Spring 5-2021

EXPRESS.JS KUNDREJT NEST.JS: KRAHAISIMI I DY NODE.JS FRAMEWORK-AVE TË DEDIKUAR PËR SERVER

Blendi Misini

Programi për Shkenca Kompjuterike dhe Inxhinierise

EXPRESS.JS KUNDREJT NEST.JS: KRAHAISIMI I DY NODE.JS FRAMEWORK-AVE TË DEDIKUAR PËR SERVER

Shkalla Bachelor

Blendi Misini

Maj / 2021
Prishtinë

Programi për Shkenca Kompjuterike dhe Inxhinierise

Punim Diplome
Viti akademik 2017-2018

Blendi Misini

EXPRESS.JS KUNDREJT NEST.JS: KRAHAISIMI I DY NODE.JS FRAMEWORK-AVE TË DEDIKUAR PËR SERVER

Mentori: Msc. Lavdim Menxhiqi

Maj / 2021

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

ABSTRAKT

Qëllimi i këtij hulumtimi është krahasimi i dy framework të Node js përkatësisht NestJS dhe ExpressJS të cilat përdoren për ndërtimin e aplikacioneve efiqiente dhë të zgjerueshm e nga ana e serverit. Node.js e (njohur edhe si Node) është një përmbledhje e paketuar e motorit JavaScript V8 të Google, për ndërtimin e lehtë të aplikacioneve të shpejta,të shkallëzueshme dhe të lehta sa i përket hapsiërs memorike që këto framework alokojnë për nevojat e tyre. Node.js është një tekonologji e cila është ndërtuar duke u përqëndruar në performancën e lartë si dhe konsumin e ulët të memories kompjuterike. Në këtë punimë, ne ofrojmë një përmbledhje të dy Node.js framework-ave shumë të popullarizuar duke i krahasuar ata ndërmjet vete më qëllim që të kemi më të lehtë që të vendosim se cili prej tyre na përshtatet më shumë për nevojat tona të krijimit të një aplikacioni nga ana e serverit. Fillimisht, ne përqendrohemi në modularitetin e Node, disa nga vequritë e tij kryesore si dhe menyra e punës së tij, qfarë na ofronë më shumë në krahasimë me gjuhët e tjera që përdoren nga ana e serverit si dhe historikun e tij, gjithashtu lidhshmërinë e ngushtë me javascript dhe arkitekturën e punës së Node.js. Karakteristika kryesore e Node.js është përdorimi i non-blocking event-driven I/O me një model programimi asinkron për të mbetur i lehtë dhe efikas në trajtimin e shumë kërkesave njëkohsishtë e njohur ndryshe edhe si concurrency. Këto përmbajnë tiparet themelore të Node.js , të cilat i diskutojmë në detaje si dhe detajet e dy framework-ave të tij. Node.js ndryshon nga JavaScript të cilën ne e përshkruajmë duke theksuar disa mangësi të mëdha në JavaScript gjë që Node.js i rregullon këto mangësi. Po ashtu, duke krahasuar dy framework-at e Node.js do të shohim diferencat e tyre në implementimin e struktures së gatshme e cila na ofrohet out-of-the-box, si dhe sqarimin e qdo file të gjeneruar automatikisht nga këta framework-a, emertimin e tyre si dhe funksionalitetin të cilin na ofrojn e gjithashtu arsyjet e përdorimit të tij. Me Node.js, mund të ndërtohen aplikacione komplekse në kohë reale që mund të shkojnë deri në miliona lidhje klientësh njëkohsishtë. Ne gjithashtu diskutojmë faktorët që mbështesin zgjedhjen e Node.js dhe pse zhvilluesit duhet ta përdorin atë, si dhe benifitet e përdorimit të framework-ave të Node.js nga zhvilluesit e aplikacioneve nga ana e serverit por jo vetëm kaq, përdorimi i Node.js nga ana e serverit lehtëson edhe punen e zhvilluesve të aplikacionit nga ana e

klientit si dhe ul kostot e mirembajtjes dhe redukton numrin e kerkesave që duhet bërë për të punuar aplikacioni gjë që kemi trajtuar dhe permendur raste ku Node.Js ka permirsuar dhe optmizuar punën e aplikacioneve egzistues që kan kaluar pastaj ne Node.js. Do të diskutojmë dhe sqarojmë se si duhet të instalohen këto dy framework-a, dhe njëkohsishtë qfarë na ofrojnë ata nga instalimi bazik i tyre. Do të krijojmë një aplikacion bazikë ku do të shqyrtojmë se si duhët të krijohet i njejti aplikacion në dy framework-at. Gjithashtu, Node.js ofronë mundësin e lidhjes me databazë, ku suporton thuajse të gjitha llojet e databazave egzistuese. Do të ofrojmë nje shembull të lidhjes me databazë të të dy framework-ave me MongoDB një databazë e cila punonë shume mire me Node.js. Kur flasim për zgjerueshmërin e aplikacioneve komplekse dhe të mëdha, Node.js është një tekonologji e cila e ofron shumë mirë këtë shërbim, si në aplikacione më strukturë monolite edhe ato me microshërbime, do të diskutojmë gjithashtu se qfarë zgjerushmërie na ofrojnë këto dy frameworka të Node.js dhë cila është më e lehtë të zgjerohet, pasi që zgjerueshmëria e një aplikacioni është një ndër faktorët kryesorë të cilin duhet të kemi parasysh kur krijojmë një aplicakionë të madhë dhe kompleks. Node.js njihet gjithashtu për shpejtësin e saj shumë të madhe, pas krijimit të aplikacioneve tona të thjeshta ne do të bëjmë një krahasim ndërmjet tyre rreth performances si dhe qfarë mundësi na ofrojnë që të permirsojmë performancën e tyre. Përfundimi kryesor është që zhvilluesit duhet të kuptojnë se cili framework duhet të përdoret nga ta për nevojat dhe qëllimet e aplikacionit të tyre.

MIRËNJOHJE/FALENDERIME

Fillimisht, falenderimi i takon Zotit për suksesin tim, ndaj të cilit përulem me krenari.
Punimi i temës së diplomës kërkon mund dhe përkushtim të madh i cili nuk mund të realizohej pa përkrahjen e njerëzve më të dashur, që më kanë mbështetur jo vetëm tani por gjatë gjithë rrugëtimit tim.
Falenderoj babain tim, Avni Misini dhe nënën time, Shefkije Misini të cilët besuan dhe më përkrahën në suksesin tim. Një falenderim i veçant shkon ndaj mentori tim të nderuar prof. Lavdim Menxhiqi i cili në mënyrë të vazhdushme më përcillte hap pas hapi drejt studimeve, ku me mbështetjen, dhe udhëzimet e vlefshme të profesorit synonim drejt ngritjes së nivelit të punimit.
Dhe krejt në fund, një falënderim shumë i sinqert i kushtohet shoqërisë për këshillat dhe përkrahjen e vazhdueshme, si dhe për mbështetjen që më kanë dhënë në çdo çast.

PËRMBAJTJA

LISTA E FIGURAVE

1. HYRJE

Aplikacionet e bazuara në WEB janë duke rritur popullaritetin e tyre pasi po bëhen më të lehta për tu zhvilluar, mirëmbajtur dhe siguruar. Gjithashtu ato janë lehtësisht të arritshme për klientët dhe nuk kërkojnë instalime shtesë në shumicën e rasteve dhe janë të personalizueshme varësisht nga kërkesa e shfyrtëzuesit si dhe lehtë të menagjueshme. Gjithashtu rritja e popullaritetit të JavaScript ka sjellë me vete shumë ndryshime, dhe zhvillimi i web page sot ka ndryshuar në mënyrë dramatike nga më parë. Node.js lejon që JavaScript të përdoret në të dyja anët e web-page, si në server (back-end) ashtu edhe në ndërfaqen e klientit(front-end). Duke qenë JavaScript nga ana e serverit është një tjeter avantazh i admirueshëm i Node.js mbi të tjerat gjuhë programuese nga ana e serverit sepse nga një zhvillues do të kërkohet të ketë njohuri dhe përvojë vetëm të një gjuhe të vetme, dmth JavaScript, pa marrë parasysh nëse ai është zhvilluesi skripteve nga ana e klientit ose skriptet për anën e serverit. Zhvilluesit nuk i kërkohet të ndërrojë ciklet e trurit të tij për një gjuhë në anën e klientit dhe pastaj për një gjuhë tjetër në anën e serverit. Node.js gjithashtu ofron një bibliotekë të pasur me module të ndryshme JavaScript që thjeshton zhvillimin e aplikacioneve në internet duke përdorur Node.js në një masë të madhe. JavaScript fillimisht është ekzekutuar vetëm në shfletuesin e internetit, por kërkesa e konsiderueshme për të e ka sjellë edhe në anën e serverit gjithashtu. Gjërat që mund të bëjmë në internet ditët e sotme me JavaScript që ekzekutohet në server, si dhe në shfletues, ishin të vështira të imagjinoheshin vetëm disa vjet më parë, ose ishin të kapsuluara brenda ambienteve me sandbox si Flash ose Java Applelet.Wikipedia thotë se: "Node.js është një përmbledhje e paketuar e motorit JavaScript V8 të Google, shtresës së abstraksionit të platformës libuv dhe një bibliotekë thelbësore, e cila vetë shkruhet kryesisht në JavaScript." Node.js është një run-time mjedis që lejon zhvilluesit e programeve kompjuterikë të lancojnë front-end dhe back-end të aplikacioneve në internet duke përdorur JavaScript fillimisht i zhvilluar në 2009. Përtej kësaj, vlen të përmendet se Ryan Dahl, krijuesi i Node.js, po synonte të krijonte faqe të internetit që përpunonin dhe shfaqin të dhëna në kohë reale me aftësi shtytëse(push), "frymëzuar nga aplikacione si Gmail". Në Node.js, ai u dha zhvilluesve një mjet për të punuar në non-blocking, event-driven I/O paradigm. Pas mbi 20 vjet stateless-web pa `state` bazuar në paradigmën e kërkesës-përgjigje stateless, ne më në fund kemi aplikacione në internet me lidhje dy kahëshe në kohë reale. Në një fjali Node.js shkëlqen në aplikacione në kohë reale duke përdorur teknologji shtytëse mbi websockets. Çfarë është kaq revolucionare

në lidhje me këtë? Epo, pas mbi 20 vjet stateless-web bazuar në paradigmën e përgjigjjes së kërkesës stateless, ne më në fund kemi aplikacione në internet me lidhje në dy kohë, në kohë reale, ku klienti dhe serveri mund të fillojnë komunikimin, duke i lejuar ata të shkëmbejnë të dhëna lirisht . Kjo është në kontrast të plotë me paradigmën tipike të përgjigjjes në internet, ku klienti gjithmonë fillon komunikimin. Për më tepër, e gjitha bazohet në pirgun e hapur të uebit (HTML, CSS dhe JS) që kalon mbi portën standarde 80. Dikush mund të argumentojë se ne e kemi pasur këtë për vite në formën e Flash dhe Java Applets - por në të vërtetë, ato ishin thjesht ambiente me sandbox duke përdorur uebin si një protokoll transporti që do t'i dorëzohej klientit. Plus, ato drejtoheshin në izolim dhe shpesh operonin mbi porte jo standarde, të cilat mund të kenë kërkuar leje shtesë dhe të gjëra të tilla. Me të gjitha avantazhet e saj, Node.js tani luan një rol kritik në grumbullin teknologjik të shumë ndërmarrjeve të profilit të lartë që varen nga përfitimet e tij unike. Trafiku i përditshëm në internet po rrit kërkesën për të ofruar zgjidhje të reja për të përmirësuar natyrën e concurrent-te të shërbimit në applikacione [1]. Një zgjidhje të till e ofron Node.js (e cila nganjëherë quhet edhe Node ose nyja) e cila është e lehtë sa i përket hapsier memorike dhe plotëson kërkesat përmes modelit I / O të drejtuar nga eventet, dhe evente jo-bllokuese si dhe ofron JavaScript nga ana e serverit. Motori JavaScript V8 i Chrome është baza për Node.js pasi në të është ndërtuar koha e funksionimit JavaScript e Node.js [2]. V8 Google me performancë të lartë JavaScript dhe WebAssemble është motori me burim të hapur, i shkruar në C ++. Përdoret në Chrome dhe në Node.js. Zbaton ECMAScript dhe WebAssemble, dhe ekzekutohet në Windows 7 ose në versione më të reja të Windows, macOS 10.12+ dhe sisteme Linux që përdorin procesorë x64, IA-32, ARM ose MIPS. V8 mund të funksionojë i pavarur, ose mund të integrohet në çdo aplikacion C ++ [3]. Meqenëse Node.js është i bazuar në ngjarje dhe jo në thread-based, ai është gjithashtu i aftë për tu shkallëzuar në miliona lidhje njëkohësisht.

Figura 1:Disa nga framework më të popullarizuara të NodeJs [4].

2. DEKLARIMI I PROBLEMIT

Framework-at për zhvillimin e aplikacioneve të ndryshme përdoren për të organizuar progresin e zhvillimit të tyre. Zhvilluesit marrin strukturën e gatshme për bazën e tyre të kodit ku mund të aplikojnë elemente të ripërdorshëme si dhe të shpesh herë ofron shpejtësi më të madhe të aplikacionit. Përdorimi i framework-ave të uebit për pjesën e përparme është shumë më i zakonshëm. Përdorimi i framework-ave për backend është më pak i zakonshëm, por është shume i dobishëm. Kur punojmë në projekte të zhvillimit të uebit, për pjesen më të madhe të aplikacioneve përdorin framework-a për anën e klientit si dhe anën e serverit sepse është shumë efikase dhe e përshtatshëme pordorimi i framework-ave në të dyja anët.

Kur zgjedhim framework-in më të mirë të Node JS për projektet tona, gjithmonë i kushtojmë vëmendje faktorëve përcaktues. Normalisht, duhet përvojë për të zgjedhur framework-in e përshtatshm të Node.js për një detyrë ose aplikacion të veçantë, por ka disa rregulla që funksionojnë mirë për të determinuar se cilin framework të zgjedhim. Punimi ynë do të fokusohet në tri problemet kryesore qe duhët të kemi parasysh kur zgjedhim një Node Js framework, keto probleme lidhen ngushtë me këto pikat e mëposhtme hulumtuese:

- Çfarë arkitekture kanë këta dy framework-a?
- Si duhet ti instalojm?
- Si bëhet lidhja me databazë dhe sa lloje të databazave mbështesin?
- Sa janë të zgjerueshëm dhe të menagjueshëm?
- Si është performanca e përgjithshme e tyre?
- Sa ofronjnë funksionalitet?
- Sa e vështrië është të adaptohen zhvilluesit e rinjë me këta framework-a?

3. SHQYRTIMI I LITERATURËS

Në këtë pjesë do të shqyrtojmë 2 framework të Node Js të cilët janë ExpressJS dhe NestJS per zhvillimin e aplikacioneve nga ana e serverit, ku do të shqyrtojm zgjerueshëmërin menagjueshëmërinë, performancat, si dhe funksionalitet të cilat naj ofrojnë këto dy framework-a të Node.js.

3.1 ExpressJS

Express.js u themelua nga TJ Holowaychuk. Lansimi i parë, sipas depozitës në GitHub të Express.js, ishte më 22 maj, 2010. Versioni 0.12.

Në qershor 2014, të drejtat për të menaxhuar projektin u morën nga StrongLoop. StrongLoop u ble nga IBM në shtator 2015; në janar 2016, IBM njoftoi se do të vendoste Express.js nën administrimin e inkubatorit të Fondacionit Node.js [5].

Express është framework më i popullarizuar në internet i Node.js dhe është biblioteka themelore për një numër framework-ave të tjerë të njohura të Node.js në internet. Ai siguron mekanizma për të:

- Shkruar handler për kërkesat me folje të ndryshme HTTP në shtigje (rrugë) të ndryshme URL.
- Integron me motorët e paraqitjes "view" në mënyrë që të gjeneroni përgjigje duke futur të dhëna në shabllone.
- Vendos rregullat e zakonshme të aplikacioneve në internet si porti për t'u përdorur për t'u lidhur dhe vendndodhja e modeleve që përdoren për dhënien e përgjigjes.
- Shtoni përpunimin e kërkesës shtesë "middleware" në çdo pikë brenda tubacionit të trajtimit të kërkesës [6].

3.2 NestJS

NestJS (i njohur si Nest) u zhvillua nga Kamil Myśliwiec me një mision për të ndërtuar aplikacione efektive, të shkallëzueshme me Node.js për backend. Ky framework mbështet gjuhën JavaScript dhe TypeScript. Për më tepër, ai përzien përbërësit e FP (Programim Funksional), OOP (Programim i Orientuar në Objekt) dhe FRP (Programim Reaktiv Funksional) [7].

Next.js u lanësua për herë të parë si një projekt me burim të hapur në GitHub më 25 tetor 2016. Ku është zhvilluar duke u bazuar në gjashtë parime kryesore: funksionalitet jashtë kutisë që nuk kërkon instalim, JavaScript kudo, të gjitha funksionet janë të shkruara në JavaScript, ndarje automatike të kodit dhe server-rendering, marrje të dhënash të konfigurueshme, parashikim i kërkesave dhe thjeshtsimi i deployment. Next.js 2.0 u prezantua në Mars 2017 duke përfshirë disa përmirësime që e bënë më të lehtë punën me faqe të vogla të internetit. Ai gjithashtu rriti efikasitetin e ndërtimit dhe përmirësoi shkallëzimin e tiparit të zëvendësimit të modulit të nxehtë. Versioni 7.0 u lëshua në shtator të vitit 2018 me përmirësimin e trajtimit të gabimeve dhe mbështetje për React's context API për përmirësimin e trajtimit dinamik të rrugës. Ky ishte gjithashtu versioni i parë në të cilin u azhurnua version më i ri i webpack. Versioni 8.0 u prezantua në Shkurt 2019 dhe ishte versioni i parë që ofroi vendosjen pa server të aplikacioneve, në të cilin kodi ndahet në funksione lambda që ekzekutohen sipas kërkesës. Versioni gjithashtu zvogëloi kohën dhe burimet e kërkuara për eksportet statike dhe përmirësoi performancën nga versioni paraprak Versioni 9.3, i prezantuar në Mars të 2020, përfshinte optimizime të ndryshme dhe mbështetje globale të modulit Sass dhe CSS. Më 27 korrik 2020, u botua versioni 9.5 i Next.js, duke shtuar aftësi të reja duke përfshirë rigjenerimin rritës statik, rishkrimet dhe mbështetjen e re-direct [8].

3.3 Arkitektura

Një arkitekturë aplikacioni ofron teknika dhe modele të hartuara mirë për të ndërtuar një aplikacion. Për më tepër, u siguron koduesve një udhërrëfyes dhe praktikat më të mira për të ndjekur dhe zhvilluar një aplikacion të strukturuar mirë . Për shembull, struktura e dosjeve dhe arkitektura e shtresave janë disa nga faktorët kryesorë që formojnë arkitekturën e përgjithshme të aplikacionit.

3.3.1 Arkitektura e ExpressJS

Express është një framework i Node.js per aplikacione në internet që ofron veçori të ndryshme përveç moduleve kryesore të Node.js që përshpejtojnë zhvillimin e uebit, si dhe e bëjnë më të lehtë zhvillimin. Express është një kombinim i middleware-ave që ekzekutohet nga lart poshtë në ciklin e kërkesës-përgjigjjes. Çdo middleware ka qasje në kërkesën dhe objektin e përgjigjjes.

Middleware merr kërkesën, ekzekuton kodin brenda, ndryshon kërkesën dhe objektet e përgjigjjes dhe thërret funksionin tjetër i cili aktivizon programin e middleware tjetër që

gjended në radhë. Një aplikacion i ExpressJS mund të ketë qasje në middleware të nivelit të aplikacionit, middleware të nivelit të routerit, middlware të trajtimit të gabimeve, middlware e integruar dhe middlware të palës së tretë. Middleware i nivelit të aplikacionit është i lidhur me një objekt të aplikacionit duke përdorur funksionet app.use () dhe app.method () ku 'method' është një folje HTTP që ekzekutohet. Middleware i nivelit te routerit ndryshon nga middleware i nivelit të aplikacionit vetëm pasi lidhet me një shembull të routerit ekspres. Middleware i trajtimit të gabimeve lidhet me objektin e aplikacionit dhe zakonisht kërkon katër argumente, gabimin, kërkesën, përgjigjjen dhe objektet e ardhshme. Middleware-at e integruara si express.static dhe express.json shërbejnë skedarët në mënyrë statike dhe analizojnë kërkesën hyrëse përkatësisht te JSON. Për më tepër, middleware e tjera të palëve të treta e zgjerojnë funksionalitetin e aplikacionit ExpressJS ose duke hyrë në informacionin në lidhje me kërkesën që është bërë ose duke analizuar cookies. Për më tepër, routerët në ExpressJS e ndajnë aplikacionin në disa aplikacione mini ExpressJS dhe i kombinojnë ato së bashku për të formuar një aplikacion të ExpressJS. Routeri i ExpressJS është i përbërë nga një folje e HTTP kërkese e cila mund të jetë (GET, POST, PUT, dhe DELETE) dhe një rrugë ose vendndodhen e burimeve të aplikacionit.

Për më tepër, callback funksionet specifikohen gjithashtu në metodën e rutimit ku një numër funksionesh callback mund të përdoren si argumente. Funksione të tilla callback duhet të thërrasin metodën tjetër të radhës nëse ka duke përdorur funksionin next() në mënyrë që të kalojnë në një tjetër callback funksion [9].

A

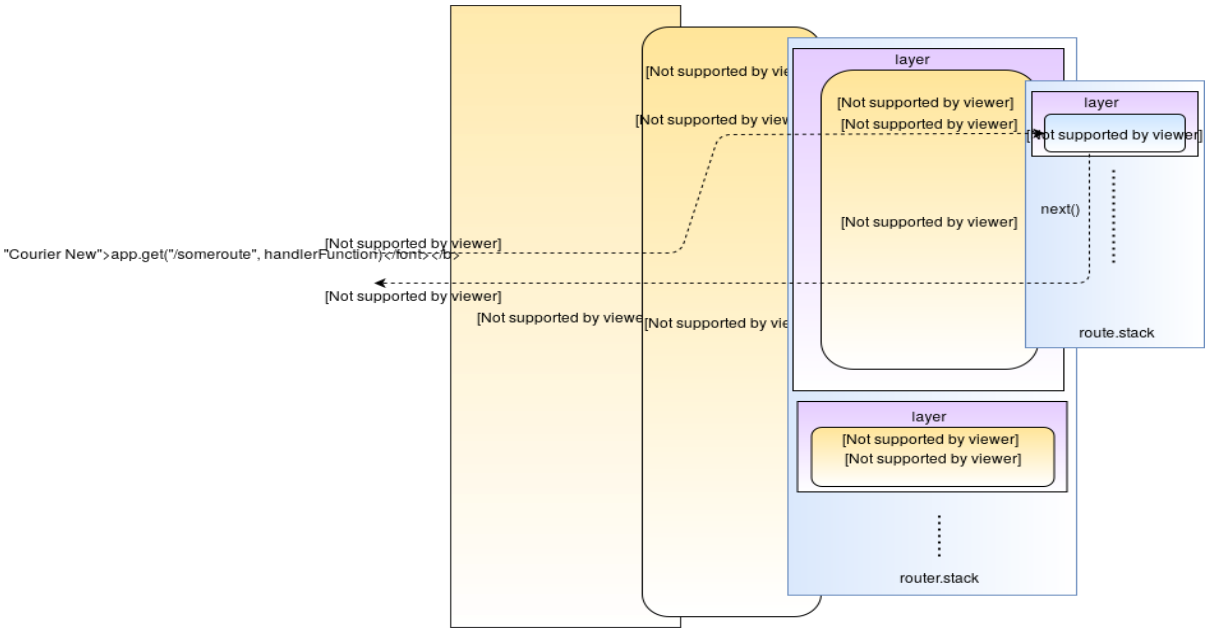

Figura 2:ExpressJS routing dhe http kerkesat.

3.3.2 Arkitektura e NestJS

Për sa i përket arkitekturës së NestJS, ekzistojnë 4 përparësi kryesore bazuar në artikujt dhe raportet ekzistuese. Së pari, NestJS është një framework i menduar mjaft mire. Një framework i menduar është një framework ku shtigjet e arkitekturës janë ndërtuar tashmë, dhe zhvilluesit duhet të ndjekin udhëzimet e dhëna të strukturës pa bërë asnjë supozim (Skowronski 2019). Duke përcaktuar një arkitekturë të arsyeshme që çdo kodues duhet të zbatojë, ai ofron një koleksion standardesh, duke rezultuar në një bazë kodesh lehtësisht të mirëmbajtshme (Yadav 2020). Ai tashmë e mbyll kontrolluesin prapa skenave me një bllok try-catch, analizon kërkesën

trupi, shton një mbajtës gabimesh, middleware, regjistrues, etj. Prandaj, kur scaffolding një projekt të ri, zhvilluesit nuk kanë nevojë të bëjnë këto gjëra të zakonshme çdo herë. Për më tepër, ata duhet të shkruajnë depot, shërbimet, kontrollorët në një mënyrë të udhëzuar (Rahman 2019). Së dyti, aplikacionet e NestJS janë shkruar në Typescript. Me gjuhë të loosely typed si JavaScript ose PHP, kodi mund të shkruhet pa i munduar shumë zhvilluesit. Prandaj, rreziku për të pasur gabime është shume i madhë pa type script. Nga ana tjetër, Nest është ndoshta i vetmi sistem i strukturuar mirë që është shkruar plotësisht me Typescript. Typescript është një version typed i JavaScript që mbështet zhvilluesit gjatë kohës së përpilimit dhe kohës së ekzekutimit. Me pak fjalë, Typescript është një JavaScript i shkallëzueshëm. Typescript gjithashtu u ofron përdoruesve qasje në veçoritë e reja të JavaScript që nuk janë prezantuar

ende. Një nga qëllimet kryesore të NestJS është të ndërtojë aplikacione të shkallëzueshme, kështu që trashëgon të mirat e TypeScript. Së treti, NestJS ka një arkitekturë solide sikur se Angular. Shumica e framework-ave të JavaScript backend nuk përballen me arkitekturë. Në Nest, struktura e file-ave bazohet shumë në Angular. Si pasojë, kjo ofron downtime minimal kur krijojm një Nest service. Secili komponent merr follderin e tij, modulet dhe skedarët kryesorë që qëndrojnë në root (plus disa skedarë shtesë konfigurimi). Ky framework është aq themelore sa tingëllon dhe i bën zhvilluesit ti kushtojnë më shumë vëmendje hartimit të pikave fundore për klientët të tyre në vend të strukturës së aplikacionit [7].

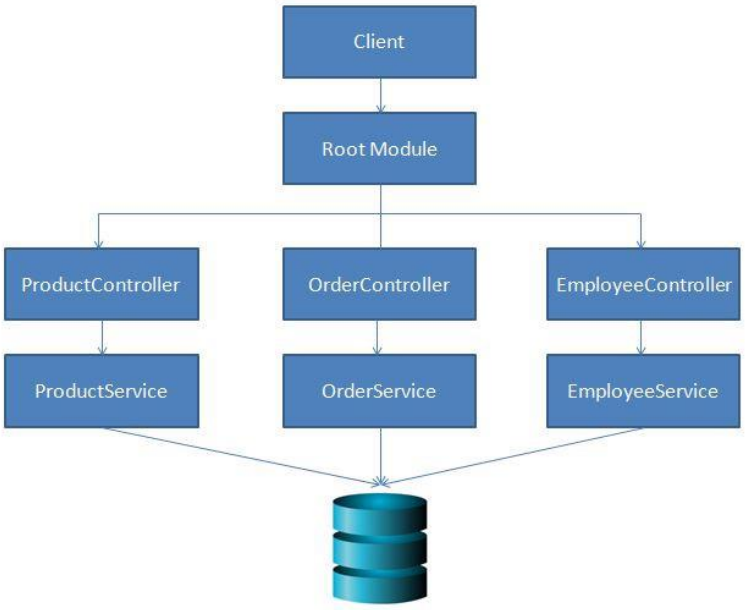

Figura 3:Arkitektura e NestJS [7]

3.4 Instalimi

ExpressJS edhe NestJS instalohen përmes Node Package Manager. Kjo mund te bëhet duhke egzekutuar disa komanda për instalimin e tyre. Pra paraprakisht duhet të kemi të instaluar NodeJ, i cili mund të shkarkohet nga web page i NodeJs.

3.4.1 Instalimi i ExpressJS

ExpressJS mund ta instalojm shumë leht dhe shpejt përmes Node.js duke përdorur komanden **npm install express –save.**

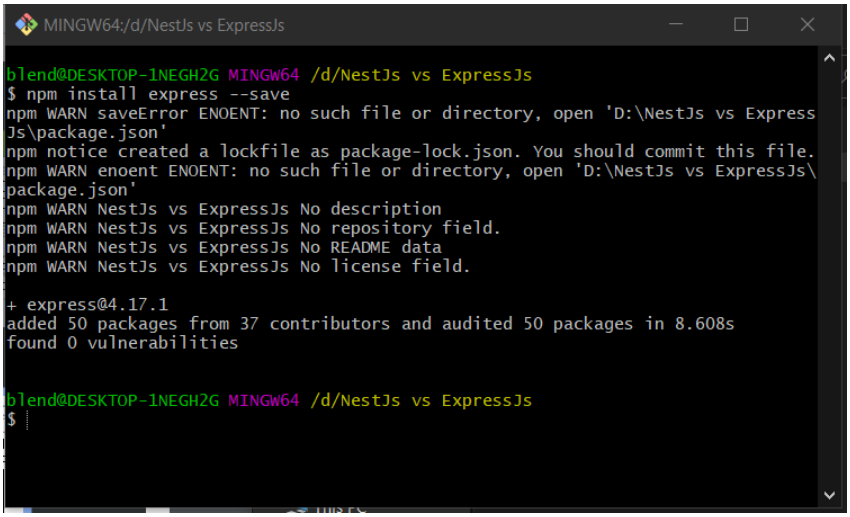

Figura 4:Instalimi i ExpressJS përmes npm.

E njejta gjë mund të bëhet edhe nëpërmjet komandes **yarn add express** nëse e kemi paraprakishtë të instaluar **yarn**. Komandat e mësipërme kërkojnë që menaxheri i paketës së Node të shkarkojë modulet e kërkuara të ExpressJS dhe t'i instalojë ato në përputhje me rrethanat.

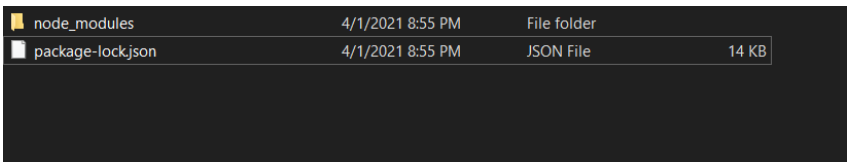

Figura 5:Përmbatja e një ExpressJS aplikacioni të sapo instaluar.

ExpressJS nuk ka nevojë për ndonjë komandë shtesë për krijimin e një aplikacioni. Le të përdorim faramework-un tonë Express të sapo instaluar dhe të krijojmë një aplikacion të thjeshtë "Hello World". Aplikacioni ynë do të krijojë një modul të thjeshtë serveri që do të dëgjojë numrin e portit 3000. Në shembullin tonë, nëse bëhet një kërkesë përmes shfletuesit në këtë numër porti, atëherë aplikacioni i serverit do të dërgojë një përgjigje 'Hello World ' tek klienti.

Për ta bërë këtë së pari duhet të krijojm një file të quajtur **index.js** në të njejtin follder që kemi instaluar ExpressJS. Pasi kemi krijuar file-in e emëruar me emrin **index.js** vendosim kodin si n figurën e mëposhtëme në atë file.

Figura 6:Kodi për krijimin e Hello World.

Tani dë të shpjegojmë kodin duke u bazuar në figurën më lart.

1. Në rreshtin tonë të parë të kodit, ne po përdorim funksionin **require** për të përfshirë "modulin e Express".
2. Para se të fillojmë të përdorim modulin e Express, duhet të krijojmë një objekt të tij.
3. Në rreshtin e tretë të kodit po krijojmë një funksion kthyes. Ky funksion do të thirret sa herë që dikush shfleton në rrënjën e aplikacionit tonë të internetit që është **http: // localhost: 3000**. Funksioni i kthimit do të përdoret për të dërguar vargun 'Hello World' në faqen e internetit
4. Në funksionin e kthimit, ne po i dërgojmë klientit vargun "Hello World". Parametri 'res' përdoret për të dërguar përmbajtje në faqen e internetit. Ky parametër 'res' është diçka që sigurohet nga moduli 'request' për të mundësuar që të dërgojë përmbajtje në faqen e internetit.
5. Atëherë ne jemi duke përdorur funksionin **app.listen** për të bërë që aplikacioni ynë server të dëgjojë kërkesat e klientit në portin nr 3000. Ju mund të specifikoni çdo port të disponueshëm këtu.

Pasi që kemi shtuar këtë kodë ne duhet që në të njejtin follder të egzekutojm komanden **node index.js** për të aktivizuar serverin tonë.

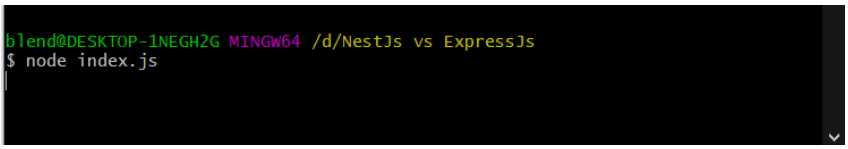

Figura 7:Komanda për egzekutimin e kodid të ExpressJS.

Për të parë rezultatin e kodit tonë, duhet të hapim shfletuesin dhe të nagivojm tek **http: // localhost: 3000.**

Figura 8:Rezultati i kodid në shfletues.

3.4.1 Instalimi i NestJS

Njejt sikur se ExpressJS, edhe NestJS instalohet përmes Node.js, vetëm se për instalimin e NestJS duhet të përdorim një komand tjetë e cila është **npm i -g @nestjs/cli**, dhe pastaj duhet të krijojmë një projekt me NestJS.

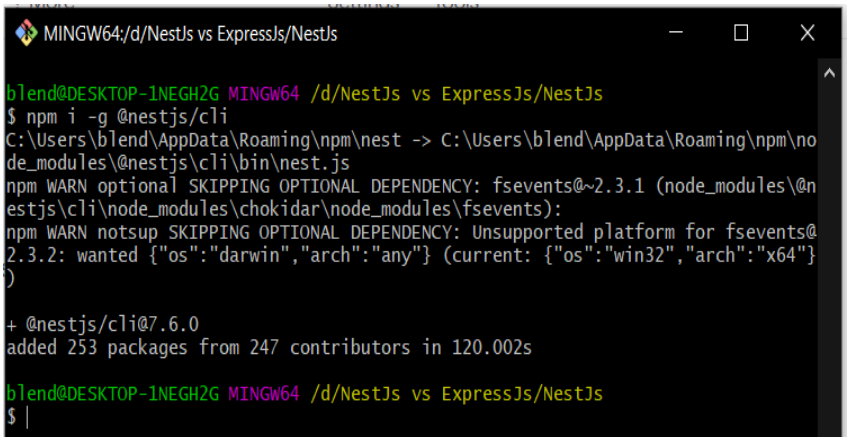

Figura 9:Instalimi i NestJS përmes npm.

Pas egzekutimit të suksesshëm të komandës më lart, navigojmë tek follderi ku dëshirojmë të ruajm projektin dhe pastaj ekzekutojm komandën **nest new nestjs-project**. Ndryshe nga Express Js, për të përdorur NestJS framework duhet të egzekutojm edhe këtë komand e cila krijon një aplikacion të zbrazët të NestJS.

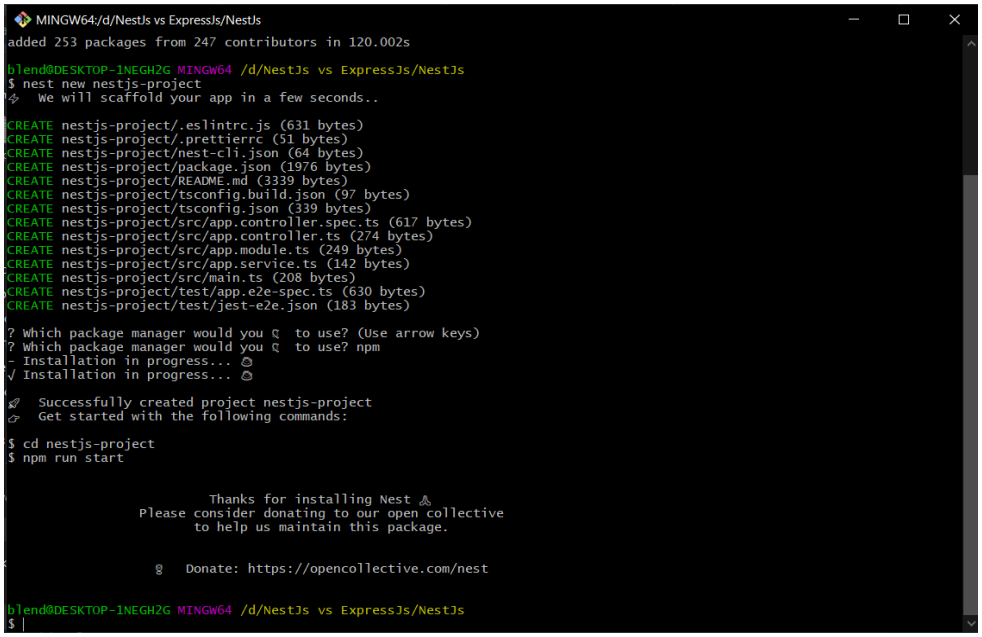

Figura 10:Krijimi i NestJS aplikacionit përmes NodeJs.

Prap pas egzekutimit të suksesshëm të komandës më lartë, navigojm tek follderi nestjs-project pra tek aplikacioni të cilin e emertuam më lart, këtu do të shohim një strukturë më ndryshe në krahasim me ExpressJS.

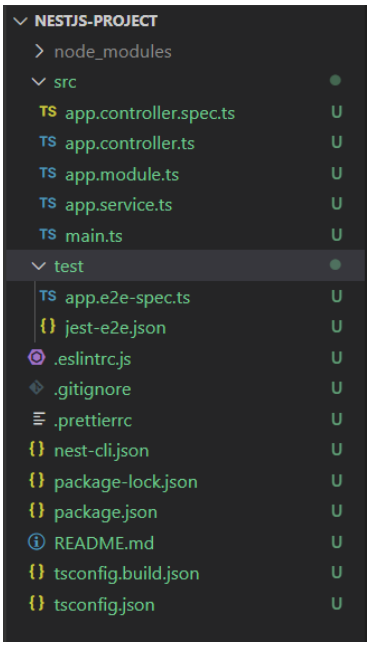

Figura 11:File dhe follderat të cilat krijohen autoamtikisht me krijimin e një NestJS aplikacionit.

Folderi më i rëndësishm i projektit është follderi src. Ky është vendi ku do të gjeni file-at TypeScript në të cilin është kodi i aplikacionit brenda. Më vonë, kur zbatojmë shembullin tonë të parë të `Hello World` sikurse me ExpressJS, do ta kalojmë shumicën e kohës duke punuar

në këtë follder, ndryshe nga ExpressJS ku nuk kishim ndonjë strukturë paraprake, apo file-a të krijuara automatikisht [10].

Brenda dosjes src mund të gjeni pesë skedarë në konfigurimin fillestar të aplikacionit:

- main.ts: Pika hyrëse e aplikacionit. Duke përdorur metodën NestFactory.create () krijohet një instancë e re e aplikacionit Nest.
- app.module.ts: Përmban implementimin e modulit rrënjësor të aplikacionit.
- app.controller.ts: Përmban zbatimin e një kontrollori bazë NestJS me vetëm një rrugë.
- app.service.ts: Përmban zbatimin bazë të shërbimit.
- app.controller.spec.ts: Skedari i testimit për kontrolluesin.

Tani le ti hedhim një sy kodit të gjeneruar automatikisht. Në file-in main.ts. Do të gjeni zbatimin vijues të paracaktuar automatikisht:

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
bootstrap();
```

Figura 12:Përmbajtja e main.ts file-it.

Kjo është pika e hyrjes së aplikacionit. Së pari, NestFactory importohet nga biblioteka @ nestjs / core. Për më tepër, AppModule importohet nga file app.module.ts i projektit tonë. Së dyti, funksioni bootstrap zbatohet dhe shënohet si async. Brenda këtij funksioni, thirret metoda NestFactory.create () dhe moduli i aplikacionit rrënjësor AppModule kalon në atë thirrje funksioni si argument. Kjo po krijon një shembull të ri të aplikacionit NestJS me modulin e bashkangjitur. Për të filluar serverin hapi tjetër është thirrja e metodës së dëgjimit dhe kalimi në portin në të cilin duhet të ekzekutohet serveri i internetit, p.sh. porti 3000 . Për shkak se metoda e dëgjimit po jep një promise kur serveri ka filluar me sukses, ne po përdorim fjalën await këtu. Më në fund skedari përmban thirrjen e funksionit bootstrap, në mënyrë që kodi të ekzekutohet.

Tjetra, le të hedhim një vështrim në zbatimin e modulit të kryesorë brenda aplikacionit që mund ta gjeni brenda file-it app.module.ts [10].

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';

@Module({
  imports: [],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

Figura 13:Përmbajtja e app.module.ts file-it.

Për të deklaruar klasën AppModule si një modul përdoret dekoruesi @Module i cili importohet nga @ nestjs / biblioteka e përbashkët. Një objekt me tre veti kalohet në zbukuruesin @Module. Tre vetitë janë:

1. imports
2. controllers
3. providers

Të gjithë kontrolluesit që duhet të jenë pjesë e AppModule duhet të jenë pjesë e grupit i cili i është caktuar pronës së kontrolluesit (në gjendjen fillestare të aplikacionit ekziston vetëm një kontrollues i cili i është caktuar modulit root AppController). Shërbimet që duhet të jenë të disponueshme në AppModule duhet të renditen në grupin të cilit i është caktuar pronës së ofruesit [10].

Tjetra, le të shohim nga afër zbatimin e AppController në app.controller.ts.

```ts
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }
}
```

Figura 14: Përmbajtja e AppController.ts file-it.

Ky është një implementim shumë i thjeshtë i një kontrollori të NestJS i cili përbëhet nga vetëm një route GET. Për ta bërë një klasë një kontrollues, duhet të shtoni zbukuruesin @Controller. Ky zbukurues importohet nga biblioteka @ nestjs /common. Një kontrollues mbështetet në një klasë shërbimi. Në këtë shembull të paracaktuar, AppController përdor një shërbim të quajtur AppService. AppService po implementohet në file-in app.service.ts dhe për këtë arsye duhet të shtohet një deklaratë përkatëse e importit. Duke përdorur konceptin e Dependency Injection, AppService futet në AppController (duke shtuar një parametër konstruktor të atij lloji). Një rrugë e paracaktuar zbatohet duke zbatuar metodën getHello. Në mënyrë që të deklarohet që kjo metodë merret me një kërkesë HTTP GET, kësaj metode i shtohet edhe zbukuruesi @Get. Metoda është duke përdorur metodën getHello nga AppService për të marrë të dhëna dhe njëkohësisht për t'i kthyer ato të dhëna si përgjigje [10].

Le të vazhdojmë dhe të shohim se çfarë ka brenda app.service.ts.

```ts
import { Injectable } from '@nestjs/common';

@Injectable()
export class AppService {
  getHello(): string {
    return 'Hello World!';
  }
}
```

Figura 15:Përmbajtja e app.service.ts file-it.

Ky file përmban implementimin e klasës AppService. Për ta bërë AppService një klasë shërbimi (e cila mund të injektohet në një kontrollues siç është parë më lartë), dekoruesi @Injectable duhet të shtohet para kësaj klase. Përsëri, ky lloj dekoruesi importohet nga paketa @ nestjs / common. Zbatimi i klasës së shërbimit është shumë i thjeshtë dhe konsiston vetëm në zbatimin e metodës getHello. Kjo metodë thjesht po kthen vargun statik "Hello World!". Në një file të botës reale, një metodë shërbimi, natyrisht, do të përdorej për të marrë të dhëna p.sh. nga baza e të dhënave, një shërbim në internet, ose ndonjë burim tjetër i të dhënave. Tani, kjo është një përshtypje e parë e blloqeve më të rëndësishme të ndërtimit të aplikacionit të parazgjedhur NestJS, tani sikurse tek ExpressJS le ta startojm applikacionin dhe të shohim se qfar rezultati do të marrim nga shfletuesi.

Për të startuar aplikacionin tonë të NestJS duhet që, brenda follderit të NESTJS-PROJECT të cilin e krijuam më parë të gzekutojmë komandën **npm run start.**

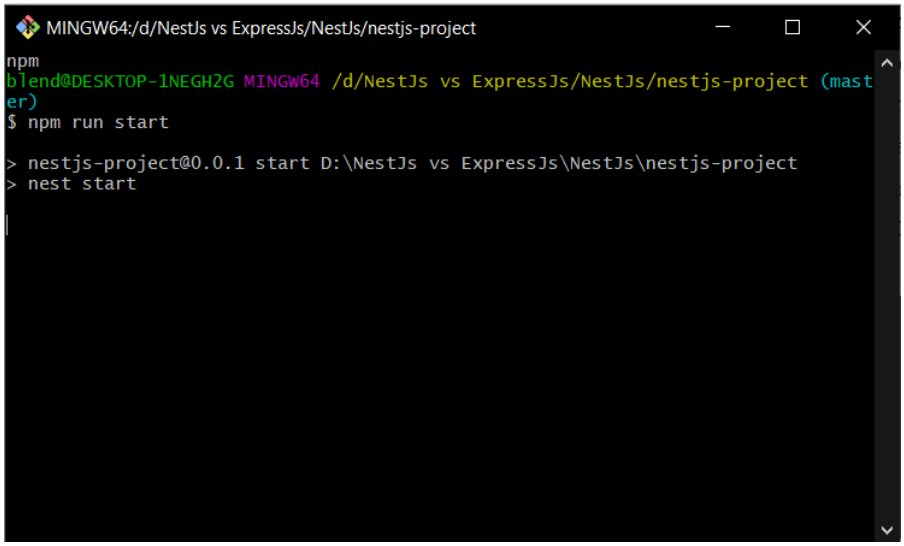

Figura 16:Startimi i aplikacionit NestJS.

Pasi kemi startuar aplikacionin e NestJS me komandën **npm run start**, tek shfletuesi ynë navigojmë tek URL **http://localhost:3000.**

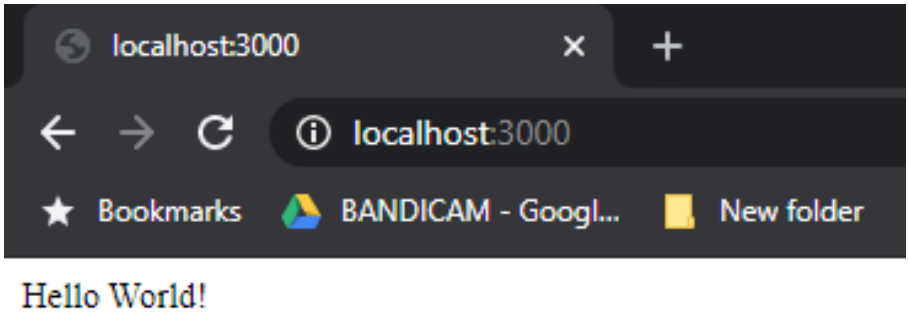

Figura 17:Egzekutimi i kodid të NestJS aplikacionit në shfletues.

3.5 Lidhja me databasë

Node.js mbështet të gjitha llojet e bazave të të dhënave pa marrë parasysh nëse është një bazë të dhënash relacionale ose një bazë të dhënash NoSQL. Sidoqoftë, bazat e të dhënave NoSQL si MongoDB janë përshtatja më e mirë me Node.js. Për tu qasur në bazën e të dhënave nga Node.js, së pari duhet të instaloni driver-in për bazën e të dhënave që dëshironi të përdorni, kjo bëhet përmes egzekutimit të komandes përkatëse pë instalimin e driver-it [11].

3.5.1 Lidhja me databasë në ExpressJS

Shtimi i aftësisë për të lidhur bazat e të dhënave në aplikacionet Express është thjesht çështje e instalimit të driver-ave përkatës të Node.js për bazën e të dhënave në aplikacionin tuaj. Ne do të shqytojm një lidhje me MongoDB. Mongoose instalohet në projektin tuaj (package.json) si çdo varësi tjetër - duke përdorur **NPM**. Për ta instaluar, përdorni komandën **npm install mongoose** në follderin kryesor tek i cili gjendet aplikacionin juaj dhe file package.json i aplikacionit tuaj.

Mongoose kërkon një lidhje me një bazë të të dhënave MongoDB. Ju mund të përdorni fjalën e rezervuar **require()** dhe të lidheni me një bazë të dhënash të hostuar lokalisht me **mongoose.connect()**, siç tregohet në figurën më posht [11].

```
//Importojm mongoose modulin
var mongoose = require('mongoose');

//Konfiguroni lidhjen me mongoose
var mongoDB = 'mongodb://127.0.0.1/my_database';
mongoose.connect(mongoDB, {useNewUrlParser: true, useUnifiedTopology: true});

//// Merrni lidhjen e paracaktuar
var db = mongoose.connection;

//Lidhni lidhjen me ngjarjen e gabimit (për të marrë njoftimin e gabimeve të lidhjes)
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
```

Figura 18:Krijimi i lidhjes me databasën MongoDB në ExpressJS.

3.5.2 Lidhja me databasë në NestJS

Nest mbështet dy metoda për integrimin me bazën e të dhënave MongoDB. Ju ose mund të përdorni modulin e integruar TypeORM, i cili ka një lidhës për MongoDB ose të përdorni Mongoose, mjeti më i njohur për modelimin e objekteve MongoDB. Në këtë kapitull ne do të përshkruajmë këtë të fundit, duke përdorur paketën e dedikuar @ nestjs / mongoose[23].

Fillojm duke instaluar varësitë e kërkuara përmes komandes **npm install --save @nestjs/mongoose mongoose** e cila duhet të egzekutohet ne folderin të cilin gjended aplikacioni. Sapo të përfundojë procesi i instalimit, ne mund të importojmë MongooseModule në root AppModule. Pra ndryshe nga ExpressJS këtu në NestJS kemi një file të dedikuar në të cilin rekomandohet që të krijohet lidhja me databasë.

```
1  import { Module } from '@nestjs/common';
2  import { MongooseModule } from '@nestjs/mongoose';
3
4  @Module({
5    imports: [MongooseModule.forRoot('mongodb://localhost/nest')],
6  })
7  export class AppModule {}
```

Figura 19:Krijimi i lidhjes me databasën MongoDB në NestJS.

Metoda forRoot () pranon të njëjtin objekt konfigurimi si mongoose.connect () nga paketa Mongoose. Pra ndryshe nga ExpressJS në të cilin për të krijuar një lidhje me MongoDB duhej që të shkruanim më shumë kodë këtu e kemi shumë më të leht dhe komapkte [12].

3.6 Performanca

Ekzistojnë dy aspekte të rëndësishme të performancës së softuerit: **responsiveness** dhe **shkallëzimi** [7]. Shumica e zhvilluesve e adhurojnë Node.js për shpejtësinë e tij të

papërpunuar, dhe kur bëhet fjalë për zgjedhjen e framework-ut, një perfeksionist mund të përçmojë çdo rrezik të performancës. Express siguron një shtresë të hollë në krye të Node.js me karakteristika të aplikacioneve në internet si rutimi bazë, middleware-at, motori i modelit dhe shërbimi i file-ave statik, kështu që performanca drastike e I / O e Node.js nuk rrezikohet [13].

3.7 Performanca e ExpressJS

Express është një framework minimal, framework pa korniz paraprake siq edhe e kemi përmendur më lart, nuk zbaton ndonjë nga modelet e përhapura të dizajnit si MVC, MVP, MVVM ose çfarëdo strukture bazike që del jashtë kutisë. Për adhuruesit e thjeshtësisë, ky është një plus i madh midis të gjitha framework-ave të tjerë, sepse ju mund ta ndërtoni aplikacionin tuaj me preferencën tuaj dhe pa kurbë të panevojshme të të mësuarit. Kjo është veçanërisht e dobishme kur krijoni një projekt të ri personal pa asnjë ngarkesë historike, por ndërsa projekti ose ekipi në zhvillim rritet, mungesa e standardizimit mund të çojë në punë shtesë për menaxhimin e projektit / kodit, dhe skenari i rastit më të keq mund të çojë në paaftësi për të mirëmbajtur aplikacionin, Pra Express siguron një shtresë të hollë të tipareve themelore të aplikimit në aplikacione, pa errësuar tiparet e Node.js që ju i njihni dhe i doni [14], dhe kjo siguron një performanc shumë të lartë të ExpressJS.

3.8 Performanca e NestJS

Ka disa diskutime ne stack overflow dhe platforma tjera online ku njerzit po krahasojnë një aplikacionë të thjesht `Hello World` midis Express dhe Nest js. Meqenëse vetë Nest përdor Express, është vërtet e pa kuptimtë të krahasosh këto të dyja. Për të marrë një performancë shumë më të mirë, Nest jep një mënyrë alternative për të ndryshuar zbatimin e kuadrit nga Express në Fastify (një tjetër framework i node). Duke përdorur Fastify ne mund të marrim performancë edhe më të mirë nga Nest sesa një "hello world" i thjeshtë i Express [15].

3.9 Shpejtësia e reagimit (Responsiveness)

Responsiveness është aftësia e një sistemi për të arritur kohën e tij të përgjigjes ose synimet e rrezultateve të ofruara për një kohë të caktuar. Në sistemet e përdoruesve fundorë, nga këndvështrimi i përdoruesve, responsiveness përcaktohet shpesh prej tyre. Responsiveness, për shembull, i referohet sasisë së kohës që i duhet një detyre nga përdoruesi për të përfunduar ose

numrit të transaksioneve që mund të përpunohen në një sasi të caktuar kohe. Responsiveness në sistemet në kohë reale është një masë se sa shpejt sistemi reagon ndaj një ngjarjeje [7].

3.9.1 Shpejtësia e reagimit e ExpressJS

Pasi që ExpressJS është nje framework minimal i NodeJs ku Non-blocking Input/Output dhe kërkesat asinkrone e bëri Node.js të aftë të përpunojë kërkesat pa asnjë vonesë. Në kontekstin e backend, përpunimi sinkron supozon që kodi ekzekutohet në një sekuencë. Kështu, secila kërkesë bllokon një thread, duke bërë që kërkesat e tjera të presin që ajo të përfundojë. Përpunimi asinkron lejon që kërkesat të përpunohen pa bllokuar (Non-blocking I / O) thread. Pra, pasi një kërkesë të përpunohet, ajo mund të shtyjë një përgjigje dhe të vazhdojë të shërbejë kërkesat. Kjo i ndihmon Node.js të shfrytëzojë në maksimum lidhjet e vetme, duke rezultuar në kohë të shkurtër të përgjigjejes dhe përpunim njëkohësisht. Një aspekt tjetër është modeli i \ event-based. Kur përdorni një gjuhë të përbashkët për të dy client/server-side, sinkronizimi ndodh shpejt, gjë që është veçanërisht e dobishme për aplikacione event-based, në kohë reale. Për shkak të natyrës së tij asinkrone, jo-bllokuese, me një thread, Node.js është një zgjedhje e popullarizuar për lojëra në internet, biseda, video konferenca ose çdo zgjidhje që kërkon të dhëna të azhurnuara vazhdimisht. Shembujt flasin vetë: shumë kompani të mëdha udhëheqëse në teknologji kaluan teknologjitë në aplikacionet e zhvilluara në Node.js dhe vunë re përmirësime të dukshme - PayPal, për shembull, vuri re një rënie prej 35 përqind të kohës së përgjigjes që nga migrimi nga Java [16].

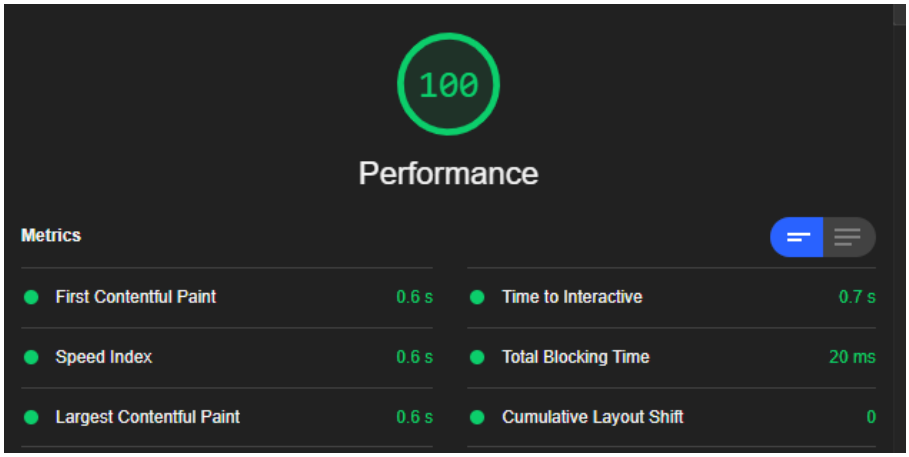

Figura 20:Performanca e ExpressJS e matur permes google lighthous.

Siç mund të shihet nga figura e mësipërme, reagimi i lartë është një avantazh i ExpressJS. Kur përdorni një aplikacion bazë Hello World, ExpressJS ka një përgjegjshmëri shumë të lartë me rezultatin 100 në raportin e siguruar nga Lighthouse - një mjet i automatizuar për matjen e performances së faqeve në internet.

3.9.2 Shpejtësia e reagimit e NestJS

Lidhur me NestJS, Nest gjithashtu ofron pajtueshmëri me bibliotekat e tjera, të tilla si Fastify ose edhe vet Express.js, sipas dokumentacionit të tyre zyrtar. Duke zbatuar një përshtatës të sistemit

qëllimi kryesor i të cilit është të ndërmjetësojë ndërmjetësuesit dhe administruesit për zbatime të përshtatshme specifike për bibliotekën përkatëse, Nest arrin këtë pavarësi të sistemit. Për shkak se Fastify dhe Express.js janë dy nga framework më të shpejta të Node.js, kjo rezulton në rritjen e përgjegjësisë së përgjithshme [7].

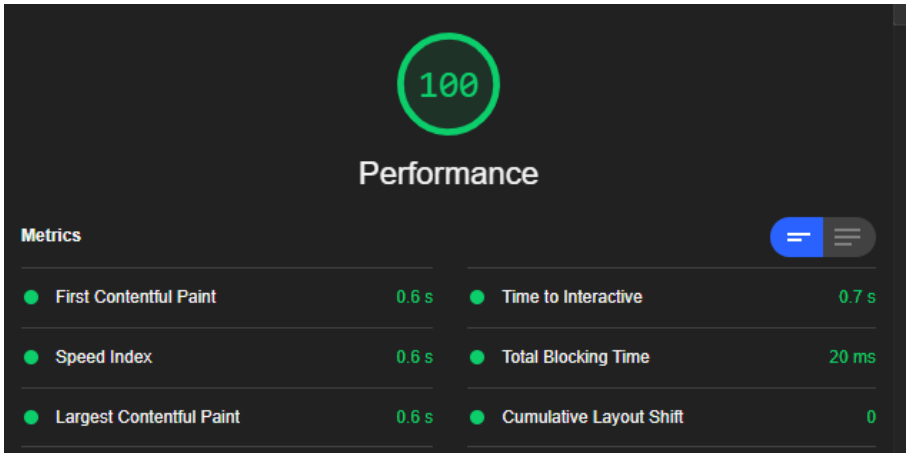

Figura 21:Performanca e NestJS e matur permes google lighthous.

Pra siq mund të shihet edhe nga figura me lart, performanca e NestJS nuk ndryshon fare nga ajo e ExpressJS, duke arritur rezultatet maksimale ne Lighthouse.

3.10 Shkallëzueshmëria (Scalability)

E thënë në terma të thjeshtë, shkallëzimi është aftësia e aplikacionit tuaj në internet për të përballuar një numër në rritje të përdoruesve që bashkëveprojnë njëkohësisht me aplikacionin tuaj. Si pasojë, një aplikacion i shkallëzuar i uebit është ai që performon me performancë të lart me një ose një mijë përdorues dhe i përballon ulje-ngritjeve të trafikut.

Kur bëhet fjalë për faktorët që ndikojnë në shkallëzimin e aplikacioneve në internet, është e rëndësishme të dijmë:

- Arkitektura e aplikacioneve në internet me dizajnin e kodit të pastër luan një rol të rëndësishëm në ndërtimin e aplikacioneve të shkallëzueshme të uebit. Framework-at kan një ndikim të madhë në performancën e aplikacionit (zgjidhni atë që funksionon më mirë për ju).

- Testi i ngarkesës ndihmon për të gjetur dhe kapërcyer anët e dobëta të aplikacionit tuaj në mënyrë që të garantojë performancën stabile të aplikacionit.
- Hardueri mund të ndikojë me sukses në shkallëzim vetëm nëse është ai që pengon zgjerueshmërin e sistemit.
- Shërbimet e palëve të treta gjithashtu duhet të zgjidhen me kujdes, përndryshe, ato mund të shkaktojnë dështime operacionale [17].

Meqenëse është një mjet i lehtë për teknologji, përdorimi i Node.js dhe ExpressJS për arkitekturën e mikrosherbimeve është një zgjedhje e shkëlqyeshme. Ky stil arkitektural përshkruhet më së miri nga Martin Fowler dhe James Lewis si "një qasje për zhvillimin e një aplikacioni të vetëm si një komplet shërbimesh të vogla, secili funksionon në procesin e vet dhe komunikon me mekanizmat e lehtë, shpesh me një API të burimit HTTP".Në përputhje me rrethanat, thyerja e logjikës së aplikacionit në module më të vogla, mikrosherbime, në vend që të krijoni një bërthamë të vetme, të madhe monolite, ju mundëson një fleksibilitet më të mirë dhe vendosni bazat për rritje të mëtejshme. Si rezultat, është shumë më e lehtë të shtosh më shumë mikroshërbime mbi ato ekzistuese sesa të integrosh veçori shtesë me funksionalitetin bazë të aplikacionit [17].

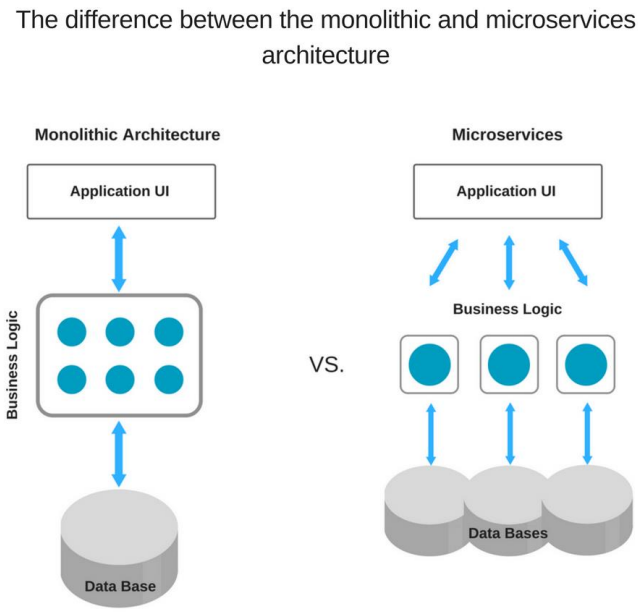

Figura 22:Arkitektura monolitike vs arkitektura mikrosherbuese [16].

Gjetjet më të fundit tregojnë se teknologjitë e lidhura me mikroshërbimet si Docker dhe Kubernetes përjetuan rritje të përdorimit në 2018, pasi ky stil arkitektural po bëhet më i

popullarizuar kohëve të fundit. Pra, çdo mikrosherbim që komunikon me bazën e të dhënave direkt përmes streams, ka një arkitekturë të tillë që lejon performancë dhe shpejtësi më të mirë të aplikacionit. ExpressJS framework rendis IBM dhe Uber në mesin e përdoruesve të saj, ndërsa restify përdoret nga npm dhe Netflix.

Si një shembull i zbatimit të drejtpërdrejtë të NodeJs është zhvendosja e Walmart në arkitekturën e mikrosherbimeve me Node.js ku rezultoi në përfitimet të menjëhershme të cilat janë:

1. Brenda natës pati 20 përqind rritje konvertimi në përgjithësi si dhe 98 përqind rritje konvertimi celular.
2. Njëqind për qind uptime në `Black Friday`, ku kishte rreth 500 milion page views.
3. Kurseni deri në 40 për qind në pajisje dhe 20-50 për qind në operacionet e përgjithshme

Një shembull tjetër i ndritshëm se si Node.js mund të tejkalojë konkurrencën për sa i përket performancës është rasti i GoDaddy. Ku drejtimin e fushatës reklamuese për SuperBowl, kompania ishte në gjendje të trajtojë 10.000 kërkesa për sekondë pa downtime, duke përdorur vetëm 10 për qind të pajisjeve hardware falë Node.js [17].

3.10.1 Shkallëzueshmëria e ExpressJS dhe NestJS

Sa i përket shkallëzueshmëris së NestJS si dhe ExpressJS, ata qëndrojn thuajse njësoj, NestJS si dhe ExpressJS është i pajtueshëm me JavaScript progresiv, i zhvilluar me Typescript me dhe përfshin elementë të këtyre tre koncepteve:

1. OOP (Programimi i orientuar në objekte)
2. FP (Programimi i bazuar ne funksione)
3. FRP (Programimi Reaktiv Funksional).

Kjo mundëson që NestJS si dhe ExpressJS të cilat janë dy framework-a shumë të shkallëzuar ku mund të krahasohen edhe Angular.js sa i përket shkallzueshmërisë. Gjithashtu, NestJS vjen me kontejnerin e tij të injektimit të varësive, si Angular. Për më tepër, NestJS e bën jetën e 18 zhvilluesve shumë më të thjeshtë dhe më të lumtur kur mban serverin shumë të testueshëm. Prandaj, kjo kornizë është shumë e shkallëzuar e njejta vlen edhe për ExpressJS pasi që NestJS është i ndërtuar mbi ExpressJS [7].

3.11 Testimi (UNIT TESTING)

Unit testing është një nivel i testimit të softuerit ku testohen njësi individuale / përbërësit e softuerit. Qëllimi është që të vërtetojmë që secila njësi e softuerit kryen punën ashtu siç është krijuar dhe planifikuar. Një njësi është pjesa më e vogël e testueshme e çdo softueri. Zakonisht ka një ose disa hyrje dhe zakonisht një dalje të vetme. Në programimin procedural, njësi mund të jetë një program individual, funksion, procedurë, etj [18]. Testimi në përgjithësi është një metodë e vlerësimit të performancës së një aplikacioni softuer për të identifikuar ndonjë defekt në program. Teston nëse softueri i prodhuar i plotëson kërkesat e specifikuara dhe njeh ndonjë defekt në softuer për rezultatet më të mira. Për të gjetur ndonjë defekt, gabim ose specifikim jo të plotë në kundërshtim me kërkesat aktuale, zhvilluesit shpesh ekzekutojnë një sistem testimi.

3.11.1 Testimi ne ExpressJS

ExpressJS nuk ofron ndonjë ambient të parazgjedhur testues, pra ato duhet të krijohen vetë si dhe struktura e tyre të definohet nga vetë zhvilluesi. Egzistojnë disa librari të cilat ofrojnë korniza testimi për ExpressJS të cilat mund të integrohen në aplikacione të reja si dhe ato egzistuese.

3.11.2 Testimi ne NestJS

Testing starter i automatizuar është një tipar dhe avantazh i madh i NestJS pasi rrit thjeshtësinë e testimit në aplikacion. NestJS krijon automatikisht një starter të aplikacioneve që vjen me një mjedis të paracaktuar testimi. Testimi i automatizuar besohet të jetë një pjesë thelbësore e çdo përpjekjeje serioze për të përmirësuar softuerin. Automatizimi bën të mundur që gjatë prodhimit të përsërisni testet individuale ose suitat e provave shpejt dhe lehtë. Kjo ndihmon që çdo zhvillim të përmbushë objektivat e cilësinë dhe efikasitetit. Si pasojë, NestJS ofron mjete të paracaktuara të testimit për të mbajtur automatikisht testet e parazgjedhura të njësive për komponentët dhe testet e2e për aplikacion.

4. METODOLGJIA

Punimi i kësaj teme të diplomës është bërë duke u bazuar kryesishtë në metodologjinë e mbledhjes së të dhënave sekondare. Të dhënat janë mbledhur duke u konsideruar nga burime të sigurta dhe literaturë përkatëse për fushën në të cilën është edhe tema. Kjo temë ka pasur për synim që të krahasojë dy NodeJS framework-a, teknologji të cilat janë shumë të ngjajshme dhe shpeshë herë është e vështirë të bëjmë diferencen ndermjet tyre pa hulumtuar në detaje mënyren e funksionimit dhe shërbimet që ato ofrojnë.

Realizimi i këtij punimi bazohet në tri metodologji:

Shqyrtimi i literaturës – ku kjo metodë përdoret për hulumtim të materialve të ndryshme për të mbledhur informata dhe hulumtime të ngjajshme. Është faza kryesore për përgaditjen e punimit.

Metoda e kërkimit – kjo metodë përdoret për testimin e të gjitha informatave të mbledhura duke bërë testimet ku bëhet përshtatja e metodologjive të një softuerit.

Metoda e mbledhjes së të dhënave sekondare – kjo metodë përdoret per grumbullimin dhe shqyrtimin e të dhënave sekondare.

5. REZULTATET

Krijimi i një aplikacion duke përdorur Node.js shpesh është mjaft i lehtë, për shkak të kompleksitetit të ulët të krijimit të një Node.js backend aplikacioni, por kjo bëhet edhe me e lehtë me përdorimin e framework-ave të tij, të cilit ne i krahasuam, por shumë shpejt, do ta gjeni veten duke bërë pyetjet e mëposhtme:

1. Pse ta zgjedhim ExpressJS?
2. Cilat janë përparsit dhe mangësit e ExpressJS?
3. Pse ta zgjedhim NestJS?
4. Cilat janë përparsit dhe mangësit e NestJs?

5.1 Pse ta zgjedhim ExpressJS?

A e dini se ka disa produkte, të cilat përdorin Express.Js për qëllimin e tyre të zhvillimit të backend?

Disan nga ato janë si më poshtë:

1. IBM
2. PayPal
3. GoDaddy
4. Walmart
5. Storylens
6. Flickr

Ka shumë më tepër produkte të tilla që përdorin Express.Js për qëllimet e tyre të zhvillimit.

Përfitimet e përdorimit të Express.js për zhvillimin e backend janë shumë të mëdha, edhe pse deri më tani, duhet të kem diskutuar disa përfitime të përdorimit të Express.JS. Por në listën më poshtë, ne do të diskutojmë disa nga përfitimet kryesore të përdorimit të Express.JS për zhvillimin e backend e ato jane:

Zgjeroni shpejt aplikacionin tuaj - përfitimi i parë i përdorimit të Express.JS për zhvillimin e backend është që ju të jeni në gjendje të shkallëzoni aplikacionin tuaj shpejt. Siç e dini që ekziston një mbështetje e Node.JS, kështu që me ndihmën e shtimit në Node.js dhe shtimin e burimeve shtesë në të, ju mund ta shkallëzoni shpejt aplikacionin tuaj në çdo mënyrë.

Zgjeroni shpejt aplikacionin tuaj - siç u përmend më lart se JavaScript është shumë i famshëm plus i lehtë për të mësuar atë gjuhë. Ju do të jeni në gjendje ta përdorni atë për qëllimet e zhvillimit në Express.JS.

Mundësia për të përdorur të njejtën gjuhë edhe në anen e klientit - një përfitim tjetër i përdorimit të ExpressJS është që ju jeni në gjendje të bëni kodin e frontendit dhe backend me ndihmën e JavaScript. Megjithëse ka disa platforma të tilla, ato ofrojnë mbështetje të ndryshme për frontend dhe backend, gjë që e bën atë mjaft sfiduese për të punuar me të. Por kurrë nuk do të përballeni me ndonjë problem të tillë me Express.JS.

Më pak kosto nga zhvilluesit për të mirëmbajtur aplikacionin - jo të gjithë e dinë që Express.JS është një JavaScript i plotë për shkak të të cilit nuk do të duhej të punësonit zhvillues të ndryshëm për menaxhimin e frontit dhe backend të një aplikacioni.

Community support - Nëse hasni ndonjëherë ndonjë problem gjatë punës me Express.JS, në atë kohë, lehtë mund të merrni ndihmë nga komuniteti shumë mbështetës dhe i gjërë që ai ka.

Mbështet caching - Express.js mbështet tiparin e memorizimit dhe avantazhi i kësaj është se nuk do të duhet të ekzekutoni përsëri kodet përsëri dhe përsëri. Për më tepër, kjo do të ndihmojë që faqet e internetit të ngarkohen më shpejt se kurrë [19].

5.2 Përparsit dhe mangësit e ExpressJS

Disa nga përparsit e ExpressJS janë:

1. Avantazhi i parë dhe më i rëndësishëm i përdorimit të Express.JS është që ju të jeni në gjendje të merrni përvojë të shpejtë të zhvillimit të aplikacionit me të.
2. Disa platforma të tilla nuk janë në gjendje të trajtojnë një nivel më të lartë të kërkesave, por me ndihmën e Express.JS, ju do të jeni në gjendje të trajtoni kërkesat në mënyrë efikase pasi ju ofron mbështetjen e trajtimit të kërkesave I / Q.
3. Express.JS ka një komunitet të gjerë, mjaft mbështetës me burim të hapur.
4. Ju do të jeni në gjendje të integroni disa aplikacione dhe shërbime të palëve të treta me Express.JS [19]

Disa nga mangësit e ExpressJS janë:

1. Disa probleme të kërkesës së klientit ballafaqohen me sistemin e softuerit të mesëm të ofruar me Express.js.
2. Disa probleme me callback methods [19].

5.3 Pse ta zgjedhim NestJS?

Ka shumë arsye që e bëjnë Nest.js të preferohet më shumë se framework-at e tjerë të NodeJS, disa prej tyre janë:

- Përputhshmëria me TypeScript
- Përkrah Monorepo
- I dizajnuar për arkitektura monolite dhe mikrosherbime
- Mbështetja e DevOps
- CLI të fuqishëme
- Mbështetje e integruar për mikrosherbimet dhe shtresat e transportit
- Container për integrim të varësive
- Ndërveprimi i lehtë me bazën e të dhënave
- Ndërfaqe e integruar për GraphQL
- Natyra përshtatëse e framework
- Zhvillimi i drejtuar nga domeni
- Struktura e dosjeve e bazuar ne Angular
- Exception filter te integruar
- Përdorimi i moduleve të palëve të treta

Këto janë disa nga pikat që qdo zhvillues do të donte që një framework të kishte, gjithashtu vlen të theksohet se NestJS ka një arkitekturë të mrekullueshme si dhe nje CLI jashtëzakonisht të fuqishme, gjithashtu edhe built-in Exception Filters dhe framework independecy janë feature të mrekullueshme të cilat NestJS i ofron [20].

5.4 Cilat janë përparsit dhe mangësit NestJS?

Disa nga përparsit e NestJS janë:

- Ka një arkitekturë të mirë menduar
- NestJS është i shkruar në Typescript
- Ka një arkitekturë solide Angular-like
- Ofron CLI e cila i lejon zhvilluesit punë më të lehtë
- Ka një përgjegjshmëri të lartë sa i përket performancës
- Shumë i zgjerueshëm
- Filluesi i tij i automatizuar i testimit rrit thjeshtësinë e testimit [7]

Disa nga mangësit e NestJS janë:

- Mungesa e dokumentimit
- Komunitet jo shume të fuqishëm ndryshe ExpressJS
- Më i vështirë të mësohet për zhvilluesit e rinjë [7]

6. DISKUTIME DHE PËRFUNDIME

Ne kemi diskutuar për Node.js nga teoria në praktikë, duke filluar me qëllimet dhe ambiciet e saj, dhe duke përfunduar me pikat e saj të forta si dhe pikat e saj të dobëta. Kur njerëzit hasin probleme me Node, kjo hapë rrugë të reja se qfarë alternative duhet të zgjedhim, por Node.js ofron shume opcione që të menjanojm problemet si dhe ta kemi më të lehtë punën me të. Më poshtë diskutohen rezultatet pozitive të implementimit të këtyre framwork-ave si rezultat i rishikimit të literaturës.

- Node.js e edhe më shume framework-at e saj e kanë bërë punën e Full Stack Developers një ëndërr të realizuar. Në mungesë të Node.js, ishte e vështirë për një zhvillues të mësonte disa gjuhë dhe mjedise të ndryshme për të menaxhuar sistemin e plotë në anën e serverit dhe atë të klientit.
- Famework-at e Node.js e bënë më të lehtë arritjen e operacioneve me ngarkesë të lartë, dhe mund të përdoren me besueshmëri në çdo fushë ku madhësitë e skedarëve janë të larta ose gjerësia e brezit të rrjetit konsumohet shumë. Node.js do t'i bëjë operacionet e tilla më të shpejta dhe do të ketë më pak nevojë për gjerësi të madhe brezit të internetit.
- Komunitetit i pëlqen tipari i tij që e njëjta gjuhë po përdoret edhe në anën e serverit edhe në anën e klientit.

Si përfundim, nga një analizë dhe vlerësim i duhur i krahasimit të hartuar, mund të konkludohet se këto dy framework-a të Node.js janë shumë të ngjajshme në funksionalitetin që na ofrojnë, por dallojnë në arkiteturen e tyre si dhe në disa opcione të ndryshme që NestJS naj ofron ndërsa tek ExpressJS gjithqka duhet të bëhet manualisht, pra mundë të themi se NestJS ofron më shumë se ExpressJS por është më i komplikuar për zhvilluesit e rinjë. Të dy këta framework mund të bëjnë të gjitha gjërat e njejta, por është qështje preference se si dëshironi të ndërtoni aplikacionin tuaj, nëse doni të shkruani gjithqka vetë dhe të keni një arkitekturë të definuar nga ju atëherë ExpressJS mund të jetë zgjidhja jote e duhur, ndërsa nëse doni një strukturë të definuar mirë, TypeScript të integruar në sistemin tuaj si dhe një CLI të fuqishme dhe jeni një zhvillues me përvojë atëher NestJS mund të jetë zgjidhja juaj e duhur.

7. REFERENCAT

[1] T. Capan, «Toptal,» [Në linjë]. Available: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js. [Qasja 2021].

[2] «Node JS,» [Në linjë]. Available: https://nodejs.org/en/about/. [Qasja 2021].

[3] «V8,» [Në linjë]. Available: https://v8.dev/. [Qasja 2021].

[4] S. K. Arora, «Hackr.io,» 05 January 2021. [Në linjë]. Available: https://hackr.io/blog/nodejs-frameworks. [Qasja 2021].

[5] Wikipedia, «Wikipedia,» 17 03 2021. [Në linjë]. Available: https://en.wikipedia.org/wiki/Express.js.

[6] A. Rimal, «Theseus,» 25 Janaury 2019. [Në linjë]. Available: https://www.theseus.fi/bitstream/handle/10024/159951/Rimal_Aashis.pdf. [Qasja 04 Februrary 2021].

[7] A. D. Pham, «Theseus,» 2020. [Në linjë]. Available: https://www.theseus.fi/bitstream/handle/10024/353200/Pham_Duc.pdf.

[8] Wikipedia, «Wikipedia,» 26 April 2021. [Në linjë]. Available: https://en.wikipedia.org/wiki/Next.js.

[9] «Express JS,» [Në linjë]. Available: https://expressjs.com/en/guide/using-middleware.html. [Qasja 2021].

[10] SEBASTIAN, «Coding The Smart Way,» 09 Februar 2020. [Në linjë]. Available: https://codingthesmartway.com/nestjs-for-absolute-beginners/. [Qasja 2021].

[11] «Expressjs,» [Në linjë]. Available: https://expressjs.com/en/guide/database-integration.html. [Qasja 15 February 2021].

[12] «Documentation | NestJS - A progressive Node.js framework,» [Në linjë]. Available: https://docs.nestjs.com/techniques/mongodb. [Qasja 05 March 2021].

[13] C. YANG, «Toptal,» [Në linjë]. Available: https://www.toptal.com/nodejs/nodejs-frameworks-comparison. [Qasja 2021].

[14] «ExpressJS,» [Në linjë]. Available: https://expressjs.com/. [Qasja 2021].

[15] S. M. A. Rahman, «Monstarlab,» 20 August 2019. [Në linjë]. Available: https://medium.com/monstar-lab-bangladesh-engineering/why-i-choose-nestjs-over-other-node-js-frameworks-6cdbd083ae67. [Qasja 2021].

[16] «Altexsoft,» 21 October 2019. [Në linjë]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-node-js-web-app-development/. [Qasja 2021].

[17] A. Kryzhanovska, «Gearheart,» 17 09 2019. [Në linjë]. Available: https://gearheart.io/blog/how-build-scalable-web-applications.

[18] S. Guru, «Medium,» 21 October 2019. [Në linjë]. Available: https://medium.com/serverlessguru/how-to-unit-test-with-nodejs-76967019ba56. [Qasja 2021].

[19] S. CM, «Techomoro,» 17 November 2020. [Në linjë]. Available: https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/. [Qasja 2021].

[20] Habileabs, «Medium,» 16 July 2020. [Në linjë]. Available: https://medium.com/habilelabs/why-choose-nest-js-over-other-node-frameworks-68a13fa1e2c8. [Qasja 2021].

[21] G. K. Campigoto, «Dev,» 07 January 2020. [Në linjë]. Available: https://dev.to/giovannikleincampigoto/nodejs-express-vs-nestjs-a-vision-about-architecture-and-good-pratices-571i. [Qasja 2021].

[22] «MDN Web Docs,» [Në linjë]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose. [Qasja 14 January 2021].

[23] S. M. Islam, «Medium,» 4 January 2018. [Në linjë]. Available: https://medium.com/@zibon/getting-started-with-nodejs-and-expressjs-2018-51689dae024b.

[24] H. S. &. T. R. Soomro, «Global Journals,» June 2017. [Në linjë]. Available: https://globaljournals.org/GJCST_Volume17/8-Node-Js-Challenges-in-Implementation.pdf. [Qasja 2021].