VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

Primerjava JavaScript knjižice React.js z JavaScript ogrodjem Angular

Kandidat: Matic Kerec

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: Aljaž Sahornik, dipl. inž. kom. (UN)

Lektor: Patricija Bračič, mag. prof. slov. jez. in knjiž.

Maribor, 2024

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Matic Kerec, sem avtor diplomskega dela z naslovom Primerjava knjižice React.js z ogrodjem Angular, ki sem ga napisal pod mentorstvom mag. Ervin Schaffa.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel/a, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli, kot moje lastne kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, junij 2024 Podpis študenta:

ZAHVALA

Rad bi se zahvalil svojemu mentorju mag. Ervin Schaffu, ki mi je pomagal s strokovno pomočjo, prav tako pa mi je hitro in skrbno odgovarjal na vsa vprašanja, povezana z diplomsko nalogo.

Zahvaljujem se svoji družini in vsem prijateljem, ki so me podpirali v času študija.

POVZETEK

Diplomska naloga proučuje razlike med JavaScript knjižico React.js in JavaScript ogrodjem Angular s poudarkom na njuni uporabni vrednosti. Diplomska naloga zajema ustvarjanje dveh primerljivih spletnih aplikacij z uporabo obeh orodij in pregled delovanja obeh tehnologij pri obdelavi Big Data. Pregledali smo učno krivuljo, dostopnost učnih virov in spletnih tečajev, podporo skupnosti ter kompleksnost vsakega orodja.

Diplomsko delo se osredotoča na različne načine shranjevanje podatkov. V njem sta predstavljena dva pristopa shranjevanja podatkov s poudarkom na uporabi z veliko količino podatkov (Big Data). Prav tako primerjamo podatkovna tipa baz SQL in NoSQL v kontekstu njune uporabe za Big Data in primerjamo, katera bi bila boljša izbira za delo z Big Data.

V diplomski nalogi smo prav tako predstavili in primerjali dve vodilni JavaScript tehnologiji, React in Angular, kar je bil tudi naš glavni cilj. Skozi primerjalno analizo smo identificirali prednosti, slabosti in primerljivost obeh ogrodij v različnih kontekstih, kot so učinkovitost, hitrost in uporabniška izkušnja. Pri tem smo se osredotočili tudi na njuno delovanje pri obdelavi velikih količin podatkov (Big Data), kar je omogočilo celovit vpogled v njuno zmogljivost in prilagodljivost pri obravnavi kompleksnih podatkovnih izzivov.

V okviru raziskave smo se osredotočili tudi na vlogo Big Data v kontekstu razvoja aplikacij v Reactu in Angularju. Raziskali smo njuno definicijo, različne formate podatkov, s katerimi se srečujemo pri delu z velikimi količinami podatkov, ter obravnavali izzive, povezane z njihovo hitrostjo in obdelavo. Posebej smo analizirali, kako se lahko tehnologiji React in Angular prilagodita za učinkovito obdelavo in prikazovanje velikih količin podatkov. Ugotovili smo, da lahko oba okolja omogočata učinkovito uporabo Big Data, pri čemer je izbira med njima odvisna predvsem od specifičnih zahtev projekta in strokovnih znanj razvijalcev.

V zaključku diplomske naloge smo razvili dve aplikaciji za predstavitev Pokemonov, pri čemer smo eno izdelali v okolju React, drugo pa v Angularju. Glavni cilj raziskave je bil primerjati funkcionalnosti, učinkovitost in uporabniško izkušnjo obeh aplikacij ter ugotoviti, katera platforma je bolj primerna za izdelavo tovrstnih aplikacij. Na koncu smo izvedli tudi meritve, ki so nam pomagale oceniti, katera aplikacija je boljša glede na različne vidike, kot so hitrost delovanja, odzivnost in enostavnost uporabe.

Ključne besede: React, Angular, Podatkovne baze, JavaScript ogrodja, Big Data

ABSTRACT

Comparison of the javascript library React.js with the javascript framework Angular

The thesis examines the differences between the JavaScript library React.js and the JavaScript framework Angular, with an emphasis on their practical value. It encompasses the creation of two comparable web applications using both tools and a review of the performance of both technologies in handling Big Data. I reviewed the learning curve, accessibility of learning resources and online courses, community support, and the complexity of each tool.

The thesis focuses on different methods of data storage. It presents two data storage approaches with an emphasis on handling large amounts of data (Big Data). We also compare the SQL and NoSQL database types in the context of their use for Big Data and compare which would be a better choice for working with Big Data.

Additionally, we presented and compared two leading JavaScript technologies, React and Angular, which was also our main goal. Through comparative analysis, we identified the strengths, weaknesses, and comparability of both frameworks in various contexts, such as efficiency, speed, and user experience. We also focused on their performance in handling large amounts of data (Big Data), providing a comprehensive insight into their capability and adaptability in addressing complex data challenges.

As part of the research, we also focused on the role of Big Data in the context of application development in React and Angular. We explored its definition, various data formats encountered when working with large amounts of data, and addressed challenges related to their speed and processing. We specifically analyzed how the React and Angular technologies can be adapted for efficient processing and visualization of large amounts of data. We found that both environments can enable effective use of Big Data, with the choice between them depending primarily on the specific requirements of the project and the developers' expertise.

In the conclusion of the thesis, we developed two applications for presenting Pokémon, one built in React and the other in Angular. The main objective of the research was to compare the functionalities, efficiency, and user experience of both applications and to determine which platform is more suitable for developing such applications. Finally, we conducted measurements to help us evaluate which application performs better in various aspects such as speed, responsiveness, and ease of use.

Keywords: React, Angular, Databases, JavaScript frameworks, Big data

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KAZALO GRAFOV

1 UVOD

Programski jezik JavaScript obstaja že vrsto let in je eden najbolj znanih in najpogosteje uporabljen front-end programski jezik v spletnem razvoju. Vendar se JavaScript še vedno razvija in s pojavom ogrodij JavaScript (JSF) je prišlo do velike spremembe v tem, kako razvijalci danes razvijajo programsko opremo.

V sodobnem digitalnem okolju, kjer se tehnološki napredek odvija s svetlobno hitrostjo, je obvladovanje in analiziranje obsežnih podatkovnih zbirk postalo ključnega pomena. Pojem "big data" ne predstavlja zgolj količine podatkov, temveč tudi izzive, ki izhajajo iz njihove kompleksnosti, hitrosti pridobivanja in raznovrstnosti. Zato se vedno več organizacij in razvijalcev odloča za uporabo specializiranih orodij in ogrodij, ki omogočajo učinkovito obdelavo in analizo velikih količin podatkov.

V svetu spletnega razvoja prevladujeta predvsem tehnologiji React in Angular, sledita jima Vue.js. React, razvito in vzdrževano s strani Facebooka, in Angular, orodje, ki ga je razvila ekipa Google. Sta izrazito različni ogrodji s svojimi prednostmi in slabostmi. Namen te raziskave je analizirati njune značilnosti, zmogljivosti ter enostavnost uporabe v kontekstu obdelave velikih podatkov.

1.1 Opis področja in opredelitev problema

Področje spletnega razvoja zadnja leta bliskovito napreduje. Pri tem pa pri vsakem projektu razvijalce postavlja pred vprašanje, katero tehnologijo uporabiti za razvoj svojih aplikacij. Najbolj popularni sta ReactJS in Angular, vsaka s svojimi prednostmi in slabostmi. V praktičnem delu se bomo lotili izdelave dveh enakih aplikacij, ki bosta predelali veliko količino podatkov naenkrat in jih izpisali v obliki tabele. Aplikacija se poveže na javni api https://pokeapi.co/api/v2/pokemon/ in prikaže vse podatke naenkrat. Pri tem se lahko pojavi problem – ker je teh podatkov res veliko, lahko nalaganje traja dlje časa ali pa pride do napake pri obdelavi take količine podatkov.

1.2 Namen, cilji in osnovne trditve

Konkreten cilj je primerjati in ugotoviti, katera tehnologija je bolj optimalna oziroma učinkovitejša pri obdelavi večje količine podatkov. Pogledali bomo čase merjenj in načine,

kako poteka implementacija pri pridobitvi take količine podatkov. V diplomskem delu bomo natančneje analizirali obe tehnologiji, njune pristope, prednosti in slabosti ter na podlagi tega dati novim razvijalcem novo perspektivo pri učenju tehnologije za spletno razvijanje.

V diplomski nalogi smo si zastavili naslednje hipoteze:

- H1: Pri obdelavi velike datoteke se bolje obnese NOSQL podatkovna baza kot relacijska podatkovna baza.
- H2: Knjižica React.js nudi več različnih gradiv za začetnike in je lažji za začetek kot Angular.
- H3: Pri prikazu manjše količine podatkov se bolje obnese ReactJS kot Angular.
- H4: Pri prikazu velike količine podatkov se bolje obnese ReactJS kot Angular.

1.3 Predpostavke in omejitve

Glavne omejitve v raziskovalnem delu bi lahko nastopile pri zahtevnosti tehnologije za začetnika, saj bi lahko bilo morda učenje teh orodij za nas lažje kot za koga drugega. Ob tem bi lahko prišlo do težav pri implementaciji tega projekta, pri katerem moramo v aplikacijo implementirati veliko količino podatkov, tukaj lahko pride do zrušenja aplikacije, kar pomeni, da bomo morali aplikacijo večkrat sprogramirati, da bo delovala pravilno v obeh tehnologijah.

1.4 Uporabljene raziskovalne metode

V svoji diplomski nalogi bomo uporabili naslednje raziskovalne metode:

- uporaba osebnih zapiskov,
- primerjalna metoda,
- pregled literature,
- meritve,
- analiza podatkov.

Podrobneje so uporabljene raziskovalne metode opisane v poglavju metodologija (10).

2 ZGODOVINA SPLETNIH STRANI

V začetku 90. let so se na internetu začele pojavljati prve spletne strani. Razvoj le teh je bil takrat močno povezan z razvojem svetovnega spleta oz. world wide web (WWW). Leta 1991 je Tim Berners-Lee na CERN-u (European Organization for Nuclear Research) razvil prvi spletni brskalnik, imenovan WorldWideWeb, in prvi spletni strežnik. Prva spletna stran je bila postavljena na ip-ju: http://info.cern.ch in je vsebovala osnovne informacije o WWW-projektu. To je bila prva javna dostopna spletna stran na svetovnem spletu.

2.1 HTML (HyperText Markup Language)

HTML je osnovni jezik, ki se uporablja za izdelavo in oblikovanje vsebine na spletnih straneh. HTML omogoča strukturiranje spletnih strani s pomočjo, tako imenovanih značk, kar pomeni, da označimo različne dele besedila, slike, povezave in druge elemente na spletni strani. (HTML Tutorial, 2024)

2.1.1 Značke

V HTML (Hypertext Markup Language) so »značke« (ang. tags) posebne oznake, ki se uporabljajo za označevanje različnih delov vsebine na spletni strani. Značke delujejo kot gradniki, ki opisujejo strukturo dokumenta in določajo, kako naj brskalnik prikaže vsebino uporabniku. Značke se običajno sestavljajo iz odpirajočega dela, imenovanega »začetna oznaka« (<tag>), in zapiralnega dela (</tag>). Med temi deli se nahaja vsebina, ki jo želimo označiti.

2.1.2 Strukturne značke

Strukturne HTML oznake igrajo ključno vlogo pri organizaciji in strukturiranju vsebine na spletnih straneh. Te oznake pomagajo ustvariti smiselno hierarhijo in poudarjajo različne dele dokumenta (HTML Tutorial, 2024).

Tukaj so nekatere pomembne strukturne HTML oznake:

- <html> – označuje začetek in konec HTML dokumenta. Vse ostale oznake so običajno znotraj te oznake.

- <head> – se uporablja za vključevanje meta informacij, povezav na zunanje vire (npr. stilski listi, skripte), naslov strani in druge pomembne informacije,
- <body> – obdaja dejansko vsebino strani, vključno z besedilom, slikami, povezavami itd.

2.1.3 Osnovni elementi

Osnovni elementi HTML so gradniki, ki tvorijo strukturo dokumenta. Tukaj so nekatere osnovne HTML oznake:

- **<h1>** do **<h6>** – Oznake za naslove različnih ravni.
- **<p>** – Oznaka za odstavek.
- <a> – Oznaka za povezavo (hiperlink).
- <img> – Oznaka za vstavljanje slik.
- – Oznaka za neurejene sezname.
- – Oznaka za urejene sezname.
- – Oznaka za posamezen element v seznamu.

2.1.4 Atributi

HTML atributi so dodatne informacije, ki se uporabljajo v oznakah (tags) in pomagajo določiti značilnosti ali obnašanje določenega elementa na spletni strani. Tukaj je nekaj osnovnih HTML atributov:

- Id – Atribut id se uporablja za določitev unikatne identifikacije za posamezen element na strani. To je uporabno, ko želite ciljati določen element s CSS ali JavaScriptom.
- Class – Atribut class se uporablja za dodelitev ene ali več razredov elementu. Razredi se pogosto uporabljajo za ciljanje skupin elementov s CSS in določanje enakih stilskih lastnosti.
- Style – Atribut style se uporablja za neposredno določanje stila posameznega elementa. Vsebina tega atributa je lahko enakovredna CSS pravilom.

- Href – Atribut href se uporablja v oznaki <a> za določanje ciljne povezave (URL) ali poti do dokumenta, do katerega naj povezava vodi.

2.1.5 Struktura

Osnovna struktura HTML dokumenta vključuje več ključnih elementov, ki so bistveni za vsako spletno stran. Začetek dokumenta je označen z oznako **<!DOCTYPE html>**, ki opredeljuje različico HTML, ki se uporablja. Nato sledi element **<html>**, ki omejuje celoten dokument, vključno s svojo glavo (**<head>**) in vsebino (**<body>**). V glavi dokumenta se nahajajo meta informacije, naslov strani (**<title>**), povezave do zunanjih virov, kot so stilski listi in skripte. Vsebina strani, ki vključuje besedilo, slike, povezave in druge elemente, je znotraj elementa **<body>**. Poleg tega se pogosto uporabljajo strukturne oznake, kot so **<header>**, **<footer>**, **<nav>**, **<main>**, **<section>**, in **<article>**, ki pomagajo organizirati in poudariti različne dele spletne strani. S pomočjo teh osnovnih strukturnih elementov lahko razvijalci ustvarijo smiselno in dostopno hierarhijo vsebine, ki omogoča enostavno razumevanje strukture spletnega mesta brskalnikom, orodjem za iskanje in uporabnikom (HTML Tutorial, 2024).

```
<!DOCTYPE html>
<html>
<head>
    <title>Naslov Strani</title>
</head>
<body>
    <!-- VSEBINA -->
</body>
</html>
```

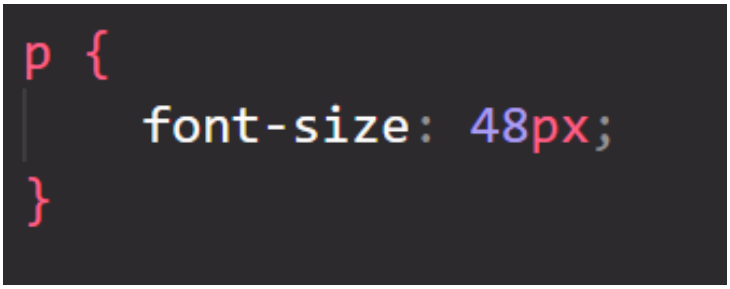
Slika 1: Primer opisa osnovne strukture HTML dokumenta

(Vir: Lasten vir)

2.2 *CSS (Cascading Style Sheeets)*

CSS oziroma Cascading Style Sheets je jezik, ki se uporablja za opisovanje strukture in oblikovanja naših HTML dokumentov. Njegov namen je ločevanje HTML vsebine od njegovega videza in oblikovanja. Imamo različne značke, ki jim moramo opisati oblikovanje. Oblikujemo HTML značke (div, p, h1 ipd.), razrede in id-je posameznih html elementov. Ta dokument se lahko vključi v glavi HTML dokumenta (CSS Tutorial, 2024).

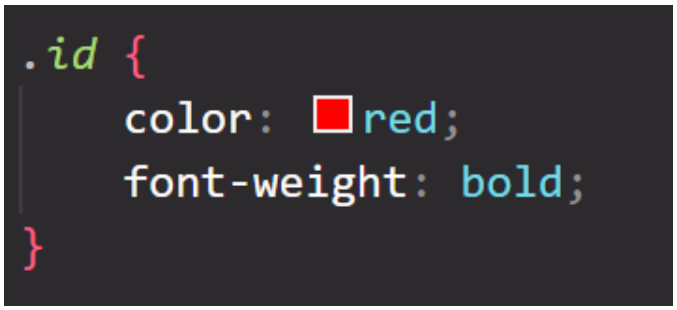
Na naslednjih slikah je prikazan potreben pristop za opisovanje oblikovanja značk v css dokumentu. Pri osnovnih HTML značkah ni potrebno dodajati nobenega posebnega znaka, saj brskalnik sam zazna, da gre za HTML značko.



Slika 2: Primer opsa HTML značke

(Vir: Lasten vir)

Drugi po vrsti je na vrsti id, ki se uporablja, kadar želimo posamezen element oblikovati drugače kot druge. Ti se označijo z znakom ., kar pomeni, da brskalnik zazna vsak posamezen id in ga temu primerno oblikuje.



Slika 3: Primer opisa ID značke

(Vir: Lasten vir)

Nazadnje lahko opisujemo razrede. Tukaj po navadi označimo več elementov, za katere želimo imeti enako oblikovanje. Razrede označimo z #, s tem pa brskalnik zazna vse elemente, ki imajo v opisu html značke označen razred s tem imenom.

Slika 4: Opisa razreda v css

(Vir: Lasten vir)

2.3 JavaScript

JavaScript je programski jezik, ki se uporablja za interaktivnost na spletnih straneh. Razvijalci ga uporabljajo za izboljšanje uporabniških izkušenj, manipulacijo z vsebinami na strani in izvajanje asinhronih operacij.

2.4 JavaScript ogrodja

JavaScript ogrodja so orodja, ki razvijalcem omogočajo lažje in učinkovitejše ustvarjanje spletnih aplikacij. V dinamičnem svetu spletnega razvoja so se pojavila številna ogrodja, ki rešujejo različne izzive pri razvoju kompleksnih in odzivnih uporabniških vmesnikov.

3 PODATKOVNE BAZE

V današnjem informacijskem okolju podatkovne baze igrajo ključno vlogo pri shranjevanju, upravljanju in pridobivanju podatkov. Podatkovna baza (angl. »database«) je organizirana zbirka podatkov, ki jih je mogoče enostavno shranjevati, urejati, posodabljati in po potrebi pridobivati. Glavni cilj podatkovne baze je zagotoviti učinkovit način za shranjevanje in upravljanje velikih količin podatkov ter omogočiti hiter dostop do teh podatkov. Običajno podatkovne baze uporabljajo sistem med seboj povezanih tabel na način, ki omogoča učinkovito iskanje in pridobivanje informacij. Podatkovne baze lahko razdelimo v različne vrste relacijskih (SQL) in nerelacijskih (NoSQL) podatkovnih baz, odvisno od načina, kako shranjujejo podatke.

3.1 Relacijske Podatkovne Baze (SQL)

Relacijske podatkovne baze temeljijo na konceptu relacij, tabel in SQL (Structured Query Language). Te baze podatkov sledijo strogi strukturi, kjer so podatki organizirani v tabelah z določenimi relacijami med njimi. Glavne značilnosti relacijskih baz vključujejo doslednost, integriteto podatkov, transakcije in možnost izvajanja kompleksnih poizvedb s pomočjo SQL. Primer relacijske podatkovne baze je na primer MySQL ali PostgreSQL (Carder, 2020), (Smallcombe, 2024).

Podatki znotraj teh baz so organizirani po tabelah, kjer vsaka tabela predstavlja določen tip podatkov. Vsaka vrstica v tabeli predstavlja posamezen zapis ali entiteto. Stolpci v teh tabelah pa predstavljajo različne atribute ali lastnosti entitet. Povezave med tabelami se vzpostavljajo preko t. i primarnih in tujih ključev. Primarni ključ identificira edinstven zapis v tabeli, medtem ko tuj ključ vzpo

stavlja neposredno povezavo z drugo tabelo.

Tabela 1: 10 najbolj priljubljenih relacijskih podatkovnih baz

Ime podatkovne baze	Prva verzija (Leto)	Zadnja verzija (Leto)
MySQL	1995	2023
PostgreSQL+	1996	2024
Microsoft SQL Server	1998	2022
SQLLite	2000	2024
Oracle Database	1979	2021
IBM Db2	1983	2023
MariaDB	2009	2024
Amazon Redshift	2012	2024
Amazon Aurora	2014	2021
IBM Informix	1985	2020

Pridobljeno iz: Analytics Insight: SQL Databases for Data Science: Top 10 Options, 2024

3.2 Nerelacijske Podatkovne Baze (NOSQL)

Nerelacijske podatkovne baze, znane tudi kot NoSQL baze, so se razvile kot alternativa relacijskim bazam, zlasti za potrebe hitrejšega in prilagodljivejšega shranjevanja podatkov. Te baze uporabljajo različne modele za shranjevanje podatkov, kot so dokumenti, ključ-vrednostni pari, stolpci ali grafi. NoSQL podatkovne baze so prilagodljivejše in skalabilnejše ter omogočajo shranjevanje nestrukturiranih podatkov. Primeri NoSQL baz vključujejo MongoDB, Cassandra in Redis (Introduction to NoSQL, 2023).

Tabela 2: 10 najbolj priljubljenih NoSQL podatkovnih baz

Ime podatkovne baze	Prva verzija (Leto)	Zadnja verzija (Leto)
MongoDB	2009	2023
Apache Cassandra	2008	2022
Redis	2009	2022
RavenDB	2010	2024
ScyllaDB	2016	2024
Azure Cosmos DB	2017	2024
Neo4j	2007	2024
CouchDB	2005	2023
Google Cloud Datastore	2013	2024
Apache CouchDB	2005	2023

Pridobljeno iz: Medium: 10 best NoSoSQL databases for 2023, 2023

3.2.1 Dokumentne podatkovne baze

Dokumentne podatkovne baze shranjujejo podatke v dokumentih, običajno v formatu JSON ali XML. Vsak dokument lahko vsebuje različne vrste podatkov in se lahko razširi brez predhodne določitve sheme, kar omogoča večjo prilagodljivost pri shranjevanju podatkov.

3.2.2 Stolpčne podatkovne baze

Stolpčne podatkovne baze shranjujejo podatke v stolpcih namesto vrsticah, kar omogoča učinkovitejše poizvedovanje in skaliranje za velike količine podatkov. Podatki so organizirani po stolpcih, kar omogoča hitro iskanje in dostop do posameznih atributov.

3.2.3 Ključ-vrednostne podatkovne baze

Ključ-vrednostne podatkovne baze shranjujejo podatke v obliki parov ključ-vrednost. Vsakemu ključu je dodeljena vrednost, ki je lahko karkoli, od preprostih nizov do kompleksnih objektov. Te podatkovne baze so izjemno hitre in primerne za shranjevanje predpomnilnika, sej in drugega podobnega hitro dostopnega podatka.

3.2.4 Grafične podatkovne baze

Grafične podatkovne baze so zasnovane za shranjevanje in poizvedovanje grafov. Podatki so predstavljeni kot vozlišča, povezani z robovi, kar omogoča učinkovito reprezentacijo povezav med entitetami. Te podatkovne baze so optimalne za modele podatkov, ki temeljijo na povezavah, kot so socialna omrežja, priporočilni sistemi in geografski podatki.

3.2.5 MongoDB

MongoDB je dokumentna, shematsko prilagodljiva, NoSQL podatkovna baza, ki temelji na konceptu dokumentov. Vsak dokument v MongoDB-ju je predstavljen v obliki JSON podatkovne strukture, kar omogoča enostavno shranjevanje in obdelavo podatkov. Baza podatkov se lahko razteza horizontalno in vertikalno, kar pomeni, da je primerna za obravnavo velikih obsegov podatkov. MongoDB omogoča tudi visoko razpoložljivost in odpornost na napake, saj podpira replikacijo in fragmentacijo podatkov. Poleg tega MongoDB ponuja bogat nabor funkcij za poizvedovanje, analizo in upravljanje podatkov, kar ga uvršča med priljubljene izbire za različne aplikacije, ki zahtevajo skalabilnost, fleksibilnost in zmogljivost (MongoDB, 2024).

3.2.6 Stroški z MongoDB

Stroški uporabe MongoDB-ja so lahko odvisni od več dejavnikov, vključno z načinom uporabe, različico produkta ter morebitnimi dodatnimi storitvami in funkcijami. MongoDB nam nudi 3 glavne verzije podatkovne baze: Dedicated, Serverless in Shared.

Prva verzija je Dedicated, ki je namenjena razvijanju aplikacij z večjimi zahtevani pri obremenitvi. Ta verzija nam nudi med 10 gigabajtov in 4 terabajti prostora za shranjevanje, med 2 in 768 gigabajti RAM-a, omrežno izolacijo in natančen nadzor dostopa ter je na voljo v več regijah in na večjih različnih oblakih. Ta paket nas stanem 57 evrov na mesec. Druga verzija je Serverless, ki je na voljo za 0.10 ameriških dolarjev za milijon branj. Ta verzija je namenjena za aplikacije brez strežnika z redkim prometom. Prav tako je potrebna minimalna konfiguracija. Tukaj nam pripada do 1 terabajt prostora za shranjevanje. Plačujemo samo za operacije, ki jih izvajamo. Na koncu še moramo vedeti, da nam omogoča stalno varnost in varnostne kopije. Zadnja verzija je Shared verzija, ki je brezplačna. Ta verzija je namenjena za učenje in rahziskovanje MongoDB podatkovne baze v oblaku. Ponuja nam zgolj osnovne konfiguracije. Prav tako nam nudi med 512 megabajti in 5 gigabajti prostora za pomnilnik in skupni deljen RAM. Tukaj ne potrebujejo niti bančne kartice. V kolikor bi želeli več funkcionalnosti, moramo verzijo nadgraditi na Dedicated (MongoDB, 2024).

4 BIG DATA

4.1 Definicija

Big data označuje velike in kompleksne podatkovne sete, ki jih tradicionalne podatkovne baze in orodja ne morejo učinkovito obdelati. Te podatkovne sete odlikujejo štirje ključni dejavniki, znani kot štiri "V": volumen, hitrost (velocity), raznolikost (variety) in verodostojnost (veracity). Volumen se nanaša na ogromne količine podatkov, hitrost na hitrosti, s katero se podatki generirajo in obdelujejo, raznolikost na različne tipe in vire podatkov, verodostojnost pa na kakovost in natančnost podatkov. Big data zahteva nove metode in tehnologije za shranjevanje, obdelavo in analizo podatkov, kar omogoča pridobivanje koristnih vpogledov in informacij (Marr, 2015).

4.2 Formati podatkov za Big Data

4.2.1 Strukturirani podatki

Strukturirani podatki so organizirani v urejene tabele ali baze podatkov, kot so SQL baze podatkov. Ti podatki imajo jasno definirano shemo, kar omogoča enostavno iskanje in obdelavo. Primeri strukturiranih podatkov vključujejo finančne podatke, transakcije, tabele strank in druge vrste podatkov, ki se običajno nahajajo v relacijskih bazah podatkov (Shacklett, 2023).

4.2.2 Polstrukturirani podatki

Polstrukturirani podatki imajo delno organizacijo, vendar ne ustrezajo standardnim strukturam baz podatkov. Ti podatki vsebujejo označevalce ali oznake, ki omogočajo identifikacijo različnih elementov podatkov. Primeri polstrukturiranih podatkov vključujejo XML, JSON in HTML datoteke, ki so pogosto uporabljene za prenos in shranjevanje informacij med sistemi (Shacklett, 2023).

4.2.3 Nestrukturirani podatki

Nestrukturirani podatki nimajo določene oblike ali strukture, zaradi česar jih je težje analizirati in organizirati. Ti podatki lahko vsebujejo veliko količino koristnih informacij, vendar zahtevajo posebne tehnike za obdelavo in analizo. Primeri nestrukturiranih podatkov vključujejo besedilne dokumente, e-pošto, video posnetke, avdio datoteke in slike, ki se pogosto uporabljajo v različnih aplikacijah (Shacklett, 2023).

4.2.4 Meta podatki

Meta podatki so podatki o podatkih, ki opisujejo vsebino, izvor, kontekst ali strukturo drugih podatkov. Meta podatki pomagajo pri organizaciji, iskanju in uporabi podatkov, saj zagotavljajo informacije, kot so avtorji dokumentov, datumi ustvarjanja in ključne besede. Primeri meta podatkov vključujejo informacije, ki spremljajo digitalne slike, dokumente in baze podatkov ter pomagajo pri učinkovitem upravljanju in iskanju podatkov (Marr, 2015).

4.3 Učinkovitost

Učinkovitost pri obdelavi big data je ključna za zagotavljanje hitrih in natančnih analiz ter vizualizacij podatkov. Pri uporabi React.js se poudarja modularnost in prilagodljivost komponent, kar omogoča razvijalcem, da hitro in učinkovito gradijo in prilagajajo uporabniške vmesnike (Wieruch, 2018). Po drugi strani pa Angular ponuja vgrajene funkcionalnosti, ki lahko pospešijo razvoj aplikacij, vendar zahteva večje začetno učenje zaradi svoje kompleksnosti (Marr, 2015).

4.4 Hitrost

Hitrost obdelave podatkov je ključna pri delu z big data, še posebej, ko gre za realnočasne analize in vizualizacije. React.js se ponaša s hitrim virtualnim DOM-om, ki omogoča učinkovito posodabljanje uporabniškega vmesnika, kar je ključnega pomena pri obdelavi velikih količin podatkov. Angular pa ponuja funkcionalnosti za reaktivno programiranje in uporabo asinhronih podatkovnih tokov z RxJS, kar omogoča hitro in odzivno delovanje aplikacij (Daniels & Atencio, 2017).

4.5 Uporaba React.js pri Big Data

4.5.1 Virtualni DOM

React uporablja virtualni DOM za učinkovito upravljanje sprememb v uporabniškem vmesniku. To je ključno pri obdelavi velikih količin podatkov, saj omogoča hitro in učinkovito posodabljanje samo tistih delov vmesnika, ki se spreminjajo (Wieruch, 2018).

4.5.2 Komponentna Arhitektura

Modularnost React komponent omogoča enostavno ponovno uporabo in kombiniranje različnih delov uporabniškega vmesnika, kar je koristno pri vizualizaciji kompleksnih podatkov (Wieruch, 2018).

4.5.3 State Management

Knjižnice, kot sta Redux in Context API, pomagajo pri upravljanju stanja aplikacije, kar je pomembno pri sinhronizaciji podatkov med različnimi deli aplikacije.

4.6 Uporaba Angularja pri Big Data

Angular je celovito ogrodje za razvoj spletnih aplikacij, ki ga vzdržuje Google. Je bolj mnenjsko usmerjen kot React in ponuja številne vgrajene funkcionalnosti, ki poenostavljajo obdelavo velikih podatkovnih nizov.

4.6.1 Dvostransko Vezanje Podatkov (Two-Way Data Binding)

Angular omogoča avtomatsko sinhronizacijo med modelom in pogledom, kar olajša delo z dinamičnimi podatki in interaktivnimi komponentami (Angular, 2024).

4.6.2 RxJS in reaktivno programiranje:

Angular vključuje RxJS, knjižnico za reaktivno programiranje, ki omogoča delo z asinhronimi podatkovnimi tokovi. To je še posebej uporabno pri obdelavi podatkov v realnem času (Angular, 2024).

4.6.3 Modularnost in Dependency Injection:

Angularjeva modularna arhitektura in sistem za vbrizgavanje odvisnosti (dependency injection) olajšata upravljanje kompleksnih aplikacij in ponovno uporabo kode (Angular, 2024).

4.7 Primerjava React.js in Angular v Kontekstu Big Data

4.7.1 Učinkovitost

React.js zaradi svojega virtualnega DOM-a ponuja boljšo učinkovitost pri obdelavi velikih količin podatkov v primerjavi z Angularjevim dvostranskim vezanjem podatkov, ki je lahko manj učinkovito pri zelo velikih podatkovnih setih (React, 2024).

4.7.2 Učinkovitost razvoja

Angular, kot celovito ogrodje, nudi številne vgrajene funkcionalnosti, ki lahko pospešijo razvoj aplikacij, vendar pa zaradi svoje kompleksnosti zahteva večje začetno učenje. React.js je bolj prilagodljiv in lažje vstopen za razvijalce, vendar zahteva dodatne knjižnice za nekatere funkcionalnosti

4.7.3 Skalabilnost

Obe tehnologiji ponujata dobre možnosti za skalabilnost. Vendar pa Angularjeva struktura in orodja za upravljanje stanja ponujajo bolj celovito rešitev za zelo velike aplikacije, medtem ko je React bolj modularen in prilagodljiv, kar olajša razširjanje aplikacij po potrebi.

4.8 SQL in NoSQL za Big Data

Za obdelavo in shranjevanje velikih količin podatkov (Big Data) se NoSQL baze podatkov pogosto izkažejo za boljše od SQL baz podatkov. NoSQL baze, kot sta MongoDB in Cassandra, so zasnovane za horizontalno skaliranje, kar omogoča enostavno širjenje kapacitet s povečevanjem števila strežnikov. To je ključno pri obdelavi ogromnih količin podatkov, kjer sta potrebni visoka zmogljivost in hitrosti branja/pisanja. Poleg tega NoSQL baze omogočajo shranjevanje nestrukturiranih in polstrukturiranih podatkov, kar je pogosto značilno za Big Data aplikacije, kjer podatki prihajajo iz različnih virov in v različnih formatih. V primerjavi s tem so SQL baze bolj omejene zaradi svoje toge sheme in vertikalnega skaliranja, kar lahko postane ovira pri hitrem naraščanju obsega podatkov. Zaradi teh razlogov so NoSQL baze pogosto prva izbira za podjetja in organizacije, ki se ukvarjajo z velikimi podatkovnimi nizi in potrebujejo visoko skalabilnost ter prilagodljivost (Smallcombe, 2024).

5 VIRTUAL DOM

5.1 DOM

DOM (Document Object Model) je struktura dokumenta v spletni strani, ki predstavlja hierarhijo elementov, atributov in lastnosti v HTML, XML ali XHTML formatu. Ta hierarhija se dinamično spreminja glede na uporabniške interakcije ali spremembe v vsebini strani. Vsaka sprememba v normalnem DOM-u, na primer dodajanje, odstranjevanje ali spreminjanje elementov, povzroči ponovno izgradnjo in ponovno upodabljanje celotnega dokumenta, kar lahko vpliva na učinkovitost in odzivnost spletnih aplikacij (Introduction to the DOM, 2023).

5.1.1 Slabosti DOM-a

Čeprav je normalni DOM ključen za upodabljanje spletnih strani, ima tudi nekaj negativnih lastnosti, ki lahko vplivajo na učinkovitost in odzivnost spletnih aplikacij. Ena od glavnih težav je njegova izvedba, ki je lahko počasna pri kompleksnih aplikacijah z velikim številom elementov. Vsaka sprememba v normalnem DOM-u zahteva ponovno izgradnjo in ponovno upodabljanje celotnega drevesa elementov, kar lahko privede do zamud pri odzivnosti uporabniškega vmesnika (Tuama, 2024).

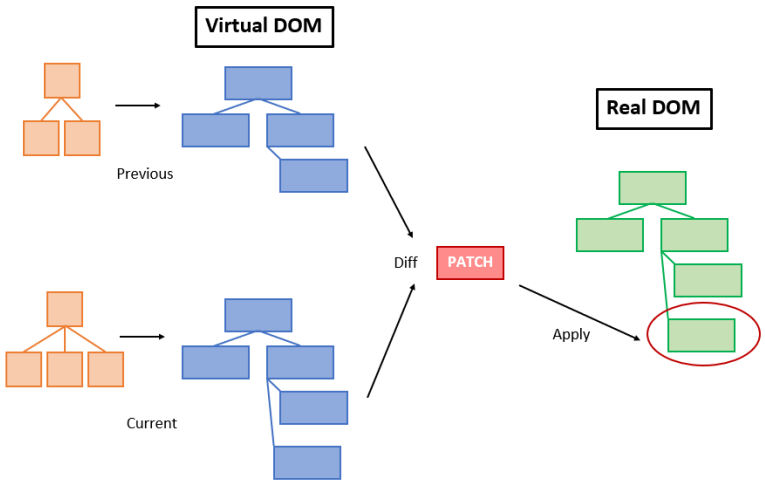
5.2 Virtual DOM

Virtualni DOM (VDOM) je ključna konceptualna sestavina sodobnih spletnih tehnologij, ki igra pomembno vlogo pri izboljšanju učinkovitosti in odzivnosti spletnih aplikacij. Namesto neposrednega manipuliranja z normalnim DOM-om, spletna ogrodja, kot je React.js, uporabljajo virtualni DOM kot abstraktno predstavitev dejanskega DOM-a v pomnilniku (Patel, 2023).

5.2.1 Delovanje virtualnega DOM-a

Ko pride do posodobitve uporabniškega vmesnika, se najprej ustvari virtualna kopija obstoječega DOM-a. Ta virtualni DOM nato primerjamo s prejšnjim stanjem virtualnega DOM-a, da ugotovimo, kateri deli se dejansko spreminjajo. Namesto, da bi neposredno posredovali spremembe na dejanskem DOM-u, se izvedejo le spremembe, ki so bile ugotovljene med

primerjavo virtualnega DOM-a. To omogoča optimizacijo posodabljanja, s tem se lahko izognemo nepotrebnemu ponovnemu upodabljanju celotnega DOM-a.



Slika 5:Primer delovanje virtualnega DOM-a

(Vir: https://blog.bitsrc.io/incremental-vs-virtual-dom-eb7157e43dca)

5.2.2 Prednosti virtualnega dom-a

- Hitrejše posodobitve: Ker se spremembe najprej izvajajo na virtualnem DOM-u, lahko aplikacije izvajajo hitrejše posodobitve uporabniškega vmesnika, tako se izognejo nepotrebnemu preurejanju dejanskega DOM-a.
- Učinkovita uporaba virov: Virtualni DOM omogoča optimizacijo uporabe virov, kar omogoča skupno obdelavo več sprememb naenkrat.
- Boljša uporabniška izkušnja: Zaradi hitrejših in učinkovitejših posodobitev se izboljša tudi uporabniška izkušnja, saj se zmanjša zamik pri odzivnosti spletnih aplikacij.

6 REACT.JS

6.1 Zgodovina knjižice

Razvijalci Facebook-a so maja leta 2013 predstavili prvo različico brezplačne odprtokodne knjižice JavaScript-a za izgradnjo front-end aplikacij po imenu React. React je uvedel nov koncept JSX, ki je razširitev JavaScripta, ter omogoča pisanje XML/HTML sintakse znotraj JavaScripta. Knjižica je ena najbolj priljubljenih orodij za gradnjo aplikacij, zlasti enostranskih (Single Paged Applications – SPA). S svojo preprostostjo omogoča razvijalcem hitro učenje in vzame veliko manj dela za izdelavo aplikacij. Za uporabo knjižice mora razvijalec poznati samo osnove HTML ter JavaScripta (Taylor, 2023).0

6.2 Komponentna arhitektura

React temelji na komponentni arhitekturi, ki aplikacijo razdeli na manjše, samostojne gradnike – komponente. Vsaka komponenta ima vnaprej določene funkcionalnosti in lahko deluje neodvisno ali v sodelovanju z drugimi komponentami. S tem imajo razvijalci boljšo možnost, da bolje razumejo, vzdržujejo in nadgrajujejo svojo kodo. Ker je koda razbita na manjše dele, se lahko razvijalci bolje znajdejo v lastni kodi, saj pri večjih projektih lahko pride do podvojevanja kode ter s tem do napak znotraj projekta (Onyekani, 2023).

React uporablja komponentno arhitekturo za izdelavo uporabniških vmesnikov s pomočjo komponent. Znotraj knjižice je komponenta osnovni element, ki združuje logiko z predstavitveno plastjo (HTML), včasih pa tudi slog (CSS).

6.3 React Native

Po tem, ko je Facebook uspešno predstavil React.js knjižnico, so nadaljevali z njegovim razvojem in 2 leti kasneje (2015) predstavili React Native. React Native je tako kot React odprtokodno ogrodje, ki je v glavnem namenjeno razvoju mobilnih aplikacij. Razvijalcem nudi razvijanje mobilnih aplikacij z uporabo JavaScript in tehnologije React.

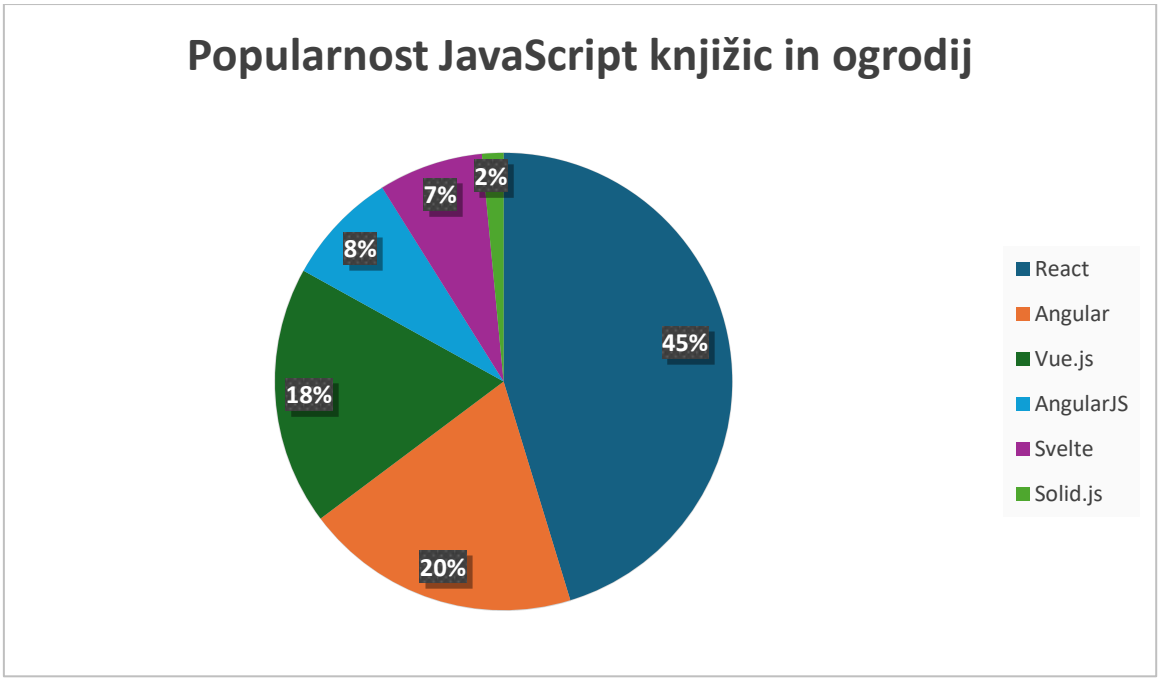
Ena izmed ključnih prednosti, ki jo nudi React Native, je, da lahko razvijalcem omogoča delitev kode za različne platforme, kot so iOS in Android. Razvijalci lahko uporabijo enako kodo, tako eno kot drugo platformo, kar znatno skrajša čas razvoja, kar pomeni, da ni potrebe po vzdrževanju dveh ločenih kodnih baz (Budziński, 2024).

6.4 *Virtualni DOM*

Knjižica React.js je znana po svojem inovativnem pristopu k upravljanju z uporabniškim vmesnikom s pomočjo virtualnega DOM-a. Pri Reactu se vsaka sprememba najprej izvede na virtualnem DOM-u, nato pa se virtualni DOM primerja s prejšnjim stanjem, da se ugotovijo spremembe, ki so se zgodile, in se nato te spremembe učinkovito izvedejo na dejanskem DOM-u. Ta pristop omogoča hitro in učinkovito posodabljanje uporabniškega vmesnika, ne da bi bilo treba neposredno manipulirati z dejanskim DOM-om (Patel, 2023).

6.5 *Zahtevnost*

Če pogledamo trend popularnosti, lahko vidimo, da zadnja leta prevladuje React, tik za njim sta Angular in Vue. Eno izmed glavnih vprašanj je, ali se je težko priučiti te tehnologije? Odgovor na to vprašanje je odvisen od učenca, a v večini primerov novi razvijalci nimajo velikih težav z učenjem te tehnologije. Moramo vedeti, da je React knjižica in ne polno ogrodje JavaScripta, kar pomeni, da nima vsega, kar imata Angular in Vue.js. React se osredotoča na eno stvar – gradnjo uporabniških vmesnikov. Nabor funkcij v Reactu je sicer manjši kot pri Angularju in Vue-u, kar pomeni, da se popolnemu začetniku ni treba priučiti ogromno funkcij in je zato manj zahteven za začetnike.



Graf 1: Popularnost JavaScript knjižnic v letu 2023

Vir: Github: Frontend framework popularity, 2023

Zaradi svoje popularnosti lahko na spletnih portalih, kot so Udemy, LinkedIn, YouTube najdemo ogromno gradiva, tako za začetnike kot tudi za napredne razvijalce. Če pogledamo po številkah ugotovimo, da ima portal Udemy 188 krožkov, na YouTubu imamo okoli 21 različnih videoposnetkov in 679 krožkov na Coursera.

React ima na spletu tudi ogromno dokumentacije, ki nazorno in korak za korakom razlaga ter prikazuje uporabo knjižice. Sam sem se učil iz spletnega dokumenta mdn web docs, ki nas postopno uči uporabe te tehnologije, prav tako pa so mi pomagali tudi videoposnetki s platforme YouTube. Če bi moral sam oceniti učenje Reacta, bi ga ocenil kot lahkega, saj s predznanjem HTML in JavaScripta lahko začnemo, po le nekaj lažjih projektih, samostojno graditi React aplikacije.

6.6 Prednosti

- Komponentna arhitektura: Razvijalci imajo možnost razvijanja lahkih, hitrih komponent katere se lahko ponovno uporabijo na drugih delih kode (Technostacks, 2024).
- Veliko gradiva in virov za učenje: Ker je trenutno še vedno najbolj priljubljena tehnologija, ima na spletu ogromno materiala za učenje. Od »step-by-step« dokumentov do velike količine videov na platformi YouTube, s katero se lahko tudi popolni začetniki hitro in korak za korakom naučijo uporabljati knjižico (React.JS: Advantages and Disadvantages, 2023).
- Virtual DOM: React uporablja virtualni DOM (Document Object Model), kar pripomore k učinkovitemu upravljanju z manipulacijami DOM-a. Spremembe se najprej izvajajo v virtualnem DOM-u, nato pa se ugotovi, katere dele dejanskega DOM-a je potrebno posodobiti, kar pripomore k boljši učinkovitosti (React.JS: Advantages and Disadvantages, 2023).
- Podpora aplikacijam: React se pogosto uporablja v kombinaciji z drugimi orodji, kot sta npm in Webpack, kar omogoča sestavljanje kompleksnih aplikacij. To vključuje uporabo zunanjih knjižnic ter optimizacijo in združevanje datotek (React.JS: Advantages and Disadvantages, 2023).

- Aktivna skupnost: React je podprt s strani velike in aktivne skupnosti razvijalcev, kar pomeni, da se redno posodablja, uvajajo se nove funkcionalnosti in odpravljajo morebitne težave (Technostacks, 2024).

6.7 Slabosti

- SEO omejitve: V preteklosti je bilo kar nekaj težav z indeksiranjem v iskalnikih zaradi asinhronega nalaganja vsebine, ki ga pogosto uporabljajo aplikacije React (React.JS: Advantages and Disadvantages, 2023).
- Velikost Paketa: Čeprav je veliko izboljšav v smislu velikosti paketov, so nekatere aplikacije, zlasti manjše, lahko občutljive na velikost končnega paketa, ki ga generira React (Technostacks, 2024).
- Prevelika kompleksnost za manjše projekte: Za manjše projekte, kjer ni potrebno upravljanje stanja na kompleksen način, lahko uporaba Reacta privede do nepotrebne dodatne kompleksnost (Technostacks, 2024).
- Primernejši za front-end razvoj: Čeprav se lahko uporablja tudi za razvoj celotnih aplikacij, se React v glavnem uporablja za front-end razvoj. Za kompleksnejše aplikacije, ki vključujejo tudi back-end logiko, lahko postane bolj primerno drugo orodje ali okvir (Shagufta, 2023).

7 ANGULAR

Angular je odprtokodno ogrodje (framework) za izdelavo spletnih aplikacij. Razvijajo in vzdržujejo ga pri podjetju Google. Angular je zasnovan za lajšanje razvoja dinamičnih, enostranskih (Single Page Applications - SPA) spletnih aplikacij. S pomočjo Angularja lahko razvijalci ustvarijo kompleksne spletne aplikacije z visoko interaktivnostjo in dinamičnostjo. Angular temelji na jeziku TypeScript, ki je nadgradnja JavaScripta z dodanimi značilnostmi in statičnim tipiziranjem, kar pripomore k večji robustnosti in preglednosti kode. Poleg tega Angular vključuje tudi številne vnaprej pripravljene komponente, ki olajšajo izdelavo uporabniških vmesnikov (Angular, 2024).

7.1 Zgodovina AngularJS

Leta 2010 je bila ustvarjena prva različica, imenovana AngularJS. Razvijalci iz Googla so ustvarili to ogrodje, ki je bilo revolucionarno, zaradi koncepta dvosmernega podatkovnega vezja, ki je omogočilo avtomatsko posodabljanje uporabniškega vmesnika ob spremembah podatkov. Kljub uspehu AngularJS-a so se pojavile nekatere omejitve. Zato se je leta 2016 pojavil Angular 2, ki je bil popolnoma prenovljen. Spremenila se je arhitektura, uveden je bil TypeScript, in komponente so postale ključen gradnik ogrodja. Ker se Angular nenehno razvija, so izšle nadaljnje različice, kot so Angular 4, 5, 6 in tako naprej. Posodobitve so prinesle izboljšave zmogljivosti, nove funkcionalnosti in odpravile morebitne pomanjkljivosti (Gavigan, 2018).

Angular je danes eno najbolj priljubljenih ogrodij za razvoj spletnih aplikacij. Ponaša se s široko skupnostjo razvijalcev, obsežno dokumentacijo, in se uporablja za različne vrste projektov, vključno z enostranskimi aplikacijami (SPA), podjetniškimi rešitvami in drugimi.

7.2 Arhitektura

Angular sledi komponentni arhitekturi, ki temelji na gradnji aplikacije iz ločenih in samostojnih komponent. Ta arhitektura ponuja organizirano strukturo in izboljšuje vzdržljivost, ponovno uporabo ter testiranje kode. Ključni gradniki arhitekture so komponente, moduli, predloge (templates), dvosmerno podatkovno vezje, direktive, vbrizganje odvisnosti (dependancy injection), usmerjanje (routing), storitve, http komunikacija in ngModuli (Angular, 2024).

7.2.1 Komponente

Angular sledi komponentni arhitekturi, ki temelji na gradnji aplikacije iz ločenih in samostojnih komponent. Ta arhitektura ponuja organizirano strukturo in izboljšuje vzdržljivost, ponovno uporabo ter testiranje kode (Emadamerho-Atori, 2023).

7.2.2 Moduli

Moduli v Angularju so logične enote, ki združujejo povezane komponente, storitve in druge sestavne dele. Omogočajo boljšo organizacijo kode, poenostavljajo vzdrževanje ter omogočajo enostavno dodajanje ali odstranjevanje funkcionalnosti (Angular, 2024).

7.2.3 Templates (Predloge)

Predloge v Angularju so HTML datoteke z oznakami, ki omogočajo dinamično generiranje uporabniškega vmesnika. To poenostavlja manipulacijo z uporabniškim vmesnikom na podlagi sprememb v podatkovnem modelu (Angular, 2024).

7.2.4 Dvosmerno podatkovno vezje (Two – Way Data Binding)

Funkcionalnost dvosmernega podatkovnega vezja omogoča samodejno posodabljanje uporabniškega vmesnika ob spremembi podatkovnega modela in obratno. To izboljšuje odzivnost aplikacije in olajšuje delo z dinamičnimi podatki (AngularJS Data Binding, 2024).

7.2.5 Direktive

Angular vgrajuje direktive, kot so *ngFor in *ngIf, te omogočajo dinamično manipulacijo DOM-a. To olajša upravljanje s strukturo predloge in pogojnim prikazovanjem elementov (Angular, 2024).

7.2.6 Dependancy Injection

Vbrizgavanje odvisnosti v Angularju omogoča, da komponente dobijo svoje odvisnosti (npr. storitve) od zunanjih virov. To izboljšuje preizkušanje, ponovno uporabo in vzdrževanje kode (Angular, 2024).

7.2.7 Routing

Angular Router omogoča upravljanje navigacije med različnimi deli aplikacije. Zmožnost usmerjanja je ključna za gradnjo enostranskih aplikacij (SPA) in za ustvarjanje večstranskih izkušenj (Angular, 2024).

7.2.8 HTPP Komunikacija

Angularjev modul **HttpClient** poenostavlja komunikacijo z zunanjimi strežniki preko protokola HTTP. To omogoča prenos podatkov med odjemalcem in strežnikom ter integracijo z različnimi storitvami.

7.2.9 NgModules

NgModules so gradniki, ki organizirajo in strukturirajo aplikacijo. Root modul je osnovni modul vsake Angular aplikacije, ki ga lahko dopolnjujejo dodatni moduli za boljšo organizacijo in vzdrževanje kode (Angular, 2024).

7.3 Angular in DOM (Document Object Model)

Pristop, ki ga uporablja Angular za upravljanje s DOM-om, se imenuje »Change Detection«. Gre za tehniko, ki omogoča spremljanje sprememb v modelu podatkov in nato posodabljanje DOM-a v skladu s temi spremembami. Angular v svojem jedru vključuje mehanizem za samodejno zaznavanje sprememb, ki se imenuje Zone.js. Omogoča spremljanje sprememb v kontekstu aplikacije. Ko pride do spremembe v modelu podatkov v Angularju, se sproži mehanizem Change Detection, ki primerja novo stanje modela podatkov s prejšnjim stanjem. To se pogosto izvaja z uporabo t. i. »dirty checking« tehnike, kjer se preveri, ali so se vrednosti

lastnosti modela podatkov spremenile. Če se zazna sprememba, Angular posodobi ustrezne dele DOM-a, da se ujema z novim stanjem modela podatkov. Prednost pristopa Change Detection v Angularju je ta, da ni potrebe po izgradnji virtualnega DOM-a in primerjavi z dejanskim DOM-om, kot je to v primeru React.js. To lahko v nekaterih primerih izboljša učinkovitost in zmanjša kompleksnost aplikacije. Poleg tega Change Detection v Angularju omogoča tudi bolj predvidljivo obnašanje, saj Angular prevzame več nadzora nad posodabljanjem DOM-a. Vendar pa lahko ta pristop v nekaterih primerih privede do slabše učinkovitosti, saj se vsakič, ko pride do spremembe v modelu podatkov, izvede celoten postopek Change Detection, kar lahko postane neučinkovito pri aplikacijah z velikim številom elementov in kompleksno logiko. Poleg tega je uporaba Change Detection-a v Angularju specifična za okolje Angularja in ni tako prenosljiva kot pristop z virtualnim DOM-om, ki ga uporablja React.js (Tiwari, 2023).

7.3.1 Renderer2 vmesnik

Renderer2 vmesnik v Angularju zagotavlja neodvisen način manipulacije DOM-a. S pomočjo tega vmesnika se doseže večja prenosljivost kode in enostavnejše prehajanje med različnimi platformami (Angular, 2024).

7.4 Zahtevnost

Za razliko od Reacta je Angular celovito ogrodje JavaScripta, kar posledično pomeni, da je za začetek potrebno vedeti več kot pri prej omenjeni tehnologiji. Razliko predstavlja tudi TypeScript, ki se uporablja v vseh Angularjevih aplikacijah. Vanilla JavaScript pa se uporablja za React aplikacije. TypeScript predstavlja superskupino JavaScripta, pri katerih je treba definirati tip neznank, in omogoča dodatne funkcionalnosti od navadnega JavaScripta.

7.5 Prednosti

- Celovita rešitev: Angular je celovito ogrodje za razvoj spletnih aplikacij. Vključuje vse, kar potrebujemo za razvoj, od upravljanja stanja, usmerjanja, HTTP zahtev, do testiranja. To pomeni, da ni treba uporabljati različnih orodij in knjižnic, saj je vse že vključeno v okvir (altexsoft, 2023).

- Tipiziran jezik (TypeScript): Angular je zgrajen na TypeScriptu, ki prinaša tipiziran pristop k razvoju. To pomaga pri prepoznavanju napak med razvojem in zagotavlja boljšo razumljivost kode (Agilway, 2023).
- Obsežna podpora za testiranje: Angular ponuja obsežno podporo za testiranje z orodji, kot sta Jasmine za enote testiranje in Protractor za funkcionalno testiranje. To pomaga pri zagotavljanju kakovosti kode in preprečevanju napak (Agilway, 2023).
- Modularna struktura: Angular spodbuja uporabo modularne strukture, kar pomeni, da lahko aplikacijo razdelite na manjše, samostojne module. To olajša razumevanje, testiranje in vzdrževanje kode (altexsoft, 2023).

7.6 Slabosti

- Poraba pomnilnika: Angular aplikacije lahko zahtevajo večjo porabo pomnilnika v primerjavi z drugimi lažjimi ogrodji, kar lahko vpliva na delovanje v okoljih s strožjimi omejitvami glede porabe pomnilnika.
- Velikost: Angular aplikacije imajo lahko večje obsege datotek v primerjavi z drugimi lažjimi okvirji, kot je React. To lahko vodi do daljših časov nalaganja in poveča zahtevnost vzdrževanja za manjše projekte.
- Težje integracija z drugimi ogrodji: V primerjavi z nekaterimi drugimi ogrodji, kot je React, lahko integracija Angularja z drugimi knjižnicami ali ogrodji zahteva dodatno delo. To je lahko ovira pri združevanju z obstoječimi sistemskimi rešitvami.

8 RAZVOJ SPLETNIH APLIKACIJ

Aplikacija Pokedex je namenjena prikazu informacij o 1000 različnih Pokemonih, pridobljenih iz zbirke podatkov s strežnika PokeAPI. Aplikacija bo implementirana tako v Reactu kot v Angularju z namenom primerjave teh dveh priljubljenih tehnologij za razvoj spletnih aplikacij.

8.1 React aplikacija

Ker gradimo našo prvo aplikacijo, moramo najprej naložiti program Node.js in npm, s pomočjo katerih bomo dostopali do metod za inicializacijo React aplikacije. Kot naslednje moramo izbrati razvojno okolje, v katerem bomo gradili naše aplikacije. Mi smo izbrali Visual Studio Code, s pomočjo katerega smo že prej gradili svoje aplikacije in delali na projektih za fakulteto. Menimo, da je Visual Studio Code najlažji za začetnike in nudi ogromno razširitev, s katerimi poenostavi marsikatero operacijo.

Ko je namestitev vseh programov opravljena, se pomaknemo v terminal, v katerem moramo vpisati naslednji ukaz:

```
npx create-react-app pokedex-react
```

Slika 6: Kreacija React projekta

(Vir: Lasten vir)

Po vpisanem ukazu se je ustvaril direktorij pokedex-react, kjer lahko vidimo vse potrebne komponente in module, ki jih potrebujemo za gradnjo React aplikacije. V terminal vpišemo: Cd pokedex-react, z njo se pomaknemo v novonastali direktorij. Nato lahko vidimo, da se nam v razvojnem okolju izpišejo vsi direktoriji in datoteke, ki so se naložile za naš projekt.

8.1.1 Uvoz knjižnic in stilov

Prva vrstica vključuje React, temeljno knjižnico za izgradnjo uporabniških vmesnikov. To omogoča uporabo komponent in drugih osnovnih funkcionalnosti, ki jih ponuja React. Osnovna uporabljena React hook-a (kavlja) sta useState in useEffect, omogočata spreminjanje stanja in stila pri razvoju funkcionalnih komponent.

UseState se uporablja za sledenje in posodabljanja stanja v komponenti, medtem ko useEffect omogoča izvajanje stranskega učinka v komponentah.

Druga vrstica vključuje knjižnico axios, ki se uporablja za izvajanje asinhronih HTTP zahtev. Ta knjižnica je ključna za pridobivanje podatkov iz zunanje storitve, v tem primeru PokeAPI, ki zagotavlja informacije o Pokémonih.

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import '../styles/Pokedex.css';
```

Slika 7: Uporabljene knjižice v aplikaciji

(Vir: Lasten vir)

8.1.2 PokemonCard komponenta

Komponenta PokemonCard predstavlja ključni del aplikacije, saj omogoča prikaz podrobnosti posameznega Pokémona. Uporaba stanja (useState) je ključna za shranjevanje podatkov o Pokémonu, ki so pridobljeni preko asinhronega klica API-ja. Stanje omogoča dinamično posodabljanje komponente ob vsaki spremembi podatkov, kar prispeva k odzivnosti uporabniškega vmesnika.

Dodatno, PokemonCard izkorišča učinek (useEffect), ki omogoča izvajanje asinhronih operacij v funkcionalnih komponentah. V tem primeru se učinek uporablja za pridobivanje podatkov o Pokémonu ob prvem priklicu komponente ter tudi ob vsaki spremembi imena Pokémona. To zagotavlja, da se podatki osvežijo, ko uporabnik izbere različnega Pokémona za ogled, s tem ohranjajoč doslednost prikaza informacij.

8.1.3 Pokedex komponenta

Komponenta Pokedex je oblikovana tako, da uporabniku omogoča pregleden prikaz seznama Pokémonov s pomočjo paginacije. S pomočjo stanja, ki sledi z uporabo useState, se spremlja trenutni seznam Pokémonov, ki se osvežuje ob vsaki spremembi. Za implementacijo paginacije se uporablja useEffect, ki ob prvem priklicu komponente izvaja asinhrono pridobivanje

seznama Pokémonov s pomočjo HTTP zahtevka na PokeAPI. Parametra count in startIndex se uporabljata za določitev števila Pokémonov na strani in začetnega indeksa paginacije.

```
const [pokemonList, setPokemonList] = useState([]);
const count = 1000; // Število Pokémonov na strani
const startIndex = 0; // Začetni indeks za paginacijo
```

Slika 8: Primer paginacije

(Vir: Lasten vir)

V samem prikazu komponente se znotraj diva z razredom »pokemon-container« dinamično generira seznam Pokémonov, kjer se za vsakega ustvari komponenta PokemonCard. To omogoča enostaven prikaz in prehajanje med različnimi stranmi Pokémonov, hkrati pa ohranja preglednost in učinkovitost. Paginacija zagotavlja, da se uporabniku predstavi le omejeno število Pokémonov naenkrat, kar prispeva k izboljšani uporabniški izkušnji. Celotna implementacija omogoča enostavno upravljanje in navigacijo med Pokémoni v okviru Pokedex komponente.

8.1.4 Pridobivanje podatkov z HTTP zahtevkom

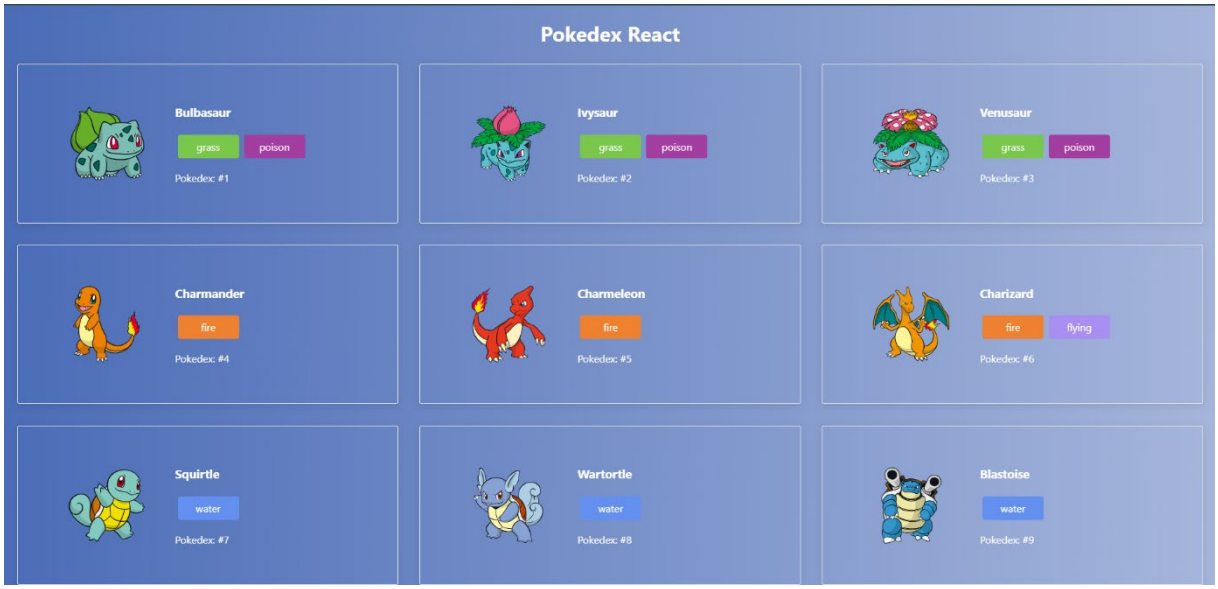
Podatki v Pokédex aplikaciji so pridobljeni preko asinhronega HTTP zahtevka, ki ga izvajamo s pomočjo funkcije fetch(). Ta metoda omogoča preprosto pridobivanje podatkov iz zunanjega vira, v našem primeru API-ja PokeAPI. Z uporabo fetch() funkcije v JavaScriptu lahko pošljemo zahtevo na določen URL in nato obdelamo odgovor, ki ga prejmemo.

8.1.5 Meritve

Spletna stran uspešno naloži podatke o Pokémonih iz PokeAPI v izjemno hitrem času, le 2.85 sekund. Ta rezultat kaže na učinkovito izvajanje asinhronih zahtev na API, kar omogoča hitro pridobivanje informacij o posameznih Pokémonih. Kljub temu pa je vredno opozoriti, da je celoten čas nalaganja strani, vključno s slikami, daljši in znaša 11 sekund. Ta razlika med hitrostjo pridobivanja podatkov in celotnim časom nalaganja lahko nakazuje, da so slike bistveno prispevale k skupnemu času nalaganja strani. To poudarja pomembnost optimizacije velikosti in načina nalaganja slik ter kaže na možnosti za izboljšanje učinkovitosti celotne uporabniške izkušnje.

8.1.6 Končna aplikacija

Pokédex aplikacija predstavlja pregledno in uporabniku prijazno orodje za raziskovanje sveta Pokémonov. S pomočjo te aplikacije lahko uporabniki enostavno pregledujejo seznam Pokémonov, podatki so pridobljeni iz API-ja (PokeAPI). Vsak Pokémon je predstavljen s sliko, imenom in vrsto, kar omogoča pogled vsakega lika. Aplikacija uporablja učinkovito paginacijo, ki omogoča brskanje skozi seznam, tako se obdržita preglednost in optimalna hitrost nalaganja strani.



Slika 9: Izgled React aplikacije

(Vir: Lasten vir)

8.2 Angular aplikacija

Pri tej aplikaciji smo ustvarili eno komponento in en »service«. V Angular aplikaciji je bilo ustvarjanje teh dokumentov zelo lahko, saj smo v terminal vpisali samo kratico ng generate pokemon-list in ng generate service pokemon, aplikacija pa deluje na enak način kot React aplikacija.

8.2.1 App.module.ts dokument

Ta dokument predstavlja Angular modul, imenovan AppModule, ki je osrednji del Angular aplikacije. V tem modulu so navedene deklaracije komponent, uvoz modulov in storitve, ki jih

aplikacija uporablja. Deklaracija komponent vključuje AppComponent in PokemonListComponent, ki predstavljata osnovne gradnike aplikacije. Uvoz modulov vključuje BrowserModule, AppRoutingModule in HttpClientModule, ki zagotavljajo osnovno funkcionalnost aplikacije, definicije poti in komunikacijo s strežnikom preko HTTP protokola. Na koncu je navedena tudi glavna komponenta, ki se inicializira ob zagonu aplikacije, in sicer AppComponent. Ta modul predstavlja temelj Angular aplikacije in vključuje vse potrebne elemente za njeno delovanje.

```
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { PokemonListComponent } from './pokemon-list/pokemon-list.component';

@NgModule({
  declarations: [
    AppComponent,
    PokemonListComponent,
  ],
  imports: [
    AppRoutingModule,
    HttpClientModule
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Slika 10: Implementacija app.module dokumenta

(Vir: Lasten vir)

8.2.2 Storitev – Pokemon

Datoteka pokemon.service.ts predstavlja Angular storitev, ki je odgovorna za komunikacijo s PokeAPI-jem in pridobivanjem podatkov o Pokémonih. Na začetku datoteke so uvožene potrebne knjižnice iz Angular frameworka, kot so Injectable in HttpClient, ki omogočajo komunikacijo s strežnikom preko HTTP protokola. Observable je uvožen iz knjižnice rxjs, kar omogoča asinhrono obdelavo podatkov.

Storitev DataService je označena s @Injectable() dekoratorjem, kar omogoča, da je dostopna za uporabo v drugih komponentah ali storitvah v aplikaciji. V konstruktorju je uporabljen

HttpClient objekt, ki ga bomo uporabljali za izvajanje HTTP zahtev na API strežnik Pokémonov.

```
import { Component, OnInit } from '@angular/core';
import { DataService } from '../services/pokemon.service';
import { forkJoin } from 'rxjs';

@Component({
  selector: 'app-pokemon-list',
  templateUrl: './pokemon-list.component.html',
  styleUrls: ['./pokemon-list.component.css']
})
export class PokemonListComponent implements OnInit {
  pokemons: any[] = [];
  currentPage = 1;
  itemsPerPage = 52;

  constructor(private dataService: DataService) { }
```

Slika 11: Uvožene knjižice in konstruktor aplikacije

(Vir: Lasten vir)

Metoda getPokemons omogoča pridobivanje seznama Pokémonov z določeno stranjo in velikostjo strani. Parametra page in pageSize določata, katero stran Pokémonov in koliko Pokémonov na stran bomo pridobili. Glede na podane parametre se izračuna offset, ki predstavlja začetni indeks v API rezultatu, od katerega bomo pridobivali podatke. Nato se uporabi HttpClient objekt za izvajanje GET zahteve na URL API-ja, ki pridobi želeno število Pokémonov.

Metoda getMoreData omogoča pridobivanje dodatnih podatkov o določenem Pokémonu na podlagi njegovega imena (pokemonName). Uporablja se tudi HttpClient objekt za izvajanje GET zahteve na URL API-ja, ki pridobi podrobnosti o izbranem Pokémonu.

```
  getPokemons(page: number, pageSize: number): Observable<any> {
    const offset = (page - 1) * pageSize;
    return this.http.get(`https://pokeapi.co/api/v2/pokemon?offset=${offset}&limit=${pageSize}`);
  }

  getMoreData(pokemonName: string): Observable<any> {
    return this.http.get(`https://pokeapi.co/api/v2/pokemon/${pokemonName}`);
  }
}
```

Slika 12: Metodi getPokemons in getMoreData

(Vir: Lasten vir)

8.2.3 PokemonList komponenta

Ta datoteka predstavlja Angular komponento PokemonListComponent, ki je odgovorna za prikazovanje seznama Pokémonov v aplikaciji. Uvožene so potrebne knjižnice iz Angular frameworka, kot so Component in OnInit, te omogočajo definiranje komponent in inicializacijo ob zagonu. Uvožena je storitev DataService iz datoteke pokemon.service.ts, ki je odgovorna za pridobivanje podatkov o Pokémonih iz API-ja. Prav tako je uvožena forkJoin funkcija, ki omogoča več asinhronih operacij v eno in čaka na zaključek preden vrne rezultate.

```
import { Component, OnInit } from '@angular/core';
import { DataService } from '../services/pokemon.service';
import { forkJoin } from 'rxjs';

@Component({
  selector: 'app-pokemon-list',
  templateUrl: './pokemon-list.component.html',
  styleUrls: ['./pokemon-list.component.css']
})
export class PokemonListComponent implements OnInit {
  pokemons: any[] = [];
  currentPage = 1;
  itemsPerPage = 52;

  constructor(private dataService: DataService) { }
```

Slika 13: Uvoz knjižic za PokemonList

(Vir: Lasten vir)

Metoda loadPokemons() uporablja storitev DataService za pridobivanje seznama Pokémonov s pomočjo getPokemons() metode. Ko se odziv vrne, se za vsakega Pokémona v seznamu izvede dodatno pridobivanje podatkov s pomočjo getMoreData() metode. Uporaba forkJoin funkcije nam omogoča, da se vsi asinhroni klici dokončajo, preden se izvede nadaljnja logika. Končni rezultat je seznam vseh Pokémonov, ki se shrani v pokemons lastnost komponente.

```
ngOnInit(): void {
  this.loadPokemons();
}

loadPokemons(): void {
  this.dataService.getPokemons(this.currentPage, this.itemsPerPage)
    .subscribe((response: any) => {
      const requests = response.results.map((result: any) =>
        this.dataService.getMoreData(result.name)
      );

      forkJoin(requests)
        .subscribe((responses: any) => {
          this.pokemons = responses;
          console.log(this.pokemons); // Verify if data is populated correctly  ×31 ᵐ Array(52) [ { abilities:
        });
    });
}
```

Slika 14: Implementacija loadPokemons metode

(Vir: Lasten vir)

Prav tako imamo implementirane metode nextPage() in previousPage(), ki predstavljajo paginacijo stani in omogočajo prehod na naslednjo oz. prejšnjo stran Pokémonov tako da povečata oz. zmanjšata vrednost currentPage in ponovno pokličeta metodo loadPokemons().

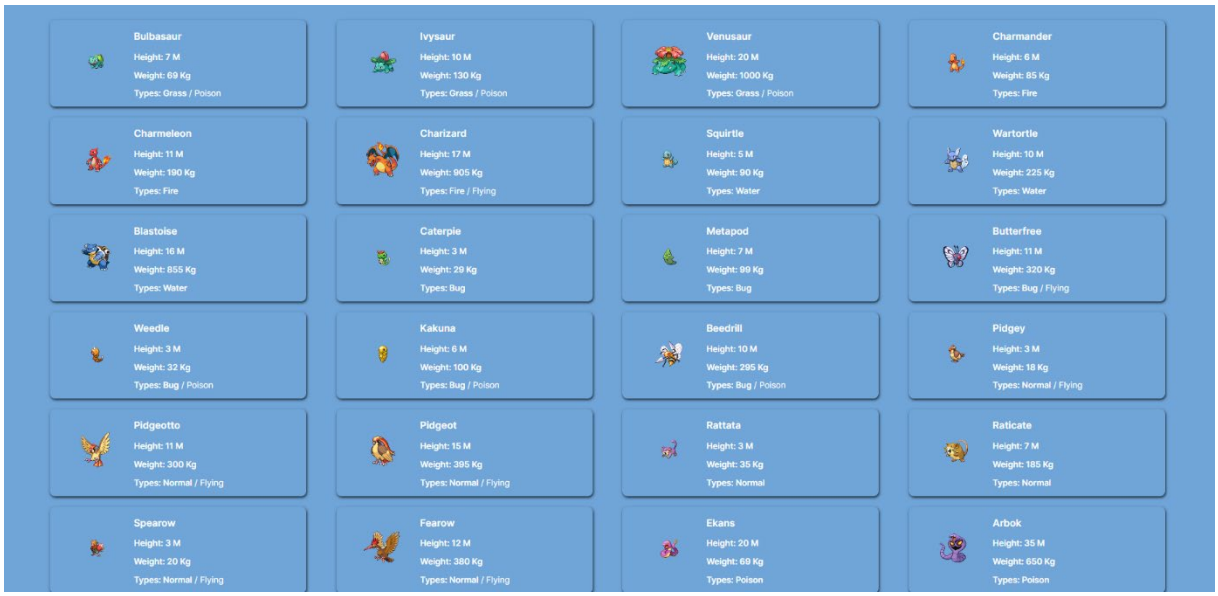
8.2.4 Meritve

Aplikacija je zgrajena po enakih principih kot ta v React-u. Prav zaradi tega bomo videli realno razliko med obema tehnologijama.

Celotna aplikacija potrebuje 7.7 s, da naloži vse Pokemone oziroma jih naloži s paginacijo. Posamezne slike se nalagajo v paketih po 52 naenkrat, za kar je odgovorna paginacija. Za prvih 52 slik Pokemonov potrebuje aplikacija natanko 0.02 s. Pri tem moramo upoštevati, da naloži sprva 52 Pokemonov in ne vseh 1000 naenkrat.

8.2.5 Končna aplikacija

Aplikacija je zasnovana tako, da omogoča hitro in pregledno pregledovanje seznamov Pokémonov, pri čemer so podatki pridobljeni preko zanesljivega API-ja (PokeAPI). Vsak Pokémon je predstavljen s sliko, imenom, višino, težo in vrsto, kar omogoča popoln vpogled v značilnosti vsakega lika. Aplikacija izkorišča učinkovito paginacijo, ki omogoča gladko brskanje skozi seznam, hkrati pa ohranja preglednost in optimalno hitrost nalaganja strani.



Slika 15: Izgled Angular aplikacije

(Vir: Lasten vir)

9 MERITVE

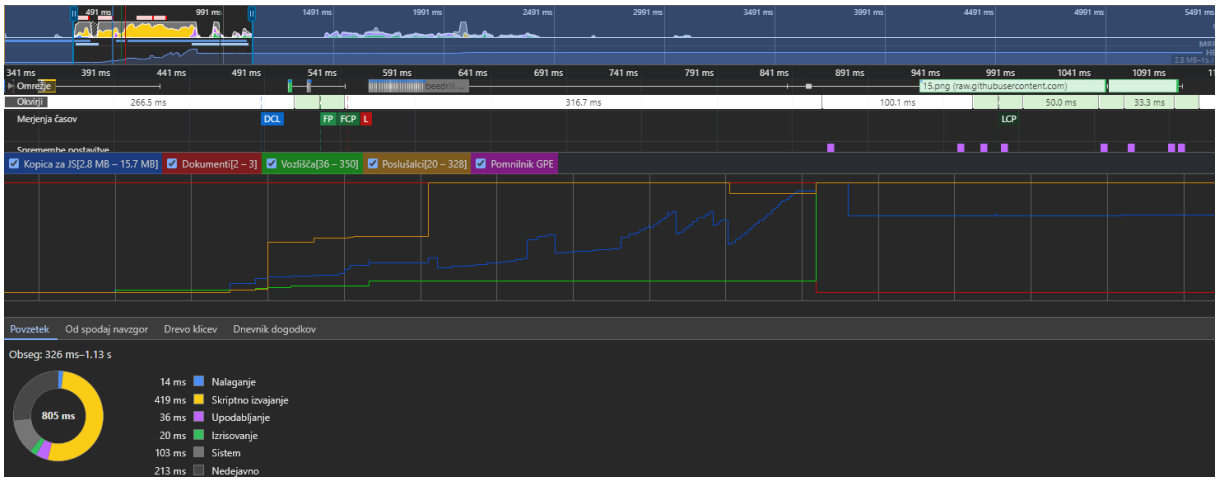
Merjenje in analiza različnih značilnosti ter zmogljivosti sta ključna koraka pri primerjavi obeh tehnologij in objektivni primerjavi obeh tehnologij. S pomočjo različnih parametrov in različno količino podatkov bomo ocenili obremenjenost sistema pri gostitelju kot tudi pri odjemalcu in hitrost nalaganja aplikacije.

9.1 Obremenjenost gostitelja

Potrebne meritve obeh aplikacij bomo izvedli s pomočjo brskalnika Google Chrome in razvijalskih orodij, ki so že vgrajena v brskalnik. S tem bomo nudili lahek pregled raznih parametrov aplikacije. Meritve bomo opravili za različne količine podatkov, ki jih bo aplikacija pridobila z API-ja. Odločili smo se, da bomo uporabili meritve za 15, 100 in 250 pridobljenih paketov s prej omenjenega API-ja

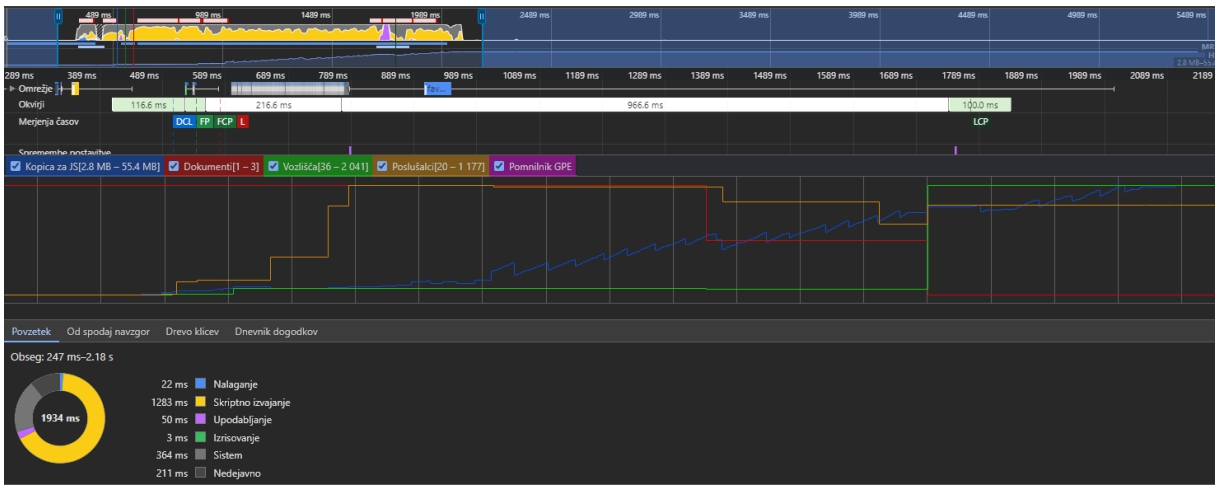
9.2 Meritve React aplikacije

9.2.1 Obremenjenost pomnilnika



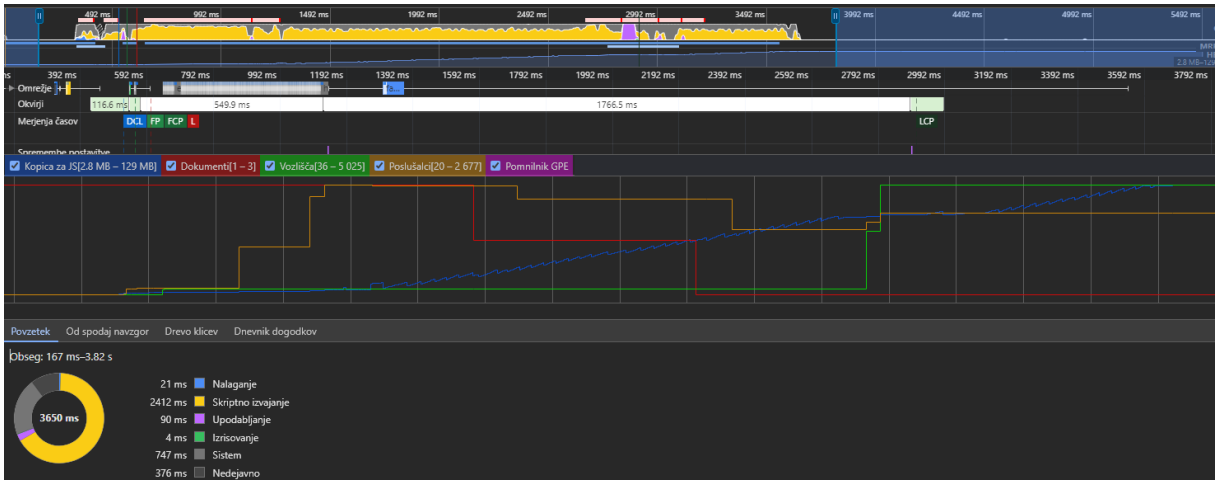
Slika 16: Posnetek zaslona obremenjenosti pomnilnika za 15 api klicev – React

(Vir: Lasten vir)



Slika 17: Posnetek zaslona obremenjenosti pomnilnika za 100 api klicev – React

(Vir: Lasten vir)

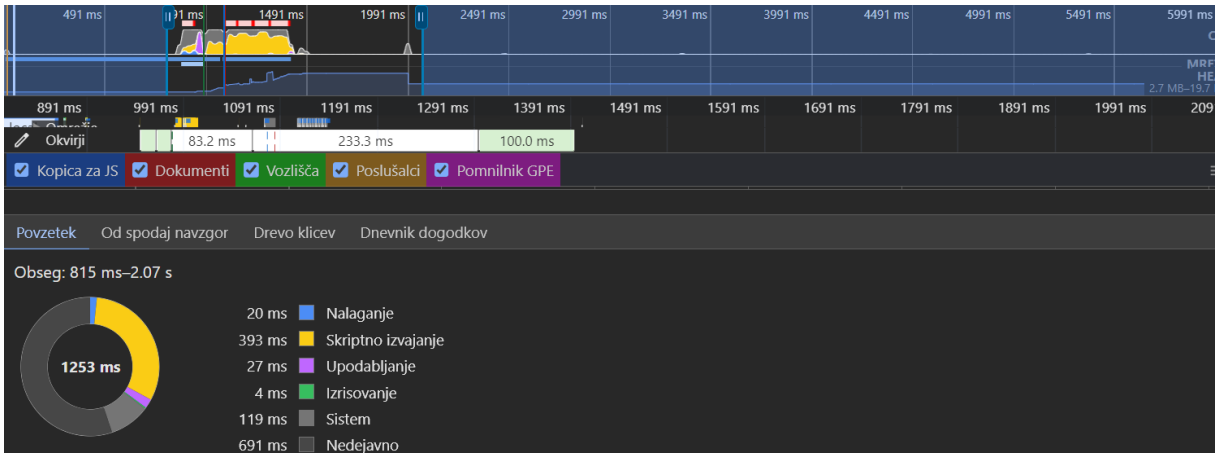


Slika 18: Posnetek zaslona obremenjenosti pomnilnika za 250 api klicev – React

(Vir: Lasten vir)

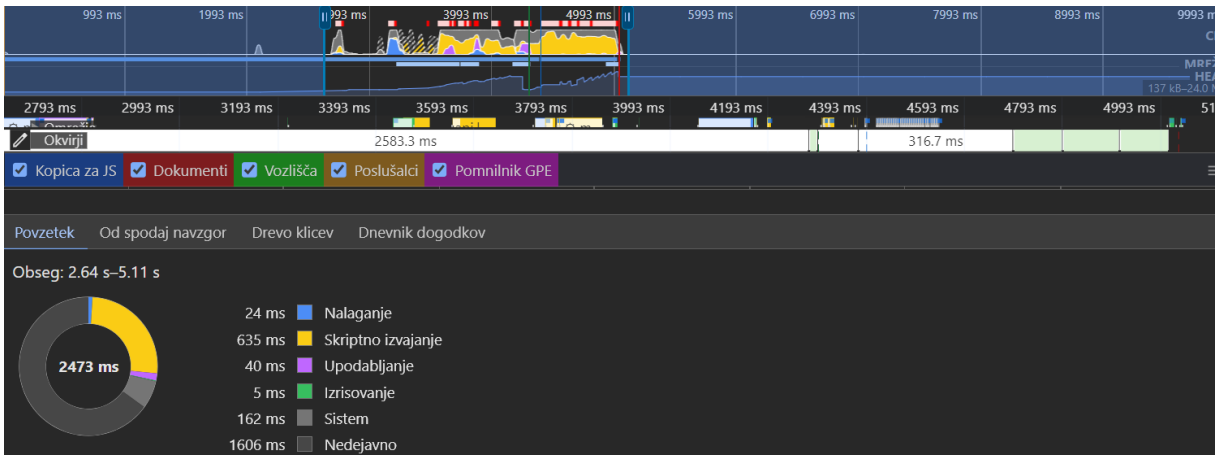
9.3 *Meritve Angular aplikacije*

9.3.1 Obremenjenost pomnilnika



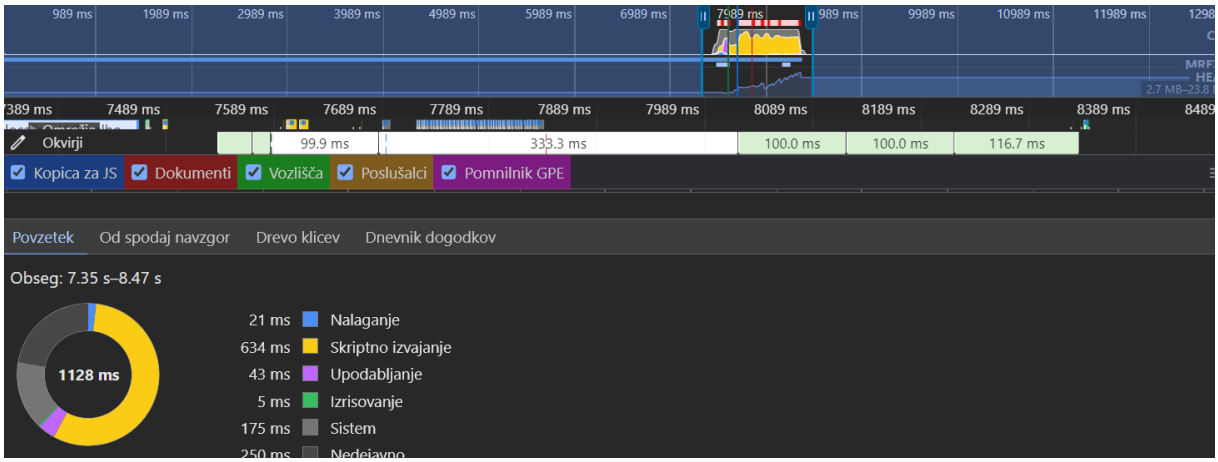
Slika 19: Posnetek zaslona obremenjenosti pomnilnika za 15 api klicev – Angular

(Vir: Lasten vir)



Slika 20: Posnetek zaslona obremenjenosti pomnilnika za 100 api klicev – Angular

(Vir: Lasten vir)



Slika 21: Posnetek zaslona obremenjenosti pomnilnika za 250 api klicev – Angular

(Vir: Lasten vir)

9.4 Primerjava meritev

Tabela 3:Primerjava pridobljenih meritev za obe aplikaciji (Lasten vir)

	React aplikacija	Angular aplikacija
Čas za 15 api klicev	326 ms – 1.13 s	815 ms – 2.07 s
Čas za 100 api klicev	247 ms – 2.18 s	2.64 s – 5.11 s
Čas za 250 api klicev	167 ms – 3.82 s	7.35 s – 8.47 s

Vir: Lasten vir

10 OCENITEV UPORABNIŠKE IZKUŠNJE

Angular je celovito ogrodje, ki ponuja veliko funkcionalnosti že v osnovi. Razvijalci imajo na voljo vgrajene rešitve za usmerjanje, upravljanje stanja aplikacije, oblikovanje obrazcev in še več. Ta pristop zagotavlja enoten način dela in visoko stopnjo doslednosti med različnimi aplikacijami. Poleg tega Angular močno spodbuja uporabo TypeScripta, kar omogoča boljšo strojno podporo in večjo varnost kode.

Po drugi strani pa je React.js knjižnica za gradnjo uporabniških vmesnikov, ki se osredotoča na komponentno arhitekturo. Razvijalci imajo več svobode pri izbiri dodatnih knjižnic in orodij za gradnjo aplikacij. To omogoča večjo prilagodljivost in možnost izbire najboljše rešitve za specifične potrebe projekta. Poleg tega React ne postavlja zahtev za uporabo določenega jezika, zato ga lahko uporabljajo tako ljubitelji JavaScripta kot TypeScripta.

10.1 Produktivnost

Pri produktivnosti razvijalca ima Angular nekatere prednosti zaradi svoje celovitosti. Ker že vsebuje večino funkcionalnosti, razvijalci porabijo manj časa za iskanje in integracijo zunanjih knjižnic ter orodij. Poleg tega ima Angular boljšo vgrajeno dokumentacijo, kar olajša učenje in razvoj aplikacij.

Vendar pa je React prav tako priljubljen zaradi svoje enostavnosti in prilagodljivosti. Razvijalci imajo več svobode pri izbiri pristopa in arhitekture aplikacije, kar lahko pripomore k boljši produktivnosti in učinkovitosti v določenih primerih. Poleg tega je veliko skupnosti in knjižnic, ki so združljive z Reactom, kar omogoča hitrejši razvoj s podporo že obstoječih rešitev.

10.2 Učinkovitost

Ko gre za učinkovitost, sta tako Angular kot React sposobna zagotoviti odzivne in zmogljive uporabniške izkušnje. Vendar pa se lahko njihove značilnosti razlikujejo glede na specifične zahteve in implementacije aplikacij. Angular ima na splošno dobro optimizirano infrastrukturo, ki omogoča učinkovito izvajanje kompleksnih aplikacij.

Po drugi strani pa React ponuja virtualni DOM, ki omogoča inteligentno upravljanje s prenovo in posodabljanjem DOM-a. To lahko privede do boljše učinkovitosti pri določenih operacijah, kot je na primer dinamično dodajanje ali odstranjevanje komponent. Vendar pa je odvisno od razvijalca, da pravilno uporabi te funkcije za izboljšanje učinkovitosti aplikacije.

11 METODOLOGIJA

11.1 Zbiranje podatkov

Prvi korak metodologije je bilo zbiranje podatkov o ogrodju Angular in knjižici React.js. To smo izvedli s pomočjo lastnih zapiskov, kjer smo dokumentirali ključne značilnosti, prednosti, slabosti in uporabo obeh ogrodij. Poleg lastnih zapiskov smo uporabili tudi internetne vire, kot so uradne dokumentacije, spletni viri, blogi in forumi, ter knjižne vire, kot so knjige in znanstveni članki, ki obravnavajo temo ogrodij.

11.2 Analiza podatkov

Naslednji korak je bila analiza zbranih podatkov, kjer ssmo podrobno preučili značilnosti, prednosti, slabosti in uporabo obeh ogrodij. Pri analizi smo upoštevali različne vidike, vključno s sintakso, arhitekturo, usklajenostjo s standardi, učinkovitostjo, vzdržljivostjo, skupnostno podporo in razpoložljivimi orodji ter tehnologijami.

11.3 Raziskovalne metode

V svoji diplomski nalogi smo uporabljali naslednja raziskovalna pristopa: primerjalno metodo in pregled literature.

11.3.1 Primerjalna metoda

Pri primerjalni metodi smo v glavnem uporabljali in primerjali tehnologiji React in Angular ter primerjali razliko med relacijskimi in nerelacijskimi podatkovnimi bazami. Pri primerjavi knjižnice React.js in ogrodja Angular smo uporabili primerjalno metodo, ki nam je omogočila sistematično oceno obeh tehnologij.

11.3.2 Pregled literature

Pri izdelavi diplomske naloge smo uporabili širok spekter virov, ki so nam omogočili poglobljeno razumevanje obravnavanih tem in podporo za izvedbo raziskave. Med ključnimi viri so bile strokovne knjige in članki, ki opisujejo osnovne in napredne koncepte razvoja

spletnih aplikacij z uporabo React.js in Angular. Pomembni za nastanek naloge so bili še uradna dokumentacija in priročniki za obe tehnologiji, ki so nam nudili tehnične specifikacije, primere uporabe in najboljše prakse.

Za izvedbo primerjalnih analiz in tehničnih meritev smo se zanašali na orodja, kot so Google Chrome DevTools, Windows Performance Monitor, ter uradna razvojna okolja in vtičniki za Visual Studio Code. Vsi ti viri so skupaj prispevali k celovitemu in verodostojnemu zaključku naše raziskave ter omogočili natančno primerjavo med React.js in Angular.

11.3.3 Meritve

11.4 Analiza zmogljivosti

Za objektivno primerjavo zmogljivosti smo izvedli teste hitrosti, obremenitve in odzivnosti v obeh okoljih. To smo storili z uporabo Google Chrome orodji za testiranje zmogljivosti Lighthouse.

12 SKLEP

S to diplomsko nalogo smo pridobili ogromno novega znanja in ugotovili, da čeprav sta obe tehnologiji (React in Angular) narejeni za pomoč razvijalcem spletnih aplikacij, se njuna uporaba in znanje, potrebno za uporabo, precej razlikujeta. Predvsem smo ugotovili, da sta tako React kot tudi Angular zelo dobri orodji z veliko skupnostjo in ogromno viri ter literatur za začetek učenja in nadaljnjo uporabo le teh. Prav tako smo pridobili veliko znanja na področju podatkovnih baz in ugotovili, da je izbira pravega tipa le te odvisna od več faktorjev, ki jih je potrebno vzeti v obzir.

V preteklosti smo imeli kar nekaj izkušenj z orodji, ki pomagajo pri ustvarjanju spletnih strani, a smo s to diplomsko nalogo uspeli povečati svoj renome znanja na področju spletnih strani. Ugotovili smo, da imamo razvijalci ogromno izbiro med tehnologijami, ki nam pomagajo, da si olajšamo proces gradnje aplikacij, predvsem pa smo dobil kar nekaj znanja iz Reacta in Angularja ter funkcij, ki nam jih nudita le ti.

Na področju razvoja aplikacij z uporabo React.js in Angularja v kombinaciji z obdelavo Big Data in podatkovnimi bazami je še veliko prostora za napredek. Prihodnje raziskave bi lahko vključevale integracijo naprednih tehnik, kot so strojno učenje in umetna inteligenca, ki bi izboljšale učinkovitost in hitrost obdelave velikih količin podatkov v teh aplikacijah. Raziskovanje teh področij bo razvijalcem omogočilo boljše razumevanje in izkoristek potenciala React.js in Angularja pri gradnji zmogljivih, učinkovitih in prilagodljivih aplikacij za obdelavo Big Data.

Prav tako bi lahko nadaljnje raziskave izvedli pri optimizaciji zmogljivosti aplikacij, ki nalagajo veliko število API zahtevkov, kot so aplikacije za sledenje Pokemonov. Pri takšnih aplikacijah je ključno učinkovito upravljanje z veliko količino podatkov, pridobljenih iz različnih virov, ter njihova hitra in zanesljiva obdelava. Raziskave bi se lahko osredotočile na uporabo naprednih tehnik predpomnjenja in optimizacije API klicev za zmanjšanje časa nalaganja in izboljšanje uporabniške izkušnje.

Poleg tega bi se lahko prihodnje raziskave osredotočile na optimizacijo zmogljivosti aplikacij, ki nalagajo veliko število API zahtevkov, kot so aplikacije za sledenje Pokemonov. V takih aplikacijah je ključno učinkovito upravljanje z veliko količino podatkov, pridobljenih iz različnih virov, ter njihova hitra in zanesljiva obdelava, katero mislim da bi lahko v prihodnje še raziskali in izboljšali, da bodo aplikacije še hitrejše. Poleg tega bi raziskave na področju

razporejanja nalog in razdelitve bremen med strežniki lahko pripomogle k boljši skalabilnosti in zanesljivosti teh aplikacij.

Prav tako menimo, da je še ogromno potenciala za razvijanje Big Data, saj bo v prihodnosti vedno več vsega, za kar bo potrebnega ogromno raziskovanja prav na področju hitrosti in upravljanja s tako veliko količino podatkov. Raziskave bi lahko vključevale tudi boljše metode za shranjevanje in upravljanje podatkov, kot so napredne strategije shranjevanja in replikacije podatkov, ki bi lahko izboljšale dostopnost in zanesljivost podatkovnih sistemov. Raziskovanje novih pristopov k obdelavi in analizi Big Data, kot so distribuirani sistemi za obdelavo podatkov, bi lahko odprlo nove možnosti za povečanje zmogljivosti in učinkovitosti aplikacij. Ob tem bi lahko prihodnje raziskave vključevale tudi izboljšave na področju obdelave podatkov v realnem času. To bi bilo še posebej pomembno za aplikacije, ki zahtevajo sprotno obdelavo podatkov, kot so analitične platforme in sistemi za priporočila. Integracija teh naprednih tehnologij in tehnik bi razvijalcem omogočila gradnjo bolj zmogljivih in prilagodljivih aplikacij, ki lahko učinkovito obdelujejo velike količine podatkov.

H1: *Pri obdelavi velike datoteke se bolje obnese NOSQL podatkovna baza kot relacijska podatkovna baza*

[Poglavja: 2., 2.1., 2.2., 2.3., 3., 3.8]

Hipoteza je potrjena. Pri obdelavi velikega obsega podatkov je bilo opaziti, da je NOSQL podatkovna baza bolj prilagodljiva in skalabilna ter omogoča hitrejši dostop do podatkov, kar je ključnega pomena pri velikih obremenitvah. Razlogi za to bi lahko vključevali boljšo horizontalno skaliranje in optimizacijo za delo z nestrukturiranimi podatki, kar je pogosto potrebno pri obdelavi velikih datotek.

H2: *Knjižica React.js nudi več različnih gradiv za začetnike in je lažji za začetek kot*

Angular

[Poglavja: 5., 5.5., 6., 6.5., 10., 10.2]

Hipoteza je potrjena. Skupnost Reacta je res ogromna, ob temu pa je na spletu ogromno primerov projektov, videoposnetkov in dokumentacije, kjer so zelo podrobno opisane funkcionalnosti za začetek programerske poti. React je definitivno bolj primeren za začetnike, ki želijo vstopiti v svet spletnih aplikacij, ker mora programer za sam začetek vedeti veliko manj o knjižici kot pri Angularju. Sintaksa v JSX dokumentu (React) se priporoča za začetnike, saj je njena struktura zelo lahka za gradnjo komponent.

H3: *Pri prikazu manjše količine podatkov se bolje obnese ReactJS kot Angular*

[Poglavje: 3., 3.7., 8., 8.2., 8.3.]

Hipoteza je potrjena. Moramo pa upoštevati, da smo vzeli več faktorjev, kljub temu je hipoteza za večino aplikacij resnična. Ugotovitve temeljijo na meritvah obremenitve procesorja, porabe pomnilnika in časa nalaganja. ReactJS s svojim virtualnim DOM-om omogoča bolj učinkovito obdelavo sprememb v uporabniškem vmesniku, kar vodi do hitrejših posodobitev in boljšega upravljanja z velikimi podatkovnimi nizi.

H4: *Pri prikazu velike količine podatkov se bolje obnese ReactJS kot Angular*

[Poglavje: 3., 3.7., 8., 8.2., 8.3.]

Hipoteza je potrjena. V aplikaciji, ki pridobiva veliko količino podatkov naenkrat, smo videli, da se časovno gledano bolje obnese ReactJS. Ključnega pomena so funkcije, vgrajene v ReactJS, kot je virtualni DOM, ki omogočajo učinkovitejše posodabljanje in prikazovanje podatkov.

13 VIRI IN LITERATURA

Agilway. (17. 11. 2023). *Pros and Cons of Angular Development*. Pridobljeno iz Agilway: https://agiliway.com/pros-and-cons-of-angular-development/

altexsoft. (6. 9. 2023). *The Good and the Bad of Angular Development*. Pridobljeno iz altexsoft: https://www.altexsoft.com/blog/the-good-and-the-bad-of-angular-development/

Angular. (10. 2. 2024). *What is Angular?* Pridobljeno iz https://v17.angular.io/guide/what-is-angular

Budziński, M. (10. 5. 2024). *What is React Native?* Pridobljeno iz Netguru: https://www.netguru.com/glossary/react-native

Carder, J. (8. 7. 2020). *What Is SQL Database?* Pridobljeno iz Openlogic: https://www.openlogic.com/blog/what-sql-database

CSS Tutorial. (10. 5. 2024). Pridobljeno iz W3School: https://www.w3schools.com/css/

Daniels, P., & Atencio, L. (2017). *RxJS in Action.* Manning.

Deshpande, C. (5. 10. 2023). *The Best Guide to Know What Is React*. Pridobljeno iz Simplilearn: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

Dhaduk, H. (6. 4. 2023). *Angular vs React: Which to Choose for Your Front End in 2024?* Pridobljeno iz Simform: https://www.simform.com/blog/angular-vs-react/

Dulanga, C. (2. 12. 2020). *Incremental vs Virtual DOM*. Pridobljeno iz Medium: https://blog.bitsrc.io/incremental-vs-virtual-dom-eb7157e43dca

Elar, S. (10. 5. 2019). *JavaScript frameworks: Angular vs React vs Vue*. Pridobljeno iz Theseus: https://www.theseus.fi/handle/10024/261970

Emadamerho-Atori, N. (31. 8. 2023). *Angular vs. React vs. Vue.js: Comparing performance*. Pridobljeno iz LogRocket: https://blog.logrocket.com/angular-vs-react-vs-vue-js-comparing-performance/

Gavigan, D. (3. 4. 2018). *The history of Angular*. Pridobljeno iz Medium: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7

Haidi, Z. (8. 2019). *Primerjava ogrodij za upodabljanje spletnih strani na strežniku.* Pridobljeno iz DKUM: https://dk.um.si/Dokument.php?id=144041&lang=slv

HTML Tutorial. (10. 5. 2024). Pridobljeno iz W3School: https://www.w3schools.com/html/

Introduction to NoSQL. (13. 10. 2023). Pridobljeno iz GeeksForGeeks: https://www.geeksforgeeks.org/introduction-to-nosql/

Introduction to the DOM. (29. 11. 2023). Pridobljeno iz mdn web docs: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Krotoff, T. (10. 5. 2024). *Frontend Freamework Popularity*. Pridobljeno iz Github: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

Maowa, J., Sajedul Hoque, A., Mustafa, R., & Osiur Rahman, M. (3. November 2017). A COMPARATIVE STUDY ON BIG DATA HANDLING. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 1-8.

Marr, B. (2015). *Big Data: Using SMART Big Data, Analytics and Metrics To Make Better Decisions and Improve Performance.* Wiley.

MongoDB. (10. 5. 2024). *Uradna dokumentacija*. Pridobljeno iz MongoDB: https://www.mongodb.com/

Onyekani, C. (5. 9. 2023). *How to Use React Components – Props, Default Props, and PropTypes Explained*. Pridobljeno iz freeCodeCamp: https://www.freecodecamp.org/news/how-to-use-react-components/

Patel, H. (15. 2. 2023). *How Virtual DOM in React Works and Why It Matters?* Pridobljeno iz LinkedIn: https://legacy.reactjs.org/docs/faq-internals.html

Pros and Cons of ReactJS. (10. 5. 2024). Pridobljeno iz JavaTPoint: https://www.javatpoint.com/pros-and-cons-of-react

React. (10. 5. 2024). *Uradna dokumentacija*. Pridobljeno iz React: https://legacy.reactjs.org/docs/getting-started.html

React getting started. (25. 12. 2023). Pridobljeno iz mdn web docs.

React.JS: Advantages and Disadvantages. (6. 9. 2023). Pridobljeno iz Infoline Tagline: https://taglineinfotech.com/reactjs-advantages-and-disadvantages/

ReactJS Virtual DOM. (22. 11. 2023). Pridobljeno iz GeeksForGeeks: https://www.geeksforgeeks.org/reactjs-virtual-dom/

Shacklett, M. (3. 11. 2023). *Structured vs Unstructured Data: Key Differences*. Pridobljeno iz Datamation: https://www.datamation.com/big-data/structured-vs-unstructured-data/

Shagufta, N. (10. 5. 2023). *React JS/Node JS.Pros and Cons*. Pridobljeno iz LinkedIn: https://www.linkedin.com/pulse/react-jsnode-jspros-cons-shagufta-naz/

Shyam, S. (2018). *Angular: Up and Running: Learning Angular, Step by Step.* O'Reilly Media, Inc.

Smallcombe, M. (15. 2. 2024). *SQL vs NoSQL: 5 Critical Differences*. Pridobljeno iz Integrate.io: https://www.integrate.io/blog/the-sql-vs-nosql-difference/

Taylor, C. (21. 6. 2023). *The History of ReactJS*. Pridobljeno iz HrTech Cube: https://hrtechcube.com/the-history-of-reactjs/

Technostacks. (2. 1. 2024). *React vs Angular: Which Is A Better JavaScript Framework?* Pridobljeno iz Technostacks: https://technostacks.com/blog/react-vs-angular/

Tiao, S. (11. 3. 2024). *What Is Big Data?* Pridobljeno iz Oracle: https://www.oracle.com/big-data/what-is-big-data/

Tiwari, V. (10. 6. 2023). *A Guide to DOM Manipulation in Angular 14: Improved Performance*. Pridobljeno iz Medium: https://tiwarivikas.medium.com/a-guide-to-dom-manipulation-in-angular-14-improved-performance-65f0d7478d8d

Tuama, D. Ó. (10. 5. 2024). *What is DOM?* Pridobljeno iz CodeInstitute: https://codeinstitute.net/global/blog/what-is-dom/

W3. (10. 5. 2024). *AngularJS Data Binding*. Pridobljeno iz W3School: https://www.w3schools.com/angular/angular_databinding.asp

Wieruch, R. (2018). *The Road to React: Your journey to master plain yet pragmatic React.js.*

14 PRILOGE

Priloge so dodane na priloženem CD-ju.