SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Ime Prezime

Naslov rad

DIPLOMSKI/ZAVRSNI RAD

Pula, svibanj, 2022. godine

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Ime Prezime

Naslov rad

DIPLOMSKI/ZAVRSNI RAD

JMBAG: 12342, redoviti/izvanredni student
Studijski smjer: Informatika

Kolegij: Ime kolegija
Znanstveno područje : Društvene znanosti
Znanstveno polje : Informacijske i komunikacijske znanosti
Znanstvena grana : Informacijski sustavi i informatologija

Mentor: Ime mentora

Pula, svibanj, 2022. godine

Sažetak

Odabir optimalne strategije iscrtavanja web aplikacije (CSR, SSR, SSG, ISR) ključan je za postizanje dobrih performansi i korisničkog iskustva. Ovaj rad usporedno analizira navedene strategije u tri popularna programska okvira za izradu web aplikacija Next.js, Nuxt.js i SvelteKit. U svakom od programskih okvira implementirana je identična demo aplikacija sa stranicama statičnog i dinamičnog sadržaja, te su nad njima vršeni testovi ključnih metrika performansi prikupljeni Lighthouse CLI alatom u kontroliranim mrežnim uvjetima. Rezultati prikazuju prednosti i mane svake strategije iscrtavanja ovisno o tipu stranica, te koji programski okviri postižu najbolje performanse u kojim strategijama iscrtavanja.

Ključne riječi : CSR, SSR, SSG, ISR, Next.js, Nuxt.js, SvelteKit, performanse

Abstract

Selecting the optimal rendering strategy for a web application (CSR, SSR, SSG, ISR) is crucial for achieving good performance and user experience. This paper provides a comparative analysis of these strategies across three popular web application frameworks: Next.js, Nuxt.js, and SvelteKit. An identical demo application containing both static and dynamic content pages was implemented in each framework, and key performance metrics were tested using the Lighthouse CLI tool under controlled network conditions. The results highlight the advantages and disadvantages of each rendering strategy depending on the page type, as well as which frameworks achieve the best performance under specific rendering strategies.

Keywords : CSR, SSR, SSG, ISR, Next.js, Nuxt.js, SvelteKit, performance

Sadržaj

1 Uvod

U današnjem okruženju razvoja web aplikacija, više nego ikada prije, vidljiva je težnja i potreba industrije dolazi za postizanjem boljih performansi, skalabilnosti i korisničkog iskustva. Ta potreba potaknula je istraživanje i razvoj novih rješenja kao odgovor na pitanje kako najoptimalnije i najbrže dostaviti web sadržaj korisniku? Kritična stavka u odgovoru na ovo pitanje je upravo odabir odgovarajuće strategije iscrtavanja web aplikacija s obzirom na vrstu sadržaja koji se korisniku servira. Web aplikacije uvelike su evoluirale od starog pristupa serviranja statičnih stranica do današnjih vrlo interaktivnih dinamičnih aplikacija koje zahtijevaju sofisticirane pristupe dostavljanja i prikazivanja sadržaja krajnjem korisniku. [1] Tradicionalne metode iscrtavanja poput iscrtavanja na serveru (SSR) gdje se svježi sadržaj generira na svaki zahtjev korisnika i generiranja statičnih stranica (SSG) kod koje se svaka stranica unaprijed iscrtava u statičnu datoteku već duže vrijeme čine temelj i osnovni standard u razvoju web aplikacija. Povećanje kompleksnosti aplikacija i potreba za sve višim performansama izrodili su i nove hibridne metode iscrtavanja poput inkrementalne statičke generacije (ISR).

Svaka od ovih metoda ima specifične prednosti i nedostatke koji se odražavaju na metrike poput početnog vremena učitavanja sadržaja, razine interaktivnosti, optimizacije za tražilice (SEO) i dr. Odabir strategije iscrtavanja uvelike utječe na percepciju krajnjeg korisnika, troškove infrastrukture te kompleksnost u razvoju i arhitekturi aplikacije.

Popularni programski okviri za razvoj web aplikacija kao što su Next.js, Nuxt i SvelteKit podržavaju većinu novih strategija iscrtavanja čime otvaraju programerima mnoge mogućnosti za optimiziranje procesa dostave sadržaja korisniku.

Pri odabiru programskog okvira uzimaju se u obzir mnogi faktori, a jedan od njih su i performanse, koje direktno utječu na korisničko iskustvo pogotovo u ograničenim mrežnim uvjetima. U ovom radu provedena je usporedna analiza performansi sva tri programska okvira u svakoj od četiri odabrane strategije iscrtavanja (CSR, SSR, SSG i ISR). Postavljeni ciljevi su:

1. Izrada i implementacija funkcionalno i stilski identične web aplikacije sa pod stranicama dinamičnog i statičnog sadržaja (blog).

2. Sustavno mjerenje i usporedba odabranih metrika performansi što uključuje: vrijeme do prvog bajta (TTFB), prvo iscrtavanje sadržaja (FCP), iscrtavanje najvećeg sadržaja (LCP), ukupno vrijeme blokiranja (TBT), vrijeme do interaktivnosti (TTI), veličinu paketa (bundle size) i vremena izgradnje (build time) nad svakom kombinacijom programskog okvira i strategije iscrtavanja te na sve 3 definirane pod stranice.

3. Utvrđivanje prednosti i nedostataka svake strategije iscrtavanja unutar svakog programskog okvira analizom prikupljenih podataka

4. Na temelju rezultata analize donijeti zaključke o tome koji programski okvir nudi najbolje performanse s obzirom na odabranu strategiju iscrtavanja.

Kako bi rezultati bili reprezentativni i usporedivi sa stvarnim korisničkim iskustvom, sva mjerenja izmjerena su u stvarnim uvjetima na aplikacijama postavljenima na platformi Vercel, koja u trenutku pisanja ovog rada jedina nudi mogućnost postavljanja sve četiri strategije iscrtavanja u svim navedenim programskim okvirima. Za testiranje je osigurana stabilna i brza internetska veza prema Internetu, no sami testovi provedeni su uz postavljena ograničenja kako bi se jasnije istaknule razlike u performansama. Odabrano je ograničenje veze koje simulira sporu 4G vezu te dvostruko smanjenje brzine procesora.

Ovo istraživanje nastoji pružiti bolji uvid u trenutne mogućnosti iscrtavanja koje su dostupne web developerima, te im time pomoći u donošenju boljih odluka pri odabiru programskog okvira i strategije iscrtavanja za svoju web aplikaciju. Prvi dio rada baviti će se pregledom najpopularnijih strategija iscrtavanja i programskih okvira, a drugi dio prezentacijom i analizom dobivenih rezultata, raspravom te zaključnim razmatranjima.

1.1 Strategije iscrtavanja

Pojam strategije iscrtavanja odnosi se na proces pretvaranja koda u vizualni sadržaj koji korisnik može vidjeti i s kojim može vršiti interakciju kroz web preglednik. [1] Ovaj proces utječe na korisničko iskustvo određujući koliko brzo će se sadržaj učitati korisniku i koliko će aplikacija biti prilagodljiva i dinamična. Odabir strategije iscrtavanja također bitno utječe i na SEO tj. vidljivost kod pretraživanja internetskim tražilicama. [2]

Moderne SPA aplikacije najčešće podržavanju različite strategije iscrtavanja, a određene platforme poput Vercela i Netlifyja pružaju dodatnu podršku popularnim programskim okvirima, olakšavajući njihovu konfiguraciju i integraciju. Slijedi kratak pregled strategija iscrtavanja obrađenih u ovom radu.

1.1.1 Iscrtavanje na strani klijenta (CSR)

Ova metoda iscrtavanja nastala je još početkom 21. stoljeća razvojem programskih okvira poput AngularJS-a, Reacta i Vuea kada je nastao veliki prijelaz u industriji sa monolitne web arhitekture na tzv. jednostranične aplikacije (SPA) koje se za ažuriranje sučelja, navigaciju i dohvaćanje podataka oslanjaju na JavaScript kod koji se izvršava u pregledniku.

Kod ove strategije iscrtavanja, na klijent se šalje jedan prazan HTML dokument, zajedno sa svim drugim resursima (CSS i JavaScript paketi). Umjesto da je sav sadržaj već unesen u HTML dokument i odmah spreman za iscrtavanje u pregledniku, preglednik najprije mora pričekati da se preuzme sav potreban JavaScript kod, te se njegovim izvršavanjem HTML dokument ispunjava elementima koji će se iscrtati. Kod navigacije između stranica, ne dolazi do dohvaćanja novog HTML dokumenta, već JavaScript ažurira postojeći HTML novim sadržajem oslanjajući se na AJAX i XML. Time se izbjegavaju ponovna učitavanja koja usporavaju rad aplikacije i štede mrežne resurse. [3]

Prednosti ove strategije su:

- Responzivnost i interaktivnost – promjene na stranici vidljive su odmah, nema potrebe za dohvaćanjem nove stranice prilikom navigacije.
- Ušteda resursa poslužitelja – umjesto dohvaćanja cijelog novog HTML dokumenta za svaku stranicu, dohvaća se samo onaj dio podataka koji je potreban (najčešće u JSON formatu)
- Mogućnost korištenja aplikacije kada mrežna veza nije dostupna (offline) – ovakva funkcionalnost postiže se pametnim predmemoriranjem (engl. caching).
- Uvijek svježi podaci – budući da stranice nisu prethodno statički generirane, osigurava se svježina prikazanih podataka.

Nedostaci ove strategije su:

- Početna brzina učitavanja – iako je aplikacija generalno brža nakon početnog učitavanja JavaScripta, prvo učitavanje kada korisnik posjeti stranicu može biti znatno sporije zbog potrebe da se učita i izvrši sav potreban JavaScript kod prije iscrtavanja sadržaja. Ovo ograničenje posebno dolazi do izražaja prilikom učitavanja na sporijim mrežama.
- Lošiji SEO – iako današnji pretraživači
  footnoteEng. crawler – automatizirani robot koji tražilica koristi za istraživanje, pronalazak i indeksiranje web stranica [4] mogu očitati i stranice iscrtane ovom strategijom, primarno su optimizirani su za čitanje statičnog HTML-a. Zbog ovog nedostatka, ova se strategija se sve manje koristi za web aplikacije kod kojih je važan dobar SEO.
- Lošije performanse na slabijim uređajima – zbog činjenice da je potrebno najprije izvršiti JavaScript kod na strani klijenta, kod slabijih i starijih uređaja može doći do usporavanja učitavanja i pada performansi.
- Manjak podrške za korisnike koji nisu omogućili JavaScript u pregledniku – bez JavaScripta sadržaj stranice se ne može iscrtati.

S obzirom na navedene prednosti i nedostatke ova strategija iscrtavanja najbolja je za tipove aplikacija koje imaju slijedeća obilježja:

- Potreba za visokom razinom interaktivnosti (dashboards)
- Gdje SEO nije bitan čimbenik (admin i korisničke stranice)
- Progresivne web aplikacije
- Potreba za smanjenjem opterećenja poslužitelja

1.1.2 Iscrtavanje na strani poslužitelja (SSR)

Kako bi se uklonili nedostatci koje sa sobom nosi iscrtavanje na strani klijenta, razvila se nova popularna strategija iscrtavanja – iscrtavanje na strani poslužitelja. Ova strategija se naizgled vraća korak prema već poznatom i prvobitnom načinu iscrtavanja – iscrtavanju na poslužitelju ili serveru i višestraničnoj web aplikaciji (MPA) kakve su postojale od začetaka Interneta.

No moderni programski okviri poput Nuxta i SvelteKita spajaju SSR i CSR na način da se prilikom prvobitnog posjeta stranici ona iscrtava na poslužitelju, te se korisniku šalje već popunjen HTML dokument koji preglednik može odmah krenuti prikazivati, a u pozadini se događa proces zvan hidracija. Ovo je proces u kojem se učitava sav potreban JavaScript kod koji se izvršava i povezuje sa HTML elementima stranice te čini stranicu interaktivnom, slično kao kod CSR strategije. Svaki slijedeći zahtjev za navigaciju između stranica ili ažuriranje podataka događa se na strani klijenta i funkcionira kao CSR aplikacija.

Next.js programski okvir u novijim verzijama koristi tzv. iscrtavanje na strani poslužitelja u stvarnom vremenu (Streaming SSR). U ovom načinu rada, na serveru se postupno iscrtavaju zasebni dijelovi HTML dokumenta po redoslijedu kako podaci postaju dostupni i takav dokument se odmah šalje pregledniku na prikaz. Ovime se nastoji izbjeći nedostatak klasičnog SSR-a a to je čekanje na poslužitelj da generira cijeli HTML dokument prije nego ga pošalje klijentu. [5]

Kako bi SSR strategija mogla funkcionirati, na poslužitelju je potrebno odgovarajuće okruženje koje podržava izvođenje JavaScript koda i generiranje HTML stranica. To je najčešće Node.js [6]

Prednosti ove strategije su:

- Nema čekanja na učitavanje JavaScripta – budući da se na zahtjev klijenta na poslužitelju generira HTML datoteka koja mu se odmah dostavlja spremna za prikaz, nema potrebe za čekanjem na preuzimanje i izvršavanje JavaScript koda.
- Bolji SEO – statičke HTML datoteke popunjene sadržajem mnogo su bolje za mrežne pretraživače, koji iz njih brže i lakše dolaze do relevantnih podataka o stranici.
- HTML datoteka mogu se spremiti u pričuvnu memoriju preglednika, što omogućava pregled stranice i kada nema pristupa internetu.
- Svježina podataka – na svaki zahtjev klijenta generira se novi HTML dokument sa ažuriranim podacima.

Nedostaci ove strategije su:

- Čekanje na poslužitelju – iako nema čekanja na izvršavanje JavaScript koda (prije prikaza sadržaja) na strani klijenta, prilikom prve posjete stranici potrebno je pričekati na poslužitelja da izgradi HTML dokument koji nije prethodno generiran. Prethodno spomenuti streaming SSR pokušava ublažiti i ovaj nedostatak.
- Čekanje do interaktivnosti – unatoč brzom prikazu početnog sadržaja stranice, ipak je potrebno pričekati na proces hidracije kako bi stranica postala interaktivna. To uključuje preuzimanje JavaScript paketa i izvršavanje koda što može blokirati glavnu procesorsku nit i time povećati vrijeme do interaktivnosti (TTI).
- Povećano korištenje resursa poslužitelja – generiranje HTML dokumenata na svaki zahtjev korisnika rezultira i većom potrošnjom računalnih resursa poput memorije i procesorske snage, što je kod CSR-a prebačeno na klijentski uređaj.
- Složeniji proces razvoja – zbog činjenice da se određeni dijelovi koda mogu izvršavati samo na klijentu ili samo na poslužitelju, raste i kompleksnost u razvoju aplikacije. [3]

S obzirom na navedene prednosti i nedostatke ova strategija iscrtavanja najbolja je za tipove aplikacija koje imaju slijedeća obilježja:

- Stalna potreba za svježim i ažuriranim podacima
- Visoka razina personalizacije (custom dashboards)
- Vizualizacija podataka u realnom vremenu
- Dobar SEO [1]

1.1.3 Generiranje statičkih stranica (SSG)

Kod ove strategije iscrtavanja HTML stranice se unaprijed generiraju na serveru prilikom izgradnje aplikacije (engl. build time), te se zatim poslužuju klijentima na zahtjev, bez potrebe za ponovnim generiranjem na svakom zahtjevu kao kod SSR-a. Ovakve stranice se mogu i spremiti u priručnu memoriju (engl. cache) koristeći CDN radi brže distribucije korisnicima [7, 8]

Glavne prednosti ove strategije su: [1]

- Najbrže moguće učitavanje stranice – budući da su stranice statični HTML, nije potrebno čekati na njihovo generiranje na poslužitelju ili na preuzimanje i izvršavanje JavaScript koda
- Odlične su za SEO – budući da se brzo učitavanju i sadrže sav potrebni sadržaj idealne su za pregled od strane pretraživača
- Nisko opterećenje poslužitelja – nema potrebe za obradom ili generiranjem HTML-a, već je spreman za slanje klijentu

- Najniži troškovi infrastrukture

Nedostatci ove strategije su:

- Duže vrijeme izgradnje ako postoji veliki broj stranica
- Za ažuriranje sadržaja potrebno je ponovno inicirati izgradnju i postavljanje (build and deploy)
- Nije pogodno za stranice sa dinamičnim i često promjenjivim sadržajem

S obzirom na navedene prednosti i nedostatke ova strategija iscrtavanja najbolja je za stranice kod kojih se sadržaj gotovo nikada ili vrlo rijetko mijenja poput:

- Marketinške stranice
- Blog postovi
- E-commerce proizvodi
- Dokumentacija i pomoć

Generalno pravilo je postaviti pitanje, može li se stranica generirati unaprijed, prije korisničkog zahtjeva? Ako je odgovor potvrdan, SSG se nameće kao logičan izbor. [7]

Postoji mogućnost kombiniranja SSG-a i CSR-a gdje se stranica servira sa prethodno generiranim statičnim dijelom koji se ne mijenja, a po učitavanju JavaScripta na klijentu se ispunjavaju dinamični dijelovi stranice svježim podacima. Ovo je alternativa korištenju SSR strategije, koja također ima svoje prednosti i nedostatke. [7]

1.1.4 Inkrementalna statička generacija (ISR)

Ovu strategija iscrtavanja neki nazivaju i hibridnim web razvojem jer kombinira generiranje statičnih stranica (SSG) sa iscrtavanjem na strani poslužitelja (SSR).

Funkcionira na način da se najprije generira statična stranica kojoj se odredi vrijeme revalidacije, tj. vrijeme nakon kojega će se ona smatrati zastarjelom i biti će ju potrebno ponovno generirati. No okidač za ovu generaciju biti će zahtjev prvog posjetitelja nakon vremena isteka revalidacije. Tom posjetitelju će se odmah isporučiti ova stara verzija stranice bez obzira koliko je dugo prekoračeno njeno vrijeme validacije, te će se nakon uspješne ponovne generacije u pozadini, nova verzija stranice poslužiti prvom slijedećem klijentu. Na ovaj način novo generirana stranica se dodaje u web aplikaciju.

Uobičajeni način objavljivanja na poslužitelj jest atomsko objavljivanje, tj. kada se cjelokupni kod, resursi i konfiguracija ažuriraju u isto vrijeme. Ovaj način

objavljivanja čuva integritet stranice i omogućava jednostavan opoziv objave i povratak na prijašnju verziju u slučaju potrebe (engl. rollback). Svaka pojedinačna objava je jedna cjelovita verzija web aplikacije. ISR strategija razbija ovaj integritet budući da stalno nadodaje novo generirane stranice u web aplikaciju, odvojeno od početno izgrađenog koda. Zbog ovog spremanja u pričuvnu memoriju (engl. caching) je vrlo teško učiniti povratak na prethodnu verziju, a pri tome osigurati da svi korisnici dobiju istu verziju stranice. [9]

Potrebno je naglasiti da se ova strategija oslanja na mrežu za isporuku sadržaja (CDN) koju pruža platforma na koju je postavljena web aplikacija drastično olakšavajući cijeli proces konfiguracije i postavljanja. Ovu metodu moguće je implementirati i neovisno o platformi, ali to podiže kompleksnost.

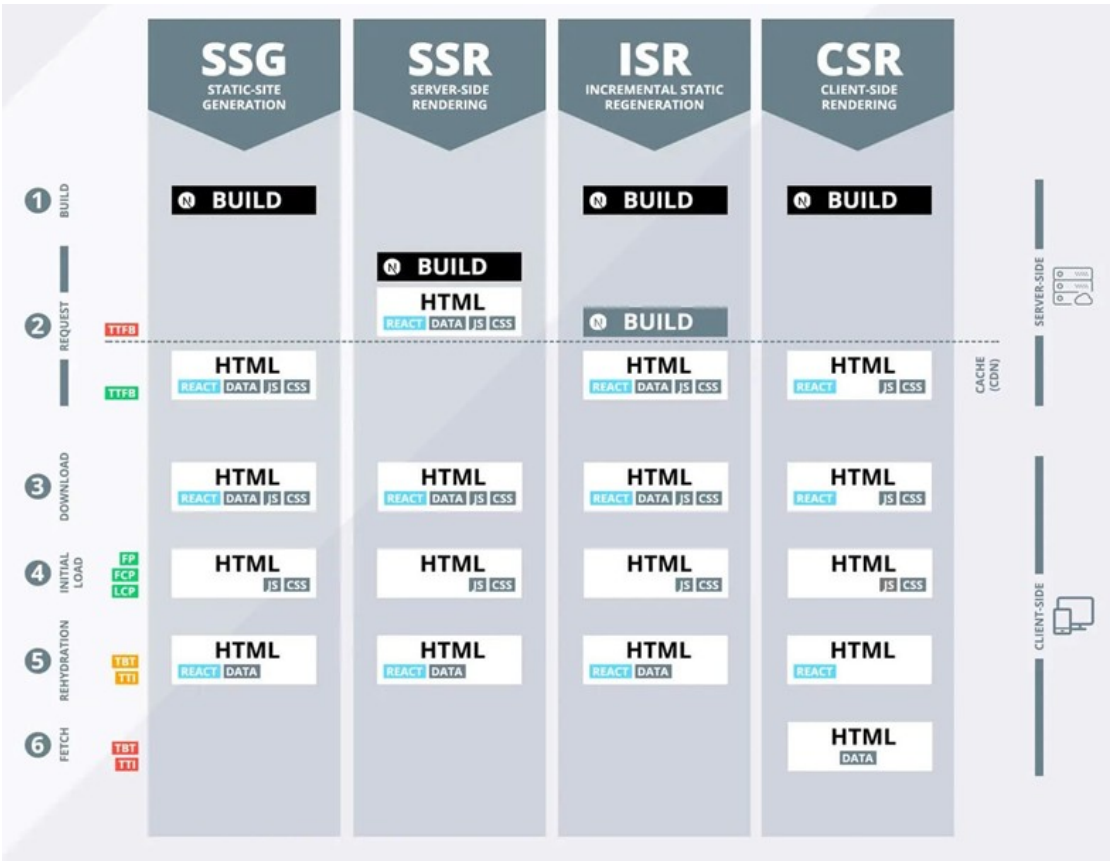
Implementacija ove strategije uvelike se razliku među programskim okvirima. Npr. Next.js generira statične stranice za dinamične rute prilikom vremena izgradnje aplikacije (build time) te ih obnavlja na prvu posjetu korisnika nakon isteka vremena revalidacije. Nuxt pak generira traženu stranicu tek na prvi zahtjev korisnika a ne prilikom izgradnje aplikacije. [10]

Prednosti ove strategije su:

- Visoke performanse poput SSG strategije – za većinu korisnika koji dobiju prethodno spremljenu stranicu putem CDN-a.
- Mogućnost periodičnog ažuriranja stranica novim sadržajem bez potrebe za ponovnom izgradnjom čitave aplikacije
- Niži trošak od SSR-a
- Efektivno skaliranje do velikog broja stranica
- Odličan SEO

Nedostatci ove strategije su:

- Korisnici koji prvi posjete stranicu nakon isteka perioda revalidacije dobivaju zastarjelu verziju stranice
- Potrebna veća razina konfiguriranja – postavljanje optimalnog vremena revalidacije na svaki tip stranice posebno
- Složenije debugiranje – teže razumjevanje problema koji nastanu zbog spremanja u priručnu memoriju (engl. cache). [9]



Slika 1: Pregled strategija iscrtavanja u Next.js programskom okviru

2 Zaključak

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Literatura

[1] A. A. Moore. How to choose the best rendering strategy for your app. Vercel, Srpanj 2024. [Mrežno]. Available: https://vercel.com/blog/how-to-choose-the-best-rendering-strategy-for-your-app. [Pokušaj pristupa 12 Lipanj 2024].

[2] P. Bratslavsky. What is website rendering: Csr, ssr, and ssg explained. strapi, Svibanj 2025. [Mrežno]. Available: https://strapi.io/blog/what-is-website-rendering. [Pokušaj pristupa 13 Lipanj 2025].

[3] I. Beran. Usporedba metoda renderiranja web aplikacija, Sječanj 2023. [Mrežno]. Available: https://repozitorij.pmfst.unist.hr/islandora/object/pmfst:1621. [Pokušaj pristupa 13 Lipanj 2025].

[4] Google. In-depth guide to how google search works. [Mrežno]. Available: https://developers.google.com/search/docs/fundamentals/how-search-works?hl=en. [Pokušaj pristupa 14 6 2025].

[5] Next.js. Loading ui and streaming. [Mrežno]. Available: https://nextjs.org/docs/14/app/building-your-application/routing/loading-ui-and-streaming. [Pokušaj pristupa 14 Lipanj 2025].

[6] Vue.js. Server-side rendering (ssr). [Mrežno]. Available: https://vuejs.org/guide/scaling-up/ssr.html. [Pokušaj pristupa 14 Lipanj 2025].

[7] Next.js. Static site generation (ssg). [Mrežno]. Available: https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation. [Pokušaj pristupa 14 Lipanj 2025].

[8] Sanity. Static site generation (ssg) definition. [Mrežno]. Available: https://www.sanity.io/glossary/static-site-generation. [Pokušaj pristupa 14 Lipanj 2025].

[9] Incremental static regeneration: Its benefits and its flaws. Netlify, Ožujak 2021. [Mrežno]. Available: https://www.netlify.com/blog/2021/03/08/incremental-static-regeneration-its-benefits-and-its-flaws. [Pokušaj pristupa 15 Lipanj 2025].

[10] O. Troyan. Comparing nuxt 3 rendering modes: Swr, isr, ssr, ssg, spa. RisingStack, Svibanj 2024. [Mrežno]. Available: https://blog.risingstack.com/nuxt-3-rendering-modes/#isr. [Pokušaj pristupa 15 Lipanj 2025].

Popis slika

Popis tablica