Usporedba JavaScript biblioteka za rad s interaktivnim kartama

Perak, Matej

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:552538

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni stručni studij Elektrotehnike, smjer Informatika

Usporedba JavaScript biblioteka za rad s interaktivnim kartama

Završni rad

Matej Perak

Osijek, 2020. god.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 12.07.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime studenta:	Matej Perak
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4575, 27.09.2019.
OIB studenta:	88277526023
Mentor:	Izv. prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva 1:	Izv. prof. dr. sc. Marijan Herceg
Član Povjerenstva 2:	Izv. prof. dr. sc. Mario Vranješ
Naslov završnog rada:	Usporedba JavaScript biblioteka za rad s interaktivnim kartama
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Opisati i usporediti svojstva nekoliko Javascript biblioteka za rad s interaktivnim kartama (leaflet.js, openlayers.js itd.).
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Dobar (3)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 1 razina
Datum prijedloga ocjene mentora:	12.07.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 22.07.2020.

Ime i prezime studenta:	Matej Perak
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4575, 27.09.2019.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Usporedba JavaScript biblioteka za rad s interaktivnim kartama**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Marijan Herceg

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. Uvod 1
1.1. Zadatak Završnog rada 5
2. Pregled postojećih radova 6
3. Korišteni alati i biblioteke 7
3.1. Leaflet 8
3.2. OpenLayers 8
3.3. GoogleMaps 9
4. Usporedba algoritama 13
4.1. Algoritam za kreiranje karte 14
4.2. Algoritam zumiranja s dva prsta 15
4.3. Algoritam pronalaženja najkraće rute 16
4.4. Algoritam za prikaz na punom ekranu 18
4.5. Algoritam za najmanji mogući zum 19
4.6. Algoritam za toplinske karte 20
4.7. Algoritam za grupiranje 21
5. Zaključak 24
LITERATURA 25
SAŽETAK 27
ABSTRACT 28
ŽIVOTOPIS 29

1. Uvod

JavaScript je programski jezik koji se koristi za kreiranje i kontrolu sadržaja *web* stranice te bilo čega što se pomiče ili se na neki drugi način mijenja automatski. Također stvoren je kako bi „oživio“ *web* stranice [1].

Značajke *JavaScript* jezika su [2],[3]:

- Objektno orijentiran programski jezik
- Tehnologija klijenta (engl. *Client Edge* technology)
- Malo zauzeće memorije
- Rukovanje događajima (engl. *Event Handling*)
- Automatsko dovršavanje teksta
- Ostali dodaci kao što je interaktivno dugme (engl. *Button*)

JavaScript je razvio Brendan Eich (Slika 1.1.) 1995. godine, koji se pojavio u *Netscape-u*, popularnom pregledniku tog vremena [4].

Slika 1.1. Brendan Eich [5]

Jezik se u početku zvao *LiveScript*, a kasnije je preimenovan u *JavaScript*. Mnogo programera misle da su *JavaScript* i *Java* isto, no zapravo *JavaScript* i *Java* su u velikoj mjeri nepovezani i jako se razlikuju po dizajnu jezika. *Java* je vrlo složen programski jezik dok je *JavaScript* samo skriptni jezik. Na *JavaScript* je najviše utjecao programski jezik C.

CSS:

```
<style>
CSS goes here
</style>
```

JavaScript:

```
<script type="text/javascript">
JavaScript code goes here
</script>
```

Slika 1.2. Usporedba *JavaScript* i CSS-a [2]

Na slici 1.2. prikazan je način pisanja *JavaScript* koda koji se piše odvojeno kao i CSS (engl. *Cascading Style Sheets*), a oba se nalaze u HTML (engl. *HyperText Markup Language*) datoteci. Funkcionalnosti *JavaScript* jezika mogu se vidjeti kroz dva primjera kojima se korisnici interneta često susreću. Prvi primjer je kada se *Facebook* vremenska traka automatski ažurira na zaslonu ili kada tražilica *Google* predloži najsličnije pojmove za pretraživanje na temelju napisanih početnih slova [2]. Programi u ovom jeziku nazivaju se skripte te se mogu napisati u HTML-u *web* stranice i pokrenuti automatski kada se stranica učita. Skripte se izvode kao tekstualne datoteke i ne trebaju posebne pripreme ili prevođenja za pokretanje kao kod C ili C++ programskog jezika. U ovom aspektu *JavaScript* se jako razlikuje od drugog jezika zvanog *Java* [1].

```java
/* This is a simple Java program.
   FileName : "HelloWorld.java". */
class HelloWorld
{
    // Your program begins with a call to main().
    // Prints "Hello, World" to the terminal window.
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```

Slika 1.3. Primjer *Hello World* koda napisan u *Javi* [6]

```
<html>
<head>
    <title>My First JavaScript code!!!</title>
    <script type="text/javascript">
        alert("Hello World!");
    </script>
</head>
<body>
</body>
</html>
```

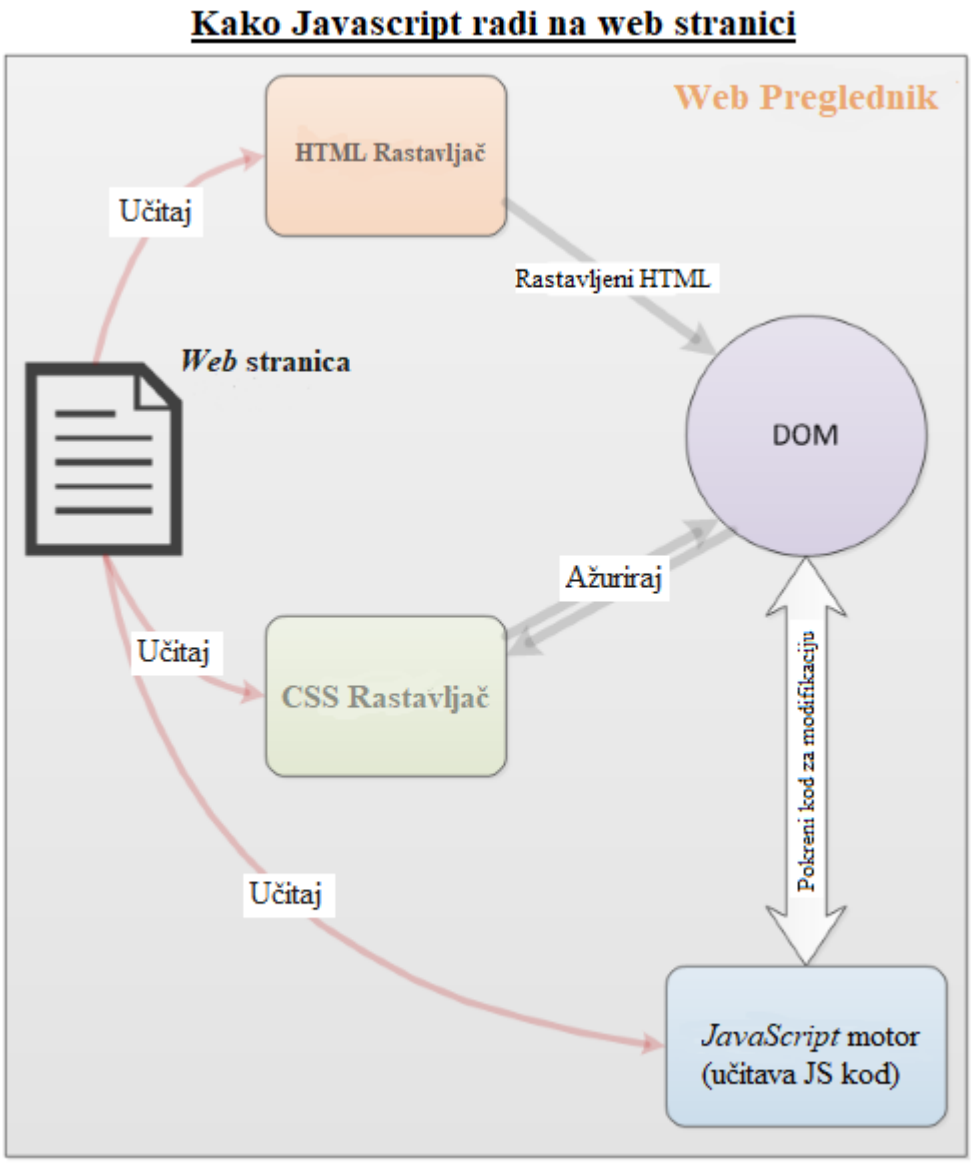
Slika 1.4. Primjer *Hello World* koda napisan u *JavaScript* jeziku [4]

Kako bi se bolje razumjela razliku između *Jave* i *JavaScript* programskog jezika na slici broj 1.3. i broj 1.4. prikazan je primjer ispisivanja *Hello World* primjera za oba jezika. *JavaScript* podržava većina modernih *web* preglednika kao što su *Google Chrome*, *Firefox*, *Safari*, *Microsoft Edge*, *Opera*, itd. Također, većina mobilnih preglednika za *Android* i *iOS* podržava *JavaScript* [7].

Tri su komponente od kojih se sastoje *web* stranice i aplikacije: HTML, CSS i *JavaScript*. Svaka od njih ima ulogu u stvaranju *web* aplikacije [7].

- HTML je jezik koji stvara kostur *web* stranice. Svi odlomci, odjeljci, slike, naslovi i tekst pisani su u HTML-u. Sadržaj se prikazuje na *web* mjestu redoslijedom kojim je napisan u HTML-u.
- CSS kontrolira stil i dodatne aspekte izgleda. CSS se koristi za kreiranje dizajna *web* stranice stvarajući boje, fontove, stupce, obrube, itd.
- Treći element je *JavaScript*. HTML i CSS stvaraju strukturu, ali od nje ne rade ništa, pa *JavaScript* stvara dinamičnu aktivnost na vašoj stranici. Pisanje *JavaScript-e* je ono što kontrolira funkcije kada se kliknu na gumbe, kako se autentificiraju lozinke, i ostale stvari.

Za bolje shvaćanje rada *JavaScript* unutar web stranice na slici 1.5. prikazan je princip rada web preglednik i DOM (engl. *Document Object Model*).



Slika 1.5. Veza *web* preglednika i DOM-a [7]

Web preglednik učitava *web* stranicu, rastavlja HTML i stvara sadržaj koji je poznat kao DOM. DOM prikazuje izgled *web* stranice u stvarnom vremenu. Preglednik će dohvatiti sve povezano s HTML-om kao što su slike i CSS datoteke. CSS informacije dolaze iz CSS *Parser-a*. HTML i CSS sastavlja DOM kako bi prvo stvorio *web* stranicu, potom preglednikov *JavaScript engine* učitava *JavaScript* datoteke nakon učitavanja HTML-a i CSS-a. Nakon toga *JavaScript* se izvršava redoslijedom kojim je napisan. Potom se DOM ažurira s *JavaScript* kodom i prikazuje se pomoću preglednika [7].

1.1. Zadatak Završnog rada

U ovom Završnom radu zadatak je usporediti *Leaflet, OpenLayers i GoogleMaps JavaScript* biblioteke po njihovoj brzini izvođenja te memorijskim otisku. Za svaki algoritam pronađeni su najbliži odnosno najsličniji algoritmi za navedene biblioteke. Algoritmi koji su bili uspoređivani su: Algoritam za kreiranje karte, algoritam zumiranja s dva prsta, algoritam za pronalaženje najkraće rute, algoritam za prikaz na punom ekranu, algoritam za najmanji mogući zum, algoritam za toplinske karte te algoritam za grupiranje.

2. Pregled postojećih radova

U radu [13] napravljena je usporedba API (engl. *Application programming interface*) nekoliko biblioteka s obzirom na njihovu upotrebljivost. API predstavlja skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava ili drugog složenog programa kao biblioteke. U radu su uspoređena tri različita API-ja: API za *GoogleMaps*, *ArcGIS* API za *JavaScript*, i biblioteka za mapiranje *OpenLayers JavaScript*. Usporedba je napravljena na dva načina: usporedba algoritama s istim funkcionalnostima i usporedba prema definicijama API-ja.

U radu [14] napravljena je usporedba najpopularnijih *JavaScript* biblioteka, a to su *jQuery, MooTools, Prototype i YUI*. Zatim je napravljena usporedba tih biblioteka u smislu kompatibilnosti preglednika, sudjelovanja korisničke zajednice, performanse te značajki i podrške za buduće *web* tehnologije kao što su HTML5 i CSS3. U radu je pokazano da *jQuery* ima najbolju podršku za novije verzije preglednika. *jQuery* je bio najbolji i po sudjelovanju korisničke zajednice, a što se tiče performanse *MooTools* je bio najbolji gdje je veličina biblioteka bila 7.4KB.

U radu [15] napravljena je usporedba *JavaScript* biblioteka, kao što su *Yandex.Maps*, *GoogleMaps*, *OpenLayers* te *Leaflet*. U uvodu rada svaka od navedenih biblioteka ukratko je opisana te je prikazan pregled osnovnih funkcija i prednosti koje pružaju. U radu posebna pozornost se posvećuje ugrađenim alatima kao što su: skaliranje karte, dodavanje novih objekata na kartu, grupiranje slojeva karte, itd.

3. Korišteni alati i biblioteke

Alati koji su bili potrebni za realizaciju ovog rada je *text editor* za pisanje kodova i preglednik za prikaz *web* stranica koje napravite. U ovom radu korišten je *Visual Studio Code* i *Google Chrome* (verzija 83.0.4103.116), a neki od najpoznatijih *text editora* su *Notepad++, Visual Studio Code, Sublime Text, Atom*, i ostali, a od preglednika najpoznatiji su *Google Chrome, Firefox, Microsoft Edge, Internet Explorer, Tor...* [4].

Općenito, *JavaScript* biblioteke su zbirke već napisanih kodova koji se mogu koristiti za obavljanje uobičajenih *JavaScript* funkcija. *JavaScript* biblioteke mogu se naknadno priključiti u ostatak projekta, ovisno o potrebi. Ako postoji potreba za recimo dodavanje automatskog dovršavanja riječi, samo dodamo odgovarajući isječak *jQuery* [17] koda u projekt. Zatim, kada korisnik unese tekst u traku za pretraživanje, isječak koda *jQuery* dohvaća funkcionalnost iz *jQuery* biblioteke i prikazuje u *web* pregledniku. Neki primjeri biblioteka su: *jQuery, React JS*[18], *Leaflet.js* [19], *OpenLayers* [20], te poznati *GoogleMaps JS* [21].

Iako su *JavaScript* biblioteke specijalizirani alat za korištenje po potrebi, *JavaScript Framwork* su potpuni skup alata koji pomažu u oblikovanju i organiziranju *web* stranice ili *web* aplikacije. Za lakše razumijevanje na *JavaScript* biblioteke se može gledati kao na komade namještaja koji dodaju stil i funkcionalnost već izgrađenoj kući, dok je *Framework* predložak koji koristite za izgradnju same kuće. Primjeri *Frameworka* su: *Angular*[22], *Ember JS*[23], *Vue*[24], i mnogi ostali. Prednost korištenja *JavaScript Frameworka* su sveukupna učinkovitost i organizacija koju donose u projekt – projekt će biti uredno strukturiran, no svaki kod mora slijediti pravila specifična za taj *Framework,* ograničavajući slobodu koju imate kada koristite biblioteke.

Većina aplikacija i *web* stranica koje se upotrebljavaju na svakodnevnoj bazi koristi lokaciju i razne podatke na kartama. Kao primjer su ugostiteljski objekti koji koriste karte za dostavu hrane preko internetskih trgovina i mobilnih aplikacija kako bi prilikom dostave lakše našli kupca, a da ne troše puno vremena na pronalaženje lokacije. Tu se javlja pitanje koje alate koristiti za rad sa kartama, a odgovor je jednostavan ukoliko znate što tražite. Ovisno da li se traže posebne aplikacije za neka određena natjecanja, istraživanja, putovanja ili planinarenja, bit će potrebna rješenja otvorenog koda kao što je *Leaflet* [19] *i OpenLayers* [20], no ako projekt ne zahtjeva previše mogućnosti prilagođavanja tu je pouzdani *GoogleMaps* sa osnovnim stvarima bez puno mogućnosti uređivanja [8].

3.1. Leaflet

Slika 3.1. Prikaz *Leaflet* karte

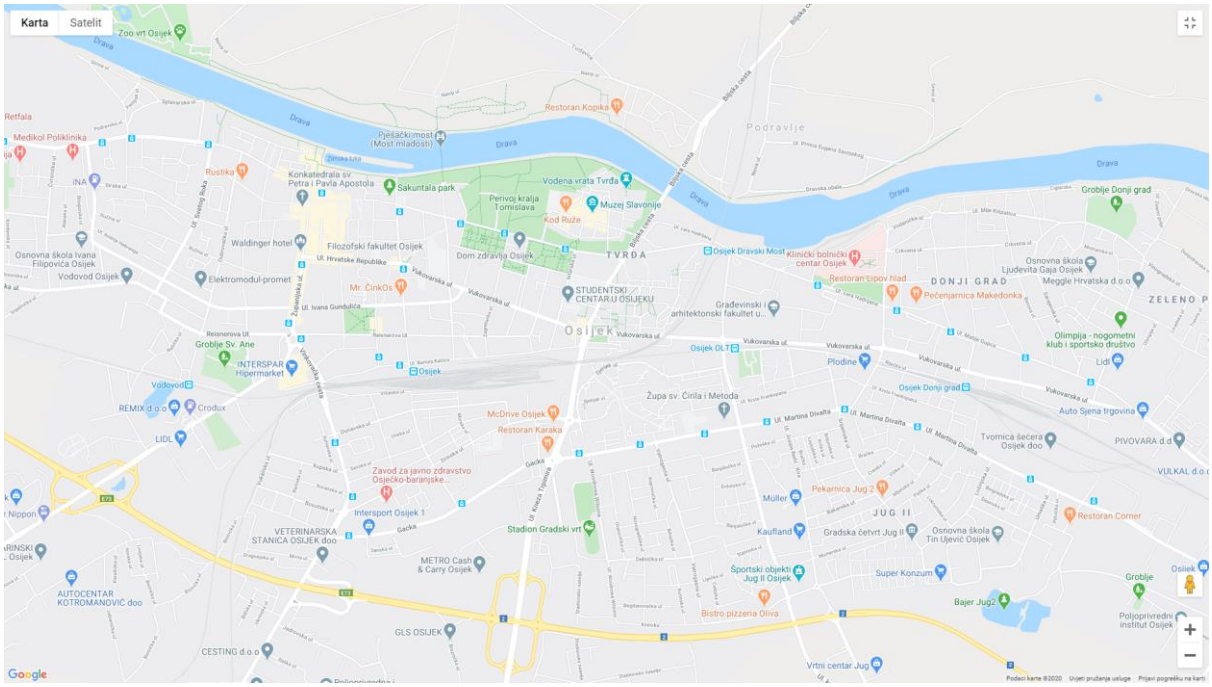
Leaflet je vodeća *open-source JavaScript* biblioteka za interaktivne karte prilagođene mobilnim uređajima, a izgled karte prikazan je na slici 3.1. Teži samo oko 38KB, a ima sve značajke za rad sa kartama koje će većini programera biti dovoljne, a ako postoji potreba za dodatnim stvarima tu su dodaci odnosno dodataka (engl. *plugins*). Može se koristiti s *CartoDB, MapBox, Google* kartama i s još mnogo platformi koji pružaju mapiranje [9]. *Leaflet* je dizajniran za jednostavnu uporabu, brzinu i kvalitetu. Jednostavno i učinkovito djeluje na gotovo svim *desktop* i mobilnim platformama, također postoji mogućnost proširenja s puno dodataka, jednostavan i dobro dokumentiran API za početnike. *Leaflet-ov* nedostatak je što za kompleksnije zadatke treba dodatnog istraživanja po internetu i forumima [16].

3.2. OpenLayers

Slika 3.2. Izgled *OpenLayers* karte [10]

OpenLayers je biblioteka s visokim performansama i značajkama za mapiranje, a njen izgled prikazan je na slici 3.2. Stvoren je za upotrebu geografskih informacija svakakvih vrsta. Također je *open-source*, objavljen pod *BSD* licencom (*FreeBDS*). Pruža API za izradu bogatih *web*-geografskih aplikacija sličnih *Google* i *Bing* kartama, također velik plus *OpenLayers-a* je taj da se može integrirati u bilo koju *closed* ili *open source* aplikaciju (BSD licenca). Lošija strana *OpenLayers-a* je ta što ne pružaju potpuna objašnjenja za početnike te za pojedine stvari korisnici moraju koristiti *Google* i ostale načine kako bi se snašli. Također, korisnici su imali jako puno grešaka koristeći *OpenLayers* jer „jednostavno nije radio“ koristeći verzije *OpenLayers 2 i 3*.

3.3. GoogleMaps



Slika 3.3. Izgled *GoogleMaps* karte

GoogleMaps jedna je od *JavaScript* biblioteka korištena za izgradnju prilagođenih karti koje donose stvarni svijet pomoću statičnih i dinamičkih karata, slika te 360° prikaza koju pruža *Google*. Do 360° prikaza se može doći pritiskom na žutog čovječuljka u donjem desnom kutu (slika 3.3.). *GoogleMaps*-om su pružene detaljne informacije o geografskim regijama, *web* lokacijama, ulicama, izračunu puta za putovanja pješke, automobilom te zračnim putem kao i satelitskim prikazima mnogih mjesta u brojnim zemljama svijeta. *GoogleMaps* API daje slobodu za izradu prilagođenih karata koje mogu imati različite funkcije [11]. Glavni nedostatak *GoogleMaps-a* je ograničenost, a pod tim se misli na 200$ besplatnog kredita mjesečno s kojima

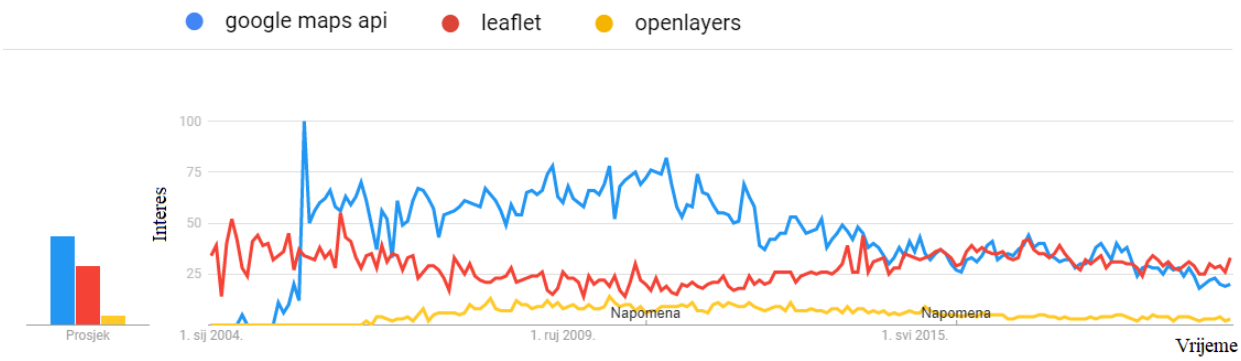
možete raditi s njihovim kartama. Ukoliko se potroši 200$ kredita, automatski se kreće naplaćivati po njihovim cijenama.

U tablici 3.1. prikazana je usporedba biblioteka po cijeni, memorijskoj veličini, po mogućnosti satelitskog pogleda, pogleda s ulice te po broju dodataka (*plugins*) koji se mogu koristiti u pojedinim bibliotekama. Što se tiće sateliskog pogleda i pogleda s ulice, *Leaflet i OpenLayers* nemaju svoj satelitski pogled kao ni pogled s ulice, već ih mogu implemetirati od strane *GoogleMapsa, NASA-e, Yandex-a* i ostalih davatelja usluga.

Tablica 3.1. Usporedba biblioteka po osnovnim parametrima [19][20][21][25]

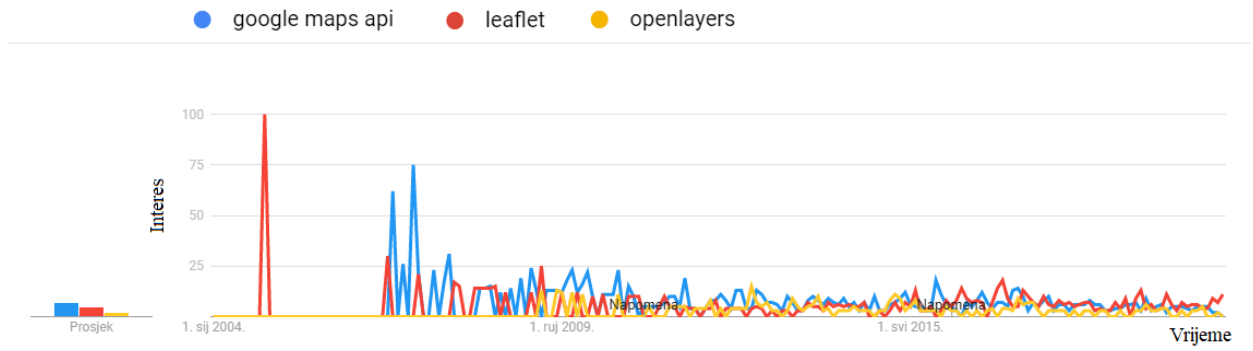
	Cijena:	**Veličina:**	**Satelitski pogled:**	**Pogledi s ulice:**	**Dodaci (*plugins*):**
Leaflet	*open-source*	38KB	Da – *GoogleMaps, NASA, ...*	Da – *GoogleMaps, Yandex, ...*	Puno
OpenLayers	*open-source*	391KB	Da – *GoogleMaps, Bing, ...*	Da – *GoogleMaps, ...*	Malo
GoogleMaps	Besplatan do 200$ *Google* kredita po mjesecu	Nemoguće izmjeriti	Da	Da	Srednje

Na slici 3.4. su prikazane američke analize najčešće korištenih biblioteka, a od ove tri je bila *Google Maps* (gledajući od 2004. do danas), no danas se sve više ljudi okreću *Leaflet* biblioteci, te *OpenLayers* kao zadnjoj korištenoj.



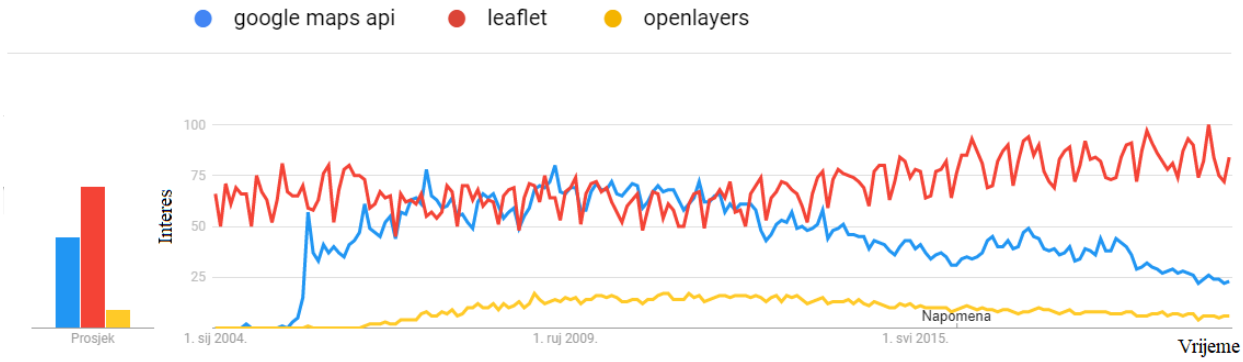
Slika 3.4. Interes korisnika za pojedine biblioteke u Americi od 2004 godine do danas [12]

U hrvatskoj su rezultati puno drugačiji, gdje je zainteresiranost blizu nuli što možemo vidjeti na slici broj 3.5., a gotovo svi zainteresirani su iz grada Zagreba.



Slika 3.5. Interes korisnika za pojedine biblioteke u Hrvatskoj od 2004 godine do danas [12]

U svijetu je potpuno drugačije, gdje se sve više ljudi okreće *Leaflet* biblioteci (slika 3.6.), s najvećim postotkom u Indiji, Australiji, Južnoj Africi te Saudijskoj Arabiji i Egiptu. Dok *GoogleMaps* koriste u velikoj količini u ostatku svijeta kao što su Sjeverna i Južna Amerika te Rusija, Europa i Skandinavske zemlje (slika 3.7.).



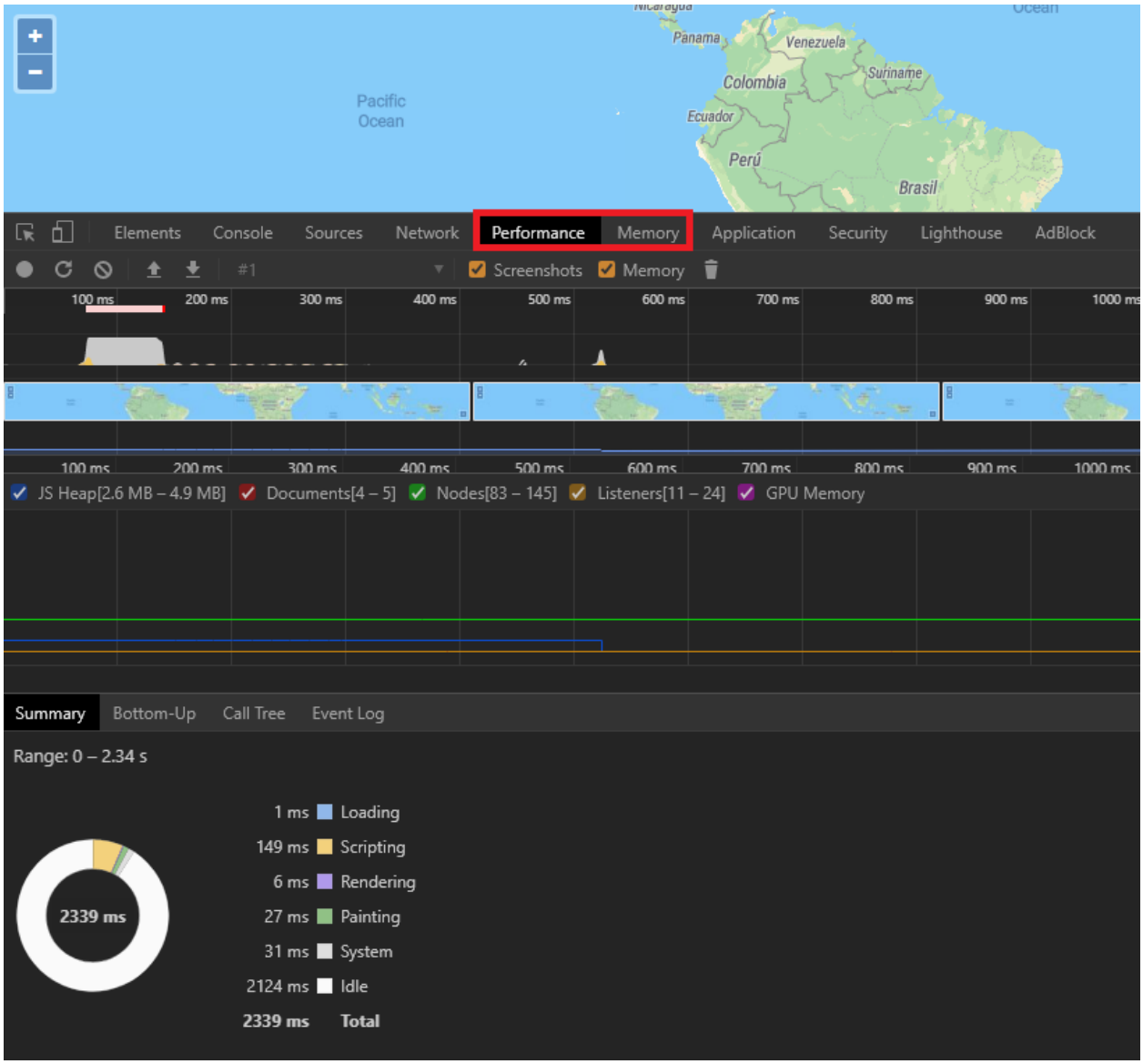
Slika 3.6. Interes ljudi za pojedine biblioteke u svijetu od 2004 godine do danas [12]

Slika 3.7. Usporedna raščlamba po podregiji u svijetu [12]

Prema analizama i grafovima biblioteke se razlikuju na puno načina, najbolja biblioteka je *GoogleMaps* po funkcionalnosti no i *Leaflet* kojoj interes raste s vremenom kao i njena funkcionalnost. Vrijednosti za svaku točku na grafu temelje se na djelomičnim odnosno nepotpunim podacima [12].

4. Usporedba algoritama

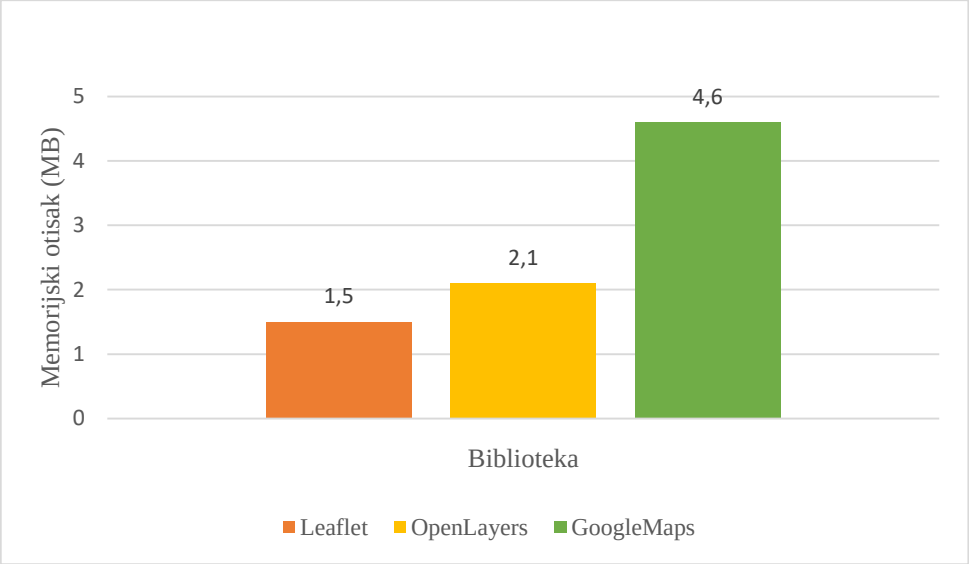
U sljedećem odlomku napravljena je usporedba navedenih biblioteka s obzirom na njihovu brzinu izvođenja i njihov memorijski otisak, a za svaki algoritam pronađen isti ili najsličniji algoritam po funkcionalnosti kako bi usporedba bila što preciznija i točnija. Performanse su mjerene preko *Google Chrome* preglednika preko opcije provjeri (engl. *Inspect*) koja ima tabove *Performance* i *Memory* koje su korištene u ovom radu (Slika 4.1). Algoritmi koji su uspoređivani su: algoritam za kreiranje karte, algoritam zumiranja s dva prsta, algoritam za pronalaženje najkraće rute, algoritam za prikaz na punom ekranu, algoritam za najmanji mogući zum, algoritam za toplinske karte te algoritam za grupiranje.



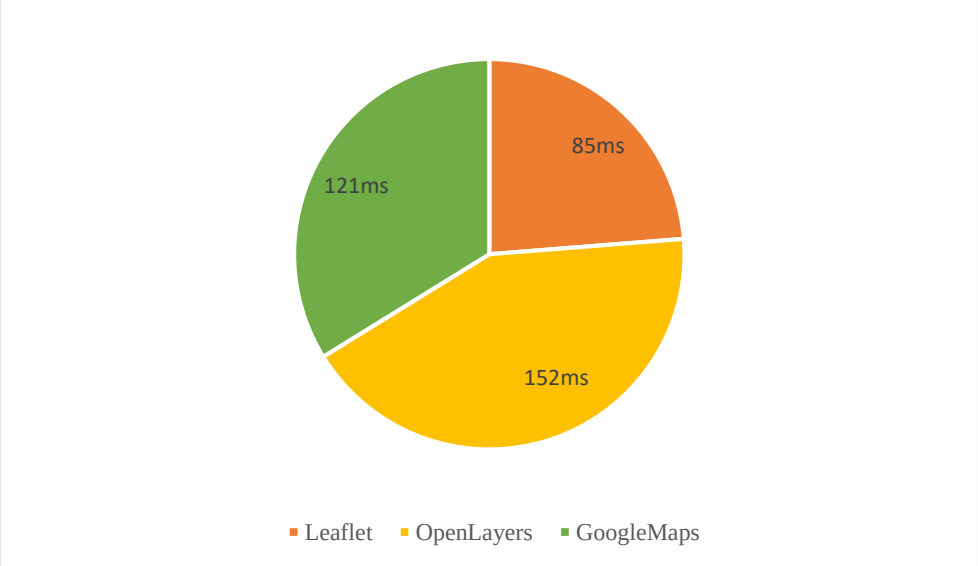
Slika 4.1. Prikaz *inspect element* opcije na *Google Chrome* pregledniku

4.1. Algoritam za kreiranje karte

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za kreiranje karte. Ovaj algoritam služi kako bi se stvorila karta na kojoj bi kasnije bile dodane ostale značajke i funkcionalnosti, dakle algoritam služi samo za kreiranje karte.



Graf 4.1. Memorijski otisak algoritma za kreiranje karte



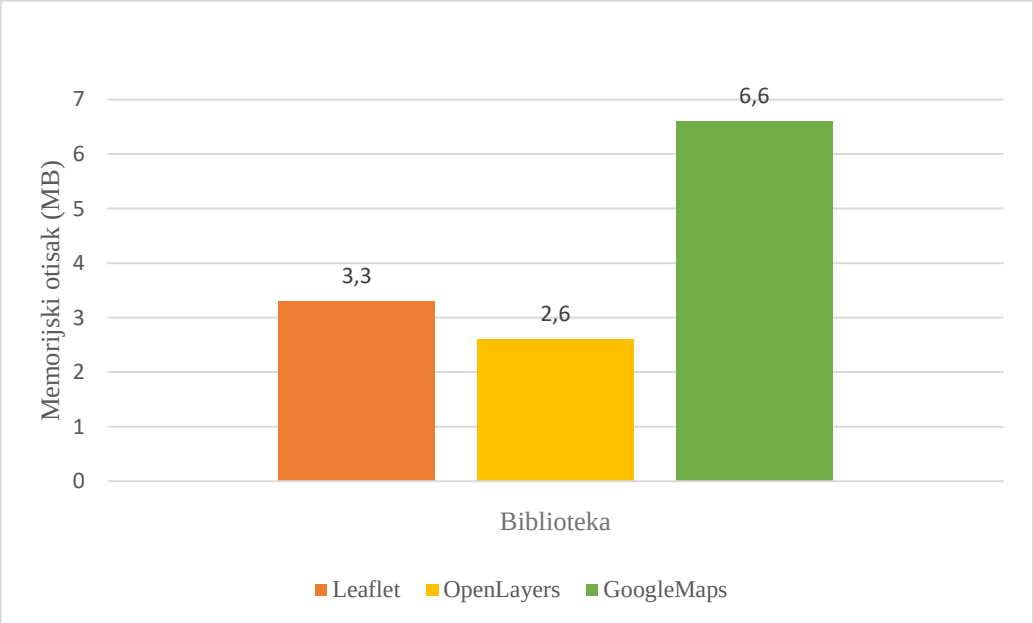
Graf 4.2. Brzina izvođenja algoritma za kreiranje karte

Graf 4.1. prikazuje memorijski otisak algoritma za kreiranje karte svih triju biblioteka. *Leaflet* algoritam zauzima 1.5 MB dok *OpenLayers* zauzima 2.1 MB, a *GoogleMaps* zauzima 4.6

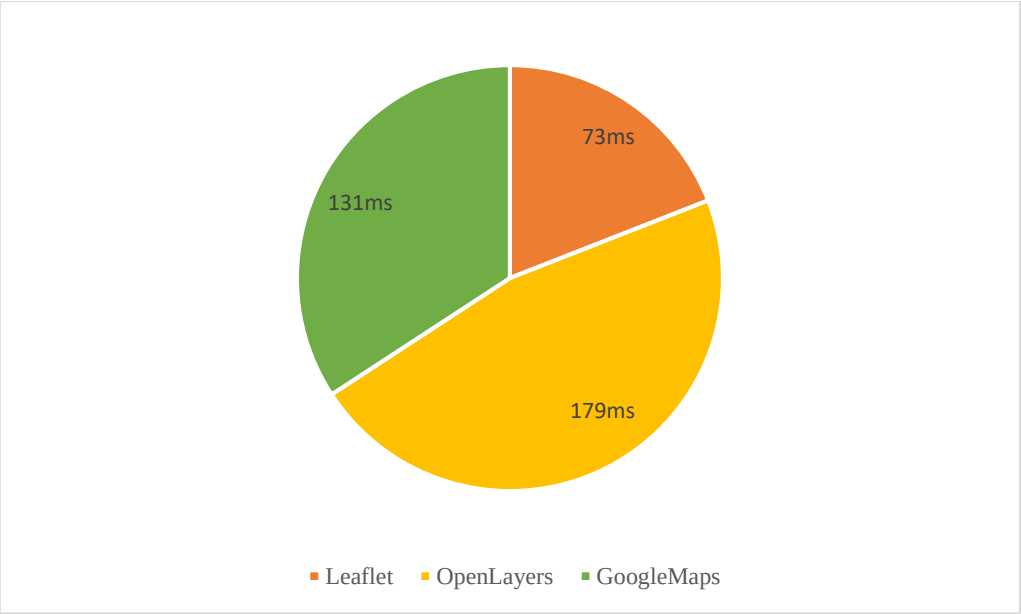
MB. Razlika je u tome što je kod *Leafleta* i *OpenLayersa* samo implementacija karte, dok kod *GoogleMapsa* dobiva i niz drugih mogućnosti kao što je satelitski pogled, pogled terena te je zbog toga malo sporiji od ostalih biblioteka. Što se tiče brzine izvođenja algoritma, ukupno vrijeme potrebno da se algoritam izvrši kod *Leaflet* biblioteke iznosi 85 ms (milisekundi), dok je kod *OpenLayersa* ukupno potrebno 152 ms te kod *GoogleMapsa* 121 ms (Graf 4.2.). *Leaflet* u ovom slučaju ima prednost jer je potrebno 55 ms za proces *scripting*, dok je *OpenLayersu* potrebno 86 ms te *GoogleMapsu* 62 ms. Proces *scripting* je proces kod kojeg program izvršava skriptu odnosno algoritam, što znači da poziva ostale funkcije, *evente* i ostale stvari potrebne za izvršavanje. Za ovaj algoritam *Leaflet* ima najbolje performanse, dok je drugi najoptimiziraniji kod od *GoogleMapsa* te *OpenLayers* kao posljednji najoptimiziraniji.

4.2. Algoritam zumiranja s dva prsta

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma zumiranja s dva prsta. Ovaj algoritam se koristi na mobilnim platformama te služi kako bi korisnici mogli lakše zumirati kartu. Algoritam je testiran na mobilnom uređaju preko *Mozilla Firefox* preglednika.



Graf 4.3. Memorijski otisak algoritma zumiranja s dva prsta

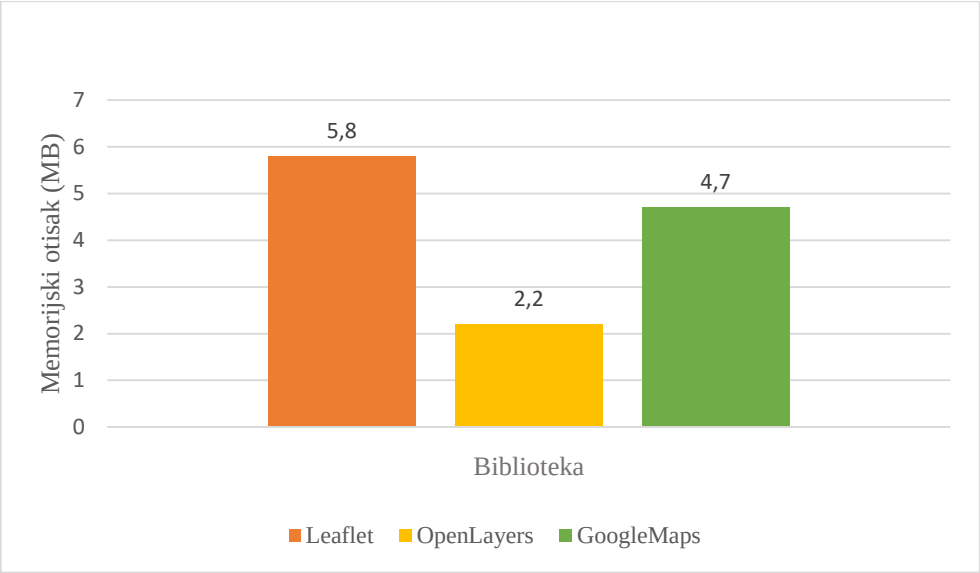


Graf 4.4. Brzina izvođenja algoritma zumiranja s dva prsta

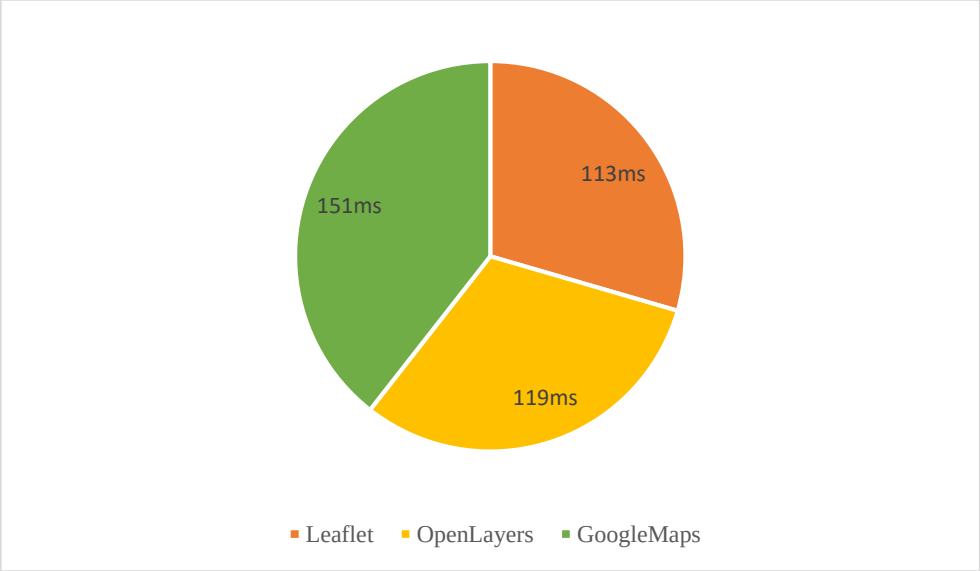
Graf 4.3. prikazuje memorijski otisak algoritma za zumiranje s dva prsta. *Leaflet* algoritam zauzima 3.3 MB dok kod *OpenLayersa* zauzima 2.6 MB, a kod *GoogleMapsa* zauzima 6.6 MB. *OpenLayers* se pokazao najbolji po usporedbi po memorijskom otisku, no *GoogleMaps* i dalje ostaje najbolji jer se dobiva niz drugih mogućnosti kao što su satelistki pogled i *street view*. Što se tiče brzine izvođenja algoritma (Graf 4.4.) *Leaflet* i *OpenLayers* biblioteci bilo je potrebno učitavanje koje iznosi 2 ms za obje biblioteke. Najveća razlika zamijećena je kod *scripting* dijela, kod *OpenLayersa* bilo je potrebno čak 129 ms, na drugom mjestu nalazi se *GoogleMaps* sa 77 ms, te najoptimiziraniji i dalje ostaje *Leaflet* sa 39 ms.

4.3. Algoritam pronalaženja najkraće rute

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za pronalaženje najkraće rute. Ovaj algoritam služi kako bi se korisnicima olakšalo biranje najkraće putanje do željenog mjesta. Glavna razlika između ove tri biblioteke je kod *Leaflet-a* koji opravdava svoj memorijski otisak tako što se kod njega mogu markeri pomicati u stvarnom vremenu ako želimo drugom rutom poći do istog mjesta te algoritam automatski nađe najkraći put preko željenog mjesta.



Graf 4.5. Memorijski otisak algoritma za pronalaženje najkraće rute

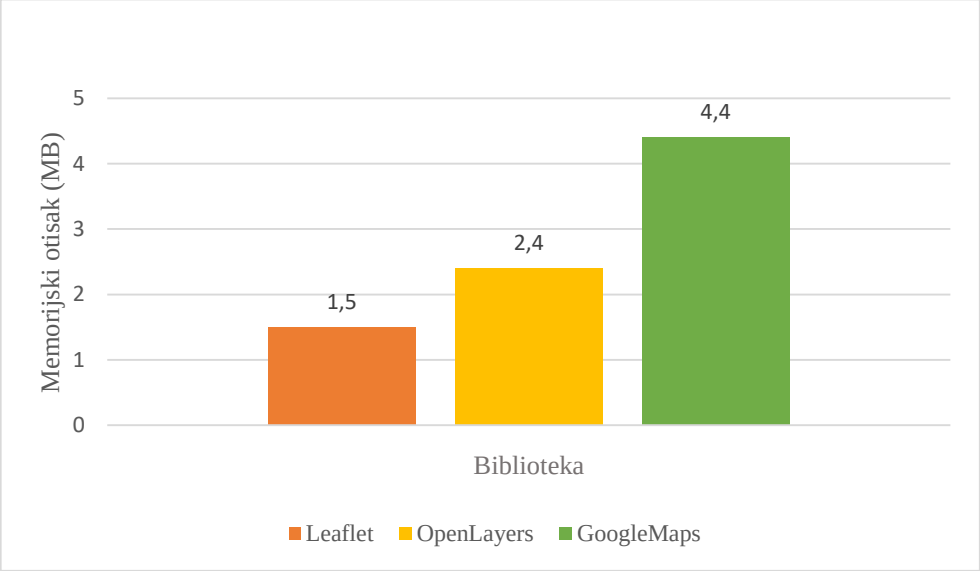


Graf 4.6. Brzina izvođenja algoritma za pronalaženje najkraće rute

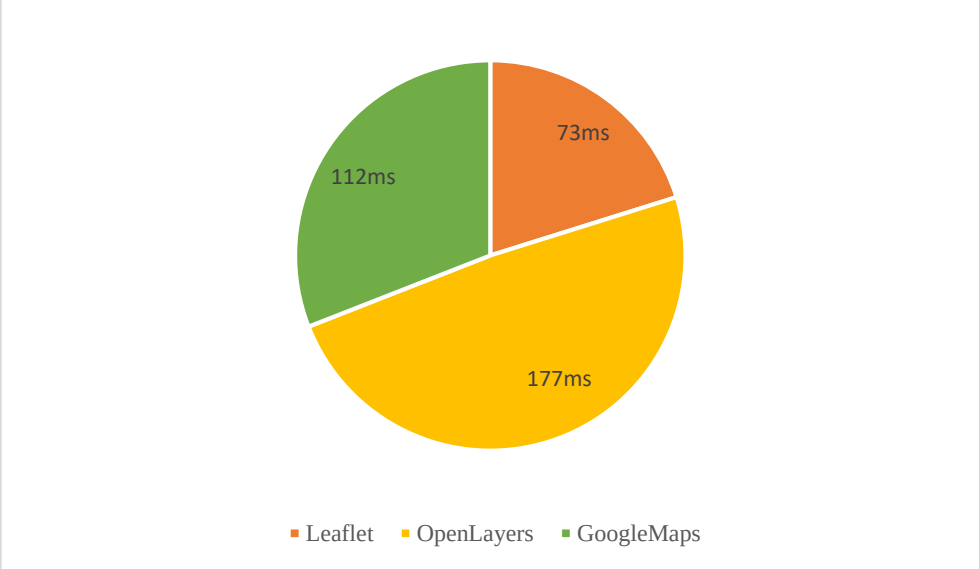
Po grafu 4.5. vidimo da *Leaflet-ov* algoritam u ovom slučaju zauzima najviše memorije. Dok je kod ostalih biblioteka običan *routing* sistem gdje u određena polja upisujemo ime gradova, mjesta, itd.. Grafom 4.6. vidimo brzinu izvođenja navedenog algoritma. *GoogleMaps-u* je potrebno 151 ms, *OpenLayers-u* 119 ms, te *Leaflet-u* 113 ms. U procesu *scriptinga* sve tri biblioteke imale su približno vrijeme izvođenja, *Leaflet* sa 56 ms, *OpenLayers* sa 57 ms te *GoogleMaps* sa 55 ms, a kod *renderinga* se pojavila najveća razlika, *Leaflet* sa 28 ms *OpenLayers* sa 30 ms te *GoogleMaps* sa 54 ms.

4.4. Algoritam za prikaz na punom ekranu

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za prikaz na punom ekranu. Ovaj algoritam služi kako bi se karta povećala preko cijelog zaslona.



Graf 4.7. Memorijski otisak algoritma za prikaz na punom ekranu



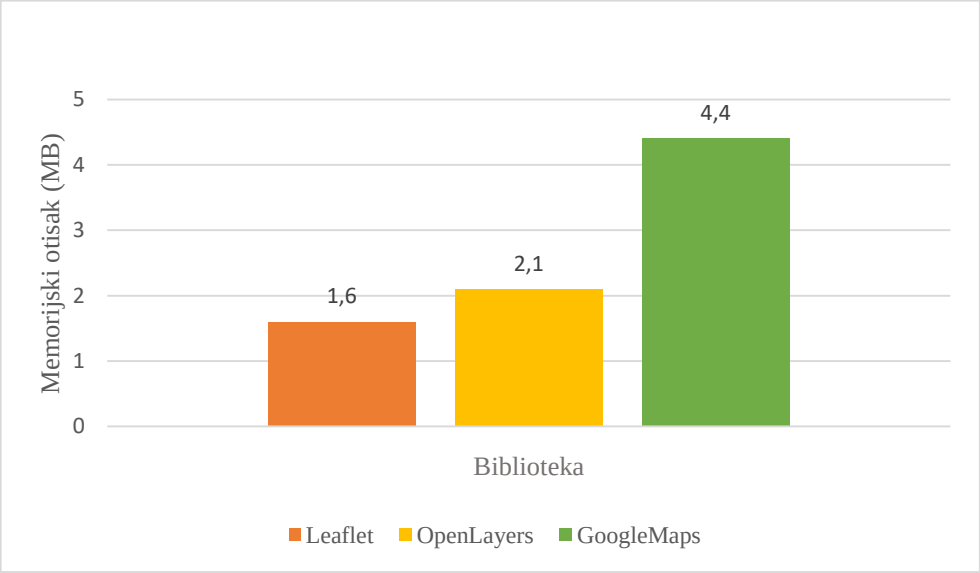
Graf 4.8. Brzina izvođenja algoritma za prikaz na punom ekranu

U algoritmu za prikaz na punom ekranu, *Leaflet* zauzima 1,5 MB, *OpenLayers* 2,4 MB te *GoogleMaps* sa 4,4 MB (Graf 4.7.). U usporedbi s algoritmom za kreiranje karte, *Leaflet* zauzima isto za oba algoritma, *OpenLayers* algoritam za kreiranje mape zauzimao je 2,1 MB, što stvara razliku od 0,3 MB, a *GoogleMaps* zauzima 4,4 što je 0,2 MB manje od svog algoritma za kreiranje karte. Brzina izvođenja algoritma prikazana je na grafu 4.8. U ovom slučaju

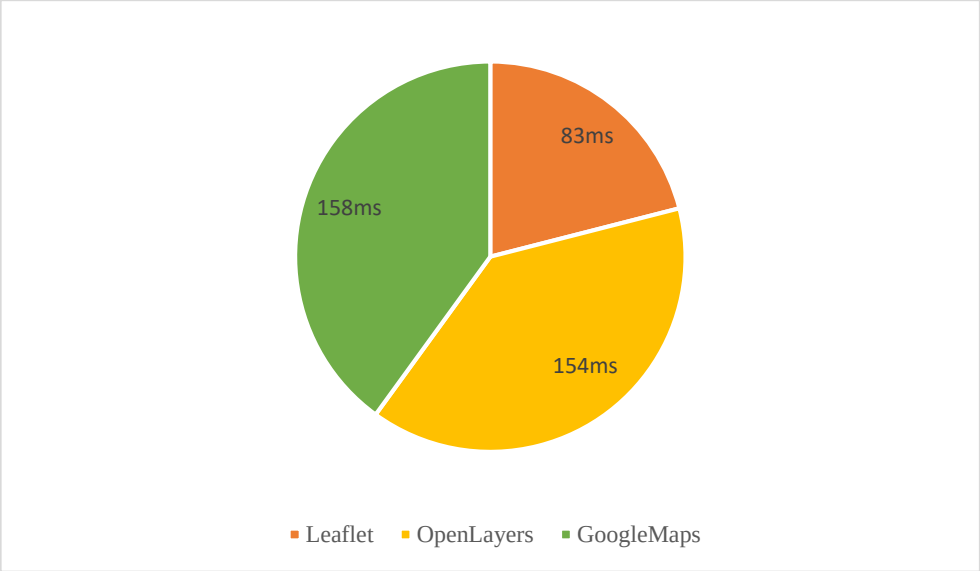
najoptimiziraniji kod je kod *Leaflet-a* za čije je izvršavanje trebalo samo 73 ms, drugi po redu je *GoogleMaps* sa 112 ms, te treći i najlošiji iznosi 177 ms, što je više od 2 puta nego *Leaflet-ov* algoritam.

4.5. Algoritam za najmanji mogući zum

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za najmanji mogući zum. Ovaj algoritam služi kako bi ograničili minimalno zumiranje karte, također postoji i algoritam za najveći mogući zum.



Graf 4.9. Memorijski otisak algoritma za najmanji mogući zum

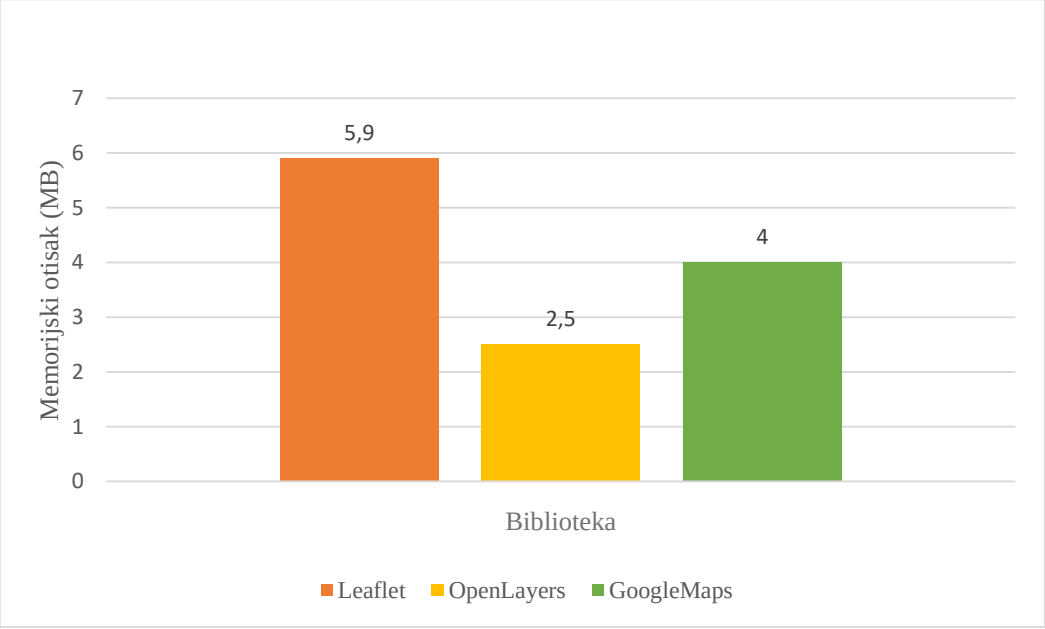


Graf 4.10. Brzina izvođenja algoritma za najmanji mogući zum

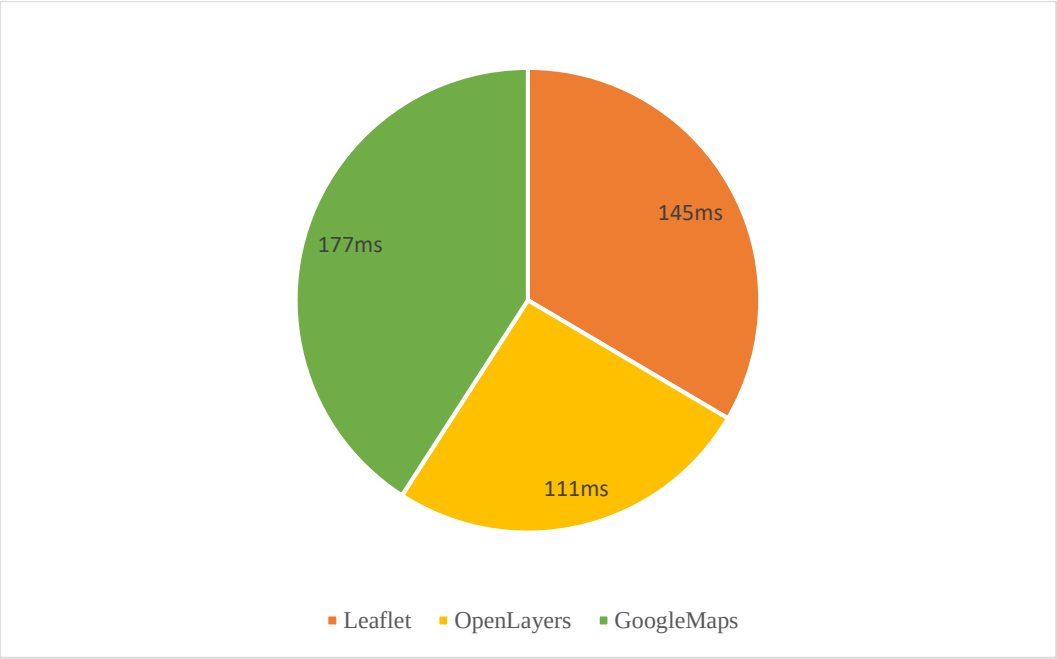
Algoritam za najmanji mogući zum kod *Leaflet-a* zauzima 1,6 MB, kod *OpenLayers-a* 2,1 MB te kod *GoogleMaps-a* 4,4 MB (Graf 4.9.). Ovaj algoritam definira najmanji *zoom level* koji se može postići prilikom zumiranja. Također postoji i opcija *max zoom* koja definira maksimalno zumiranje no nju ne uspoređujemo u ovom algoritmu. Najkraća brzina izvođenja algoritma je u slučaju *Leaflet-a* sa 83 ms, dok su *OpenLayers* i *GoogleMaps* podjednaki, *OpenLayers* sa 154 ms te *GoogleMaps* sa 158 ms, a prikazani su na grafu 4.10. U ovom slučaju najoptimiziraniji kod ostaje *Leaflet.*

4.6. Algoritam za toplinske karte

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za toplinske karte. Ovaj algoritam služi kako bi vizualno predočili infekcije u svijetu, kako bi označili žarišta potresa kao i vizualno predočavanje topline na određenim mjestima, gustoću ljudi na nekom mjestu, te niz drugih stvari. U ovoj usporedbi primjer je bio označavanje žarišta potresa. Algoritam za toplinske karte može služiti i za prikazivanje gužvi na autocestama, u gradovima ili za prikazivanje širenja virusa.



Graf 4.11. Memorijski otisak algoritma za toplinske karte

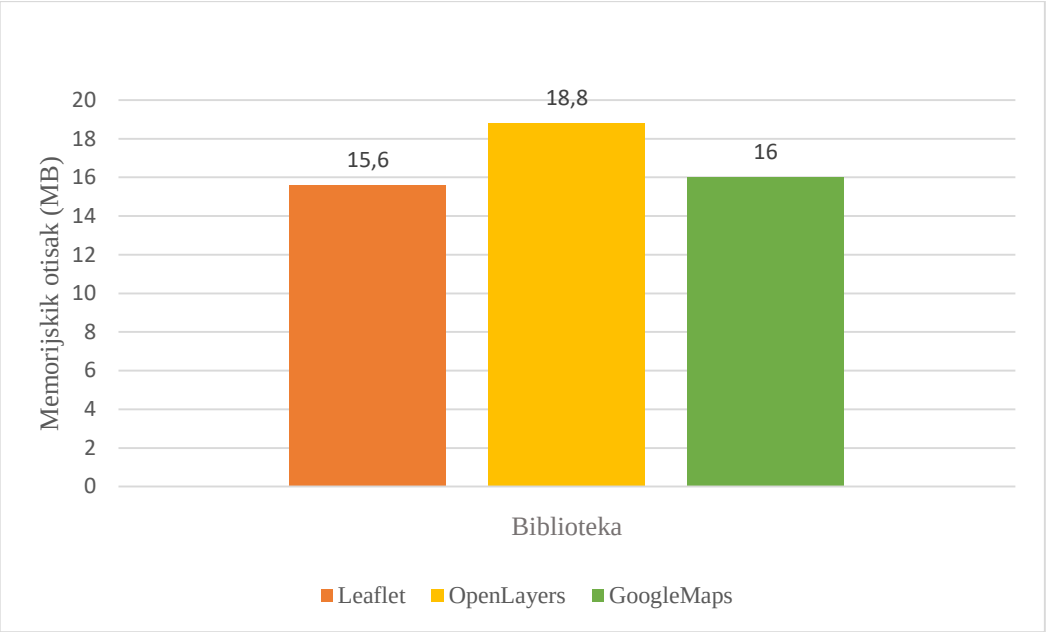


Graf 4.12. Brzina izvođenja algoritma za toplinske karte

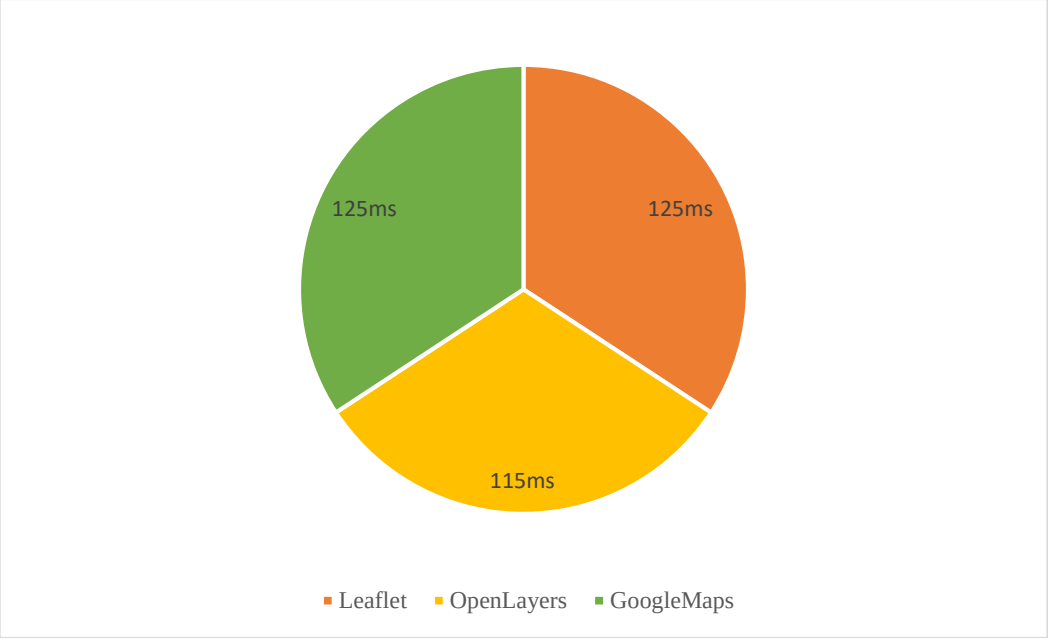
Leaflet-ov i *GoogleMaps-ov* algoritam je potpuno običan algoritam gdje crvena prikazuje jezgru žarišta dok kod *OpenLayers-a* postoje i mogućnosti za povećavanje radijusa žarišta kao i mogućnost zamagljivanja (eng. *Blur*) u stvarnom vremenu. Prema memorijskom otisku vidimo na grafu 4.11. da *Leaflet-ov* zauzima najviše sa 5,9 MB, *OpenLayers-ov* sa 2,5 MB te *GoogleMaps-ov* sa 4 MB. No u brzini izvođenja algoritma koji je prikazan na grafu 4.12., *GoogleMaps* prestiže *Leaflet* za 32 ms dok je *OpenLayers-u* potrebno 34 ms manje nego *Leaflet-u*. Algoritam za toplinske karte često je korišten i u prognozi vremena gdje se prikazuju boje od plave koja predstavlja hladnoću do crvene koja predstavlja vrućinu.

4.7. Algoritam za grupiranje

U ovom potpoglavlju uspoređena je brzina izvođenja i memorijski otisak algoritma za grupiranje. Ovaj algoritam služi kako bi skupinu oznaka ili markera prilikom zumiranja raširili odnosno skupili sve u jednu oznaku ili marker. Algoritam za grupiranje radi tako što skupinu markera koji su u blizini prikazuje kao jedan marker sa sveukupnim brojem bližnjih markera.



Graf 4.13. Memorijski otisak algoritma za grupiranje



Graf 4.14. Brzina izvođenja algoritma za grupiranje

Za *Leaflet* biblioteku memorijski otisak je 15,6 MB, kod *OpenLayers-a* je 128,8 MB te kod *GoogleMaps-a* je 16 MB (graf 4.13.). Njihova brzina izvođenja je slična gdje *Leaflet* i *GoogleMaps* imaju jednaki rezultat sa 125 ms, dok *OpenLayers* koji zauzima najviše memorije se najbrže izvršava sa 115 ms, a prikazani su na grafu 4.14.

Tablica 4.1. Memorijski otisak biblioteka ovisno o algoritmima

	Kreiranje karte	**Zum s dva prsta**	**Pronalaženje najkraće rute**	**Prikaz na punom ekranu**	**Najmanji mogući zum**	**Toplinske Karte**	***Cluster***
Leaflet	1.	2.	3.	1.	1.	3.	1.
OpenLayers	2.	1.	1.	2.	2.	1.	3.
GoogleMaps	3.	3.	2.	3.	3.	2.	2.

Tablica 4.2. Brzina izvođenja biblioteka ovisno o algoritmima

	Kreiranje karte	**Zum s dva prsta**	**Pronalaženje najkraće rute**	**Prikaz na punom ekranu**	**Najmanji mogući zum**	**Toplinske Karte**	***Cluster***
Leaflet	1.	1.	1.	1.	1.	2.	2.
OpenLayers	3.	3.	2.	3.	2.	1.	1.
GoogleMaps	2.	2.	3.	2.	3.	3.	2.

U tablicama 4.1. i 4.2. prikazani su sumirani rezultati usporedbe biblioteka, iz kojih je vidljivo da je *Leaflet* u 4 od sveukupno 7 algoritama zauzimao najmanje memorije, drugi po redu je *OpenLayers* koji je u 3 algoritma uspio imati najmanje memorijskog otiska, te *GoogleMaps* kao posljednji. Što se tiče brzine izvođenja algoritma, *Leaflet* je također najbrže izvodio algoritme gdje je uspio u 5 od 7 algoritama najbrže izvesti, što znači da je najefikasniji, drugi je *OpenLayers* te iza njega *GoogleMaps* kao posljednji.

Kako bi ispravno odabrali biblioteku potrebno je razmisliti za što bismo ju koristili i u kojim količinama. *GoogleMaps* je odličan odabir ako ne prelazite 200$ *Google* kredita te ako je potreban za male projekte bez puno dodataka, no sa besplatne *open-source* strane tu je uvijek *Leaflet* koji zbog velike korisničke zajednice i jako puno *plugin*-a fleksibilan te ostaje kao najbolji izbor trenutno. *Leaflet* također sadrži sve značajke koje programerima treba za izradu karte, a kako bi to upotpunili koriste se *Leaflet plugins*-i [16]. Također, *Leaflet* biblioteka se unapređuje iz minute u minutu jer njegovi developeri i korisnici stalno poboljšavaju biblioteku.

5. Zaključak

Kako se danas tehnologija razvija brzo, i kako su nam aplikacije i karte potrebne iz dana u dan, potrebno ih je i dodatno razvijati i usavršavati. U ovom radu napravljena je usporedba *JavaScript* biblioteka za rad sa interaktivnim kartama.

Uspoređivane su biblioteke *Leaflet, OpenLayers i GoogleMaps* i one se razlikuju po svojoj funkcionalnosti, fleksibilnosti i brzini. Algoritmi koji su bili uspoređivani su: Algoritam za kreiranje karte, algoritam zumiranja s dva prsta, algoritam za pronalaženje najkraće rute, algoritam za prikaz na punom ekranu, algoritam za najmanji mogući zum, algoritam za toplinske karte te algoritam za grupiranje. Algoritmi su uspoređivani po njihovoj brzini izvođenja te memorijskom otisku. Za svaku biblioteku pronađen je isti ili najsličniji algoritam kako bi usporedba bila valjana. *Leaflet* je u 4 od 7 algoritama imao najbolje rezultate što se tiče memorijskog otiska, drugi po redu bio je *OpenLayers* koji je u ostala 3 algoritma bio najbolji po memorijskom otisku, a ostaje *GoogleMaps* kao zadnji po usporedbi po memorijskom otisku. Što se tiče brzine izvođenja algoritma *Leaflet* je u 5 od 7 algoritama bio najbrži po izvođenju, drugi po redu je *OpenLayers* koji je jako blizu *GoogleMapsu* koji se nalazi na trećem mjestu. Iz provedenih usporedbi jasno se vidi da je najoptimiziranija biblioteka *Leaflet* biblioteka, zbog svoje fleksibilnosti, korisničke zajednice gdje njegovi korisnici aktivno ažuriraju *Leaflet* dodatke (engl. *Plugins)*. Također njegovo korištenje je poraslo sa godinama jer sadrži sve značajke koje su potrebne programerima za rad sa kartama. *OpenLayers* biblioteka također je jako korisna no teža je za početnike, što je ujedno i razlog zašto je *Leaflet* tako poznat i cijenjen. Sa druge strane korisnici koji trebaju biblioteku za sitne projekte mogu koristiti *GoogleMaps* biblioteku ako ne prelaze 200$ *Google* kredita.

LITERATURA

[1] Javascript, *„Intro“* - https://javascript.info/intro, pristup ostvaren 9.travnja 2020.

[2] Scott Morris, *„What is Javascript“* - https://skillcrush.com/blog/javascript/, pristup ostvaren 1.travnja 2020.

[3] *„Overview of Features of JavaScript“* - https://www.educba.com/features-of-javascript/, pristup ostvaren 4.lipnja 2020.

[4] Guru99, *„Introduction to Javascript“* - https://www.guru99.com/introduction-to-javascript.html, pristup ostvaren 9.travnja 2020.

[5] Brandan Eich - https://www.google.hr/search?q=brendan+eich&sxsrf=ALeKk02FRLDW24YR_wyPGFaDKAZVDH4XMg:1591789941306&source=lnms&tbm=isch&sa=X&ved=2ahUKEwirgojQl_fpAhUDHHcKHYilAl0Q_AUoAXoECBcQAw&biw=1920&bih=937#imgrc=8iHN01Q8s6fZaM, pristup ostvaren 10.lipnja 2020.

[6] Java, Hello World Example - https://www.geeksforgeeks.org/beginning-java-programming-with-hello-world-example/, pristup ostvaren 9.travnja 2020.

[7] Anthony Grant, *„What is Javascript and How Does It Work?“* - https://www.makeuseof.com/tag/what-is-javascript/, pristup ostvaren 9.travnja 2020.

[8] Anastasia Ovchinnikova, „*Top JavaScript API and Libraries*“ - https://medium.com/flatlogic/top-javascript-maps-api-and-libraries-162523cef967, pristup ostvaren 5.lipnja 2020.

[9] Jonathan Soma, *„Absolutely Everything* you need to know about online mapping tools“* - http://ledeprogram.com/2015/absolutely-everything-you-need-to-know-about-mapping-tools/, pristup ostvaren 13.svibnja 2020.

[10] Maptiler cloud - https://cloud.maptiler.com/maps/streets/openlayers-raster, pristup ostvaren 13.travnja 2020.

[11] *Alternative.me* - https://alternative.me/leaflet#Alternatives_to_Leaflet, pristup ostvaren 1.travnja 2020.

[12] *Google Trends* - https://trends.google.com/trends/explore?geo=US&q=google%20maps%20api,leaflet,openlayers, pristup ostvaren 15.travnja 2020.

[13] Ana Isabel Fernandes, Miguel Goluao, Armanda Rodriguez, *„A Comparison of Maps Application Programming Interfaces“* - https://agile-

online.org/conference_paper/cds/agile_2013/short_papers/sp_s5.3_fernandes.pdf, pristup ostvaren 4.lipnja 2020.

[14] Johan van der Geest, Mark Ettema, *„Comparison of JavaScript libraries“* - https://www.rug.nl/staff/r.smedinga/pubs/studcol2011.pdf#page=108, pristup ostvaren 1.travnja 2020.

[15] M.O. Khitrin, *„Comparison of JavaScript libraries for Web-Cartography“* - https://cyberleninka.ru/article/n/comparison-of-javascript-libraries-for-web-cartography/viewer, pristup ostvaren 10.lipnja 2020.

[16] Alfiya Tarasenko, *„Leaflet vs OpenLayers. What to choose?“* - https://www.geoapify.com/leaflet-vs-openlayers/, pristup ostvaren 1.travnja 2020.

[17] *jQuery* - https://jquery.com/, pristup ostvaren 19.lipnja 2020.

[18] *ReactJS* - https://reactjs.org/, pristup ostvaren 19.lipnja 2020.

[19] *LeafletJS* - https://leafletjs.com/, pristup ostvaren 1.travnja 2020.

[20] *OpenLayersJS* - https://openlayers.org/, pristup ostvaren 1.travnja 2020.

[21] *GoogleMapsJS* - https://developers.google.com/maps/documentation/javascript/tutorial, pristup ostvaren 31.ožujka 2020.

[22] *AngularJS* - https://angularjs.org/, pristup ostvaren 27.lipnja 2020.

[23] *EmberJS* - https://emberjs.com/, pristup ostvaren 27.lipnja 2020.

[24] *VueJS* - https://vuejs.org/, pristup ostvaren 27.lipnja 2020.

[25] Thomas Gratier, Paul Spencer, Erik Hazzard, *„OpenLayers 3: Beginner's Guide“* - https://books.google.hr/books?id=HnFuBgAAQBAJ&pg=PA380&lpg=PA380&dq=openlayers+size+in+kb&source=bl&ots=PZYrBXqxK9&sig=ACfU3U1Y2sEIXM7d-A8rudnhEj4S0dnqqg&hl=hr&sa=X&ved=2ahUKEwjay47n_6bqAhWC4aYKHdFEDPYQ6AEwAHoECAoQAQ#v=onepage&q=openlayers%20size%20in%20kb&f=false, pristup ostvaren 29.lipnja 2020.

SAŽETAK

Naslov: Usporedba *Javascript* biblioteka za rad sa interaktivnim kartama

U ovom Završnom radu uspoređene su *Leaflet, OpenLayers i GoogleMaps JavaScript* biblioteke po njihovoj brzini izvođenja te memorijskim otisku. Algoritmi koji su uspoređivani su: algoritam za kreiranje karte, algoritam zumiranja s dva prsta, algoritam za pronalaženje najkraće rute, algoritam za prikaz na punom ekranu, algoritam za najmanji mogući zum, algoritam za toplinske karte te algoritam za grupiranje. Iz provedenih usporedbi vidljivo je da je najoptimiziranija biblioteka *Leaflet* biblioteka.

Ključne riječi: *JavaScript* biblioteke, *Leaflet, OpenLayers, GoogleMaps,* Memorijski otisak, Brzina izvođenja algoritma, Algoritam

ABSTRACT

Title: Comparison of Javascript libraries for working with interactive maps

The goal of this Final Paper is comparison of the Leaflet, OpenLayers and GoogleMaps JavaScript libraries by their execution speed and memory footprint. The compared algorithms are: map creation algorithm, two-finger zoom algorithm, shortest route finding algorithm, full screen algorithm, lowest zoom algorithm, heat map algorithm, and grouping algorithm. From the comparisons it can be seen clear that the most optimized library is the Leaflet library.

Key words: Javascript libraries, Leaflet, OpenLayers, GoogleMaps, Memory impression, Algorithm execution speed, Algorithm

ŽIVOTOPIS

Matej Perak rođen je 4. velječe 1998. godine u Požegi. 2004. godine započinje osnovnoškolsko obrazovanje u OŠ Julije Kempfa u Požegi, a nakon toga upisuje Tehničku školu, smjer Elektrotehničar u Požegi. Tijekom školovanja sudjeluje na Državnom natjecanju iz tehničke kulture u Puli 2011. godine., te na 5. Državnom prvenstvu zrakoplovnih modelara u Osijeku 2011. godine sa postignutim 2. mjestom u kategoriji F1N-150, te 1. mjesto u kategoriji F1N-150 ekipno i postignutim 3. mjestom u kategoriji F1N ekipno. Akademske godine 2016/2017 upisuje Stručni sveučilišni studij elektrotehnike, smjer Informatika na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.