

Programi për Shkenca Kompjuterike dhe Inxhinierisë

**APLIKACION PËR KOMUNITETIN E LEHONAVE**

Shkalla Bachelor

Eroll Muslijae

Prill / 2022

Prishtinë



Programi për Shkenca Kompjuterike dhe Inxhinierisë

Punim Diplome

Viti akademik 2015-2016

Eroll Muslija

**APLIKACION PËR KOMUNITETIN E LEHONAVE**

Mentori: MSc. Medina Shamolli

Prill / 2022

Ky punim është përpiluar dhe dorëzuar në përmbushjen e kërkesave të pjesshme për Shkallën Bachelor

# ABSTRAKT

Në këtë punim do të shtjellojmë Framework-un Javascript ku hulumtimi kryesorë është në pjesën e frontend në Recat Native.

Rritja dhe zhvillimi i kësaj teknologjie në mënyrë eksponenciale ka shtyrë shumë kompani dhe shumë zhvillues të marrin njohurit e duhur sepse gjen zbatim të duhur në produktet përkatëse të tyre.

Në këtë punim do të shtjelloj elementet e web-it, ndarjet e tij në anën ndërfaqës së përdoruesit front-end dhe të logjikës pra back-end gjithashtu dhe mënyrën e funksionimit.

Punimi bazohet në popullaritetin dhe mbështetjen, performancat, skriptat dhe renderët gjithashtu avantazhet dhe disavantazhet e kësaj teknologjie dhe përdorimin e saj nga industritë më të mëdha në botë. Gjithashtu ky punime permban në vete nje projket me React Native duke shtjelluar më në detaje pjesen disa pjese te caktuara te aplikaccionit.

# MIRËNJOHJE/FALENDERIME

Këtë punim diplome u’a dedikoj familjes sime, shoqërisë sime dhe të gjithë mësimdhënësve të mi që më shoqëruan dhe mundësuan arritjen e këtij rrugëtimi shumë të rëndësishëm profesional.

# PËRMBAJTJA

# LISTA E FIGURAVE

# LISTA E SHKURTESAVE DHE KONCEPTEVE KRYESORE

**Front-end/Client-Side** – Pjesa e web faqes që shihet nga përdoruesi, ndërfaqja grafike e aplikacionit.Pra, pjesa që ekzekutohet në shfletuesit e përdoruesit.

**Back-end/Server-side** – Pjesa e web faqes që nuk shihet nga përdoruesi, e furnizon dhe e përgatite pjesën front-end për të u shfaqur. Pra. pjesa që ekzekutohet në servera, jo në shfletuesit.

**DOM** (Document Object Model) -Një pemë standarde lloj strukture apo modeli të një dokumenti HTML për qasje të elementeve.

**JSON** (JavaScript Object Notation) - Formati i të dhënave të lexueshme nga njeriu, i përdorur në JavaScript.

**MDM** (Master Data Management) - Proceset, strategjitë dhe menaxhimet të lidhura me të dhënat master të kompanisë.

**MVC** (Model-View-Controller) - Modeli arkitektonik i softuerit, i cili ndan një aplikacion në një model, një pamje dhe një kontrollues.

**UI** - User interface

**TypeScript** - Një superset i JavaScript që shton optional typing në JavaScript.

**SPA** (Single-page application) - Në arkitekturën e uebit, aplikacioni në internet i cili përmban teknikisht vetëm një faqe, dhe faqet krijohen në mënyrë dinamike dhe të

modifikuar programatikisht.

**Framework** – Platformë për zhvillimin e aplikacioneve softuerike. Siguron një bazë mbi të cilin zhvilluesit e aplikacioneve mund të ndërtojnë aplikacione për një platformë specifike.Riorganizon procesin e zhvillimit pasi që programuesit nuk kanë nevojë përsëri të zbulojnë rrotën çdo here që të zhvillojnë një aplikim të ri.

Mund të përfshijë klasa dhe funksione të paracaktuara që mund të përdoren për përpunimin e të dhënave, menaxhimin e pajisjeve harduerike dhe bashkëpunimin me sistemin operativ.

**API** – Ndërfaqe për programimin e aplikacioneve, është një grumbull rutinash, strukturave të dhënave, klasave si dhe protokolleve të ofruara nga libraritë ose/edhe shërbimet të sistemeve me qëllim të përkrahjes të zhvillimit aplikacioneve.

**HTTP** – HyperText Transfer Protocol

**HTTPS** – HyperText Transfer Protocol Secure

**DNS** – Domain Name Servers

**CSRF**- Cross-Site Request Forgery është një sulm që detyron një përdorues përfundimtar të ekzekutojë veprime të padëshiruara në një aplikacion në internet në të cilin aktualisht janë vërtetuar.

# 1.HYRJE

Në fushën e programimit shpesh hasim në dy terme: Front End dhe Back End, të cilat së bashku përbëjnë programimin në tëresi.

Front End është gjithëçka që përfshihet ne atë që përdoruesi sheh, duke përfshirë dizajnin, fotografitë, përmbajtjen, strukturën. Kjo mundësohet përmes gjuhëve HTML, CSS dhe Javascript. Back End është gjithëcka që mundëson funksionimin e elementëve, logjikën dhe të dhënat. Gjuhët që përdoren për Back End janë të shumëllojshme, si psh. JavaScript, Python, PHP, C#, SQL, JAVA, RUBY, MongoDB etj. Një tjetër tërm mjaft i njohur, i cili krijohet pas bashkimit të Front-it dhe Back-it është Full Stack. Nuk ka ndonjë dallim strikt në mes të dy termeve të lartcekura, disa programerë shpeshherë dijnë të programojnë në Front End por jo në Back End, por edhe anasjelltas.

Në këtë punim diplome do të ndalem më shumë te gjuha programuese JavaScript dhe framework-un e saje Vue.js. Javascript është një gjuhë programuese kompjuterike dinamike. Është gjuhë e lehtë (nuk shfrytëzon shumë memorie të kompjuterit), cross-platform (përdorët në shumë platforma) dhe më së shumti përdoret si pjesë e ueb faqeve, implementimet e së cilës e lejojnë skriptën e anës së klientit të ndërveproj me shfrytëzuesin dhe të bëjë faqet dinamike.

VueJS është një kornizë progresive me burim të hapur JavaScript që përdoret për të zhvilluar ndërfaqe interaktive të uebit. Eshtë një nga kornizat e famshme që përdoret për të thjeshtuar zhvillimin e internetit. VueJS përqendrohet në shtresën e pamjes. Mund të integrohet lehtësisht në projekte të mëdha për zhvillimin e Front-endit pa ndonjë problem.

Kur vendosni të kodoni një faqe në internet, mund të kodoni çdo aspekt të asaj faqe nga e para, por ka disa karakteristika të zakonshme të faqes në internet që kanë më shumë kuptim të zbatohen nga një model dhe këtu hyjnë apo na duhen framework-et e JavaScript. Kur përdorni framework-et JS procesi është më holistik - një framework nuk ofron vetëm një zgjidhje individuale për një problem të kodimit, ai siguron një strukturë (si një skelet ose një kornizë) që organizon pjesët e faqes tuaj ku implementohet framework-u.

Ana kryesore e përdorimit të framework-eve JavaScript është efikasiteti dhe organizimi i përgjithshëm që ato sjellin në një projekt - kodi juaj do të jetë i strukturuar mjeshtërisht dhe korniza do të sigurojë zgjidhje të gatshme për problemet e zakonshme të kodimit.

Nga ana tjetër, e gjithë ajo strukturë mund të jetë e keqja e punës me një fremework - çdo kod JavaScript që shkruani mbi kornizat JS duhet të ndjekë rregullat dhe konventat specifike për kornizën, duke kufizuar lirinë që keni kur kodoni plotësisht me dorë .

# 

# 2.DEKLARIMI I PROBLEMIT

Përcaktimi i Framework-ut më të mira JavaScript me të cilin deshirojmë të punojmë është sfiduese, pasi opsionet e konsideruara si më të mirat po ndryshojnë vazhdimisht. Në Framework-et më të njohura JavaScript përfshihen: AngularJS, Angular, React.js, Vue.js, Ember.js, Meteor.js, Aurelia.js, Polymer.js, Backbone.js, Knockout.js dhe Mercury .js. Këto janë korniza ose biblioteka të JavaScript, të cilat janë duke u zhvilluar në mënyrë aktive dhe kanë një bazë të madhe përdoruesish. Ne do të mundohemi të ndalemi te Vue. Të ndajmë përparsitë dhe magësitë e saj, dhe të dalim me një perfundim se sa është dhe ku është më i mirë.

Pyetjeve të cilave do të mundohemi t’u përgjigjemi në këte punim diplome janë:

Pse të zgjidhnim Vue.js para kornizave të tjera?

Si rrjedhin të dhënat midis komponenteve në një aplikacion Vue.js?

# 3. SHQYRTIMI I LITERATURËS

Teknologjitë bazë në krijimin e çdo web faqeje janë HTML, CSS dhe JavaScript. Teknologjitë e lartëpërmendura i përkasin strukturimit të web faqes dhe njihen si “client side” teknologji pasi i përkasin ndërfaqes së përdoruesit (Front-End) për të cilat zhvilluesit e web-it duhet të kenë njohuri të mira.

## 3.1 JavaScript

Javascript fillimisht njihej si LiveScript në vitin 1995, por në vitin 1996 Netscape e ndërroi emrin në Javascript, ndoshta për shkak të famës që gjenerohej nga gjuha programuese JAVA. Një vit më vonë, në vitin 1997 Javascript u pranua në një organizatë të quajtur Ecma International në mënyrë që të krijojë një standard të ri. EcmaScript 1 (ES1) u bë verzioni i parë i gjuhës Javascript. Pra kur themi EcmaScript i referohemi standardit ndërsa kur themi Javascript i referohemi gjuhës programuese. Pas një dekade e më shumë, më saktësisht në vitin 2009 EcmaScript 5 (ES5) u lansua me shumë tipare të reja, edhe pse ju deshën vite shfletuesve që ti implementojnë këto ndryshime të reja. Në vitin 2015 u lansua EcmaScript 6 (ES6), që ishte ndryshimi më i madh i bërë në gjuhën programuese Javascript, me shumë gjëra të reja dhe praktike për zhvilluesit. Trendi vazhdon edhe në vitet në vazhdim, ES2016 ose ES7, ES2017 ose ES8, ES2018 ose ES9, ES2019 ose ES10. ES5 mbetët verzioni që përkrahet nga të gjithë shfletuesit, ndërsa verzionet e lansuara më vonë vazhdojnë të implementohen. [1]

Pra, JavaScript është një gjuhë skriptimi ose gjuhe programimi që ju lejon të zbatoni veçori komplekse në faqet e internetit - çdo herë që një faqe në internet bën më shumë sesa thjesht të shfaq informacion statik për të cilin mund të shikoni p.sh - duke shfaqur azhurnime të përmbajtjes në kohë, harta interaktive, 2D të animuara / Grafikë 3D, etj - mund të thoni se ndoshta JavaScript është i përfshirë. Është shtresa e tretë e shtresës së shtresave të teknologjive standarde të internetit.

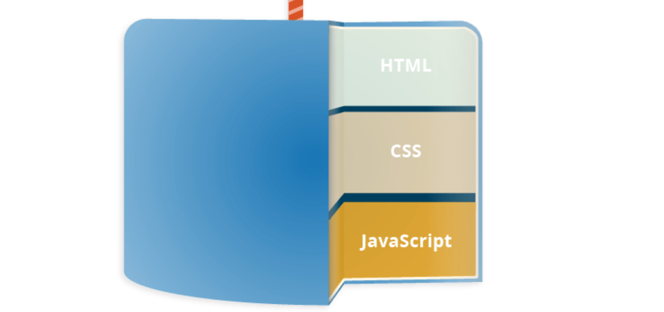


Figure 1. Struktura HTML, CSS, JavaScrpit

JavaScript nga ana e klientit përbëhet nga disa karakteristika të zakonshme të programimit që ju lejojnë të bëni gjëra të tilla si:

• Ruani vlera të dobishme brenda variablave

• Operacione në pjesë të tekstit (të njohura si "vargjet" në programim).

• Drejtimi i kodit në përgjigje të ngjarjeve të caktuara që ndodhin në një faqe në internet..

Edhe me shume!

Ajo që është edhe më emocionuese është funksionaliteti i ndërtuar në krye të gjuhës JavaScript nga ana e klientit. Të ashtuquajturat Application Programming Interfaces (API) ju ofrojnë superfuqi shtesë për t'u përdorur në kodin tuaj JavaScript. [1]

Një API është një grup i kodit që mundëson transmetimin e të dhënave midis një produkti softuer dhe një tjetri. Ai gjithashtu përmban kushtet e këtij shkëmbimi të të dhënave.

Përbëhet nga dy përbërës: Specifikimi teknik që përshkruan opsionet e shkëmbimit të të dhënave midis zgjidhjeve me specifikimet e bëra në formën e një kërkese për përpunimin dhe protokollet e shpërndarjes së të dhënave dhe ndërfaqja e softuerit e shkruar në specifikimin që e përfaqëson atë.

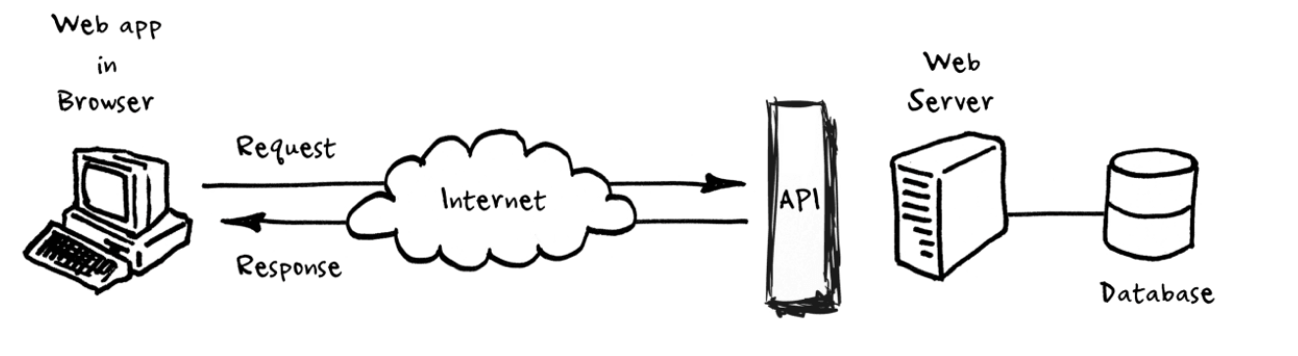


Figure 2. Si punon API

API-të shërbejnë për qëllime të shumta. Në përgjithësi, ato mund të thjeshtojnë dhe përshpejtojnë zhvillimin e softuerit. Me fjalë të tjera, API-të shërbejnë si një shtresë abstraksioni midis dy sistemeve, duke fshehur kompleksitetin dhe detajet e punës së këtij të fundit. [2]

Sa i përket politikave të lëshimit, API-të mund të jenë private, partnere dhe publike.

Por le te kthehemi prap tek JavaScript. JavaScript zbatohet në faqen tuaj HTML në një mënyrë të ngjashme me CSS. Ndërsa CSS përdor elemente <link> për të aplikuar fletë stilesh të jashtme dhe elemente <style> për të aplikuar fletë stilesh të brendshme në HTML, JavaScript ka nevojë vetëm për një tag në botën e HTML - elementin <script>.

Kur JavaScript debutoi në 1996, ai shtoi ndërveprimin dhe ngazëllimin e herëpashershëm në një rrjet që, deri në atë kohë, përbëhej nga dokumente statike. Rrjeti u bë jo vetëm një vend për të lexuar gjëra, por për të bërë gjëra. Popullariteti i JavaScript u rrit në mënyrë të vazhdueshme. Zhvilluesit të cilët kanë punuar me JavaScript shkruajnë mjete për të zgjidhur problemet me të cilat përballen dhe i paketuan ato në paketa të ripërdorshme të quajtura biblioteka, në mënyrë që të mund të ndajnë zgjidhjet e tyre me të tjerët. Ky ekosistem i përbashkët i bibliotekave ndihmoi në formësimin e rritjes së internetit.

Ardhja e kornizave moderne JavaScript e ka bërë shumë më të lehtë ndërtimin e aplikacioneve shumë dinamike, ndërvepruese. Një kornizë është një bibliotekë që ofron mendime se si ndërtohet softueri. Këto mendime lejojnë parashikueshmëri dhe homogjenitet në një aplikacion; parashikueshmëria lejon që softueri të shkallëzohet në një madhësi të madhe dhe të jetë ende i qëndrueshëm; parashikueshmëria dhe mirëmbajtja janë thelbësore për jetëgjatësinë e softuerit.

## 3.2 Përkufizimi i një kornize JavaScript

Një kornizë JavaScript është një grup i madh funksionesh dhe mjetesh lehtësuese, me rrjedhën e vet të kontrollit. Si një abstraksion, një kornizë ofron një platformë gjithëpërfshirëse të zhvillimit, një mënyrë standarde për ndërtimin e aplikacioneve. Mund të sigurojë një menaxhim të varësisë, një strukturë të sistemit të skedarëve..

Shumë nga kornizat përbëhen nga përbërës(komponentë). Komponentët janë blloqet kryesore të ndërtimit të kornizës. Mënyra se si komponentët iniciohen dhe përpunohen ndryshon midis kornizave, por karakteristikat kryesore janë të njëjtat: komponentët mund të marrin informacionin si parametra, të kryejnë veprime dhe ndoshta të japin ndonjë rezultat. Komponentët komunikojnë me njëri-tjetrin, mund të përdorin vetitë e njëri-tjetrit dhe mund të kenë prindër ose përbërës fëmijë.

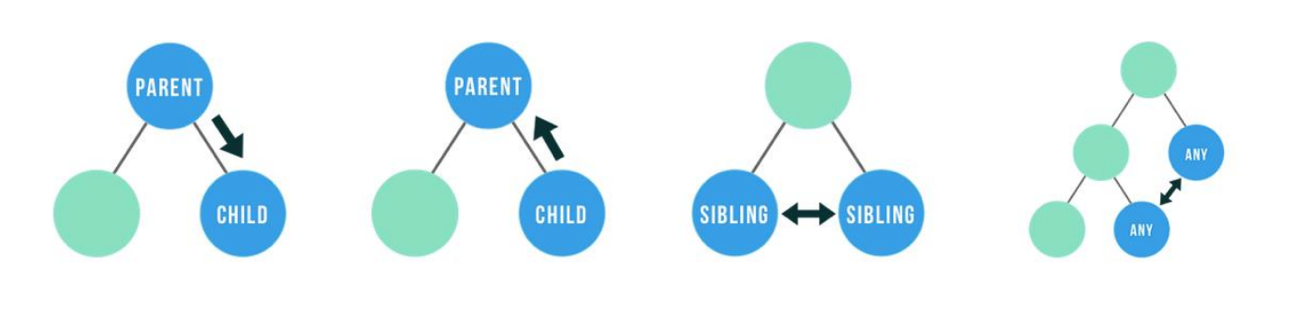


Figure 3. Relationships between components.

Një kornizë JavaScript mund të jetë një skedar i vetëm JavaScript i përfshirë në aplikacionin në internet. Pastaj, zhvilluesi mund të përdorë të gjitha funksionet e kornizës. Sidoqoftë, mund të ketë kaq shumë funksione dhe mjete që zëvendësojnë versionet e native JavaScript, sa që kodi nuk duket ose ndjehet si kod JavaScript, por gjuhë krejtësisht e re programuese.

Pra, termi "Framework" përdoret, kur rrjedha e ekzekutimit të programit zhvendoset nga zona e përgjegjësisë së zhvilluesit(developer), në fushën e përgjegjësisë së kornizës. Kjo ndan kornizat JavaScript nga bibliotekat, një bibliotekë ofron vetëm një grup funksionesh, ndërsa një kornizë menaxhon fazat e përpunimit dhe rrjedhën e të dhënave në tërë aplikacionin. [3]

## 3.3 React.js

I zhvilluar nga Facebook, u lëshua fillimisht në 2013. React është një library e JavaScript e bazuar ne framework krijuar nga Jordan Walke (Facebook Software Engineer). Facebook përdorë gjerësisht React në produktet e tyre (Facebook, Instagram dhe WhatsApp). Versioni aktual i qëndrueshëm është 16.X, i lëshuar në nëntor 2018 (me azhurnime më të vogla rritëse që nga ajo kohë). Përdoret për të ndërtuar ndërfaqet e përdoruesit (UI), pra React është shtresa e pamjes së një aplikacioni MVC (Model View Controller).

Një nga aspektet më të rëndësishme të React është fakti që ju mund të krijoni komponentë, të cilët janë si elementë HTML të personalizuar, të ripërdorshëm, për të ndërtuar shpejt

dhe me efikasitet ndërfaqet e përdoruesit. React gjithashtu rregullon mënyrën se si ruhen dhe trajtohen të dhënat, duke përdorur gjendjen dhe rekuizitat.

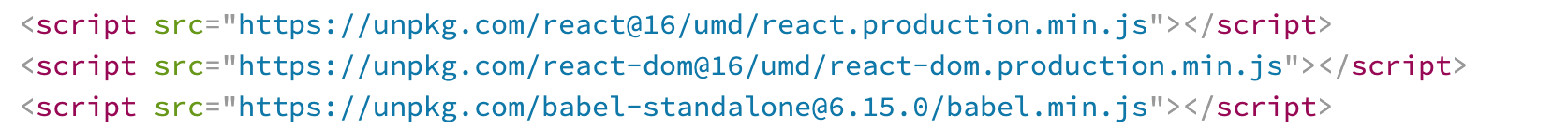
React krijon një DOM VIRTUALE në memorje.

Në vend që të manipulojë direkt ne DOM të browserit, React krijon një DOM virtual në memorje, ku bën të gjitha manipulimet e nevojshme, përpara se të bëjë ndryshimet në DOM të browserit.

Për të mund të parë qka është Rackt mund të shkruani kodin React direkt në HTML mirpo në mënyrë që të përdorni React më gjatë, ju duhet të instaloni NPM dhe Node.js.

Mënyra më e shpejtë për të filluar mësimin React është të shkruani React direkt në skedarët tuaj HTML.

Filloni duke përfshirë tre skripta, dy të parat na lejoni të shkruajmë React code në JavaScript tonë dhe e treta, Babel, na lejon të shkruajmë sintaksën JSX dhe ES6 në shfletuesit e vjetër.



<body>

<div id="mydiv"></div>

<script type="text/babel">

class Hello extends React.Component {

render() {

return <h1>Hello UBT!</h1>

}

}

ReactDOM.render(<Hello />, document.getElementById('mydiv'))

</script>

</body>

Kjo mënyrë e përdorimit të React mund të jetë në rregull për qëllime testimi, por për prodhim do t'ju duhet të krijoni një React Environment.

Nëse keni të instaluar NPM dhe Node.js, mund të krijoni një aplikacion React duke instaluar fillimisht create-react-app

Instaloni create-react-app duke ekzekutuar këtë komandë në terminalin tuaj:

Atëherë ju jeni në gjendje të krijoni një aplikacion React, përshembull ubtreact.



Pasi ta startojmë aplikacionin që kemi krijuar, një dritare e re e shfletuesit do të shfaqet me aplikacionin tuaj të sapo krijuar React!

Per te vazhduar me codin duhet të shkojmë tek follderi scr pastaj App.js dhe do të nau shfaqet ky kodë:

import logo from './logo.svg';

import './App.css';

function App() {

return (

<div className="App">

<header className="App-header">

<img src={logo} className="App-logo" alt="logo" />

<p>

Edit <code>src/App.js</code> and save to reload.

</p>

<a

className="App-link"

href="https://reactjs.org"

target="\_blank"

rel="noopener noreferrer"

>

Learn React

</a>

</header>

</div>

);

}

export default App;

Ne pastaj mund te vazhdojmë duke e modifikuar, sikur ne kodin më poshtë :

import React, { Component } from 'react';

class App extends Component {

render() {

return (

<div className="App">

<h1>Hello UBT 2020!</h1>

</div>

);

}

}

export default App;



3.4 Popullariteti dhe Mbështetja

Popullariteti i Framework-eve Front-end rritet çdo vit. Dhe në ditët e sotme vështirë se mund të imagjinohet procesi i zhvillimit të front-end pa ndonjë nga kornizat JavaScript.

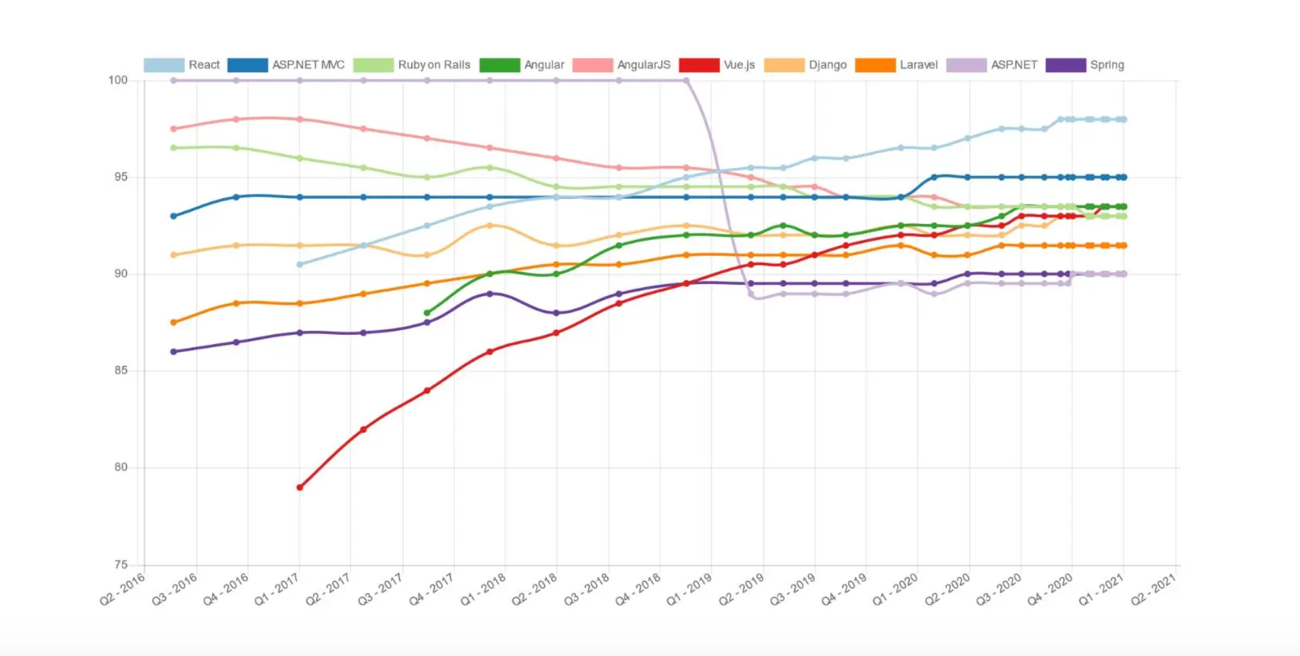
React përdorë DOM virtual .

Dallimi kryesor midis frameworkeve te tjere dhe React është se React përdor JSX, e cila është pak a shumë një zgjerim JS që lejon përdorimin e HTML brenda tij. Kjo do të thotë që React kërkon zbatime më komplekse edhe për detyra të thjeshta (krahasuar me Vue.js apo edhe Angular), në fund të fundit i duhet më shumë kohë për të zhvilluar një komponent kompleks.

Pra të dy React eshte i përshtatshme për projekte të vogla dhe të mesme.

Pyetjet kryesore në realizueshmërinë e kornizave JavaScript nga kompanitë janë sa të popullarizuar jane dhe çfarë lloj perspektivash kanë. Përzgjedhja e një kornize që vdes mund të jetë fatale për një kompani, dhe ka shumë të ngjarë të çojë në humbjen e një shume të madhe parash dhe kohe. Popullariteti ndikon shumë në mbështetjen e komunitetit, dhe kështu në efikasitetin e zhvillimit të aplikacionit. Qëllimi i kësaj pjese është të shqyrtojë popullaritetin aktual të kornizave dhe të vlerësojë se sa e garantuar është vazhdimësia e procesit të zhvillimit në secilin rast.

Me poshte do te bazohemi ne rezultatet e shfaqura ne HotFramewors. HotFrameworks është një faqe në internet që jep rezultate relative të kornizave, bazuar në yjet e tyre të GitHub dhe numrin e stack Overflow të pyetjeve të shënuara në korniza. Rezultatet normalizohen në shkallën 0-100.



.

Fig 3. HotFrameworks vlersimet mbi popullaritetin e Frameworkeve.

Sipas matjeve të ilustruara në Figurën 3, React (ngjyra blu) ka pasur rritjen më të sheshtë dhe më të ngadaltë gjatë këtij viti. Vue (ngjyra e gjelbër) ka një ngritje të pjerrët pastaj te sheshtë, por përveç kësaj, ka matjet më të mëdha për periudhën më të gjatë.

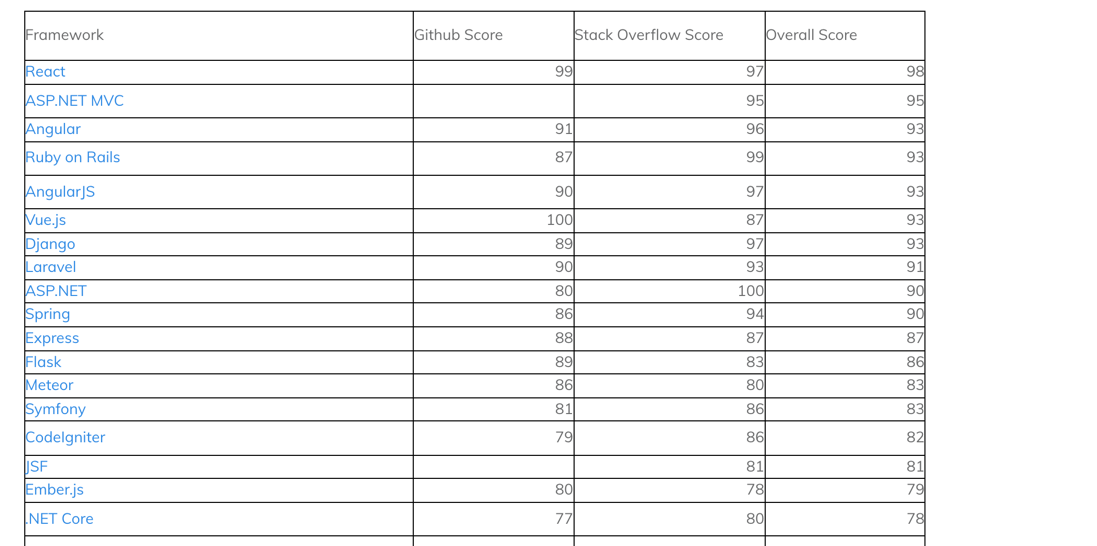


Fig 4. Piket ne Github, Stack Overflow dhe Overall te frameworkeve.

Siç është e qartë bazuar në yjet e GitHub dhe pyetjet e Stack Overflow, React aktualisht ka pjesën më të madhe të popullaritetit.

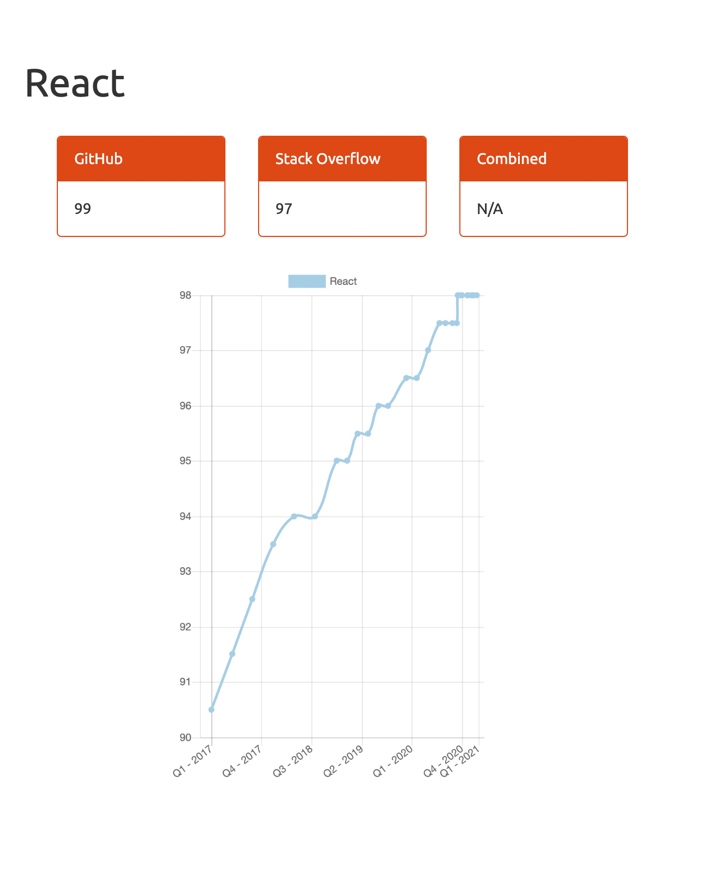


Fig 5. Rezultati nga HotFrameworks për React.

React ka qenë një lider për popullarite midis Framework-eve JavaScript. Ai zë vendin e parë me 48,718 përdorues, duke u percjelle nga Vue.js qe është korniza e dytë më e popullarizuar JavaScript me gjysmën e përdouresve - 21,575, siç raportohet nga Node Package Manager.

Sa I perket pytjeve ne Stack Overflow React është një fitues i qartë në këtë kategori. Komuniteti i tij i madh ka shpërnda shumë tutoriale, kurse online, artikuj dhe 3+ herë më shumë pyetje në Stackoverflow.

Gjithashtu inxhinierët e Facebook po punojnë vazhdimisht në mirëmbajtjen e React, duke e përmirësuar atë dhe duke investuar në të, gjë që e bën React mjetin me rritjen më të shpejtë në botën e developerave.

Kemi paraqitur gjithashtu edhe një shembull të thjeshtë i implementimit të login-it për React:

****

Fig 5. **Login implementation example for React.**

## 3.5 Performancat

Ekzistojnë dy metrika kryesore që përcaktojnë shpejtësinë e një aplikacioni: (start-up time) koha e fillimit dhe (runtime performance) performanca e ekzekutimit.

React ka madhësi të vogla të paketave që përshpejton ngarkesën fillestare 32,5 / 101,2 KB të pakompresuara.

React ka performancë te shkëlqyeshme, e sidomos ja vlene te ceket që është pak më i shpejtë në (runtime performance) kohën e ekzekutimit.

Shpejtësia mund të ndryshojë shumë në varësi të madhësisë së aplikacionit dhe përpjekjeve për optimizim.

React ka :

* server-side rendering (SSR);
* tree shaking;
* bundling.

**3.5.1 Startup performanca**

Startup performanca është koha e kaluar e një SPA (Singel-page-applications) për të ngarkuar dhe inicializuar për herë të parë. Përveç madhësisë së përmbajtjes së aplikacionit që shkarkohet, gjithashtu përpunimi dhe renderimi kërkon kohë.

Figura më poshtë tregon tryezën e referimit të kornizave JavaScript nga Stefan Krause.

Ai jep një referencë treguese për Startup performanca për secilën kornizë. Ne table kemi React-redux-v16.4.1+0.4.4- keyed dhe React-redux-v16.4.1+3.7.2- keyed.

Në rreshtin e parë është paraqitur interaktiviteti i vazhdueshem në milisekonda + - devijimi standard për React. Në rreshtin e dytë janë raportet e kohës së nisjes së skriptës dhe milisekondat e nevojshem për të analizuar / përpiluar / vlerësuar të gjitha skriptat e faqes.

Në rreshtin e tretë eshte treguar kostoja kryesore e punes me thread, pra koha totale e kaluar duke bërë punën në threadin kryesor; përfshin stilin / paraqitjen / etj.

Ndresa në rreshtin e katërt është paraqitur pesha totale e bajtit, kostoja e transferimit të rrjetit (pas-kompresimit) e të gjitha burimeve të ngarkuara në faqe.

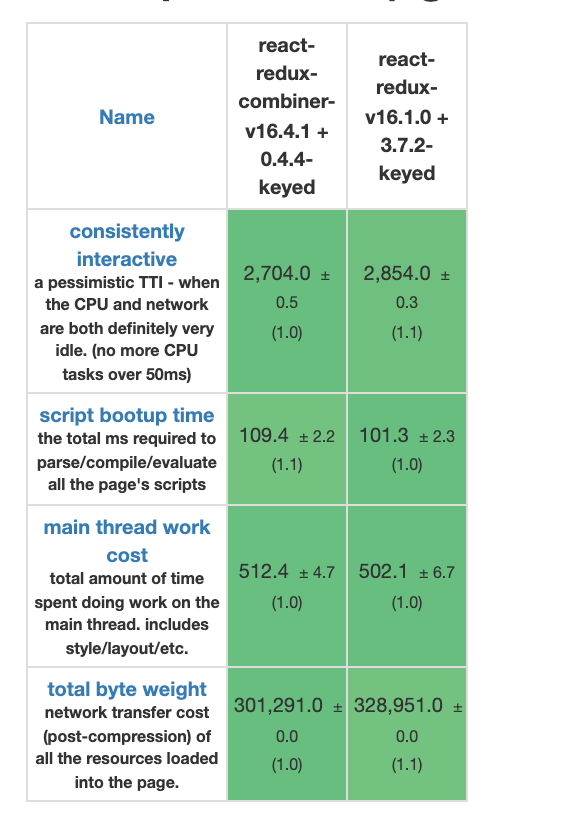


Fig 10. Keyed results for React in startup time.

Cka duhet të merret në konsideratë është mënyra se si ndryshonë startup time-i kur rritet aplikacioni, cilët janë faktorët më të mëdhenj në secilën kornizë në lidhje me këtë, dhe cilat mjete ofrojnë ato.

Në rastin e React, startup time zakonisht varet shumë nga ajo që biblioteka përfshihen në aplikim dhe mënyra se si kjo bibliotekë e trajtojnë inicimin. React-router, për shembull, lejon ndarjen e përmbajtjes JavaScript në copa, për t'u marrë nga serveri veçmas me çdo ndryshim të faqes. Kjo zvogëlon kohën fillestare të ngarkimit, pasi do të shkarkohet një herë vetëm një pjesë(cope) e pastaj tjetra.

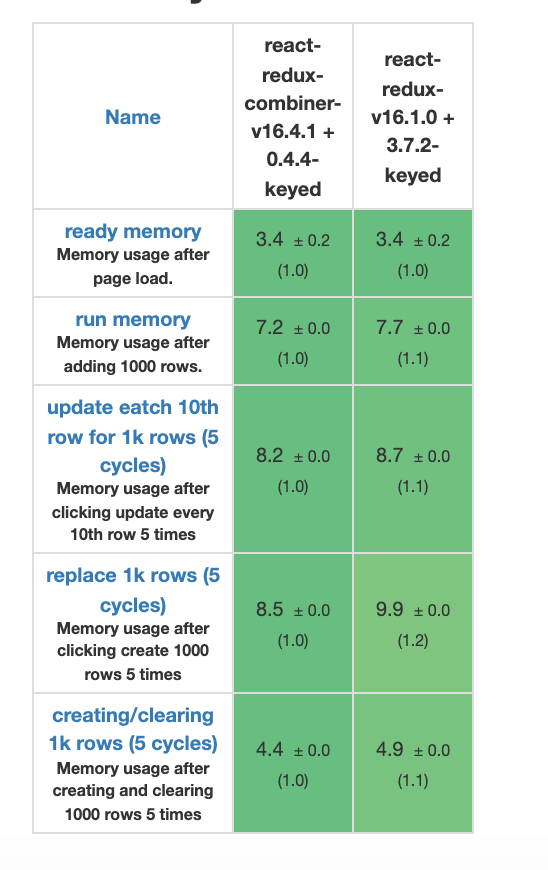
**3.5.2 Scripta dhe renderet**

React ka një mënyrë unike për krijimin e DOM (Document object model), e kemi permendur të ashtuquajturin DOM virtual. Ta themi më thjesht, DOM-i virtual është versioni i vetë Reactit ,i cila mund të aksesohet dhe modifikohet shumë më shpejt se ai i zakonshmi në memorjen e shfletuesit të klientit. Kjo natyrisht na jep shumë benefite në fushen e performancës se skriptes.

Në praktikë, DOM virtual punon në mënyrë që React të mbajë dy raste të të njëjtit element, një për DOM të vërtetë dhe një për atë virtual. Të gjitha ndryshimet që janë të nevojshme për t’u bërë, së pari bëhen në versionin virtual. Kur përmbajtja e elementit nuk ndryshohet më dhe dihet pra ndryshimi i përmbajtjes fillestare dhe asaj të fundit, kryhet operacioni i fundit: futja e tij në DOM të vërtetë. Avantazhi i DOM virtual është performanca e tij shumë më e shpejtë.

Në figuren me poshte gjeni grafiku qe përfshin kohëzgjatjen e matur të detyrave të ndryshme që lidhen me manipulimin e DOM, për React dhe Vue. Numrat e paraqitur janë kohëzgjatja në milisekonda. Një vlerë mesatare gjeometrike është e dukshme në rreshtin e fundit. Ngjyra e gjelbert e errët do të thotë përpunim më i shpejtë.

Në React, kur gjendja e një komponenti ndryshon, ajo shkakton ri-pasqyrimin e të gjithë nën-pemës së komponentit, duke filluar nga komponenti rrënjë. Për të shmangur riprodhimet e panevojshme te komponenteve femijë, duhet të përdorni *PureComponent* ose të implementoni *shouldComponentUpdate* sa herë që mundeni. Ju gjithashtu mund të duhet të përdorni struktura të pandryshueshme të të dhënave për t'i bërë ndryshimet e gjendjes tuaj më të optimizuara. Sidoqoftë, në raste të caktuara mund të mos jemi në gjendje të mbështetemi në optimizime të tilla sepse *PureComponent* / *shouldComponentUpdate* supozon se të gjithë rezultatet për renderimin e nën-pemës përcaktohen nga rekuizitat e komponentës aktual. Nëse nuk është kështu, atëherë optimizime të tilla mund të çojnë në gjendje jo konsistente të DOM.



### Fig 12. Memory allocation in MBs ± standard deviation for React.

Ne fig me poshte do ti paraqesim gjithashtu dhe “non keyed results”.

Implementimet pa key lejohen të ripërdorin elementët ekzistues dom. Si pasojë, futja ose fshirja e një elementi në grupin e të dhënave mund të shtohet pas ose të fshijë rreshtin e fundit të tabelës dhe të përditësojë përmbajtjen e të gjithë elementëve pas indeksit të futjes ose fshirjes. Kjo mund të funksionojë më mirë, por mund të shkaktojë probleme nëse gjendja dom modifikohet nga jashtë.



Fig 13. React non keyed results.

# 

Në React, gjithçka është vetëm JavaScript. Jo vetëm që strukturat HTML shprehen përmes JSX, tendencat e fundit gjithashtu tentojnë të vendosin menaxhimin e CSS brenda JavaScript gjithashtu. Kjo qasje ka përfitimet e veta, por gjithashtu vjen me shkëmbime të ndryshme që mund të mos duken të vlefshme për çdo developer.

# 4. METODOLGJIA

Për të zhvilluar aplikacionin e lartë përmendur duhet të përdoret një metodologji e duhur e cila do të ofron rezultate të mira, kosto të ulët dhe të shpenzojmë kohë më pak në zhvillim. Fillimisht kemi treguar përparësitë dhe mangësitë e Framework-ut Vue. Është bërë një shqyrtim për aplikacionet në përgjithësi, pastaj për platformat dhe gjuhen programuese e përdorur me arsye. Për këto sqarime, të gjithë aplikacionet të lartë përmendur janë testuar. Nga këto hulumtime janë vendosur qëllimet e aplikacionit, ndërfaqja e përdoruesit dhe detajet tjera të aplikacionit. Ky aplikacion në thelb funksionon me dy aplikacione të ndara, front-end apo pjesa e aplikacioni që e shohin përdoruesit dhe back-end apo serveri. Aplikacioni tërë kohën komunikon me serverin, për të marrë të dhënat, për të bërë përditësime në bazë të dhënave, për autentifikim, etj.

# 5. REZULTATET

# 7.REFERENCAT

|  |  |
| --- | --- |
| [1]. | "MDN WEB Docs," Apr Apr 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript. |
| [2]. | "AltexSoft software r&d engineering," [Online]. Available: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/ . |
| [3]. | "MDN Web Docs," 14 Apr 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Tools\_and\_testing/Client-side\_JavaScript\_frameworks . |
| [4]. | "VUE.js," 2021. [Online]. Available: https://vuejs.org/. |
| [5]. | "GitHub Vue.js," 2021. [Online]. Available: https://github.com/vuejs/vue. |
| [6]. | "Vue.js," [Online]. Available: https://vuejs.org/v2/guide/#Getting-Started. |
| [7]. | "Vue Components," [Online]. Available: https://vuejs.org/v2/guide/components.html. |
| [8]. | "Enter/Leave & List Transitions," [Online]. Available: https://vuejs.org/v2/guide/transitions.html. |
| [9]. | "What is Vuex?," [Online]. Available: https://vuex.vuejs.org/. |