**УНИВЕРЗИТЕТ У БЕОГРАДУ**

**ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА**

**ЗАВРШНИ РАД**

**Веб апликација „Тениски турнири на мапи“**

**Ментор Студент**

**Др Душан Бараћ, Наталија Никић, 2020/0129**

**Редовни професор**

**Београд, 2025. године**

*Овај рад посвећујем родитељима који су ми омогућили све у животу и брату тенисеру који ме је инспирисао за исти .*

**АПСТРАКТ**

Овај рад се бави развојем веб апликације „Тениски турнири на мапи“, са главним фокусом на коришћење технологије React. Кључни аспект је на иновацији управљања подацима о тениским турнирима и њиховој доступности. Рад обухвата два дела: прикупљање података о турнирима и њихову визуализацију на интерактивној мапи. Циљ рада је да се омогући младим тенисерима као и љубитељима тениса олакшано проналажење турнира уз помоћ мапе и напредног филтрирања.

Методологија која је примењена у раду укључује две главне фазе. Прва фаза подразумева прикупљање података о тениским турнирима са различитих сајтова коришћењем Node.js-a и библиотеке Puppeteer за екстракцију података и њихово форматирање у JSON фајлове. Друга фаза пројекта представља креирање интерфејса, где су прикупљени подаци интегрисани у React веб апликацију која визуализује турнире на мапи уз помоћ библиотеке OpenLayers, омогућавајући филтрирање по одговарајућим критеријумима.

Кључни резултати укључују аутоматизацију процеса прикупљања података и развој функционалне апликације која значајно поједностављује претрагу тениских турнира. Закључак рада показује да је овај систем користан за унапређење доступности информација о турнирима, уз потенцијал за проширење функција и коришћење у другим спортовима.

**Садржај**

[УВОД 5](#_Toc189048960)

[ПРЕГЛЕД ЛИТЕРАТУРЕ 6](#_Toc189048961)

[1. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ 6](#_Toc189048962)

[1.1. HTML, CSS (SCSS) и JavaScript 6](#_Toc189048963)

[1.2. TypeScript 8](#_Toc189048964)

[1.3. Node.js 10](#_Toc189048965)

[1.4. React 12](#_Toc189048966)

[1.5. Библиотеке Puppeteer и ОpenLayers 16](#_Toc189048967)

[2. УПОТРЕБА И ИНТЕГРАЦИЈА АПИ-ЈА У АПЛИКАЦИЈИ 17](#_Toc189048968)

[2.1. Основни појмови и принципи функционисања 17](#_Toc189048969)

[2.2. Веб АПИ-ји коришћени у пројекту 18](#_Toc189048970)

[3. ПРЕГЛЕД ПОСТОЈЕЋИХ СОФТВЕРСКИХ РЕШЕЊА У ДОМЕНУ ТЕНИСКИХ ТУРНИРА 19](#_Toc189048971)

[3.1. Анализа постојећих решења 20](#_Toc189048972)

[3.2. Поређење решења 20](#_Toc189048973)

[3.3. Предност софтверског решења „Тениски турнири на мапи“ 21](#_Toc189048974)

[ИМПЛЕМЕНТАЦИЈА 22](#_Toc189048975)

[1. СТРУКТУРА КОДА 22](#_Toc189048976)

[2. ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА 24](#_Toc189048977)

[2.1. Модул зависности 24](#_Toc189048978)

[2.2. Екстракција података и форматирање у ЈSON фајлове 25](#_Toc189048979)

[2.3. Конфигурација АПИ-ја 28](#_Toc189048980)

[3. ИНТЕГРАЦИЈА И ВИЗУАЛИЗАЦИЈА ПОДАТАКА 33](#_Toc189048981)

[3.1. Дрво компоненти 33](#_Toc189048982)

[3.2. Коришћење OpenLayers библиотеке 33](#_Toc189048983)

[3.3. Управљање подацима о тениским турнирима 36](#_Toc189048984)

[3.4. Функционалности 41](#_Toc189048985)

[ПРИКАЗ СОФТВЕРСКОГ РЕШЕЊА ЗА МАПУ ТЕНИСКИХ ТУРНИРА 42](#_Toc189048986)

[ТЕСТИРАЊЕ 43](#_Toc189048987)

[ЗАКЉУЧАК 44](#_Toc189048988)

[ЛИТЕРАТУРА 45](#_Toc189048989)

# УВОД

У свету тениса, млади играчи и љубитељи овог спорта се могу суочити са изазовом проналажења информација о тениским турнирима. Већина информација је распршена по различитим сајтовима и недоступна на једном месту. Ова веб апликација пружа решење тако што на једноставан и интуитиван начин визуелно приказује мапу са доступним турнирима. Корисници могу брзо и лако прегледати када и где се турнири одржавају, користећи филтере за прецизнију претрагу. На овај начин доступност и прегледност релативних података поједностављује планирање и учествовање на тениским турнирима.

Овај рад се састоји из два дела за које су коришћене раличите технологије како би апликација била функционална. Први део пројекта, у којем се прикупљају подаци са различитих веб сајтова о тениским турнирима коришћени су JavaScript, NodeJS и Puppeteer библиотека. Други део се односи на део пројакта у којем се развија интерактивна веб апликација у React-u, уз коришћење TypeScript-a и OpenLayers библиотеке за визуализацију турнира на мапи, омогућавајући корисницима лако претраживање и филтрирање података по различитим критеријумима.

Друго поглавље овог рада ће се бавити прегледом литературе, укључујући техничку документацију и релевантну библиографију. Посебан акценат ће бити на опису постојећих софтверских решења у домену тениских турнира, чиме ће се идентификовати празнине које нова апликација може попунити.

У трећем поглављу, детаљно ће бити описан процес имплементације апликације. Биће обухваћена структура кода и дрво компоненти, конфигурација АПИ-ја, као и опис делова задужених за прикупљање података и кориснички интерфејс. Такође, функционалности апликације ће бити анализиране кроз преглед одговарајућих делова кода.

Четврто поглавље ће пружити свеобухватан приказ софтверског решења за тениске турнире, док ће пето поглавље обухватити процес тестирања апликације.

# ПРЕГЛЕД ЛИТЕРАТУРЕ

## КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

За први део програма, односно за део апликације који је задужен за екстракцију података са страница, коришћене су технологије *JavaScript*, *NodeJS*, *Google Chrome* и библиотека *Puppeteer*.

За други део програма, односно за део апликације који је задужен за визуализацију прикупљених података, коришћене су технологије *React, TypeScript, HTML, SCSS* и библиотека *OpenLayers.*

### HTML, CSS (SCSS) и JavaScript

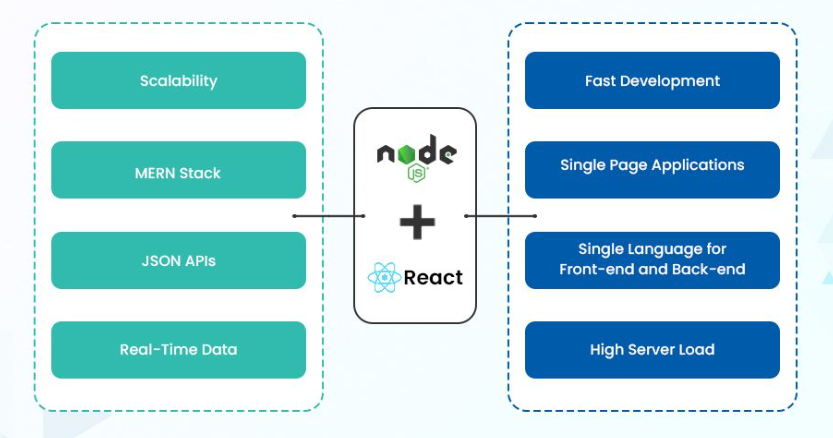
**HTML** (HyperText Markup Language) је језик којим се дефинише структура и садржај веб странице. Он омогућава корисницима да додају различите елементе попут заглавља, текстова, слика, линкова и формулара, организујући садржај у логичне делове. HTML je попут оквира или скелета странице који одређује шта ће бити приказано. (W3schools, 2025 )

**CSS** (Cascading Style Sheets) је језик за стилско обликовање HTML елемената и омогућава контролу над изгледом веб странице. Док HTML дефинише структуру странице, CSS омогућава да се та структура визуелно обликује и учини привлачнијом. „Каскадно“ у имену указује на начин наслеђивања стилова, омогућавајући прилагођавање изгледа у различитим деловима странице и њену прилагодљивост на различитим уређајима. (Codecademy.com, 2018) У овом пројекту, коришћен је **SCSS** који има неке додатне могућности и део је **Sass-а**, популарног препроцесора CSS-a.

**JavaScript** је програмски језик који омогућава да веб сајтови и апликације буду динамични и интерактивни. Створен је 1995. године за Netscape Navigator, тада популарни прегледач (browser). (Ketan Varshneya, 2023) Првобитна намена JavaScript-а била је извршавање у веб прегледачу, али је касније добио и друге намене када је развијена серверска подршка за овај језик. Проширења на страни сервера омогућавају апликацији комуникацију са базом података, обезбеђују континуитет информација из једног позива апликације у други или омогућавају манипулацију датотекама на серверу. (Geeksforgeeks.org, 2025) Најпознатија платформа за покретање JavaScript-а на серверу је **Node.js.**

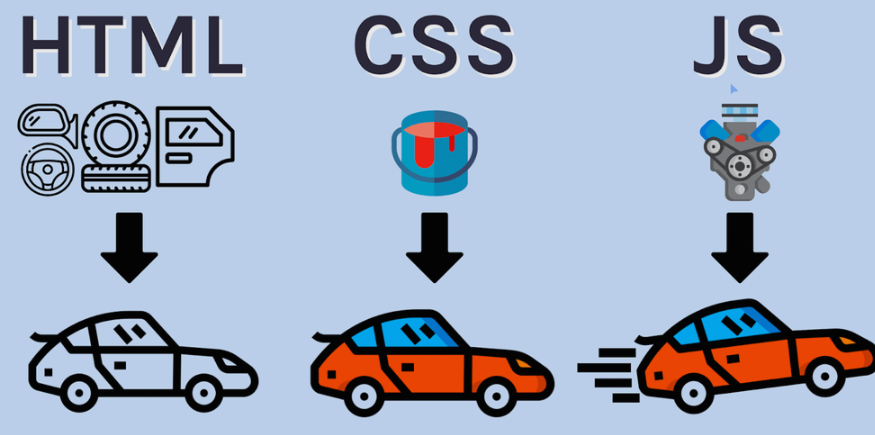
На страни клијента омогућава манипулацију претраживачем и његовим Document Object Model-ом (DOM). (Geeksforgeeks.org, 2025) Популарна библиотека за рад на страни клијента која се користи у овом пројекту је **React**.

React и Node.js су идеална комбинација за развој веб апликација о тениским турнирима. React омогућава брз и интерактиван кориснички интерфејс, док Node.js, као стандард за рад са веб апликацијама, пружа снажну подршку и управљање подацима. . (Ved Raj, 2021)



Slika 1 – Why Use Node.js with React for Web App Development? (https://radixweb.com/blog/build-web-apps-with-react-and-node-js)

JavaScript, CSS и HTML заједно чине елементе корисничког интерфејса већине веб сајтова и апликација. Неки веб сајт се се може упоредити са аутомобилом који треба да се направи, а програмери су механичари задужени за његову изградњу. Могло би се рећи да се HTML користи за основну структуру аутомобила или „скелет“. CSS је ту да се ауто улепша фарбањем, додавањем кожних седишта и алуминијумских фелни. Помоћу ЈS-a додају се завршни делови аутомобила попут мотора, звучника и осветљења који га заправо претварају у функциално возило. (Jaye, 2022)



Slika 2 – How HTML, CSS and JavaScript work (https://v1.scrimba.com/articles/html-css-javascript/)

### TypeScript

**TypeScript** је програмски језик отвореног кода који је развила и одржава компанија Microsoft. То је строг синтактички надскуп JavaScript-а, што значи да се надограђује на JavaScript додавањем нових функција, посебно статичког типизирања (иако омогућава опционално статичко типизирање). Док JavaScript може директно да се извршава у веб претраживачу, TypeScript захтева компајлер да претвара код у JavaScript. Када се користи TypeScript, могу се идентификовати грешке током компајлирања уместо током извршавања.

TypeScript је први пут представљен јавности на GitHub-u 2012. године и омогућава коришћење напредних JavaScript функција на старијим претраживачима који их обично не подржавају. Главне функције TypeScript -а:

* Статичко типизирање: За разлику од JavaScript-а, TypeScript вам омогућава да дефинишете типове података за променљиве, параметре функција и повратне вредности. Ово доводи до мањег броја грешака и лакшег одржавања кода јер се грешке хватају током процеса компилације уместо током извршења.
* Инференција типова: TypeScript може аутоматски закључивати типове чак и ако их не дефинишете експлицитно.
* Интероперабилност са JavaScript-ом: Као надскуп JavaScript -а, сав JavaScript код је валидан TypeScript код. Може се постепено усвојити TypeScript у постојећем JavaScript пројекту.
* Модерни JavaScript функционалности: TypeScript подржава најновије JavaScript функционалности и пружа будућу компатибилност преводећи их у обичан JavaScript.
* Подршка за алате и уређиваче: TypeScript нуди одличну подршку за алате, укључујући моћно аутоматско довршавање, навигацију и алатке за рефакторисање у уређивачима.

*Предности TypeScript -а:*

* Побољшан квалитет кода: Статичко проверавање типова помаже у раном хватању грешака.
* Унапређена читљивост: Јасно дефинисани типови чине код лакшим за разумевање.
* Склабивост: Лакши за управљање великим базама кода због своје модуларне природе.
* Компатибилност: Беспрекорно се интегрише са постојећим ЈаваСцрипт библиотекама и оквирима.

*Мане TypeScript-a:*

TypeScript има неколико потенцијалних недостатака. Један од њих је дужи временски период за припрему извршавања програма, јер је потребно прво компајлирати TypeScript код у JavaScript. Због тога JavaScript може бити погоднији избор ако додатне функционалности TypeScript-а нису неопходне.

*Заједничко за JavaScript и TypeScript:*

Све што је TypeScript у основи је JavaScript, стога су оба објектно оријентисана језика која се користе за развој динамичког садржаја и интерактивности за веб сајтове. У суштини, TypeScript је само програмски језик који помаже програмерима да пишу бољи JavaScript код.

### Node.js

Node.js је вишеплатформско JavaScript радно окружење отвореног кода за извршавање JavaScript -а на серверској страни. Node.js омогућава да се JavaScript користи за скрипте на серверској страни које омогућавају да се садржај динамичних веб страница генерише на серверу пре него што се пошаље до веб прегледача корисника. (википедија)

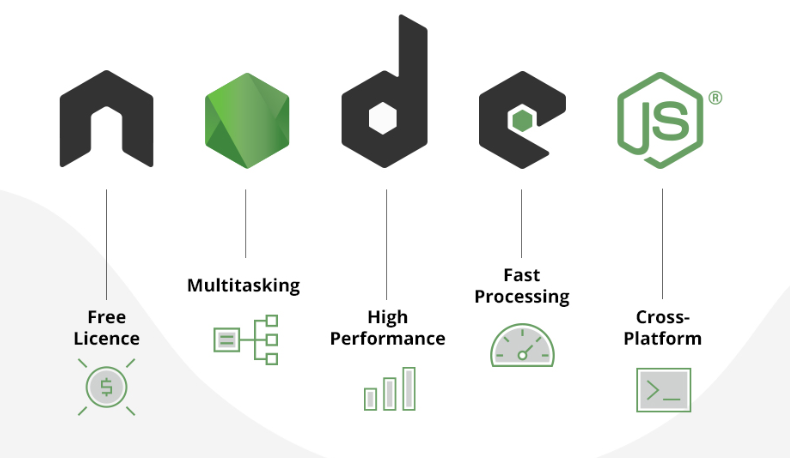
Креирао га је Ryan Dahl 2009. Године, када су сервери имали проблема са обрадом великог броја конекција.

JavaScript је углавном био повезан са веб претраживачима, а Node.js омогућава програмерима да програмирају и на клијентској и серверској страни користећи исти језик.

Node.js нуди бројне карактеристике које могу да одговарају специфичним потребама пројекта. Његове посебне карактеристике су:

* Асихроно програмирање: Оно омогућава истовремено обрађивање више веза и задатака без чекања да се операције заврше. Ово га чини посебно погодним за изградњу апликација у реалном времену.
* Развој на серверској страни: Може да обавља задатке као што су сервирање веб страница, обрада форми, интеракција са базама података и руковање АПИ захтевима.
* Високе перформансе: V8 JavaScript мотор, који покреће Google Chrome, нуди брзо извршавање и ефикасну употребу ресурса. Овај екосистем такође подржава више пакета и библиотека који доприносе и вишим перформансама.
* Модули и NPM: Node.js пружа модуларни приступ програмирању, где се код може организовати у поново употребљиве модуле. Node package manager (NPM) у Node.js омогућава инсталацију, управљање и дељење библиотека и пакета који побољшавају функционалност апликације, што значајно убрзава развој.
* Алати за командну линију: Node.js се може користити за изградњу алата и скрипти за командну линију, што омогућава аутоматизацију задатака, комуникацију са АПИ-има и обављање различитих задатака користећи JavaScript.

Node.js је одличан за рад у реалном времену са великом количином информација, омогућава прикупљање и визуализацију података у облику контролних табли. Поред тога веома је брз, чува податке у ЈSON формату, има велику заједницу програмера која га подржава, омогућава врхунски streaming података, подржан је од стране Linux фондације, нуди одличне опције hosting-a и погодан је за развој софтвера за различите платформе.



Slika 3

Многи гиганти у ИТ сфери користе Node.js za backend развој, што им доноси одличне перформансе апликација па и тим раст популарности. Међу великим компанијама које користе Node.js су Microsoft, Uber, PayPal, Amazon, Netflix, eBay, Intel и Linkedin.

Кроз овај пројекат за прикупљање и обраду података о тениским турнирима коришћењем библиотеке Puppteer и Mapbox АПИ-ja, Node.js омогућава:

* Руковање HTTP захтевима, односно слање HTTP захтева АPI-ју Mapbox-a како би се приступило географској локацији тениских терена, чувајући податке у JSON формату за даљу анализу и приказивање на мапи.
* Модул fs (File System) у Node.js-u омогућава читање, писање и рад са датотекама у систему, што је кључна функција пакета datafetch овог пројекта за ажурирање и смештање података о турнирима у ЈSON фајлове.
* Могућност асинхроног програмирања (Користећи async/await и callback функције), чиме се могу ефикасно преузимати и обрађивати подаци без блокирања апликације. Овај приступ је примењен у функцијама за приступ Mapbox АPI-ју и асихроно учитавање страница са Puppteer-ом.
* Коришћењем библиотеке Puppteer, Node.js омогућава контролу над Chrome прегледачем, отварање страница, преузимање података са сајтова и аутоматизовано чување тениских података у ЈSON формату

### React

React је open-source библиотека за JavaScript коју је развила компанија Facebook, а користи се првенствено за израду корисничких интерфејса, посебно за апликације са једном страницом (single-page applications). Представља једну од најпопуларнијих библиотека за израду веб апликација. Запослени компаније Facebook Tom Occhino укратко је објаснио зашто је React тако моћан алат: „React обавија императивни АПИ декларативним. Права снага React-а лежи у томе како те тера да пишеш код.“ (Fedosejev, 2015)

React значајно поједностављује писање кода чинећи га интуитивним и декларативним, уместо традиционалног, императивног стила. Уместо да се прецизно дају инструкције за сваки корак промене у апликацији, код једноставно описује како апликација треба да изгледа у одређеном стању, а React се брине за ажурирање у позадини. Оваквим приступом код постаје једноставнији, читљивији и лакши за одржавање.

На пример, у JavaScript-у ако је потребно да се промени текст унутар div елемента, код би изгледао овако:

document.getElementById('myDiv').textContent = 'Pozdrav';

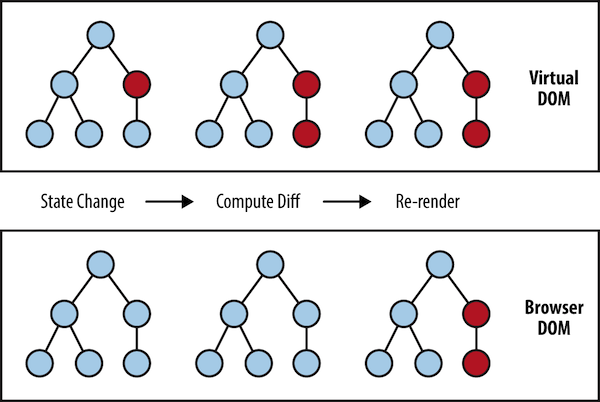
Међутим, у React-у, декларативни стил омогућава да једноставно дефинишемо стање и излаз који желимо. React аутоматски брине о ажурирању интерфејса, односно апликација са једном страницом учитава само један HTML документ приликом првог захтева а затим ажурира специфичан део, садржај или тело веб странице помоћу JavaScript-a.Овај образац је познат као рутирањена страни клијента јер клијент не мора поново да учитава целу веб страницу да би добио нову страницу сваки пут када корисник изврши захтев. Промена div елемента би у React-у изгледала овако:

const MyComponent = () => { return <div>Pozdrav</div>; };

Овај декларативни приступ ослобађа од ручног управљања DOM-oм (virtual Document Object Model) и омогућава фокусирање на стање апликације и логику, остављајући React-у да ажурира приказ. React се ослања на виртуелни DOM, што је копија стварног DOM-a.

DOM је HTML приказ веб странице у прегледачу. Када неки корисник ступи у интеракцију са веб страницом (то може бити клик на дугме, попуњавање обрасца итд.), прегледач ажурира DOM како би одразио те промене. Прегледач ће затим поново рендеровати страницу да би се приказао ажурирани HTML.

Виртуелни DOM с друге стране је апстракција правог DOM-а, коју креирају и одржавају JavaScript библиотеке попут React-a. Виртуелни DOM је копија правог DOM-a, што омогућава брже ажурирање и побољшане перформансе. Када се стање апликације промени, React прво ажурира виртуелни DOM, израчунава разлику („diff“) између старог и новог стања и само те промене примењује на стварни DOM. Овај се процес зове реконцилијација и значајно убрзава перформансе јер се минимизују непотребна ажурирања и манипулације стварног DOM-a. (Singh, 2023)



Slika 4 – Virtual Dom (<https://www.codecademy.com/resources/docs/react/virtual-dom>)

За разлику од других оквира као што је Angular, React не намеће строга правила за конвенције кодирања или организацију датотека. То значи да програмери и тимови имају слободу да поставе конвенције које им највише одговарају и имплементирају React на начин који сматрају најбољим.

Помоћу React-а може се креирати само једно дугме, неколико делова интерфејса, или целокупан кориснички интерфејс апликације. Можете се постепено усвојити и интегрисати у већ постојећу апликацију са мало интерактивности или, још боље, користити се за изградњу моћних React апликација од нуле, у зависности од потреба.

React обухвата неколико кључних концепата који га чине моћним и лаким за употребу (Choudhary, 2024):

* Компоненте – Представљају основне елементе сваке React апликације. Компоненте омогућавају развој интерфејса тако што деле кориснички интерфејс (UI) на независне, поново употребљиве делове, што значајно поједностављује организацију и управљање кодом. Овај модуларни приступ омогућава лако ажурирање и тестирање појединачних делова апликације.
* JSX (JavaScript XML) – Синтаксичка екстензија за JavaScript која дозвољава да се HTML код интегрише директно унутар JavaScript-а. Коришћење JSX-а чини код читљивијим и логичнијим, јер дозвољава програмерима да структуру корисничког интерфејса дефинишу директно унутар компоненти. Овај приступ помаже да се визуелни и логички аспекти апликације повежу и олакшава разумевање структуре кода.
* Стање (state) – Представља унутрашње податке компоненте који могу да се мењају током времена. Када дође до промене у state-у, React аутоматски поново рендерује компоненту са новим подацима, омогућавајући динамичко ажурирање UI-а без потребе за ручним интервенцијама, чиме се осигурава интерактивност апликације.
* Hooks – Специјалне функције које уводе могућност коришћења state-а и других функционалности у функционалним компонентама, чиме се оне изједначавају са класним компонентама у погледу могућности. Hooks, као што су useState и useEffect, олакшавају рад са state-ом и ефектима у апликацијама, омогућавајући приступ најновијим React функцијама (Olowatosin Isijola, 2023).
* Својства (props) – Омогућавају размену података између компоненти, што је од суштинског значаја за изградњу комплексних интерфејса. За разлику од state-а који представља интерно стање компоненте, props омогућавају прослеђивање података из родитељских у дечије компоненте, чиме се обезбеђује флексибилна и динамична структура апликације.

Поред тога што користи виртуални DOM, React нуди бројне предности које га чине идеалним за израду модерних веб апликација. Његова једноставна синтакса и релативно лако учење омогућавају тимовима да брзо савладају основе, док могућност поновне употребе компоненти смањује време развоја и одржава висок ниво перформанси. React такође унапређује SEO оптимизацију кроз server-side рендеринг, што смањује оптерећење клијентске стране и побољшава видљивост у претраживачима. Снажна заједница и константна подршка с бројним ресурсима и алатима доприносе популарности React-а, док једносмерно повезивање података омогућава лако праћење промена и чини ток података предвидљивијим. (Imran Hossain, 2022)



Slika 5 – Why Use ReactJS? (https://www.linkedin.com/pulse/why-use-react-js-imran-hossain/)

Захваљујући својим изузетним карактеристикама као што су висока ефикасност, брзина развоја, могућност додавања нових функција у реалном времену и одлична подршка за програмере, React се користи у око 46% веб сајтова данас (Machalica, 2024). Неколико великих компанија као што су *Facebook, Instagram, Airbnb, Netflix, Twitter, Pinterest, Dropbox, Discord, Reddit, PayPal* и *The New York Times* такође се ослањају на React у својим апликацијама. (Grzybowska, 2023)

### Библиотеке Puppeteer и ОpenLayers

**Puppeteer** је Node.js библиотека која пружа API високог нивоа за рад са Google Chrome-ом или Firefox-ом путем DevTools Protocol-а. Већина ствари које се ручно могу урадити у прегледачу, могу се обавити и коршћењем Puppeteer-a. (Pptr.dev, 2025) Примери за то су (Pptr.dev , 2025):

* Аутоматизовање слања формулара, тестирања корисничког интерфејса, унос тастатуре, итд.;
* Аутоматизовано креирање тест окружења коришћењем најновијих ЈavaScript и функција прегледача;
* Снимање временске траке сајта ради дијагнозе проблема са перформансама
* Тестирање Chrome екстензија;
* Генерисање снимака екрана и PDF-ovе страница;
* Претрага SPA (једнаоструке веб апликације) и генерисање пред-рендерованог садржаја. У овом пројекту Puppeteer се користи за прикупљање информација са сајтова тениских турнира и уписивање тих података у JSON фајлове. Он омогућава аутоматизован приступ различитим страницама са детаљима везаних за турнире, претраживање или „скреповање“ специфичних података као и упис прикупљених података у JSON фајлове.

JSON је идеалан формат за складиштење оваквих података јер je лак за читање и може се једноставно обрадити у различитим окружењима.

**OpenLayers**је JavaScript библиотека отвореног кода за креирање било које апликације за веб мапирање. Пружа АПИ за изградњу богатих географских веб апликација сличних Google Maps и Bing Maps. Основне карактеристике OpenLayers библиотеке укључују (Antonio Santiago Perez, 2012) :

* *Подршка за више извора података*: Омогућава коришћење широког спектра комерцијалних и бесплатних извора слика и векторских података;
* *Интеракција са мапама*: Пружа могућности за интеракцију са мапама, укључујући зумирање, померање и селектовање објеката;
* *Компатибилност*: Рад на свим модерним прегледачима, као што су *Chrome, Firefox, Safari и Edge*, уз подршку за мобилне уређаје;
* *Могућности проширења*: Омогућава додавање прилагођених слојева и интеракција.

## УПОТРЕБА И ИНТЕГРАЦИЈА АПИ-ЈА У АПЛИКАЦИЈИ

### Основни појмови и принципи функционисања

АПИ (Application Programming Interface) представља интерфејс који омогућава комуникацију између различитих софтверских система. То је скуп правила, протокола и алата који омогућавају различитим програмима да комуницирају, размењују податке и функционалности.

АПИ омогућава софтверским компонентама да се повезују и комуницирају, користећи унапред договорене поруке. За разлику од корисничког интерфејса који је намењен људима, повезује рачунаре или делове софтверских система међусобно. На овај начин, програмери користе АПИ-је за интеграцију различитих компоненти у већу целину.

Карактеристике АПИ-ја

* Методе (крајње тачке): АПИ се састоји од различитих метода које омогућавају комуникацију са другим системима. Ове методе, познате и као захтеви или крајње тачке (endpoints), дефинишу како програмер може да приступи различитим функционалностима.
* Сакривање унутрашњих детаља: Дизајниран је тако да сакрије унутрашње детаље рада система и излаже само оне функције које ће бити корисне програмерима.
* Интерфејс за комуникацију: Омогућава комуникацију између различитих компоненти без потребе за директним контактом између крајњег корисника и система.

Типови АПИ-ја

* Библиотеке и framework-ови: Ови АПИ-ји описују интерфејс према софтверским библиотекама или framework-овима. Пружају алате и функције које програмери користе за креирање апликација.
* Оперативни системи: Дефинишу интерфејс између апликација и оперативних система, као што је POSIX, који омогућава приступ системским ресурсима.
* Даљински АПИ-ји: Омогућавају програмерима да манипулишу удаљеним ресурсима, користећи стандарде као што су SOAP или RPC (Remote Procedure Call).
* Веб АПИ-ји: Омогућавају интеракцију између апликација и услуга путем интернета, користећи HTTP протоколе. Web АПИ-ји су најпопуларнији и често користе формате као што су JSON или XML за размену података.

Процес функционисања АПИ-ја

АПИ омогућава комуникацију између различитих компоненти или система путем захтева и одговора, што изгледа овако:

Захтев: Програм (клијент) шаље HTTP захтев серверу (или другом систему) уз помоћ једне од HTTP метода као што су GET, POST, PUT или DELETE.

Одговор: Сервер одговара на захтев, најчешће са подацима у JSON или XML формату, као и са статусним кодом који указује на успех или неуспех захтева (нпр. 200 - успех, 404 - није пронађено, 500 - серверска грешка).

### Веб АПИ-ји коришћени у пројекту

Веб АПИ-ји омогућавају апликацијама да комуницирају преко интернета користећи стандарде као што су HTTP, HTTPS, SOAP или REST. Најпопуларнији су RESTful веб АПИ-ји који користе HTTP протоколе и једноставан формат за размену података (најчешће JSON). Веб АПИ-ји су омогућили развој различитих интеграција, као што су интеграција апликација са друштвеним мрежама, системима за плаћање, базама података и слично. У овом пројекту коришћена су два веб АПИ-ја: **Mapbox Geocoding АПИ** и **ИТФ АПИ**.

**Mapbox АPI** представља платформу која омогућава рад са геолокацијом, картама и визуализацијом података на мапи. То је моћна услуга која омогућава програмерима да интегришу мапу у своју апликацију и да раде са географски повезаним подацима. Mapbox Geocoding АПИ је коришћен у овој апликацији и он омогућава апликацијама да конвертују називе места у географске координате (ширину и дужину) као и обрнуто. У апликацији „Тениски турнири на мапи“ овај АПИ се користи за преузимање координата за места са листе. На основу назива места се шаље HTTP захтев ка Mapbox АПИ-ју, који одговара враћајући JSON фајл са подацима о тој локацији, укључујући координате које се користе даље у апликацији.

За разлику од јавних АПИ-ја, **ИТФ АПИ** се такође користи у овом пројекту. Он представља интерно коришћен АПИ на званичној ITF страници, којем се може приступити кроз мрежне захтеве, али није званично објављен или документован за широку употребу. Овај АПИ је технички доступан путем url-а, али није намењен као јавни сервис. ITF АПИ се користи за преузимање података о турнирима који се одржавају током године, за различите категорије. Након што се подаци преузму они се чувају у JSON фајловима и даље користе у апликацији.

## ПРЕГЛЕД ПОСТОЈЕЋИХ СОФТВЕРСКИХ РЕШЕЊА У ДОМЕНУ ТЕНИСКИХ ТУРНИРА

У данашњем дигиталном добу, информације о тениским турнирима су доступне на различитим платформама, од званичних сајтова до апликација које пружају информације о резултатима у реалном времену. Расутост информација може захтевати од корисника претраживање више извора како би пронашли детаље које их интересују. Постојећа решења јесу званични сајтови ITF (https://www.itftennis.com/en/) и Тennis Europe (https://www.tenniseurope.org/) који пружају информације о турнирима, али којима недостаје визуализација и напредне функционалности попут интерактивног приказа на мапи.

Овај одељак је усмерен на истраживање тренутно доступних софтверских решења у домену тениских турнира, са фокусом на њихове предности и мане. Циљ јесте да се идентификују празнине које ново окружење решење може попунити, наручито у погледу пружања побољшаног корисничког искуства кроз визуализацију података на мапи и увођење фунција које омогућавају корисницима да лакше и брже приступе релативним информацијама о турнирима.

### Анализа постојећих решења

Међународна тениска федерација (ITF) је светска тениска федерација, одговорна за регулисање и промоцију овог спорта на глобалном нивоу. Tennis Europe (TE) је главна организација која управља тенисом на јуниорском нивоу у Европи. Она је део шире структуре Међународне тениске федерације (ITF) и фокусира се на развој и промоцију тениса у европским земљама. Они имају своје званичне сајтове који представљају софтверска решења за промоцију и праћење тениских турнира на глобалном и европском нивоу. Ова два сајта представљају изворе података за ову апликацију и као такви пружају информације о свим постојећим турнирима уз функционалности филтрирања.

Такође сајтови попут Tennis (https://www.tennis.com/) и Tennis TV (https://www.tennistv.com/) омогућавају преглед ATP и WTA турнира за текућу годину.

### Поређење решења

Предности већ постојећих софтверских решења је тај што нуде свеобухватне информације о турнирима, играчима и рангирањима. Њихове функционалности рангирања омогућавају преглед турнира по различитим критеријумима. Оно што постоји јесу текстуалне листе и табеле без интегрисаних мапа па се може рећи да је мана постојећих софтверских решења недостатак визуализације, посебно у погледу непостојања интерактивних мапа које омогућавају бољи преглед уз динамичке филтере.

Кориснички интерфејс је интуитиван за тениске професионалце, али може бити комплекснији за повремене кориснике који желе брз и лак приступ информацијама. Фрагментација информација тера кориснике да посећују више различитих сајтова како би добили свеобухватну слику о турнирима. Уместо тога, корисници уз веб апликацију „Тениски турнири на мапи“ имају користи од интерфејса који визуелно приказује све турнире на једној интерактивној мапи. Овај приступ омогућава да се сви релевантни догађаји налазе на једној страници.

Са оваквим решењем, корисници би могли да кликну на било који турнир директно на мапи који би одвео до званичног сајта где би добили све потребне додатне информације поред оних које директно виде на страници. Ова врста визуелизације значајно поједностављује корисничко искуство, омогућавајући корисницима да лако пронађу и прате турнире који их занимају, све на једној интегрисаној платформи.

Свеобухватан приказ тениских турнира на мапи не постоји, тако да се овај пројекат не може упоредити са нечим што има исту идеју, већ представља јединствену иновацију у области тениских турнира.

### Предност софтверског решења „Тениски турнири на мапи“

Ова веб апликација представља јединствен приступ који обједињује информације из различитих извора и омогућава корисницима да на једноставан и прегледан начин прате турнире. Док постоје појединачне платформе које пружају информације о тениским турнирима, ниједна не нуди интегрисано решење са интерактивном мапом и напредним филтрирањем.

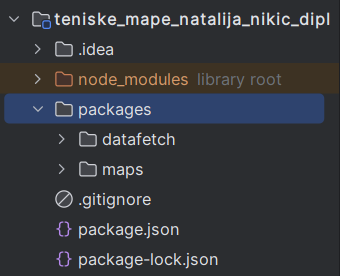
Веб апликација Тениски турнири на мапи не само да попуњава ту празнину, већ и пружа значајну вредност како професионалним тенисерима који учествују на турнирима нижих категорија, тако и љубитељима тениса који желе да прате те догађаје. Поред тога, платформа омогућава корисницима да добију све релевантне информације на једном месту, чиме се смањује потреба за преласком са једног извора на други, што знатно побољшава корисничко искуство.

Овакав приступ значајно олакшава праћење и планирање учешћа на турнирима, што је од великог значаја за младе тенисере као и њихове тренере. Пројекат „Тениски турнири на мапи“ поставља нове стандарде у информисању и праћењу тениских турнира, и тиме доприноси даљем развоју и популаризацији овог спорта на глобалном нивоу.

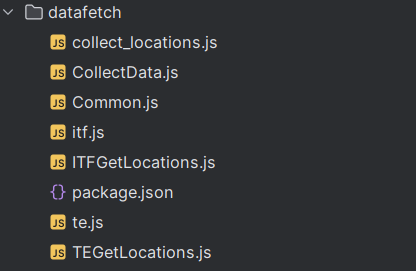
# ИМПЛЕМЕНТАЦИЈА

## СТРУКТУРА КОДА

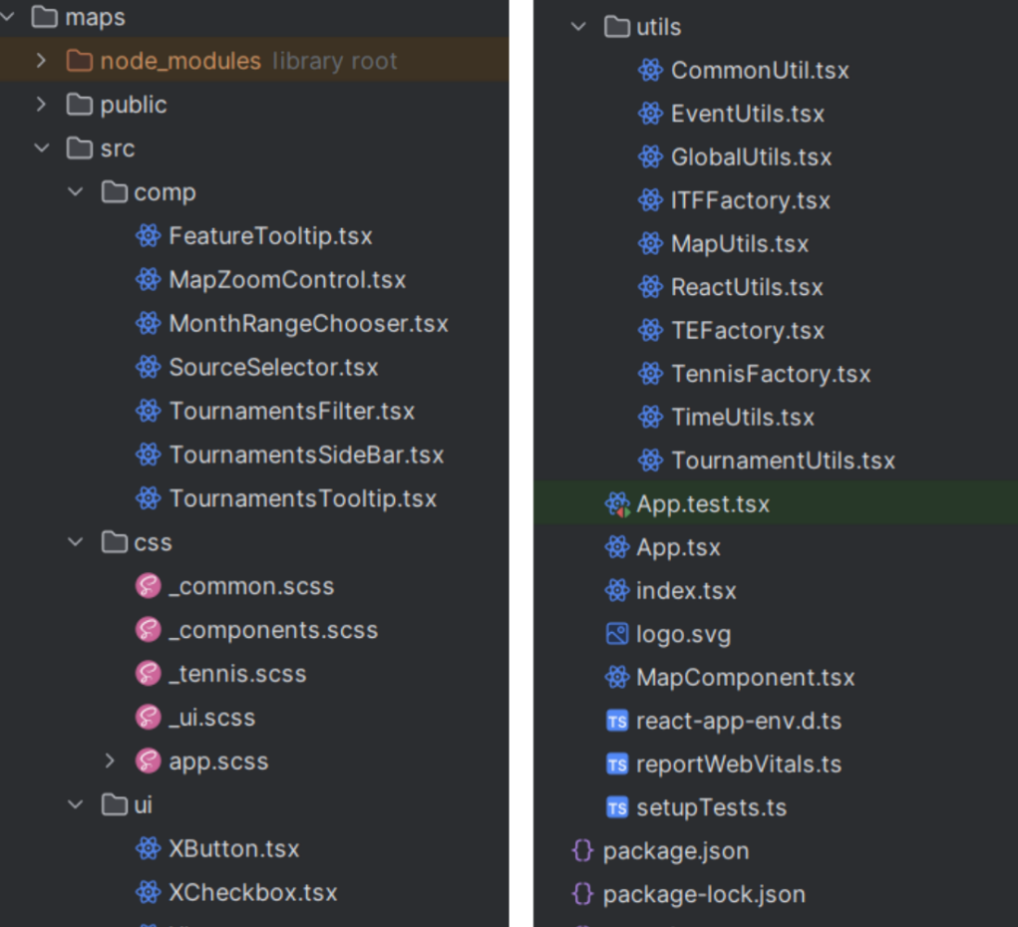
Апликација је развијена у IntelliJ развојном окружењу, што је омогућило брзо и ефикасно кодирање. Иако је IntelliJ пружио изузетну подршку у развојном процесу, пројекат је могао бити имплементиран и у другим популарним окружењима, с обзиром на популарности коришћених технологија и широке подршке за исте. Структура пројекта је организована тако да је код који се бави екстракцијом података смештен у фолдеру datafetch, док је код који имплементира интерфејс помоћу React-а постављен у фолдеру maps. Оба фолдера су организована унутар главног фолдера packages, чиме се јасно одвајају различити делови пројекта и олакшава навигација.



Фолдер **datafetch**, који је задужен за екстракцију података са веб сајтова, садржи одговарајуће JavaScript фајлове са прилагођеним називима који јасно одражавају функције које сваки фајл обавља.



Други фолдер, **maps**, који је одговоран за приказивање свих екстракованих података, садржи структуру која је организована на следећи начин: унутар фолдера public налазе се слике као и фолдер *data*, у којем се чувају JSON фајлови који се генеришу покретањем кода из претходног дела апликације. Фолдер maps садржи и фолдер *comp*, који обухвата React компоненте, као и фолдеeа *ui* и *utils* који су задужени за интерфејс и помоћне функције апликације. MapComponents је главна компонента у апликацији, јер је она која се учитава у Аpp компоненту и одговорна је за приказивање и управљање свим подацима и интеракцијама на мапи. Такође, фолдер maps садржи и фолдер *css*, у којем се налазе SCSS фајлови који се користе за стилизовање апликације.



## ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ СТРАНИЦА

Идеја овог дела пројекта јесте засебан програм који врши екстракцију података са два официјална сајта *ITF Tennis (ИТФ)* и *Tennis Europe (ТЕ).* С обзиром на структуре ових сајтова које су различите, потребно их је истражити и изабрати методе које су доступне и најефикасније за прикупљање података.

Како би се прикупили подаци са ових сајтова, односно извршила екстракција података о тениским турнирима, коришћен је Puppeteer, алат за аутоматско управљање browser-ом који омогућава рад са ЈS-ом на веб страницама, као и интеракцију са садржајем страница без потребе за ручним уносом података. На почетку кода су увезена три пакета: puppeteer-core, find-chrome-bin и **fs** (File System).

### Модул зависности

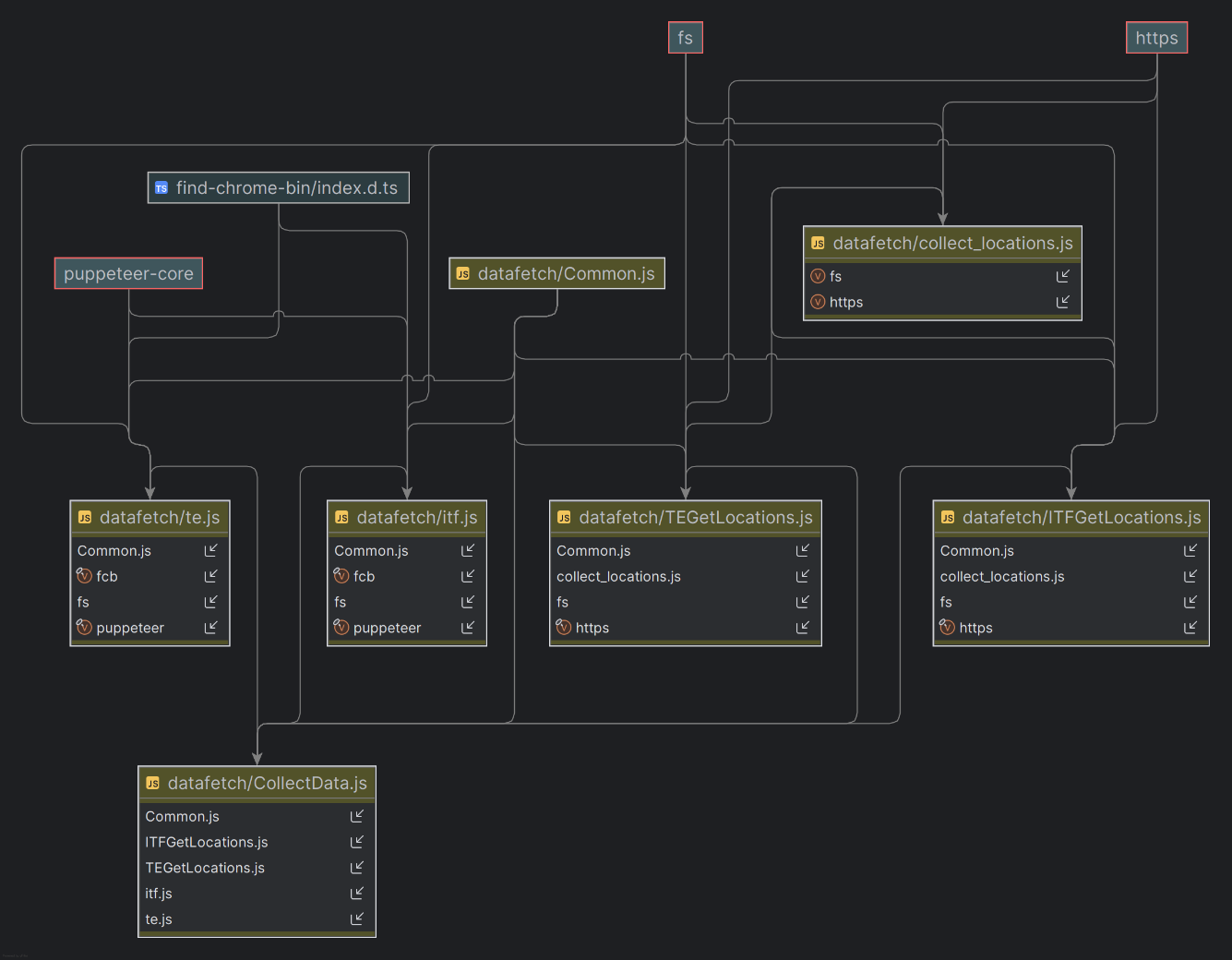
Дијаграм на слици испод приказује зависности које постоје између различитих модула дела пројекта у оквиру фолдера datafetch који је задужен за екстракцију података и њихово смештање у JSON фајлове. Овај дијаграм је кључан за разумевање структуре и међусобних зависности различитих модула унутар пројекта. Омогућава нам бржи увид у начин на који модули комуницирају и деле ресурсе.

Може се увидети како index.d.ts (TypeScript датотека) зависи од puppeteer-core. Разлог што ова датотека користи Puppeteer Core за интеракцију са Chrome претраживачем је та што Puppeteer омогућава контролу претраживача АПИ-ја, што је кључно за аутоматизацију и екстракција података са веб страница.

Ту је и основни модул puppeteer-core као заједничка зависност, јер она представља основну библиотеку коју користе различите датотеке унутар пројекта. Омогућава им извршавање задатака као што је отварање страница, слање захтева и преузимање података.

Дијаграм приказује и зависност од fs модула који управља датотекама, односно обавља читање и писање података у JSON формату и https модула који се користи за извршење сигурних http захтева, што је неопходно за преузимање података са веб-а.

У целости, овакве зависности омогућавају модуларност и флексибилност пројекта, олакшавајући одржавање кода и развој нових функционалности.



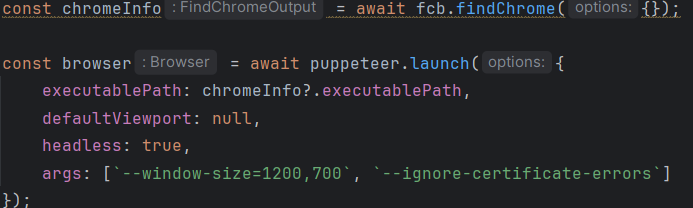
### Екстракција података и форматирање у ЈSON фајлове

Као што је већ раније поменуто, приликом прикупљања података са веб страница, важно је пажљиво анализирати те странице како би се одабрао најприкладнији и најефикаснији начин за екстракцију података. Сваки сајт је другачији, па тако се и код за прикупљање информација са ИТФ и ТЕ веб страница разликују. У оба фајла се користи Puppeteer библиотека и подаци се смештају у JSON фајлове. Процес се састоји из неколико кључних корака: проналажење Chrome претраживача, отварање страница, екстракција релевантних података, складиштење података у JSON формату, и затварање претраживача.

* Проналажење Chrome претраживача и покретање Puppeteer-a

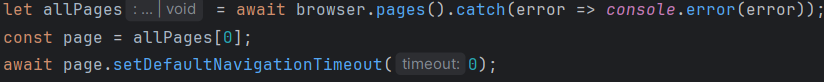
Коришћење find-chrome-bin за проналажење инсталираног Chrome и затим покретање Puppeteer-a:





* Навигација до циљних страница

Отварање страница са подацима и постављање навигационих подешавања:

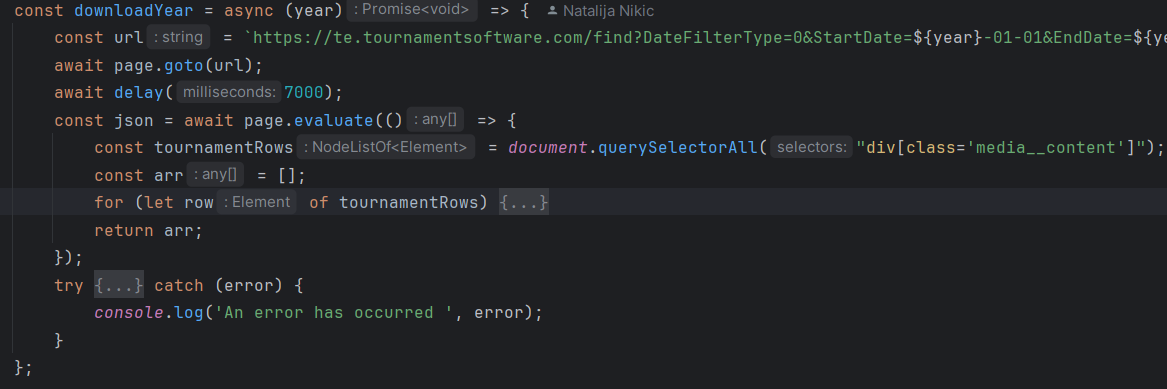


* Екстракција података са страница

За ИТФ податке, користи се **page.evaluate** функција да би се извукли подаци са странице. Захваљујући доступном АПИ-ју лако се скидају подаци у JSON формату.

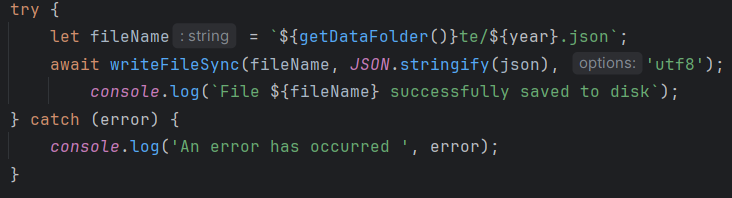


Za TE податке, екстракција података се врши уз помоћ библиотеке Puppeteer. Puppeteer омогућава отварање страница, чекање на учитавање података и прикупљање информација као што су назив турнира, URL, локација, датуми, површина, категорија и узрасна група, које се затим смештају у JSON објекат. Подаци се такође екстрактују тако што се користи **page.evaluate** функција, унутар које је важно напоменути да се селектују сви редови које садрже информације о турнирима. Итерирањем кроз сваки ред подаци се смештају у низ.

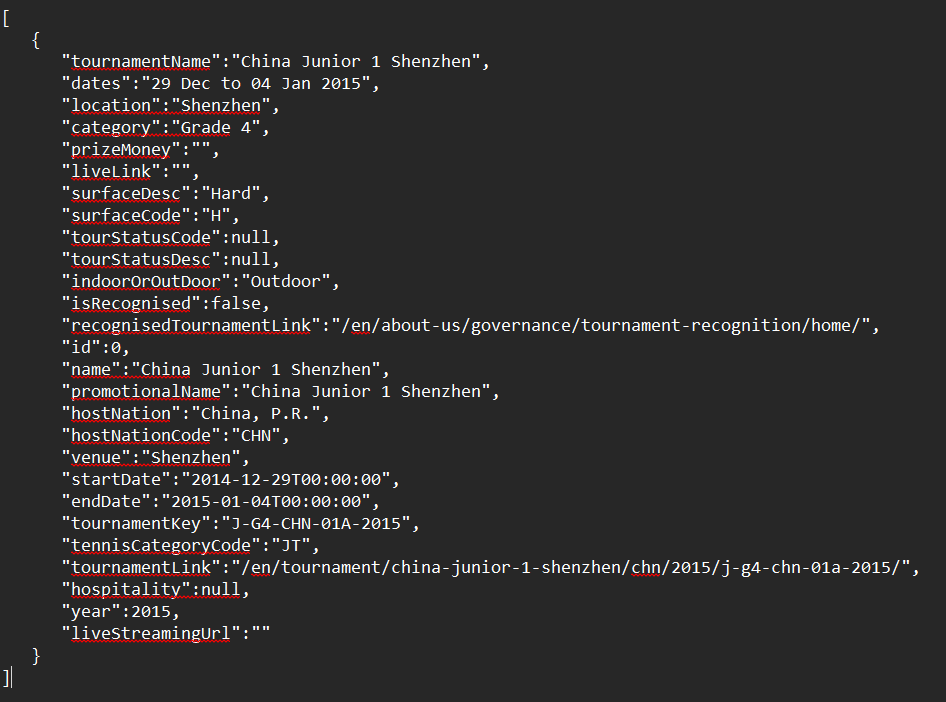


* Складиштење података у JSON формат

Након што су подаци прикупљени и формирани у JSON структуру, користе се фунције **writeFileSync** из fs модула за писање података у JSON фајл. Називи фајлова се генеришу на основу године и директоријума за податке код ТЕ турнира, а на основу врсте турнира и године и директоријума за податке код ИТФ турнира.



Пример формата валидног JSON фајла:

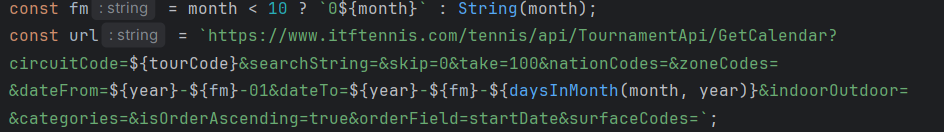


### Конфигурација АПИ-ја

* Конфигурација приватног АПИ-ја:

За ИТФ турнире са званичне веб странице коришћен је АПИ endpoint. Функција **downloadMonth** је одговорна за конфигурацију url-a за АПИ позив, слање захтева и обраду одговора.

Функција downloadMonth прима три параметра: month (месец), year (година) i tourCode (код за врсту турнира, подразумевана вредност је "JT").



АПИ позив се генерише на основу прослеђених параметара url-а.

**await page.goto(url);**

Подаци који стигну као одговор на АПИ позив се анализирају помоћу **page.evaluate.** Ова функција се извршава унутар контекста странице, где се користи **document.querySelector("body").innerText** како би прочитала садржај тела одговора као текст и затим га парсира у JSON формат.

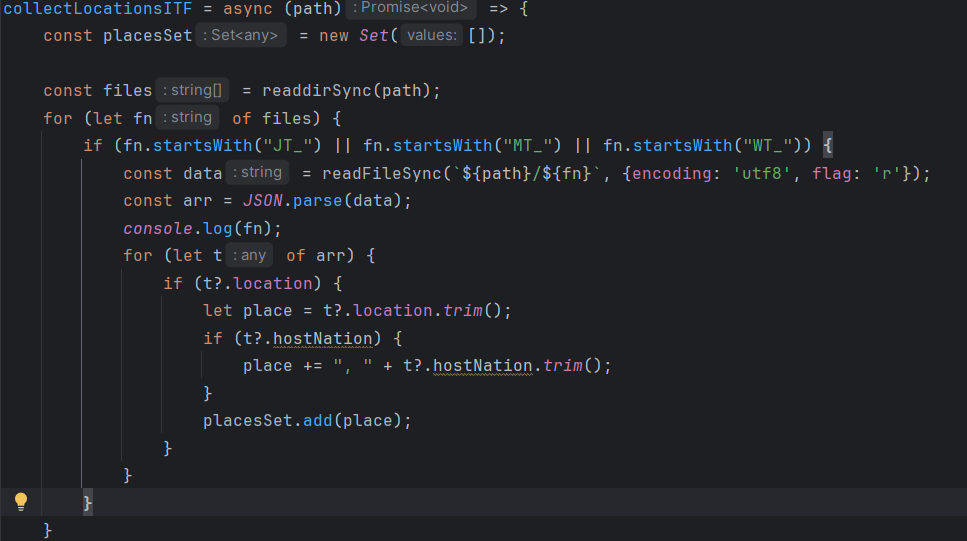
Прикупљени подаци о турнирима се додају у низ **allTournaments**, који садржи све турнире преузете за одређени месец и годину.

Коришћење ИТФ АПИ-ја омогућава аутоматизовано прикупљање прецизних и ажурираних информација о турнирима директно са званичне web странице. Ови подаци укључују назив турнира, датуме почетка и завршетка, локацију, површину терена, категорију, узрасну групу и URL ка детаљним информацијама о турниру. АПИ олакшава интеграцију ових података у апликације или системе за анализу, што омогућава бржу и ефикаснију обраду и приказивање информација о тениским турнирима.

* Конфигурација Mapbox AПИ-ја:

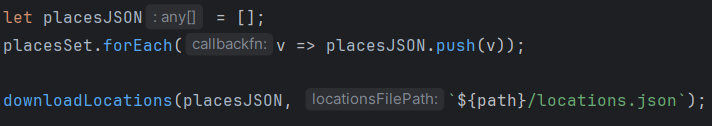
Други коришћени АПИ је **Mapbox Geocoding АПИ**. Након што су подаци екстрактовани са веб сајтова, било је потребно преузети географске координате за дате локације. За ову сврху, користи се функција **downloadLocations** која шаље захтеве Mapbox АПИ-ју, а затим обрађује добијене податке.

Прво је било потребно да се помоћу функција collectLocationsITF и collectLocationsTE из одређених директоријума добијених претходним складиштењем прочитају. Обрађују су задати фајлови (нпр. само фајлови чија имена почињу са "JT\_", "MT\_" или "WT\_" ако су у питању ITF турнири) и резултат учитавања је скуп свих пронађених локација. То се ради тако што за сваки ред у JSON фајловима се извлачи вредност поља location. Ако постоји поље hostNation, тај податак се додаје на крај имена локације. Скуп **placesSet** садржи све јединствене локације у формату "Локација, Држава".





Скуп placeSet се конвертује у низ placesJSON, који садржи све локације у формату погодном за обраду и представља излаз који је спреман за слање у функцију downloadLocations.



У наставку је објашњен процес конфигурације и коришћења Mapbox АПИ-ја:

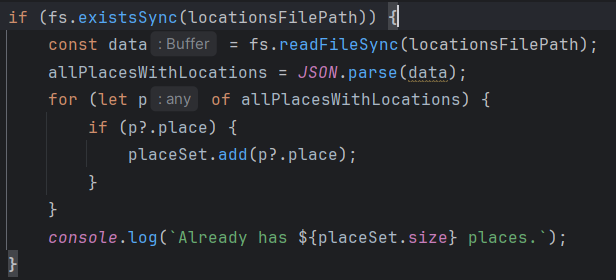
* Добијање Access Token-а:

За коришћење Mapbox Geocoding АПИ-ја неопходно је креирати налог на [Mapbox](https://www.mapbox.com/).

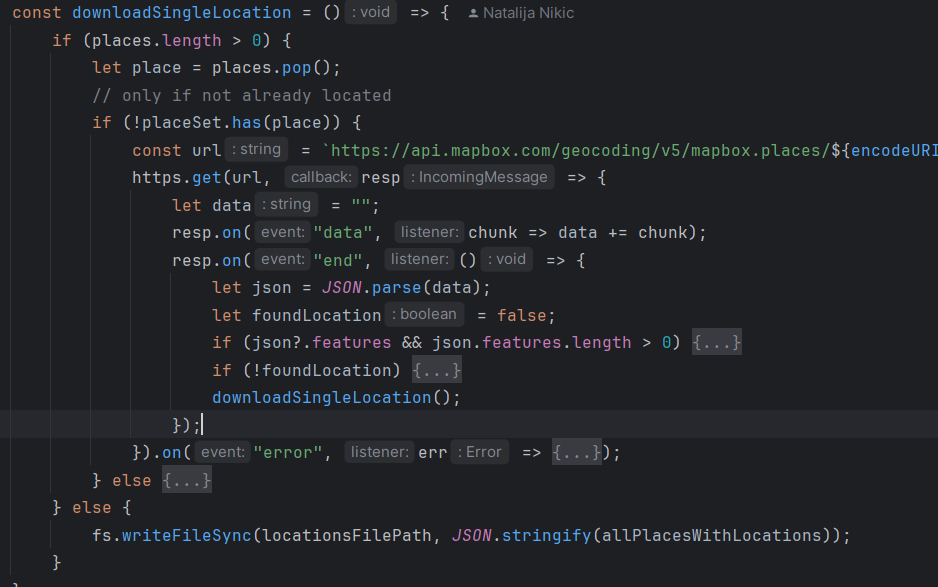
Након регистрације, у корисничком профилу може се креирати нови access token, који ће бити потребан за аутентикацију приликом слања захтева.

* Конфигурација АPI-ја:

Функција downloadLocations прихвата низ places који садржи имена локација и locationsFilePath где ће се сачувати резултати. За сваки елемент из низа, проверава се да ли већ постоји у placeSet.



Ако не постоји, прави се захтев према Mapbox Geocoding АПИ-ју. URL захтева садржи име локације које је енкодирано коришћењем encodeURI, и параметар access\_token са вредношћу Access Token-а. Одговор од АПИ-ја се обрађује и ако је пронађена локација, њене координате се додају у allPlacesWithLocations. На крају, резултати се чувају у locationsFilePath.



* Захтеви за АПИ:

URL захтева треба да буде у формату:

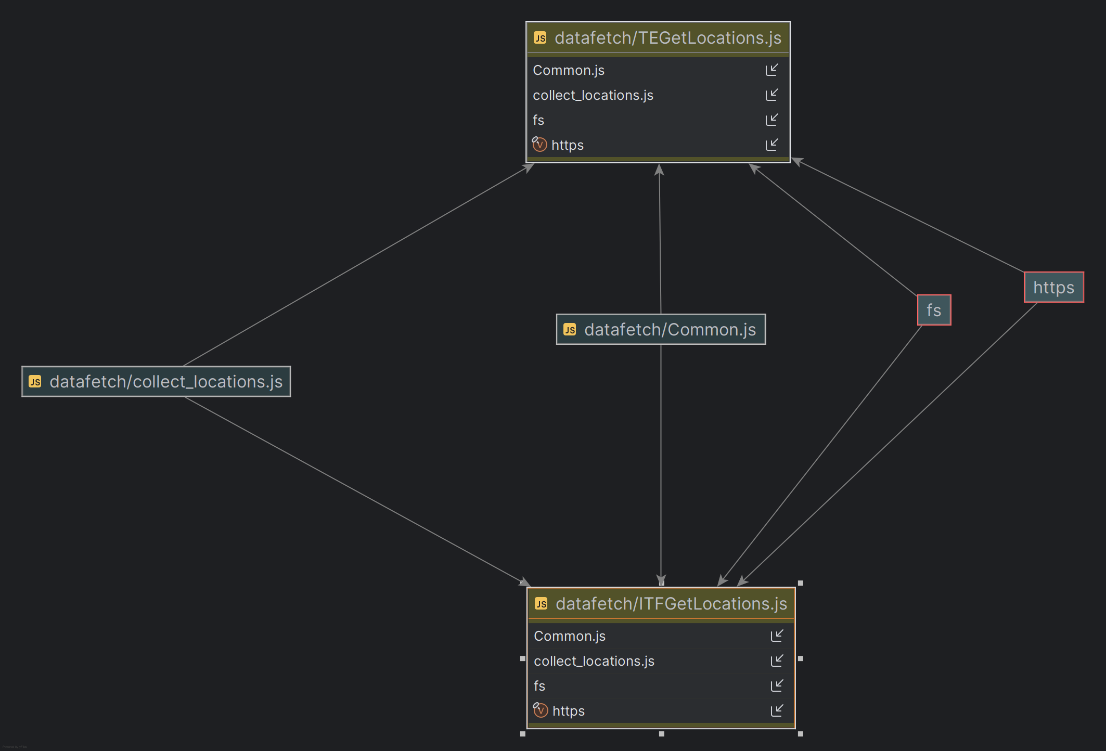
[**https://АПИ.mapbox.com/geocoding/v5/mapbox.places/{place}.json?limit=1&access\_token=YOUR\_ACCESS\_TOKEN**](https://АПИ.mapbox.com/geocoding/v5/mapbox.places/%7bplace%7d.json?limit=1&access_token=YOUR_ACCESS_TOKEN)

* Чување резултата:

Након успешног преузимања координата, резултати се чувају у JSON формату у наведеном фајлу (locationsFilePath).

Овај процес омогућава аутоматско преузимање географских координата за велике листе локација, што је веома корисно за географски оријентисане апликације попут ове.

Коначан приказ дијаграма зависности JS фајлова задужених за овај део пројекта је дат на слици испод.



## ИНТЕГРАЦИЈА И ВИЗУАЛИЗАЦИЈА ПОДАТАКА

У овом пројекту у циљу визуализације прикупљених података на мапи коришћен је React и OpenLayers библиотека. React је изабран за развој због брзог и ефикасног управљања корисничким интерфејсом (UI) и динамичког ажурирања података на мапи без потребе за поновним учитавањем странице. Такође React је коришћен у комбинацији са ТypeScript-ом како би се дефинисали типови за податке на мапи. OpenLayers као моћна библиотека је изабрана због тога што подржава широк спектар географских формата и омогућава лако интегрисање различитих слојева на мапи.

### Дрво компоненти

### Коришћење OpenLayers библиотеке

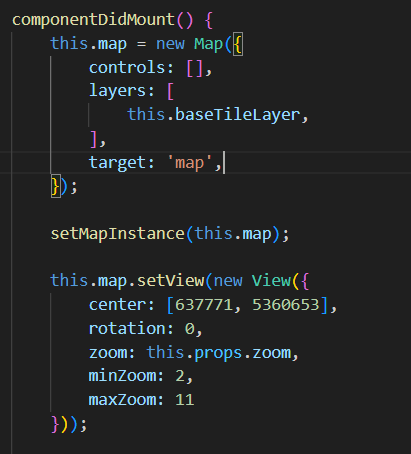
За коришћење OpenLayers библиотеке, потребно је увести њене различите делове које су потребне за функције које се користе. Најважнији импорти:

* import Map from "ol/Map"; (Маp креира нову мапу)
* import TileLayer from "ol/layer/Tile"; (TileLayer додаје слојеве на мапу)
* import \* as Proj from "ol/proj"; (Proj садржи функције за рад са координатама)
* import {Feature, View} from "ol"; (Feature представља геометријски објекат на мапи, а View дефинише приказ мапе, укључујући центар, зумирање и ротацију)
* import OLXYZSource from "ol/source/XYZ"; (OLXYZSource је извор података за плочице са XYZ координатама)
* import OLSource from "ol/source/Source"; (OlSource је основна класа за изворе података)
* import {Cluster, Vector} from "ol/source"; (Cluster креира извор података који групише блиске тачке ради боље прегледности на мапи, a Vector креира нови векторски извор података који садржи геометријске објекте као што су тачке, линије или маркери)
* import VectorLayer from "ol/layer/Vector"; (VectorLayer додаје векторски слој на мапу, што омогућава приказивање геометријских објеката)
* import {Point} from "ol/geom"; (Point представља тачку на мапи)
* import {Extent} from "ol/extent"; (Extent представља правоугаони опсег координата, односно подручје које је тренутно видљиво на мапи)
* import Overlay from "ol/Overlay"; (Оverlay се користи за приказивање додатних информација на мапи, као што су tooltipovi)

Функционалности коришћене из OpenLayers библиотеке:

1. Креирање мапе

У componentDidMount методи, креира се нова мапа са основним слојем (ТyleLayer) и поставља се приказ (View) са центром, зумирањем и ротацијом.



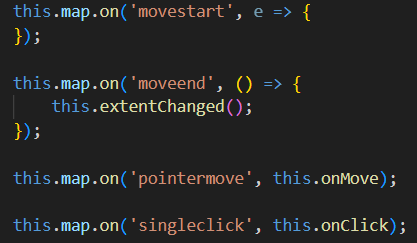
1. Додавање слојева

Додају се векторски слојеви и кластери за приказивање тачака на мапи. Proj.fromLonLat конвертује географске координате (longitude, latitude) u координатни систем коришћен од стране OpenLayers.



1. Интеракција са мапом

Додају се event listener-и за различите догађаје на мапи као што су померање, клик и померање показивача.



‘movestart’ додаје event listener за почетак померања мапе.

‘moveend’ додаје event listener за крај померања мапе.

‘pointermove’ додаје event listener за померање покаѕивача миша.

‘singleclick’ додаје event listener за клик на мапу.

1. Tooltip и Overlay

Приказују се додатне информације на мапи користећи Overlay када се корисник помери показивачем преко тачке или кликне на њу.



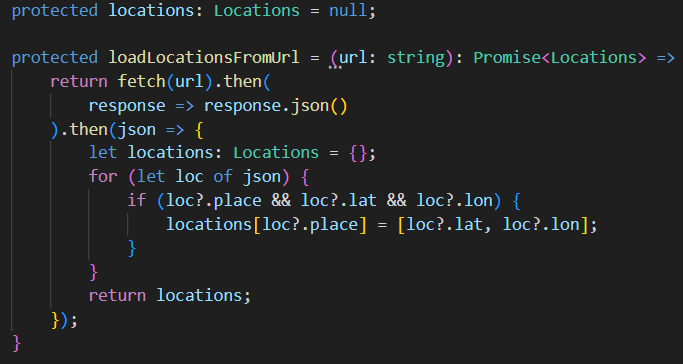
### Управљање подацима о тениским турнирима

Након што су екстракцијом података са званичних сајтова подаци прикупљени и смештени у ЈSON фајлове, што је био први део овог пројекта, потребно је да се информације о турнирима прикажу на мапи и на бочном панелу.

Имплементација кода је организована у фолдеру Utils, у TypeScript фајловима ITFFactory, TEFactory, TennisFactory и TournamentUtils.

У TennisFactory фајлу је дефинисана апстрактна класа TennisFactory која пружа флексибилан и проширив оквир за учитавање и управљање подацима о тениским турнирима. Подкласе ITFFactory и TEFactory имплементирају апстрактне методе и попуњавају конкретне податке.

TennisFactory класа садржи заштићене (protected) методе од којих **loadLocationsFromUrl()** учитава локације из url-а. Метода дохтава JSON податке са прослеђеног url-a, затим тај JSON парсира у објекат Locations и враћа Promise<Locations> који садржи учитане локације.



Апстрактне методе које имплементирају подкласе су:

**loadLocations()** – учитава примарне локације турнира.

**loadLocationPatches()** – учитава евентуалне додатне корекције локација.

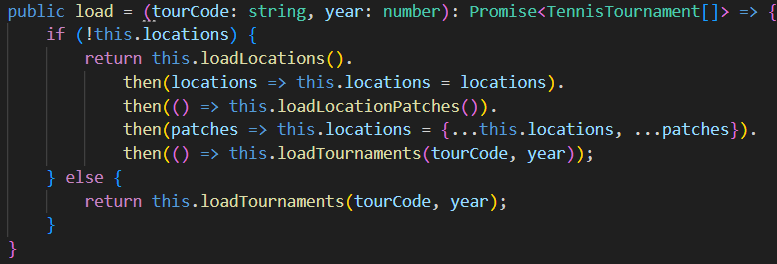
**fetchCalendarData(tourCode, year)** – дохвата календар турнира из спољног извора.

**createTournament(json)** – креира објекат TennisTournament на основу JSON података

Метода **loadTournaments()** учитава турнире тако што најпре позива **fetchCalendarData()** како би преузела њихову листу. Након тога, за сваки турнир креира се објекат TennisTournament. На крају, метода враћа јединствену листу турнира, без дупликата.

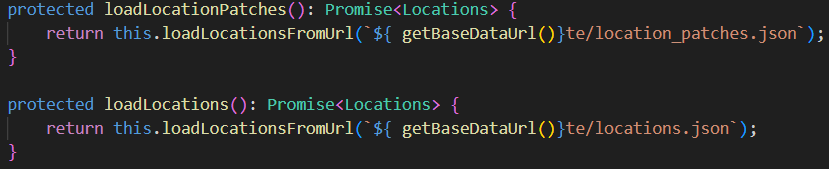


Главна метода за учитавање је **load()** и она омогућава кеширање учитаних локација како би се избегло вишеструко преузимање. Ако locations није учитан (null), учитава га из loadLocations(), затим примењује корекције из loadLocationPatches(), и тек онда преузима податке о турнирима. Ако locations већ постоји, директно преузима турнире (loadTournaments()).



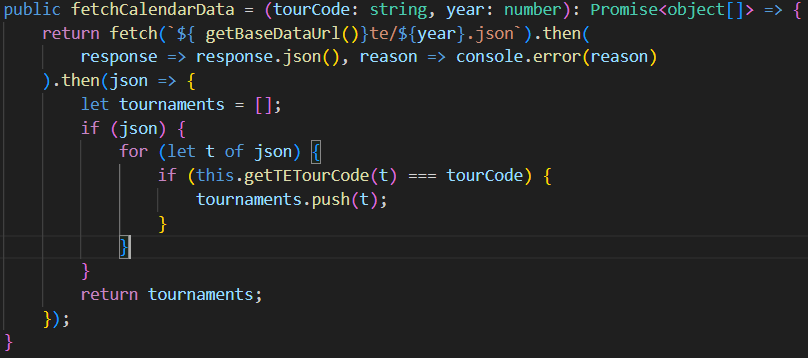
TEFactory и ITFFactory наслеђујu TennisFactory и додајu специфичне имплементације.

Дакле**, loadLocations()** учитава податке о локацијама из JSON-а. Методе користе loadLocationsFromUrl(), која чита JSON и форматира га у мапу (Locations).

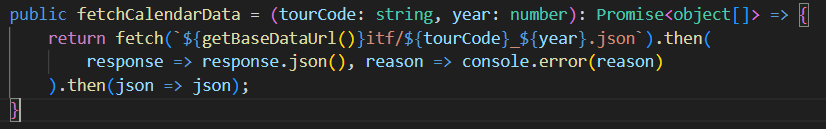
****

* Метода **fetchCalendarData():**

**За ТЕ**: Прави захтев за добијање података о турнирима (getBaseDataUrl() + "te/" + year + ".json") и затим филтрира те турнире на основу категорије (tourCode), враћајући само оне који припадају тој категорији.



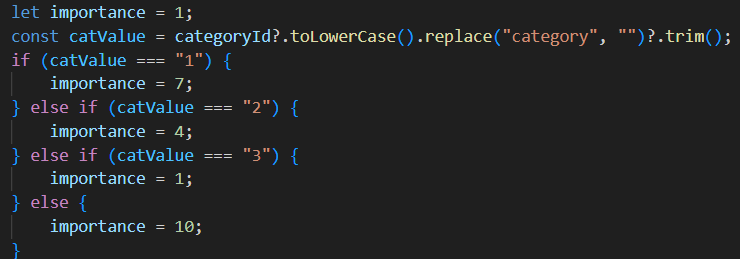
**За ITF**: Прави захтев за добијање података о турнирима за одређени код турнира и годину (getBaseDataUrl() + "itf/" + tourCode + "\_" + year + ".json"), али не филтрира податке, већ враћа све податке добијене из JSON-а.



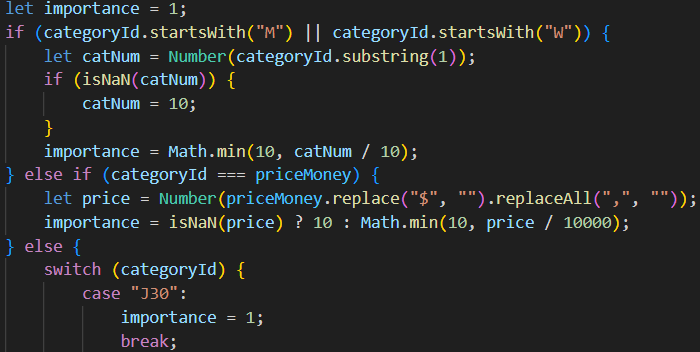
Обе методе користе fetch за асинхрону обраду података, али разлика је у томе што се код TE методе врши додатно филтрирање на основу категорије турнира, док се код ITF методе само враћају добијени подаци.

* Метода **createTournament():**

**За TE** користи категорије турнира да би одредила важност, такође користи друге податке као што су локација, површина и остале, да би креирала објекат **TennisTournament**. **Важност турнира се одређује** на основу вредности категорије турнира (categoryId). Категорије се претварају у вредности које одређују важност турнира.



**За ITF** такође користи категорије и друге податке као што су локација и површина, али се додатно користе и финансијски подаци као што је новчана награда како би се одредила важност турнира. Категорије турнира се користе за израду вредности **importance**, која означава важност турнира. Користи се комбинација категорије турнира и новчане награде да се одреди важност.



У TournamentUtils фајлу имплементиран је код који служи за учитавање и руковање подацима о тениским турнирима, рад са енумерацијом подлога, управљање фабрикама које креирају објекте турнира, приказ информација и валидацију активних турнира.

### Функционалности

Апликација развијена у React-у омогућава корисницима преглед турнира на интерактивној мапи са различитим опцијама филтрирања (по месецу, категорији, подлози и тренутном опсегу мапе). Корисници могу да одаберу турнир и годину, да претражу по кључним речима, као и да прегледају турнире у бочном менију.Такође на мапи се константно ажурира број приказаних турнира на мапи према филтрирању. Кликом на турнир, мапа се аутоматски помера на локацију турнира, а при преласку преко маркера приказује се tooltip. Такође, могуће је преусмерити корисника на званичну страницу турнира.

Кључне функционалности апликације су:

1. Приказ мапе са турнирима
2. Одабир турнира и године
3. Претрага по кључној речи
4. Филтрирање према месецу
5. Филтрирање према категоријама
6. Филтрирање према подлози
7. Филтрирање према тренутном опсегу мапе
8. Преглед турнира у бочном менију
9. Одабиром турнира скок на локацију на мапи
10. Приказ tooltip-а при прелазу преко маркера
11. Редирекција на званичну страницу турнира
12. Приказ броја турнира
13. Приказ мапе са турнирима

Компонента MapComponent служи за приказ интерактивне мапе са тениским турнирима користећи OpenLayers библиотеку. Мапа се рендерује унутар <div> елемента са ID-јем „map“ а као основни слој користи Mapbox tiles.

Ова функционалност омогућава приказ мапе, зумирање, кретање по мапи, приказ турнира у виду маркера и кластеровање турнира ако су близу једни другима.

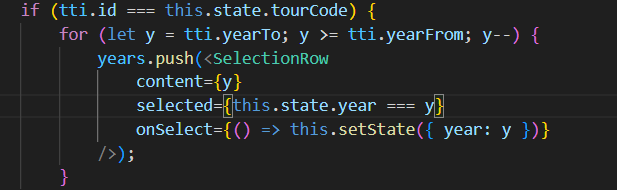
За приказ турнира на мапи је задужена функција **addPointsToMap()** која их додаје и приказује у облику маркера и примењује кластеровање. У поглављу „“ је приказан код (слика број ) и објашњено је више о овоме.

1. Одабир турнира и године

Kомпонента **SourceSelector** је модални прозор који омогућава кориснику да изабере тениски турнир и годину (сезону). Компонента пружа функционалности као што су:

* Приказ листе турнира
* Динамичко ажурирање доступних сезона након избора турнира
* Избор године
* Потврда избора кликом на дугме "Select"
* Затварање модала кликом ван њега

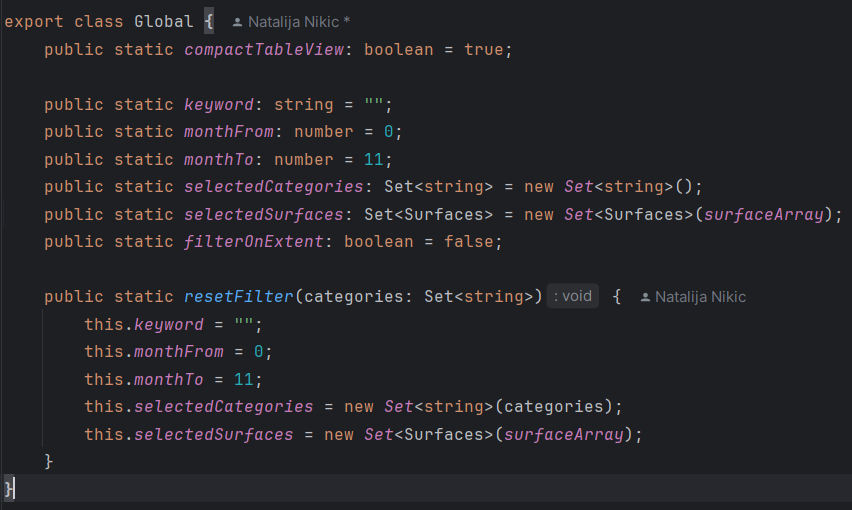
Najvažniji deo koda je dinamičko ažuriranje liste sezona, jer pokazuje kako React ažurira prikaz na osnovu korisničkog izbora. У овом коду се проверава да ли је тренутни турнир (tti.id) онај који је корисник изабрао (this.state.tourCode). Креира се листа сезона у опсегу од yearTo до yearFrom (од последње до прве године). Кликом на годину, ажурира се state.year и означава се изабрана сезона.



Остали делови компоненте SourceSelector раде следеће: Функција onClick затвара модал ако корисник кликне изван њега, док дугме "X" такође затвара модал. Листа турнира се генерише помоћу getTennisTourInfos(), а кликом на турнир ажурира се tourCode и аутоматски бира последња доступна сезона. Дугме "Select" потврђује избор и позива this.props.onSelect(this.state.tourCode, this.state.year).

Филтрирање по различитим критеријумима је имплементирано помоћу компоненте TournamentFilter, TorunamentSideBar и глобалне класе Global која се налази у GlobalUtils фајлу.

Класа **Global** представља важан део апликације јер служи као глобални простор за складиштење и управљање различитим параметрима који се користе за филтрирање и приказ података у различитим компонентама. Класу користе различити делови апликације, као што су компоненте за приказ резултата, филтрирање по месецима, категоријама, подлогама и кључним речима. Ова структура помаже да се одржи доследност података у целој апликацији и осигура да све компоненте користе исти сет вредности.



Компонента **TournamentsFilter** омогућава филтрирање турнира према различитим критеријумима, укључујући кључну реч, датуме, категорије, подлоге и географски опсег мапе.

Компоненти се шаљу следећи props:

* categorySet: Скуп категорија који се користи за филтрирање.
* tournaments: Листа турнира која се филтрира према критеријумима.
* filter: Функција која се позива приликом промене филтера како би се ажурирали приказани турнири.
* onDragStart (опционо): Функција која се позива када почне повлачење компоненте.
* onDrag (опционо): Функција која се позива током повлачења компоненте.

Даље, у **TorunamentSideBar** компоненти треба истаћи функцију **filter** која ради следеће:

* Кључна реч (kwd) се узима из глобалног стања (Global.keyword) и проверава да ли неки од атрибута турнира (као што су locationId, name, categoryId, surfaceCode) садржи кључну реч.
* Проверава се да ли месец почетка и краја турнира спадају у опсег месеци дефинисан у Global.monthFrom и Global.monthTo.
* Проверава се категорија и подлога турнира у односу на оне које су изабране (Global.selectedCategories, Global.selectedSurfaces).
* Ако је укључено филтрирање по опсегу мапе (Global.filterOnExtent), проверава се да ли су координате турнира унутар тренутног опсега мапе.
* Ако су сви услови испуњени, турнир се додаје у низ filteredTournaments.
* После филтрирања, резултати се шаљу родитељској компоненти преко this.props.onFilterApplied(filteredTournaments).
* На крају, стање се ажурира са новим филтрираним турнирима (this.setState({ tournaments: filteredTournaments })), што покреће поновно рендеровање компоненте.

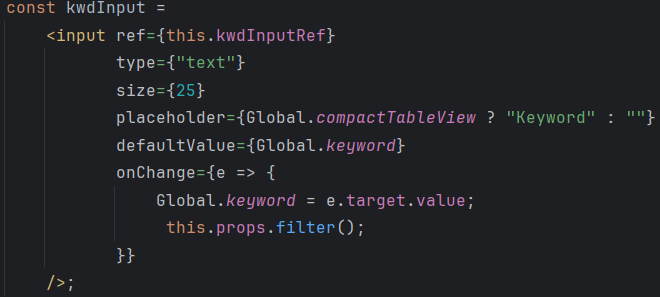


1. Претрага по кључној речи

Функционалност претраге по кључној речи омогућава кориснику да филтрира податке на основу унетог текста. Кључна реч се чува у променљивој **keyword** која се налази у класи Global и користи се за филтрирање података у компонентама где је потребно.

У TournamentFilter компоненти је имплементиран код који омогућава кориснику да уноси кључну реч у текстуално поље.

defaultValue={Global.keyword} поставља иницијалну вредност инпута према глобалној променљиви Global.keyword која се користи за складиштење тренутне кључне речи. Када корисник унесе или промени вредност у инпуту, позива се функција onChange која ажурира вредност кључне речи и позива функцију this.props.filter() која врши филтрирање на основу нове вредности.



Како корисник уноси нову реч, одмах се покреће филтрирање, што значи да је ово „real-time“ филтрирање.

1. Филтрирање према месецу

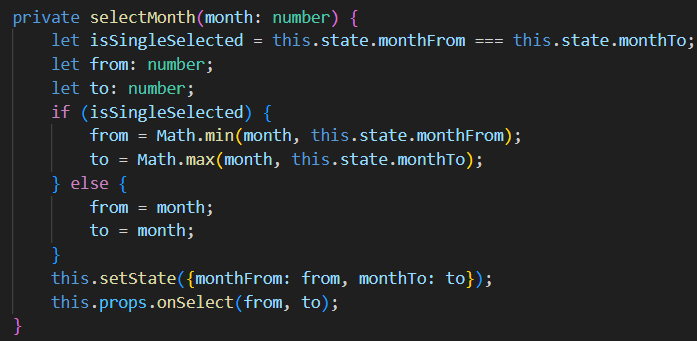
Компонента **MonthRangeChooser** омогућава кориснику да изабере опсег месеци. Ова компонента користи локално стање за чување тренутно изабраног опсега месеци (monthFrom и monthTo), и омогућава кориснику да кроз интеракцију са интерфејсом изабере месец или промену опсега месеци. Кључне функционалности:

* **Приказ месеца**: Компонента приказује све месеце године, при чему свако дугме које представља одређени месец има класу која се динамички мења у зависности од тога да ли се тај месец налази унутар изабраног опсега.
* **Интеракција са корисником**: Корисник може кликнути на месец да би га изабрао. Уколико је већ изабран један месец, кликом на други месец проширује се опсег. Уколико је опсег већ проширен, кликом на месец тај месец се изабира као нови опсег.
* **Ажурирање стања**: Када корисник изабере нови месец, стање компоненте се ажурира са новим опсегом. Након што се стање промени, родитељска компонента се обавештава о ново изабраном опсегу.
* **Реакција на мобилним уређајима**: Компонента има различите стилове за мобилне уређаје. На мобилним уређајима користи се другачија класа за приказ месеца.

Кључни делови кода су везани за рендеровање и селектовање месеца.

У render методи генеришемо низ monthElements који садржи све месеце. Свак месец је дугме (div) са одговарајућим CSS разредом који се мења у зависности да ли је месец унутар изабраног опсега (selected класа). Кроз петљу пролазимо кроз све месеце и додајемо их на екран. На сваки месец може да се кликне и тада се позива метод selectMonth(month) са одговарајућим месецом.

Метода **selectMonth(month)** се позива када корисник кликне на месец. Ако су већ селектовани месеци (monthFrom === monthTo), опсег месеци се проширује (ако је селектован само један месец). Ако нису селектовани месеци, месец се поставља као почетни и завршни месец селектованог опсега. Након ажурирања стања са новим месецима (monthFrom и monthTo), позива се this.props.onSelect са новим опсегом месеци, чиме се обавештава родитељска компонента о корисниковом избору.



1. Филтрирање према категоријама

Филтрирање према категоријама омогућава корисницима да одаберу које категорије турнира желе да. Сет одабраних категорија чува се у selectedCategories у класи Global.

Филтрирање по категоријама функционише тако што омогућава кориснику да изабере једну или више категорија из скупа доступних категорија (categorySet). При свакој промени избора категорија, компонентa ажурира глобалну листу селектованих категорија (Global.selectedCategories) и позива функцију filter како би се поново извршило филтрирање приказаних турнира.

Код имплементиран у TournamentsFilter генерише интерфејс за избор категорија и омогућава филтрирање турнира на основу изабраних вредности.



1. Филтрирање према подлози
2. Филтрирање према тренутном опсегу мапе
3. Преглед турнира у бочном менију
4. Одабиром турнира скок на локацију на мапи
5. Приказ tooltip-а при прелазу преко маркера
6. Редирекција на званичну страницу турнира
7. Приказ броја турнира

# ПРИКАЗ СОФТВЕРСКОГ РЕШЕЊА ЗА МАПУ ТЕНИСКИХ ТУРНИРА

1. Одабир турнира и године

# ТЕСТИРАЊЕ

# ЗАКЉУЧАК

Пројекат је резултат рада на развоју веб апликације која интегрише више технологија и алата како би се пружило корисничко искуство које је истовремено функционално и привлачно. Основна сврха апликације била је омогућавање корисницима да претражују и приступају подацима о тениским турнирима путем аутоматизованог процеса сакупљања информација са интернета, чиме је постигнута значајна уштеда времена и ресурса.

Успешна имплементација коришћењем Puppeteer библиотеке за аутоматизацију прегледача омогућила је ефикасно скупљање података са веб страница. Овај алат је доказао своју вредност у ситуацијама када није могуће користити јавне АПИ-је, чиме је демонстрирана способност да се прилагоди и превазиђе ограничења доступних ресурса. Један од најважнијих аспеката пројекта био је осигуравање тачности и конзистентности података. Примена метода за обраду и чување података у JSON формату олакшала је манипулацију и складиштење информација, што је омогућило њихову даљу анализу и употребу у различитим контекстима.

Посебан акценат је стављен на визуелизацију података кроз интеграцију мапе и података о тениским турнирима користећи React и Open Layers. Коришћењем Mapbox API-ја, апликација пружа преглед турнира на мапи, што корисницима олакшава проналажење информација о турнирима на одређеним локацијама. Апликација укључује функционалности као што су кластеровање тачака на мапи, интерактивно приказивање детаља о турнирима, као и напредна филтрирања која омогућавају корисницима да лако прегледају и пронађу информације од интереса.

Пројекат је такође обухватио тестирање и верификацију функција, осигуравајући да апликација испуњава све постављене захтеве и ради исправно. Корисничко искуство је додатно унапређено кроз прилагођавање интерфејса и интеграцију визуелних елемената који олакшавају навигацију и употребу апликације.

# ЛИТЕРАТУРА

1. Choudhary, J. (2024) Every React Concept Explained in 5 Minutes [https://dev.to/jitendrachoudhary/every-react-concept-explained-in-5-minutes-39b9, датум приступа: 12.12.2024.]
2. Codeacademy.com (2021) What Is a Framework [https://www.codecademy.com/resources/blog/what-is-a-framework/, датум приступа: 12.12.2024.]
3. Fedosejev, A. (2015) React.js Essentials. Packt Publishing.
4. Grzybowska, K. (2023) What is React.js Used For: Top 10 Examples [https://www.monterail.com/blog/what-is-react-used-for, датум приступа: 12.12.2024.]
5. Machalica, A. (2024) Top 10 Companies Using React.js

[https://selleo.com/blog/top-10-companies-using-reactjs, датум приступа: 12.12.2024]

1. Oluwatosin Isijola, T. (2023) Understanding React Hooks [https://www.getfishtank.com/blog/understanding-react-hooks, датум приступа: 12.12.2024.]
2. Singh, S. (2023) Virtual DOM and Real DOM: Understanding the Differences [https://medium.com/@surksha8/virtual-dom-and-real-dom-understanding-the-differencesda8f3fab4261, датум приступа: 12.12.2024.]
3. Pptr.dev (2025) What is Puppeteer? [https://pptr.dev/guides/what-is-puppeteer, датум приступа: 05.01.2025.]
4. Wikipedia (2024) OpenLayers [https://en.wikipedia.org/wiki/OpenLayers, датум приступа: 22.01.2025.]
5. Jaye, H (2022) How HTML, CSS, and JavaScript work [https://v1.scrimba.com/articles/html-css-javascript/, датум приступа: 19.01.2025.]
6. Ketan Varshneya (2023) The History of JavaScript: A Journey from Netscape to Frameworks and Libraries [https://www.techaheadcorp.com/knowledge-center/history-of-javascript/, датум приступа: 19.01.2025.]
7. Geeksforgeeks.org (2025) Introduction to JavaScript [https://www.geeksforgeeks.org/introduction-to-javascript/, датум приступа: 22.01.2025.]
8. W3schools.com (2025) HTML Introduction [https://www.w3schools.com/Html/html\_intro.asp/, датум приступа: 22.01.2025.]
9. Codecademy.com (2018) What does the “cascading” in CSS mean? [https://discuss.codecademy.com/t/what-does-the-cascading-in-css-mean/340936, датум приступа: 22.01.2025.]
10. Imran Hossain (2022) Why use React Js [https://www.linkedin.com/pulse/why-use-react-js-imran-hossain/, датум приступа: 22.01.2025.]
11. Antonio Santiago Perez (2012) OpenLayers Cookbook, Packt Publishing Ltd