

Programi për Shkenca Kompjuterike dhe Inxhinierisë

**APLIKACION PËR KOMUNITETIN E LEHONAVE**

Shkalla Bachelor

Eroll

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 Vue.js.

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 Vue.js duke shtjelluar më në detaje pjesen e Login dhe Navigation bar. Në rrafshin professional dhe atë personal si zhvilluesve kam pasur rezultate të mira ne lidhje me frameworkun Vue.js.

# MIRËNJOHJE/FALENDERIME

Punimi i temës së diplomës kërkon mund dhe përkushtim të madh i cili nuk mund të realizohej pa përkrahjen e njerëzve më të dashur, prandaj ju jam mirënjohese dhe i falenderoj pafund të gjithë ata që më kanë mbështetur jo vetëm tani por për gjate gjithë rrugëtimit tim studentor. Është shumë e vështirë për të gjetur fjalë të duhura për të shprehur falënderime dhe mirënjohje ndaj mentorit tim të nderuar Prof. Lavdim Menxhiqi i cili në mënyrë të vashdueshme më përcillte hap pas hapi drejt studimit të punimit me mbështetjen, inkurajimin, dhe udhëzimet e vlefshme të cilat synonin drejt ngritjes së nivelit të punimit. Një falënderim i veҫant shkon për të gjithë profesorët e tjerë të cilët kontribuan me njohurit e tyre gjatë gjithë kohes së studimit dhe koletivin e fakultetit, poashtu falënderimet shkojnë edhe për kolegët (studentët) të cilët më mbështeten me mendimet dhe sugjerimet e tyre. Dhe krejt në fund, një falënderim shumë i sinqert i kushtohet- dedikohet familjes sime të cilët më mbështeten gjatë kohës së studimeve me dashurin e tyre , durimin dhe kurajon si dhe duke më dhënë forcë për të përfunduar me sukses këtë punim.

# PËRMBAJTJA

[LISTA E FIGURAVE IV](#_Toc80039922)

[LISTA E SHKURTESAVE DHE KONCEPTEVE KRYESORE V](#_Toc80039923)

[1.HYRJE 1](#_Toc80039924)

[2.DEKLARIMI I PROBLEMIT 3](#_Toc80039925)

[3. SHQYRTIMI I LITERATURËS 4](#_Toc80039926)

[3.1 JavaScript 4](#_Toc80039927)

[3.2 Përkufizimi i një kornize JavaScript 6](#_Toc80039928)

[3.3 VUE.JS 7](#_Toc80039929)

[3.3.1 Komponentet ne Vue 10](#_Toc80039930)

[3.3.2 Transicionet ne Vue 11](#_Toc80039931)

[3.3.3 Vuex 11](#_Toc80039932)

[3.3.4 Popullariteti dhe Mbështetja 12](#_Toc80039933)

[3.3.5 Performancat 17](#_Toc80039934)

[3.3.6 Startup performanca 18](#_Toc80039935)

[3.3.7 Scripta dhe renderet 19](#_Toc80039936)

[4. METODOLGJIA 22](#_Toc80039937)

[5. REZULTATET 23](#_Toc80039938)

[5.1 Zhvillimi i nje App me Vue.js 23](#_Toc80039939)

[5.1.1 Login Module 32](#_Toc80039940)

[5.1.2 Shiriti i Navigimit 35](#_Toc80039941)

[6. DISKUTIME DHE PËRFUNDIME 39](#_Toc80039942)

[7.REFERENCAT 40](#_Toc80039943)

# LISTA E FIGURAVE

[Figure 1. Struktura HTML, CSS, JavaScrpit 5](#_Toc72428429)

[Figure 2. Si punon API 5](#_Toc72428430)

[Figure 3. Relationships between components. 7](#_Toc72428431)

[Figure 4. Components in vue. 11](#_Toc72428432)

[Figure 5. HotFrameworks vlersimet mbi popullaritetin e Vue.js 13](#_Toc72428433)

[Figure 6. Rezultati nga HotFrameworks për React. 14](#_Toc72428434)

[Figure 7. % Stack Overflow questions about Vue.js vs React. 15](#_Toc72428435)

[Figure 8. Downloads in past 6 Months. 16](#_Toc72428436)

[Figure 9**.** Login implementation example for Vue. 17](#_Toc72428437)

[Figure 10. Keyed results for Vue.js in startup time. 19](#_Toc72428438)

[Figure 11. Keyed results for Vue in various tasks of DOM manipulation. 20](#_Toc72428439)

[Figure 12. Keyed results for Vue in memory allocation. 21](#_Toc72428440)

[Figure 13. Pamje e faqes fillestare 27](#_Toc72428441)

[Figure 14. Pamje e faqes “Keshilla” 27](#_Toc72428442)

[Figure 15. Pamje e faqes “Reth Nesh” 28](#_Toc72428443)

[Figure 16. Pamje e forms per Kyqje 28](#_Toc72428444)

[Figure 17. Pamje e forms per Regjistrim 29](#_Toc72428445)

[Figure 18. Pamje e forms per Ndryshim Fjalkalimi 29](#_Toc72428446)

[Figure 19. Pamje e forms per Propozimin e ushqimeve 30](#_Toc72428447)

[Figure 20. Pamje e informatave pa filter 30](#_Toc72428448)

[Figure 21. Pamje nga faqja e te gjitha recetave te ruajtura 31](#_Toc72428449)

[Figure 22. Pamje nga faqja e te gjithe Perdoruesve 31](#_Toc72428450)

# 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.

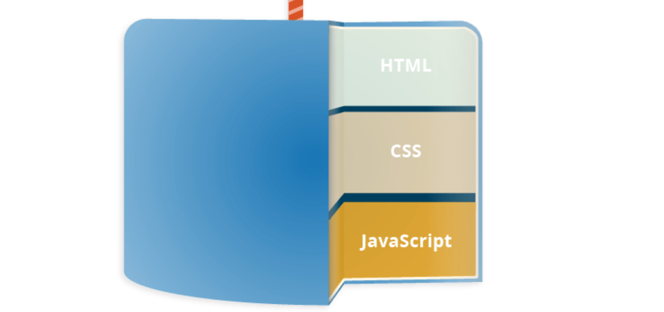


Figure 1. Struktura HTML, CSS, JavaScrpit

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.

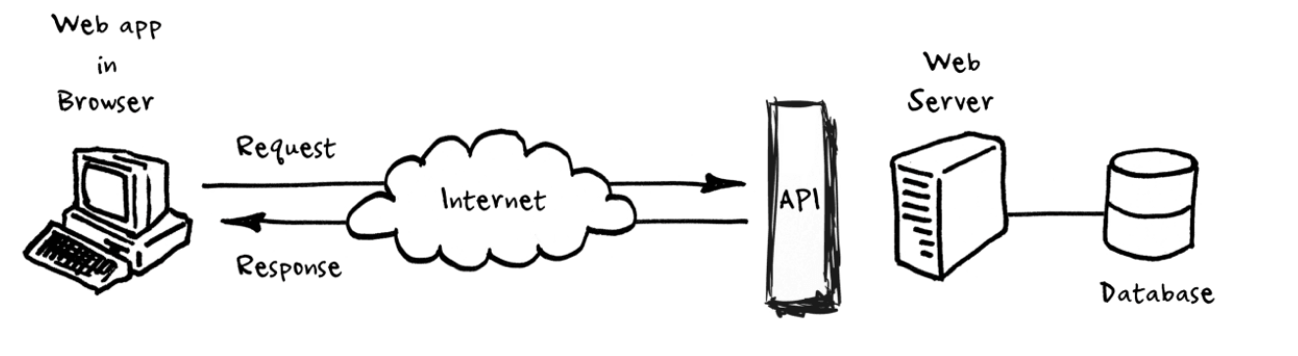


Figure 2. Si punon API

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.

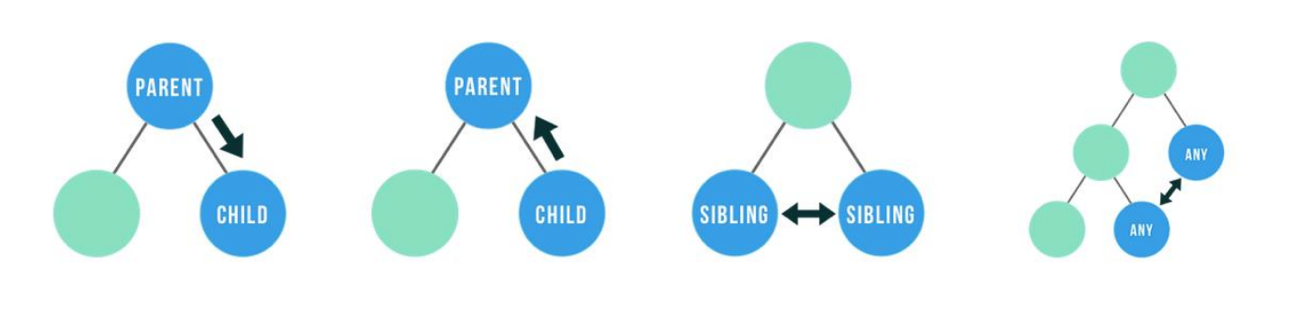


Figure 3. Relationships between components.

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 VUE.JS

Krijuar nga Evan You, Vue.js është një kornizë progresive me burim të hapur JavaScript për ndërtimin e ndërfaqeve të përdoruesve (UI) dhe aplikacioneve me një faqe. Kjo kornizë përdor "high decoupling", duke lejuar zhvilluesit të krijojnë në mënyrë progresive ndërfaqet e përdoruesit (UI).

Vue përmban në vete një arkitekturë të qëndrueshme dhe me mundësi zgjerimi në të ardhmen. Vue.js fokusohet në paraqitjen deklarative dhe përbërjen e komponentëve dhe mund të shtohet në faqet ekzistuese.

Sistemi përbërës i Vue.js është reaktiv, që do të thotë se Vue.js di të komunikojë përmes ngjarjeve asinkrone; për shembull, një përbërës fëmijë mund të komunikojë me përbërësin e tij prind përmes ngjarjeve. Me Vue.js nuk ka fërkime me bibliotekat ose burimet e tjera, me fjalë të tjera, ne mund të përdorim mjetin me të cilin jemi më të qetë. Për shembull, ne mund të shkruajmë vetëm HTML dhe JavaScript ose nëse duam mund të shtojmë CSS, JSX ose TypeScript. [4]

Është framework-u i 3-të me i përdorur në botë sa i perket front-endit. Pra është framework i bazuar ne arkitekturen model-view-viewmodel. Vue plotësohet me disa librari të tjera për ti arritur rezultatet e dëshiruara.

Aktualisht ka 382 kontribues te cilet kan kontribu ne codebasing e vue js. [5]

Ka mënyra të ndryshme për të përfshirë Vue.js në projektin tuaj:

* Shkarkimi i skedarit Vue.js dhe përfshirja e tij me etiketën <script> direkt në skedarin HTML
* Përdorimi i CDN duke përfshirë etiketën <script> në skedarin HTML
* Instalimi i tij duke përdorur Node Package Manager (NPM)
* Përdorimi i Vue-cli për të vendosur projektin tuaj

Vue nuk mbështet IE8 dhe më poshtë, sepse ai përdor veçoritë ECMAScript 5 që nuk mund të shndërrohen në IE8. [6]

ECMAScript është gjuhë e bazuar në objekte, këto objekte komunikojne mes vete, pra përcakton një koleksion të objekteve të ndërtuara që përmbledhin përkufizimin e ECMAScript subjekteve

Në thelb Vue.js është një sistem që na mundëson të japim në mënyrë deklarative të dhëna në DOM. Vue përdor një sintaksë template të bazuar në HTML që lejon paraqitjen e DOM në mënyrë deklarative me të dhënat e instancës Vue duke përdorur sintaksën e drejtpërdrejtë të shabllonit:

HTML

<div id="app">

{{ message }}

</div>

JS

var app = new Vue({

el: '#app',

data: {

message: 'Hello UBT!'

}

})



Mund t'i lidhim të dhënat jo vetëm me tekstin dhe atributet, por edhe me strukturën e DOM. Për më tepër, Vue gjithashtu ofron një sistem të fuqishëm të efektit të tranzicionit që mund të zbatojë automatikisht efektet e tranzicionit kur elementet futen / azhurnohen / hiqen nga Vue.

Ekzistojnë mjaft direktiva të tjera, secila me funksionalitetin e vet të veçantë. Për shembull, direktiva “v-for” mund të përdoret për të shfaqur një listë të artikujve duke përdorur të dhënat nga një Array:

HTML

<div id="lista">

<ol>

<li v-for="todo in todos">

{{ todo.text }}

</li>

</ol>

</div>

JS

var app4 = new Vue({

el: '#lista',

data: {

todos: [

{ text: 'UBT' },

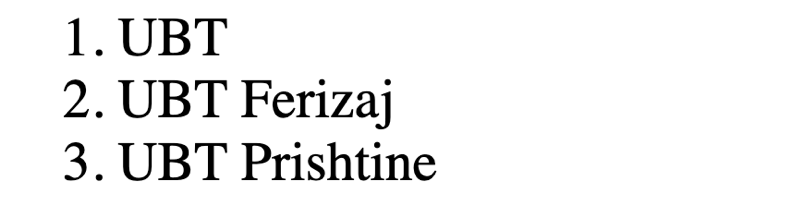
{ text: 'UBT Ferizaj' },

{ text: 'UBT Prishtine' }

]

}

})



Për të lejuar përdoruesit të ndërveprojnë me aplikacionin, ne mund të përdorim direktivën “v-on” për të bashkangjitur dëgjues të ngjarjeve që thirren metoda në instancat tona Vue:

HTML

<div id="reverse">

<p>{{ message }}</p>

<button v-on:click="reverseMessage">Reverse Message</button>

</div>

JS

var app5 = new Vue({

el: '#reverse',

data: {

message: 'Hello UBT!'

},

methods: {

reverseMessage: function () {

this.message = this.message.split('').reverse().join('')

}

}

})

### 3.3.1 Komponentet ne Vue

Sistemi i komponentëve është një koncept tjetër i rëndësishëm në Vue, ato janë një nga veçoritë më të fuqishme të Vue sepse është një abstraksion që na lejon të ndërtojmë aplikacione në shkallë të gjerë të përbërë nga përbërës të vegjël, të pavarur dhe shpesh të ripërdorshëm. Në Vue, një komponent është në thelb një instance e Vue me opsione të paracaktuara. Ato pranojnë të njëjtat opcione siç janë të dhënat, computed, watchers, metodat dhe lifecycle hooks. [7]

Nëse e mendojmë, pothuajse çdo lloj ndërfaqe e aplikacionit mund të abstragohet në një pemë të përbërësve:

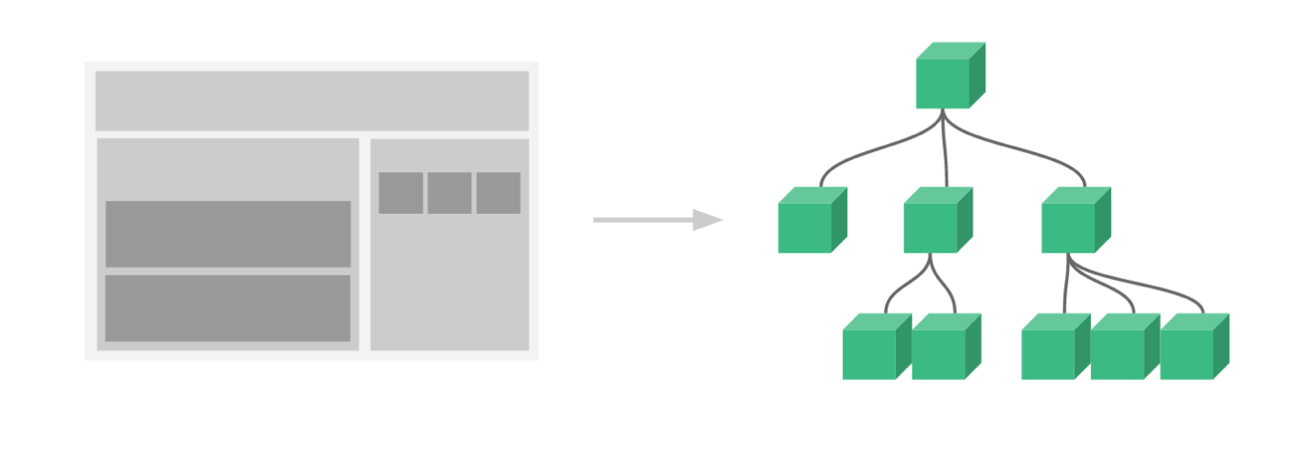


Figure 4. Components in vue.

### 3.3.2 Transicionet ne Vue

Vue përfshin tools për aplikimin automatik të klasave për tranzicionet dhe animacionet CSS që integrojnë third-part librarite,si Animate.css e cila përdorë JavaScript për të manipuluar drejtpërdrejt DOM ku gjatë hooks të tranzicionit integrohen libraritë e animacionit të JavaScript të palëve të treta, siq është si Velocity.js. Kur nuk zbulohen transicionet ose animacionet CSS e as nuk ofrohen Javascript hooks, operacionet e DOM për futje ose largim ekzekutohen menjëherë në kornizën tjetër. [8]

### 3.3.3 Vuex

Vuex është një model i menaxhimit të gjendjes, sa i perket gjendjës që e udhëheqë aplikacionin, pamjen apo ndërfaqjen e përdoruesit qe është hartim deklarativ i gjendjes, reagimin ndaj inputeve të përdoruesve nga ndërfaqja e përdoruesit është një aplikacion i pavarur. Ështe në shërbim të të gjitha komponentëve në një aplikacion, me rregulla që sigurojnë që gjendjet mund të ndryshoj vetëm në një mënyrë të parashikueshme. [9]

Vuex është një model menaxhimi + bibliotekë për aplikacionet Vue.js. Ai shërben si një dyqan i centralizuar për të gjithë përbërësit në një aplikacion, me rregulla që sigurojnë që state mund të shndërrohet vetëm në një mënyrë të parashikueshme. Ai gjithashtu integron me shtesën zyrtare të devotols të Vue (hap dritare të re) për të siguruar karakteristika të përparuara të tilla si korrigjimi i zero-config time-travel dhe state snapshot export / import. Për t’a thjeshtuar vërtet se çfarë është Vuex, në thelb, ju mund ta mendoni atë si një objekt gjigant Javascript që tregon gjendjen e aplikimit tuaj. Çdo komponent ka qasje në këtë gjendje dhe mund ta ndryshojë atë duke përdorur veprime të paracaktuara. Në këtë mënyrë ju gjithmonë do të dini se si aplikacioni juaj përfundoi në një gjendje të caktuar.

Nëse aplikacioni juaj është i vogël, është ndoshta një zgjedhje më e mirë për t'iu përmbajtur modelit të vanilla Vue props dhe data. Kurdoherë që shtoni Vuex në aplikacionin tuaj, ju shtoni një tërësi të kodit boilerplate. Pra, nëse aplikimi juaj është i vogël, fitimi nuk është aq i madh.

Vuex bën punën e abstragimit të gjendjes së një moduli të tërë dhe mbajtjen e tij në një dyqan të centralizuar, i cili ndjek modelin e menaxhimit të state dhe komunikon me përbërësit tuaj në përputhje me rrethanat. Vuex gjithashtu lehtëson testimin e njësive në të gjithë aplikacionin tuaj, i cili mund të jetë i dobishëm nëse aplikacioni juaj rritet në kompleksitet, duke e bërë gjendjen tuaj të vështirë për t'u menaxhuar. Pra, është një mjet i fuqishëm që mund të jetë i dobishëm ose jo, varësisht nga rasti i përdorimit.

• **Virtual DOM** – është një kopje e lehtë e DOM-it origjinal. I ka të gjitha atributët e DOM-it real. Kur bëhet ndonjë përditësim, i gjithë DOM-i virtual përditësohet. Është shumë më i shpejtë se DOM-i real sepse i përditëson vetëm ndryshimet e bëra dhe jo të gjithë DOM-in real, duke bërë krahasimin në mes të DOM-it real dhe Virtual DOM-it.. Termi Virtual DOM ka tendencë të shtojë misterin e asaj që është koncepti në të vërtetë. Në fakt është vetëm një objekt në Javascript. Procesi ne tërsi quhet reconciliation(pajtim). Më thjeshte Virual DOM është objekt që përfaqëson ndërfaqen e përdoruesit. [10]

.

### 3.3.4 Popullariteti dhe Mbështetja

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

Kemi thënë që Vue përdorë DOM virtual.

Vue.js përdorë shabllone me interpretim deklarative, e cila është pak a shumë një zgjerim JS që lejon përdorimin e HTML brenda tij.

Vue.js është në nivelit më të lartë dhe ka një pikë shumë më të ulët hyrëse pasi Vue.js nuk kërkon që të kuptojë thellësisht JS Core për të implementuar komponent. Ështe i përshtatshme për projekte të vogla dhe të mesme.

Pyetjet kryesore në realizueshmërinë e kornizave JavaScript nga kompanitë janë sa e popullarizuar është dhe çfarë lloj perspektivash ka. 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 kornizaes dhe të vlerësojë se sa e garantuar është vazhdimësia e procesit të zhvillimit në secilin rast.

Një metodë matëse e përdorur në këtë seksion janë të dhenat e marrura nga HotFrameworks. 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. [11]

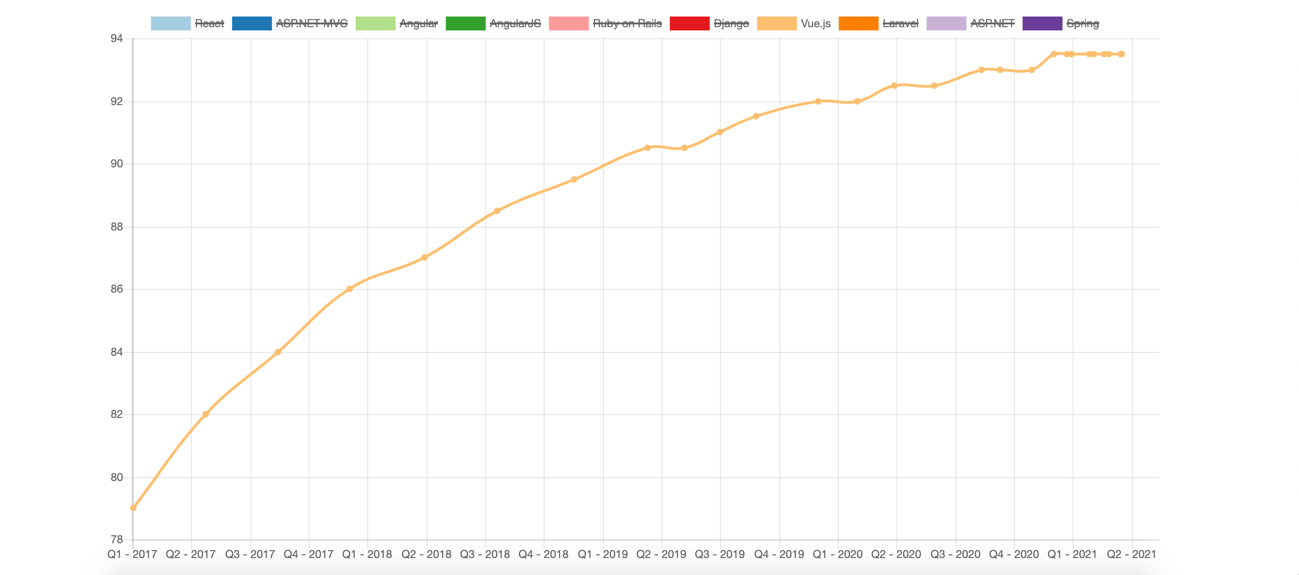


Figure 5. HotFrameworks vlersimet mbi popullaritetin e Vue.js

Sipas matjeve të ilustruara në Figurën 5, Vue ka një ngritje të pjerrët pastaj te sheshtë, por përveç kësaj, ka matjet më të mëdha për periudhën më të gjatë. [11]



Figure 6. Rezultati nga HotFrameworks për React.

Vue.js është korniza e dytë më e popullarizuar JavaScript me numer të përdouresve - 21,575, siç raportohet nga Node Package Manager. Megjithëse, nëse hedhim një vështrim në statistikat në fund të vitit 2018, do të shohim se personat qe perdorinin Vue ishin trefishuar.

Stack Overflow jep të dhënat e mëposhtme bazuar në numrin e pyetjeve të bëra nga zhvilluesit që aktualisht punojnë brenda një platforme të veçantë:

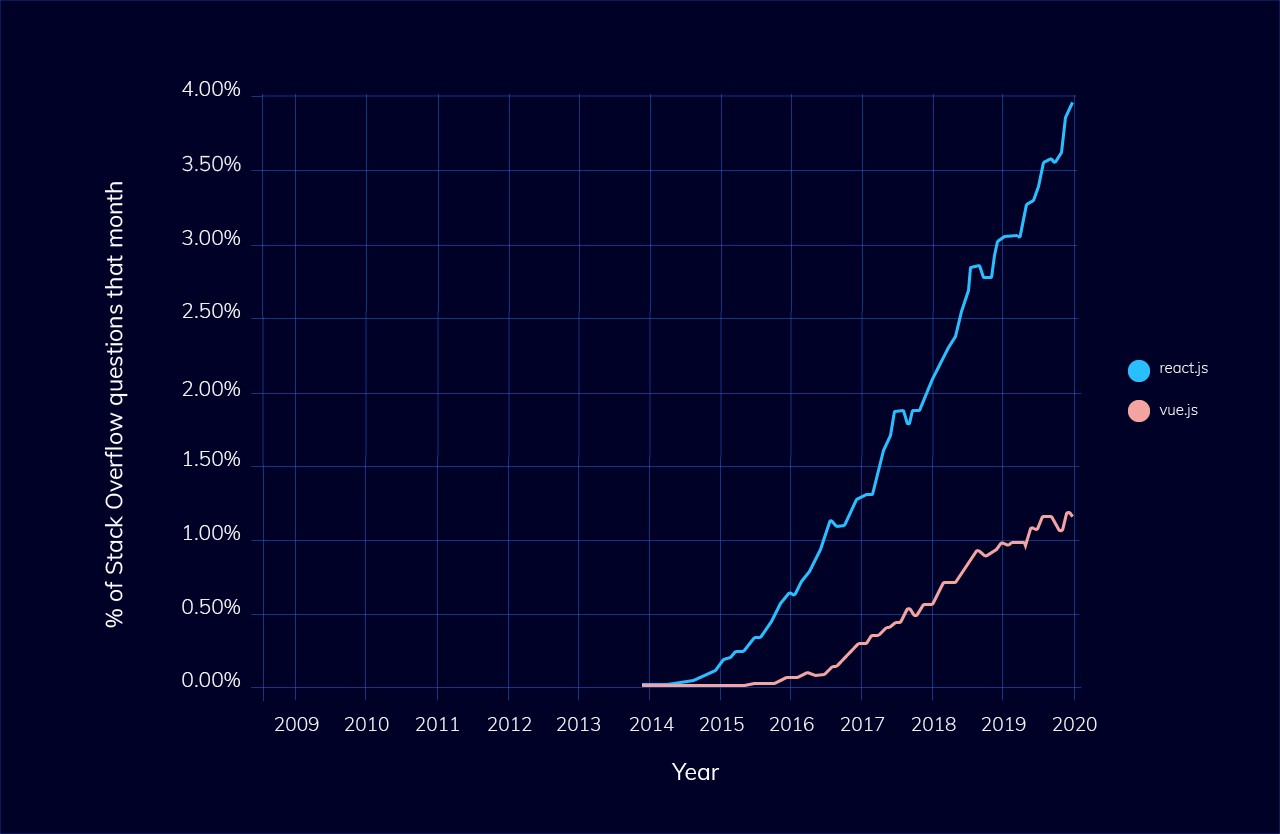


Figure 7. % Stack Overflow questions about Vue.js vs React.

Në këto raste nese Vue e krahasojmë me React atëhere 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.

Megjithëse mund të shohim se numri i pyetjeve në lidhje me Vue.js po rritet, por pa dyshim që React është ende në vendin e parë të podiumit.

Ndresa sipas edicionit të katërt të JavaScript Rising Stars që vlerëson numrin e yjeve të shtuar në Github gjatë çdo viti, Vue ka zënë vendin e parë në 2019.

Duke u mbështetur në statistikat e Github, mund të bëhet një përfundim se sa i popullarizuar është Vue.js midis komunitetit të tij me burim të hapur, madje edhe pa mbështetjen nga një korporatë të nivelit të lartë. [12]

Në anen tjeter 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.

Për fat të keq, komuniteti i React është pak më i fragmentuar se Vue dhe shpesh duhet të kaloni më shumë kohë duke kërkuar zgjidhje për çështje të zakonshme.

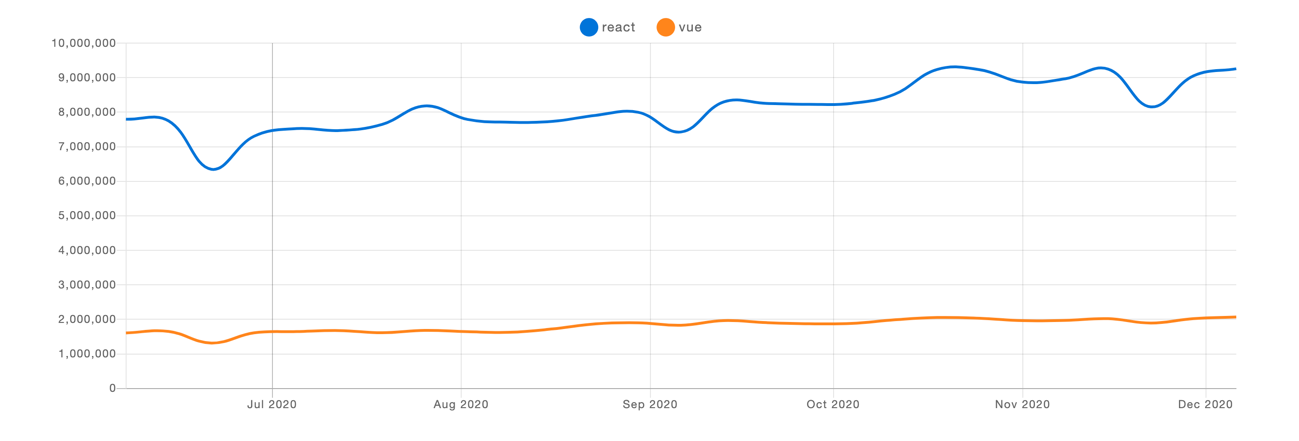


Figure 8. Downloads in past 6 Months.

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

Figure 9**. Login implementation example for Vue.**

### 3.3.5 Performancat

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

Vue ka madhësi të vogla të paketave që përshpejton ngarkesën fillestare (31 KB / 84,4 KB të pakompresuara).

Ka performancë të shkëlqyeshme. Vue është më i mirë në (memory allocation) alokimin e kujtesës dhe (startup) kohën e fillimit.

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

Vue ka:

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

### 3.3.6 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 10. më poshtë tregon tryezën e referimit të kornizave JavaScript nga Stefan Krause. Ai jep një referencë treguese për Startup performanca për secilën kornizë. Ne tablë kemi Vue v2.5.16- keyed. Në rreshtin e parë është paraqitur interaktiviteti i vazhdueshem në milisekonda + - devijimi standard. 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.

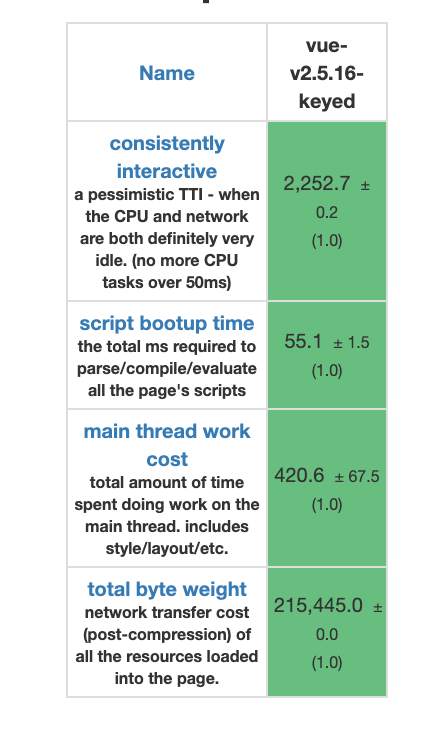


Figure 10. Keyed results for Vue.js in startup time.

Cka duhet të merret në konsideratë është mënyra se si ndryshonë startup time-i kur rritet aplikacioni.

Vue referohet si korniza më e shpejtë në dispozicion e JavaScript. Në versionet më të reja të Vue.js, ai zakonisht ka një përparësi të vogël, por karakteristike në startup time. [13]

### 3.3.7 Scripta dhe renderet

Runtime performance e Vue.js është shumë e optimizuar edhe në këtë rast. Vue shfrytezon virtual DOM-in, në një menyre më te avancuar. Një karakterisike e vogël dhe interesante është menyra e funksionimit te Vue kur gjendja përbërëse ndryshon. Vue trajton këtë pjesë pak më me mençuri , duke re-renderuar saktësisht vetëm përbërësit e nevojshëm.

Në figuren 8. gjeni grafiku qe përfshin kohëzgjatjen e matur të detyrave të ndryshme që lidhen me manipulimin e DOM, për 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.

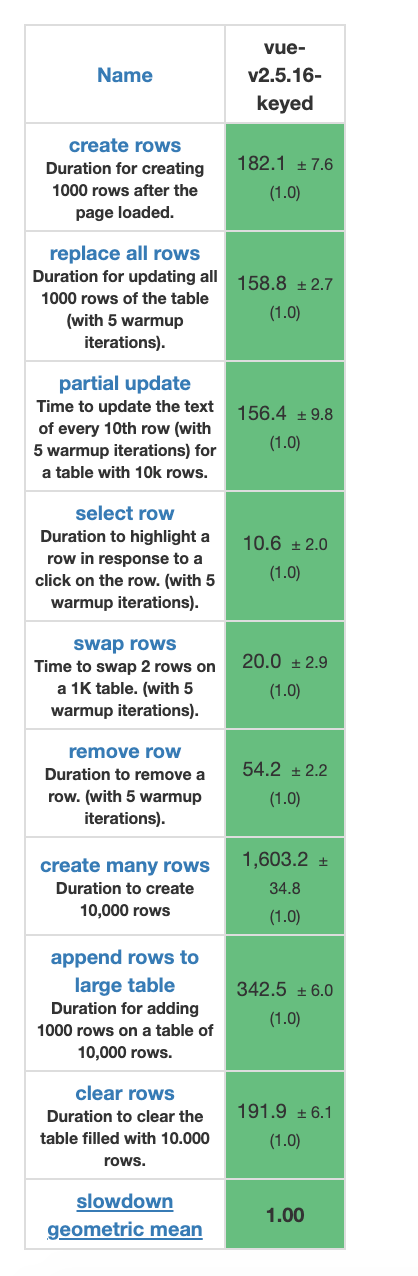


Figure 11. Keyed results for Vue in various tasks of DOM manipulation.

Edhe në fig 8. tregohet se performanca e Vue është, në fakt, është më e shpejta. Sidoqoftë, edhe në këtë rast ndryshimet midis performancave te framework-eve janë mjaft të vogla. Edhe në rastin e krijimit të 10,000 rreshtave, ndryshimi psh midis React dhe Vue, është vetëm më pak se gjysma e sekondës. Ekzistojnë disa stile të ndryshme të trajtimit të DOM midis kornizave. Dallime në performancë janë mjaft të vogla, por Vue ka fushat e veta fituese. Një krahasim i rreptë i performancës së kornizës në skriptimin dhe interpretimin ofron disa udhëzime, por ndryshimet bëhen domethënëse vetëm në rastin e paraqitjes së strukturave shumë të mëdha të të dhënave. Prandaj, per mendimin time ndryshimet në performancë nuk duhet të theksohen për vlerësime të përgjithshëm.

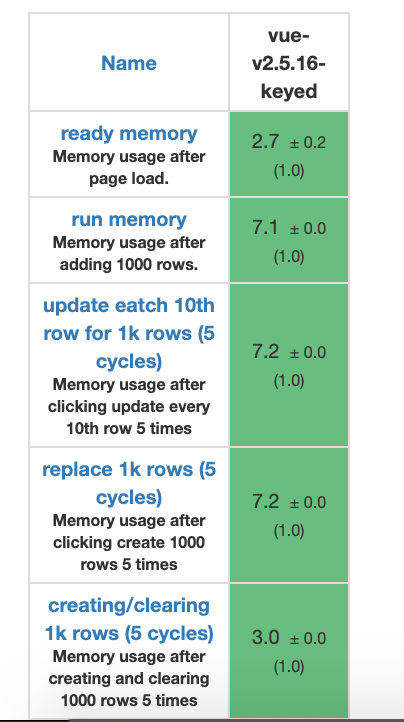


Figure 12. Keyed results for Vue in memory allocation.

# 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

Kur një zhvillues zgjedhë një teknologji, duhet të hulumtoj për të mirat dhe të këqijat e asaj teknologjie.

Për t’iu pergjigjur pyetjes se pse duhet të zgjedhim Vue para frameworkeve të tjerë, Vue shfaq performancë të ngjashme me dy prej Frameworkeve me te popullarizuara React dhe Angular. Vue mund të mësohet dhe të përdoret më lehtë dhe kjo mund të quhet si përparsi. Pra është gradualisht i adaptueshëm. Kjo do të thotë që mund ta lësh atë në projekte të trashëguara dhe të fillosh ta përdorësh pa rindërtuar të gjithë aplikacionin nga themeli. Vue.js përdor teknologji thelbësore të uebit me të cilat zhvilluesit e faqeve tashmë janë njohur: HTML, CSS dhe JavaScript të thjeshtë (ES5). Përdorimi i mjeteve të ndërtimit nuk është i nevojshëm: Ju mund të përfshini Vue vetëm me një etiketë <script>.

Ndresa për të përshkruar rrjedhën e të dhënave midis përbërësve në një aplikacion Vue.js,

të dhënat u kalohen komponentëve fëmijë nga një përbërës i caktuar prind duke përdorur një prop ose një atribut të personalizuar që bëhet një property në instancen e përbërësit fëmijë. Kur përbërësi prind përditëson një vlerë prop, ajo azhurnohet automatikisht në përbërësin fëmijë. Mutacioni i një property brenda një përbërësi fëmijë nuk duhet të bëhet. Gjithashtu, nuk ndikon në përbërësin prind (përveç nëse është një objekt ose array).

Komponenti fëmijë mund t'i komunikojë prindit përmes një eventi. Prindi mund të caktojë një mbajtës për çdo event të emetuar nga instanca e përbërësit të fëmijës dhe të dhënat mund t'i kalojnë prindit. Komponenti fëmijë mund të lëshojë një event të veçantë për azhurnimin e rekuizitave që i janë kaluar.

Për ta përmbledhur, Vue.js është në pozitë të barabartë me kornizat e tjera kur ndërton aplikacione në shkallë të gjerë, dhe shkëlqen kur duhet të punosh me aplikacione të trashëguara ose të futesh brenda dhe të bësh diçka shpejt.

## 5.1 Zhvillimi i nje App me Vue.js

Sistemi ynë do të zhvillohet kryesisht nga Vue.js korniza e Javascript.

Front End do te kete : HTML, CSS, Bootstrap (librari e CSS) – gjuhë stiluese VueJS Vuex, nuxt.js .

Në këtë kapitull do të shqyrtojmë strukturën e kodit të aplikacionit dhe mënyrën e funksionimit të moduleve të aplikacionit. Pasi që shpjegimi dhe analiza i të gjithë kodit do të jetë shumë e gjatë, do të shqyrtojmë vetëm modulin për login-autentifikim dhe modulin e navigation-bar gjithashtu the definimin e kërkesave.

Kërkesat për një sistem janë përshkrimet e shërbimeve që një sistem duhet ti siguroj dhe kufizimet në veprimet e tij. Këto kërkesa reflektojnë nevojat e klientëve për një sistem që shërben për një qëllim të caktuar psh. kontrollimin e një pajisje, bërjen e një porosie ose me gjetë informacione. Procesi i gjetjes, analizimit, dokumentimit dhe kontrollimit të këtyre shërbimeve dhe kufizimeve quhen kërkesat inxhinierike. Termi kërkesë nuk është përdorur rregullisht në industrinë e softuerëve. Në disa raste, një kërkesë është thjeshtë një shprehje abstrakte e një shërbimi, që një sistem duhet të siguroj ose të kufizoj në një sistem. Shpesh kërkesat e sistemeve softuerike klasifikohen si funksionale ose jo-funksionale. Kërkesat funksionale janë deklarimi i sherbimeve që sistemi duhet ti siguroj, si do të reagoj sistemi në hyrje të caktuara dhe si do të veproj sistemi në situata të caktuara. Në disa raste, kërkesat funksionale munden në mënyrë eksplicite të tregojnë se çfarë nuk duhet të bëjë sistemi. Kërkesat jo-funksionale janë kufizimet në shërbimet ose funksionet e ofruara nga sistemi. Këto përfshijnë kufizime kohore, kufizime në procesin e zhvillimit, dhe kufizimet e imponuara nga standarde të ndryshme. Kërkesat jo-funksionale shpeshë aplikohen për sistemin në tërësi, sesa për disa veçori të caktuara të sistemit

Definimi i kërkesat për aplikacionin tonë në bazë të llojit të përdoruesit. Aplikacioni ynë ka tre lloje të përdoruesve: vizitor, antarin dhe mirembajtesin.

**Vizitori** - Kërkesat funksionale: Vizitorit duhet t’i ofrohet mundësia e informimit përmes galerisë së ushqimeve, recetave të ushqimeve , mundësia për regjistrim dhe kyçje në sistem, mundesim per shtim të recetave të reja, mundësia për çasje nga pajisjet mobile.

**Antari** - Kërkesat funksionale: Përveç shërbimeve si vizitor, antari do ti ofrohet mundësia: për të marr dhe per të dhëne keshilla, për të shikuar dhe shtuar receta të ushqimeve, për ta shikuar prejardhjen dhe informata tjera rreth një ushqimi te caktuar.

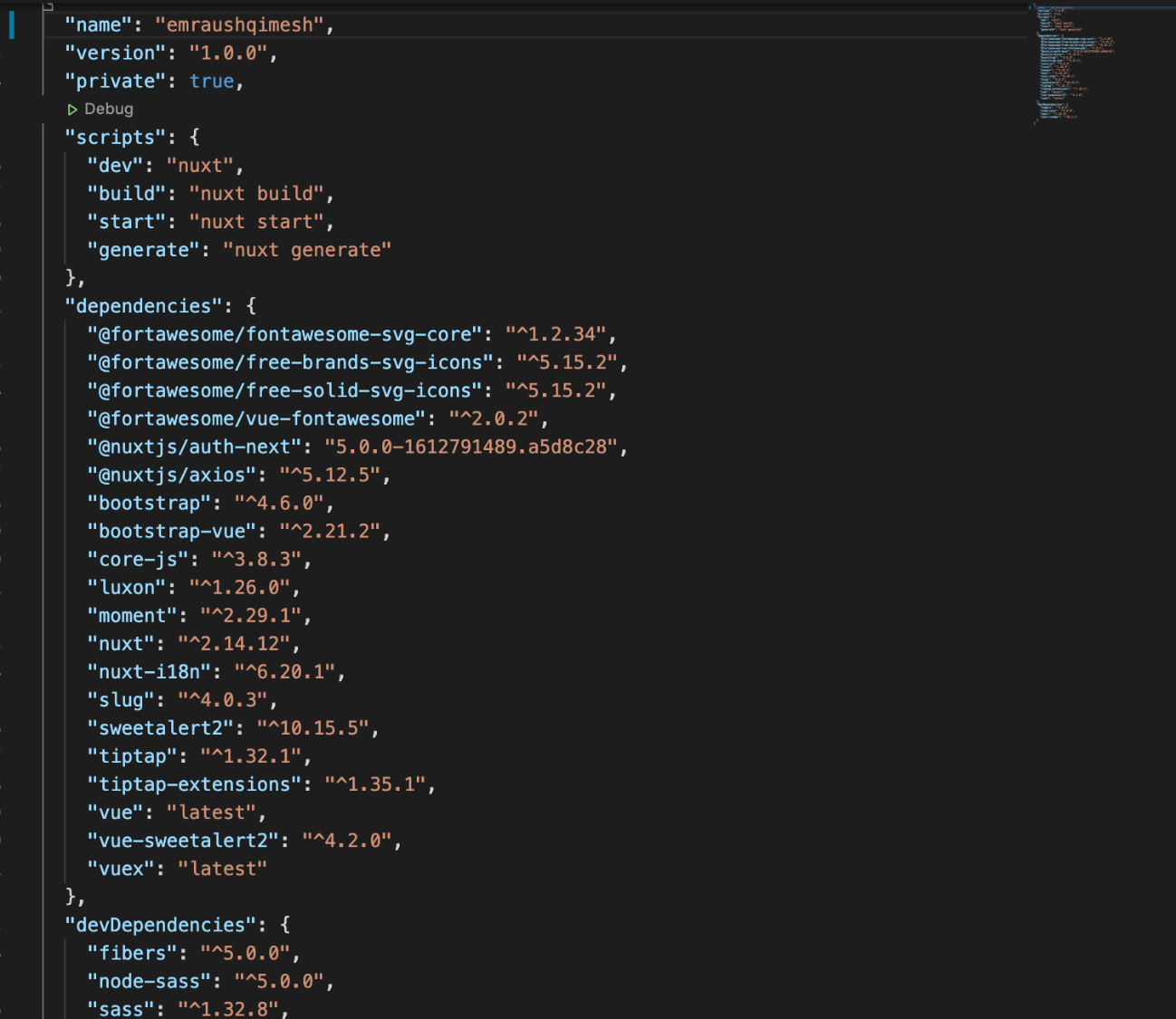
**Mirëmbajtesi**- duhet t’i ofrohet mundësia ti shikoj antaret ekzistues, tu a mundsoj atyre t’i shikojnë/ndryshojnë recetat e ushqimeve, vetë do të mund t’i shikoj / shtoj / ndryshoj /fshij recetat.

Kodi i aplikacionit “Kerko Ushqimin” strukturalisht ndahet në dy pjesa. Ana e përdoruesit dhe ana e serverit. Këto dy pjesa funksionojnë si aplikacione të ndarë. Ana e përdoruesit dërgon kërkesë në anën e serverit dhe merr përgjigje nga ai. Pastaj e përpunon këtë përgjigje dhe ja shfaq përdoruesit. Këto kërkesa dhe përgjigje dërgohen përmes HTTP-së.

Secila pjesë e aplikacionit para se të funksionojnë kërkojnë instalimin e moduleve të duhur nga Node.js. Po flasim per varësitë e aplikacionit apo dependencies e cila ndahet ne Varësitë(Dependencies) dhe Varësitë e Zhvillimit(Dev Dependencies).

Të gjitha këto dhe informata të tjera rreth aplikacionit mund të shihen në fajllet “package.json” që përmbajnë JSON objekte.

Me poshtë shohim package.json:



Në pjesën e dependencies po shohim frameworket dhe modulet të ndryshme, si Vue, Vuex.

Komand e cila na mundëson startimin e serverit për zhvillimin e aplikacionit është **npm run dev** neFront ndresa ne Back **php artisan serve**

Në këte projekt do të perdorim edhe Nuxt.js

Shumica e faqeve të internetit do të kenë më shumë se një faqe (d.m.th. një faqe në Home, About, Conntact etj.). Për t'i treguar këto faqe, na duhet një Router. Këtu hyn vue-routeri. Kur punoni me aplikacionin Vue, duhet të konfiguroni një skedar (d.m.th. router.js) dhe të shtoni të gjitha rrugët tuaja manualisht në të. Nuxt.js gjeneron automatikisht konfigurimin e router-vue për ju, bazuar në skedarët tuaj Vue të siguruar brenda direktorisë së faqeve. Kjo do të thotë që ju kurrë nuk duhet të shkruani një konfigurim router përsëri! Nuxt.js gjithashtu ju jep ndarjen automatike të kodeve për të gjitha rrugët tuaja.

Me fjalë të tjera, gjithçka që duhet të bësh për të patur routing në aplikacionin tënd është të krijosh skedarë .vue në dosjen e faqeve.

Për të lundruar midis faqeve të aplikacionit tuaj, duhet të përdorni përbërësin NuxtLink. Ky komponent përfshihet në Nuxt.js dhe prandaj nuk keni pse ta importoni ashtu siç bëni me komponentët e tjerë. Është e ngjashme me etiketën HTML <a>, përveç që në vend që të përdorim një href = "/ about" ne përdorim për = "/ about". Nëse keni përdorur router-in vue më parë, mund të mendoni për <NuxtLink> si një zëvendësim për <RouterLink>

Një lidhje e thjeshtë në faqen index.vue në dosjen e faqeve tuaja:

<template>

<NuxtLink to="/">Home</NuxtLink>

</template>

Pas përfundimit këtyre hapave, aplikacioni ynë do të duket si në figurat 10 deri 25.

• Faqja fillestar „/“, figura 13.



Figure 13. Pamje e faqes fillestare

• Faqja fillestar „/Keshilla“, figura 14.



Figure 14. Pamje e faqes “Keshilla”

• Faqja fillestar „/about “, figura 15.

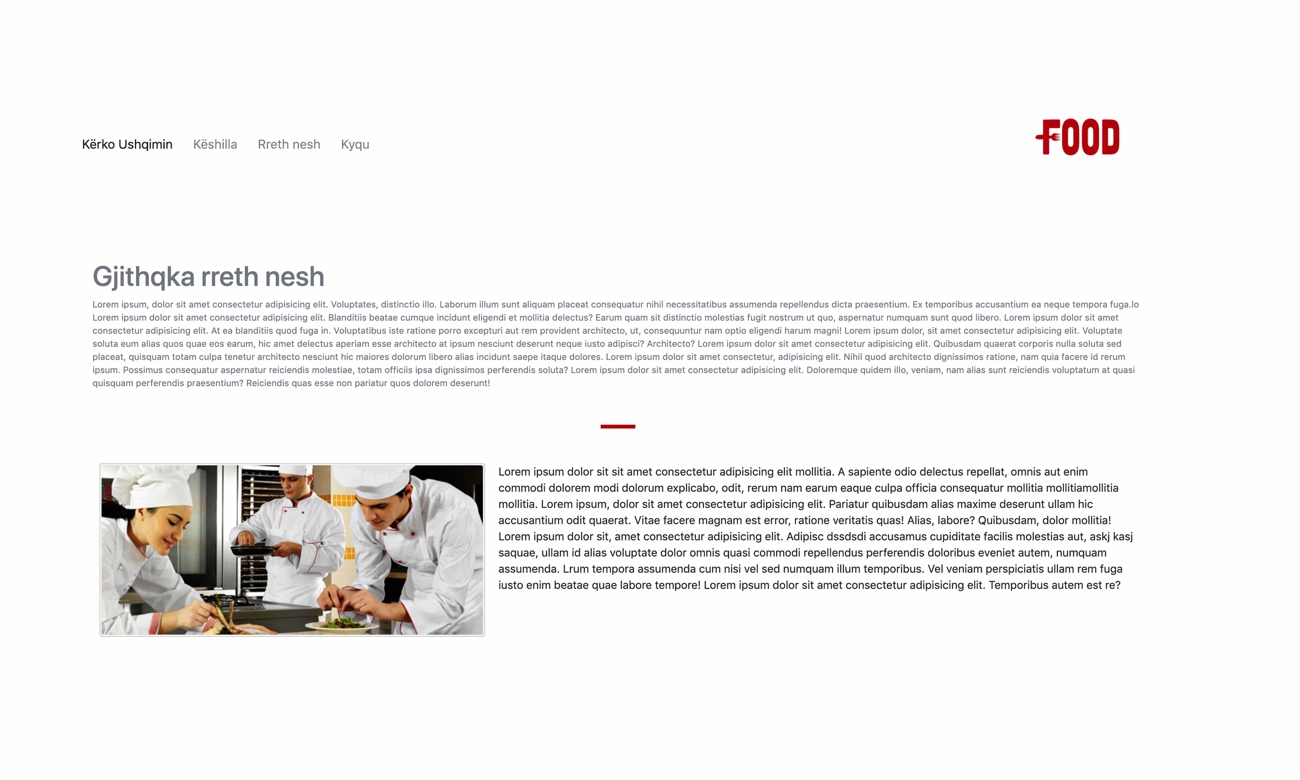


Figure 15. Pamje e faqes “Reth Nesh”

• Faqja “/login” për kyçje në sistem, figura 16

.

Figure 16. Pamje e forms per Kyqje

• Faqja “/register” për regjistrim në sistem, figura 17



Figure 17. Pamje e forms per Regjistrim

• Faqja “/forgetPassword” për regjistrim në sistem, figura 18



Figure 18. Pamje e forms per Ndryshim Fjalkalimi

• Faqja “/propozoUshqime” për regjistrim në sistem, figura 19



Figure 19. Pamje e forms per Propozimin e ushqimeve

• Faqja “/results” për regjistrim në sistem, figura 20

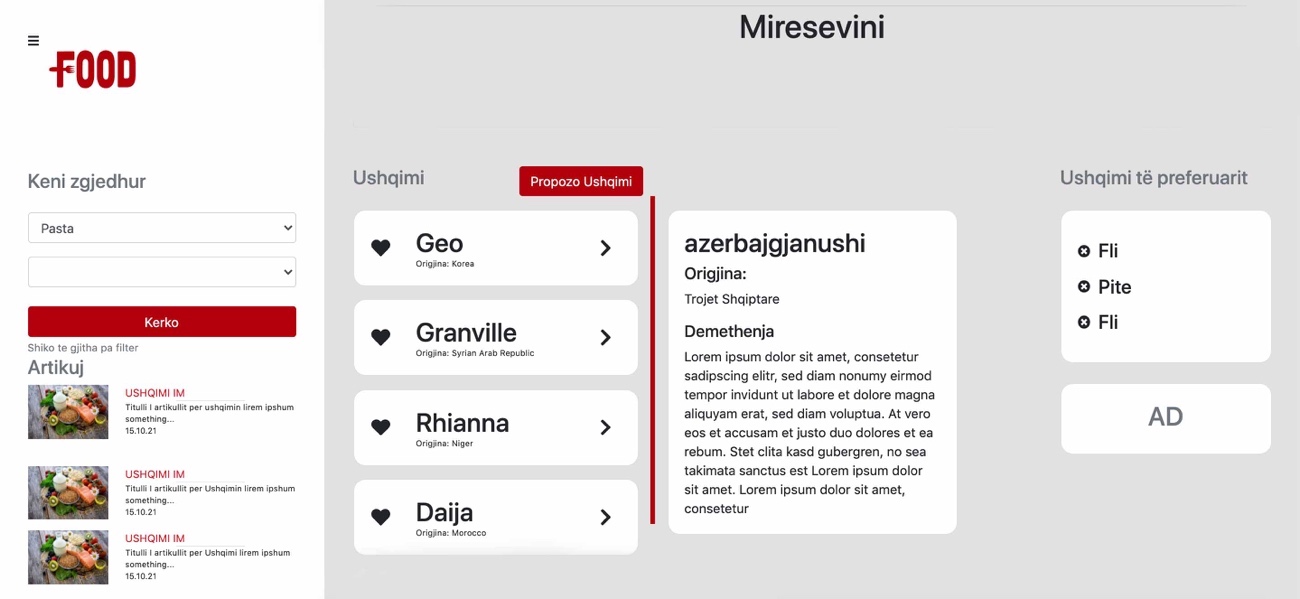


Figure 20. Pamje e informatave pa filter

• Faqja “/ushqime“ për shfaqje të të gjitha recetave, figura 21

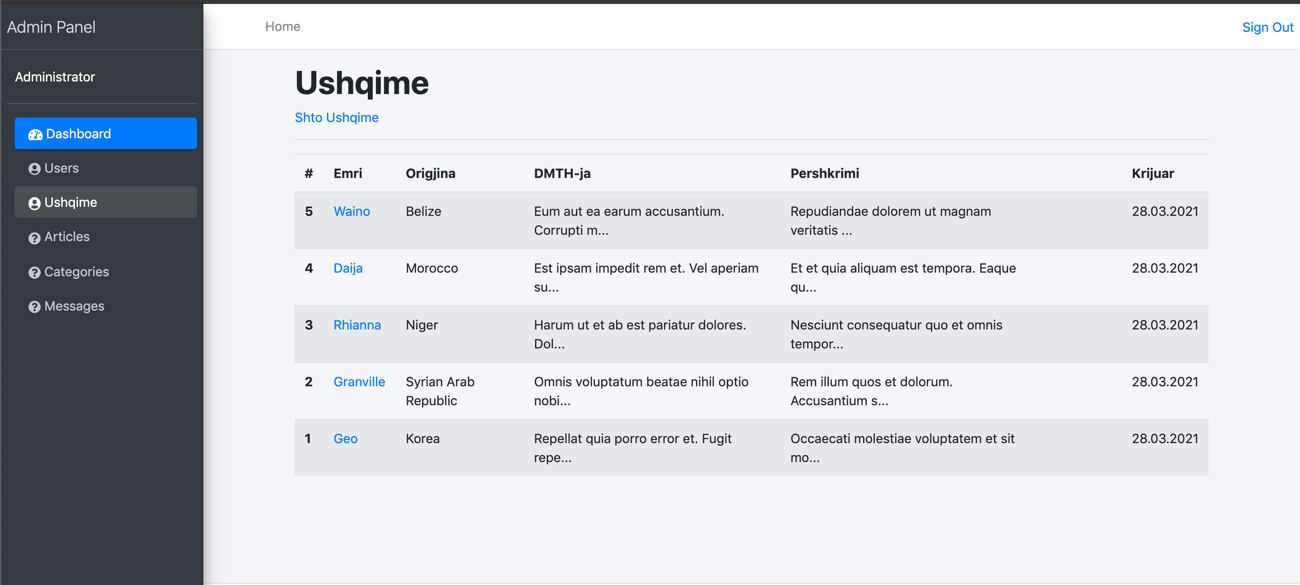


Figure 21. Pamje nga faqja e te gjitha recetave te ruajtura

• Faqja “/users“ për shfaqje të të gjitha perdoruesve, figura 22

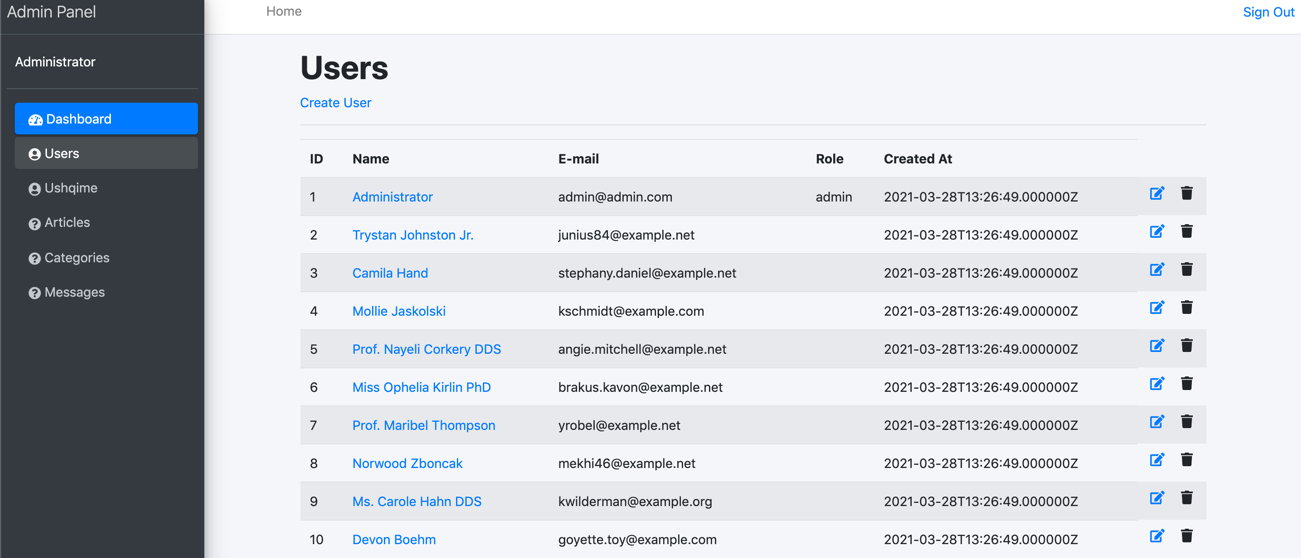


Figure 22. Pamje nga faqja e te gjithe Perdoruesve

### 5.1.1 Login Module

Për të loguar në aplikacion përdoruesi shkruan emailin e regjistruar dhe fjalëkalimin, dhe në klikimin në login thirrim metoden @submit.prevent=”login” Kjo metodë mbledhë të dhënat në një objekt dhe dërgon ato në metodën komunikues login().

<template>

<div id="login">

<div class="login-container">

<div class="form-signin">

<!-- <p class="error-message">{{ errorMessage }}</p> -->

<logo />

<p class="alert alert-danger" v-if="errorMessage">{{ errorMessage }}</p>

<div class="login-form-wrapper">

<div class="login-title">

<h4>Kyqu</h4>

<div class="sep"></div>

</div>

<form @submit.prevent="login">

<div class="form-group">

<label for="inputEmail" class="sr-only">Email address</label>

<input type="email" v-model="loginForm.email" @focus="hideErrorMessage" id="inputEmail" class="form-control" placeholder="Email address" required="" autofocus="">

</div>

<div class="form-group">

<label for="inputPassword" class="sr-only">Password</label>

<input type="password" v-model="loginForm.password" @focus="hideErrorMessage" id="inputPassword" class="form-control" placeholder="Password" required="">

</div>

<div class="checkbox mb-3">

<label class="float-right">

<input type="checkbox" value="remember-me"> Remember me

</label>

</div>

<button class="btn btn-lg btn-danger btn-block" type="submit">

<font-awesome-icon v-if="loading" fas icon="cog" class="fa-spin"></font-awesome-icon>

Sign in

</button>

<nuxt-link to="register" class="btn btn-lg btn-outline-danger btn-block" >Regjistrohu</nuxt-link>

<nuxt-link to="forgotPassword" style="color:black;">Forgot password?</nuxt-link>

</form>

</div>

</div>

</div>

</div>

</template>

Para se të krijojmë metodën e login() te ciles i japim një objekt me email dhe fjalëkalimin e përdoruesit, le të konfigurojmë nuxt-auth për të përdorur Laravel Sanctum.

E vendosim Laravel Sanctum provider si strategji që moduli nuxt-auth do të përdorë për vërtetim.

<script>

export default {

layout: 'blank',

middleware: ['guest'],

data() {

return {

loginForm: {

email: '',

password: ''

},

errorMessage: '',

loading: false

}

},

methods: {

async login() {

try {

this.loading = true;

await this.$auth.loginWith('laravelSanctum', {data: this.loginForm});

} catch (err) {

switch (err.response.data.status) {

case 422:

this.errorMessage = 'incorrect credentials'

this.loading = false;

break;

case 500:

this.errorMessage = 'Server is not responding please try later';

this.loading = false;

break;

case 403:

this.errorMessage = 'User not verified, please check you email';

this.loading = false;

break;

default:

this.errorMessage = 'Something went wrong, try again later';

this.loading = false;

break;

}

}

},

hideErrorMessage() {

this.errorMessage = false;

}

}

}

</script>

Së pari, ne përcaktojmë disa Data të të dhënave. Pastaj kemi metodën login(), ku po regjistrohemi duke përdorur ofruesin Laravel Sanctum.

Si vijim, ofruesi së pari bën një kërkesë në / sanctum / csrf-cookie për të marrë një shenjë CSRF dhe për ta vendosur atë si një cookie XSRF-TOKEN, e cila përdoret në kërkesat pasuese. Pastaj, bën një kërkesë POST në pikën përfundimtare të hyrjes / l ". Pas hyrjes së suksesshme, përdoruesi ridrejtohet në dashbord.

Sanctum përdorë vërtetimin e seancave të bazuara në cookie të Laravel për të vërtetuar përdoruesit... kjo na lejon të bëjmë kërkesa të mbrojtura nga CSRF në pikat përfundimtare normale.

Pra, forma përmban dy fusha: email dhe fjalëkalim. Kur bëhet submit, do të thirret një metodë login(). Duke përdorur loginWith të modulit Auth () dhe duke kaluar përgjatë të dhënave të përdoruesit, ne e regjistrojmë përdoruesin. Nëse vërtetimi ishte i suksesshëm, ne e ridrejtojmë përdoruesin në Dashbord. Përndryshe vendosni errorin në mesazhin e gabimit të marrë nga përgjigja e API. Përsëri, po përdorim përbërësin e njoftimit nga më herët për të shfaqur per error.

Inside routes/web.php kemi:

<?php

use Illuminate\Support\Facades\Route;

use Illuminate\Foundation\Auth\EmailVerificationRequest;

use Illuminate\Http\Request;

Route::get('/', function () {

return view('welcome');

});

Route::get('/email/verify/{id}/{hash}', function (EmailVerificationRequest $request) {

$request->fulfill();

return redirect('/');

})->middleware(['auth', 'signed'])->name('verification.verify');

Këtu, ne përpiqemi të vërtetojmë përdoruesin me detajet e dhëna. Nëse nuk gjendet asnjë ndeshje, ne thjesht kthejmë një përgjigje të përshtatshme. Përndryshe, përdoruesin mund të vazhdoj sesionin.

Pastaj tek routes/api.php krijojmë /logout :

Route::post('logout', [AuthController::class, 'logout']);

logout () heq detajet e përdoruesit nga sesioni. Pastaj, ne e zhvlerësojmë sesionin e përdoruesit dhe, së fundmi, rigjenerojmë CSRF-në.

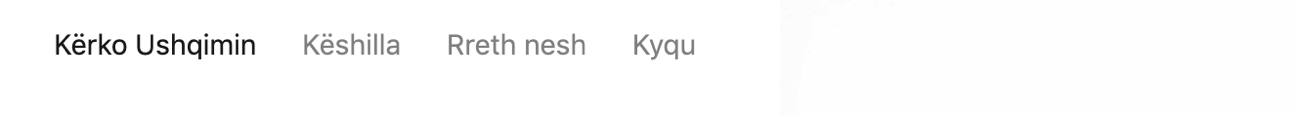
### 5.1.2 Shiriti i Navigimit

Për të mundesuar kalimin apo navigim nga një faqe në faqen tjeter e kemi shiritin e navigimit apo ‘navigation bar’.

Elementi <nav> përfaqëson një seksion të një faqe qëllimi i të cilit është të sigurojë lidhje navigimi, qoftë brenda dokumentit aktual, qoftë drejt dokumenteve të tjerë. Shembuj të zakonshëm të seksioneve të navigimit janë menut, tabelat e përmbajtjes dhe indekset.

Roli i shiritit të navigimit është një rol i rëndësishëm. Ofron një mënyrë për të identifikuar organizimin dhe strukturën e një faqe në internet. Duke klasifikuar dhe etiketuar seksionet e një faqe, informacioni strukturor i përcjellë vizualisht përmes navigacionit përfaqësohet në mënyrë programore. Ashtu si elementi HTML <nav>, monumentet e navigimit sigurojnë një mënyrë për të identifikuar grupet (p.sh. listat) e lidhjeve që synojnë të përdoren për navigimin në përmbajtjen e faqes në internet. Nëse një faqe përfshin më shumë se një pikë referimi të navigimit, secila duhet të ketë një etiketë unike.

Në aplikacionin tone ‘Kerko Ushqimin” shiriti I navigimit duket si më poshtë:

****

Pra, shiriti përbehet nga një unordered list ([<ul>](https://www.w3schools.com/tags/tag_ul.asp)) qe në vete përmban 4 <li>  list items.

<template>

<div>

<nav class="navbar navbar-expand-lg navbar-light top-navbar">

<div class="collapse navbar-collapse" id="navbarTogglerDemo03">

<ul class="navbar-nav mr-auto mt-2 mt-lg-0">

<li class="nav-item active">

<nuxt-link class="nav-link" to="/">Kërko Ushqimin</nuxt-link>

</li>

<li class="nav-item">

<nuxt-link class="nav-link" to="/Auth/blog">Këshilla</nuxt-link>

</li>

<li class="nav-item">

<nuxt-link class="nav-link" to="/Auth/about">Rreth nesh</nuxt-link>

</li>

<li class="nav-item">

<div v-if="this.$auth.loggedIn">

<nuxt-link v-if="this.$auth.user.role === 'admin'" class="nav-link" to="/admin/dashboard">Dashboard</nuxt-link>

<nuxt-link v-if="this.$auth.user.role !== 'admin'" class="nav-link" to="/profile">{{ this.$auth.user.name }}</nuxt-link>

</div>

<nuxt-link v-if="!this.$auth.loggedIn" class="nav-link" to="/Auth/login">Kyqu</nuxt-link>

</li>

</ul>

Tek <li> e parë e kemi nuxt-linkun me path tek faqja krysore to=“/” dhe e emeruar si “Kerko Ushqimin” .

Linku i dytë na dregon tek follderi Auth në fajllin blog.vue dhe emerohet si “ Keshilla”, dhe i treti gjithashtu tek follderi Auth por në fajllin about.vue dhe emerohet si “Rreth Nesh”.

Aftësia për të treguar ose fshehur elemente bazuar në kushtet është një tipar themelor i çdo kornizë frontend. Vue.js na ofron një sërë direktivash thelbësore për të arritur këtë efekt: v-if, v-else dhe v-show, v-if ka avantazhin kur bëhet fjalë për kohën fillestare të pasqyrimit.

Direktiva v-if mund të tregojë ose fshehë vetëm një element (dhe elementët e tij fëmijë), por gjithashtu mund të kontrolloni shumë elementë me një v-if të vetme duke zvogëluar dyfishimin.

Më poshtë kemi një if statement “ v-if ” duke pytur nese jemi të kyqur, pastaj nese jemi të kyqur dhe roli i userit është ‘admin’ atëherë nav-linku do të të dërgoj tek “/admin/dashboard” , ndresa nese roli i userit nuk është admin atëhere dërgoje tek “/profile” apo me thjeshtë tek profili i userit.

<nuxt-link v-if="!this.$auth.loggedIn" class="nav-link" to="/Auth/login">Kyqu</nuxt-link>

Tek kjo pjesë e kodit kemi prap nje if statment nese nuk jemi te kyqur nav-linku na dergon tek "/Auth/login" e cila emerohet si “Kyqu” duke na e shfaqur login formën.

Gjithashtu kemi edhe pjesën, e cila na shfaqet kur applikacioni hapet ne mobile:

<nav class="navbar-small">

<div class="search">

<font-awesome-icon fas icon="search" />

</div>

<a class="navbar-brand mx-auto" href="/">

<logo />

</a>

<div class="bars">

<font-awesome-icon fas icon="bars" />

</div>

</nav>

</div>

Nga font-awsome-icon kemi marr dy icon, njera eshte search icona dhe tjetra eshte bars icona. Ne momentin kur aplikacioni hapet ne mobile atehere navigation bar do te shfaqet si ne fig. me poshte:



Figura 23. Pamje nga Navigation bar ne Mobile

# 6. DISKUTIME DHE PËRFUNDIME

Roli i Font-End tekonologjive është shumë i rëndesishëm, ndikimi i tyre shtrihet edhe pas periudhës së ndertimit të projekteve. Front-end teknologjte garojnë mes vete duke shtuar karakteristika të veqanta dhe duke e avancuar ndërtimit në webit. Javascript edhe sot qendron e para si gjuhë programuese më e përshtatshme në shumë aspekte, e në saj të saj hyne edhe dy prej framework-eve të lart permendura.

Si perfundim Vue.js është në pozitë të barabartë me kornizat e tjera kur ndërton aplikacione në shkallë të gjerë e më së shumti shkëlqen kur duhet të punosh me aplikacione të trashëguara ose të futesh brenda dhe të bësh diçka shpejt. Çfarë pelqejnë zhvilluesit në lidhje me VueJS është : Të mësuarit e lehtë, stili dhe modelet elegante të programimit gjithashtu edhe dokumentacioni i mirë.

Vue.js renditet nder kornizat më të dashura në ueb në sondazhin Stack Overflow 2020, duke e zënë vendin e dytë.

Vue.js është mjete i shkëlqyer për ndërtimin e ndërfaqeve interaktive të përdoruesit. Për të zgjedhur Vue për projektin tuaj të ardhshëm, duhet të merrni parasysh një numër faktorësh, duke përfshirë rastin tuaj specifik të përdorimit, nevojat e biznesit tuaj, mjedisin, disponueshmërinë e zhvilluesit, buxhetin dhe kornizën kohore me të cilën po punoni.

Shpresoj shumë qe ky punim hulumtimi të shërbejë për qëllime informative dhe profesionale për komunitetin e web zhvilluesve por edhe atyre që janë në dilema se ciliën prej këtyre dy teknologjive të zgjedhin.

# 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/. |