Uloga kolačića u personalizaciji i prilagodbi web aplikacija

Dražin, Matea

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

ULOGA KOLAČIĆA U PERSONALIZACIJI I PRILAGODBI WEB APLIKACIJA

Matea Dražin

Split, rujan 2023.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

ULOGA KOLAČIĆA U PERSONALIZACIJI I PRILAGODBI WEB APLIKACIJA

Matea Dražin

SAŽETAK

Cilj ovog završnog rada je definirati pojam kolačića te objasniti njihovu funkcionalnost. U radu je opisana povijest kolačića, vrste kolačića, njihova upotreba i funkcionalnost. Praktični dio rada prikazuje stvarnu primjenu kolačića u web aplikaciji.

Ključne riječi: kolačić, web stranica, web aplikacija

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 32 stranice, 10 grafičkih prikaza, 1 tablica i 37 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: **Dr. sc. Saša Mladenović,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Neposredni voditelj:

Dino Nejašmić, mag. educ. math. et inf., *predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ocjenjivači: **Dr. sc. Saša Mladenović,** *izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Dino Nejašmić, mag. educ. math. et inf., *predavač Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Ivana Marin, mag. math., *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Rad prihvaćen: **rujan, 2023.**

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of computer science

Ruđera Boškovića 33, 21000 Split, Croatia

THE USAGE OF COOKIES FOR PERSONALIZATION AND CUSTOMIZATION OF WEB APPLICATIONS

Matea Dražin

ABSTRACT

The aim of this final thesis is to define the concept of cookies and explain their functionality. The thesis describes the history of cookies, types of cookies, their usage, and functionality. The practical part of the thesis demonstrates the real-world implementation of cookies in a web application..

Key words: cookie, web page, web application

Thesis deposited in library of Faculty of science, University of Split

Thesis consists of: 32 pages, 10 figures, 1 table and 37 references

Original language: Croatian

Mentor: **Saša Mladenović, Ph.D.** *Associate Professor at the Faculty of Science, University of Split*

Supervisor: **Dino Nejašmić, mag. educ. math. et inf.,** *Lecturer at the Faculty of Science, University of Split*

Reviewers: **Saša Mladenović, Ph.D.** *Associate Professor at the Faculty of Science, University of Split*

Dino Nejašmić, mag. educ. math. et inf., *Lecturer at the Faculty of Science, University of Split*

Ivana Marin, mag. math. *Assistant at the Faculty of Science, University of Split*

Thesis accepted: **September,2023**

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom ULOGA KOLAČIĆA U PERSONALIZACIJI I PRILAGODBI WEB APLIKACIJA izradila samostalno pod voditeljstvom Dina Nejašmića, mag. educ. math. et inf., pred. U radu sam primijenila metodologiju znanstvenoistraživačkog rada i koristila literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo/la u završnom radu na uobičajen, standardan način citirala sam i povezala s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Studentica

Matea Dražin

Sadržaj

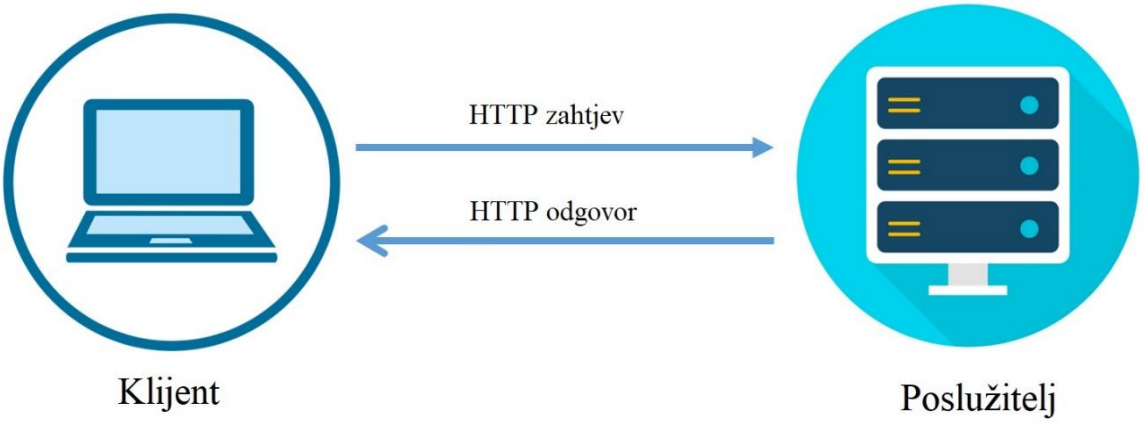
UVOD

U današnje vrijeme, gotovo svatko se susreo s pojmom internetskih kolačića. Kada pristupimo većini web stranica ili aplikacija, često se pojavi obavijest o kolačićima. Kolačići su mali tekstualni podaci koji se pohranjuju na uređaju tijekom korištenja web stranica ili aplikacija. Ovaj rad pruža općenite informacije o kolačićima, njihovoj klasifikaciji i svrsi, te temeljima web aplikacija i internetskog protokola putem kojeg se kolačići razmjenjuju. U praktičnom dijelu rada, prikazat ćemo konkretnu primjenu kolačića. S obzirom na sveprisutnost interneta u našem svakodnevnom životu, ključno je dublje razumjeti značaj kolačića i razloge zašto se koriste gotovo na svakoj web stranici ili u svakoj aplikaciji koju upotrebljavamo. Cilj ovog rada je produbiti naše razumijevanje osnovnih koncepata kolačića i njihove šire primjene kako bismo bolje razumjeli njihov utjecaj na našu interakciju na internetu. U praktičnom dijelu rada, demonstrirat ćemo njihovu funkcionalnost putem stvarne web aplikacije, pružajući dublji uvid u njihovu ulogu u suvremenom digitalnom okruženju.

1. UVODNO O WEB APLIKACIJAMA

1.1. HTTP

Svjetski web preglednici, poslužitelji i povezane web aplikacije međusobno komuniciraju putem HTTP-a, protokola za prijenos hiperteksta (Hypertext Transfer Protocol) 0 . HTTP dopušta prijenos informacija putem interneta, pružajući korisnicima mogućnost pristupa web stranicama i drugim mrežnim resursima. Za izum izvornog HTTP-a zaslužni su Tim Berners-Lee i njegov tim u CERN-u, kao i HTML-a i povezane tehnologije za web poslužitelj i klijentsko sučelje nazvano web preglednik. Berners-Lee je dizajnirao HTTP s ciljem da podrži implementaciju svoje druge ideje - projekta "WorldWideWeb", koji je prvi put predložen 1989. godine i danas je poznat kao World Wide Web [2] . Kroz godine se HTTP razvijao te je postojalo mnogo verzija, a neke koje se danas koriste su: HTTP/1.1, HTTP/2, HTTP/3. HTTP/3, čiji su prvi nacrti objavljeni 2020. godine, počeo je biti usvajan od strane glavnih web preglednika i web poslužitelja. HTTPS (Hypertext Transfer Protocol Secure) je proširenje protokola HTTP koje koristi enkripciju za sigurnu komunikaciju preko računalne mreže i općenito je korišteno na internetu. Komunikacijski protokol u HTTPS-u je kriptiran pomoću Transport Layer Security-ja (TLS), ili prije korištenog Secure Sockets Layer-a (SSL). Glavni ciljevi HTTPS protokola su osiguravanje autentičnosti web stranice koja se posjećuje, kao i zaštita privatnosti i integriteta podataka koji se razmjenjuju tijekom prijenosa [3] . HTTP je komunikacijski protokol u aplikacijskom sloju koji radi povrh TPC/IP-a (Transmission Control Protocol/Internet Protocol) [4] . Putem HTTP protokola, podaci se razmjenjuju između klijenata i poslužitelja putem interneta. Klijenti i poslužitelji uspostavljaju kontakt razmjenom individualnih poruka. Zahtjevi su poruke koje šalje klijent, a odgovori poruke koje šalje poslužitelj (Slika 1.1). Klijenti šalju zahtjeve poslužiteljima za informacije potrebne za učitavanje web stranice, a poslužitelji šalju odgovore natrag klijentima kako bi ispunili te zahtjeve [5] . U stvarnosti, između klijenta i poslužitelja, postoji više računala i uređaja koji obrađuju zahtjev. Ti uređaji uključuju usmjerivače, modeme i druge komponente koje su dio slojevitog dizajna weba.



Slika 1.1 Klijent – poslužitelj: razmjena podataka [6]

HTTP podržava nekoliko različitih naredbi zahtjeva, koje se nazivaju HTTP metode. Svaki HTTP zahtjev ima metodu. Metoda govori poslužitelju koju akciju treba izvršiti (dohvatiti web stranicu, pokrenuti pristupni program, brisanje datoteke, i tako dalje) 0 . Tablica 1 navodi pet osnovnih i najčešćih HTTP metoda.

HTTP metoda	**Opis**
GET	Šalje imenovani izvor s poslužitelja na klijent.
POST	Šalje podatke klijenta i traži da poslužitelj napravi neku promjenu.
PUT	Imenovani izvor kreira ili ažurira svoje stanje.
DELETE	Izbriše imenovani izvor s poslužitelja.
PATCH	Imenovani izvor modificira svoje stanje. Zahtjev mora sadržavati tijelo.

Tablica 1 Najčešće HTTP metode [1]

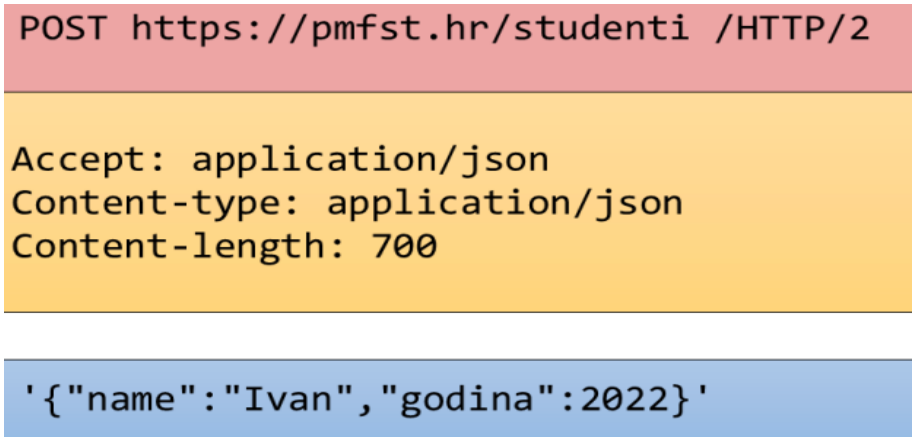
Svaki zahtjev se sastoji od nekoliko dijelova:

- **METODA** - predstavlja što klijent zahtijeva od poslužitelja,
- **URL** - označava odredišnu adresu za slanje zahtjeva,
- **PROTOKOL** - postavlja se od strane korištenog HTTP alata,
- **ZAGLAVLJA** (headers) - sadrže dodatne informacije o zahtjevu.

Postoji i neobvezni dio:

- **TIJELO** (body) - tijela zahtjeva prenose podatke na poslužitelj, a tijela odgovora prenose podatke natrag klijentu; može sadržavati proizvoljne binarne podatke (npr. slike, videozapisi, audio zapisi, softverske aplikacije) ili tekst.

Slika 1.2 prikazuje primjer jednog HTTP zahtjeva. Polje roze boje odnosi se na metodu, URL i protokol. Polje žute boje prikazuje zaglavlja nakon kojih se nalazi prazno (bijelo) polje. Zaglavlja završavaju praznim poljem, označavajući kraj popisa zaglavlja i početak tijela entiteta ako ono postoji (plavo polje) [7].

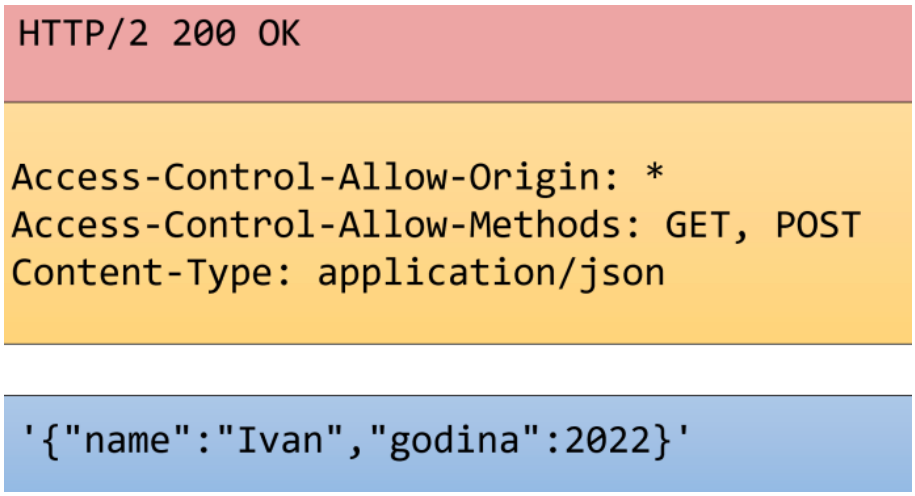


Slika 1.2 HTTP zahtjev [7]

Nizovi informacija koje klijenti i poslužitelji razmjenjuju putem HTTP zahtjeva i odgovora nazivaju se HTTP zaglavlja. Ova polja obično ostaju nevidljiva običnim korisnicima, ali igraju ključnu ulogu u komunikaciji između poslužiteljskih i klijentskih aplikacija jer omogućuju razmjenu podataka i upravljanje zahtjevima i odgovorima na primjeren način [8] . Tijelo zahtjeva je završni dio, no nije prisutan u svim vrstama zahtjeva. Poslužitelj na svaki zahtjev uzvraća HTTP odgovorom. Svaki odgovor sadrži nekoliko informacija:

- **PROTOKOL** – postavlja ga korišteni HTTP alat,
- **STATUSNI KOD** - brojčana oznaka o stanju odgovora,
- **STATUSNA PORUKA** – opisno stanje odgovora koje označava rezultat obrade zahtjeva,
- **ZAGLAVLJE** - dodatne informacije o samom odgovoru,
- **TIJELO** - podaci koje poslužitelj šalje klijentu (neobavezan dio HTTP odgovora).

Na Slici 1.3 prikazan je primjer jednog HTTP odgovora.



Slika 1.3 HTTP odgovor [7]

Statusna linija (roza polje) odnosi se na protokol, statusni kod i statusnu poruku. Zatim slijedi žuto polje koje sadrži zaglavlje samog odgovora, iza kojeg slijedi bijelo polje koje označava kraj polja zaglavlja te početak polja tijela ako ono postoji (plavo polje). Cilj HTTP odgovora je zadovoljiti klijenta pružajući mu traženi sadržaj, potvrđujući izvršenje tražene radnje ili obavještavajući ga o pogrešci koja se dogodila tijekom obrade zahtjeva [9] . Protokol se pojavljuje na početku statusne linije u formatu HTTP/x.y. x.y. Statusni kod, koji slijedi nakon protokola, govori klijentu što se dogodilo. Iza statusnog koda slijedi statusna poruka koja je lako razumljiva čovjeku. Različiti statusni kodovi grupirani su u klase svojim troznamenkastim numeričkim kodovima. Informativni odgovori su prikazani statusnim kodovima u rasponu 100-199. Statusni kodovi između 200 i 299 predstavljaju uspjeh. Kodovi između 300 i 399 označavaju da je izvor bio pomaknut. Kodovi između 400 i 499 znače da je klijent nešto krivo napravio u zahtjevu. Kodovi između 500 i 599 znače da je nešto pošlo po zlu na poslužitelju 0 .

1.2. Arhitektura web aplikacija

Arhitektura web aplikacije opisuje međusobnu povezanost njezinih komponenti i njihovu komunikaciju. Definira kako se podaci dostavljaju putem HTTP-a i osigurava da klijentski i poslužiteljski dio mogu razumjeti podatke. Arhitektura web aplikacije ima zadaću uskladiti sve komponente aplikacije: ono što korisnici vide i sa čime stupaju u interakciju te način na koji softver kontrolira unutarnjim operacijama [10] . Dvije su osnovne komponente svake web aplikacije:

- Klijentska strana – obično je to preglednik, komponenta na strani klijenta, to jest komponenta na prednjoj strani (front-end) koja igra ključnu ulogu u interakciji s korisnikom [11] .
- Poslužiteljska strana – web poslužitelj, također nazivan pozadinskom komponentom (back-end) ili komponentom na strani poslužitelja, preuzima ključnu ulogu u upravljanju poslovnom logikom i procesuiranju zahtjeva korisnika [11] .

Komponente na klijentskoj strani pisane su kombinacijom HTML-a, CSS-a i Javascript-a. Komponente poslužiteljske strane pisane su u C#-u, Javi, Javascript-u. Python-u, PHP-u i Ruby-ju. Neke od vrsta arhitekture web aplikacija su arhitektura klijent – poslužitelj, troslojna arhitektura, arhitektura mikrosevisa, arhitektura bez poslužitelja itd.

1.2.1. Klijent – poslužitelj

Distribuirana struktura aplikacije poznata kao klijent-poslužitelj model podrazumijeva podjelu zadataka ili opterećenja između entiteta koji osiguravaju resurse i usluge, nazvanih poslužitelji, te entiteta koji traže te usluge, poznatih kao klijenti. Komunikacija klijenta i poslužitelja se većinom odvija preko računalne mreže na odvojenom hardveru, no klijent i poslužitelj se mogu nalaziti u istom sustavu [12] . Osnovna struktura klijent-poslužitelj arhitekture se naziva dvoslojna struktura. To je softverska arhitektura u kojoj se prezentacijski sloj realizira na klijentskoj strani, a podatkovni sloj se pohranjuje na poslužitelju [13] . Prezentacijski sloj uključuje HTML, CSS i JavaScript koji se koriste za izradu i stiliziranje korisničkog sučelja. Ovaj sloj korisnicima omogućava unos podataka, pregledavanje sadržaja i interakciju s web aplikacijom. Aplikacijski sloj sadrži poslovnu logiku i obradu podataka. Ovaj sloj sadrži funkcionalnosti koje upravljaju podacima, obrađuju zahtjeve korisnika i izvršavaju sve nužne informacije kako bi osigurali ispravan rad web aplikacije. Dvoslojna arhitektura je osnovna i jednostavna, što je čini prikladnom za manje projekte i manje iskusne razvojne timove. Također, s manje slojeva za upravljanje, razvoj aplikacije može biti brži, posebno kada je brza izrada prototipa bitna. No, dvoslojna arhitektura može imati poteškoća sa skalabilnošću jer se prezentacijski i aplikacijski sloj izvode zajedno, što može uzrokovati probleme s opterećenjem prilikom povećanja broja korisnika. Zato što prezentacijski i aplikacijski sloj nisu odvojeni, teže je održavati i ažurirati kod prilikom rasta web aplikacije. Stoga se većinom koristi troslojna arhitektura web aplikacije jer je naprednija i fleksibilnija od dvoslojne arhitekture. Troslojna arhitektura se dijeli na tri sloja: prezentacijski, aplikacijski i podatkovni sloj. Prezentacijski i aplikacijski sloj imaju istu ulogu kao i kod dvoslojne

arhitekture, a podatkovni sloj se bavi upravljanjem podacima i komunikacijom s izvorima podataka kao što su baze podataka ili datoteke. Podatkovni sloj sadrži operacije za dohvaćanje, spremanje i manipulaciju podacima. Prednost troslojne arhitekture leži u tome da svaki sloj radi neovisno na vlastitoj infrastrukturi. To omogućava istodoban razvoj i ažuriranje slojeva odvojenih timova, bez utjecaja na druge slojeve. Skaliranje ili proširenje jednog sloja ne utječe na funkcioniranje ostalih slojeva [14] . No, komunikacija između slojeva može dovesti do povećanja vremena odziva aplikacije, posebno ako nije pravilno optimizirana.

1.3. HTML, CSS, JavaScript

HTML (Hypertext Markup Language) je temeljni računalni jezik koji se koristi za izradu web aplikacija. On definira strukturu web aplikacije i njezinog sadržaja. Sastoji se od različitih elemenata koji se upotrebljavaju za definiranje i organiziranje različitih dijelova sadržaja. Ovi elementi omogućuju da je struktura sadržaja baš onakva kakvu je želimo [15] . Jedina zadaća HTML bi trebala biti označavanje dijela teksta (kako što sami naziv programskog jezika nalaže – „markup language“ – jezik za označavanje teksta). Za vizualni prikaz elementa koristimo CSS (Cascading Style Sheets), računalni jezik koji definira izgled web aplikacija. Na početku svake HTML stranice mora postojati oznaka <!DOCTYPE html>. To zapravo nije ni oznaka ni element, već informacija pregledniku kakvu vrstu dokumenta treba očekivati. Na ovaj način preglednik zna kako renderirati stranicu. Upravo ova vrijednost označava standard HTML5 koji je najnovija verzija HTML standarda te ga skoro sve web stranice i aplikacije koriste. Meta oznake su oznake koje sadrže metapodatke dokumenta. Kada stvaramo neki HTML dokument ili stranicu, postoje neke meta oznake koje su bitne za daljnju funkcionalnost HTML dokumenta ili stranice. Neke od njih su:

- <meta charset="UTF-8"> - označava skup znakova za enkodiranje stranice što je važno za prikaz posebnih znakova,
- <meta name="viewport" content="width=device-width> - označava prilagodbu veličine ekrana i prikaza na manjim ekranima.

Postoje dvije osnovne grupe HTML elemenata koje su važne za definiranje strukture stranice: blok elementi i inline elementi. Blok elementi su oni elementi koji će uvijek biti prikazani u novom redu, a inline elementi se nastavljaju na susjedne elemente. Otvarajući HTML elementi mogu imati atribute koji daju dodatne informacije o tom elementu. Bitno je razlikovati naziv i vrijednost pripadajućeg atributa. Jedan od jako korisnih atributa je id atribut koji služi za jedinstvenu identifikaciju elementa u odnosu na ostale elemente na

stranici. Svaki element ne mora sadržavati ovaj atribut te dva elementa ne bi smjela koristiti istu vrijednost id atributa na istoj stranici. Postoji i class atribut koji se koristi kada različite elemente na stranici želimo logički povezati u zajedničku cjelinu iako se može koristiti i samostalno. Ova dva atributa se najčešće koriste u kombinaciji sa CSS-om ili JavaScriptom kada želimo postići točno određeni izgled aplikacije ili dati jedinstvenu funkcionalnost određenom elementu [16] . Slika 1.4 prikazuje Primjer 1 u kojem je navedena struktura HTML stranice koja sadrži: <!DOCTYPE html>, meta oznake, primjer nekih elemenata (<p>, , <ul>) i primjer id i class atributa.

primjer1.html > html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Primjer 1</title>
7   </head>
8   <h1>Glavni naslov</h1>
9   <body>
10      <p id="id">Tijelo stranice</p>
11      <div class="klasa">Div element</div>
12      <ul>
13          <li>1.element liste</li>
14          <li>2.element liste</li>
15          <li>3.element liste</li>
16      </ul>
17  </body>
18  </html>
```

Slika 1.4 Primjer 1 – HTML

CSS (Cascading Style Sheets) je računalni jezik koji se koristi za kreiranje i opisivanje izgleda HTML stranice ili dokumenta. Ovaj jezik se bavi stilskom obradom i prikazom HTML elemenata korisnicima pa se zato i naziva stilskim jezikom [17] . Temelj CSS-a su pravila pomoću kojih se definira stil ili grupa stilova koja se primjenjuje na određeni element ili skupinu elemenata. Svako pravilo započinje selektorom koji definira na koji HTML element ili skupine elemenata će se primijeniti navedena stilska pravila. Nakon selektora slijedi deklaracijski blok unutar kojeg se navode različite deklaracije koje se pišu kao parovi „svojstvo:vrijednost“ (property:value pairs). CSS ima mnogo svojstava kao i mjernih jedinica za izražavanje vrijednosti svojstava. Vrijednosti mogu biti apsolutne – one koje se baziraju na stvarnim mjernim jedinicama te ne ovise o veličini ekrana ni drugim elementima i relativne – one kojima konačna vrijednost ovisi o ostalim elementima. Vrste selektora koje se koriste su:

- univerzalni – služe za odabiranje svih elemenata stranice (oznaka: *),
- selektori tipa – služe za odabiranje svih elemenata tog tipa (neki od tipova: p, div, h1),
- selektori atributa – služe za odabiranje elemenata po vrijednosti tog atributa,
- id selektori – služe za odabiranje elemenata po vrijednosti id atributa (oznaka: #),
- selektori klase – služe za odabiranje elemenata prema vrijednosti class atributa,
- pseudo-klase – služe za odabiranje elemenata čija se svojstva ne mogu postaviti kroz HTML strukturu,
- pseudo-elementi – služe za odabir određenog dijela HTML elementa,
- kombinacijski selektori – većina selektora se može kombinirati kako bi mogli odabrati točno određeni element,

„Cascading“ dio naziva CSS-a odnosi se na način na koji preglednik određuje prioritet nekog pravila. CSS sadrži 4 osnovne razine kaskade:

1. **VAŽNOST –** oznaka važnosti (!important) se može dodati CSS deklaracijama kako bi premostili važnost ostalih deklaracija,
2. **IZVOR –** izvor stilskih pravila može biti preglednik, korisnik (kroz postavke preglednika) i web autori (programeri),
3. **SPECIFIČNOST –** u praksi je najčešća pojava specifičnost pravila; specifičnije pravilo ima veću važnost,
4. **REDOSLIJED –** poredak pravila u CSS dokumentu je od velike važnosti. Ako dva pravila imaju slična svojstva, kasnije navedeno pravilo ima prioritet. Ako imamo dvije deklaracije u istom pravilu, kasnije dodana ima prioritet.

Većina CSS pravila se nasljeđuje. Neka svojstva se nikada ne nasljeđuju, a to su margine, visina i širina elementa. Ako neki element nema postavljeno svojstvo koje se može naslijediti, redom se pretražuje roditelj tog elementa da se provjeri ima li tu vrijednost koja se može naslijediti. Kad želimo naslijediti neko svojstvo kojem nije zadano automatsko nasljeđivanje, koristimo vrijednost inherit. Pomoću ove vrijednosti je moguće naslijediti samo od direktnog roditelja. Initial vrijednost se upotrebljava kada ne želimo naslijediti neko svojstvo. Initial postavlja vrijednost svojstva na stil zadan preglednikom. Još postoji i vrijednost revert koja je slična initial vrijednosti, no ona nasljeđuje vrijednost od roditelja (ako je u mogućnosti) ili vrijednost postavlja na početnu vrijednost zadanu u samom CSS dokumentu [18] . Programski jezik **JavaScript** (skraćenica JS) je osnovna tehnologija World Wide Web-a, uz HTML i CSS. JavaScript je najpoznatiji kao skriptni jezik za web aplikacije, no upotrebljavaju ga i druga okruženja bez preglednika, kao što su Node.js, Adobe Acrobat i Apache CouchDB [19]

. Dinamičke mogućnosti koje JavaScript posjeduje su dinamičko stvaranje skripte, oporavak izvornog koda, popise parametara varijabli, implementacija kolačića i druge. Pokretačka jedinica JavaScript engine je program koji JavaScript kod pretvara u strojni kod (oblik razumljiv računalu). Neki od poznatih JS engina su V8, Chakra, SpiderMonkey, JavaScriptCore. JavaScript je interpreterski jezik koji koristi Just In Time (JIT) kompiliranje koje prevodi JavaScript kod u izvršni byte-kod prilikom pokretanja web aplikacije. U slučaju web aplikacija, JS se koristi za implementaciju aplikacijske logike što se odnosi na dinamičnost prikaza i komunikaciju sa poslužiteljem. Tipovi podataka u JS-u dijele se na primitivne (string, number, boolean), posebne (null, undefined, symbo) i složene tipove podataka (objekti). Primitivni tipovi podataka imaju samo jednu vrijednost, a složeni mogu sadržavati više vrijednosti. Primitivne vrijednosti predstavljaju osnovne podatke koji nisu objekti i nemaju pridružene metode. Ove vrijednosti su nepromjenjive, što znači da im se dodijeljena vrijednost ne može promijeniti. Kada koristimo operator „=“ za dodjelu vrijednosti varijabli, zapravo u varijablu pohranjujemo novu vrijednost, dok se postojeća vrijednost ne mijenja. Suprotno primitivnim vrijednostima su objekti koji predstavljaju skup podataka, to jest svojstva objekta. Svako svojstvo se zapisuje kao par ključ:vrijednost (key:value pair). Ključ prikazuje naziv svojstva te se zapisuje kao string. Vrijednost predstavlja vrijednost tog svojstva i može biti bilo kojeg tipa. Objekti mogu sadržavati i metode, to jest svojstva kojima je vrijednost neka funkcija. U JavaScriptu postoje 3 načina deklariranja varijabli, a to su let, const i var. Let se koristi za deklariranje varijabli na razini bloka, što znači da je varijabla dostupna samo unutar bloka u kojem je definirana. Const se upotrebljava za deklaraciju varijabli čija se vrijednost neće mijenjati nakon što je prvi put deklarirana. Var je izvorna naredba za deklaraciju varijabli i ima funkcionalnu dostupnost, što znači da je dostupna globalno ili na razini cijele funkcije u kojoj je definirana [20] .

2. KOLAČIĆI

2.1. Povijest kolačića

Kolačići (eng. *cookies*), također zvani HTTP kolačići, Internet kolačići, web kolačići ili jednostavno kolačići su mali skupovi podataka koje web poslužitelj pohranjuje dok korisnik provodi vrijeme na nekoj web stranici ili aplikaciji [21] . 1994. godine, Lou Montulli, koji je u to vrijeme bio programer tvrtke Netscape Communications koja je razvijala aplikaciju za e-trgovinu za MCI (MCI, Inc. je bila telekomunikacijska tvrtka), došao je na ideju da iskoristi već postojeći tzv. „čarobni kolačić“ (ovaj termin se odnosi se na neproziran identifikator koji se prenosi između različitih dijelova softvera [22]) u web komunikacijama. MCI tvrtka nije željela da njihovi poslužitelji čuvaju podatke o djelomičnim stanjima transakcija pa su od Netscape-a tražili da pronađu način rješavanja tog problema (da se ti podaci pohranjuju u računala njihovih korisnika). Kolačić je riješio problem pouzdane implementacije virtualne košarice za kupnju. Montulli je te iste godine zajedno s Johnom Giannandreom napisao početnu specifikaciju Netscape kolačića. Prva praktična primjena kolačića izvan laboratorijskih okruženja bila je za provjeru jesu li su posjetitelji web stranice koristili Netscape preglednik. 1995. godine, Montulli je podnio zahtjev za patentiranje tehnologije kolačića, koji je odobren 1998. godine. U Internet Exploreru verzije 2, koja je objavljena u listopadu 1995. godine, integrirana je podrška za kolačiće. U to vrijeme, uvođenje kolačića nije bilo široko poznato javnosti. Kolačići su bili automatski prihvaćeni prema zadanim postavkama, bez informiranja korisnika o njihovoj prisutnosti. Javnost je postala svjesna kolačića nakon što je Financial Times objavio članak o njima 12. veljače 1996. godine. Iste godine kolačići su se našli pod snažnim medijskim reflektorom, posebno zbog potencijalnih utjecaja na privatnost. O kolačićima se diskutiralo na dva saslušanja američke Savezne komisije za trgovinu 1996. i 1997. Rasprave o formalnoj specifikaciji započele su u travnju 1995. te se tada formirala posebna radna skupina unutar Internet Engineering Task Force (IETF). Dva alternativna prijedloga za uvođenje stanja u HTTP transakcije bila su predložena, jedan od strane Briana Behlendorfa, a drugi od strane Davida Kristola. Grupa koju su vodili Kristol i Montulli, odlučila se na korištenje Netscape-ove specifikacije kao početnu točku. Ta grupa je 1996. otkrila kolačiće trećih strana koji su bili značajna prijetnja privatnosti.

Specifikaciju koja navodi da kolačići trećih strana nisu dozvoljeni ili bar nisu omogućeni izradila je grupa koju su vodili Kristol i Montulli [21] .

2.2. Općenito o kolačićima

Nekada su se kolačići koristili kao opća metoda pohrane na klijentskoj strani. Kolačići su prvotno bili korišteni za pamćenje artikala u virtualnoj košarici. Danas se koriste kako bi se trenutne informacije o korisniku pohranile na pregledniku. Glavna upotreba kolačića je podijeljena u tri kategorije: upravljanje sesijom, personalizacija i praćenje. Današnje web stranice i aplikacije više nisu u mogućnosti funkcionirati kako treba bez upotrebe kolačića. Zbog toga postoji skupina kolačića koji su nužni za normalan rad aplikacije. Digitalizacijom se uvelike proširila upotreba kolačića. Kao što možemo primijetiti, svaka web stranica nam pokušava nametnuti što veći broj kolačića. Veći broj kolačića znači veći broj informacija o korisnicima. Tim informacijama se kasnije može trgovati. Primjerice, Google pamti korisnikove unose u tražilicu i prosljeđuje te informacije drugim aplikacijama koje će prikazivati oglase na temelju tih pretraga. Preglednik pomoću kolačića može stvoriti profil s nizom korisničkih podataka.

2.3. Osnovna podjela kolačića

Kada posjetitelj pristupi web stranici ili izvrši određenu radnju na njoj, s web stranice se prenosi mala tekstualna datoteka koja se pohranjuje u pregledniku posjetitelja.. Kada se posjetitelj vrati na web stranicu, poslužitelj može pristupiti kolačiću koji je pohranjen u njegovom pregledniku i dohvatiti informacije o njemu, kao što su njegove prethodne radnje na web stranici. Kolačiće općenito dijelimo na temelju njihovih karakterističnih svojstava kao što su izvor, trajanje i namjena. Postoje 3 glavne vrste kolačića, a to su: kolačići na temelju izvora, kolačići na temelju trajanja i kolačići na temelju namjene [23] .

2.3.1. Kolačići na temelju izvora

Kolačići na temelju izvora dijele se na kolačiće prve strane i kolačiće treće strane. **Kolačići prve strane** su bezopasnija vrsta web kolačića koje postavlja web stranica ili aplikacija koju se pregledava. Zbog toga što kolačiće prve strane mogu čitati samo web stranice koje ih postavljaju, korisnika web stranice nije moguće pratiti. Kolačići prve strane pridonose ugodnijem iskustvu prilikom korištenja interneta jer automatski pohranjuju informacije. Na

primjer, nećete morati ponovno unositi svoje spremljene podatke prilikom kupovine, a web stranica će zapamtiti vaš odabrani jezik. Kolačići prve strane pohranjuju korisničke podatke kao što su individualne postavke web stranice, poput odabira jezika i podataka za prijavu. Neki od primjera kolačića prve strane su:

- Tehnički nužni kolačići – neophodni su za osnovno djelovanje web stranice i sadrže samo tehnički nužne funkcionalnosti; na primjer, u online trgovini, oni omogućuju funkcionalnost poput košarice za kupnju,
- Funkcionalni kolačići – neki od njih su ključni za omogućavanje naprednih značajki izvan osnovnih funkcija, kao što su poboljšani fontovi, reprodukcija videa ili interaktivne mogućnosti Web 2.0,
- Statistički kolačići – uloga im je prikupiti informacije o korisnikovom ponašanju na posjećenoj web stranici,
- Marketinški kolačići – upotrebljavaju se za prilagođavanje oglasa korisnicima i zadržavanje informacija o njihovim interesima [24] .

Postoje određene prednosti za operatere web stranica, ali i za korisnike web stranice. Glavna prednosti koju koriste web operateri je da kolačiće prve strane može koristiti softver za analizu koji pruža informacije o korisničkom ponašanju posjetitelja web stranice. Pomoću tih rezultata, softver kreira korisnički profil i prati kretanje korisnika. Prikupljeni podaci se mogu iskoristiti i da bi učinili korištenje web stranica ugodnijim. Prednost za posjetitelje web stranice je što samo operater te web stranice može pristupiti jedinstvenom identifikacijskom broju posjetitelja, dok vanjske strane nemaju takav pristup. Također, možete sačuvati svoje postavke, odabire i određene podatke u svom pregledniku kako biste ih izbjegli ponovno unositi svaki put. Kolačići prve strane obično su omogućeni prema zadanim postavkama u vašem pregledniku i mogu se isključiti ručno. **Kolačići treće strane** za razliku od onih prve strane, omogućuju pristup na više domena. Kolačići treće strane su također poznati kao "kolačići za praćenje" jer im omogućuju praćenje vaših aktivnosti kao posjetitelja na različitim web stranicama. Prednost kolačića treće strane je njihova sposobnost praćenja ponašanja posjetitelja web stranice kako bi se omogućilo ciljano prikazivanje personaliziranih oglasa. Još jedna prednost kolačića treće strane je za vlasnike web stranica koji pomoću njih mogu bolje razumjeti interakciju korisnika i njihove web stranice. To im pomaže u razvoju i usavršavanju njihove stranice [28] . Iako je za većinu korisnika web stranica ova prednost zapravo nedostatak jer korisnici ne žele da se njihove aktivnosti prate. Njihove privatne informacije se mogu zloupotrijebiti, zbog toga zadane postavke mnogih preglednika

automatski blokiraju kolačiće treće strane. Na primjer, u Mozilla Firefox pregledniku, kolačići treće strane su blokirani od rujna 2019. godine, dok Apple Safari preglednik više ne prihvaća ove kolačiće od ožujka 2020. godine [24] .

2.3.2. Kolačići na temelju trajanja

Kolačići na temelju trajanja mogu se podijeliti u dvije skupine, a to su kolačići sesije i trajni kolačići. **Sesijski kolačići**, za razliku od drugih, se nikad ne pohranjuju na korisnikov uređaj. Kolačići sesije traju jednu sesiju, to jest period od pokretanja web mjesta do napuštanja tog web mjesta. Za to vrijeme kolačići s informacijama se pohranjuju na privremenu memorijsku lokaciju koja se na kraju sesije izbriše. Ovi kolačići prate kretanje korisnika unutar web stranice i podatke kao što su korisnički unosi. Poslužitelj nasumično generira „ID sesije“, koji privremeno pohranjuje kolačić sesije. Ovi kolačići su specifični za poslužitelj i ne dijele se drugom stroju. Kolačići sesije su jako bitni za korisničko iskustvo jer web stanica ili aplikacija pomoću njih pamti korisnike i njihovo ponašanje na web stranicama. Najčešći primjer korištenja sesijskog kolačića je u kontekstu online trgovina, posebno kada koristite košaricu za kupnju. Kada posjetite web trgovinu i dodate proizvode u svoju košaricu, sesijski kolačić pamti te informacije. Ovo omogućuje da vaša košarica zadrži odabrane proizvode dok se pripremate za plaćanje. Bez sesijskog kolačića, stranica za plaćanje ne bi mogla zapamtiti vaš izbor i vaša košarica bi ostala prazna. Kolačići sesije spadaju u nužne kolačiće prema Općoj uredbi o zaštiti podataka (GDPR) [25] koja je na snazi u Europskoj uniji i Ujedinjenom Kraljevstvu. **Trajni kolačići**, za razliku od kolačića sesije, ostaju na pregledniku korisnika znatno dulje. Njima je unaprijed određeno vrijeme zadržavanja na računalu korisnika, nakon isteka vremena, oni će se automatski izbrisati. Prate informacije, postavke ili podatke za prijavu te prepoznaju različite korisnike [27] . Ideja iza ovih kolačića je pružanje boljeg i efikasnijeg korisničkog iskustva. Problem s trajnim kolačićima je što mogu sakupljati i privatne informacije koje mogu biti zlouporabljene. Još jedan potencijalni problem je što zauzimaju prostor na tvrdom disku [26] .

2.3.3. Kolačići na temelju namjene

Kolačići na temelju namjene mogu se podijeliti u četiri skupine [23] : strogo neophodni kolačići, kolačići izvedbe, funkcionalni kolačići i oglašivački kolačići. **Strogo neophodni (nužni) kolačići** kao što im ime sugerira nužni su za kvalitetno funkcioniranje web stranice [30] . Značajni su kod radnji kao što su prijava, dodavanje artikla u košaricu, odjava i

plaćanje. Strogo neophodni kolačići ne prikupljaju osobne podatke ni navike pregledavanja. Služe samo kako bi olakšali boravak korisnika na stranici, zbog toga su automatski prihvaćeni od strane preglednika. U ovu kategoriju spadaju kolačići prve strane. **Kolačići izvedbe** su posebni kolačići koji služe za prikupljanje podataka o tome kako korisnici upotrebljavaju web stranicu [29] . Oni bilježe najčešće posjećene stranice na web mjestu i prate pojavu poruka o pogreškama na stranicama. Ovi kolačići služe isključivo za poboljšavanje funkcionalnosti stranice, a ne prikupljanje identifikacijskih podataka korisnika. Prikupljeni podaci su anonimni i koriste se primjerice za brojanje posjeta stranici, brzine učitavanja stranice, stope napuštanja stranice i vrijeme mirovanja korisnika na stranici. Ove informacije se koriste za razumijevanje interesa korisnika kako bi se osigurala komunikacija s korisnikom te pružanje usluga ili isporuka proizvoda. Primjer ovih kolačića su kolačići treće strane. **Funkcionalni kolačići** su slični kolačićima izvedbe, ali za funkcionalne kolačiće korisnik odlučuje hoće li dati svoj pristanak. Izbor i pristanak na uporabu funkcionalnih kolačića često se doživljava kao manje stresno iskustvo jer ovi kolačići ne prate osobne podatke korisnika na web stranicama. Ovi kolačići se koriste kako bi omogućili personalizaciju stranice po korisnikovom izboru [31] . Primjerice, ako korisnik podijeli svoju lokaciju, web stranica može pružiti posebne značajke kao što su vremenska prognoza ili lokalne vijesti. Funkcionalni kolačići mogu uključivati kolačiće prve strane, treće strane, kolačiće sesije ili trajne kolačiće. U **oglašivačke kolačiće** spadaju kolačići za praćenje aktivnosti i ponašanja korisnika na mreži kako bi se izgradio profil interesa korisnika i prikazale im relevantne reklame na drugim web stranicama. Uglavnom su to kolačići treće strane koje koriste, na primjer Google Ads, Amazon Publisher Services i ostale oglašivačke mreže [23] . Obično ih oglašivačke mreže instaliraju u preglednik korisnika. Mogu jednostavno prepoznati korisnikov preglednik, uređaj i lokaciju, navike pretraživanja i različite postavke, ali ne uzimaju izravno osobne podatke.

2.4. Posebni tipovi kolačića

Neki od posebnih tipova kolačića su: superkolačići, zombi kolačići, Flash kolačići i sigurni kolačići. **Zombi kolačići** su dobili ime po svojoj sposobnosti da se „vrate iz mrtvih“. Korisnik nije u mogućnosti ukloniti ovu vrstu kolačića zbog mehanizama za sprječavanje brisanja podataka od strane korisnika. Ovi kolačići se postavljaju izvan prostora za pohranu kolačića preglednika. Zombi kolačići često uspješno izbjegavaju sva ograničenja ili blokiranje kolačića treće strane, jer se ponovno stvaraju ili obnavljaju. S obzirom na to da ne ovise o standardnim protokolima kolačićima, preglednik korisnika može nastaviti obnavljati izbrisane kolačiće,

čak i nakon što je korisnik odlučio ne prihvaćati ih [32] . **Flash kolačić**, koji je također poznat kao lokalni dijeljeni objekt, poslužitelj šalje klijentu kad preglednik zatraži sadržaj kompatibilan s Adobe Flash, popularnim dodatkom za preglednike. Flash kolačići se razlikuju od običnih kolačića preglednika po količini podataka koju mogu pohraniti i načinu na koji se kolačić može obrisati [33] . Suprotno od standardnog kolačića preglednika, brisanje Flash kolačića zahtijeva pristup postavkama Adobe Flash Playera jer se ovi kolačići pohranjuju u posebnu Adobe datoteku. Flash kolačići ponekad sadrže slične informacije kao HTTP kolačići, ali dodatno mogu pohraniti specifične podatke za Flash, kao što su informacije o tome gdje je korisnik zaustavio reprodukciju videa ili kada se animirani reklamni natpis zaustavio rotirati. Kad korisnik izbriše HTTP kolačiće svog preglednika, Flash kolačići mogu ostati nepromijenjeni i koristiti se za obnovu izbrisanih HTTP kolačića. Kako bi se suzbila moguća zloupotreba lokalnih dijeljenih objekata, Adobe je izvršio izmjene u Flashu kako bi otežao obnavljanje Flash kolačića i također pružio upute na svojoj web stranici o tome kako upravljati Flash kolačićima. **Superkolačići** su sitni dijelovi koda koji se postavljaju na korisnikovom uređaju bez njegova znanja. Oni ostaju nevidljivi korisniku i ne zahtijevaju njegovu izričitu dozvolu za instalaciju, što ih čini izazovnima za detekciju [34] . Superkolačići se razlikuju od standardnih kolačića jer ih nije moguće jednostavno ukloniti ili blokirati putem postavki preglednika te se mogu pohraniti na različitim lokacijama na uređaju, kao što su registri ili Flash Player. Ovi kolačići predstavljaju veliku opasnost za korisničku privatnost jer ih je gotovo nemoguće potpuno ukloniti. Brisanje pregledničke predmemorije ili upotreba alata za blokiranje oglasa i praćenje privatnosti obično nije dovoljno za ograničavanje njihove upotrebe ili uklanjanje s uređaja [23] . Jedna vrsta superkolačića poznata kao "Evercookie" predstavlja JavaScript aplikacijsko programsko sučelje (API) koje otkriva i obnavlja kolačiće koje korisnici namjerno izbrišu iz pohrane svog preglednika [35] . **Sigurni kolačići** su posebna vrsta HTTP kolačića koji su označeni atributom "Secure". Taj atribut ograničava opseg kolačića na komunikaciju preko "sigurnih" kanala, što je obično definirano od web preglednika. Kada kolačić sadrži atribut "Secure", preglednik će uključiti taj kolačić u HTTP zahtjev samo ako se taj zahtjev prenosi preko sigurnog kanala, što je obično HTTPS [36] . Iako se čini kao korisna zaštita kolačića od aktivnih mrežnih napadača, atributu Secure je jedini cilj očuvanje povjerljivosti kolačića. Aktivni mrežni napadač može izbrisati sigurne kolačiće s nesigurnog kanala, što dovodi do narušavanja integriteta kolačića. Neki preglednici, poput Chrome verzije 52 i Firefox verzije 52, odlučili su se odreći ove specifikacije u korist veće sigurnosti i zabranjuju nesigurnim web stranicama (HTTP) postavljanje kolačića putem Secure direktive.

2.5. Sigurnost kolačića

Zbog sve veće zloupotrebe kolačića, sigurnost kolačića postaje sve važnija. Napadači mogu zloupotrijebiti kolačiće na zlonamjeran način čineći jedno od sljedećeg:

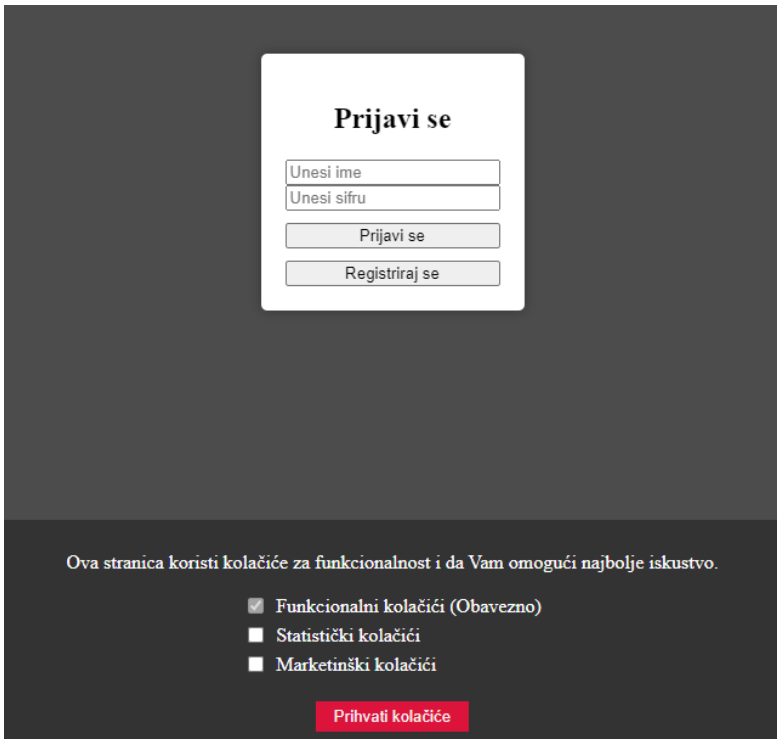
- krađa kolačića koji sadrže osjetljive podatke poput ID-ova sesije ili autentifikacijskih kolačića,
- iskorištavanje ukradenih kolačića kako bi se ponovno pristupilo autentificiranim dijelovima web stranice ili postojećim korisničkim sesijama,
- manipulaciju kolačićima kako bi se stvorili lažni autentifikacijski podaci radi pristupa autentificiranim dijelovima web stranice ili postojećim sesijama.

U specifikaciji HTTP protokola nalazi se nekoliko mehanizama koje programeri mogu iskoristiti kako bi smanjili rizik od neovlaštenog pristupa napadača, ponovne upotrebe ili krivotvorenja kolačića koji sadrže osjetljive podatke. Kako bi se zaštitili podaci koje kolačić sadržava, potrebno je kolačiće slati samo preko šifriranih veza. Postavljanjem sigurne oznake (engl. „*secure flag*“) onemogućuje se slanje kolačića preko nešifrirane veze. U osnovi, postavljanjem sigurne oznake pregledniku se nalaže da kolačić nikada ne šalje s nešifriranim zahtjevima prema poslužitelju. Kolačić će se dodavati samo tijekom komunikacije putem sigurnih veza kao što je HTTPS (HTTP preko Transport Layer Security - TLS). Bilo bi dobro uvijek koristiti Secure oznaku u kolačićima koji sadrže osjetljive podatke. Secure oznaku bi trebalo koristiti čak i ako koristimo HTTPS jer postoji mogućnost da mrežni napadač presretne HTTP zahtjeve i preusmjeri ih da uhvate otvorene kolačiće. Postoji i HTTPOnly oznaka koja sprječava skripte da čitaju kolačiće. Kao što samo ime „HTTPOnly“ sugerira, preglednik će koristiti kolačić isključivo u okviru HTTP(S) zahtjeva. Ova mjera sprječava hakere da iskoriste ranjivosti kao što su XSS (Cross-Site Scripting) kako bi pristupili sadržaju kolačića. Oznaka SameSite je eksperimentalna značajka koju je Google uveo u Chrome verziji 51 s ciljem smanjenja rizika od CSRF (Cross-Site Request Forgery) napada. Kada poslužitelj postavi svoju vrijednost na „strogo“ (engl. „*strict*“), preglednik neće slati kolačić na vašu web stranicu ako zahtjev dolazi s druge domene, čak ni kada korisnik izravno klikne na vezu. Oznaka HostOnly određuje dostupnost kolačića poddomenama. To je zastavica koju preglednik automatski postavlja kada je atribut domene prazan. Preporučljivo je ostaviti atribut domene praznim, osim ako izričito želite dijeliti sadržaj kolačića sa svim poddomenama i znate da je to sigurno [37] .

3. PRAKTIČNI DIO

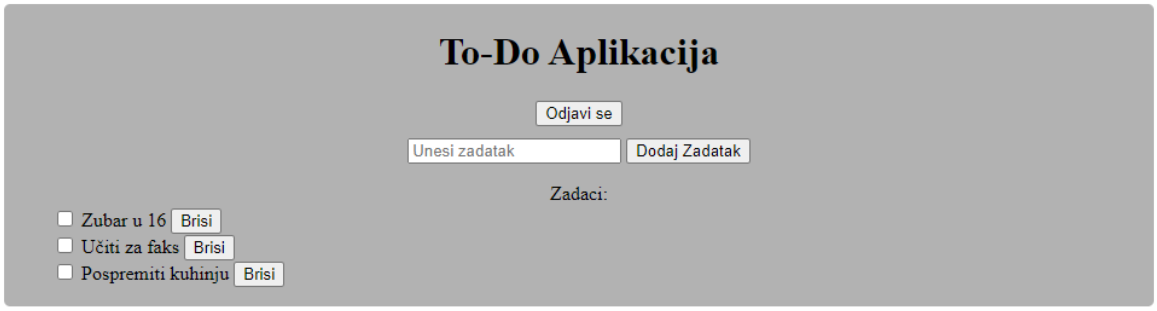
3.1. To-Do Aplikacija

To-Do Aplikacija je naziv web aplikacije koja je izrađena u okviru ovog projekta kao primjer upotrebe privremenih kolačića za prijavu te trajnih kolačića za zadatke na tom računu. Pošto je ovo web aplikacija, radi na svim računalima sa internetskim preglednikom. Kada se aplikacija pokrene, na početnom ekranu vidimo formu za prijavu ili registraciju u aplikaciju sa elementima koji primaju korisničko ime i lozinku. Kada se korisnik registrira ili prijavi, ova forma se skriva, a prikazuje mu se nova forma gdje može spremati svoje zadatke i „čekirati“ ih kao završene nakon što ih obavi.



Slika 3.1 Prikaz forme za registraciju/prijavu

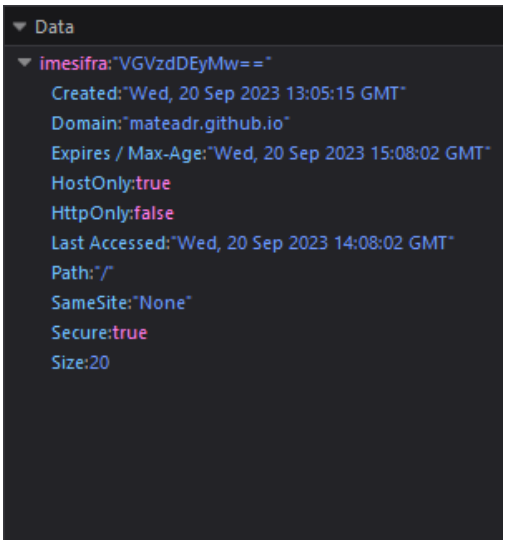
U ovom dijelu ćemo pokazati kako se sve to odvija pomoću kolačića. Slike koje prikazuju korištenje aplikacije i kôd vlastito snimljene iz izrađenog projekta te zbog toga nemaju reference.



Slika 3.2 Prikaz dijela aplikacije gdje korisnik sprema zadatke te ih „čekira“)

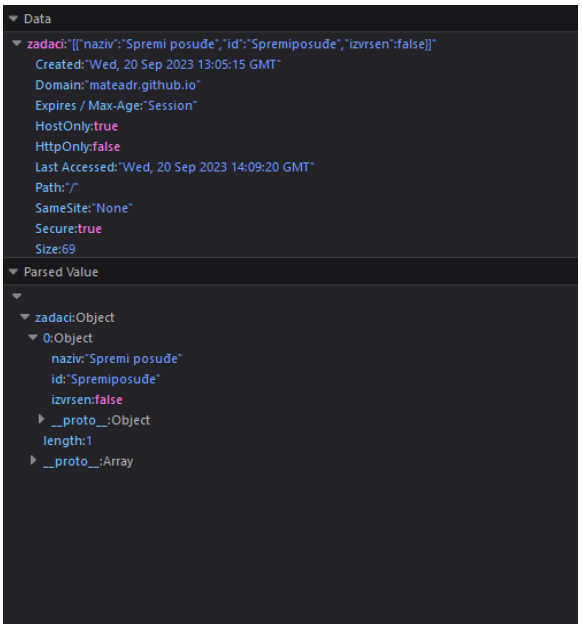
3.2. Rad aplikacije

Rad aplikacije je vrlo jednostavan. Pošto je bit na upotrebi kolačića, aplikacija nema servera za dohvat korisničkih podataka, nego kada korisnik stisne na dugme „Registriraj se“, on zapravo sprema svoje unesene podatke u kolačiće koji nisu trajni, nego će isteći nakon 1h nekorištenja. Treba napomenuti da će se kolačići za prijavu osvježiti ako se korisnik ponovno prijavi.



Slika 3.3 Primjer kolačića gdje su spremljeni korisnički podaci

Za razliku od kolačića za prijavu, kolačići u kojima su spremljeni korisnički zadaci su trajni i oni će ostati tu sve dok god se korisnik ponovno ne registrira. Zadaci su spremljeni u niz objekata, a svaki objekt se sastoji od imena zadatka i informacije je li taj zadatak izvršen ili ne.



Slika 3.4 Primjer kolačića gdje su spremljeni zadaci korisnika

3.3. Kod aplikacije

Slika 3.5 Cookie Consent Banner

Slika 3.6 Izgled kolačića gdje je spremljen korisnikov odabir

Kad se pokrene aplikacija, na dnu prozora se vidi takozvani „cookie consent banner“, banner u kojem korisnik može izabrati koje kolačiće prihvaća, a koje ne. Takvi kolačići uglavnom služe za praćenje korisnikovih akcija. Ti kolačići se zatim mogu prosljeđivati nekome koga zanimaju korisnikove akcije.

```
document.addEventListener("DOMContentLoaded", () => {
  // Provjera da li su prihvaćeni kolačići
  if (dohvatiKolacice("kolacici") != "") {
    let kolacici = dohvatiKolacice("kolacici");
    kolacici = JSON.parse(kolacici);
    if (kolacici.dugmestisnuto === false) {
      console.log("Dugme nije stisnuto, prikazuje se prozor za
kolacice");
      document.querySelector("#kolacici").style.display =
"flex";
    }
  } else {
    let expires = new Date();
    expires.setTime(expires.getTime() + 365 * 24 * 60 * 60 *
1000); // Traju 1 godinu
    let objektKolacica = {
      dugmestisnuto: false,
      obavezni: true,
      statisticki: false,
      marketinski: false,
    }; // Pocetni kolacici
    console.log(
      "Funkcionalni kolačići su automatski prihvaćeni za rad
aplikacije."
    );
    document.cookie = `kolacici=${JSON.stringify(
      objektKolacica

)};expires=${expires.toUTCString()};path=/;SameSite=None;Secure
`;
    document.querySelector("#kolacici").style.display = "flex";
  }
```

```
  // Provjera da li su kolačići za prijavu istekli, ako jesu
onda se očistu
  if (document.cookie.indexOf("imesifra") < 0) {
    // Ako je kolačić istekao, brišu se svi kolačići osim onih
koji sadržavaju koje je kolačiće korisnik prihvatio
    console.log("Kolačići za prijavu su istekli.");
    spremiKolacice("zadaci", JSON.stringify([])); // Ocisti
postojece zadatke
    spremiTrenutneKolace("imesifra", "");
  }
});
```

Kôd 1 Kôd za "cookie consent banner"

U Kôdu 1 vidi se sav kod potreban za „cookie consent banner“. Prvi događaj se izvršava čim se stranica, tj. DOM (Document Object Model) učita. Tada se provjerava je li se taj isti banner prihvatio, ako nije, pravi se novi kolačić u kojem se nalazi objekt koji sadrži sva svojstva za sami „cookie consent banner“. Zadano je da se samo kolačići za funkcionalnost prihvaćaju automatski, a ostali kolačići su opcionalni i oni su zadano postavljeni da nisu prihvaćeni. Sve dok korisnik ne pritisne na dugme „Prihvati“, „cookie consent banner“ će biti vidljiv. Drugi „event listener“ čeka da se to dugme „Prihvati“ pritisne i onda će ažurirati kolačić u kojem se nalazi objekt za „cookie consent banner“. Ovaj kolačić je napravljen da istekne za godinu dana od njegovog stvaranja, odnosno ažuriranja. Ovaj kod prikazuje i provjeru za kolačić koji sadrži korisničko ime i lozinku koja provjerava je li taj kolačić istekao. Ako je istekao, obrisat će se svi kolačići vezani uz korisnika.

```
document.querySelector("#Registracija").addEventListener("click
", (e) => {
  e.preventDefault();
  let korisnickoIme =
document.querySelector("#KorisnickoIme").value;
  let sifra = document.querySelector("#Sifra").value;
  if (korisnickoIme != "" && sifra != "") {
    spremiKolacice("zadaci", JSON.stringify([])); // Ocisti
postojece zadatke
    spremiTrenutneKolace("imesifra", btoa(korisnickoIme +
sifra));
    alert("Registracija završena.");
```

```
    document.querySelector("#Aplikacija").style.display =
"flex";
    document.querySelector("#Prijava").style.display = "none";
  }
});
```

Kôd 2 Kôd za registraciju

U Kôdu 2 vidi se kod za registraciju. Na dugme „Registriraj se“ dodali smo „event listener“ koji čeka da se ovo dugme pritisne. Kada se dugme pritisne, prvo vršimo provjeru imamo li korisničko ime i lozinku unesenu. Nakon te provjere očistit ćemo kolačiće gdje se nalaze već postojeći zadaci (ako ih ima) i onda spremamo korisničko ime i lozinku kao jednu riječ enkriptiranu u base64 formatu zbog sigurnosti. Nakon toga skrivamo formu za registraciju/prijavu i prikazujemo glavni dio aplikacije u kojoj korisnik može dodavati, „čekirati“ i brisati zadatke.

```
document.querySelector("button#Prijava").addEventListener("clic
k", (e) => {
  e.preventDefault();
  let korisnickoIme =
document.querySelector("#KorisnickoIme").value;
  let sifra = document.querySelector("#Sifra").value;
  if (korisnickoIme != "" && sifra != "") {
    let korisnickoImeKolaci =
atob(dohvatiKolacice("imesifra"));
    if (korisnickoImeKolaci != "") {
      if (korisnickoImeKolaci == korisnickoIme + sifra) {
        spremiTrenutneKolace("imesifra", btoa(korisnickoIme +
sifra));
        prikaziZadatke();
        document.querySelector("#Aplikacija").style.display =
"flex";
        document.querySelector("#Prijava").style.display =
"none";
        document.querySelector("p#greska").style.display =
"none";
      } else {
        document.querySelector("p#greska").style.display =
"flex";
      }
```

```
    }
  }
});
```

Kôd 3 Kôd za prijavu

U Kôdu 3 se vidi kod za prijavu. Radi slično kao i registracija, samo što ovdje još provjeravamo je li unesena kombinacija korisničkog imena i lozinke dobra. Ako je, onda osvježava trajanje kolačića, učitavaju se zadaci, te se skriva forma za registraciju/prijavu i prikazuje se aplikacija.

```
function spremiTrenutneKolace(imeKolaca, vrijednost) {
  let vrijemeTrajanja = new Date();
  vrijemeTrajanja.setTime(vrijemeTrajanja.getTime() + 60 * 60 *
1000); // 60 minuta
  document.cookie =
    imeKolaca +
    "=" +
    vrijednost +
    "; expires=" +
    vrijemeTrajanja.toUTCString() +
    "; path=/; SameSite=None; Secure";
}
```

Kôd 4 Kôd za spremanje trenutnih, privremenih kolačića kao što su enkriptirano korisničko ime i lozinka

U Kôdu 4 imamo funkciju koja prikazuje kako se enkriptirani kolačić koji sadrži korisničko ime i lozinku sprema. Dohvaćamo datum i vrijeme koje je za 1h unaprijed pomaknuto od trenutnog vremena pokretanja ove funkcije i zatim spremamo taj kolačić sa dohvaćenim vremenom kad će taj isti kolačić isteći.

```
function prikaziZadatke() {
  let listaZadataka = document.querySelector("#listaZadataka");
  listaZadataka.innerHTML = "";
  let zadaci = dohvatiKolacice("zadaci");
  zadaci = zadaci ? JSON.parse(zadaci) : [];
  if (zadaci.length > 0) {
    zadaci.forEach((zadatak, index) => {
      let noviZadatak = document.createElement("li");
```

```
      noviZadatak.setAttribute("data-value", zadatak.id);
      noviZadatak.innerHTML = `<input id="${zadatak.id}"
type="checkbox"> ${zadatak.naziv} <button
id="brisiZadatak">Brisi</button>`;
      noviZadatak.querySelector(`#${zadatak.id}`).checked =
zadatak.izvrsen;
      listaZadataka.appendChild(noviZadatak);
      noviZadatak
        .querySelector(`#${zadatak.id}`)
        .addEventListener("change", (e) => {
          let zadaci = dohvatiKolacice("zadaci");
          zadaci = zadaci ? JSON.parse(zadaci) : [];
          zadaci.forEach((Zadatak) => {
            if (Zadatak.naziv == zadatak.naziv) {
              Zadatak.izvrsen = e.target.checked;
            }
          });
          spremiKolacice("zadaci", JSON.stringify(zadaci));
        });
    });
    document.querySelectorAll("#brisiZadatak").forEach((brisi)
=> {
      brisi.addEventListener("click", () => {
        let postojeciZadaci = dohvatiKolacice("zadaci");
        let Zadaci = postojeciZadaci ?
JSON.parse(postojeciZadaci) : [];
        Zadaci = Zadaci.filter(
          (zadatak) =>
            zadatak.id !== brisi.parentNode.getAttribute("data-
value")
        );
        spremiKolacice("zadaci", JSON.stringify(Zadaci));
        prikaziZadatke();
      });
    });
  }
}
```

Kôd 5 Kôd za prikaz zadataka

U Kôdu 5 je jednostavna funkcija pomoću koje prikazujemo zadatke od korisnika. Dohvaćamo sve zadatke iz kolačića i za svaki zadatak radimo posebni element koji dodajemo u listu na stranici. Svaki element ima svoj „Checkbox“ i dugme „Briši“. Za svaki element moramo dodati 2 „event listenera“. Prvi služi za „Checkbox“, a on čeka na promjenu stanja tog „Checkboxa“. Kada se promijeni stanje „Checkboxa“, taj „event listener“ se aktivira i dohvaćamo kolačiće u kojima se nalazi niz svih zadataka. Zatim prolazimo kroz taj niz i kada naiđemo na zadatak kojem pripada taj „Checkbox“, promijenit ćemo mu atribut „izvrsen“ u vrijednost tog „Checkboxa“ na true ili false, zavisi o stanju tog „Checkbox-a“. Drugi služi za dugmad za brisanje. Dohvaćamo svako dugme na stranici i postavljamo „event listener“ koji čeka da se to dugme pritisne. Kada se neko dugme pritisne, dohvatit ćemo kolačić u kojem se nalazi niz zadataka i iz tog niza ćemo izbrisati zadatak kojemu pripada ovo dugme za brisanje i zatim ćemo novi niz zadataka spremiti u kolačiće i ponovno prikazati zadatke, tj. osvježiti postojeći prikaz novim, ažuriranim rasporedom zadataka.

```
function generirajID(id, niz) {
  let brojac = 1;
  let jedinstveniID = `${id}${brojac}`;
  console.log(niz);
  while (zauzetID(jedinstveniID,niz)) {
    jedinstveniID = `${id}${brojac}`;
    brojac++;
  }
  return jedinstveniID;
}
function zauzetID(id, niz) {
  for (let i = 0; i < niz.length; i++) {
    if (niz[i].id === id) {
      return true;
    }
  }
  return false;
}
```

Kôd 6 Prikaz funkcija za generiranje ID-a zadatka

Funkcije generirajID i zauzetID nam omogućavaju generiranje jedinstvenih ID-ova za zadatke. Funkcija generirajID sadrži brojač i onda sa funkcijom zauzetID provjeravamo

postoji li ID sa tom vrijednošću brojača. Ako postoji, onda povećavamo vrijednost brojača i opet provjeravamo i tako sve dok ne pronađemo slobodan ID (Kôd 6).

```
function spremiKolacice(imeKolaca, vrijednost) {
  document.cookie = imeKolaca + "=" + vrijednost + "; path=/;
SameSite=None; Secure";
                                    }
```

Kôd 7 Prikaz funkcije za spremanje trajnih kolačića

Ova funkcija u Kôdu 7 je veoma slična funkciji za spremanje privremenih kolačića, samo što ovdje nemamo vrijeme isteka kolačića, nego će oni biti spremljeni sve dok se ne izbrišu.

```
function dohvatiKolacice(imeKolaca) {
  let kolaci = document.cookie.split("; ");
  for (let i = 0; i < kolaci.length; i++) {
    let kolac = kolaci[i].split("=");
    if (kolac[0] === imeKolaca) {
      return kolac[1];
    }
  }
  return "";
}
```

Kôd 8 Prikaz funkcije za dohvaćanje kolačića

Funkcija u Kôdu 8 služi za dohvaćanje kolačića po njihovom imenu. Svi kolačići su odvojeni znakom jednakosti („=“). Sa tim znanjem ih možemo lako dohvatiti tako što dobavimo sve kolačiće i prvo ih podijelimo u niz gdje su prvo podijeljeni znakom „točka-zarez“ (;). Zatim, svaki element možemo podijeliti u novi „mini“ niz, ali ovdje umjesto znaka „točka-zarez“, koristimo znak jednakosti. Ako je prvi element u tom nizu jednak imenu kolačića, funkcija će vratiti njegovu vrijednost, odnosno drugi element u tom novom „mini“ nizu. Ako nismo našli kolačiće, funkcija će vratiti samo prazni string.

```
document.querySelector("#dodaj").addEventListener("click", ()
=> {
  let inputZadatak =
document.querySelector("#unesiZadatak").value.trim();
  if (inputZadatak != "") {
```

```
    let postojeciZadaci = dohvatiKolacice("zadaci");
    let zadaci = postojeciZadaci ? JSON.parse(postojeciZadaci)
: [];
    let postojeciZadatak = false;
    for (let index = 0; index < zadaci.length; index++) {
      if (zadaci[index].naziv == inputZadatak) {
        postojeciZadatak = true;
        break;
      }
    }
    if (postojeciZadatak === false) {
      let noviZadatak = {
        naziv: inputZadatak,
        id: generirajID("Zadatak", zadaci),
        izvrsen: false,
      };
      zadaci.push(noviZadatak);
      spremiKolacice("zadaci", JSON.stringify(zadaci));
      document.querySelector("#unesiZadatak").value = "";
      prikaziZadatke();
    } else {
      alert("Zadatak vec postoji");
      document.querySelector("#unesiZadatak").value = "";
    }
  }
});
```

Kôd 9 Kôd za dodavanje zadataka

Za kraj imamo dodavanje novih zadataka. Event listener u Kôdu 9 čeka da se pritisne dugme „Dodaj zadatak“. Kada se to dugme pritisne, vrši se provjera je li unos ispravan. U slučaju ove aplikacije, bitno je da unos zadatka nije prazan. Ako je unos ispravan, dohvaćamo niz zadataka iz kolačića te provjeravamo je li taj zadatak već unesen. Ako zadatak nije unesen, pravimo novi objekt u kojem imamo naziv zadatka, njegov ID, što je zapravo tekst zadatka kao jedan string i boolean izvršen koji nam govori je li „Checkbox“ zadatka „čekiran“. Zatim ga dodajemo u dohvaćeni niz zadataka te spremamo novi niz zadataka u kolačiće i ponovno prikazujemo zadatke.

ZAKLJUČAK

HTTP je protokol koji omogućuje komunikaciju između klijenata i poslužitelja putem interneta. Klijenti šalju zahtjeve, a poslužitelji šalju odgovore kako bi razmijenili informacije. Web aplikacije se temelje na različitim arhitekturama, a jedna od najpoznatijih je klijent-poslužitelj. Za izradu struktura web aplikacija obično se koristi HTML, jezik za označavanje teksta. CSS se koristi za stilizaciju web aplikacija, dok se dinamička svojstva, poput oporavka izvornog koda ili implementacije kolačića, opisuju pomoću programskog jezika JavaScript. Kolačići su mali podaci koje web stranice pohranjuju na uređaju korisnika. Prvobitno su ih 1994. godine koristili za stvaranje virtualnih košarica za kupovinu, ali danas imaju mnogo širu primjenu. Kolačići se klasificiraju prema svojim svojstvima, kao što su izvor, trajanje i namjena. Kolačići na temelju izvora mogu biti kolačići prve strane (web stranica koju korisnik trenutno posjećuje) ili kolačići treće strane (web stranica različita od trenutne). Trajne kolačiće čuva preglednik korisnika i nakon zatvaranja preglednika, dok se sesijski kolačići brišu nakon zatvaranja preglednika. Kolačići se također klasificiraju prema namjeni, kao što su kolačići izvedbe, nužni, funkcionalni i oglašivački kolačići. Postoje i posebne vrste kolačića, kao što su superkolačići, zombi, Flash i sigurni kolačići, koji imaju specifične karakteristike. Većina kolačića može pripadati više različitim skupinama, a ne samo jednoj. U praktičnom primjeru prikazano je kako se kolačići koriste u To-Do aplikaciji, gdje su korišteni kolačići prve strane koji spadaju u kategoriju sesijskih i funkcionalnih kolačića. Važno je napomenuti da neki kolačići nisu bezopasni, pa je važno pažljivo birati koje kolačiće prihvaćamo i razumjeti zašto su potrebni.

LITERATURA

[1] D. Gourley, B. Totty, M. Sayer, A. Aggarwal, S. Reddy, HTTP: The Definitive Guide

[2] HTTP, https://en.wikipedia.org/wiki/HTTP

[3] HTTPS, https://en.wikipedia.org/wiki/HTTPS

[4] HTTP (Hypertext Transfer Protocol), https://www.javatpoint.com/http

[5] HTTP (Hypertext Transfer Protocol), https://www.techtarget.com/whatis/definition/HTTP-Hypertext-Transfer-Protocol

[6] How HTTP request and response works, https://bytesofgigabytes.com/networking/how-http-request-and-response-works/

[7] G. Zaharija, Arhitektura web aplikacija

[8] List of HTTP header fields, https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

[9] HTTP responses, https://www.ibm.com/docs/en/cics-ts/5.2?topic=protocol-http-responses

[10] The Guide to Web Application Architecture, https://jelvix.com/blog/guide-to-web-application-architecture

[11] Web Application Architecture: The Latest Guide 2022, https://www.clickittech.com/devops/web-application-architecture/

[12] Client – server model, https://en.wikipedia.org/wiki/Client%E2%80%93server_model

[13] Two-Tier Architecture, https://www.techopedia.com/definition/467/two-tier-architecture

[14] What is three-tier architecture?, https://www.ibm.com/topics/three-tier-architecture

[15] HTML basics, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

[16] G. Zaharija, HTML

[17] What is CSS?, https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

[18] G. Zaharija, Oblikovanje web stranica

[19] JavaScript, https://developer.mozilla.org/en-US/docs/Web/JavaScript

[20] G. Zaharija, Uvod u JavaScript

[21] HTTP cookie, https://en.wikipedia.org/wiki/HTTP_cookie

[22] David M. Kristol, HTTP Cookies: Standards, Privacy, and Politics

[23] All About Internet Cookies, https://www.cookieyes.com/blog/internet-cookies/

[24] What are first party and third party cookies?, https://devowl.io/2021/first-party-cookies-third-party-cookies-definition/

[25] What are session cookies? Do they need consent?, https://www.cookieyes.com/blog/session-cookies/

[26] Session Cookies vs Persistent Cookies: Understanding the Differences, https://secureprivacy.ai/blog/session-cookies-vs-persistent-cookies

[27] What are persistent cookies?, https://www.cookieyes.com/knowledge-base/cookies-101/what-are-persistent-cookies/

[28] All You Need to Know About Third-Party Cookies, https://cookie-script.com/all-you-need-to-know-about-third-party-cookies.html

[29] What are Performance Cookies?, https://www.cookiepro.com/knowledge/what-are-performance-cookies/

[30] Cookie Consent Exemptions: Strictly Necessary Cookies, https://www.cookieyes.com/blog/cookie-consent-exemption-for-strictly-necessary-cookies/

[31] What are Functionality Cookies?, https://cookie-script.com/blog/functionality-cookies

[32] Zombie cookie, https://en.wikipedia.org/wiki/Zombie_cookie

[33] What is a Flash Cookie?, https://www.cookiepro.com/knowledge/what-is-a-flash-cookie/

[34] What are Supercookies and How Can They Affect Your Online Privacy?, https://www.privacysense.net/terms/supercookies/

[35] Evercookie, https://en.wikipedia.org/wiki/Evercookie

[36] Secure cookie, https://en.wikipedia.org/wiki/Secure_cookie

[37] What all Developers need to know about: Cookie Security, https://techblog.topdesk.com/security/cookie-security/

POPIS SLIKA