti koje se kopiraju u projekt, čime razvojni programer zadržava potpunu kontrolu nad kôdom, može ga prilagoditi specifičnim potrebama aplikacije i izbjegava dodatni teret zasebne UI biblioteke dok se aplikacija izvršava (engl. *runtime*). Ovo je osobito korisno u većim projektima, gdje se očekuju dugoročno održavanje, prilagodba dizajna i dosljedan vizualni identitet [11].

**Radix UI** predstavlja skup pristupačnih (engl. *accessible*) niskorazinskih komponenti tzv. „primitiva” poput dijaloških okvira, skočnih prozora, kartica (engl. *tabs*), skočnih izbornika i slično. Fokus Radix UI-ja je na ispravnom ponašanju i pristupačnosti: osigurava ispravne ARIA (engl. *Accessible Rich Internet Applications*) attribute, upravljanje fokusom i podršku za tipkovničku navigaciju u skladu s WAI-ARIA (engl. *Web Accessibility Initiative – ARIA*) smjernicama, dok izgled i stilizacija ostaju u potpunosti u domeni razvojnih timova [12].

**Tailwind CSS** je utilitaristički (engl. *utility-first*) razvojni okvir za CSS koji umjesto gotovih vizualnih komponenti nudi velik broj malih, jednoznačnih klasa za oblikovanje (npr. razmaci, boje, tipografija, raspored – fleksibilni i mrežni (engl. *grid*) rasporedi). Te se klase kombiniraju izravno u markup-u ili JSX/TSX kôdu, čime se smanjuje potreba za pisanjem zasebnih CSS datoteka i olakšava održavanje dosljednog dizajna u komponentnom pristupu Reacta. Tailwind je visokokonfigurabilan putem konfiguracijske datoteke koja definira dizajnerske tokene (boje, tipografiju, radijuse, razmake), a u produkcijskom okruženju alat automatski uklanja neiskorištene klase (engl. *tree shaking*), što rezultira malim konačnim CSS paketom i boljim performansama [13].

U razvijenoj aplikaciji kombinacija predstavljenih UI biblioteka omogućava izradu modernog, responzivnog i pristupačnog korisničkog sučelja uz dobru ravnotežu između brzine razvoja, kontrole nad dizajnom i tehničke kvalitete.

## 2.3. Podatkovni sloj

Za pohranu podataka u aplikaciji koristi se relacijska baza podataka PostgreSQL, uz objektno-relacijsko mapiranje pomoću Prisma ORM-a i upravljani *hosting* preko platforme Supabase.

**PostgreSQL** (Postgres) snažan je sustav za upravljanje relacijskim bazama podataka (engl. *RDBMS – Relational Database Management System*) otvorenog kôda s dugom poviješću razvoja, visokom razinom usklađenosti sa SQL (engl. *Structured Query Language*) standardom te podrškom za napredne tipove podataka i transakcijske operacije. Kao objektno-relacijski sustav, PostgreSQL osim klasičnih relacijskih tablica podržava i pohranu polustrukturiranih podataka (npr. JSON/JSONB tipovi, polja, geografski tipovi), što ga čini pogodnim za moderne web aplikacije koje kombiniraju strogo strukturirane podatke (npr. korisnici, termini rezervacija, recenzije) s fleksibilnijim strukturama. [14].

**Prisma ORM** (engl. *Object-Relational Mapper*) je biblioteka za objektno-relacijsko mapiranje za Node.js i TypeScript koji omogućuje tipno siguran (engl. *type-safe*) pristup bazi podataka. U središtu Prisma pristupa nalazi se datoteka `schema.prisma` u kojoj se deklarativno definira shema podataka putem Prisma Schema Language (PSL) – definiraju se modeli, relacije i mapiranja na tablice u bazi, a Prisma na temelju te sheme automatski generira klijenta za pristup podacima, zajedno s odgovarajućim TypeScript tipovima. Time se postiže da su sve upite prema bazi moguće statički provjeriti u vrijeme prevođenja, uz bogatu podršku za automatsko dovršavanje i otkrivanje neispravnih upita prije pokretanja aplikacije, što smanjuje broj pogrešaka i ubrzava razvoj. Uz Prisma Client za rad s podacima, Prisma nudi i alate za migracije (Prisma Migrate) te vizualni pregled podataka (Prisma Studio) koji olakšavaju upravljanje razvojnim i produkcijskim okruženjima. [15].

**Supabase** je platforma otvorenog kôda tipa „*backend kao usluga*” (engl. *Backend-as-a-Service*) izgrađena na PostgreSQL-u, koja nudi upravljanu bazu podataka, autentikaciju, pohranu datoteka, *real-time* funkcionalnosti i poslužiteljske funkcije. U praksi to znači da se PostgreSQL baza podataka može vrlo brzo podići u oblaku, uz automatski izložene API-je, ugrađenu podršku za sigurnost na razini retka (engl. *Row-Level Security*) i mogućnost „osluškivanja” promjena u tablicama u stvarnom vremenu putem WebSoketa. Supabase pruža i CLI alat te mogućnost lokalnog razvoja putem Docker okruženja, što olakšava repliciranje produkcijske baze u lokalnom okruženju i testiranje sheme podataka. [16].

U ovoj aplikaciji PostgreSQL služi kao temeljna baza podataka, Prisma ORM kao tipno siguran sloj pristupa podacima u okruženju Next.js/TypeScript, dok Supabase pruža *hosting* baze i *real-time* funkcionalnosti, čime se dobiva skalabilan i moderni podatkovni sloj prilagođen potrebama web aplikacije za repetitive. Korištena su sljedeća izdanja tehnologija: PostgreSQL 17.x, Prisma ORM 7.x i Supabase 2.x.

## 2.4. Komunikacijski sloj

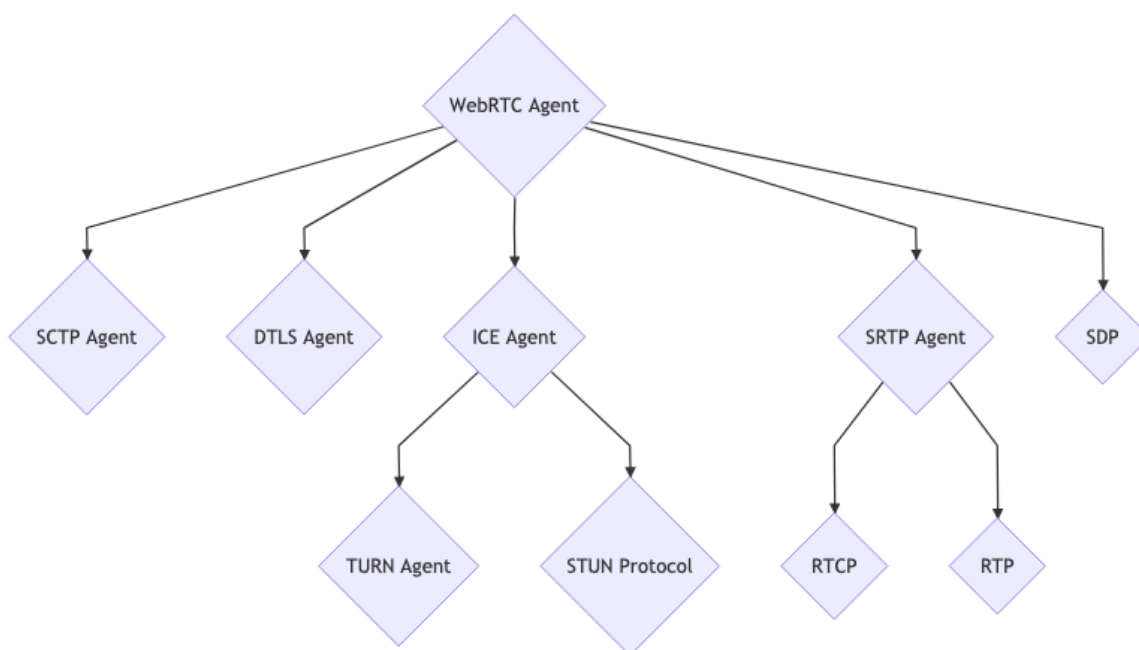
Komunikacijski sloj u ovoj aplikaciji zadužen je za uspostavljanje i održavanje komunikacije u stvarnom vremenu između tutora i učenika tijekom online repetitiva. U tu svrhu koristi se **WebRTC** (engl. *Web Real-Time Communication*).

WebRTC preuzima brigu o dohvatanju audio i video tokova, uspostavi šifriranog kanala te prilagodbi kvalitete prijenosa uvjetima mreže, dok se logika aplikacije fokusira na upravljanje sobama, sudionicima i korisničkim sučeljem. U nastavku su detaljnije opisane ključne značajke i funkcionalnosti tehnologije WebRTC.

### 2.4.1. WebRTC

Tehnologija WebRTC je skup standardiziranih tehnologija otvorenog kôda (FOSS) koja omogućuje izravnu *peer-to-peer* komunikaciju u stvarnom vremenu putem interneta, i to kroz moderne web preglednike ili native klijente u slučaju mobilnih aplikacija. Krajnji korisnici mogu izmjenjivati audio, video i podatke bez potrebe za dodatcima ili posebnim softverom. WebRTC se sastoji od WebRTC protokola i WebRTC API-ja specificiranog za JavaScript, dok je za native klijente dostupna biblioteka koja pruža iste funkcionalnosti. [17] Izvorni kôd dostupan je u repozitoriju na poveznici `webrtc.googlesource.com/src`.

WebRTC protokol u suštini je sastavljen od niza standardiziranih protokola i stoga je često opisivan kao kolekcija protokola jer orkestrira različite protokole i mehanizme za uspostavu, održavanje i osiguravanje komunikacije u stvarnom vremenu, što je prikazano na slici 1 [18].



**Slika 1:** WebRTC kao kolekcija protokola (Izvor: [webrtcforthe curious .com](http://webrtcforthe curious.com))

Nastao je kao pokušaj da se tehnologije za govor i video u stvarnom vremenu, koje su ranije bile zatvorene i vlasničke, ugrade izravno u web preglednike kao otvoren standard. Google je 2010. kupio tvrtku Global IP Solutions (GIPS) te otvoreno licencirao kôd njihovih RTC komponenti, što je bio ključni korak u razvoju WebRTC-a. Nakon toga, Google je 2011. pokrenuo projekt WebRTC, a standardizacija je trajala nekoliko godina i iziskivala velike napore i suradnju između dvaju standardizacijskih tijela: W3C-a (engl. *World Wide Web Consortium*) i IETF-a (engl. *Internet Engineering Task Force*) te velikog broja tvrtki i pojedinaca iz industrije. Uz Google, u razvoj specifikacija i implementaciju uključile su se i tvrtke poput Mozille, Microsofta, Applea, Intel Corporations, Cisca i Ericssona, a podrška je postupno uvedena u svim modernim web preglednicima, čime je WebRTC postao standardni mehanizam za *in-browser* audio-video komunikaciju [19]. Danas se primjenjuje u širokom spektru aplikacija, od video konferencija, do online igara, udaljenog upravljanja, IoT (engl. *Internet of Things*) komunikacije i drugih scenarija koji zahtijevaju nisku latenciju i visoku kvalitetu prijenosa.

### 2.4.1.1. Topologije sustava WebRTC

WebRTC topologije mogu se podijeliti na *peer-to-peer* (P2P) i *client/server* pristup [18].

U najjednostavnijem slučaju dvije krajnje točke uspostavljaju izravnu vezu „1 na 1” te međusobno razmjenjuju medijske tokove i podatke. Za grupne scenarije moguće je koristiti više različitih topologija, ovisno o broju sudionika, dostupnoj propusnosti i zahtjevima aplikacije.

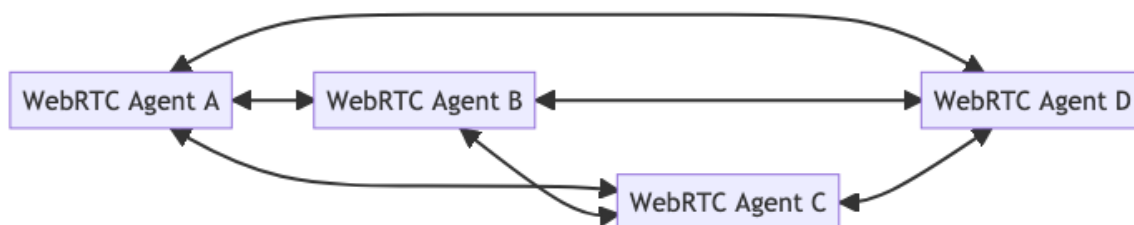
Topologija *full mesh* podrazumijeva da je svaki sudionik povezan izravno sa svakim drugim sudionikom. Svaki klijent tada mora zasebno kodirati i slati svoj video tok prema svakom udaljenom sudioniku, što povećava opterećenje mreže i ograničava praktičan broj sudionika, ali uz relativno jednostavnu implementaciju i bez dodatne poslužiteljske infrastrukture za medije. Alternativno, *hybrid mesh* smanjuje broj izravnih veza tako da se mediji proslijeđuju preko pojedinih sudionika.

Topologije *client/server* tipa koriste specijalizirane poslužitelje: SFU (engl. *Selective Forwarding Unit*) koji selektivno proslijeđuje pristigle tokove svim klijentima te MCU (engl. *Multipoint Conferencing Unit*) koji višestruke ulazne tokove spaja u jedan kompozitni izlazni tok.

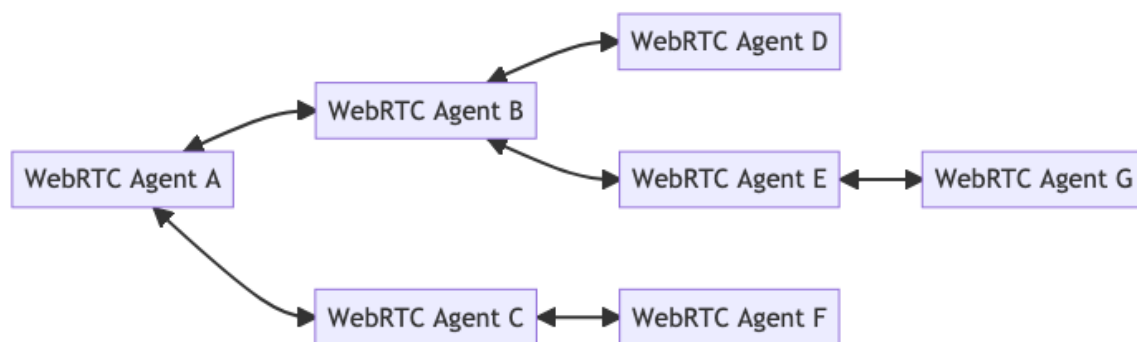
Na slikama 2 – 6 prikazane su različite WebRTC topologije.



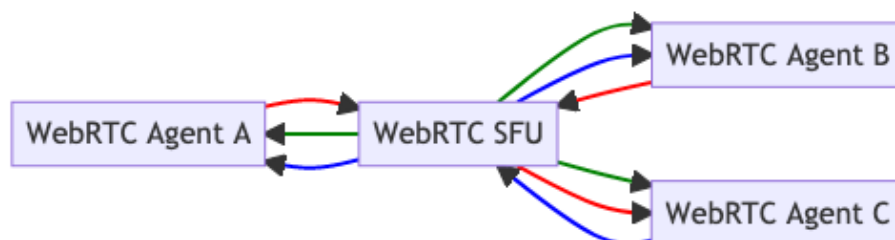
**Slika 2:** Topologija „1 na 1” (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))



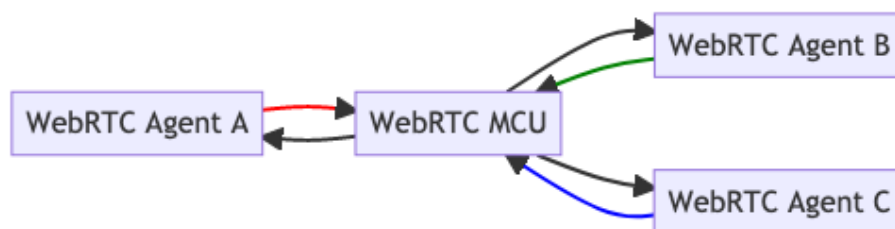
**Slika 3:** Topologija *full mesh* (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))



**Slika 4:** Topologija *hybrid mesh* (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))



**Slika 5:** Topologija SFU (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))



**Slika 6:** Topologija MCU (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))

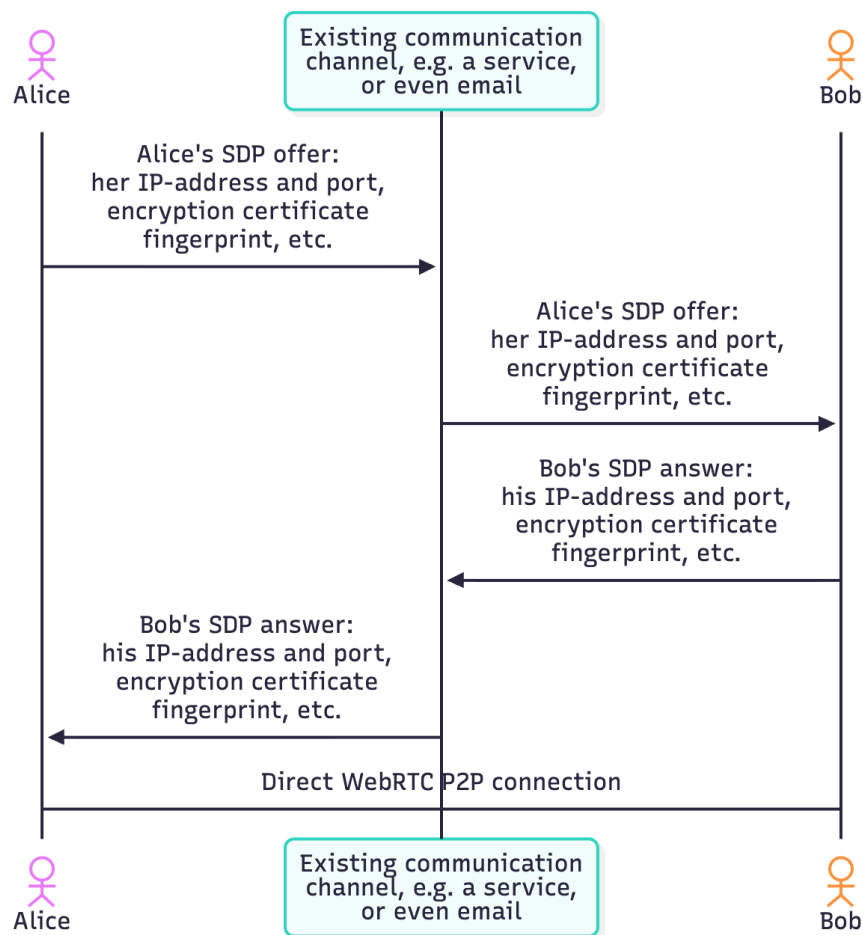
U ovoj aplikaciji koristi se *peer-to-peer* topologija *full mesh* što znači da svaki klijent u sobi uspostavlja po jednu vezu `RTCPeerConnection` prema svakom drugom sudioniku.

#### 2.4.1.2. Faze uspostave WebRTC veze

Uspostava WebRTC veze može se konceptualno rastaviti na četiri uzastopna koraka: signalizaciju, uspostavu povezivanja, osiguravanje veze i samu komunikaciju, pri čemu svaki sljedeći korak započinje tek kada je prethodni u potpunosti uspješno dovršen.

Prva faza, **signalizacija**, obuhvaća sve korake kojima se dva (ili više) sudionika međusobno „pronađu” i dogovore kako će komunicirati, prije nego što krene stvarni prijenos audio-video podataka. Signalizacija nije standardizirana u samom WebRTC-u, već se odvija

„*out-of-band*” i smatra dijelom aplikacijske logike – razvojni tim slobodno bira protokole i formate poruka (npr. REST, WebSocket i sl.). Najprije sudionici uspostave vezu sa zajedničkim signalizacijskim kanalom i na neki način identificiraju sobu ili razgovor u kojem sudjeluju. Zatim preko tog kanala razmjenjuju SDP (engl. *Session Description Protocol*) poruke s informacijama o tome koje medijske tokove žele slati te koje *kodeke* i konfiguracije podržavaju, kao i pripadajuće signalizacijske poruke (ponuda/odgovor, kandidati i sl.) potrebne za uspostavu veze [18]. Signalizacija je ilustrirana na slici 7.



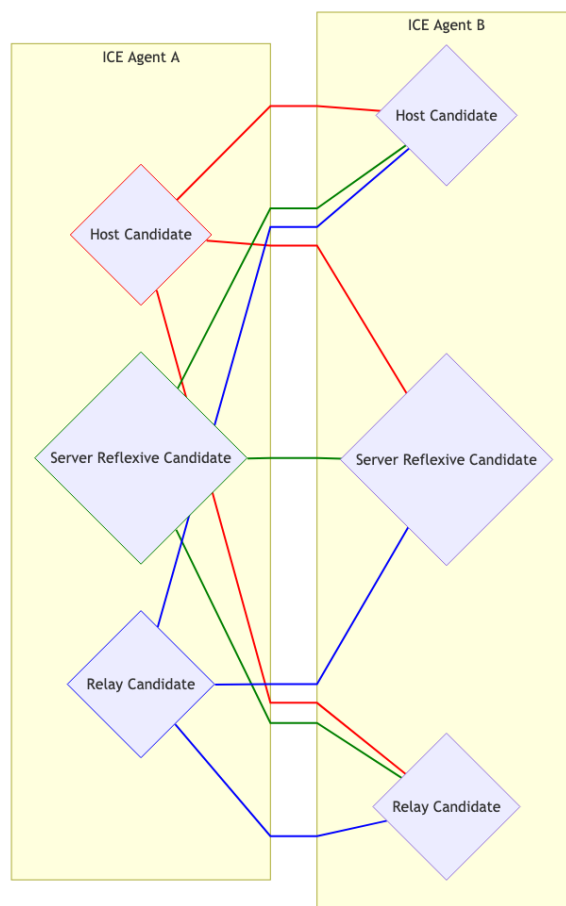
**Slika 7:** Signalizacija (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))

Budući da WebRTC ne specificira način signalizacije, ova aplikacija koristi već spomenute *real-time* funkcionalnosti platforme Supabase kao signalizacijski kanal. To znači da se poruke potrebne za uspostavu veze razmjenjuju preko Supabase *real-time* kanala, dok se sami medijski tokovi (audio i video) nakon dogovora o parametrima uspostavljaju izravno između preglednika sudionika. Na taj se način podatkovni sloj (Supabase/PostgreSQL) i

komunikacijski sloj (WebRTC) integriraju u cjelinu – Supabase osigurava pouzdanu i skalabilnu razmjenu signalnih poruka, a WebRTC preuzima prijenos multimedije u stvarnom vremenu.

Nakon što je signalizacijom razmijenjena osnovna konfiguracija sesije, slijedi druga faza – **povezivanje**, u kojoj WebRTC pokušava uspostaviti izvedivu mrežnu putanju između sudionika. U stvarnim uvjetima većina klijenata ne posjeduje javnu IP adresu, već se nalazi iza usmjernika (engl. *router*), NAT (engl. *Network Address Translation*) uređaja i vatrozida, što onemogućuje izravno adresiranje između dvaju računala. Kako bi unatoč tome ostvario *peer-to-peer* komunikaciju, WebRTC koristi kombinaciju okvira ICE (engl. *Interactive Connectivity Establishment*) te STUN (engl. *Session Traversal Utilities for NAT*) i TURN (engl. *Traversal Using Relays around NAT*) infrastrukture za tzv. *NAT traversal*, odnosno pronalaženje puteva kroz mrežnu konfiguraciju [18].

ICE na svakoj strani prikuplja različite mrežne „kandidate” — tzv. ICE kandidate (engl. *ICE candidates*) — što uključuje lokalne adrese, javne adrese dobivene putem STUN poslužitelja te, po potrebi, relejske (engl. *relay*) adrese dobivene preko TURN poslužitelja. Ti se kandidati zatim razmjenjuju signalizacijskim kanalom i sustavno testiraju kako bi se pronašao par kandidata preko kojih je moguće uspostaviti dvosmjernu komunikaciju uz što manju latenciju. Testiranje parova kandidata (engl. *connectivity checks*) prikazano je na slici 8. Na slici se nalaze dva ICE agenta, svaki s po tri različita kandidata (što podrazumijeva ukupno devet parova kandidata). U idealnom slučaju odabire se izravna P2P putanja (npr. putem javne adrese otkrivene STUN-om), a ako to nije moguće zbog ograničenja NAT-a ili vatrozida, komunikacija se preusmjerava preko TURN poslužitelja koji djeluje kao relej između sudionika. Na taj način faza povezivanja dinamički prilagođava mrežnu putanju kako bi osigurala da se medijski tokovi mogu prenositi i u složenim mrežnim okruženjima, pri čemu se uvijek preferiraju najizravnije i najmanje „skupe” putanje [18].



**Slika 8:** ICE kandidati i testiranje parova (Izvor: [webrtcforthe curious.com](http://webrtcforthe curious.com))

U trećoj fazi, **osiguravanju veze**, WebRTC osigurava da se svi medijski i podatkovni tokovi prenose sigurno. U tu svrhu koristi kombinaciju dvaju protokola: DTLS (engl. *Data-gram Transport Layer Security*) i SRTP (engl. *Secure Real-time Transport Protocol*). DTLS je UDP inačica TLS-a koji se koristi u HTTPS-u, zadužen za sigurno pregovaranje kriptografskih parametara i razmjenu ključeva preko nepouzdanog UDP transporta, pri čemu štiti od prisluškivanja, krivotvorenja poruka i njihove izmjene u prijenosu [18].

SRTP nadograđuje RTP tako da dodaje enkripciju sadržaja medijskih paketa, provjeru integriteta i zaštitu od ponovnog reproduciranja paketa (engl. *replay attacks*), ali pritom ostaje dovoljno lagan da zadovolji zahtjeve komunikacije u stvarnom vremenu. U WebRTC-u se ključevi koji se koriste u SRTP-u ne šalju kao *plain-text*, već se izvode iz DTLS sesije u postupku poznatom kao DTLS-SRTP – najprije se između sudionika uspostavlja siguran DTLS kanal, zatim se iz njega generira potrebni ključni materijal za SRTP, a nakon toga se audio i video tokovi šifrirano prenose upravo preko SRTP-a. Time WebRTC zahtijeva da sav medijski promet bude enkriptiran i autentificiran, bez mogućnosti isključenja enkripcije

u produkcijskim implementacijama, čime se krajnjim korisnicima osigurava visoka razina povjerljivosti i zaštite komunikacije [18].

Konačno, završna faza odnosi se na **stvarnu komunikaciju**. WebRTC koristi dva protokola za stvarnu razmjenu podataka između sudionika: RTP i SCTP. RTP je zadužen za prijenos audio i video tokova u stvarnom vremenu: paketi s medijskim okvirima nose vremenske oznake i redne brojeve, što omogućuje sinkronizaciju, detekciju gubitka paketa i prilagodbu reprodukcije mrežnim uvjetima. U WebRTC-u se RTP gotovo uvijek koristi u kombinaciji sa SRTP-om, koji dodaje enkripciju i zaštitu integriteta, pa se stvarni medijski promet prenosi kao SRTP paketi preko ranije uspostavljenog sigurnog kanala. Za proizvoljne podatkovne tokove WebRTC koristi SCTP (engl. *Stream Control Transmission Protocol*) inkapsuliran unutar DTLS-a, što omogućuje pouzdanu, redosljednu komunikaciju između sudionika, uz logičko dijeljenje prometa u više neovisnih strujanja (engl. *streams*) [18].

## 2.5. Ostali alati

Osim glavnih tehnologija koje su temelj aplikacije, u procesu razvoja korišteni su i brojni drugi alati koji su olakšali razvoj, verzioniranje, isporuku i održavanje kôda.

Kao uređivač kôda korišten je **Visual Studio Code**, uz popularne ekstenzije za već prethosno spomenute tehnologije. **git** je korišten za verzioniranje kôda, uz **GitHub** kao odabranu platformu za objavu repozitorija. **Vercel** je odabran kao platforma za *hosting* i *deployment* Next.js aplikacije, zbog svoje duboke integracije s Next.js-om, automatskog skaliranja, globalne mreže isporuke sadržaja (CDN) i jednostavnog procesa isporuke putem integracije s GitHub-om. Platforma **Resend** korištena je za slanje obavijesti korisnicima e-poštom, zahvaljujući svojoj jednostavnosti, pouzdanosti i modernom API-ju koji se lako integrira s okruženjem Node.js.

### **3. Prikaz praktičnog dijela**

#### **3.1. Potpoglavlje**

*TODO: Add your content here*

##### **3.1.1. ABC**

*TODO: Add your content here*

##### **3.1.1.1. 123**

*TODO: Add your content here*

##### **3.1.1.2. 456**

*TODO: Add your content here*

##### **3.1.2. DEF**

*TODO: Add your content here*

#### **3.2. Još jedno potpoglavlje**

*TODO: Add your content here*

## **4. Zaključak**

U zadnjem poglavlju dan je zaključak.

## Literatura

- [1] B. Jokić and Z. Ristić Dedić, “Privatne instrukcije u Republici Hrvatskoj: biznis iz sjene kojeg pandemija nije ugrozila,” <http://idiprints.knjiznica.idi.hr/id/eprint/1048> (posjećeno 30.1.2026.), Institut za društvena istraživanja, Zagreb, Project Report, 2022.
- [2] —, “Organizirane pripreme za državnu maturu i prijemne ispite: obrazovni biznis koji u Hrvatskoj i dalje raste,” <http://idiprints.knjiznica.idi.hr/id/eprint/1046> (posjećeno 30.1.2026.), Institut za društvena istraživanja, Zagreb, Project Report, 2022.
- [3] <https://github.com/vercel/next.js> (posjećeno 30.1.2026.).
- [4] OpenJS Foundation, “About Node.js®,” <https://nodejs.org/en/about/> (posjećeno 30.1.2026.).
- [5] Microsoft, “TypeScript Docs,” <https://www.typescriptlang.org/docs/> (posjećeno 30.1.2026.).
- [6] npm, Inc., “npm Docs,” <https://docs.npmjs.com/> (posjećeno 30.1.2026.).
- [7] <https://github.com/facebook/react> (posjećeno 30.1.2026.).
- [8] Meta Platforms, Inc., “React,” <https://legacy.reactjs.org/> (posjećeno 30.1.2026.).
- [9] —, “React Reference Overview,” <https://react.dev/reference/react> (posjećeno 30.1.2026.).
- [10] Vercel, Inc., “Next.js Docs,” <https://nextjs.org/docs> (posjećeno 30.1.2026.).
- [11] shadcn at Vercel, “shadcn/ui Docs,” <https://ui.shadcn.com/docs> (posjećeno 30.1.2026.).
- [12] Radix by WorkOS, “Radix UI - Primitives: Introduction,” <https://www.radix-ui.com/primitives/docs/overview/introduction> (posjećeno 30.1.2026.).
- [13] Tailwind Labs, Inc., “Tailwind CSS Docs - Core Concepts: Styling with Utility Classes,” <https://tailwindcss.com/docs/styling-with-utility-classes> (posjećeno 30.1.2026.).
- [14] The PostgreSQL Global Development Group, “PostgreSQL - About,” <https://www.postgresql.org/about/> (posjećeno 30.1.2026.).
- [15] Prisma Data, Inc., “Prisma Docs,” <https://www.prisma.io/docs/> (posjećeno 30.1.2026.).
- [16] Supabase Inc., “Supabase Docs,” <https://supabase.com/docs> (posjećeno 30.1.2026.).

- [17] Google for Developers, “WebRTC - Real-Time Communication for the Web,” <https://webrtc.org/> (posjećeno 30.1.2026.).
- [18] S. DuBois, C. Mogren, A. Zhukov, and webrtcforthe curious.com team, *WebRTC for the Curious*. webrtcforthe curious.com, 2020, <https://github.com/webrtc-for-the-curious/webrtc-for-the-curious> (posjećeno 30.1.2026.).
- [19] World Wide Web Consortium, “Web Real-Time Communications (WebRTC) transforms the communications landscape; becomes a World Wide Web Consortium (W3C) Recommendation and multiple Internet Engineering Task Force (IETF) standards,” *W3C Press Releases*, 2021, <https://www.w3.org/press-releases/2021/webrtc-rec/> (posjećeno 30.1.2026.).

# Dodatci

## Popis slika

|   |  |    |
|---|--|----|
| 1 | WebRTC kao kolekcija protokola (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> )          | 10 |
| 2 | Topologija „1 na 1” (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . .           | 11 |
| 3 | Topologija <i>full mesh</i> (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . .   | 11 |
| 4 | Topologija <i>hybrid mesh</i> (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . . | 12 |
| 5 | Topologija SFU (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . .                | 12 |
| 6 | Topologija MCU (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . .                | 12 |
| 7 | Signalizacija (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> ) . . . . .                 | 13 |
| 8 | ICE kandidati i testiranje parova (Izvor: <a href="http://webrtcforthe curious.com">webrtcforthe curious.com</a> )       | 15 |

## Popis tablica

## Popis ispisa kôda