



LV3 - JavaScript

Web programiranje

2024./2025.

1

Cilj laboratorijske vježbe

Cilj ove laboratorijske vježbe je upoznati se s osnovama razvoja web aplikacija koje se poslužuju sa strane poslužitelja (engl. *server-side*) korištenjem okruženja `Node.js` i okvira `Express`. Za razliku od prethodnih vježbi, gdje su se HTML i CSS datoteke otvarale lokalno izravno u pregledniku, u ovoj vježbi web stranica se pokreće pomoću poslužiteljske aplikacije koja služi kao "server" i klijentu dostavlja potrebne resurse.

Važno je naglasiti da se u ovoj vježbi sva poslovna logika, uključujući dohvaćanje, prikaz i manipulaciju podacima, odvija isključivo na strani preglednika korištenjem JavaScripta. Drugim riječima, **JavaScript se koristi samo na klijentskoj strani**, dok se **Node.js i Express koriste samo za posluživanje statičkog sadržaja (HTML, CSS, slike, JavaScript) te eventualno podataka u obliku .json ili .csv**.

Dodjeljene su dvije vrste ulaznih podataka: tablica podataka o temperaturama i tablica podataka o filmovima. Ovisno o vrsti podatkovne datoteke, izrađuju se funkcionalnosti kao što su:

- dinamičkog prikaza i skrivanja dijelova sadržaja;
- filtriranja i pretraživanja podataka na strani preglednika, prema odabranim kriterijima (npr. žanr, temperatura, godina)
- sortiranje podataka prema određenim stupcima;
- reagiranja na korisničke akcije (klikovi, unos podataka);
- dinamički prikaz dodatnih informacija o pojedinom retku;

Dodatno je objašnjeno za svaki zadatak posebno što i kako je potrebno prikazati rješenje.

Zadaci za laboratorijsku vježbu:

1. zad: Dohvat i prikaz podataka [3.1](#)
2. zad: Filtriranje i prikazivanje [3.2](#)
3. zad: Interaktivno upravljanje sadržajem [3.4](#)
4. Zad: Dodatno [3.5](#)

Cilj je da se svi ovi elementi izvedu korištenjem JavaScripta u pregledniku, bez potrebe za serverskom obradom podataka. Podatke je potrebno dohvatiti pomoću JavaScript funkcija (npr. `fetch`) te ih ugraditi u DOM pomoću manipulacije elementima stranice.

Završnu verziju stranice potrebno je postaviti na odgovarajući servis (npr. Railway, Render, Cyclic) te poveznicu navesti u `README.md` dokument unutar pripadajućeg GitHub repozitorija. U repozitoriju također trebaju biti komentirani dijelovi koda i svi dodatni resursi koji su korišteni pri izradi rješenja.

U izradi web aplikacije koristiti jednu od sljedećih baza podataka za prikaz podataka, ovisno o tome koju ste koristili pri izradi rješenja zadataka na prvim laboratorijskim vježbama:

- **Podaci o vremenu**
- **Podaci o filmovima.**

Integracija rješenja na postojeću web stranicu

Sva rješenja zadataka (1., 2. i 3. zadatak) potrebno je integrirati na već kreiranu početnu (`index.html`) stranicu iz prethodnih laboratorijskih vježbi ili dodati novu html stranicu, ali povezati ju s već postojećom web stranicu.

Moguće na već postojećoj `index.html` stranici implementirati funkcionalnosti koje moraju biti vidljive i dostupne korisniku unutar iste web stranice, ispod postojećeg sadržaja (npr. ispod tablice prikazane u prvom zadatku).

Napomena: Nije dozvoljeno kreirati novu zasebnu stranicu za ove zadatke. Svi elementi (prikaz podataka, filtriranje, interaktivno upravljanje sadržajem) trebaju biti dio postojeće stranice, čime se osigurava kontinuitet u razvoju aplikacije.

Po završetku, potrebno je ažurirati projekt na odabranom hosting servisu (npr. Railway) kako bi sve funkcionalnosti bile dostupne nakon osvježavanja stranice i postaviti sve dokumente na github - LV3, dostupan link dan na Merlinu.

2

Uvod u JavaScript

JavaScript je dinamičan skriptni programski jezik visoke razine. JavaScript (JS) je interpretativnog tipa što znači da se JS kod direktno izvršava (JIT- Just in time) bez pretvorbe u strojni jezik (binarni). JS omogućuje interpretaciju klijentske strane logike unutar samog web preglednika. Prvobitno kreiran za web preglednike, JS se unutar zadnjih par godina popularizacijom okvira Node.js koristi i na serverskoj strani (backend), kao i u izradi desktop/mobilnih aplikacija (Electron, Phaser.js).

Osim što je definiran objektno orijentiranom paradigmom, JavaScript podržava i funkcionalno programiranje. U JavaScriptu sve je predstavljeno kao objekt. Nema razlike između vrsta objekata. Ono što razlikuje JavaScript od drugih objektno orijentiranih jezika (npr. Java) je to što nasljeđivanje funkcionira putem mehanizma prototipa, a svojstva i metode mogu se dinamički dodati bilo kojem objektu, dok u Javi objekti su podijeljeni u klase i instance, a nasljeđivanje se provodi putem hijerarhije klasa. Klase i instance ne mogu dinamički dodavati svojstva ili metode. Kako se JavaScript nalazi unutar web preglednika i na klijentskoj strani - nije moguće vršiti I/O radnje s klijentovog diska.

2.1 ECMAScript 2024 (ES15) – standard JavaScripta

JavaScript je standardiziran prema organizaciji **Ecma International**, globalnom tijelu za standardizaciju informacijskih i komunikacijskih sustava (ECMA je izvorno bila skraćenica za Europsko udruženje proizvođača računala). Cilj standardizacije je razvoj i održavanje međunarodnog programskog jezika temeljenog na JavaScriptu, kako bi programski kod bio dosljedan i interoperabilan u svim okruženjima koja podržavaju taj standard.

Standardizirana verzija JavaScripta naziva se **ECMAScript** i definira pravila, sintaksu i funkcionalnosti jezika. Svi suvremeni web preglednici i JavaScript okruženja implementiraju ECMAScript standard, što osigurava konzistentno ponašanje koda bez obzira na platformu.

ECMAScript standard se kontinuirano razvija i svake godine izlazi nova verzija s dodatnim mogućnostima i poboljšanjima. Trenutačna, 15. verzija, **ECMAScript 2024 (ES15)**, objavljena je u lipnju 2024. godine i donosi brojne nove značajke, uključujući metode za grupiranje podataka (`Object.groupBy`, `Map.groupBy`), poboljšanja rada s Promise objektima (`Promise.withResolvers`), naprednije operacije nad skupovima (nove metode za `Set`), te novu `/v` zastavicu za regularne izraze.

Prijašnje verzije, poput **ECMAScript 2015 (ES6)**, bile su prekretnica u razvoju jezika jer su uvele moderne koncepte poput klase, modula, `let` i `const` deklaracija, "arrow" funkcija, Promise objekata i mnogih drugih mogućnosti koje su omogućile razvoj velikih i složenih aplikacija.

Danas većina preglednika i JavaScript okruženja brzo implementira nove značajke iz najnovijih ECMAScript standarda, čime se osigurava brza dostupnost novih funkcionalnosti širokoj zajednici programera.

Više na:

- ECMAScript [W3Schools](#)
- JavaScript [Guide](#)
- [New features in es15](#)
- Before and After ES15: [examples](#)

3 Zadatak laboratorijske vježbe

U sklopu ove laboratorijske vježbe potrebno je izraditi web aplikaciju koja koristi JavaScript na klijentskoj strani za dohvaćanje i prikaz podataka o filmovima ili temperaturama. Podaci se dohvaćaju iz lokalne `.json` datoteke korištenjem funkcije `fetch`, a sadržaj se dinamički generira u pregledniku. Stranica se pokreće putem Express poslužitelja, no sva poslovna logika ostaje na klijentskoj strani.

Odrađuje se tri zadatka, ovisno o vrsti dodijeljenih podataka (filmovi ili temperatura). U nastavku su opisani zadaci koje treba implementirati. Kratki opis strukture projekta:

- `public/` – sadrži sve statičke datoteke koje se učitavaju na klijentskoj strani (HTML, CSS, JS, podatci).
- `index.html` – glavna stranica koja se prikazuje korisniku.

```

my-project/
├── public/
│   ├── index.html
│   ├── style.css
│   ├── script.js
│   ├── filmovi.json
│   ├── filmovi.csv
│   ├── vrijeme.json
│   └── vrijeme.csv
├── server.js
└── package.json

```

Slika 1: Struktura projekta bez korištenja EJS-a

- `script.js` – klijentski JavaScript koji dohvaća i prikazuje podatke.
- `filmovi.csv` `filmovi.json` `vrijeme.csv`, `vrijeme.json` – lokalna datoteka s podacima o filmovima, vremenu.
- `server.js` – jednostavan Express poslužitelj koji omogućuje dohvat datoteka iz mape `public`.
- `package.json` – definira ovisnosti projekta i osnovne informacije.

3.1 Zadatak 1. Dohvat i prikaz podataka

Pomoću JavaScripta s klijentske strane potrebno je dohvatiti sadržaj datoteke `movies.csv` ili `temperature.csv` i prikazati ga u pregledniku. Sadržaj treba prikazati u tabličnom obliku, a svaki redak mora sadržavati relevantne informacije:

1. naslov filma, žanr, godina, trajanje, ocjena;
2. ID, temperaatura, sezona, lokacija, vrsta vremena

Stranica se treba automatski popuniti podacima pri učitavanju, bez potrebe za osvežavanjem ili dodatnim unosom.

Primjer 3.1. Primjer sadržaja `filmovi.csv`

```

title,year,genres,duration,rating,directors
Inception,2010,"Action;Sci-Fi",148,8.8,"Christopher Nolan"
The Matrix,1999,"Action;Sci-Fi",136,8.7,"Lana Wachowski;Lilly
Wachowski"

```

U nastavku predložka dani su primjeri koda za kreiranje pojedinih dijelova zadatka, a rješenja se prikazuju na primjeru dokumenta s filmskim podacima. Prilikom izrade zadatka potrebno je prilagoditi kod prema strukturi podataka koje koristite. Također, vodite računa o vizualnom izgledu stranice te uključite odgovarajuću `.css` datoteku za stilizaciju tablice.

3.1.1 Parsiranje CSV datoteke

Prvo je potrebno parsirati dokument s podacima, što je moguće učiniti na dva načina. Jedan pristup je parsiranje izravno unutar datoteke `scripts.js`, no taj pristup nije optimalan jer se kod za parsiranje izvršava prilikom svakog pokretanja skripte.

Koristi se biblioteka PapaParse da bi "pročitao" CSV datoteku i pretvorio je u strukturirani objekt – niz redaka gdje svaki redak postaje objekt. To se naziva parsiranje. Dobiveni rezultat spremamo u varijablu `rezultat`. Kako `Papa.parse(...)` vraća objekt koji sadrži polje `data` – niz objekata gdje svaki objekt predstavlja jedan redak iz CSV datoteke. Odnosno predstavlja niz (array) objekata, a Svaki objekt u tom nizu odgovara jednom retku iz CSV dokumenta, odnosno jednom filmu.

Taj dio koda izgleda ovako:

```
fetch('filmtv_movies.csv')
  .then(res => res.text())
  .then(csv => {
    const rezultat = Papa.parse(csv, {
      header: true,
      skipEmptyLines: true
    });
```

Uključivanje vanjske biblioteke za parsiranje CSV datoteka u HTML dokumentu

Za potrebe parsiranja CSV datoteka u JavaScriptu, koristi se vanjska biblioteka **PapaParse**. Ova biblioteka omogućuje jednostavno i učinkovito učitavanje te obradu CSV podataka na klijentskoj strani bez potrebe za dodatnim pozadinskim (backend) rješenjima.

Kako bi omogućili korištenje ove biblioteke u projektu, potrebno ju je **uključiti unutar HTML dokumenta** pomoću `<script>` taga. Preporuka je uključivanje vanjskih biblioteka u `<head>` dijelu HTML dokumenta kako bi bile dostupne prije izvršavanja JavaScript koda aplikacije (npr. `scripts.js`).

Primjer uključivanja PapaParse biblioteke:

Primjer 1: Uključivanje PapaParse biblioteke

```
<head>
...
<script src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.4.1/papaparse.min.js"></script>
</head>
```

Važno! Potrebno je osigurati da se ova biblioteka učita prije skripte (npr. `scripts.js`) koja koristi funkciju `Papa.parse()`. Zbog toga se `<script>` tag za PapaParse stavlja iznad skripti ili unutar `<head>` sekcije, dok se skripte najčešće pozicioniraju na kraju `<body>` dijela dokumenta.

Primjer 2: Struktura HTML dokumenta s ispravnim redoslijedom učitavanja

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Virtualna Videoteka</title>
```

```

<script src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse
  /5.4.1/papaparse.min.js"></script>
</head>
<body>
  <!-- Sadržaj stranice -->

  <!--skripta -->
  <script src="scripts.js"></script>
</body>
</html>

```

Na ovaj način se osigurava da će funkcionalnosti za parsiranje CSV datoteka biti dostupne u trenutku kada ih vaš kod zatreba.

3.1.2 Obrada i priprema podataka za prikaz

Nakon parsiranja sirovih podataka iz CSV-a pomoću biblioteke `PapaParse`, potrebno je dodatno obraditi svaki redak kako bi podaci bili u pravilnom i korisnom formatu za prikaz i daljnju obradu. Međutim, sve vrijednosti koje `Papa.parse(...)` vraća kao objekt koji sadrži polje `data`, u tom trenutku još uvijek su stringovi, uključujući i numeričke vrijednosti poput godine, trajanja ili broja glasova.

Primjer 3: Primjer dohvaćanja podataka iz CSV-a

```

const rezultat = Papa.parse(csv, { header: true,
  skipEmptyLines: true });

```

Kako bi se pripremili strukturirani objekti s pravilnim tipovima podataka, odnosno za obradu svakog pojedinačnog filma koristi se metoda `.map()`. Ova metoda prolazi kroz svaki element niza `rezultat.data` te omogućuje kreiranje novog niza s transformiranim podacima. U ovom slučaju, unutar `map()` funkcije koristi se lokalna varijabla `film`, koja pri svakom prolasku predstavlja trenutni objekt (film) iz niza.

```

const filmovi = rezultat.data.map(film => ({
  title: film.title,
  year: Number(film.year),
  genre: film.genre,
  duration: Number(film.duration),
  country: film.country?.split(',').map(c => c.trim()) ||
    [],
  total_votes: Number(film.total_votes)
}));

```

U ovoj obradi:

- `year`, `duration` i `total_votes` pretvaraju se iz stringova u numeričke tipove pomoću funkcije `Number()`.
- `country` se razdvaja u niz država ako ih ima više (odvojenih zarezom), uz uklanjanje viška razmaka.

Na ovaj način dobiva se novi niz objekata pod nazivom `filmovi`, koji je spreman za prikaz u tablici, filtriranje, sortiranje i druge funkcionalnosti korisničkog sučelja.

Odnosno, novi niz objekata pod nazivom `filmovi` sadrži strukturirane i pravilno formatirane podatke spremne za prikaz korisniku ili daljnju obradu.

Ukratko:

- `rezultat.data` je niz objekata gdje svaki objekt predstavlja jedan film.
- Metoda `map()` prolazi kroz svaki od tih objekata.
- Varijabla `film` je parametar anonimne funkcije unutar `map()` metode i odnosi se na trenutno obrađivani film.
- Tijekom obrade:
 - Numeričke vrijednosti poput godine (`year`), trajanja (`duration`) i broja glasova (`total_votes`) pretvaraju se iz stringova u brojčane tipove pomoću funkcije `Number()`.
 - Polje `country` se, ako sadrži više država, dijeli u niz pomoću funkcije `split(',')` te se uklanjaju suvišni razmaci s metodom `trim()`.

Na taj način kreira se novi niz pod nazivom `filmovi`,

U nastavku slijedi prikaz načina kako se obrađeni podaci prikazuju u HTML tablici pomoću JavaScript-a.

Rad s CSV i JSON datotekama u web aplikacijama

U ovoj vježbi koristimo podatke pohranjene u obliku CSV ili JSON datoteka. Važno je razumjeti razliku između ova dva formata te mogućnosti i ograničenja prilikom njihove uporabe na klijentskoj strani (u pregledniku).

3.1.3 Učitavanje i manipulacija podacima

- **CSV (Comma-Separated Values)** je tekstualni format u kojem su podaci zapisani kao nizovi stringova odvojeni zarezima. Za obradu CSV datoteka u JavaScript-u koristi se biblioteka `PapaParse`, kojom je moguće parsirati podatke u strukturirani oblik (niz objekata).
- **JSON (JavaScript Object Notation)** je format koji prirodno odgovara JavaScript jeziku. Podaci su odmah dostupni kao objekti i nizovi s ispravnim tipovima podataka (brojevi, stringovi, nizovi itd.).

3.1.4 Dodavanje novih elemenata

Bez obzira koristi li se CSV ili JSON, na klijentskoj strani moguće je dodavati nove elemente u podatke **samo privremeno** — dok je aplikacija aktivna u pregledniku. Nakon osvježavanja stranice, podaci se ponovno učitavaju iz izvorne datoteke.

Primjer: Dodavanje novog filma u niz podataka učitanih iz CSV-a ili JSON-a:

```
podaci.push({
  title: "Novi_Film",
  year: 2024,
  genre: "Drama",
  duration: 110,
  country: "Italy",
  total_votes: 0
});
```

Ova promjena vrijedi samo dok je stranica otvorena.

3.1.5 Trajno spremanje podataka

Za trajno spremanje izmjena (dodavanje, brisanje, uređivanje podataka) potrebno je koristiti serversku logiku (npr. `Node.js`) ili bazu podataka. U ovoj vježbi radimo isključivo na klijentskoj strani, stoga trajne izmjene izvornih datoteka nisu moguće.

Kod CSV datoteka moguće je generirati novu verziju datoteke korištenjem `Papa.unparse()` te ponuditi korisniku preuzimanje (download) nove datoteke.

Primjeri dohvaćanja podataka

Primjer 4: Podaci iz .csv dokumenta

```
fetch('filmovi.csv')
.then(res => res.text())
```



```

.then(csv => {
    const rezultat = Papa.parse(csv, {
        header: true,
        skipEmptyLines: true
    });
    console.log(rezultat.data);
});

```

Primjer 5: Podaci iz .json dokumenta"

```

fetch('filmovi.json')
.then(res => res.json())
.then(data => {
    console.log(data);
});

```

3.1.6 Korištenje LocalStorage za privremeno spremanje

Dohvaćanje JSON datoteke: Ako želite simulirati trajno spremanje podataka na klijentskoj strani, moguće je koristiti `LocalStorage`. Ovo omogućuje da podaci ostanu pohranjeni i nakon što korisnik zatvori ili osvježi stranicu.

```

// Spremanje
localStorage.setItem('plan', JSON.stringify(podaci));

// Dohvaćanje
const plan = JSON.parse(localStorage.getItem('plan'));

```

Napomena: `LocalStorage` je ograničen na otprilike 5MB prostora i nije pogodan za velike količine podataka.

3.1.7 Usporedba CSV i JSON formata

Primjer spremanja i dohvaćanja:

Značajka	CSV	JSON
Dodavanje podataka	Privremeno	Privremeno
Trajno spremanje	Nije moguće bez servera	Nije moguće bez servera
Generiranje nove datoteke	Jednostavno (<code>Papa.unparse</code>)	Složenije
Brzina učitavanja	Sporije (parsiranje)	Brže
Struktura podataka	Sve su stringovi	Podržava tipove podataka

3.1.8 Rad s većim skupovima podataka

Kod većih datoteka (stotine ili tisuće redaka) preporučuje se korištenje JSON formata radi bolje optimizacije i bržeg učitavanja. Također, poželjno je implementirati tehnike poput:

- Paginacije (prikaz podataka po stranicama).
- Dinamičkog učitavanja (lazy loading).
- Filtriranja podataka prije prikaza.

3.1.9 Preporuka

Za manje aplikacije i potrebe demonstracije može se koristiti CSV format. Međutim, kada je cilj optimizirati performanse i pojednostaviti rad s podacima, preporučuje se konverzija u JSON format.

Napomena: Svaki put kada se koristi CSV datoteka, parsiranje se izvršava pri svakom osvježavanju stranice, što može usporiti aplikaciju kod većih datoteka.

3.1.10 Prikazivanje podataka

Nakon što su podaci parsirani i obrađeni, potrebno ih je prikazati korisniku u preglednom formatu. Za tu svrhu koristi se funkcija `prikaziTablicu(filmovi)`, koja dinamički generira HTML tablicu na temelju niza objekata. Funkcija omogućuje višestruko pozivanje, čime se sadržaj tablice može ažurirati ovisno o korisničkim akcijama poput filtriranja ili sortiranja.

Kako bi se korisniku odmah prikazao početni skup podataka, koristi se metoda `slice()`, kojom se iz cjelokupnog niza podataka izdvaja određeni broj elemenata. U sljedećem primjeru, dohvaća se prvih 150 filmova iz niza `filmovi`:

```
const prvih20 = filmovi.slice(0, 150);
prikaziTablicu(prvih20);
```

Objašnjenje koda:

- `filmovi.slice(0, 150)` — metoda `slice` vraća novi niz koji sadrži elemente od indeksa 0 do 149 (ukupno 150 elemenata). Na ovaj način sprječava se preopterećenje sučelja prikazom prevelike količine podataka odjednom.
- Rezultat se sprema u varijablu `prvih20` (naziv varijable može biti precizniji, npr. `prvih150`).
- Funkcija `prikaziTablicu(prvih20)` poziva se s ovim nizom i generira odgovarajuće redove unutar HTML tablice.

U slučaju da dođe do pogreške prilikom dohvaćanja ili parsiranja CSV datoteke, koristi se metoda `catch()`, kojom se hvata greška i ispisuje informativna poruka u konzolu preglednika:

```
.catch(err => {
    console.error('Greska_pri_dohvacanju_CSV-a:', err);
});
```

Ova struktura osigurava robusnost aplikacije, jer omogućuje detekciju potencijalnih problema pri radu s vanjskim datotekama.

Funkcija `prikaziTablicu()` omogućuje dinamičko i pregledno prikazivanje podataka iz datoteke, te služi kao osnova za dodatne funkcionalnosti poput sortiranja, filtriranja i reagiranja na korisničke akcije, kada će se prikazivati samo oni podaci koji zadovoljavaju postavljene kriterije.

```
function prikaziTablicu(filmovi) {
    const tbody = document.querySelector('#filmovi-tablica_tbody')
    ;
    tbody.innerHTML = ''; // ocisti ako postoji

    for (const film of filmovi) {
        const row = document.createElement('tr');
        row.innerHTML = `
        <td>${film.title}</td>
        <td>${film.year}</td>
        <td>${film.genre}</td>
```

```

        <td>${film.duration}</td>
        <td>${film.country.join(', ')}</td>
        <td>${film.total_votes}</td>
        `;
        tbody.appendChild(row);
    }
}

```

Opis funkcionalnosti:

- `document.querySelector(...)` dohvaća `<tbody>` dio tablice u koji se umeću redci.
- `tbody.innerHTML = ''` - briše se prethodni sadržaj kako bi se prikaz ažurirao.
- Za svaki film se kreira redak `<tr>` s ćelijama `<td>` za sve relevantne podatke.
- Ako je polje `country` niz (što bi trebalo biti), njegove se vrijednosti spajaju u jedan string odvojen zarezima.

Kako bi ova naredba ispravno funkcionirala, potrebno je u HTML dokumentu definirati tablicu s odgovarajućim identifikatorom, npr.:

```

<table id="filmovi-tablica">
  <thead>
    <tr>
      <th>Naslov</th>
      <th>Godina</th>
      <th>Zanr</th>
      ...
    </tr>
  </thead>
  <tbody>
  </tbody>
</table>

```

U ovom slučaju, u JavaScript kodu koristimo:

```

const tbody = document.querySelector('#filmovi-tablica_tbody')
;

```

Na ovaj način selektiramo upravo `<tbody>` unutar tablice koja ima atribut `id="filmovi-tablica"`.

3.1.11 Korištenje ternarnog operatora za formatiranje podataka

Prilikom izrade datoteke `scripts.js` važno je obratiti pažnju na usklađenost sa zaglavljem tablice te na način na koji su podaci pohranjeni u svakoj varijabli. Na primjer, za varijablu `country` u dokumentu s filmovima, jedan film može imati više država. Pravilna obrada ovog podatka omogućuje ispravan prikaz u tablici, kao što je prikazano na slici 2.

U tom slučaju u `script.js` je moguće staviti ternarni operator u ovisnosti za što želimo koristiti podatke prikazane u odgovarajućem stupcu.

Popis filmova2

Naslov	Godina	Žanr	Trajanje (min)	Država	Glasova
Across the Pacific	1942	Spy	97	United States	26
Hard Ticket to Hawaii	1987	Crime	96	United States	5
Rhino!	1964	Adventure	91	United States	3
Die Flusspiraten vom Mississippi	1964	Western	86	Germany	5
Agi Murad, il diavolo bianco	1959	Adventure	100	Italy, Yugoslavia	5
Agnes of God	1985	Drama	98	United States	51
Ai margini della metropoli	1953	Drama	95	Italy	13
Aida	1953	Musical	97	Italy	9
Air America	1990	Adventure	106	United States	58
Airport '77	1977	Action	110	United States	43
Baby on Board	1991	Comedy	100	United States	6
Akiko	1961	Comedy	99	Italy	9
Al bar dello sport	1983	Comedy	100	Italy	110

Galerija slika (aside)



Slika 2: Primjer rješenja prikaza podataka iz filmovi.csv

Primjena ternarnog operatora u obradi polja `country`

U JavaScriptu je moguće koristiti **ternarni operator** kao kraći zapis za jednostavne `if-else` uvjete. Ternarni operator ima sljedeću opću formu:

```
uvjet ? vrijednostAkoJeIstina : vrijednostAkoJeNeistina
```

Koristi se kada želimo izraziti odluku između dviju vrijednosti na osnovi jednog logičkog uvjeta, pri čemu se izražava unutar jedne linije koda. U kontekstu obrade CSV podataka, to može biti korisno za uvjetno parsiranje polja koja mogu sadržavati višestruke vrijednosti (npr. više država u polju `country`).

Primjer iz koda

```
country: film.country?.includes(',')
? film.country.split(',').map(c => c.trim()).join(',_')
: film.country
```

Analiza:

- `film.country?.includes(',')`
Predstavlja **uvjet** – provjerava sadrži li string zarez (što implicira više vrijednosti u jednom zapisu). Operator `?.` (optional chaining) osigurava da se provjera izvrši samo ako objekt `film.country` nije `null` niti `undefined`, čime se sprječava izbacivanje pogreške.
- `film.country.split(',').map(c => c.trim()).join(',_')`
Ovo je **vrijednost izraza ako je uvjet istinit**. Prvo se string razdvaja po zarezima (`split`), zatim se svaki dio reže od suvišnih razmaka (`trim`), i naposljetku se sve vrijednosti ponovno spajaju u jedan string s razmakom nakon zareza (`join`).
- `film.country`
Ako uvjet nije zadovoljen (dakle, radi se o samo jednoj državi), koristi se originalna vrijednost bez dodatne obrade.

Ovakav izraz omogućuje da se korisniku u prikazu dosljedno prikaže lista država u čitkom formatu, neovisno o tome je li u podacima navedena jedna ili više njih.

Ovaj oblik je posebno pogodan kada:

- Želimo postići čitljiv tekstualni prikaz podataka koji mogu sadržavati više vrijednosti
- Želimo kompaktniji i elegantniji zapis umjesto klasične `if-else` strukture
- Nema potrebe za daljnjom programskom obradom pojedinačnih vrijednosti

Napomena: Ako se planira raditi složenija obrada pojedinačnih stavki (npr. filtriranje po zemlji), tada je prikladnije koristiti **niz** (array) umjesto spojenog stringa.

U tom slučaju može se koristiti izraz:

```
country: film.country?.split(',').map(c => c.trim()) || []
```

koji kao rezultat uvijek daje niz stringova, neovisno o tome postoji li jedna ili više država u izvornom polju.

Primjeri primjene: prikaz u tablici vs. filtriranje po državi

U nastavku su prikazana dva konkretna primjera u kojima se koristi podatak `country` – jedan za prikaz podataka, a drugi za funkcionalnu obradu (filtriranje).

Primjer 1: prikaz u HTML tablici

Ako želimo samo prikazati podatke u HTML tablici, tada je dovoljno koristiti tekstualni oblik – dakle, rezultat ternarnog izraza je string.

```
<td>${film.country}</td>
```

U tom slučaju vrijednost `country` može biti definirana ovako:

```
country: film.country?.includes(',')
? film.country.split(',').map(c => c.trim()).join(', ')
: film.country
```

Rezultat će biti čitljiv tekst: npr. "United States, Canada", spreman za prikaz korisniku bez dodatne obrade.

Primjer 2: filtriranje po državi

Ako korisnik u sučelju odabire npr. `France`, tada želimo pronaći sve filmove koji uključuju tu državu, bez obzira je li navedena samostalno ili u kombinaciji s drugima. Tada je prikladnije koristiti pristup koji vraća niz:

```
country: film.country?.split(',').map(c => c.trim()) || []
```

Filtriranje se može provesti na sljedeći način:

```
const odabranaDrzava = 'France';

const rezultati = sviFilmovi.filter(film =>
  film.country.includes(odabranaDrzava)
);
```

U ovom primjeru:

- `film.country` je niz, pa metoda `includes` provjerava nalazi li se tražena država unutar njega
- dobivamo fleksibilan način obrade, jer možemo pretraživati i kombinacije kao što su "France, Italy" i slično

Zaključak

- Za potrebe **prikaza** podataka koristi se string koji je dobiven obradom pomoću ternarnog operatora i metode `join(' ', '')`.
- Za potrebe **obrade** i **filtriranja** podataka koristi se niz vrijednosti, dobiven razdvajanjem teksta metodom `split(' ', '')` i dodatnom obradom metodom `map(trim)`.

3.2 Zadatak 2. Filtriranje i pretraživanje

Potrebno je omogućiti korisniku da filtrira prikazane podatke prema zadanim kriterijima. Primjeri filtriranja uključuju:

- za filmove: filtriranje po žanru, državi, rasponu godina ili minimalnoj ocjeni;
- za temperature: filtriranje po sezoni, lokaciji, rasponu temperatura ili vrsti vremena.

Filtriranje se može implementirati pomoću padajućih izbornika, tekstualnih polja za unos, radio gumba ili raspona vrijednosti (npr. `input type="range"`). Rezultati filtriranja trebaju se odmah prikazati korisniku, bez ponovnog učitavanja stranice.

Napomena: Potrebno je implementirati **minimalno tri (3)** različita kriterija za filtriranje. Korišteni elementi filtriranja moraju biti različiti po vrsti (npr. *kombinacija padajućeg izbornika, tekstualnog polja i slidera*) i/ili se odnositi na različite varijable u odnosu na primjere prikazane u predlošku.

Nije dozvoljeno koristiti identične kriterije i vrste elemenata kao u predlošku.

3.2.1 Koraci za rješavanje

- Priprema HTML strukture - osigurati da korisnik ima sučelje za unos kriterija filtriranja - tri različite vrste filtriranja (text, slider, radio), urediti pomoću CSS-a;
- Povezivanje filtera s JavaScriptom - dodati `event listener` na gumb "Filtriraj" (na primjer);
- prikaz filtriranih podataka - koristiti već postojeću funkciju; `prikaziTablicu(filmovi)` koja je definirana za dinamički prikaz;
- dinamičko ažuriranje vrijednosti za `range` input - da korisnik vidi koju je vrijednost postavio.

Primjer 6: Primjer filtera u HTML dokumentu

```
<!-- Filtri -->
<div id="filteri">
<select id="filter-genre">
<option value="">-- Odaberi zanr --</option>
<option value="Comedy">Comedy</option>
<option value="Drama">Drama</option>
<!-- Ostali zanrovi -->
</select>

<input type="number" id="filter-year" placeholder="Godina_od">

<select id="filter-country">
<option value="">-- Odaberi drzavu --</option>
<option value="United_States">United States</option>
<option value="Italy">Italy</option>
<!-- Ostale drzave -->
</select>

<input type="range" id="filter-rating" min="0" max="10" step="
  0.1" value="7">
<span id="rating-value">7.0</span>

<button id="primijeni-filteri">Filtriraj</button>
</div>

<!-- Tablica -->
<table id="filmovi-tablica">
<thead>
<tr>
<th>Naslov</th>
<th>Godina</th>
<th>Zanr</th>
<th>Trajanje</th>
<th>Drzava</th>
<th>Ocjena</th>
</tr>
</thead>
<tbody></tbody>
</table>
```

Napredni savjet: Višestruko filtriranje Za studente koji žele dodatno unaprijediti svoju aplikaciju, preporučuje se implementacija mogućnosti višestrukog odabira kod određenih filtera, poput žanra ili države.

Primjerice, umjesto klasičnog padajućeg izbornika (<select>), može se koristiti opcija `multiple` koja omogućuje odabir više vrijednosti:

Primjer 7: Primjer višestrukog odabira žanra

```
<select id="filter-genre" multiple>
<option value="Comedy">Comedy</option>
```



```
<option value="Drama">Drama</option>
<option value="Action">Action</option>
</select>
```

U JavaScript dijelu, dohvaćanje svih odabranih vrijednosti može se ostvariti na sljedeći način:

Primjer 8: Dohvaćanje više odabranih žanrova

```
const selectedGenres = Array.from(document.getElementById('
  filter-genre').selectedOptions)
  .map(option => option.value);
```

Prilikom filtriranja potrebno je provjeriti sadrži li film barem jedan od odabranih žanrova:

```
const filtriraniFilmovi = filmovi.filter(film => {
  return (selectedGenres.length === 0 || selectedGenres.
    some(g => film.genre.includes(g)));
  // Ostali filteri...
});
```

Napomena: Ovakav pristup omogućuje korisnicima fleksibilnije pretraživanje i dodatno poboljšava korisničko iskustvo aplikacije.

3.3 Struktura JavaScript koda za dohvat i prikaz podataka

JavaScript rješenje sastoji se od nekoliko logičkih cjelina koje omogućuju dohvat podataka iz CSV datoteke, početni prikaz filmova te filtriranje prema zadanim kriterijima.

3.3.1 Dohvat i parsiranje podataka iz CSV-a

Iako je već gore spomenuto parsiranje, bito je. Na početku skripte koristi se funkcija `fetch()` za dohvat CSV datoteke, a biblioteka `PapaParse` za parsiranje podataka u format pogodan za daljnju obradu:

Primjer 9: Dohvat i parsiranje CSV datoteke

```
let sviFilmovi = [];

fetch('filmtv_movies.csv')
  .then(res => res.text())
  .then(csv => {
    const rezultat = Papa.parse(csv, {
      header: true,
      skipEmptyLines: true
    });

    sviFilmovi = rezultat.data.map(film => ({
      title: film.title,
      year: Number(film.year),
      genre: film.genre,
      duration: Number(film.duration),
      country: film.country?.split(',').map(c => c.trim()) || [],
      avg_vote: Number(film.avg_vote),
      total_votes: Number(film.total_votes)
    }));

    prikaziPocetneFilmove(sviFilmovi.slice(0, 10));
    prikaziFiltriraneFilmove(sviFilmovi.slice(0, 10));
  })
  .catch(err => {
    console.error('Greska_pri_dohvacanju_CSV-a:', err);
  });
```

Objašnjenje:

- `fetch()` dohvaća CSV datoteku asinkrono.
- `Papa.parse()` pretvara CSV u niz objekata.
- Metoda `map()` obrađuje podatke i osigurava ispravan tip za broćane vrijednosti.
- Funkcije `prikaziPocetneFilmove` i `prikaziFiltriraneFilmove` prikazuju početni set filmova.

Lokalna i globalna pohrana podataka

Prilikom obrade podataka iz CSV datoteke, važno je odabrati odgovarajući način spremanja podataka, ovisno o potrebama aplikacije.

U početnoj verziji koda (u Zad1, 3.1.2)), podaci su se spremali u lokalnu varijablu `film` unutar funkcije za dohvat i parsiranje CSV-a.

Ovakav način pohrane koristi `const` varijablu `filmovi`, koja je dostupna isključivo unutar bloka gdje je definirana. To je primjereno kada su podaci potrebni samo za inicijalni prikaz ili jednokratnu obradu.

Međutim, kako aplikacija raste i dodaju se nove funkcionalnosti poput filtriranja i upravljanja košaricom, potrebno je omogućiti pristup tim podacima i izvan početnog bloka. U tu svrhu koristi se globalna varijabla definirana s `let` izvan funkcije:

```
let sviFilmovi = [];
```

Korištenjem globalne varijable `sviFilmovi`, omogućuje se pristup istim podacima u različitim dijelovima aplikacije, primjerice za:

- Dinamički prikaz početnih podataka.
- Filtriranje prema korisničkim kriterijima.
- Dodavanje filmova u košaricu.

Lokalne varijable (`const`) koriste se kada je potreban ograničen doseg (eng. *scope*), dok se globalne varijable (`let`) koriste kada su podaci potrebni za višestruke operacije kroz različite funkcije aplikacije. Pravilnim odabirom tipa varijable povećava se čitljivost i funkcionalnost koda.

3.3.2 Prikaz početnih filmova

Ova funkcija dinamički puni tablicu s prvim filmovima iz baze podataka.

Primjer 10: Prikaz početnih filmova

```
function prikaziPocetneFilmove(filmovi) {
  const tbody = document.querySelector('#filmovi-tablica_
    tbody');
  tbody.innerHTML = '';

  for (const film of filmovi) {
    const row = document.createElement('tr');
    row.innerHTML = `
      <td>${film.title}</td>
      <td>${film.year}</td>
      <td>${film.genre}</td>
      <td>${film.duration} min</td>
      <td>${film.country.join(', ')}</td>
      <td>${film.total_votes}</td>
    `;
    tbody.appendChild(row);
  }
}
```

Objašnjenje:

- Pomoću `document.querySelector` dohvaća se `<tbody>` unutar tablice.
- Svaki film se prikazuje kao novi redak (`<tr>`).

3.3.3 Ažuriranje prikaza vrijednosti slidera

Slider omogućuje korisniku da odabere minimalnu ocjenu za filtriranje filmova. Vrijednost se dinamički prikazuje pokraj slidera.

Primjer 11: Ažuriranje vrijednosti slidera

```
const rangeInput = document.getElementById('filter-rating');
const ratingDisplay = document.getElementById('rating-value');
rangeInput.addEventListener('input', () => {
    ratingDisplay.textContent = rangeInput.value;
});
```

3.3.4 Filtriranje filmova prema kriterijima

Glavna funkcija za filtriranje dohvaća vrijednosti iz formi i primjenjuje filtere na sve filmove.

Primjer 12: Filtriranje filmova

```
function filtriraj() {
    const zanr = document.getElementById('filter-genre').
        value.trim().toLowerCase();
    const godinaOd = parseInt(document.getElementById('
        filter-year-from').value);
    const drzava = document.getElementById('filter-country')
        .value.trim().toLowerCase();
    const ocjena = parseFloat(document.getElementById('
        filter-rating').value);

    const filtriraniFilmovi = sviFilmovi.filter(film => {
        const zanrMatch = !zanr || film.genre.toLowerCase()
            .includes(zanr);
        const godinaOdMatch = !godinaOd || film.year >=
            godinaOd;
        const drzavaMatch = !drzava || film.country.some(c
            => c.toLowerCase().includes(drzava));
        const ocjenaMatch = film.avg_vote >= ocjena;
        return zanrMatch && godinaOdMatch && drzavaMatch
            && ocjenaMatch;
    });

    prikaziFiltriraneFilmove(filtriraniFilmovi);
}
```

Objašnjenje:

- Svaki kriterij provjerava je li polje prazno ili odgovara uvjetima.
- Metoda `filter()` vraća samo one filmove koji zadovoljavaju sve uvjete.

3.3.5 Pozivanje funkcije filtriranja

Klikom na gumb `Filtriraj`, pokreće se funkcija za filtriranje:

Primjer 13: Pokretanje filtriranja

```
document.getElementById('primijeni-filtre').addEventListener('click', filtriraj);
```

3.3.6 Prikaz filtriranih rezultata

Filtrirani filmovi prikazuju se u drugoj tablici:

Primjer 14: Prikaz filtriranih filmova

```
function prikaziFiltriraneFilmove(filmovi) {  
    const tbody = document.querySelector('#  
        filtriranje_tablica_tbody');  
    tbody.innerHTML = '';  
  
    if (filmovi.length === 0) {  
        tbody.innerHTML = '<tr><td_colspan="6">Nema_  
            filmova_za_odabrane_filtre.</td></tr>';  
        return;  
    }  
  
    for (const film of filmovi) {  
        const row = document.createElement('tr');  
        row.innerHTML = `  
            <td>${film.title}</td>  
            <td>${film.year}</td>  
            <td>${film.genre}</td>  
            <td>${film.duration} min</td>  
            <td>${film.country.join(', ')}</td>  
            <td>${film.avg_vote}</td>  
        `;  
        tbody.appendChild(row);  
    }  
}
```

Napomena: Ako nema rezultata, prikazuje se poruka korisniku unutar tablice.

Virtualna Videoteka

Žanr: Komedija Godina od: 1980 Država: Italy Prosječna ocjena: 7.0 Filtriraj

Naslov	Godina	Žanr	Trajanje	Država
L'aria serena dell'Ovest	1990	Comedy	115 min	Italy
Café Express	1980	Comedy	100 min	Italy
Compagni di scuola	1988	Comedy	118 min	Italy
La famiglia	1986	Comedy	127 min	Italy
Ginger e Fred	1986	Comedy	128 min	Italy
Johnny Stecchino	1991	Comedy	123 min	Italy
Il Marchese del Grillo	1981	Comedy	135 min	Italy
Marrakech Express	1989	Comedy	95 min	Italy
Mediterraneo	1991	Comedy	99 min	Italy
Nuovo Cinema Paradiso	1988	Comedy	123 min	Italy, France
Oci Ciornie	1987	Comedy	117 min	Italy
Palombella rossa	1989	Comedy	89 min	Italy

Slika 3: Primjer rješenja prikaza filtriranih podataka

3.4 Zadatak 3. Interaktivno upravljanje sadržajem (npr. dodavanje u košaricu/plan).

Cilj ovog zadatka je omogućiti korisniku interaktivno upravljanje filtriranim podacima kroz funkcionalnost dodavanja elemenata u "košaricu", pregled košarice, uklanjanje elemenata iz košarice, te potvrdu odabira.

Opis funkcionalnosti

- Nakon filtriranja elemenata, uz svaki element potrebno je prikazati gumb `Dodaj u košaricu`.
- Klikom na gumb, element se dodaje u košaricu (koristiti JavaScript niz).
- Omogućiti pregled košarice u posebnoj tablici ili sekciji stranice.
- U košarici omogućiti:
 - Prikaz svih dodanih filmova (naslov, godina, žanr).
 - Gumb za uklanjanje pojedinog filma iz košarice.
 - Gumb `Potvrdi posudbu`.
- Nakon potvrde, prikazati obavijest korisniku o uspješnoj posudbi filmova.

Napomena: Prikaz košarice može biti implementiran na različite načine, ovisno o dizajnu i preferencijama. Neki od mogućih pristupa uključuju:

- **Aside element:** Prikaz košarice sa strane stranice, stalno vidljiv ili prikazan na zahtjev.
- **Dropdown izbornik:** Košarica se prikazuje kada korisnik klikne na ikonu ili gumb, slično kao u web trgovinama.
- **Nova HTML stranica:** Omogućiti korisniku da klikne na `Pregled košarice` koji ga vodi na novu stranicu gdje može vidjeti detalje odabranih filmova.
- **Modalni prozor (opc.):** Prikaz košarice u skočnom prozoru bez napuštanja trenutne stranice.

Odaberite pristup koji najbolje odgovara vašem korisničkom sučelju i osigurajte da je korisniku jasno vidljiv broj odabranih filmova te mogućnost pregleda i uređivanja košarice.

Primjer korištenja

1. Korisnik filtrira filmove prema željenim kriterijima.
2. Iz prikazanih rezultata odabire 3 filma klikom na `Dodaj u košaricu`.
3. Otvara prikaz košarice gdje vidi popis odabranih filmova.
4. Uklanja jedan film iz košarice.
5. Klikom na `Potvrdi posudbu`, prikazuje se poruka:

"Uspješno ste dodali 2 filma u svoju košaricu za vikend maraton!"

Tehničke upute

- Za pohranu košarice koristiti JavaScript niz (opcionalno: `LocalStorage` za trajnost podataka).
- Svaki film u tablici filtriranih rezultata mora imati gumb za dodavanje.
- Dinamički generirati prikaz košarice kroz DOM manipulaciju.
- Osigurati da korisnik može ukloniti filmove prije potvrde.
- Prikazati obavijest nakon potvrde (alert, modalni prozor ili jednostavna poruka na stranici).

Primjer implementiranja zadatka, odnosno dodavanje filmova u košaricu s prikazom košarice u aside elementu.

U ovom zadatku potrebno je implementirati funkcionalnost dodavanja filmova u "košaricu", pregled košarice, uklanjanje filmova te potvrdu odabira uz prikaz obavijesti.

Struktura rješenja

Rjesenje se sastoji od tri glavna dijela:

1. **HTML** – definiranje prostora za prikaz košarice.
2. **JavaScript** – logika za dodavanje, uklanjanje i potvrdu odabira filmova.
3. **CSS** – osnovno stiliziranje košarice.

HTML – Definiranje košarice

Za prikaz košarice koristi se HTML element `<aside>` koji omogućava da košarica bude vidljiva korisniku tijekom korištenja aplikacije.

Primjer 15: HTML struktura košarice

```
<aside id="kosarica">
<h2>Moja kosarica</h2>
<ul id="lista-kosarice">
<!-- Dinamicki popunjeno -->
</ul>
<button id="potvrди-kosaricu">Potvrди odabir</button>
</aside>
```

Također, u tablicu s filtriranim filmovima dodaje se novi stupac s gumbom za dodavanje filma u košaricu.

Primjer 16: Dodavanje gumba u tablicu

```
// Funkcija za prikaz filtriranih filmova
function prikaziFiltriraneFilmove(filmovi) {
    const tbody = document.querySelector('#filtriranje_tablica_
        tbody');
    tbody.innerHTML = ''; // "brise" sav trenutni sadrzaj unutar <
        ul> elementa.
```



```

    if (filmovi.length === 0) {
        tbody.innerHTML = '<tr><td_colspan="6">Nema_filmova_za_
            odabrane_filtre.</td></tr>';
        return;
    }

    for (const film of filmovi) {
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${film.title}</td>
            <td>${film.year}</td>
            <td>${film.genre}</td>
            <td>${film.duration} min</td>
            <td>${film.country.join(',')}</td>
            <td>${film.avg_vote}</td>
        `;
        tbody.appendChild(row);
    }
}

```

JavaScript – Logika upravljanja košaricom

Primjer 17: Dodavanje filmova u košaricu

```

let kosarica = [];

function dodajUKosaricu(film) {
    if (!kosarica.includes(film)) {
        kosarica.push(film);
        osvjeziKosaricu();
    } else {
        alert("Film_je_vec_u_kosarici!");
    }
}

function osvjeziKosaricu() {
    const lista = document.getElementById('lista-kosarice');
    lista.innerHTML = '';

    kosarica.forEach((film, index) => {
        const li = document.createElement('li');
        li.textContent = film.title;

        const ukloniBtn = document.createElement('button');
        ukloniBtn.textContent = 'Ukloni';
        ukloniBtn.addEventListener('click', () => {
            ukloniIzKosarice(index);
        });
    });
}

```

```

        li.appendChild(ukloniBtn);
        lista.appendChild(li);
    });
}

function ukloniIzKosarice(index) {
    kosarica.splice(index, 1);
    osvjeziKosaricu();
}

// Potvrda kosarice
document.getElementById('potvrdi-kosaricu').addEventListener('click',
    () => {
        if (kosarica.length === 0) {
            alert("Kosarica je prazna!");
        } else {
            alert(`Uspjesno ste odabrali ${kosarica.length} filmova
                za gledanje!`);
            kosarica = [];
            osvjeziKosaricu();
        }
    });
});

```

CSS – Stiliziranje kšsarice

Za bolju preglednost, košarica se stilizira i postavlja sa strane.

Primjer 18: Osnovni stil za košaricu

```

#kosarica {
    position: fixed;
    right: 20px;
    top: 100px;
    width: 250px;
    background-color: #f4f4f4;
    border: 1px solid #ccc;
    padding: 10px;
}

#kosarica ul {
    list-style-type: none;
    padding: 0;
}

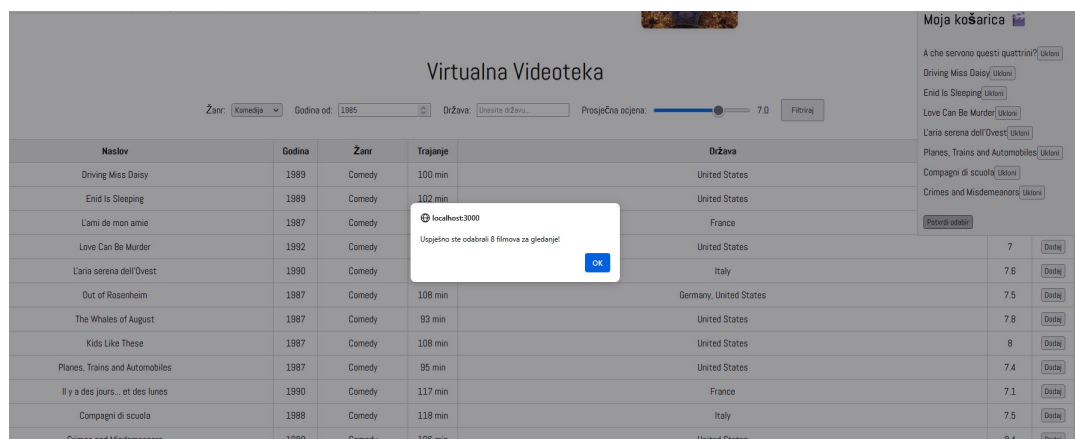
#kosarica button {
    margin-top: 10px;
}

```

Napomena

Gore navedenim dijelovima koda može se prilagoditi izgled košarice prema vlastitim idejama:

- Prikaz u `dropdown` izborniku.
- Otvaranje nove HTML stranice za detaljan pregled košarice.
- Spremanje odabranih filmova u `LocalStorage` za trajno cuvanje.



Naslov	Godina	Žanr	Trajanje	Država
Driving Miss Daisy	1989	Comedy	100 min	United States
Enid la Sleeping	1989	Comedy	102 min	United States
L'ami de mon amie	1987	Comedy		France
Love Can Be Murder	1992	Comedy		United States
L'aria serena dell'Ovest	1990	Comedy		Italy
Out of Rosenheim	1987	Comedy	108 min	Germany, United States
The Whales of August	1987	Comedy	93 min	United States
Kids Like These	1987	Comedy	108 min	United States
Planes, Trains and Automobiles	1987	Comedy	95 min	United States
Il y a des jours... et des lunes	1990	Comedy	117 min	France
Compagni di scuola	1988	Comedy	118 min	Italy
Crimes and Misdemeanors	1989	Comedy	106 min	United States

Slika 4: Primjer rješenja prikaza podataka u košarici

3.5 Dodatno: EJS kao alternativa

Studenti koji žele dodatno proširiti zadatak mogu koristiti `EJS` predloške za generiranje sadržaja na strani poslužitelja. Umjesto da JavaScript u pregledniku dinamički popunjava HTML, podaci se mogu pročitati na serveru i prikazati korištenjem `.ejs` predložaka. Ova metoda može se koristiti kao alternativa prikazu tablice pomoću DOM manipulacije u `script.js`.

U ovoj verziji projekta koristi se tehnologija `EJS` za generiranje HTML sadržaja na strani poslužitelja. Podaci se čitaju pomoću Node.js modula (npr. `fs` ili `csv-parser`), a zatim se proslijeđuju `EJS` predlošcima koji dinamički generiraju HTML. Prikazani sadržaj zatim se poslužuje klijentu putem ruta definiranih u Express aplikaciji.

```
my-project/
├── public/
│   └── style.css
├── views/
│   ├── index.ejs
│   └── filmovi.ejs
├── data/
│   └── filmovi.json
├── routes/
│   └── index.js
├── app.js
└── package.json
```

Slika 5: Primjer strukture projekta s korištenjem `EJS` predložaka

U nastavku predložka nalazi se kratak opis načina na koji je realizirano rješenje zadatka. Radi boljeg razumijevanja dokumentiranja rada te lakšeg razumijevanja implementiranih funkcionalnosti potrebno je proučiti odgovarajući projekt

4.1 Web aplikacija za planiranje aktivnosti prema vremenskim uvjetima

Cilj

Cilj ove laboratorijske vježbe je izraditi interaktivnu web aplikaciju koja omogućuje korisnicima planiranje aktivnosti na otvorenom temeljem vremenskih uvjeta. Korištenjem zadanog skupa meteoroloških podataka, potrebno je omogućiti filtriranje dana s poželjnim vremenskim uvjetima te dodavanje tih dana u osobni "plan aktivnosti".

Korisnička priča

Marko je zaljubljenik u planinarenje i želi iskoristiti nadolazeće proljetne dane za izlete u planine. No, nema vremena svakodnevno pratiti vremensku prognozu. Koristi aplikaciju koja mu omogućuje da filtrira dane prema idealnim uvjetima: proljeće, sunčano vrijeme, planinska lokacija i ugodna temperatura između 18°C i 25°C. Nakon što odabere pogodne dane, sprema ih u svoj plan aktivnosti kako bi unaprijed znao kada su najbolji uvjeti za izlet.

Opis zadatka

Potrebno je izraditi web aplikaciju koja omogućuje:

- Prikaz vremenskih podataka u tabličnom obliku.
- Filtriranje podataka prema sezoni, lokaciji, tipu vremena, temperaturi i oborinama - minimalno tri vrste filtriranja podataka
- Dodavanje odabranih dana u "plan aktivnosti".
- Pregled i uklanjanje dana iz plana - aside, dropdown, ili nova html stranica
- Prikaz informativne poruke nakon potvrde plana.

Napomena: Polje `ID` u skupu podataka predstavlja datum na koji se odnose vremenski uvjeti.

Primjer korištenja aplikacije

Korisnik koristi aplikaciju na sljedeći način:

1. Odabire filtre:

- **Sezona:** Spring
- **Lokacija:** Mountain
- **Weather Type:** Sunny

- **Temperatura:** 18°C – 25°C

2. Aplikacija prikazuje sve dane koji zadovoljavaju odabrane uvjete.
3. Korisnik klikne na "Dodaj u plan" za željene dane.
4. Korisnik ima mogućnost ukloniti neželjene dane.
5. Nakon toga, korisnik klikne na "Pregled plana" i dobiva sljedeći prikaz:

Datum (ID)	Temperatura	Lokacija	Vrijeme
W6418	20°C	Mountain	Sunny
W7358	22°C	Mountain	Sunny

Na kraju se prikazuje poruka:

"Uspješno ste isplanirali 2 dana za svoje aktivnosti na otvorenom!"

Tehničke upute

- Podatke učitati iz CSV datoteke korištenjem biblioteke `PapaParse`.
- Sva logika mora biti implementirana na klijentskoj strani (HTML, CSS, JavaScript).
- Koristiti jednostavan, pregledan dizajn.
- Omogućiti dinamičku manipulaciju DOM elementima.

Predaja rješenja

- Projekt postaviti na GitHub.
- Aplikaciju objaviti na besplatnom hosting servisu (npr. Vercel, Netlify, Railway).
- U `README.md` dokumentu navesti kratak opis aplikacije, funkcionalnosti i poveznicu na objavljenu aplikaciju.

4.2 Web aplikacija - Virtualna videoteka

Cilj

Cilj ove laboratorijske vježbe je izraditi interaktivnu web aplikaciju koja omogućuje korisnicima pregled, pretraživanje i odabir filmova iz virtualne videoteke. Korištenjem zadanog skupa podataka o filmovima, potrebno je implementirati funkcionalnosti filtriranja, sortiranja te dodavanja filmova u "košaricu za posudbu".

Korisnička priča

Ana je ljubiteljica filmova i želi kreirati svoj popis filmova za vikend maraton. Umjesto da pretražuje nepregledne liste, koristi aplikaciju koja joj omogućuje da filtrira filmove prema žanru, godini ili zemlji proizvodnje. Kada pronađe filmove koji joj se sviđaju, dodaje ih u svoju "košaricu za posudbu" kako bi lakše pratila što želi gledati.

Opis zadatka

Potrebno je izraditi web aplikaciju koja omogućuje:

- Prikaz podataka o filmovima u tabličnom obliku (naslov, godina, žanr, trajanje, država, ocjena).
- Filtriranje filmova prema žanru, godini proizvodnje (od-do) i zemlji.
- Sortiranje filmova po godini ili broju glasova.
- Dodavanje filmova u "košaricu za posudbu".
- Pregled i uklanjanje filmova iz košarice.
- Prikaz obavijesti nakon potvrde odabira filmova.

Napomena: Polje `country` može sadržavati više država, potrebno je obratiti pažnju na formatiranje prilikom prikaza.

Primjer korištenja aplikacije

- Korisnik otvara aplikaciju i filtrira filmove prema:
 - **Žanr:** Comedy
 - **Godina:** od 1980 do 2000.
 - **Država:** United States
 - **Prosječna ocjena:** između 7.0 i 10.0
- Aplikacija prikazuje sve filmove koji zadovoljavaju postavljene uvjete.
- Korisnik odabire 3 filma i dodaje ih u "košaricu za posudbu".
- Klikom na "Pregled košarice", korisniku se prikazuje popis odabranih filmova s mogućnošću uklanjanja pojedinih naslova.
- Nakon potvrde, prikazuje se poruka, u ovisnosti koliko je dodano filmova:

"Uspješno ste dodali 3 filma u svoju košaricu za vikend maraton!"

Tehničke upute

- Podatke učitati iz CSV datoteke korištenjem biblioteke `PapaParse` ili koristiti prethodno pripremljeni JSON.
- Implementirati logiku isključivo na klijentskoj strani (HTML, CSS, JavaScript).
- Omogućiti dinamički prikaz i manipulaciju podacima kroz DOM.
- Dizajnirati pregledno korisničko sučelje uz osnovno stiliziranje tablice.

Predaja rješenja

- Projekt postaviti na GitHub.
- Aplikaciju objaviti na besplatnom hosting servisu (npr. Vercel, Netlify, Railway).
- U `README.md` dokumentu navesti opis aplikacije, popis funkcionalnosti i poveznicu na objavljenu verziju aplikacije.