Zbornik radova Fakulteta tehničkih nauka, Novi Sad

**UDK: 659.2:004**
**DOI: https://doi.org/10.24867/03OO03Cicmil**

# UNAPREĐENJE SAJTA ZA OBJAVU OGLASA UPOTREBOM RENDEROVANJA NA KLIJENTSKOJ STRANI

# IMPROVEMENT OF AD SEARCH SITE USING RENDERING ON CLIENT SIDE

Milica Cicmil, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – INŽENJERSTVO INFORMACIONIH SISTEMA**

**Kratak sadržaj –** *U ovom radu prikazani su opis problema sajta za pretragu oglasa, tehnologije korišćene za poboljšanje njegovih performansi, kao i metrike, rešenja i krajnji rezultat do kog se došlo reinženjeringom istog. Kao osnovni uzrok nedostataka došlo se do razlike renderovanja stranica na serveru i na klijentu, te su u ovom radu ta razlika, prednosti i mane oba pristupa detaljnije opisani.*

**Ključne reči:** *Renderovanje, Server, Klijent, React, Symphony*

**Abstract** – *This paper describes the problem of the site for ad search, the technologies used to improve its performance, as well as the metrics, solutions and the ultimate result of its reengineering. As the main cause of disadvantages, there was a difference between rendering pages on the server and on the client side. Therefore, here you will find this difference, advantages and disadvantages of both approaches in details.*

**Keywords:** *Rendering, Server, Client, React, Symphony*

## 1. UVOD

Veliki tehnološki napredak doneo je sa sobom da se i zahtevi korisnika, koji danas najčešće komuniciraju putem interneta, svakodnevno povećavaju. Sajtovi koji danas brže i efikasnije funkcionišu, pomeraju granice i postaju konkurencija tradicionalnom poslovanju na tržištu.

Pred svakog razvojnog inženjera, svakim danom se postavlja novi zahtev za boljim performansama njegovog softverskog proizvoda. Kako bi se išlo u korak sa trendovima i rastućim zahtevima korisnika sajtova, postavljen je zadatak da se omoguće najbolje moguće performanse tog sajta.

Odatle ideja za unapređenje sajta sa performansama koje nisu zadovoljavajuće primenom metoda, tehnika i alata, kako bi se taj sajt podigao na viši nivo, poboljšale se njegove performanse, i on se prilagodio krajnjim korisnicima.

**NAPOMENA:**
**Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Stefanović, vanredni profesor**

## 2. OPIS PROBLEMA

Da bi se plasirao jedan konkurentan proizvod neophodno je biti u mogućnosti da se evidentiraju svi nedostaci, koji se ogledaju ne samo u brzini pokretanja stranica, već i u načinu prikaza stranica.

U početku je konvencionalni metod za prikaz HTML (*hypertext markup langugage*) stranice na ekranu, bio rendering na serveru. Ovaj način funkcioniše zadovoljavajuće sve dok se na stranicama prikazuju samo statički tekstovi i slike [5]. Savremeni sajtovi funkcionišu pre kao aplikacije koje samo izgledaju kao da su veb stranice. Na njima se mogu slati poruke, ažurirati informacije na mreži, ili vršiti razne kupovine.

Sagledavanje mana i problema sajta korišćenog u svrhu istraživanja, primećeno je sledeće:

- poruka koju klijent dobija zauzima više memorije, sporije se učitava,
- za promenljive podatke, uvek je potrebno uraditi *reload* na stranici i
- dok se ne dobije odgovor od servera, ne može se nastaviti sa daljim zahtevima.

### 2.1. Renderovanje stranice na serverskoj strani

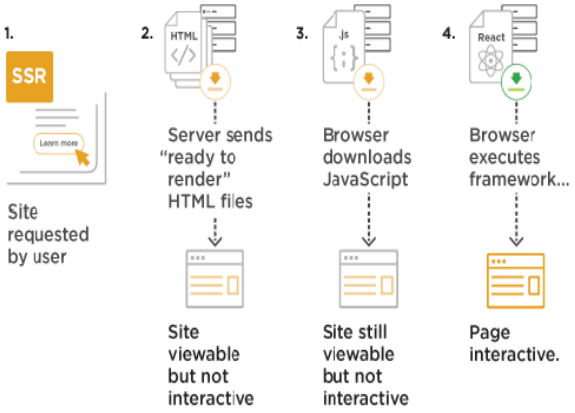
Sajt koji je posmatran, za renderovanje stranica koristio je serversko renderovanje, koje u konkretnom slučaju nije bilo dobro rešenje.

Renderovanje na serverskoj strani je najzastupljeniji način prikaza informacija na ekranu. Funkcioniše tako što generiše HTML sadržaj na serveru i generisani HTML vraća kao odgovor klijentu. Kada se poseti neki sajt, pretraživač formira zahtev za server koji sadrži informacije sa sajta. Za obradu zahteva potrebno je svega nekoliko milisekundi, ali to obično zavisi od više faktora [3]:

- brzina interneta,
- lokacija servera,
- broj korisnika koji pokušavaju da pristupe istom sajtu i
- koliko je sajt optimizovan.

Kada je zahtev procesuiran, pretraživaču će se vratiti potpuno izrenderovan HTML i prikazaće se na ekranu. Ako se odluči posetiti druga stranica na sajtu, pretraživač će napraviti novi zahtev za novom stranicom. Bez obzira na to da li nova stranica sadrži samo nekoliko drugačijih informacija od prve otvorene stranice, pretraživač će uvek tražiti od servera celu novu stranicu i opet renderovati sve

od početka. Zbog toga dolazi do čestih zahteva ka serveru, prosečno je veoma dugo vreme potrebno za renderovanje stranice, dolazi do ponovnog učitavanja svake stranice, što dovodi do sporijeg rada sajta. Na slici 1, može se jasno videti način funkcionisanja renderovanja na serverskoj strani, gde je problem u HTML fajlu kojim server prvenstveno odgovara, nakon što mu korisnik pošalje zahtev. Dok pretraživač učitava ostatak sadržaja koji čini stranicu interaktivnom, korisnik mora da čeka, jer treba više vremena dok neki delovi postanu interaktivni.




Slika 1. *Primer funkcionisanja renderovanja na serverskoj strani [1]*

## 3. METODE UNAPREĐENJA SAJTA ZA OBJAVU OGLASA

### 3.1. *Symphony* okvir

U osnovi, *Symphony* okviri se sastoje od:

- kompleta alata – skup brzo integrišućih softverskih komponenti, što znači da se može napisati manje koda uz manje rizika od grešaka. To takođe znači i veću produktivnost i sposobnost da se posveti više vremena onim stvarima koje pružaju dodatnu vrednost, kao što su upravljanje osnovnim principima ili neželjenim efektima,
- metodologija – struktuirani pristupi pri izradi aplikacije. Ovi pristupi mogu izgledati ograničavajuće, ali u stvarnosti dozvoljavaju razvojnim inženjerima da efikasno rade na najkompleksnijim aspektima zadatka, a korišćenje najboljih praksi garantuje stabilnost, održivost i nadogradivost aplikacija koje se razvijaju.

*Symphony* je jedan od najanprednijih PHP okvira. Dve najvažnije tehnološke prednosti *Symphony*-ja su paketi i komponente. Paket se može se posmatrati kao paket datoteka (PHP fajlovi, stilovi, *JavaScript*, slike) za implementaciju bilo koje funkcije. Glavna prednost ovih paketa je što su razdvojeni. Mogu se ponovo konfigurisati i koristiti za mnoge aplikacije, kako bi se smanjili ukupni troškovi razvoja. Komponente skraćuju vreme za rutinske zadatke i omogućavaju programerima da se fokusiraju na druge poslovne karakteristike. Postoji 30 *Symphony* komponenti koje olakšavaju razvojni proces. Mogu se samostalno koristiti komponente i dodavati sopstveni modeli bez uticaja na arhitekturu. Ove komponente se mogu samostalno koristiti i u drugim okvirima ili jednostavnim PHP rešenjima. Što je manje zavisnosti u arhitekturi, to će biti lakše napraviti promene bez rizika od rušenja drugih delova sistema. Stoga je rešenje prilagodljivo bilo kojim zahtevima i korisničkim scenarijima, kako bi se kreirala što fleksibilnija aplikacija. *Symphony* je jedan od nekoliko okvira koji su komercijalno podržani. Stabilan je i dobro testiran, sa redovnim ažuriranjem. Svaka nova linija koda treba biti testirana, kako bi se garantovao stabilan rad aplikacije. Ponovno korišćenje paketa, odsustvo stroge zavisnosti u arhitekturi i mogućnost kreiranja šablona dizajna doprinose boljoj održivosti i testiranju u *Symphony*-ju. *Unit* testiranje je jednostavno zbog korišćenja nezavisne biblioteke *PHP unit*. Funkcionalno testiranje je takođe automatizovano kako bi se smanjila rutina nametnuta programerima.

### 3.2. *Restful* servisi

Ovi servisi su jednostavnije integrisani sa HTTP-om od SOAP (*Simple Object Access Protocol*) servisa, ne zahtevaju XML poruke ili WSDL opise servisa. Danas se RESTfull izdvojio kao dominantan mrežni servis, potisnuo je SOAP i WSDL jer je značajno jednostavniji za korišćenje.

Podaci se najčešće prebacuju u JSON formatu mada je dostupan i XML i YAML format. Zasniva se na REST arhitekturi, veoma je fleksibilan i jednostavan za razumevanje. Može biti izvršen na bilo kom klijentu ili serveru koji ima HTTP/HTTPS podršku. RESTful servisi treba da imaju sledeće osobine i karakterisitike [4]:

- nepostojanje stanja (*stateless*),
- mogućnost keširanja (*cacheable*),
- uniformni interfejs (*uniform interface URI*),
- izričito korišćenje HTTP metoda i
- transfer XML i/ili JSON.

Kod ovog tipa servisa, resursi (npr. statičke strane, fajlovi, podaci iz baze podataka) imaju sopstveni URL ili URI koji ih identifikuje. Pristup do resursa je definisan HTTP protokolom, gde svaki poziv čini jednu akciju (kreira, čita, menja ili briše podatke). Isti URL se koristi za sve operacije, ali se menja HTTP metod koji definiše vrstu operacije. REST koristi “CRUD like” HTTP metode kao što su: GET, POST, PUT, DELETE, OPTIONS.

RESTfull servisi se koriste:

- kod ograničenog propusnog opsega i resursa (povratna informacija može biti u bilo kom obliku),
- kod operacija koje ne koriste stanja (ukoliko neka operacija treba da bude nastavljena, onda REST nije pravi pristup i SOAP verovatno predstavlja bolje rešenje) i
- kod situacija gde je moguće keširanje (ukoliko informacija može biti keširana zbog operacija koje ne koriste stanja, onda je ovaj pristup odličan).

Odlike RESTfull servisa su:

- jednostavnost,
- klijenti koji pozivaju REST servise ne moraju da formatiraju zahteve po SOAP specifikaciji i ne moraju da parsiraju SOAP odgovor kako bi iz njega izvukli rezultat,
- fleksibilnost formata vraćenih podataka i
- format u kome se podaci vraćaju nije unapred definisan i zavisi od samog servisa. Klijenti mogu da zatraže podatke u formatu koji im najviše odgovara, za razliku od SOAP formata koji, iako je standardizovan, mora da se parsira. Pa tako JavaScript može dobiti podatke u JSON formatu koga lako može da pročita, a RSS čitač u RSS-XML formatu koji može da prikaže.

### 3.3. React

*React* (poznat i kao React.js ili ReactJS) je Javascrpit biblioteka otvorenog koda koja obezbeđuje pregled podataka zapisanih preko HTML-a. *React* pregledi su obično obezbeđeni korišćenjem komponenti koje sadrže dodatne komponente definisane kao prilagođene HTML oznake. *React* obezbeđuje programeru model u kojem podkomponente ne mogu direktno da utiču na spoljašnje komponente, efikasno je ažuriranje HTML dokumenta pri promeni podataka i jasno je razdvajanje komponenti na današnjim jednostraničnim aplikacijama.

Svojstva, skup nepromenljivih vrednosti, prosleđeni su prikazivaču komponenti kao svojstvo u njegovim HTML oznakama. Komponente ne mogu direktno da promene bilo koje svojstvo koje im je prosleđeno, ali može biti prosleđena opozivajuća funkcija da promeni vrednosti. Ovaj mehanizam je izražen kao „svojstva idu dole; akcije gore“.

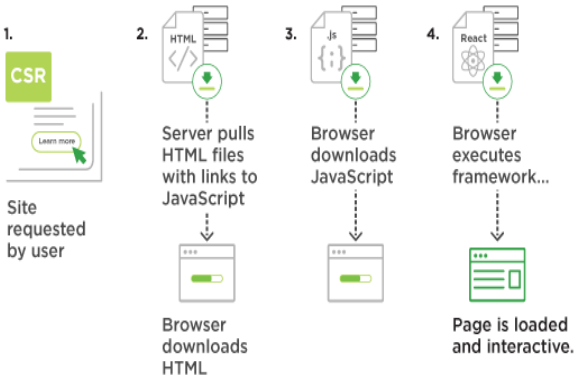
Na primer, komponenta kolica za kupovinu može da uključuje više komponenti proizvoda. Vizuelizacija proizvoda koristi samo svojstva koja su im prosleđena i to ne može da utiče na ukupnu vrednost kolica za kupovinu. Međutim, proizvod može biti prosleđen opozivajućoj funkciji kao svojstvo koje bi bilo pozvano kada se aktivira dugme „obriši ovaj proizvod“ i tada ta funkcija utiče na ukupan račun.

Još jedna specifičnost je da se koristi virtuelni DOM. *React* održava u memoriji keš podatke, izračunava aktuelne razlike, a zatim efikasno ažurira preglednikov DOM. Ovo omogućava programerima da pišu kod kao da se cela strana osvežava pri svakoj promeni, dok *React* biblioteka prikazuje samo podkomponente koje su zaista promenjene.

Na primer, komponente kolica za kupovinu mogu biti zapisane da prikažu celokupan sadržaj kolica pri bilo kojoj promeni podataka. Ukoliko podkomponente proizvoda nemaju promene u svojstvu, biće upotrebljen keširani prikaz. Ovo znači da će relativno spore izmene u preglednikovom DOM-u biti izbegnute. Pored toga, ukoliko se promeni broj proizvoda, podkomponente proizvoda će biti prikazane, a krajnji HTML se može razlikovati u samo jednom članu i samo taj član će biti izmenjen u DOM-u.

## 4. OPIS REŠENJA I METRIKE

U ovom poglavlju rada će biti predstavljeno rešenje u smislu renderovanja stranice na klijentu. Na slici 2, prikazan je rimer renderovanja na klijentskoj strani. Za razliku od pristupa renderovanju na serverskoj strani, server povlači HTML fajlove sa linkovima ka JavaScrip-u, tj. Pretraživač učitava HTML fajl. Sa ovim pristupom dobija se odmah interaktivna učitana stranica.




Slika 2. *Primer funkcionisanja renderovanja na klijentskoj strani [1]*

U narednim redovima, opisane su prednosti ovog pristupa [6]:

- moguće interakcije na „bogatim“ sajtovima,
- brzo renderovanje stranica i nakon učitavanja glavne stranice i
- uspešan za veb aplikacije.

Dok su mane ovog pristupa:

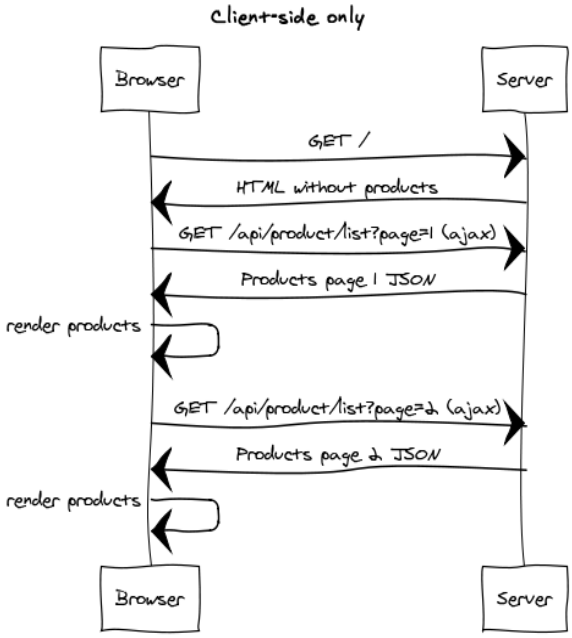
- loš SEO ako nije dobro implementiran i
- u većini slučajeva zahteva eksterne biblioteke.

Za razliku od renderovanja stranice na serveru (što je opisano u prikazu problema), renderovanje na klijentu donosi mnoge prednosti u slučaju sajta za pretragu oglasa. Kod renderovanja na klijentskoj strani, misli se na renderovanje sadržaja u samom pretraživaču, koristeći Java Script. Dakle, umesto preuzimanja celog sadržaja iz HTML dokumenta, dobija se deo HTML-a sa JavaScript fajlom koji će renderovati ostatak sadržaja koristeći pretraživač [4]. Ovo je relativno nov pristup renderingu stranica i nije bio popularan sve dok JS biblioteke nisu počele da ga ugrađuju u svoj stil razvoja. U ovom slučaju, ako se želi otvoriti link ka nekoj drugoj stranici na sajtu, pretraživač neće praviti novi zahtev ka serveru. Ovaj način renderovanja je samim tim i brži, jer se učitava samo mali deo stranice da bi se preuzeo novi sadržaj.

U slučaju ovog istraživanja, dakle, renderovanje na strani klijenta dobija prednost, prvenstveno zato što sa renderingom na serveru, server odgovara stvaranjem i vraćanjem potpuno nove stranice za svaku interakciju. Ovo često usporava vreme učitavanja, koristi više propusnog opsega i stvara lošije iskustvo korisniku. Takođe, klijentsko renderovanje izbegava izradu bespotrebnih zahteva za punom stranicom, ako se samo izmenio deo stranice. Ovo je naročito korisno s obzirom

na to da sve više ljudi pretražuje preko mobilnih telefona. Renderovanje na klijentu, još, podržava bogate animacije i transformacije.

Naravno, to može biti izvedeno i na aplikaciji koja podržava renderovanje na serveru, ali to često vodi ka održavanju istih strana i na klijentu i na serveru. Na narednoj slici (slika 3), lepo se vidi kako, gde i u kom momentu se kreću zahtevi i odgovori kada je u pitanju renderovanje na klijentskoj strani. Umesto vraćanja celog sadržaja HTML fajla, renderovanje i učitavanje samo novog dela sadržaja vrši se na klijentskoj strani, pa je samim tim i odziv stranice brži.




Slika 3. *Dijagram sekvence – način renderovanja na klijentskoj strani [2]*

## 5. ZAKLJUČAK

U okviru ovog rada istraženi su osnovni nedostaci na postojećem sajtu sa oglasima i predložen je novi način renderovanja i tehnologije kojim se može doći do unapređenja. Detaljno su navedene i opisane tehnologije korišćene za poboljšanje performansi, takođe, navedeni su i razlozi i prednosti i mane njihovog korišćenja. Shodno tome, u ovom radu predloženo je da renderovanje bude na klijentskoj strani, s obzirom na to da se ne želi da se čeka uvek učitavanje cele strane.

Takođe, ovaj pristup podrazumeva da se svi potrebni podaci kao što su šabloni i poslovna logika, u potpunosti učitavaju. Nema potrebe za čestim zahtevima ka serveru, sve dok nema potrebe za podacima koji još nisu bili učitani.

Cilj je bio zadovoljiti korisničke potrebe, pružiti im željenu brzinu pretraživanja i preglednost podataka. Ovaj tip renderovanja nije u svakom slučaju najbolje rešenje, ali svakako u slučaju velikog broja stranica, sličnih i različitih, renderovanje na klijentu je jedna od izmena koja je doprinela poboljšanju.

## 6. LITERATURA


[1] https://hackernoon.com/server-side-vs-client-side-rendering-in-react-apps-443efd6f2e87 (pristupljeno u septembru, 2018)

[2] Darko Avramović "Evaluating Web browser graphics rendering system performance by using dynamically generated SVG". U: JOURNAL OF GRAPHIC ENGINEERING AND DESIGN 3.1 (2012)

[3] https://www.upwork.com/hiring/development/ server-side-rendering/ (pristupljeno u septembru, 2018)

[4] Alexander Svensson. "Speed Performance Comparison of JavaScript MVC Frameworks". Thesis. Blekinge Institute of Technology, 2015.

[5] https://medium.freecodecamp.org/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d (pristupljeno u septembru, 2018)

[6] http://branchandbound.net/blog/web/2012/11/unify-server-side-client-side-rendering-embedding-json/ (pristupljeno u septembru, 2018)


### Kratka biografija:


**Milica Cicmil** rođena je u Nikšiću 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Inženjerstvo informacionih sistema odbranila je 2018.god.
kontakt: milica.cicmil9@gmail.com