Izrada web aplikacije Rent-a-Car pomoću alata ReactJS i WordPress CMS

Martinez, Tomislav

Sveučilište
Sjever

Završni rad br. 878/MM/2024

Izrada web aplikacije Rent-a-Car pomoću alata ReactJS i WordPress CMS

Tomislav Martinez, 6051/957

Varaždin, srpanj 2024. godine

Sveučilište
Sjever

Odjel za multimediju

Odsjek za multimediju, oblikovanje i primjenu

Završni rad br. 878/MM/2024

Izrada web aplikacije Rent-a-Car pomoću alata ReactJS i WordPress CMS

Student

Tomislav Martinez, 6051/957

Mentor

doc.dr.sc. Marko Čačić

Varaždin, srpanj 2024. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za multimediju

STUDIJ: Prijediplomski stručni studij Multimedija, oblikovanje i primjena

PRISTUPNIK: Tomislav Martinez | MATIČNI BROJ: 0336051957

DATUM: 06.03.2024. | KOLEGIJ: Web dizajn

NASLOV RADA: Izrada web aplikacije Rent-a-Car pomoću alata ReactJS i WordPress CMS

NASLOV RADA NA ENGL. JEZIKU: Development of the Rent-a-Car web application using ReactJS and WordPress CMS

MENTOR: dr. sc. Marko Čačić | ZVANJE: Docent

ČLANOVI POVJERENSTVA:

1. dr. sc. Snježana Ivančić Valenko, v. pred. - predsjednica
2. doc. dr. sc. Marko Čačić - mentor
3. Anja Zorko, pred. - članica
4. Jelena Vlašić, pred. - zamjenska članica
5.

Zadatak završnog rada

BROJ: 878/MM/2024

OPIS:

U cilju ostvarenja kvalitetnog korisničkog iskustva, front-end razvoj u modernom web dizajnu oslanja se na programski jezik JavaScript i pripadajuće framework alate, pri čemu se posebno ističe ReactJS. U kontekstu jednostavne i brze realizacije back-end funkcionalnosti zanimljivo rješenje je primjena platforme WordPress CMS i pripadajućeg API sučelja. U ovom radu detaljno će se prikazati postupak realizacije web aplikacije Rent-a-Car koja će koristiti navedene alate i tehnologije. Funkcionalnosti front-end sučelja realizirat će se primjenom alata ReactJS, a back-end funkcionalnosti će se temeljiti na platformi WordPress CMS. Komunikacija između back-end i front-end komponenti web aplikacije odvijat će se primjenom API sučelja WordPress CMS-a. Navedeni pristup predstavlja značajan iskorak u odnosu na klasičnu monolitnu arhitekturu web aplikacije, što se ogleda u modularnosti, skalabilnosti i performansama.
U radu je potrebno:
- Definirati pojam web aplikacije te objasniti što u tom kontekstu znači front-end, a što back-end
- Definirati značajke predmetne web aplikacije te izraditi plan i metodologiju razvoja
- Ukratko objasniti alate ReactJS i WordPress CMS u razmatranom kontekstu
- Detaljno prikazati postupak praktične realizacije razmatrane web aplikacije
- Izvesti zaključak rada

ZADATAK URUČEN: 26.06.2026. | POTPIS MENTORA:

REPUBLIKA HRVATSKA
SVEUČILIŠTE SJEVER

Predgovor

Ovaj završni rad bavi se izradom web aplikacije primjenom WordPress CMS platforme i pripadajućeg API sučelja u kombinaciji s alatom ReactJS te pruža uvid u navedene tehnologije i njihove funkcionalnosti.

Želim se zahvaliti obitelji i prijateljima koji su me podržavali u svakom pogledu ove prethodne tri godine studiranja. Zahvaljujem svim profesorima na prenesenom znanju i motivaciji koja mi je bila potrebna, no ponajviše mentoru doc. dr. sc. Marku Čačiću, na susretljivosti, podršci i pomoći pri izradi ovog završnog rada.

Sažetak

Ovaj rad proučava metodologiju povezivanja tehnologija ReactJS i WordPress CMS, na način da se razvoj aplikacije podijeli na *front-end* i *back-end*. Na primjeru aplikacije „Rent-a-Car“ fokusira se na prednosti svake od tehnologija i proučava na koji način funkcioniraju zajedno.

Prvi dio rada sastoji se od teorijskog dijela gdje je objašnjeno što su web aplikacije te se opisuju temeljni alati koji će se razmatrati - ReactJS i WordPress CMS. Zatim je detaljno objašnjen plan razvoja web aplikacije, metodologija te pomoćni alati i ostali potrebni resursi. Drugi dio rada prikazuje praktični dio gdje je objašnjen kompletan proces razvoja web aplikacije, od stvaranja baze podataka do gotove web aplikacije.

Ključne riječi: ReactJS, WordPress CMS, REST API, Headless WordPress, web aplikacija, HTML, JavaScript

Abstract

This paper examines the methodology of integrating ReactJS and WordPress CMS technologies in a way that separates the development of an application, developing the front-end and the back-end separately. Using the example of a „Rent-a-Car“ application, it focuses on the advantages of each technology and explores how they function together.

The first part of the paper consists of a theoretical section explaining what web applications are and describing the essential tools that will be used - ReactJS and WordPress CMS. Then, it provides a detailed explanation of the web application development plan, methodology, auxiliary tools, and other necessary resources. The second part of the paper contains the practical section, explaining the complete process of developing a web application, from creating the database to presenting the finished web application.

Keywords: ReactJS, WordPress CMS, REST API, Headless WordPress, web aplikacija, HTML, JavaScript

Popis korištenih kratica

HTML HyperText Markup Language
Opisni jezik za opisivanje strukture i sadržaja HTML dokumenata

CSS Cascading Style Sheets
Opisni jezik za opisivanje vizualne prezentacije (grafičkog dizajna) HTML dokumenata

JS JavaScript
Programski jezik za izradu interaktivnih web stranica i aplikacija

API Application Programming Interface
Sučelje putem kojeg različite aplikacije mogu komunicirati i razmjenjivati podatke

JSON JavaScript Object Notation
Format za razmjenu podataka između različitih aplikacija

NPM Node Package Manager
Upravitelj paketa za Node.js

PHP Hypertext Preprocessor
Programski jezik za izradu dinamičnih web stranica i aplikacija

JSX JavaScript Syntax Exstension
Ekstenzija JavaScript jezika koja olakšava integraciju HTML koda unutar JavaScript programa

REST Representational State Transfer
Skup pravila koja određuju kako bi se trebala ponašati arhitektura podataka koji se razmjenjuju putem API sučelja

DB Database
Baza podataka

JWT JSON Web Token
Predloženi web standard za stvaranje podataka s tokenom kao sredstvom za autentifikaciju

Sadržaj

1. Uvod

U današnje vrijeme rijetko tko može reći da ne koristi internet ili neku od brojnih pogodnosti koje nam moderan internet svakodnevno nudi. Od plaćanja računa, kupovine i iznajmljivanja smještaja za odmor pa do društvenih mreža, video poziva i navigacije, web aplikacije su sveprisutne te kao takve konstantno napreduju i proširuju se. U svijetu gdje se 3 milijarde ljudi svaki mjesec prijavljuje na Facebook, a popratno i mnoštvo ostalih web aplikacija, rast i napredak svih područja vezanih uz web razvoj neće usporiti. [1]

Uz nagli i neprestani rast broja korisnika interneta te broja web aplikacija koje ti korisnici svakodnevno koriste, programeri su u neprestanoj potrazi za boljim, bržim i jednostavnijim rješenjima. Bilo to dodavanje novih funkcionalnosti, brže učitavanje sadržaja ili efikasnije pisanje koda, današnje web aplikacije su s vremenom postale nezamislivo napredne i sposobne. Stoga, unatoč tome što kompleksnost web aplikacija raste, kompleksnost samog razvijanja web aplikacija pada. Danas postoje razni *front-end* razvojni okviri i biblioteke kao što su React, Angular, Vue.js itd. te *back-end* razvojni okviri kao Express.js, Django, *Ruby on Rails* i mnogi drugi. Uz to, postavljanje servera i baza podataka nikada nije bilo jednostavnije uz rješenja kao što su Apache i Nginx za servere te MySQL i MongoDB za bazu podataka.

U ovom radu se namjerava opisati i detaljno prikazati postupak izrade web aplikacije koristeći ReactJS za realizaciju *front-end*-a povezan s WordPress CMS-om koji će služiti kao jednostavan *back-end*. Uz uvodni dio gdje će svi korišteni alati biti opisani i objašnjeni, cilj ovog rada je približiti način na koji sve web aplikacije koje se svakodnevno koriste funkcioniraju, objasniti spomenute tehnologije te sam proces razmišljanja kod razvoja web aplikacija.

2. Web aplikacija

Uz nagli razvoj tehnologije i interneta sredinom 2000-ih, internet je ubrzano postajao sve interaktivniji za krajnjeg korisnika. Stoga, uz pojam „Web 2.0“, ubrzo se pojavio i pojam „Web aplikacija“. Web stranice se više nisu nužno sadržavale od samo tekstualnog sadržaja, već su korisnici imali mogućnost interaktivnosti sa stranicama učitanim u njihovom web pregledniku, stoga su pojedine web stranice ustvari postale web aplikacije.

Aplikacija je izvršni računalni program koji je osmišljen i izrađen da provede neki specifični zadatak, no za razliku od tradicionalnih aplikacija, web aplikacije se pokreću u samom web pregledniku.

Web aplikacije su zapravo samo web stranice koje izgledaju te funkcioniraju kao aplikacije. Uz pomoć tehnologija HTML, CSS i JavaScript, web aplikacije poprimaju strukturu i željeni izgled, a pomoću tehnologija JavaScript, Java, PHP i slično poprimaju i željenu funkcionalnost.

2.1. Prednosti i nedostaci web aplikacija

Glavna prednost web aplikacija nad običnim aplikacijama jest ta da web aplikacije rade jednako, bez obzira na kojem se uređaju koriste. Održavanje jedne web aplikacije je uvelike lakše nego održavanje, na primjer, aplikacije izrađene za Windows, iOS i Android operativne sustave zasebno. [2]

S druge strane, glavni nedostaci web aplikacija također proizlaze iz činjenice da se pokreće u web pregledniku. Web aplikacije često imaju lošije korisničko iskustvo (eng. *UX*) te nemaju toliku mogućnost povezivanja s *hardware*-om uređaja, stoga imaju i manje mogućih funkcionalnosti.

2.2. Front-end

Svaka aplikacija sadrži korisničko sučelje, dio web aplikacije koji služi kako bi korisnik mogao upravljati samom web aplikacijom. Korisnička sučelja mogu sadržavati polja za upisivanje teksta, gumbe, odabir datuma i slično.

Front-end je naziv za „prednji“ dio aplikacije, odnosno samo korisničko sučelje. To je ustvari dizajn web aplikacije, dio koji korisnik vidi i pomoću kojeg upravlja sa svim procesima koji se događaju u pozadini iza samog korisničkog sučelja.

Front-end mora biti optimiziran i jednostavan za korištenje. Glavna zadaća *front-end* programera je osigurati da se korisnici web aplikacije s lakoćom mogu snaći te komunicirati s web

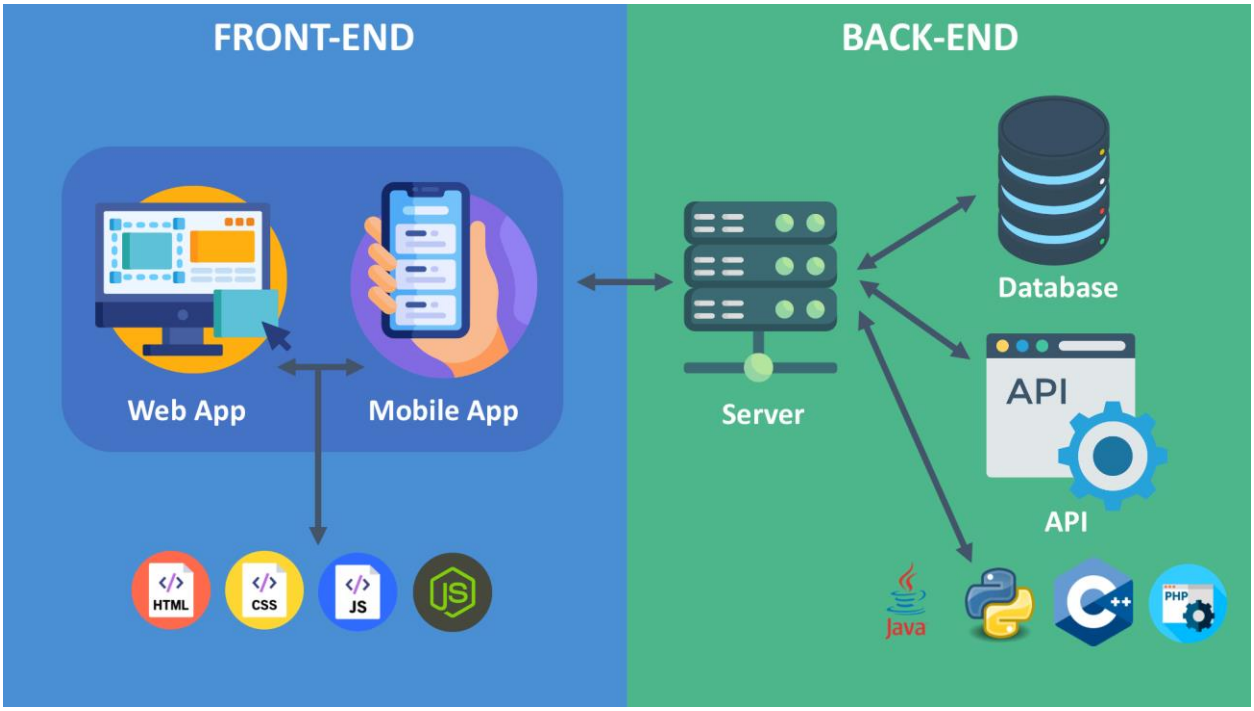
stranicom. U web aplikacijama to se postiže pomoću tehnologija HTML, koji služi za stvaranje strukture web aplikacije, CSS, koji služi za oblikovanje pojedinih HTML elemenata te JavaScript, koji služi za *front-end* funkcionalnost web aplikacije kao na primjer detektiranje kada je korisnik pritisnuo gumb. [3]

Ono što nije vidljivo iz perspektive korisnika, a čime svjesno ili nesvjesno upravlja pomoću korisničkog sučelja, naziva se *back-end*.

2.3. Back-end

Back-end je dio web aplikacije koji nije izravno vidljiv korisnicima, no koji je neophodan kako bi web aplikacija funkcionirala. To podrazumijeva baze podataka i programski kod koji se nalaze na serveru te kojima korisnik ne može pristupiti, no koji upravljaju cijelom funkcionalnošću web aplikacije. [3]

Bez *back-end*-a, web aplikacije bile bi samo prikaz dizajna na ekranu, s gumbima i poljima koji ne rade ništa. Uz pomoć *back-end*-a, web aplikacija dobiva mogućnost komuniciranja s bazom podataka gdje se podaci uneseni u *front-end*-u od strane korisnika obrađuju te ponovo šalju u *front-end*, gdje se prikazuju.



Slika 2.1 Grafički prikaz razlika front-end-a i back-end-a,
izvor: Adityagaba, https://medium.com/@adityagaba1322/streamlining-backend-frontend-integration-a-quick-guide-145eca3cca05

3. ReactJS biblioteka

JavaScript biblioteke (eng. *JavaScript libraries*), kao što je ReactJS, olakšavaju rad programera jer pružaju mnoštvo gotovih, već ugrađenih funkcija. Ovi alati uvelike ubrzavaju proces izrade *front-end*-a web aplikacija te pomažu pri strukturiranju samog koda, a posljedično je i sam kod čitljiviji i lakši za razumijevanje u slučajevima kada na projektu radi više programera.

Korištenje JavaScript biblioteka zahtjeva osnovno poznavanje JavaScript programskog jezika, no kod korištenja biblioteka programerima je većina željenih funkcija koje zahtijevaju više znanje JavaScript-a dostupne iza nekoliko linija koda, što je jedan od glavnih razloga zašto su postale toliko popularne.

3.1. Što je ReactJS

ReactJS (kratko samo React) je besplatna *front-end* JavaScript biblioteka otvorenog koda koja omogućuje jednostavniju izradu kompleksnih dinamičnih korisničkih sučelja.

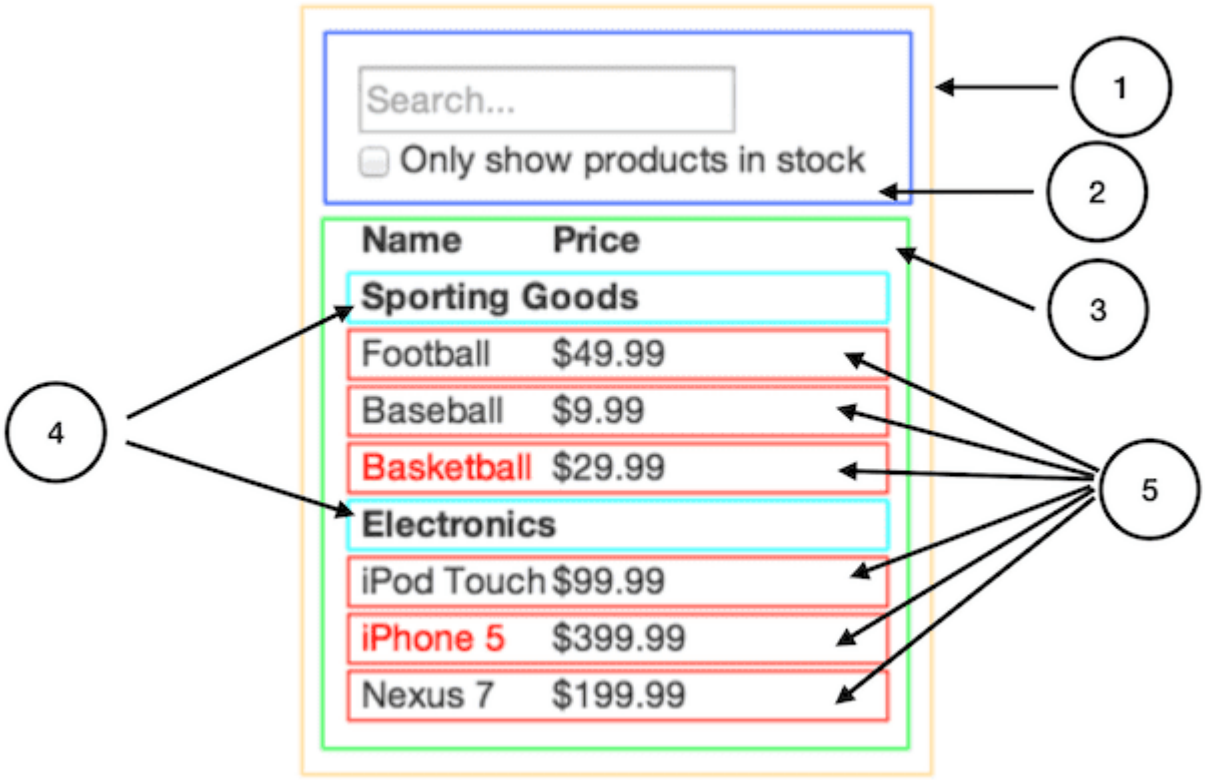
ReactJS biblioteku je izradila tvrtka Facebook u ožujku 2013. godine. Unatoč tome, i dan danas je održavana od strane tvrtke Meta (nekadašnji Facebook), kao i mnogobrojne zajednice koja konstantno unaprjeđuje samu biblioteku razvijajući vlastite dodatne funkcionalnosti koje programeri mogu ukomponirati u svoje projekte. [4]

3.2. Komponente

S vremenom, kako je internet postajao sve interaktivniji te se sadržaj na stranici konstantno trebao ažurirati ovisno o nekoj promjeni na samoj stranici, pisanje HTML-a, CSS-a i JavaScript-a zasebno postajalo je sve nepoželjnije. Iz tog razloga je React napravljen tako da su HTML i JavaScript napisani na istom mjestu, te JavaScript vraća HTML sadržaj putem funkcija koje mogu mijenjati isti sadržaj na bezbroj načina putem sintakse koja se naziva JSX. [5]

JSX ili JavaScript XML omogućuje da, na primjer, i HTML i JavaScript za funkcionalnost gumba ostanu integrirani u jedinstveni kod koji je razumljiv, jednostavan za ažuriranje i nalazi se u istoj datoteci, što uvelike olakšava sam razvoj web aplikacije. Na ovaj način se zapravo stvaraju React komponente. Svaka React komponenta je JavaScript funkcija koja može sadržavati i *markup* koji se zatim prikaže u web pregledniku. Korištenjem više ovakvih komponenti stvaraju se cijele stranice.

React komponente mogu sadržavati druge komponente, a takve se komponente nazivaju „*Parent*“. Komponente koje *Parent* komponente sadrže se nazivaju „*Child*“ komponentama. *Parent* komponenta je odgovorna za prikaz *child* komponente i za prosljeđivanje podataka na *child* komponentu. Ovakvom hijerarhijskom podjelom komponenata lako se kreiraju komponente koje rade zajedno kako bi prikazale određeni sadržaj korisniku. [6]



Slika 3.1 Grafički prikaz podjele web aplikacije na komponenete,
izvor: React, https://legacy.reactjs.org/docs/thinking-in-react.html

Na slici 3.1 je prikazana React web aplikacija za pretraživanje proizvoda upisanih u *array* listu. Bitno je primijetiti da je aplikacija podijeljena na komponente na način da svaka komponenta ima jedan zadatak. Primjerice, komponenta broj 2 je zaslužna za sortiranje, dok je komponenta broj 5 zaslužna za prikaz pojedinačnog proizvoda iz liste. Korištenjem ReactJS „*Hooks*-a“ i komuniciranja putem prosljeđivanja podataka s *parent* komponente na *child* komponentu, komponente za prikaz sadržaja vide što je korisnik upisao u tekstualno polje.

3.2.1. Prosljeđivanje *Props*-a

Parent komponenta može proslijediti takozvane „*Props*“ na *child* komponente koje sadrži te im tako proslijediti informacije. Te informacije mogu biti varijable, objekti, liste i funkcije. *Props* se prosljeđuju na način da se unutar JSX *markup*-a komponenti pridoda određeni *tag* kojem je vrijednost neka od navedenih opcija.

3.2.2. useState

Funkcija useState() je dio React biblioteke koji nam omogućuje stvaranje varijable i funkcije kojom se postavlja vrijednost te varijable. *State* se razlikuje od tipične varijable po tome što se po svakoj promjeni varijable komponenta ponovo prikaže, a s njom i novi, promijenjeni sadržaj. [7]

3.2.3. useEffect

Funkcija useEffect() je dio React biblioteke koja nam omogućuje sinkroniziranje komponente s vanjskim sustavom. Pomoću useEffect *hook*-a je moguće obnoviti komponentu kada se obnovi sama stranica, putem fiksnog intervala ili putem praćenja kada se promijeni zadana varijabla. [7]

Unutar useEffect funkcije poziva se funkcija za dohvaćanje podataka iz API-a.

3.2.4. React Router

Većina web aplikacija sadrži nekoliko stranica, kao što su na primjer početna stranica, o nama i slično. *React Router* je paket koji omogućuje implementaciju dinamične izmjene između više stranica u React aplikaciji. [8]

Koristi se za razvoj jedno-straničnih web aplikacija, odnosno aplikacija koje imaju više stranica i komponenti no nije ih potrebno zasebno učitavati. *React Router* samo prikaže određene komponente ovisno o otvorenoj stranici, te na kraj URL putanje pridoda na primjer „/o-nama“ nastavak.

Koristeći *React Router*, zaglavlje stranice može biti potpuno statično te se ne otvara nova stranica klikom na *hyperlink*, već se obnavlja samo sadržaj ispod.

4. WordPress CMS

CMS (engl. *Content Managment System*) je sustav za upravljanje sadržajem. Pomoću CMS-a programeri mogu stvarati, upravljati i objavljivati sadržaj koji se zatim prikazuje u web aplikaciji.

Osim samog upravljanja sadržajem, WordPress CMS ima i mogućnost upravljanja strukturom i izgledom web stranice. Ovakva tehnologija je izvrsna u slučajevima kada pojedinci samostalno vode svoja web sjedišta, pošto omogućuje jednostavan način upravljanja čitavom web stranicom s jednog mjesta.

CMS se sastoji od dva dijela, CMA (engl. *Content Managment Application*) koji omogućuje korisniku da upravlja sadržajem i CDA (engl. *Content Delivery Application*) koji je zapravo *back-end* CMS-a te pohranjuje sadržaj upisan u CMA i prikazuje ga krajnjem korisniku kroz web aplikaciju. Osnova headless CMS razvoja se zasniva na CMA dijelu.[9]

4.1. Headless WordPress

Headless WordPress je WordPress CMS (skraćeno WP CMS) sustav koji koristi odvojenu arhitekturu samog WordPress-a kako bi služio kao *back-end* web aplikacije. [10] U njemu se upravlja sadržajem koji se zatim prikazuje krajnjem korisniku. Ne koriste se dostupni alati za dizajn i strukturiranje web stranice koji se nalaze u WP CMS-u. *Front-end* web aplikacije razvija se zasebno te se preko API poziva preuzima sadržaj s WP CMS-a koji se zatim prikazuje.

4.2. REST API

API (engl. *Application Programming Interface*) je sučelje putem kojeg različite aplikacije mogu komunicirati i razmjenjivati podatke. Obično se API arhitektura najlakše objasni putem pojmova klijent i server. Aplikacija koja sadrži podatke se naziva server, dok se aplikacija koja šalje zahtjev zove klijent. [11] Na primjer, klijent aplikacija za vremensku prognozu šalje zahtjev server aplikaciji koja sadrži potrebne podatke, a zatim se, nakon što klijent aplikacija primi podatke sa servera, podaci o vremenskoj prognozi prikažu korisniku.

REST API radi na principu *GET*, *PUT*, *DELETE* i sličnih funkcija koje klijent aplikacije mogu koristiti kako bi pristupile podacima na serveru putem HTTP-a. Korištenjem REST API alata u sklopu WordPress CMS platforme moguće je pristupiti podacima koji se nalaze na WP stranici, uključujući i objave. [12] Ovi podaci se nalaze u JSON formatu kojem se zatim može pristupiti u kodu klijent aplikacije.

5. Plan i metodologija razvoja aplikacije „Rent-a-Car“

Web aplikacija „Rent-a-Car“ će biti aplikacija kod koje je razvoj podijeljen na *front-end* i *back-end* dijelove. ReactJS biblioteka koristiti će se za razvoj *front-end*-a, a headless WordPress kao jednostavan *back-end* u kojem se sadržaj uređuje.

5.1. Plan razvoja

Tema projekta je jednostavna „Rent-a-Car“ aplikacija sa odvojenim *back-end*-om. Svrha je prikazivanje informacija o automobilima i koji su modeli trenutačno dostupni za iznajmljivanje. Cilj je poboljšati krajnjim korisnicima iskustvo iznajmljivanja klasičnih automobila na način da su im najbitniji podaci i informacije dostupne i pregledne. Ciljana publika su ljubitelji klasičnih automobila i nedavno zaručeni parovi koji žele iznajmiti klasičan automobil.

Web aplikacija će se sadržavati početnu stranicu na kojoj će biti poruka i *call-to-action* te stranicu za pretraživanje automobila gdje će se isti moći filtrirati.

5.2. Metodologija razvoja

Razvoj web aplikacije provoditi će se u nekoliko faza:

Prva faza razvoja će biti jednostavan dizajn web aplikacije u alatu Figma. Kroz proces dizajna utvrdit će se koje komponente su potrebne te će olakšati fazu planiranja hijerarhijske podijele komponenti u React-u.

Druga faza će biti postavljanje *back-end*-a, koji će se sadržavati od lokalne baze podataka i WordPress CMS-a. U WordPress CMS sustav će se unijeti sadržaj pod „Objave“ koje će sadržavati naziv, sliku i kratki opis svakog automobila.

Treća faza će biti planiranje i kodiranje *front-end*-a aplikacije.

5.3. Potrebni alati

U projektu će se koristiti razni alati i njihove ekstenzije, odnosno „*packages*“. Za razvoj *back-end*-a koristiti će se XAMPP putem kojeg će se pokrenuti Apache server i phpMyAdmin sa MySQL bazom podataka te WordPress CMS za upravljanje sadržajem.

Za dizajn korisničkog sučelja koristiti će se alati Figma za dizajn te Adobe Photoshop za uređivanje fotografija. Kao resursi koristiti će se web stranice Pexels.com, Unsplash.com i Freepik.com za fotografije i Google Fonts za potrebne fontove.

Za kodiranje će se koristiti Visual Studio Code putem kojeg će se pokrenuti ReactJS razvojno okruženje te putem npm-a instalirati potrebni dodaci: *React Router*, *Axios* i *html-react-parser*.

5.3.1. Visual Studio Code

Visual Studio Code ili skraćeno VS Code je jedan od, ako ne i najpopularniji uređivač koda, razvijen od strane tvrtke Microsoft. Besplatan je, otvorenog koda i kompatibilan sa Windows, Linus i MacOS operativnim sustavima, što programerima pruža poznato sučelje bez obzira na kojem računalu rade na projektima. VS Code sadrži mnoštvo ekstenzija koje mogu uvelike olakšati sam proces razvoja, a u slučaju razvoja React aplikacije uvelike pomaže ekstenzija „*Better ReactJS Code Snippet*“.

5.3.2. XAMPP

XAMPP je besplatan paket programa otvorenog koda razvijen od strane tvrtke Apache Friends razvijen za više platformi. Koristi se za postavljanje lokalnog testnog servera, a pošto koristi iste tehnologije kao i većina web servera, prijenos web aplikacije iz testnog u pravo okruženje je vrlo jednostavno.

Sadrži Apache HTTP server koji programerima omogućuje izvršavanje PHP koda koji se izvršava na serveru te MySQL bazu podataka.

5.3.3. Node.js

Node.js je tehnologija koja omogućuje kreiranje lokalnog servera te da se JavaScript kod izvršava van samog web preglednika. Time omogućuje asinkrono programiranje JavaScript-a. JavaScript za svoje izvršavanje zahtjeva takozvani "*JavaScript Runtime Environment*", a Node.js predstavlja drugo mjesto za izvršavanje JavaScript koda nepovezano sa samim web preglednikom, odnosno Node.js je samostani "*JavaScript Runtime Environment*". React je ovisan o Node.js tehnologiji, stoga ga je potrebno instalirati na računalo, a s njime se istovremeno instalira i npm.

5.3.4. Npm

Npm registar ili "*node package manager*" je besplatan i ujedno najveći registar koji sadrži preko 800.000 raznih programskih paketa za Node.js. Programeri *software*-a otvorenog koda koriste npm za dijeljenje svog koda sa zajednicom.

On omogućuje jednostavno instaliranje raznih paketa u bilo koji Node.js projekt putem naredbi u naredbenom retku u programu VSCode.

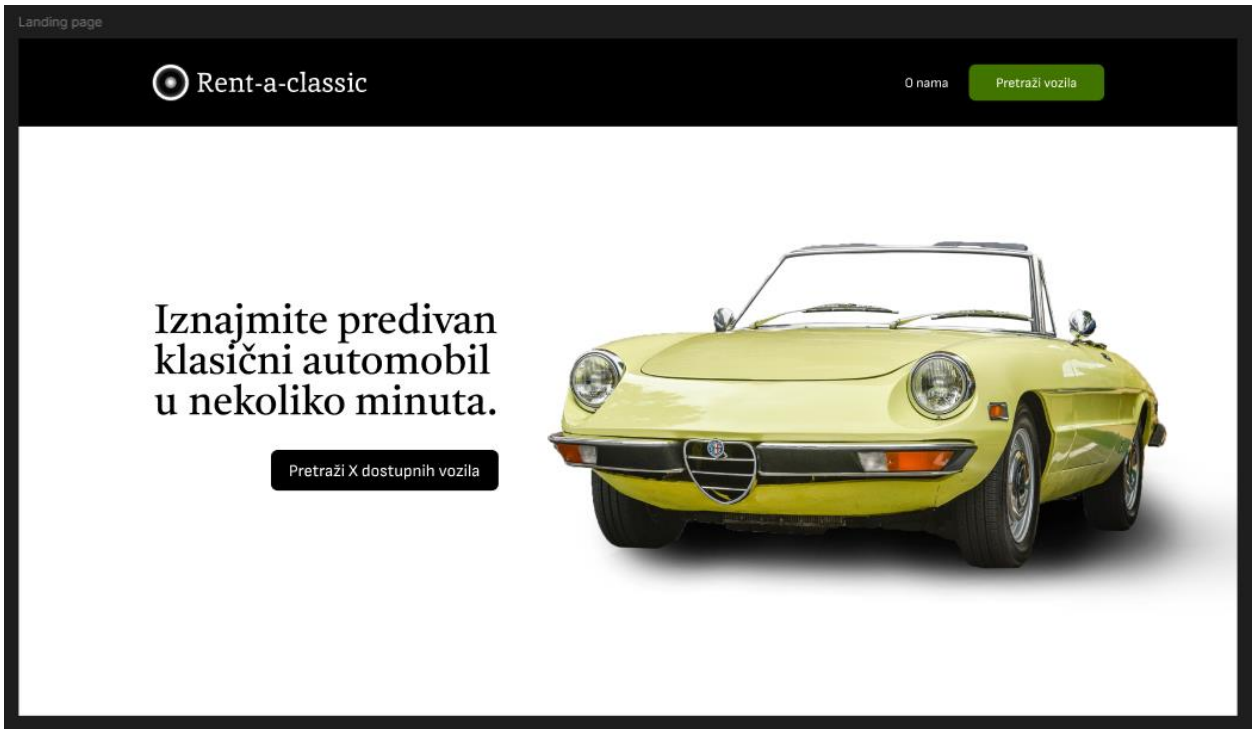
5.3.5. Axios

Axios je popularna biblioteka za komuniciranje sa *back-end*-om. Glavna funkcija joj je pojednostaviti cijeli proces dobavljanja (engl. *fetch*) sadržaja sa API-a.

6. Praktični dio

6.1. Dizajn aplikacije

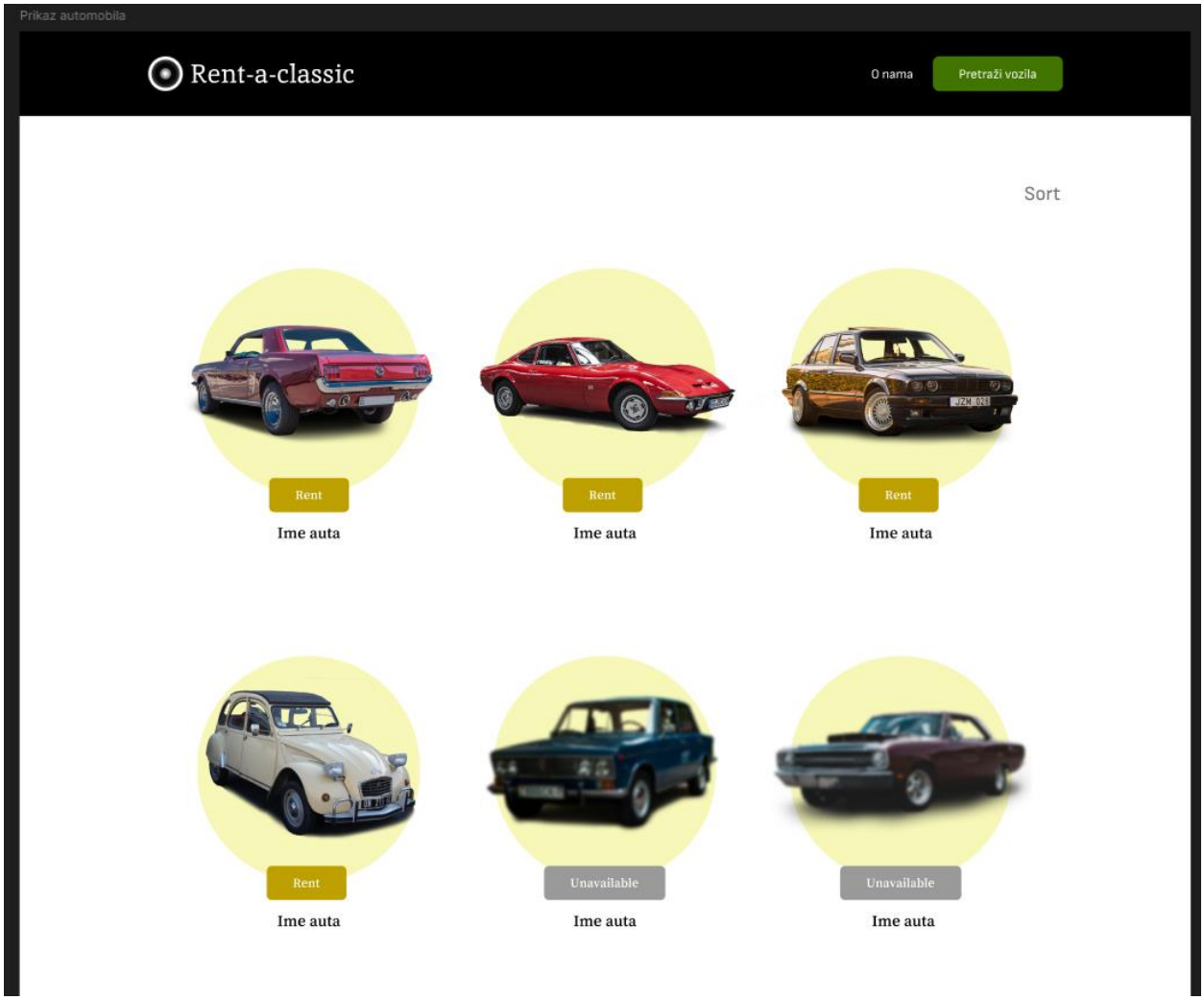
Kako bi se dobio dojam web aplikacije prije samog kodiranja, potrebno je izraditi jednostavan dizajn na temelju kojeg će se kasnije stvarati komponente te CSS stilovi. Za izradu dizajna aplikacije korišten je besplatni alat Figma.



Slika 6.1.1: Dizajn početne stranice, izvor: snimka ekrana, autorov rad

Izvor slike: EdArango100, https://pixabay.com/photos/alfa-romeo-classic-vintage-retro-7968027/

Početna stranica prikazana slikom 6.1.1 će sadržavati jednostavan naslov, sliku te gumb na kojem će pisati broj trenutno dostupnih automobila gdje će se mijenjati tekst ovisno o broju dostupnih automobila („1 dostupno“, „2 dostupna“, „10 dostupnih“...).



Slika 6.1.2: Dizajn stranice za prikaz vozila, izvor: snimka ekrana, autorov rad

Izvori slika: [1,2,3,4,5,6]

Stranica za prikazivanje vozila prikazana slikom 6.1.2 će jednostavno prikazati koji su automobili dostupni, a koji nisu, na određeni datum kojeg će korisnik odabrati. Klikom na određeni dostupni automobil otvorit će se *overlay* u kojem će pisati dodatne informacije o tom automobilu.

[1] Bergadder, https://pixabay.com/photos/car-mustang-ford-ford-mustang-139529/

[2] Rohwedder, https://pixabay.com/photos/opel-gt-antique-car-youngtimer-5190050/

[3] Audrius Strikaitis, https://www.pexels.com/photo/black-car-parked-on-the-road-8274003/

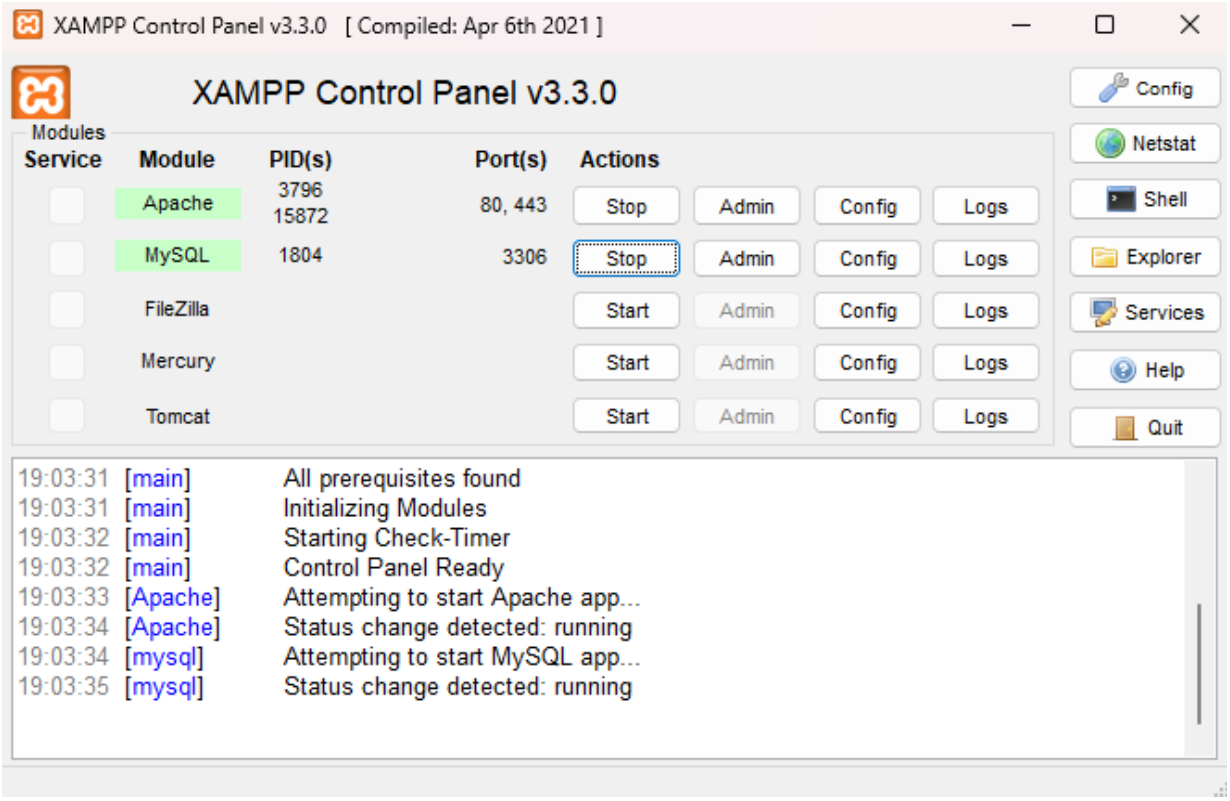
[4] Bruno Charlier, https://www.pexels.com/photo/vintage-car-parked-near-a-restaurant-9830244/

[5] Evgeni Adutskevich, https://www.pexels.com/photo/photo-of-a-blue-classic-car-14807978/

[6] Erik Mclean, https://www.pexels.com/photo/photo-of-classic-car-on-street-4517066/

6.2. WordPress *Back-end*

Za postavljanje WordPress CMS-a kao *back-end*-a kod lokalnog razvoja web aplikacije, potrebno je prvo kreirati lokalni server na računalu. Stoga, prvi korak je instaliranje XAMPP aplikacije te pokretanje Apache servera i MySQL baze podataka.



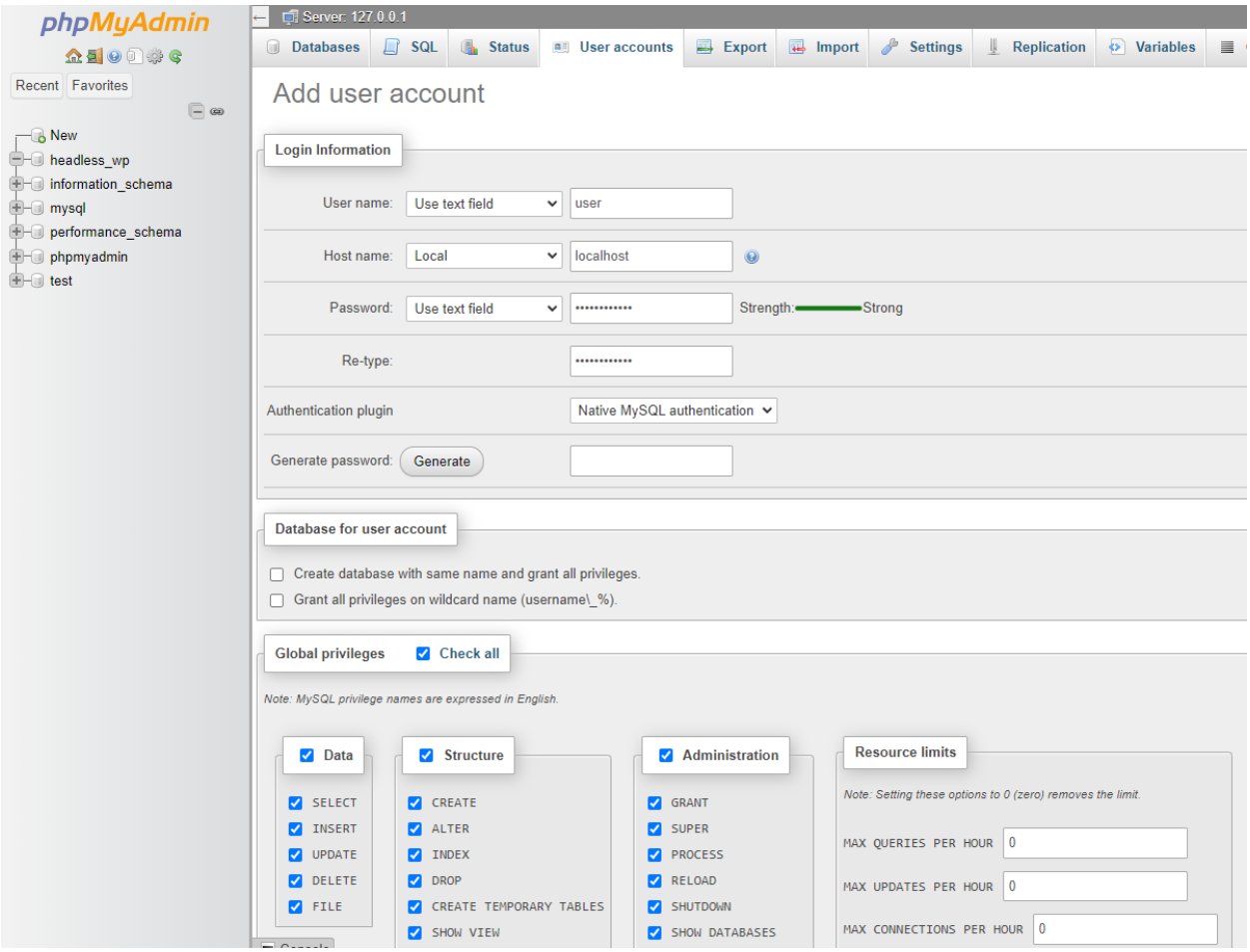
Slika 6.2.1: Prikaz sučelja aplikacije XAMPP te potrebnih postavki, izvor: snimka ekrana, autorov rad

Uključivanjem opcija sa slike 6.2.1 se na adresi „http://localhost/“ pokreće lokalni server kojemu se zatim može pristupiti putem web preglednika. Sljedeći korak je kreiranje baze podataka. Bazama podataka pristupa se putem poveznice „http://localhost/phpmyadmin“ u web pregledniku, čime se otvara phpMyAdmin sučelje.

S lijeve strane sučelja phpMyAdmin se nalaze baze podataka, te se klikom na „*New*“ otvara sučelje za kreiranje nove baze podataka. Ovdje je potrebno kreirati bazu podataka koju će WordPress CMS koristiti za pohranu sadržaja putem sučelja prikazanog na slici 6.2.2.

Slika 6.2.2: Kreiranje baze podataka u myPhpAdmin sučelju, izvor: snimka ekrana, autorov rad

Zatim je potrebno kreirati korisnika koji ima pristup zadanoj bazi i preko kojeg će WordPress imati mogućnost upravljanja podacima. Klikom na phpMyAdmin logo u gornjem lijevom kutu otvaraju se postavke sustava, a nakon odabira opcije „*User accounts*“ u gornjem izborniku se klikom na „*Add user account*“ kreira novi korisnik.

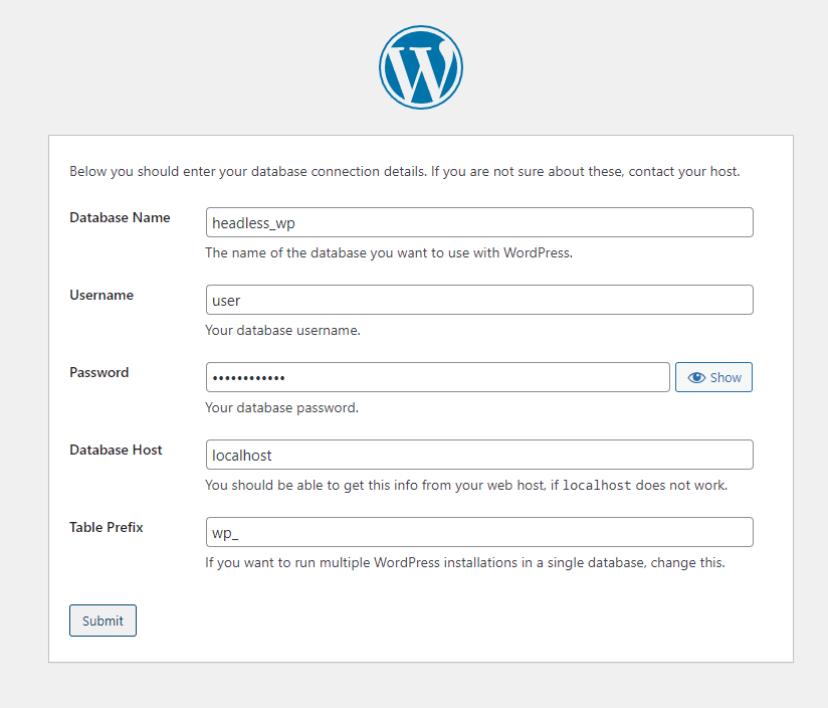


Slika 6.2.3: Kreiranje korisnika u myPhpAdmin sučelju, izvor: snimka ekrana, autorov rad

Kreiranom korisniku potrebno je dati sve privilegije nad kreiranom bazom podataka kao što prikazuje slika 6.2.3, kako bi WordPress CMS mogao ispravno funkcionirati. Zaporku je potrebno zapamtiti iz razloga što će biti potrebna kasnije kod postavljanja WordPress CMS-a.

Sljedeći korak je preuzimanje te instaliranje WordPress CMS-a. Nakon preuzimanja mapu „wordpress“ potrebno je premjestiti u mapu instalacije XAMPP-a, točnije „xampp/htdocs/“, čime WordPress CMS postaje dostupan za otvaranje u web pregledniku putem Apache servera.

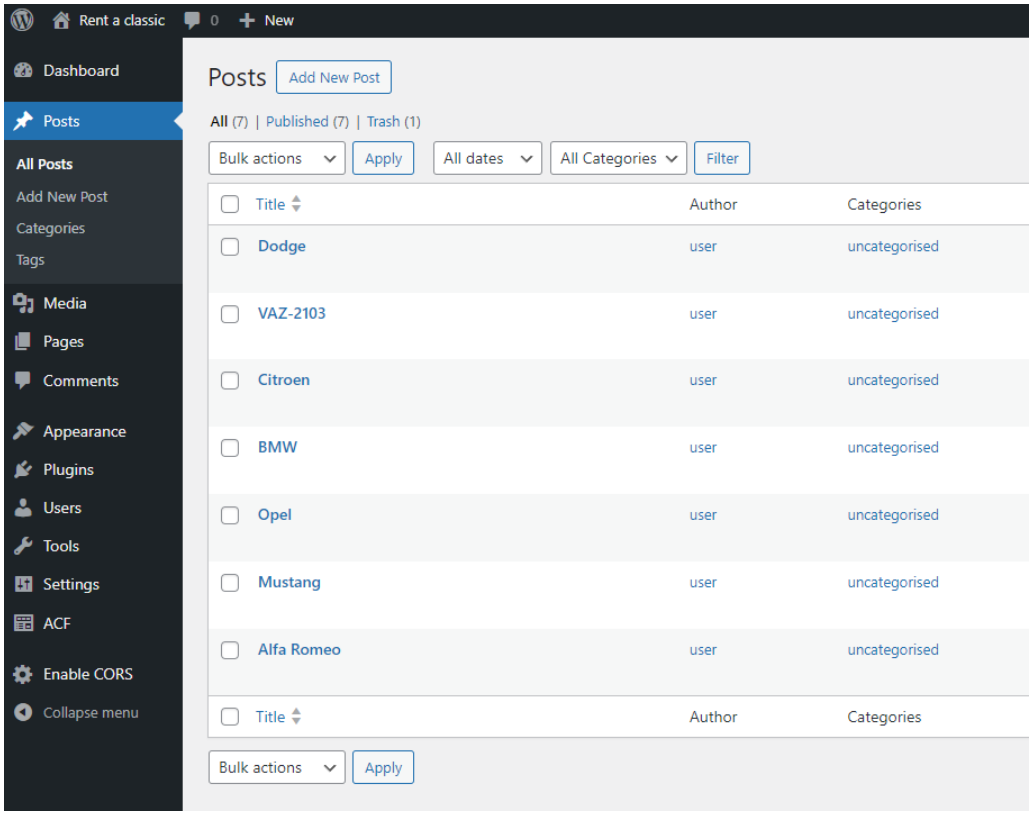
Otvaranjem poveznice „http:/localhost/wordpress“ u web pregledniku otvara se sučelje za postavljanje WordPress CMS-a.



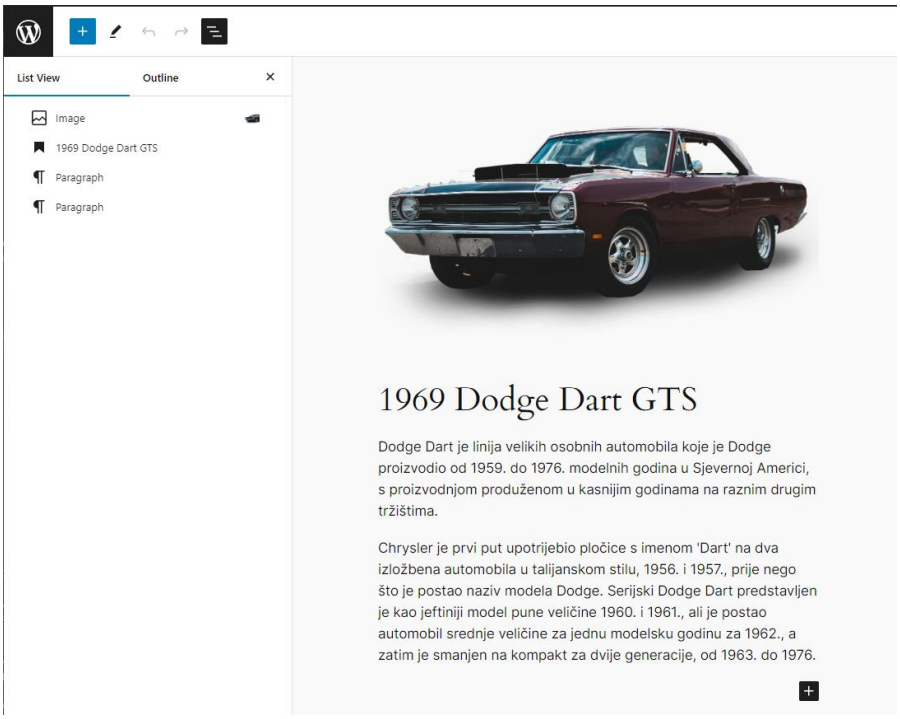
Slika 6.2.4: Zaslon inicijalnog postavljanja WordPress CMS-a, izvor: snimka ekrana, autorov rad

Slika 6.2.4. prikazuje potrebna polja za postavljanje WordPress CMS-a te za njegovo povezivanje s prethodno kreiranom bazom podataka. Ovime je struktura *back-end*-a dovršena i otvara se WordPress CMS sučelje u kojem su vidljive objave, stranice, postavke i mnogo više. U slučaju korištenja headless Wordpress-a, zanimljiv je samo prikaz objava.

Potrebno je kreirati objave koje će kasnije biti vidljive u *front-end*-u aplikacije. Svaka objava će sadržavati uređenu fotografije automobila, godinu proizvodnje i model automobila te kratki opis. Slikom 6.2.5 prikazane su kreirane objave.



Slika 6.2.5: Prikaz objava u WordPress CMS sučelju, izvor: snimka ekrana, autorov rad



Slika 6.2.6: Prikaz jedne od objava u WordPress CMS sučelju, izvor: snimka ekrana, autorov rad

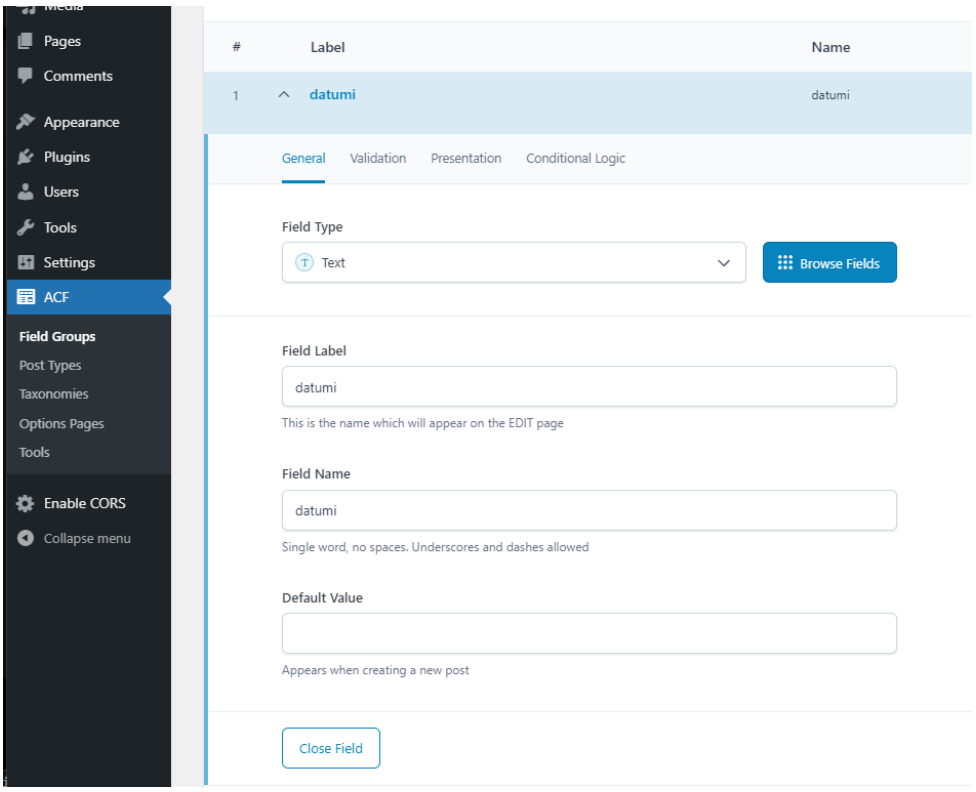
Na slici 6.2.6 prikazana je jedna od kreiranih objava. Stvaranje i uređivanje objava funkcionira na isti način kao i stvaranje web stranica u WP CMS-u, odnosno povlačenjem elemenata i dodavanjem teksta na jednostavan i intuitivan način.

6.2.1. WordPress REST API

Sadržaju WordPress platforme može se pristupiti preko već spomenutog WordPress REST API sučelja. Nakon kreiranja WP stranice i objava, objavama se pristupa preko poveznice „http://localhost/wordpress/wp-json/wp/v2/posts/“. Pojedinoj objavi pristupa se tako da se na kraj poveznice nadoda jedinstveni ključ „*id*“ koji je svakoj objavi dodijeljen automatski nakon kreiranja. Na poveznici se nalazi JSON datoteka koju zatim aplikacija može koristiti za dohvat objava. Pojednostavljeni odgovor u obliku JSON-a za automobil s ključem 21 preko poveznice „http://localhost/wordpress/wp-json/wp/v2/posts/21“ je sljedeći:

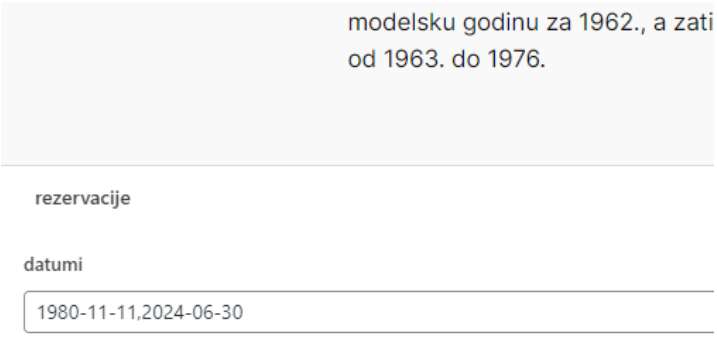

```
{
  "id": 21,
  "date": "2024-06-06T16:24:38",
  "modified": "2024-06-20T10:12:44",
  "type": "post",
  "title": {
    "rendered": "Alfa Romeo"
},
  "content": {
      "rendered":"\figure\width=\"1920\"height=\"1282\"src=\
"http://localhost/wordpress/wp-content/uploads/2024/06/alfa-12.png\"
alt=\"\" \h2\ class=\"wp-block-heading\"\ Alfa Romeo Spider \/h2\ \p\ Alfa
Romeo Spider (serija 105/115) je roadster s dva sjedala, motorom sprijeda
i stražnjim pogonom koji je proizvodio i prodavao Alfa Romeo od 1966. do
1994. u četiri različite generacije ili serije, svaka s modifikacijama u
rasponu od skromnih do opsežnih.\/p\ \p\ Kao nasljednik Giulije Spider,
Spider se proizvodio gotovo tri desetljeća. Prve tri serije sklopio je
Pininfarina u Grugliascu, a četvrtu seriju u San Giorgio Canavese.
Posljednji Spider iz te serije proizveden je u travnju 1993. — posljednji
Alfa Romeo sa stražnjim pogonom prije Alfa Romea 8C Competizione iz
2007.\/p\"
}
```

Dodavanje polja koja će se zatim zasebno prikazati u JSON datoteci odgovora moguće je putem WP *plugin-a* „*Advanced Custom Fields*“. Nakon instalacije *plugin*-a potrebno je kreirati novo polje naziva „datumi“ koje će sadržavati tekst prikazan na slici 6.2.7. Potom je važno uključiti opciju „*Show in REST API*“ kako bi kreirano polje bilo dostupno kasnije u *front-end*-u aplikacije.



Slika 6.2.7: Prikaz kreiranja dodatnog polja,
izvor: snimka ekrana, autorov rad

Dodatno polje može se uređivati na svakoj objavi zasebno na dnu uređivača svake od objava, prikazano na slici 6.2.8. U ovo polje upisuju se datumi odvojeni zarezima, no nije potrebno ručno upisivati datume iz razloga što ova vrijednost služi isključivo za upravljanje rezervacijama.



Slika 6.2.8: Prikaz uređivanja dodanog polja „datumi“,
izvor: snimka ekrana, autorov rad

Polje „datumi“ se nakon kreiranja prikazuje u JSON odgovoru za svaki automobil:

```
"acf": {
   "datumi": "1980-11-11,2024-06-30"
},
```

Ovo polje će kasnije služiti kao lista već rezerviranih datuma u koju će biti moguće i upisivati nove datume putem web aplikacije. Moguće je kreirati i novo polje u obliku liste umjesto teksta, no ta opcija u *plugin*-u „*Advanced Custom Fields*“ se naplaćuje. Iz tog razloga su datumi u tekstualnom formatu odvojeni zarezima te će ih kasnije biti moguće odvojiti i kreirati listu.

6.2.2. Autentifikacija

Kako bi korisnik mogao uređivati sadržaj, odnosno polje „datumi“ na serveru, potrebna je autentifikacija. Za omogućavanje autentifikacije vanjske aplikacije na WordPress CMS, potrebna je instalacija te aktivacija WordPress *plugin*-a „*JWT Auth*“.

Ovaj *plugin* omogućuje autentifikaciju preko REST API-a na jednostavan način, što znači da prijavljeni korisnici dobivaju mogućnost upravljanja određenim sadržajem.

6.3. ReactJS *Front-end*

Pošto će se korisničko sučelje razvijati u ReactJS-u, najprije je potrebno instalirati Node.js te ostale već spomenute alate. Nakon instalacije Node.js-a te otvaranja VS Code uređivača koda, potrebno je kreirati ReactJS aplikaciju pomoću sljedeće komande u terminalu VS Code-a:

```
npx create-react-app naziv-aplikacije
```

Ovime se u mapi u kojoj se terminal nalazi preuzimaju potrebne datoteke i stvara React aplikacija. Sljedeći korak je otvaranje mape u kojoj se nalazi aplikacija, otvaranje mape u VS Code uređivaču te pokretanje same aplikacije.

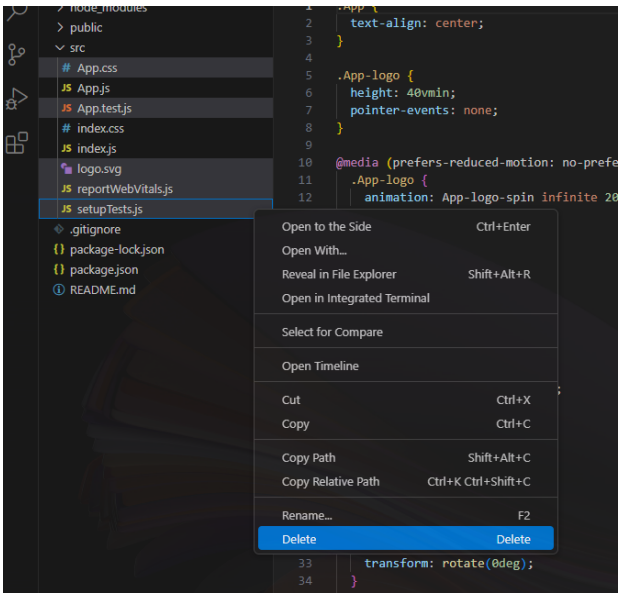
```
cd naziv-aplikacije
code .
npm start
```

Zatim je potrebno instalirati potrebne dodatne biblioteke iz npm biblioteke: *React Router*, *Axios* i *html-react-parser*:

```
npm install react-router-dom
npm install axios
npm install html-react-parser
```

Nakon što je projekt kreiran, prije dodavanja novih komponenata potrebno je ukloniti višak datoteka koji je React automatski preuzeo kao *template*, koje su prikazane na slici 6.3.1.

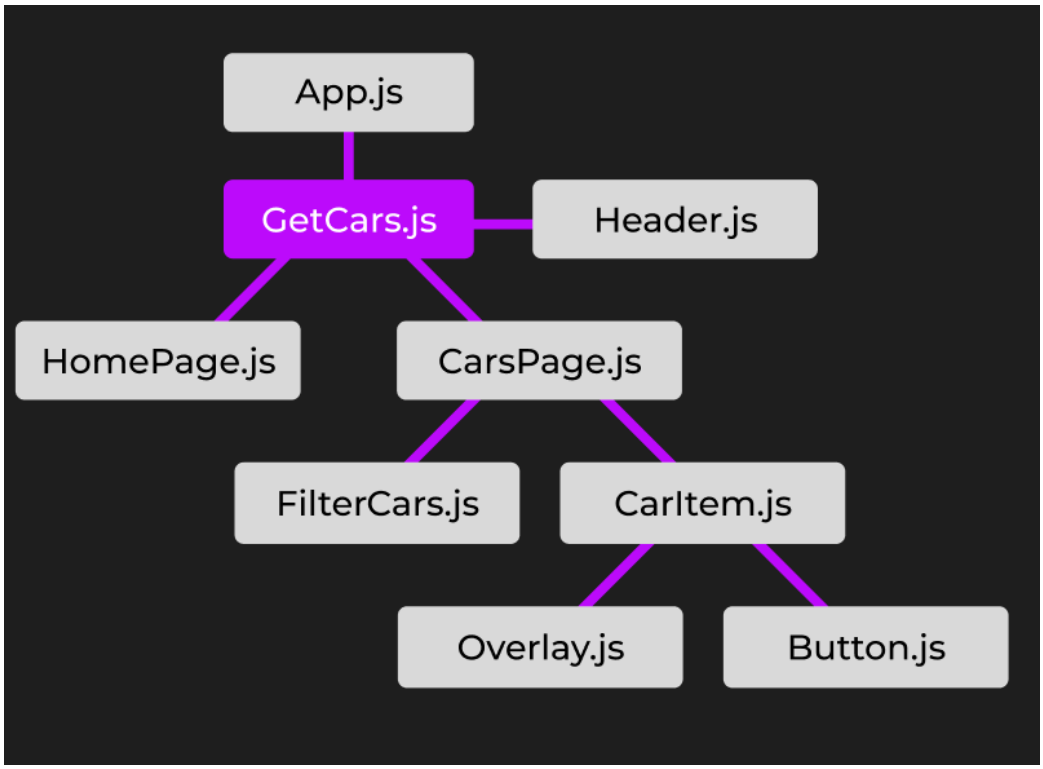
Naknadno je potrebno i ukloniti sve linije koda koje se pozivaju na prethodno izbrisane datoteke kako bi aplikacija ponovo radila bez greški. Nakon što je projekt postavljen na ovaj način, spreman je za dodavanje vlastitih komponenti i stvaranje vlastite web aplikacije od početka.



Slika 6.3.1: Brisanje nepotrebnih datoteka, izvor: snimka ekrana, autorov rad

6.3.1. Potrebne komponente

U fazi planiranja je ustanovljeno koje React komponente će biti potrebne za rad aplikacije na temelju već postavljenih pravila React-a. Na vrhu aplikacije će biti komponenta koja sadrži API poziv kako bi sve komponente imale pristup informacijama o automobilima, čija je hijerarhija prikazana na slici 6.3.2. Svaka komponenta ima svoju funkciju te je potrebno podijeliti aplikaciju na temelju potreba samih komponenti.



Slika 6.3.2: Dijagram web aplikacije Rent-a-Car, izvor: snimka ekrana, autorov rad

U App.js komponenti se uvezuje i prikazuje samo GetCars.js komponenta.

6.3.2. GetCars.js komponenta

GetCars je komponenta koja je *Parent* svim ostalim komponentama te služi kao *Router* koji će upravljati navigacijom.

Na samom vrhu je potrebno uvesti (engl. *import*) potrebne pakete i funkcije iz react biblioteke i react-router-dom biblioteke te stranice:

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import Header from "./Header";

import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import CarsPage from "../pages/CarsPage";
import HomePage from "../pages/HomePage";
```

Zatim je potrebno stvoriti useState *hook* u kojem će biti pohranjeni podaci o automobilu te useEffect *hook* unutar koga će se dohvaćati informacije iz WordPress REST API-a putem axios paketa. API poziv koji je potreban u ovom slučaju šalje zahtjev na već spomenuti *Posts* dio WordPress CMS-a. Također, pri učitavanju stranice korisniku se dodjeljuje JWT token koji se sprema u lokalnu pohranu preglednika, a s kojim se dobiva mogućnost uređivanja objava:

```
const [cars, setCars] = useState([]);

useEffect(() => {
    axios
              .post("http://localhost/wordpress/wp-json/jwt-auth/v1/token",
loginData)
      .then((res) => {
        localStorage.setItem("token", res.data.data.token);
      })
      .catch((err) => {
        console.log(err);
      });
    axios
      .get("http://localhost/wordpress/wp-json/wp/v2/posts")
      .then((response) => {
        setCars(response.data);
        console.log(response.data);
      })
      .catch((error) => {
        console.log(error);
      });
  }, []);
```

Komponenta getCars.js na sučelje web aplikacije vraća Header.js komponentu koja će služiti kao zaglavlje i navigacija te HomePage.js stranicu i CarsPage.js stranicu putem React Router-a.

Pomoću *React Router*-a se omogućuje navigacija do različitih stranice iako se stranica ustvari ne mijenja, ni ponovo učitava. To se postiže sljedećim kodom:

```
return (
  <div>
    <Router>
      <Header />
      <div className="container">
        <Routes>
          <Route path="/" element={<HomePage carInfo={cars} />} />
          <Route path="/cars" element={<CarsPage carInfo={cars} />} />
        </Routes>
      </div>
    </Router>
  </div>
);
```

Na obje stranice prosljeđuju se informacije o automobilima putem *props*-a. To omogućuje stranicama da u vlastitom kodu mogu koristiti varijablu *cars* na bilo koji način, no ne mogu joj mijenjati vrijednost što osigurava da podaci ostanu identični kao i u WordPress CMS bazi podataka.

6.3.3. Header.js komponenta

Ova jednostavna komponenta služi, kao što je već opisano, kao navigacija između različitih stranica. Potrebno je uvesti korištene pakete i komponente. Lijevo će biti logo koji također služi kao poveznica na početnu stranicu, a desno će biti navigacijska traka. To se postiže sljedećim kodom:

```
<Link to="/">
  <div className="logo">
    <img src={logo} alt="logo"></img>
    <h3>Rent-a-classic</h3>
  </div>
</Link>
<div className="navigation">
  <NavLink to={"/"}>Početna</NavLink>
  <NavLink className={"gumb"} to={"/cars"}>
    Pretraži vozila
  </NavLink>
</div>
```

6.3.4. HomePage.js komponenta

HomePage.js vraća sve elemente naslovne stranice, a sastoji se od naslova, slike automobila i dinamičnog gumba koji mijenja sadržaj ovisno o broju trenutno dostupnih automobila:

```
const carCount = carInfo.length;
```

Na temelju broja automobila, mijenja se prikaz gumba ispod naslova u kojemu se prikazuje varijabla koja sadrži broj dostupnih automobila te se na temelju te varijable postavlja tekst gumba na točnu vrijednost putem sljedećeg koda:

```
<Link to={"/cars"}>
  Pretraži{" "}
  <span className="bold">
    {carCount}{" "}
    {carCount === 1
      ? "dostupno"
      : carCount < 5
      ? "dostupna"
      : "dostupnih"}
  </span>{" "}
  vozila
</Link>
```

6.3.5. CarsPage.js komponenta

Stranica CarsPage.js vraća dvije komponente, FilterCars.js koja služi kao komponenta za sortiranje automobila po dostupnosti te CarItem.js koja se prikazuje onoliko puta koliko automobila ima te svaka sadrži jedan automobil.

U ovoj stranici se nalazi useState varijabla koja omogućuje komponenti CarItem.js da se obnovi na temelju izbora u komponenti FilterCars.js. To se postiže korištenjem useState varijable te prosljeđivanja funkcije „*setDatumFilter*“ FilterCars komponenti i postavljene varijable „*datumFilter*“ CarItem komponenti pod *props*:

```
const [datumFilter, setDatumFilter] = useState(null);
return (
  <div>
    <div className="filter">
      <FilterCars setDatumFilter={setDatumFilter} />
    </div>
    <div className="car-items">
      <CarItem carInfo={carInfo} datumFilter={datumFilter}></CarItem>
    </div>
    <div className="footer"></div>
  </div>
);
```

6.3.6. FilterCars.js komponenta

U ovoj komponenti se na temelju odabira korisnika primjeni željeni filter na sadržaj. Varijabla date pohranjuje trenutan odabir korisnika u komponenti „*input*“. Funkcija onChange() omogućuje obnavljanje varijable svaki puta kada se promjeni unos. Unos se odrađuje preko kalendara, klikom na gumb „Filtriraj“ se upisani datum pohranjuje u „datumFilter“, a klikom na gumb „Poništi“ se briše:

```
const [date, setDate] = useState(null);
return (
  <div className="filter-container">
    <input
      type="date"
      value={date}
      onChange={(e) => setDate(e.target.value)}
    ></input>
    <button onClick={() => setDatumFilter(date)}>Filtriraj</button>
    <button onClick={() => setDatumFilter(null)}>Poništi</button>
  </div>
);
```

6.3.7. CarItem.js komponenta

Ova komponenta je zaslužna za prikaz jednog automobila te sadrži Button.js komponentu kao i Overlay.js komponentu kako bi se mogao otvoriti za specifičan automobil ovisno o odabranom automobilu. Iz razloga što je objava koja se dohvaća iz WordPress CMS-a u obliku teksta kao što je prikazano u poglavlju 6.2.1, potrebno ju je podijeliti na dijelove pomoću *parser*-a te ukloniti HTML *tag*-ove kako bi sam tekst objave postao dostupan. Funkcija za uklanjanje HTML *tag*-ova:

```
const removeHTML = (s) => {
  const regex = /<(.+?)>(.+?)<\/.+?>/g;
  let match = regex.exec(s);
  const result = [];
  while (match != null) {
    result.push(match[2]);
    match = regex.exec(s);
  }
  return result;
};
```

Prima *props* „carInfo“ za informacije o svim automobilima te „datumFilter“ koji je vrijednost iz FilterCars.js komponente, na temelju kojeg se prikazuje dostupnost automobila putem mjenjanja klasa koje dobiva. Potrebno je provjeriti koji automobil u prethodno dodanom polju „Datumi“ sadrži datum koji je trenutno odabran u kalendaru, stoga se šalje upit serveru u kojem se ispituje sadrži li pojedini automobil odabrani datum. Ako sadrži potrebno je prikazati da je nedostupan, stoga se u listu „nedostupni“ pohranjuje njegov ključ, koji se zatim pohranjuje u useState funkciju naziva nedostupniID:

```
useEffect(() => {
  setNedostupniID([]);
  carItems.map((car) =>
    axios
      .get(`http://localhost/wordpress/wp-json/wp/v2/posts/${car.id}`)
      .then((res) => {
        if (res.data.acf.datumi.includes(datumFilter)) {
          nedostupni.push(car.id);
          setNedostupniID([...nedostupni]);
        }
      })
  );
}, [datumFilter]);
```

Zatim se prikazuju automobili kao dostupni ili nedostupni ovisno o tome jesu li dostupni na odabrani datum, odnosno ako je njihov ključ pohranjen u listi nedostupniID:

```
carItems.map((car) =>
  nedostupniID.includes(car.id) ? (
    <div
      key={car.id}
      className={"car-unavailable"}
      onClick={() => {
        handleClick(car.id);
      }}
    >
      <div className="img-container">
        {car.content.rendered &&
          parse(removeHTML(car.content.rendered)[0])}
      </div>
      <h3>
        {car.content.rendered &&
          parse(removeHTML(car.content.rendered)[1])}
      </h3>
      <div>
        <Button />
      </div>
    </div>
  ) : (
    <div
      key={car.id}
      className={"car-available"}
      onClick={() => {
        handleClick(car.id);
      }}
    >
      <div className="img-container">
        {car.content.rendered &&
          parse(removeHTML(car.content.rendered)[0])}
      </div>
      <h3>
        {car.content.rendered &&
          parse(removeHTML(car.content.rendered)[1])}
      </h3>
      <div>
        <Button carAvailable={car.categories}></Button>
      </div>
    </div>
  )
)}

{overlayState && (
  <Overlay
    carItems={carItems}
    currentCar={currentKey}
    setOverlayState={setOverlayState}
    removeHTML={removeHTML}
  />
)
```

HandleClick funkcija pamti koji automobil je pritisnut te otvara Overlay.js komponentu ukoliko korisnik klikne na dostupni automobil:

```
const handleClick = (id) => {
  setCurrentKey(id);
  setOverlayState(true);
  window.scrollTo(0, 0);
};
```

6.3.8. Overlay.js i funkcionalnost iznajmljivanja

Overlay.js komponenta ima proslijeđene informacije o automobilu te na temelju toga, kada je otvorena, filtrira automobile te ponovo na isti način prikazuje dodatne informacije:

```
<div className="x" onClick={handleClose}>X</div>
{carItems.filter((carItems) => carItems.id === currentCar)
 .map((filteredCar) => (
    <div className="overlay-content">
      <div>
        <h1>
          {filteredCar.content.rendered &&
            parse(removeHTML(filteredCar.content.rendered)[1])}
        </h1>
        <p>
          {filteredCar.content.rendered &&
            parse(removeHTML(filteredCar.content.rendered)[2])}
        </p>
        <p>
          {filteredCar.content.rendered &&
            parse(removeHTML(filteredCar.content.rendered)[3])}
        </p>
      </div>
      <div className="img-overlay">
        {filteredCar.content.rendered &&
          parse(removeHTML(filteredCar.content.rendered)[0])}
      </div>
      <div className="iznajmibid">
        <input
          type="date"
          placeholder="dd-mm-yyyy"
          value={date}
          onChange={(e) => setDate(e.target.value)}
        ></input>
        <button
          className="gumb gumb-overlay"
          onClick={() => handleIznajmi()}
        >Iznajmi</button>
      </div>
    </div>
  ))}
```

Na dnu *overlay*-a nalaze se unos datuma i gumb. Klikom na gumb provjerava se je li korisnik unio datum, ako je, provjerava je li automobil iznajmjen na taj datum, upitom koji gleda ima li automobil već taj datum u polju „datumi“ na serveru. Zatim, ako nije, pohrani sve već upisane datume rezervacija, dodaje novi datum ukoliko već nije prisutan te vraća nove podatke na server. Prvo se pokreće funkcija *updateNewPost()* , a zatim *updatePost()*:

```
const updateNewPost = async () => {
  if (date) {
    axios
      .get(`http://localhost/wordpress/wp-json/wp/v2/posts/${currentCar}`)
      .then((res) => {
        rezervacije = [res.data.acf.datumi.split(",")];
        if (!rezervacije[0].includes(date)) {
          rezervacije[0].push(date);
          console.log("a ", sdarray);
          if (splitDatum) {
            splitDatumString = rezervacije.toString();
          }
          sdarrayState = true;
        }
      });
  }
};

const updatePost = () => {
  if (sdarrayState) {
     axios
     .post(`http://localhost/wordpress/wp-json/wp/v2/posts/
     ${currentCar}`,
        { acf: { datumi: splitDatumString } },
        {
          headers: {
            "Content-Type": "application/json",
            Authorization: `Bearer ${localStorage.getItem("token")}`,
          },
        }
      )
      .then(() => {
        alert("Datum iznajmljivanja uspješno potvrđen!");
      })
      .catch((err) => {
        console.log(err);
      });
  } else {
    alert( "Nažalost, na taj dan automobil nije dostupan. Molimo da
    pretražite automobile prema željenom datumu." );
  }
};
```

Ovdje se koristi autentifikacija koja se dodjeljuje korisniku u komponenti GetCars.js, te kojom se dobiva pristup za uređivanje polja „datumi“. Korisnika se porukom u pregledniku obavještava porukom da taj datum nije dostupan ukoliko automobil već ima taj datum upisan, i porukom da je rezervacija prošla uspješno ukoliko nema.

6.3.9. Dodavanje CSS-a

CSS stil se dodaje u App.js komponenti putem *import*-a*:*

```
import "./style.css";
```

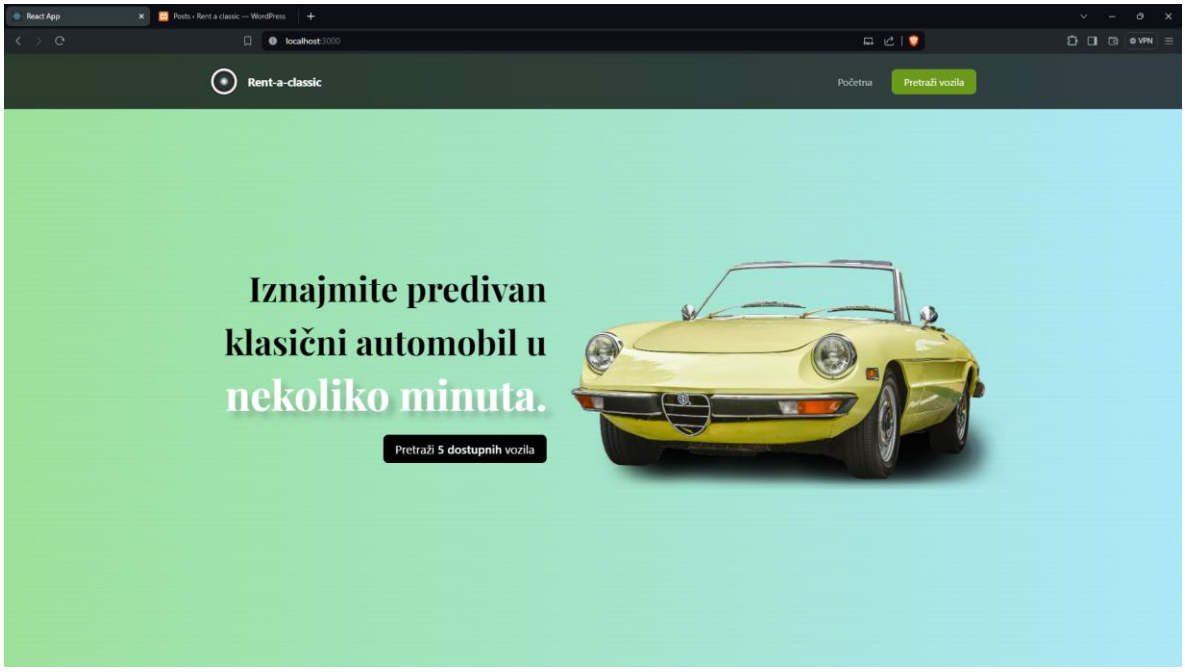
CSS je opisni jezik za opisivanje strukture i sadržaja HTML dokumenata te u slučaju ReactJS aplikacije funkcionira jednako kao i kod razvoja tradicionalnih web stranicama. Jedina razlika je što se u ReactJS-u klase elementima dodaju putem *tag*-a „*className*“ umjesto „*class*“ kao što je u HTML-u. Klase se također mogu mijenjati, što treba uzeti u obzir, na primjer, ako je u ReactJS kodu napisano:

```
className={varijabla ? "stil1" : "stil2"}
```

Kada je vrijednost varijable „*true*“, elementu će se dodijeliti klasa „stil1“ i obratno. Stoga je i CSS kod potrebno pisati s obzirom na sva moguća stanja komponenti aplikacije. Nakon pisanja CSS-a za sve elemente u datoteci *„style.css*“, web aplikacija poprima završni izgled kao i funkcionalnost.

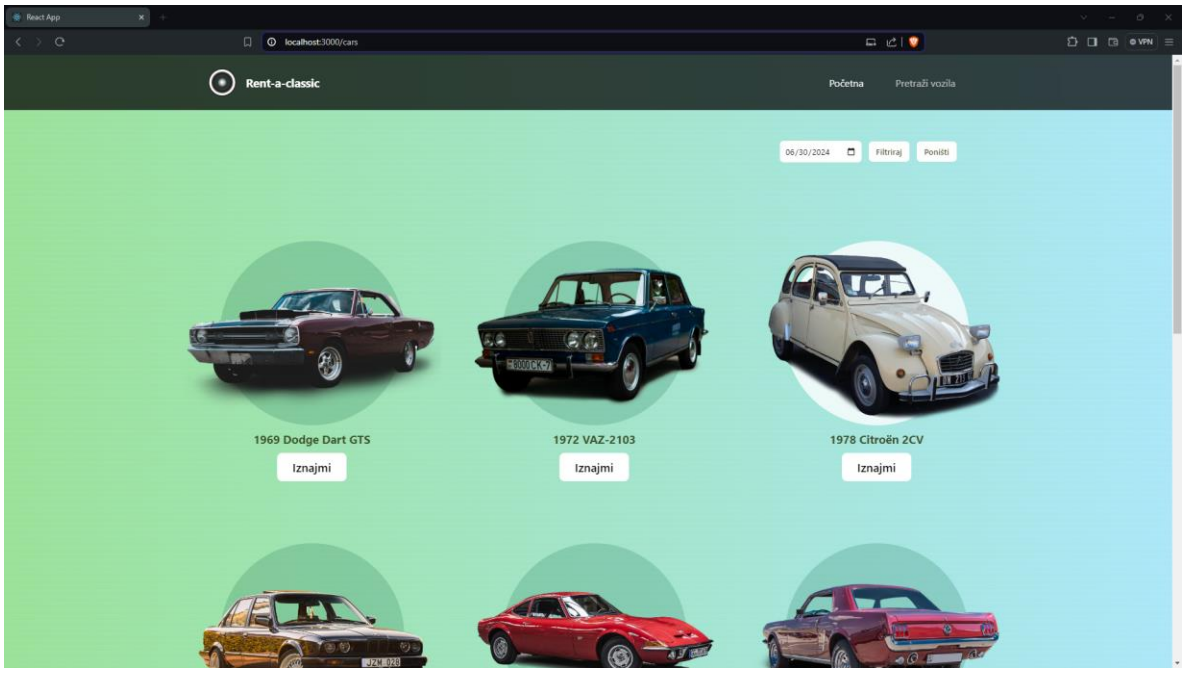
7. Prikaz razvijene web aplikacije

7.1. Zasloni

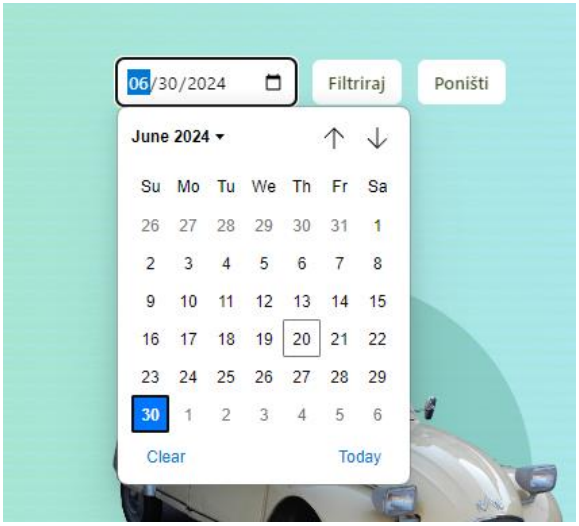


Slika 7.1.1: Prikaz početne stranice web aplikacije Rent-a-Car, izvor: snimka ekrana, autorov rad

Na slici 7.1.1 je prikaz početne stranice gdje je vidljiv dinamični gumb koji se mijenja ovisno o broju objava, odnosno automobila, u WordPress CMS-u. Na slici 7.1.2 je prikaz ekrana koji se otvara nakon klika na gumb, kao i *hover* efekt na automobilu.

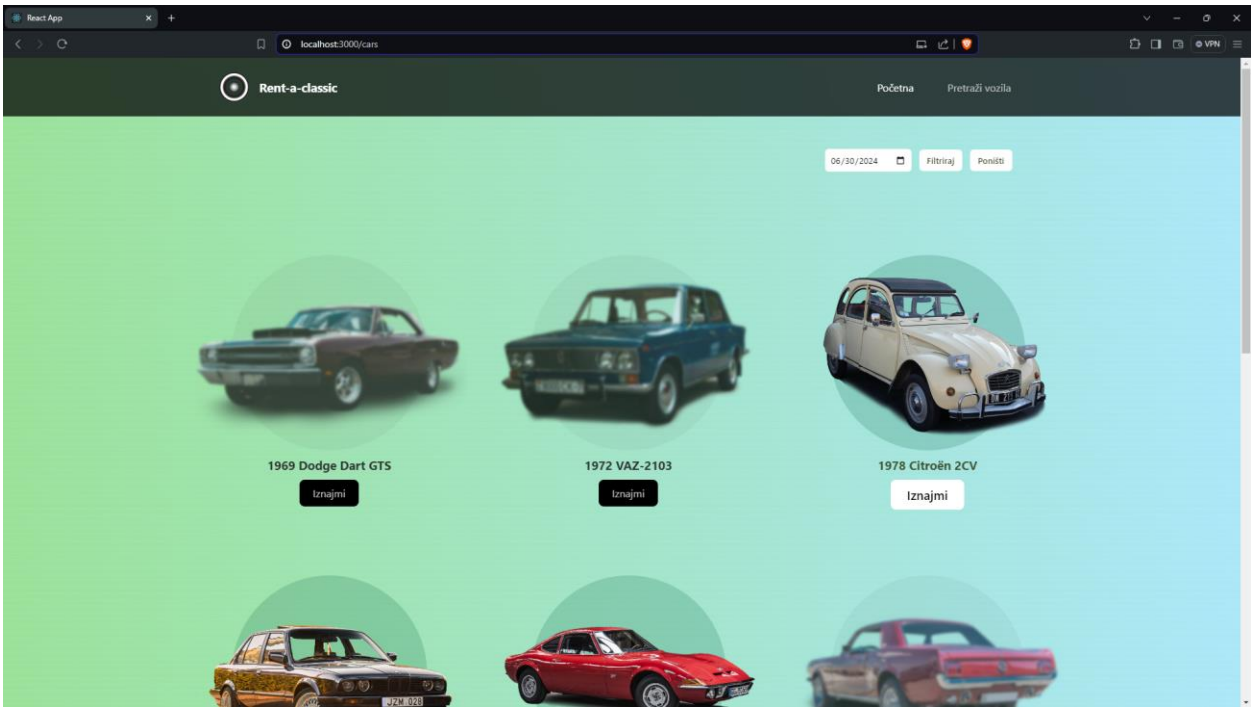


Slika 7.1.2: Prikaz stranice automobila web aplikacije Rent-a-Car, izvor: snimka ekrana,



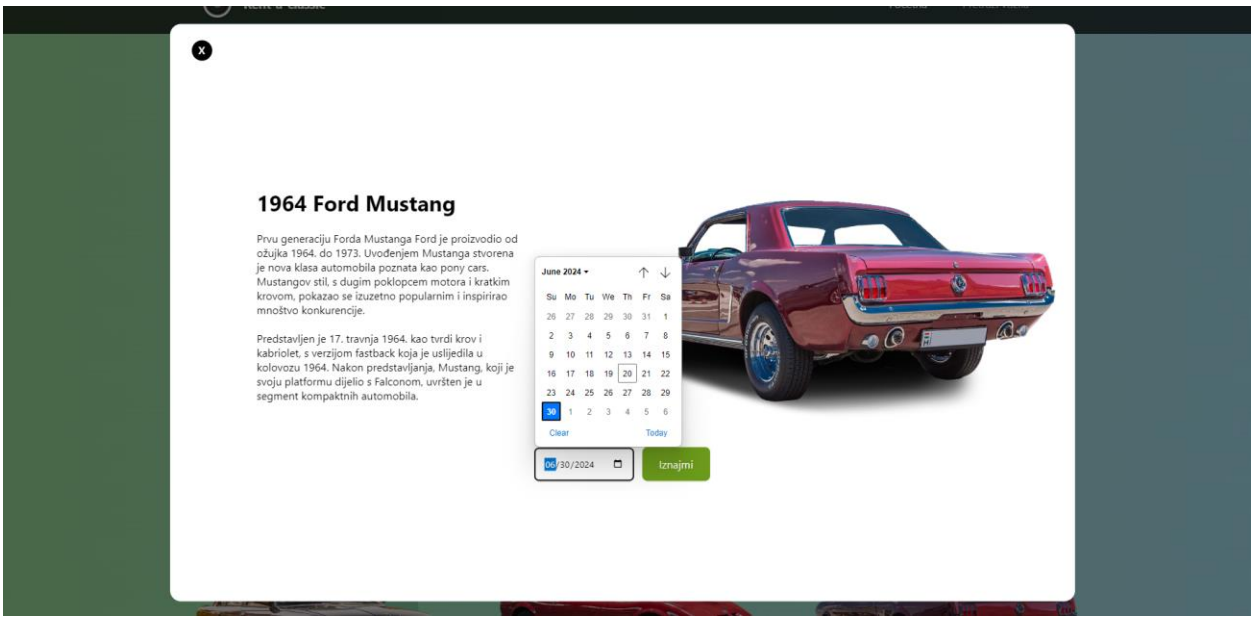
Slika 7.1.3: Prikaz odabira datuma, izvor: snimka ekrana, autorov rad

Klikom na izbornik datuma otvara se kalendar u kojem je moguće odabrati datum kao što je prikazano slikom 7.1.3. Zatim se klikom na gumb „Filtriraj“ otvara ekran ovisno o odabranom datumu i dostupnim automobilima na taj datum, a odabranog datuma je to ekran prikazan slikom 7.1.4.



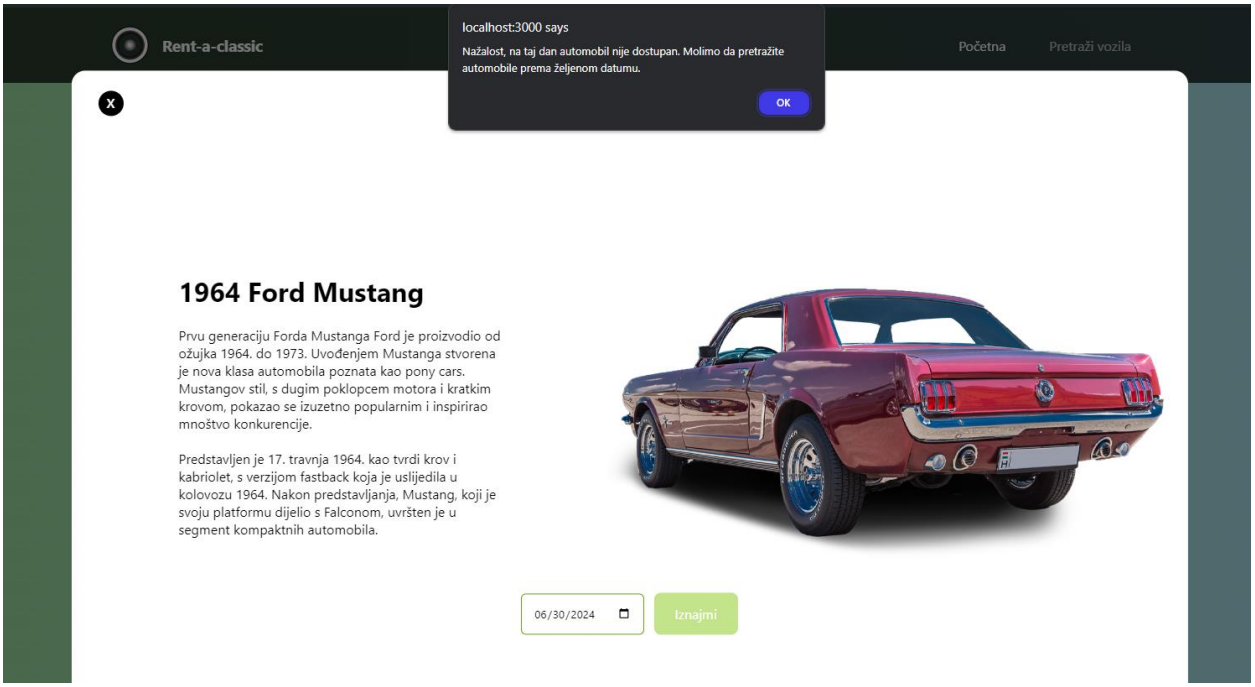
Slika 7.1.4: Prikaz automobila dostupnih datuma 30.6.2024., izvor: snimka ekrana, autorov rad

Klikom na gumb „Poništi“ se ponovo prikazuje ekran na slici 7.1.2 te su svi automobili ponovo dostupni.

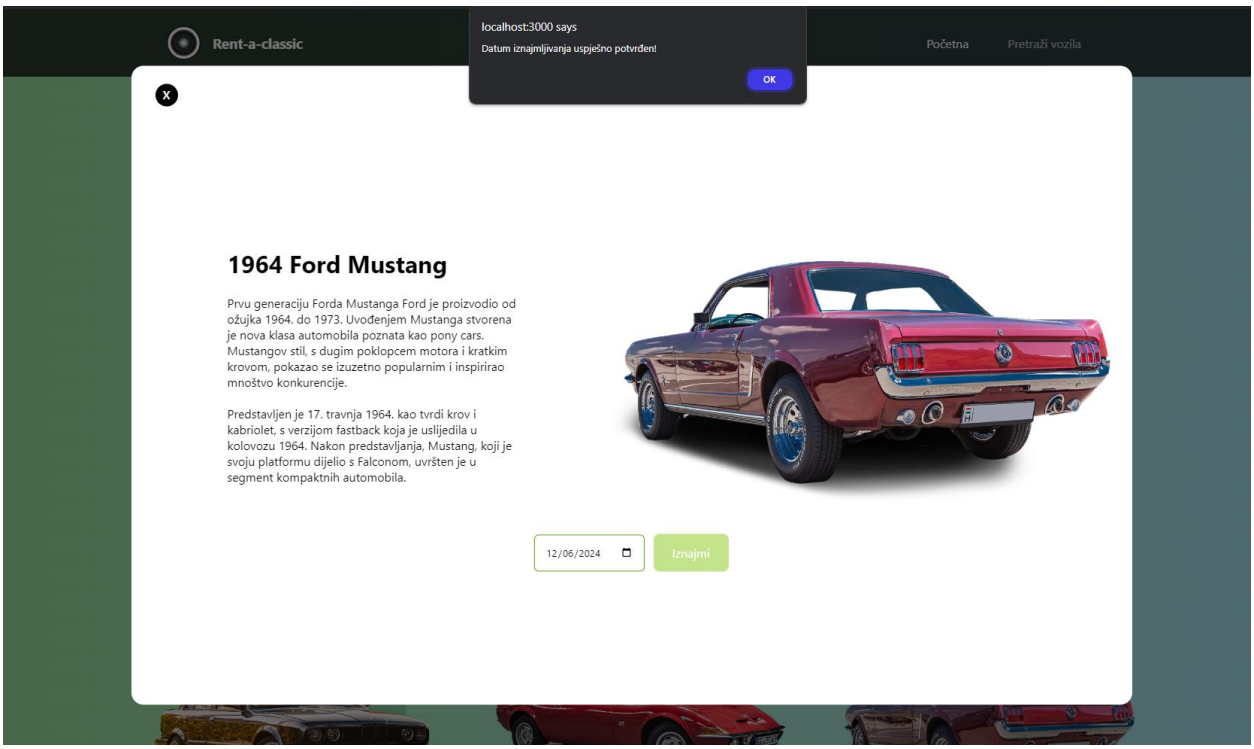


Slika 7.1.5: Prikaz overlay-a., izvor: snimka ekrana, autorov rad

Na slici 7.1.5 prikazan je *overlay* koji se otvara klikom na „Iznajmi“ ili na sam automobil, sa otvorenim unosom datuma. Na slici 7.1.4 vidljivo je da odabrani automobil nije dostupan na odabrani datum, stoga se klikom na gumb „Iznajmi“ pojavljuje poruka prikazana slikom 7.1.6.



Slika 7.1.6: Prikaz poruke kada je automobil nedostupan., izvor: snimka ekrana, autorov rad



Slika 7.1.7: Prikaz poruke kada je automobil uspješno rezerviran., izvor: snimka ekrana, autorov rad

Ukoliko korisnik odabere datum na koji je automobil dostupan te je rezervacija uspješno provedena, pojavljuje se poruka prikazana slikom 7.1.7.

8. Zaključak

Kod izrade ovog rada uočljiva je jednostavnost i skalabilnost korištenja ovih dvaju tehnologija zajedno. WordPress CMS omogućuje vrlo brzo i lako upravljanje objavama za razliku od ručnog upisivanja polja u bazu podataka, dok ReactJS pruža daleko više mogućnosti za zanimljiva i dinamična korisnička sučelja koja bi bilo nemoguće izraditi u samom WordPress-u. Također, ReactJS u odnosu na WordPress *template* web aplikacije pruža daleko bolje performanse i vrijeme učitavanja iz razloga što se komponente mogu učitavati samo kada su pozvane te ne troše resurse računala konstantno u pozadini. Korištenje ReactJS-a za uvjetno prikazivanje elemenata na web aplikaciji pruža više mogućnosti za dinamične web aplikacije nego ikad te je primjetna velika razina kontrole, ali i organizacije te urednosti samog koda.

Uz svu dostupnu dokumentaciju o korištenim tehnologijama, razvoj web aplikacije je napredovao vrlo brzo, no primijećen je nedostatak informacija o specifično ovoj kombinaciji tehnologija što je predstavilo dodatan izazov. Jedan od izazova pri izradi aplikacije bio je način na koji WordPress REST API dostavlja informacije o objavama. Naime, podaci pristižu u jedinstvenom stringu, dok bi za potrebe aplikacije bilo bolje da pristižu zasebno. No, taj nedostatak riješen je jednostavnom intervencijom u smislu reformatiranja pristiglih podataka na način koji je primjenjiv unutar razvijene aplikacije. Nedostatak ovog pristupa je što se gubi mogućnost dodavanja klasa na specifične dijelove elementa, na primjer dodavanje „<*span*>” elementa usred odlomka na određeni dio teksta. Uz dodatne korekcije u funkciji za čišćenje HTML koda unutar *string*-a i taj bi problem postao zanemariv, no to je jedini primijećeni nedostatak.

Zaključak rada je da se WordPress CMS i ReactJS izvrsno nadopunjuju u kontekstu izrade skalabilne web aplikacije. WordPress CMS je izvrstan u kontekstu brzog razvoja *back-end* sustava, a ReactJS za izradu interaktivnog *front-end* sučelja. Pokazalo se da modularni pristup izradi web aplikacije primjenom tih alata pokazuje niz prednosti u odnosu na klasičan pristup gdje se koristi WordPress tema, odnosno gdje je arhitektura aplikacije monolitna. U prikazanoj konfiguraciji WordPress CMS djeluje kao headless CMS i podatke dostavlja ReactJS *front-end*-u putem API sučelja. Time se postiže jednostavnost u smislu arhitekture aplikacije i mogućnosti modularnog razvoja, no istovremeno se situacija komplicira jer WordPress CMS podrazumijeva primjenu relacijske baze podataka i jezika PHP, dok se ReactJS temelji na jeziku JavaScript i nizu pomoćnih alata. Može se zaključiti da prikazana metodologija daje mnoštvo prednosti u smislu razvoja web aplikacija, no njezina primjena je u sferi profesionalnog web dizajna koji zahtijeva proizvode visokih performansi.

9. Literatura

[1] Rohit Shewale, » Facebook Users Statistics 2024 (Worldwide Data) « 5. 4. 2024. [Mrežno]. Dostupno: https://www.demandsage.com/facebook-statistics/. [Pokušaj pristupa 2. 6. 2024.].

[2] Casey Frechette, » What journalists need to know about the difference between Web apps and native apps « 11. 4. 2023. [Mrežno]. Dostupno: https://www.poynter.org/reporting-editing/2013/what-journalists-need-to-know-about-the-difference-between-web-apps-and-native-apps/. [Pokušaj pristupa 2. 6. 2024.].

[3] Emma, » Frontend vs Backend programeri: u čemu je razlika? « 5. 4. 2024. [Mrežno]. Dostupno: https://rck.elpros.net/frontend-vs-backend-programeri-u-cemu-je-razlika/. [Pokušaj pristupa 2. 6. 2024.].

[4] David Herbert, » What is React.js? Uses, Examples, & More « 13. 11. 2023. [Mrežno]. Dostupno: https://blog.hubspot.com/website/react-js. [Pokušaj pristupa 3. 6. 2024.].

[5] React, » Writing Markup with JSX « 3. 6. 2024. [Mrežno]. Dostupno: https://react.dev/learn/writing-markup-with-jsx/. [Pokušaj pristupa 3. 6. 2024.].

[6] Liva Jorge, » Understanding React Parent Components: Best Practices and Common Techniques « 13. 3. 2023. [Mrežno]. Dostupno: https://medium.com/@livajorge7/introduction-9e84391f4b83. [Pokušaj pristupa 4. 6. 2024.].

[7] React, » Built-in React Hooks « 3. 6. 2024. [Mrežno]. Dostupno: https://medium.com/@livajorge7/introduction-9e84391f4b83. [Pokušaj pristupa 4. 6. 2024.].

[8] geeksforgeeks, » What is react-router-dom ? « 22. 4. 2024. [Mrežno]. Dostupno: https://medium.com/@livajorge7/introduction-9e84391f4b83. [Pokušaj pristupa 4. 6. 2024.].

[9] Kinsta, » What is a content management system (CMS)? « 8. 3. 2023. [Mrežno]. Dostupno: https://kinsta.com/knowledgebase/content-management-system/. [Pokušaj pristupa 7. 6. 2024.].

[10] Gatsby, » What is Headless WordPress? « 7. 6. 2023. [Mrežno]. Dostupno: https://www.gatsbyjs.com/docs/glossary/headless-wordpress/. [Pokušaj pristupa 7. 6. 2024.].

[11] Amazon, » What is an API (Application Programming Interface)? « 12.1.2024.. [Mrežno]. Dostupno: https://aws.amazon.com/what-is/api/. [Pokušaj pristupa 8. 6. 2024.].

[12] WordPress, » REST API Handbook « 8. 6. 2023. [Mrežno]. Dostupno: https://developer.wordpress.org/rest-api/. [Pokušaj pristupa 8. 6. 2024.].

10. Popis slika

Sveučilište Sjever

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Tomislav Martinez *(ime i prezime)* pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog *(obrisati nepotrebno)* rada pod naslovom Izrada web aplikacije Rentacan pomoću alata ReactJS i WordPress CMS *(upisati naslov)* te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:
(upisati ime i prezime)

Tomislav Martinez
(vlastoručni potpis)

Sukladno čl. 83. Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Sukladno čl. 111. Zakona o autorskom pravu i srodnim pravima student se ne može protiviti da se njegov završni rad stvoren na bilo kojem studiju na visokom učilištu učini dostupnim javnosti na odgovarajućoj javnoj mrežnoj bazi sveučilišne knjižnice, knjižnice sastavnice sveučilišta, knjižnice veleučilišta ili visoke škole i/ili na javnoj mrežnoj bazi završnih radova Nacionalne i sveučilišne knjižnice, sukladno zakonu kojim se uređuje znanstvena i umjetnička djelatnost i visoko obrazovanje.