UNIVERSITETI I MITROVICËS "ISA BOLETINI"

FAKULTETI I INXHINIERISË MEKANIKE DHE KOMPJUTERIKE

DEPARTAMENTI: INFORMATIKË INXHINIERIKE

PUNIM DIPLOME

Mentori
MSc. Halil Sadiku, PhD Cand.

Kandidati
Egzon Murati

Mitrovicë, 2021

UNIVERSITETI I MITROVICËS “ISA BOLETINI”

FAKULTETI I INXHINIERISË MEKANIKE DHE KOMPJUTERIKE

DEPARTAMENTI: INFORMATIKË INXHINIERIKE

PUNIM DIPLOME

Zhvillimi i aplikacioneve për vlerësimin e përformancës së bazave të të dhënave SQL dhe NoSQL

Mentori
MSc. Halil Sadiku, PhD Cand.

Kandidati
Egzon Murati

Mitrovicë, 2021

Abstrakt

Bazat e të dhënave relacionale dhe jo-relacionale janë lloje konkuruese të modeleve të bazave së të dhënave. E para ka ekzistuar që nga viti 1979 dhe e dyta që nga viti 2000. Zgjedhja e një modeli të duhur të bazës së të dhënave për t'u përdorur është një vendim i rëndësishëm që zhvilluesit duhet të marrin bazuar në tiparet e një modeli të dhënë të bazës së të dhënave. Kërkesat e aplikacioneve moderne e kanë bërë NoSQL një bazë të dhënash të popullarizuar të zgjedhur.

Ky punim paraqet një krahasim studimi për performancën e bazave të të dhënave relacionale dhe atyre jo-relacionale. Vlerësimi i përformancës së bazave të të dhënave është bërë përmes zhvillimit të një aplikacioni i cili do të vendosë komunikimin me dy bazat e të dhënave të përmendura më lartë. Përmes këtij aplikacioni, janë implementuar operacionet e leximit, krijimit, fshirjes dhe azhurnimit në objekte të caktuara të bazave të të dhënave. Rezultatet e fituara tregojnë për një përformancë më të mirë të bazave të të dhënave jo-relacionale krahasuar me ato relacionale, mirëpo zgjedhja e bazës së të dhënave për t'u përdorur varet nga natyra e aplikacionit që do të zhvillohet sepse secili lloj i bazës së të dhënave ka sfidat, mangësitë dhe pikat e tij të forta.

Fjalët kyçe: SQL, NoSQL, MongoDB, API, CRUD

Falenderim

Thellësisht falenderues ndaj familjes për mbështjen e vazhdueshme, shoqërisë për ndihmën dhe bashkëpunimin dhe stafit akademik për mbështetjen dhe këshillat e duhura që më kanë ofruar gjatë këtyre tri viteve të studimeve. Përkushtimi dhe suksesi im i atribuohen të gjithë atyre që kanë qendruar afer meje gjatë këtij rrugëtimi në hedhjen e këtij hapi të rëndësishem të nivelit tim akademik dhe profesional. Falenderoj në veçanti mentorin tim të nderuar MSc. Halil Sadiku për punën, këshillat, rekomandimet dhe perkrahjen e vazhdueshme gjatë tërë kësaj kohe.

DEKLARATË E ORIGJINALITETIT

Me anë të kësaj deklarate dëshmojë që ky punim është punë origjinale dhe nuk është botuar më herët në ndonjë universitet apo institucion tjetër akademik. Gjithashtu, deklaroj që punimi nuk përmban material të botuar apo shkruar nga ndonjë përson tjetër, përveç siç deklarohet në përmbajtjen e këtij teksti.
Po ashtu deklarojë që kam respektuar në përpikëri rregullat themelore akademike të Universitetit të Mitrovicës “Isa Boletini” për punimin e kësaj teme të diplomes.

_____

Nënshkrimi

Tabela e Permbajtjes

1 Hyrje

Përderisa disa vite më parë, një aplikacion zakonisht kishte mijëra përdorues deri në dhjetëra mijëra përdorues në rastin më ekstrem, aktualisht ka shumë aplikacione që kanë miliona përdorues dhe të cilët janë të lidhur 24/7, 365 ditë në vit. Është e rëndësishme të përdorni një bazë të dhënash të përshtatshme, e cila mbështet lidhjen e njëkohshme të qindra mijëra përdoruesve.

Një bazë e të dhënave relacionale është një koleksion i të dhënave me relacione të paracaktuara ndërmjet tyre. Këto të dhëna organizohen në formë të tabelave me kolona dhe rreshta. Tabelat përdoren për të ruajtur informacione në lidhje me objektet që do të përfaqësohen në bazën e të dhënave. Çdo kolonë në një tabelë mban një lloj të caktuar të të dhënave dhe një fushë ruan vlerën aktuale të një atributi. Rreshtat në tabelë paraqesin një koleksion të vlerave të lidhura me një objekt ose entitet. Çdo rresht në një tabelë mund të shënohet me një numer identifikues të quajtur çelës primar, dhe rreshtat midis tabelave të shumta mund të lidhen duke përdorur çelësat e huaj (foreign key).

Këto baza të të dhënave përdoren gjerësisht në shumicen e aplikacioneve dhe ato kanë përformancë të mirë kur ato trajtojnë një sasi të kufizuar të të dhënave mirëpo për të trajtuar vëllim më të madh të të dhënave si interneti, multimedia dhe mediat sociale, përdorimi i bazave të të dhënave tradicionale relacionale është joefikas. Për ta zgjidhur këtë problem u prezantua termi “NoSQL”. Termi NoSQL u shpik nga Carlo Strozzi në 1998 dhe i referohet bazave të të dhënave jo-relacionale, term i cili më vonë u përhap në vitin 2009 nga Eric Evans [1]. Kohëve të fundit, termi ka marrë një kuptim tjetër, të quajtur “Jo vetëm SQL” (Not only SQL) i cili është një variant më i butë i përcaktimit të termit, krahasuar me atë të mëparshëm.

NoSQL, është një metodologji e përbërë nga disa mjete plotësuese dhe konkurruese. Përparësia kryesore e një baze të dhënash NoSQL është se, ndryshe nga një bazë e të dhënave relacionale, ajo mund të trajtojë të dhëna të pastrukturuara si dokumente, emaile, multimedia dhe media sociale në menyrë efikase. Bazat e të dhënave jo-relacionale nuk përdorin parimet e bazave të të dhënave relacionale dhe nuk i ruajnë të dhënat në tabela, skema nuk është fikse dhe kanë model shumë të thjeshtë të të dhënave. Në vend të kësaj, ato përdorin çelësa identifikimi dhe të dhënat mund të gjenden bazuar në çelësat e caktuar. NoSQL i referohet bazave të të dhënave me përformancë të lartë, jo-relacionale që përdorin larmi të gjerë të modeleve të të dhënave. Këto baza të të dhënave ruajnë të gjitha të dhënat e tyre në dokumente JSON, që nënkupton “Javascript Object Notation”, të cilat janë njësi të plota që dikush mund t’i lexojë dhe t’i kuptojë me lehtësi.

Modeli JSON është model shumë i thjeshtë, që mundëson ruajtjen dhe transmetimin e të dhënave të pastrukturuara. Duke përdorur një sintaksë të thjeshtë, mund të ruani lehtësisht çdo gjë, duke filluar nga një numër deri tek tekstet, vargjet apo objektet e ndryshme. Gjithashtu ju mund të përdorni vargje (array) apo objekte të ndërthurura, duke ju lejuar të krijoni struktura komplekse të të dhënave.

2 Teknologjitë e përdorura

Si raste studimi të këtij punimi janë zhvilluar tri aplikacione për vlerësimin e performancës së bazave të të dhënave relacionale dhe jo-relacionale. Aplikacionet e zhvilluara janë:

- Patient Management System (Ueb aplikacion)
- SQL Web API
- MongoDB Web API

Këto aplikacione mundësojnë realizimin e operacioneve CRUD (create, read, update, delete) bazuar në numrin e rreshtave zgjedhur nga përdoruesi. Për zhvillimin e këtij projekti janë shfrytëzuar teknologji dhe vegla të ndryshme, duke filluar nga teknologjitë për zhvillimin e ueb faqes, bazave të të dhënave për ruajtjen e të dhënave dhe zhvillimin e API-ve për dergimin apo marrjen e të dhënave. Në vazhdim të këtij kapitulli do të sqarohet se cilat janë ato teknologji të përdorura për zhvillimin e këtij projekti dhe për mundësitë që ato teknologji i ofrojnë.

2.1 React

React (i njohur gjithashtu si React.js ose ReactJS) është një librari e JavaScript, për ndërtimin e ndërfaqeve të përdoruesit ose UI (user-interface) komponenteve. Mirëmbahet nga Facebook dhe një komunitet i zhvilluesve dhe kompanive individuale. React mund të përdoret si bazë në zhvillimin e ueb-aplikacioneve ose aplikacioneve mobile. Sidoqoftë, React ka të bëjë vetëm me dhënien e të dhënave në DOM (document object model), kështu që krijimi i aplikacioneve kërkon përdorimin e librarive shtesë për menaxhim të gjendjës (ang. state) dhe linjave [2][3].

Sistemet e JavaScript-it po zhvillohen me një ritëm jashtëzakonisht të shpejtë kohëve të fundit, që do të thotë se kohë pas kohë zhvillohen versione të reja të React-it, Angular-it dhe Vue.js-it. Ne analizuam numrin e pozitave të hapura të punës në të gjithë botën që kërkojnë një njohuri specifike të një sistemi të caktuar. Si burim, u shfrytëzua ueb faqja “Indeed.com” dhe morëm shpërndarjen vijuese sipas më shumë se 60,000 ofertave të punës.

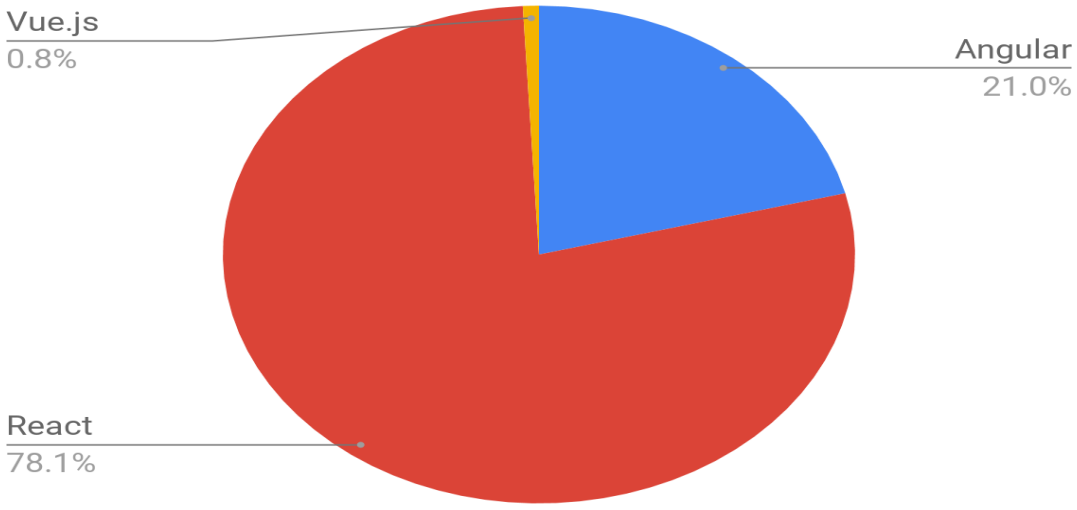

Figure 1. React krahasuar me Angular dhe Vue.js sipas 60.000 pozitave të punës në Indeed.com [5].

Kodi në React përbëhet nga entitetet e quajtura komponente. Komponentët mund t'i jepën një elementi të veçantë në DOM duke përdorur librarinë React DOM. Kur paraqitet një komponentë, në të mund të vendosim vlera të cilat njihen si "props" [4]. Dy mënyrat kryesore të deklarimit të komponenteve në React janë përmes komponenteve të bazuara në funksione dhe komponenteve të bazuara në klasë.

Një tjetër tipar është përdorimi i një modeli virtual të objektit të dokumentit, ose DOM virtual. React krijon një memorie të brendshme të strukturës së të dhënave, llogarit ndryshimet që rezultojnë dhë pastaj azhurnon në mënyrë efikase DOM-in e shfaqur të shfletuesit. Kjo i lejon programuesit të shkruaj kod sikur e gjithë faqja azhurnon ndryshimet në çdo ndryshim të kodit, ndërsa libraritë e React-it editojnë vetëm ndryshimet në nën-komkomponentët që ndryshojnë në të vërtetë. Ky azhurnim selektiv i ndryshimeve siguron një rritje të madhe të përformancës. Kursen përpjekjen e rikalkulimit së stilit CSS, paraqitjen për faqe dhe azhurnimin për të gjithë faqen.

Disa prej avantazheve të React-it janë:

- E lehtë për t'u mësuar në sajë të dizajnit të thjeshtë dhe përdorimit të JSX (një sintaksë e ngjashme me HTML).
- Zhvilluesit kalojnë më shumë kohë duke shkruar kod modern të JavaScript-it, dhe më pak kohë duke u shqetësuar për kodin specifik të librarisë.
- Jashtëzakonisht i shpejtë, duke marrë parasysh implementimin e DOM-it virtual dhe optimizimeve të ndryshme për shfaqje të rezultateve.
- Mbështetje të madhe për komunikim me anën e serverit, duke e bërë atë një librari të fuqishme për aplikacione të bazuara në përmbajtje.
- Lidhja e të dhënave është njëkahëshe, që do të thotë më pak efektë anësotë të padëshiruara.
- React zbaton konceptet e "programimit funksional", duke krijuar një kod të lehtë për t'u testuar dhe shumë të ripërdorshëm.
- Aftësitë e mësuara në React mund të zbatohen për zhvillim të aplikacioneve mobile në React Native.
-

Disavantazhet e React-it:

- React është i paopinionuar dhe i lë zhvilluesit të bëjnë zgjedhje për menyrën më të mirë të zhvillimit. Kjo mund të trajtohet nga udhëheqja e projektit dhe proceset tjera.
- Komuniteti është i ndarë në mënyrën më të mirë për të shkruar CSS në React, në mës të "CSS Modules" dhe "CSS-in-JS".
- React po largohet nga komponentet e bazuara në klasa, gjë që mund të jetë një pengesë për zhvilluesit e programimit të orientuar në objekte (OOP).
- Përzierja e modelimit me logjikën (JSX) mund të jetë konfuze për disa zhvillues fillestarë.

2.2 CSS

Cascading Style Sheets (CSS) është një gjuhë stilizuese që përdoret për të përshkruar prezantimin e një dokumenti të shkruar në një gjuhë markuese siç është HTML [6]. Ajo është krijuar për të mundësuar ndarjen e prezantimit dhe përmbajtjes, duke përfshirë paraqitjen, ngjyrat dhe shkronjat. Kjo ndarje mund të përmirësojë aksesueshmërinë e përmbajtjes, të sigurojë më shumë fleksibilitet dhe kontroll në specifikimin e karakteristikave të prezantimit, të mundësojë që faqe të shumta në internet të ndajnë formatimin duke specifikuar CSS përkatëse në një skedar të veçantë .css i cili zvogëlon kompleksitetin dhe përsëritjen e përmbajtjes strukturore dhe gjithashtu mundëson që skedari .css të memorizohet për të përmirësuar shpejtësinë e ngarkimit të faqes midis faqeve që ndajnë skedarin dhe formatimin e tij.

Ndarja e formatimit dhe përmbajtjes gjithashtu e bën të mundshme paraqitjen e të njëjtës faqe markuese në stile të ndryshme për metoda të ndryshme të dhënies, të tilla si në ekran, në shtyp dhe me zë (përmes shfletuesit të bazuar në fjalim ose lexuesit të ekranit).

CSS ka një sintaksë të thjeshtë dhe përdor një numër fjalësh kyçe në anglisht për të specifikuar emrat e vetive të stileve të ndryshme. Një fajll “css” përbëhet nga një listë e rregullave. Çdo rregull ose grup rregullash përbëhet nga një ose më shumë përzgjedhës (selectors) dhe një bllok deklarimi.

Në CSS, përzgjedhësit deklarojnë se në cilën pjesë të shënjimit zbatohet një stil duke përputhur etiketat dhe atributet në vetë shënimin.
Përzgjedhësit mund të aplikohen në:

- të gjithë elementët e një lloji specifik, p.sh. titrat e nivelit të dytë h2
- elementet e specifikuara përmes atributeve, si psh:
    - id: një identifikues unik brenda dokumentit
    - klasa: një identifikues që mund të shënojë elemente të shumta në një document
- elemente në varësi të mënyrës së vendosjes së tyre në krahasim me të tjerët në dokument

Klasat dhe ID-të janë të ndjeshme në shkronja të vogla, fillojnë me shkronja dhe mund të përfshijnë karaktere alfanumerike, vizë lidhëse dhe nënvizime. Një klasë mund të zbatohet për çdo numër të instancave të cilitdo element. Një ID mund të zbatohet vetëm për një element të vetëm.

2.3 ASP.NET Web API

Në programimin kompjuterik, krijimi, leximi, editimi dhe fshirja (CRUD) janë katër funksionet themelore të ruajtjes së vazhdueshme. Fjalët alternative përdoren ndonjëherë kur përcaktojnë katër funksionet themelore të CRUD, të tilla si rikuperimi në vend të leximit, modifikimi në vend të editimit ose shkatërrimi në vend të fshirjes. CRUD ndonjëherë përdoret gjithashtu për të përshkruar konventat e ndërfaqes së përdoruesit që lehtësojnë shikimin, kërkimin dhe ndryshimin e informacionit, shpesh duke përdorur forma dhe raporte të bazuara në kompjuter.

Transferimi i gjendjes përfaqësuese (REST) është një stil arkitektural i softuerit që përcakton një sërë kufizimesh që do të përdoren për krijimin e shërbimeve të internetit. Shërbimet e internetit që përputhen me stilin arkitektural të REST, të quajtura "RESTful Web" shërbime, sigurojnë ndërveprim midis sistemeve kompjuterike në internet. Shërbimet RESTful të uebit lejojne që sistemet kërkuese të kenë qasje dhe të manipulojnë përfaqësimet tekstuale të burimeve të uebit duke përdorur një grup të njëtrajtshëm dhe të paracaktuar të operacioneve pa gjendje.

Në një shërbim RESTful të uebit, kërkesat e bëra në URI (Uniform Resource Identifier) të një burimi do të sjellin një pergjigje me një ngarkesë të formatuar në HTML, XML, JSON ose ndonjë format tjetër. Përgjigja mund të konfirmojë që është bërë ndonjë ndryshim në gjendjen e burimit dhe përgjigjja mund të sigurojë linqe që lidhin me burime të tjera. Kur përdoret HTTP, operacionet (metodat) të disponueshme janë: GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS dhe TRACE [7].
Duke përdorur operacionet standarte të protokollit, këto sisteme synojnë performancë të shpejtë, besueshmëri dhe aftësi për t'u rritur duke ripërdorur komponentët që mund të menaxhohen dhe azhurnohen pa ndikuar sistemin në tërësi, edhe kur është duke funksionuar.

ASP.NET Web API është një sistem që e bën të lehtë ndërtimin e shërbimeve HTTP që arrijnë një gamë të gjerë klientësh, përfshirë shfletuesit dhe pajisjet mobile. Ky sistem është një platformë ideale për ndërtimin e aplikacioneve "RESTful" në .NET Framework. Kur ndërtoni API në ueb, ka disa mënyra se si mund të ndërtoni API. Këto mënyra përfshijnë HTTP/RPC, që nënkupton përdorimi i HTTP për të përdorur (thirrur) metoda ne të gjithë uebin.

Tabela e mëposhtme tregon se si përdoren metodat HTTP në API, duke përfshirë ato RESTful.

Metoda	Përshkrimi
GET	Merr një përfaqësim të gjendjes së burimit të synuar
POST	Lejon burimin e synuar të përpunojë përfaqësimin e bashkangjitur në kërkesë.
PUT	Vendos gjendjen e burimit të synuar në gjendjen e përcaktuar nga përfaqësimi i bashkangjitur në kërkesë.
DELETE	Fshin gjendjen e burimit të synuar.

Table 1. Metodat HTTP të përdorura në API

2.3.1 Principet e arkitekturës REST

Arkitektura server-client
Parimi që qëndron pas arkitekturës klient-server është ndarja e problemeve. Ndarja e ndërfaqes së përdoruesit nga ruajtja e të dhënave përmirëson transportueshmërinë e asaj

ndërfaqeje në shumë platforma. Ai gjithashtu ka avantazhin që përbërës të ndryshëm mund të zhvillohen në mënyrë të pavarur nga njëri-tjetri.

Pa-gjendje
Pashtetësia do të thotë që komunikimi midis klientit dhe serverit gjithmonë të përmbajë të gjithë informacionin e nevojshëm për të ekzekutuar kërkesën. Nuk ka gjendje të sesionit në server, ajo ruhet tërësisht tek klienti. Nëse qasja në një burim kërkon vërtetim, klienti duhet të vërtetojë vetveten për secilën kërkesë.

Fshehja
Klienti, serveri dhe çdo përbërës i ndërmjetëm mund të cache të gjitha burimet për të përmirësuar performancën. Informacioni mund të klasifikohet si i fshehtë ose jo i fshehtë.

Ndërfaqja uniforme
Të gjithë përbërësit e një REST API duhet të ndjekin të njëjtat rregulla për të komunikuar me njëri-tjetrin. Kjo gjithashtu e bën më të lehtë për të kuptuar ndërveprimet midis përbërësve të ndryshëm të një sistemi.

Sistemi me shtresa
Komponentët individuale nuk mund të shohin përtej nivelit të menjëhershëm me të cilin bashkëveprojnë. Kjo do të thotë që një klient që lidhet me një përbërës të ndërmjetëm siç është një përfaqësues nuk e di se çfarë fshihet pas tij. Prandaj, përbërësit mund të shkëmbehen ose zgjerohen lehtësisht në mënyrë të pavarur nga njëri-tjetri.

Kodi sipas kërkesës
Kodi shtesë mund të shkarkohet për të zgjeruar funksionalitetin e klientit. Sidoqoftë, kjo është opsionale sepse klienti mund të mos jetë në gjendje ta shkarkojë ose ekzekutojë këtë kod.

2.4 Microsoft Visual Studio

Visual Studio është një mjedis i integruar zhvillimi (IDE) nga Microsoft. Përdoret për të zhvilluar programe kompjuterike, gjithashtu edhe ueb aplikacione, sherbime uebi dhe aplikacione mobile. Visual Studio përdor platforma të zhvillimit të softuerit Microsoft si Windows API, Windows Forms, Windows Presentation Foundation, Windows Store dhe Microsoft Silverlight.

Visual Studio përfshin një redaktues kodi mbështetës që quhet IntelliSense (përbërësi i perfundimit të kodit) si dhe ri-faktorizimin e kodit. Rregulluesi (ang. debugger) i integruar funksionon si një korrigjues i nivelit burimor ashtu edhe një korrigjues i nivelit të makinës. Vegla të tjera të integruara përfshijnë një profilues të kodit, projektues për ndërtimin e aplikacioneve GUI, krijues të uebfaqeve, stilist të klasave dhe dizajner të skemës së bazës së të dhënave. Ajo pranon shtojca që zgjerojnë funksionalitetin në pothuajse çdo nivel – përfshirë shtimin e mbështjes për sistemet e kontrollit të burimit (si Subversion dhe Git) dhe shtimin e mjeteve të reja si redaktorët dhe dizajnerët vizualë për gjuhët specifike të domenit ose mjetet për aspektet e tjera të zhvillimit të softuerit.

Visual Studio mbështët 36 gjuhë të ndryshme programimi dhe lejon redaktuesin dhe korrigjuesin e kodeve të mbështesin (në shkalle të ndryshme) gati çdo gjuhë programimi, me kusht që të ekzistojë një shërbim specifik i gjuhës. Gjuhët e integruara përfshijnë C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML dhe CSS. Ndihma për gjuhë të tjera si Python, Ruby dhe Node.js është në dispozicion përmes shtojcave. Java (dhe J#) janë mbështetur në të kaluarën.

Edicioni më themelor i Visual Studio-s, edicioni i komunitetit, është në dispozicion falas. Slogani për edicionin e komunitetit të Visual Studio-s është "IDE falas, me tipare të plota për studentët, zhvilluesit dhe individët".

2.5 Visual Studio Code

Visual Studio Code është një redaktues falas i kodit burimor i bërë nga Microsoft për Windows, Linux dhe macOS [8]. Ky editor mund të përdoret me një larmi gjuhësh programimi, duke përfshirë Java, JavaScript, Go, Node.js dhe C++. Bazohet në kornizën Electron, e cila përdoret për të zhvilluar ueb aplikacione Node.js që funksionojnë në motorin e paraqitjes Blink. Visual Studio Code përdor të njëjtin komponent të redaktuesit (i koduar "Monaco") i përdorur në Azure DevOps (më parë i quajtur Visual Studio Online dhe Visual Studio Team Services). Karakteristikat përfshijnë mbështetje për korrigjimin e gabimeve, theksimin e sintaksave, përfundimin intelegjent të kodit, fragmente, riformimin e kodit dhe Git të bashkëngjitur. Përdoruesit mund të ndryshojnë ambientin, shkurtoret e tastierës, preferencat dhe të instalojnë vegla shtesë që shtojnë funksione të ndryshme.

Në vend të një sistemi projekti, ai lejon përdoruesit të hapin një ose më shumë dosje (ang. directories), të cilat më pas mund të ruhen në hapësirat e punës për ripërdorim në të ardhmen. Kjo e lejon atë të funksionojë si një redaktues agnostik i kodit të çfarëdo gjuhë programuese.

Në studimin e zhvilluesve të "Stack Overflow 2019", VS-Code u rendit mjeti më i popullarizuar i mjedisit për zhvilluesit, me 50.7% të 87.317 të anketuarve që raportuan se e përdorin atë [9].

2.6 SQL Server Management Studio

SQL Server Management Studio (SSMS) është një aplikacion softuerik i lansuar me Microsoft SQL Server 2005 që përdoret për konfigurimin, menaxhimin dhe administrimin e të gjithë komponenteve brenda Microsoft SQL Server. Veglat përfshijnë editorët skriptues si dhe veglat grafike që punojnë me objektet dhe tiparet e serverit [10]. SSMS mbështet shumicën e detyrave administrative të SQL Server dhe mban një mjedis të vetëm, të integruar për menaxhimin dhe shkrimin e bazës së të dhënave SQL Server.

Një tipar qendror i SSMS është Object Explorer, i cili lejon përdoruesin të shfletojë, zgjedhë dhe veprojë mbi ndonjë prej objekteve brenda serverit. Ai gjithashtu dergoi një botim të veçantë Express që mund të shkarkohej pa pagesë, megjithatë versionet e fundit të SSMS janë plotësisht të afta të lidhen dhe të menaxhojnë çdo instancë të SQL Server Express-it. Microsoft gjithashtu përfshiu kompatibilitetin për versione të vjetra të SQL

Server-it, duke lejuar kështu një version më të ri të SSMS te lidhet me versionet e vjetra të instancave.

Në ilustrimin vijues, tregohet dritarja e cila shfaqet për të realizuar lidhjen në SQL Server e cila kerkon të dhëna si tipi i serverit, emri i serverit dhe detajet e autentifikimit.

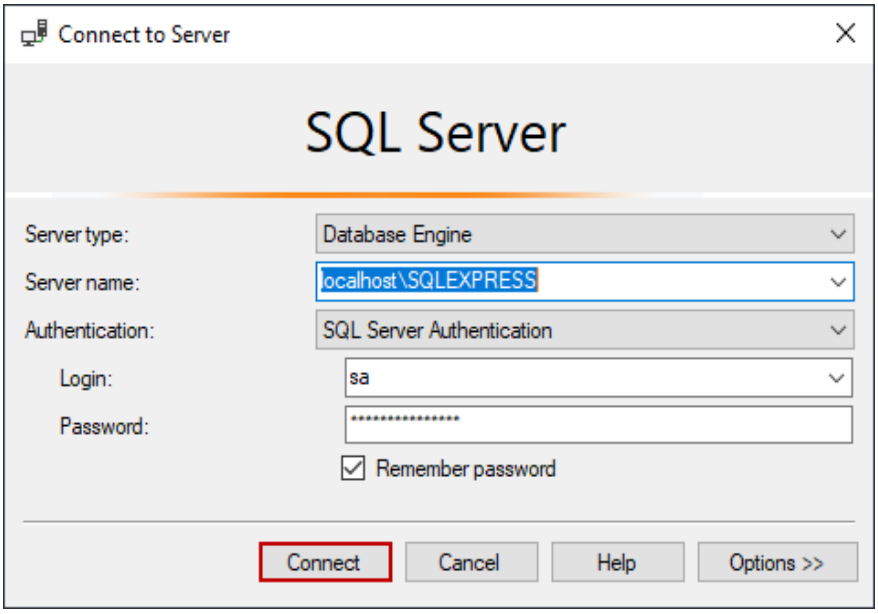

Figure 2. Lidhja me SQL Server

Pas një lidhje të suksesshme, paneli Object Explorer do të shfaqet në anën e majtë të dritares kryesore të aplikacionit. Kjo dritare na mundëson të kryejme operacione të lidhura me bazën e të dhënave si krjimi, mirëmbajtja dhe manipulimi i objekteve të databazave.

Përveq kësaj, SSMS mundëson krijimin dhe ekzekutimin e pyetësorëve T-SQL duke klikuar në butonin 'New Query' i cili gjendet në shiritin e veglave.

2.7 Robo 3T

Robo 3T është një GUI falas dhe i lehtë për MongoDB. Është një vegël e menaxhimit të MongoDB i cili ka një platformë multi-funksionale dhe mbështetet nga JSON d.m.th JavaScript Object Notation. Ky mjet nuk është tipik për vegla tjera administrative të ndërfaqes së përdoruesit të MongoDB, dmth ndërfaqja e tij mund të bashkëngjitet në Mongo Shell me një qasje të tërë si ne Mongo CLI ashtu edhe në Mongo GUI.

Me ndihmën e kësaj vegle, një përdorues mund të shikojë, modifikojë dhe fshijë dokumentet. Për më tepër, Robo 3T është një softuer me burim të hapur dhe është krejtësisht pa pagesë për përdoruesit. Mbi të gjitha, përdoruesit e Robo 3T nuk kanë nevojë të kalojnë nëpër procedurën e përdorimit të tabelave dhe rreshtave, e cila zakonisht përdoret në bazën e të dhënave relacionale. Ndryshe nga ato, kjo vegël është e ndërtuar mbi arkitekturën e koleksioneve dhe dokumenteve.
Robo 3T jo vetëm që analizon semantikën e kodit, por gjithashtu e ekzekuton atë në një makinë virtuale JavaScript, duke ofruar një auto-kompletim gjatë kohës së ekzekutimit

që është e pamundur të arrihet në mënyrë statike. Të gjitha operacionet që kryhen në MongoDB bëhen në mënyrë asinkrone dhe nuk bllokojnë punën e aplikacionit. Kjo e bën Robo 3T më të shpejtë si dhe më të lehtë për përdorim. Figura në vijim shfaq në detaje pamje të ambientit të aplikacionit softuerik Robo 3T.

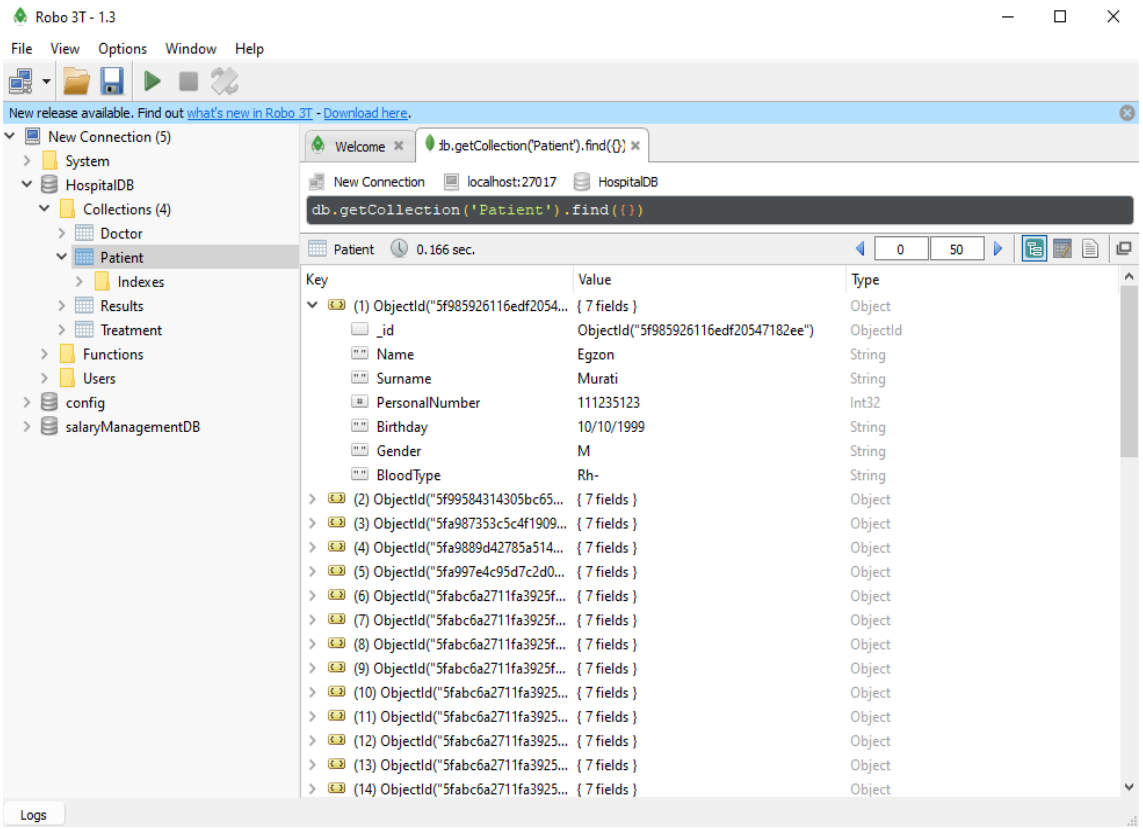

Figure 3. Pamja e aplikacionit Robo 3T

3 Bazat e të dhënave SQL dhe NoSQL

Në ndonjë moment gjatë karrierës tuaj si një zhvillues softuerik, do t'ju duhet të vendosni se cili lloj i bazës së të dhënave është më i përshtatshmi për aplikacionin tuaj. Aplikacionet softuerike përdorin baza të të dhënave relacionale (SQL) ose jo relacionale (NoSQL) për ruajtjen e vazhdueshme të të dhënave. Qëllimi i këtij kapitulli është të sigurojë një përmbledhje të thjeshtë, të nivelit të lartë të bazave të të dhënave relacionale dhe jo-relacionale, të nxjerrë në pah ndryshimet e tyre dhe të diskutojë skenarët në të cilët duhet të përdoret secila, dhe të shqyrtojë shkurtimisht sistemet e përbashkëta të menaxhimit të bazës së të dhënave për secilën.

3.1 Bazat e të dhënave SQL

Që nga ditët e hershme të kompjuterëve, ruajtja dhe manipulimi i të dhënave kanë qenë fokusi kryesor i aplikimit. Sistemi i menaxhimit të bazës së të dhënave (DBMS) i parë me qëllim të përgjithshëm, i projektuar nga Charles Bachman në fillim të viteve 1960, u quajt "Integrated Data Store". Ai formoi bazën për modelin e të dhënave të rrjetit, i cili u standardizua nga Konferenca për gjuhët e sistemeve të të dhënave (CODASYL) dhe ndikoi fuqishëm në sistemet e bazës së të dhënave gjatë viteve 1960.

Në fund të viteve 1960, IBM zhvilloi DBMS të sistemit të menaxhimit të informacionit (IMS), që është i perdorur edhe në ditët e sotme. IMS formoi bazën për një sistem alternativ të përfaqësimit të të dhënave të quajtur modeli hierarkik i të dhënave.

Termi "bazë e të dhënave relacionale" u shpik nga E.F.Codd ne IBM në 1970 [11]. Codd prezantoi këtë term në punimin e tij shkencor "A relational Model of Data for Large Shared Data Banks". Në këtë punim ai percaktoi atë që donte të thoshte me "relacionale". Një përkufizim i njohur i asaj që përbën një sistem të dhënash relacionale është i përbërë nga 12 rregullat e Codd. Sidoqoftë, asnjë zbatim komercial i modelit relacional nuk përputhet me të gjitha rregullat e Codd, kështu që termi gradualisht është ndryshuar për të përshkruar një klasë më të gjerë të sistemeve të bazës së të dhënave, e cila ne minimum:

- Paraqet të dhënat te perdoruesi si relacione (prezantim në formë tabele, dmth. si nje koleksion tabelash me secilën tabelë të përbërë nga një grup rreshtash dhe kolonash).
- Ofron operatorë relacionalë për të manipuluar të dhënat në formë tabele.

Sistemi softuerik që përdoret për t`i mirëmbajtur bazat e të dhënave është sistemi relacional i menaxhimit të bazës së të dhënave (DBMS). Perkufizimi më i zakonshëm i një RDBMS është një produkt që paraqet një pamje të të dhënave si një koleksion rreshtash dhe kolonash. Shumë sisteme relacionale të bazës së të dhënave kanë mundësi të përdorimit të SQL (gjuha e strukturuar e pyetësorëve) për pyetësorë dhe mirëmbajtjen e bazës së të dhënave [12].

Modeli relacional organizon të dhënat në një ose më shumë tabela (ose "relacione") të kolonave dhe rreshtave, me një çelës unik që identifikon secilin rresht. Rreshtat quhen ndryshe edhe si rekorde ndërsa kolonat si atribute. Në përgjithësi, secila tabelë/relacion

përfaqëson një “entitet” dhe kolonat përfaqësojnë vlerat e atribuara të asaj instance. Çdo rresht në një tabelë ka çelësin e vet unik. Rreshtat në një tabelë mund të lidhen me rreshtat në tabelat e tjera duke shtuar një kolonë për çelësin unik të rreshtit të lidhur (kolona të tilla njihen si çelësa të huaj). Në vijim është paraqitur një figurë ku shihet se si mund të krijohet një tabelë në bazën e të dhënave SQL Server si dhe ruajtja e të dhënave në atë tabelë.

```
CREATE TABLE [dbo].[Patients](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [nvarchar](30) NULL,
    [Surname] [nvarchar](30) NULL,
    [PersonalNumber] [bigint] NULL,
    [Birthday] [date] NULL,
    [Gender] [nvarchar](1) NULL,
    [BloodType] [nvarchar](3) NULL,
 CONSTRAINT [PK_Patient] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
```

100 %

Results | Messages

	Id	Name	Surname	PersonalNumber	Birthday	Gender	BloodType
1	11	John	Williams	123459632312	1980-02-16	M	0-
2	12	Richard	Brown	112845638921	1967-08-01	M	AB+
3	13	William	Miller	681270123512	1986-12-11	M	A-
4	14	Elizabeth	Taylor	123617633485	1989-04-25	F	B+

Figure 4. Krijimi i tabeles ne SQL Server

Një pjesë e këtij shqyrtimi përfshin të qenit vazhdimisht në gjendje të zgjedhësh ose modifikosh një dhe vetëm një rresht në një tabelë. Prandaj, shumica e implementimeve kanë një çelës primar (PK) për secilin rresht në një tabelë. Kur një rresht i ri shtohet në tabelë, gjenerohet një vlerë e re unike për çelësin kryesor. Çelësa të tjerë, gjithashtu mund të idendifikohen dhe përcaktohen si çelësa alternativ (AK). Shpesh nevojiten disa kolona për të formuar një AK (kjo është një arsye pse një kolonë e vetme zakonisht bëhet PK). Si PK-të ashtu edhe AK-të kanë aftësinë të identifikojnë në mënyrë unike një rresht brenda një tabele.
Çelësat kryesorë brenda një baze të të dhënash përdoren për të përcaktuar relacionet midis tabelave. Kur një PK migron në një tabelë tjetër, ai bëhet një çelës i huaj në tabelën tjetër. Kur çdo qeli mund të përmbajë vetëm një vlerë dhe PK migron në një tabelë të rregullt, ky model mund të përfaqësojë një marrëdhënie “një me një” ose “një me shumë”. Shumica e bazave të të dhënave zgjidhin marrëdhëniet “shumë me shumë” duke krijuar një tabelë shtesë që përmban PK nga të dy tabelat tjera.

Që një bazë e të dhënave relacionale të jetë efektive, të dhënat që ruhen në të duhet të strukturohen në një mënyrë shumë të organizuar. Një skemë e mire-dizajnuar minimizon tepricën e të dhënave dhe parandalon që tabelat të mos jenë të sinkronizuara, një tipar kritik për shumë biznese, veçanërisht ato që regjistrojne transaksione financiare. Një skemë e dizajnuar dobët mund të rezultojë në shume probleme për shkak të jo-

fleksibilitetit. Për shembull, një kolonë e krijuar për të ruajtur numrat e telefonit mund të kërkojë 9 shifra sepse ky është standarti për numrat e telefonit në Kosovë. Kjo ka avantazhin e refuzimit të çdo vlere të pavlefshme (për shembull, nëse një numri i mungon një kod zone). Sidoqoftë, nëse keni nevojë të ndryshoni skemën (për shembull, nëse duhet të përfshini një numër të telefonit ndërkombëtar me më shumë se 9 shifra), atëherë e gjithë baza e të dhënave duhet të redaktohet [13].

Modeli relacional specifikon që rekordet e një relacioni nuk kanë renditje të veçantë dhe se rekordet nuk imponojnë asnjë renditje të atributeve (kolonave). Aplikacionet kanë akses në të dhëna duke perdorur pyetësorë (ang. queries), të cilët përdorin operacione të tilla si:

- **Selektimi** – për identifikimin e rekordeve
- **Projektimi** – për identifikimin e atributeve
- **Bashkimi** – për të kombinuar (bashkuar) relacionet

Relacionet mund të modifikohen duke përdorur operatorët e insertimit, fshirjes dhe editimit. Rekordet e reja mund të kenë vlera të caktuara ose të rrjedhin nga pyetësorë.
Përdorimi i DBMS për të menaxhuar të dhënat ka shumë përparësi:

- **Qasje efikase**: DBMS përdor teknika të sofistikuara për të ruajtur dhe marrë të dhëna në mënyrë efikase. Kjo karakteristike është veçanërisht e rëndësishme nëse të dhënat ruhen në pajisje të jashtme të ruajtjes.
- **Pavarësia e të dhënave**: Aplikacionet nuk duhet të ekspozohen në detajet e prezantimit dhe ruajtjes së të dhënave, DBMS ofron një pamje abstrakte së të dhënave që fshehin detaje të tilla.
- **Administrimi i të dhënave**: Kur disa përdorues ndajnë të dhënat, duke u centralizuar administrimi i të dhënave mund të ofrojë përmirësime të konsiderueshme. Profesionistët me përvojë, të cilët e kuptojnë natyrën e të dhënave që menaxhohen, dhe mënyrën sesi grupe të ndryshme të përdoruesve i përdorin ato, duhet të jenë përgjegjës për organizimin e paraqitjes të të dhënave për të minimizuar redudancen dhe për rregullimin e ruajtjes së tyre.
- **Integriteti dhe siguria**: Nëse perdoruesi i'u qaset të dhënave gjithmonë përmes DBMS, mund të has në kufizime të integritetit. Për shembull, para se të fusni të dhëna mbi pagën për një punonjës, DBMS mund të kontrollojë që buxheti i departamentit të mos tejkalohet. Gjithashtu mund të zbatojë kontrollet e qasjes që rregullojnë se cilat të dhëna duhet të jenë të dukshme për grupe të ndryshme të përdoruesve.
- **Qasje e njëkohësishme dhe rikthim i shpejtë**: Sistemi organizon hyrjet e njëkohësishme në të dhëna në një mënyrë të tillë që përdoruesit mendojnë se të dhënave po i'u qaset vetëm një përdorues në të njëjtën kohë. Gjithashtu sistemi mbron përdoruesit nga efektet e dështimeve të sistemit.

Një avantazh tjetër i bazave të të dhënave relacionale është se pothuajse çdo RDBMS mbështet transaksionet. Një transaksion përbëhet nga një ose më shumë operacione në SQL që ekzekutohen në rend si një njësi e vetme. Transaksionet paraqesin një qasje "gjithçka ose asgjë", që do të thotë se çdo operacion në transaksion duhet të jetë valid që të ekzekutohet; perndryshe, i gjithë transaksioni do të dështojë [14]. Kjo është shumë e dobishme për të siguruar integritetin e të dhënave kur bëhen ndryshime në tabela.

Për të ruajtur konsistencën në një bazë të dhënash, para dhe pas transaksionit, ndiqen disa veti të cilat quhen veti "ACID" [15]. Shkurtesa "ACID" zakonisht përdoret për t'iu referuar katër vetive të transaksioneve që janë:

- **Atomiciteti:** Transaksionet mund të jenë të parealizueshme për tre lloje arsyesh. Së pari, një transaksion mund të ndërpritet, ose të përfundoj pa sukses, nga DBMS sepse disa anomali lindin gjatë ekzekutimit. Nëse një transaksion ndërpritet për ndonjë arsye të brendshme, ajo automatikisht rinis dhe ekzekutohet përsëri. Së dyti, sistemi mund të dështojë ose bie (p.sh, sepse furnizimi me energji është ndërprerë) perderisa një ose më shumë transaksione janë në proces. Së treti, një transaksion mund të hasë në një situatë të papritur (p.sh, nëse lexon një vlerë të paparashikuar të dhënash) dhe vendos të ndërpresë ekzekutimin.
- **Konsistenca:** Përdoruesit janë përgjegjës për të siguruar konsistencën e transaksionit. Kjo do të thotë se përdoruesi që paraqet një transaksion duhet të sigurojë që, kur ekzekutohet deri në përfundim, transaksioni do ta lë bazën e të dhënave në një gjendje "të qëndrueshme". Për shembull, përdoruesi mund të ketë kriterin e konsistencës që transferimet e fondeve midis llogarive bankare të mos ndryshojnë shumën totale të parave në llogari. Për të transferuar para nga një llogari në tjetrën, një transaksion duhet të debitojë një llogari, duke e lënë përkohësisht bazën e të dhënave jo-konsistente, edhe pse bilanci i ri i llogarisë mund të plotësojë çdo kufizim të integritetit në lidhje me rangun e bilancit të pranueshëm në llogari. Qëllimi i përdoruesit për një bazë të dhënash konsistente ruhet kur llogaria e dytë kreditohet me shumën e transferuar. Nëse llogaria e dytë kreditohet me shumë më të vogël sesa shuma e debituar nga llogaria e parë, DBMS nuk mund të detektojë mospërputhjet për shkak të gabimeve të tilla në logjikën e programit të përdoruesit.
- **Izolimi:** Kjo veti sigurohet duke garantuar që, edhe pse veprimet e disa transaksioneve mund të ndërlidhen mes vete, efekti mbetet sikurse ekzekutimi i të gjitha transaksioneve në një renditje serike. Për shembull, nëse dy transaksione T1 dhe T2 ekzekutohen njëkohësisht, efekti është i garantuar të jetë ekuivalent me ekzekutimin e T1 ndjekur me ekzekutimin e T2 ose anasjelltas.
- **Qëndrueshmëria:** Kjo veti siguron që sapo transaksioni të ketë përfunduar ekzekutiomin, modifikimet në bazën e të dhënave të ruhen dhe të shkruhen ne disk dhe ato të qëndrojnë edhe nëse ndodh një dështim i sistemit. Efektet e transaksionit, kështu, nuk humbasin kurrë.

Normalizimi është procesi i bazës së të dhënave ku tabelat shumë të mëdha ndahen në pjesë të shumta më të vogla. Duke ndarë një tabelë të madhe në tabela më të vogla individuale, pyetësorët që hyjnë vetëm në një pjesë të të dhënave mund të ekzekutohen më shpejt sepse ka më pak të dhëna për të skanuar. Qëllimi kryesor i normalizimit është të ndihmojë në mirëmbajtjen e tabelave të mëdha dhë të zvoglojë kohën e përgjithshmë të përgjigjes për të lexuar dhe ngarkuar të dhëna për operacione të veçanta SQL.

Ndarja vertikale e tabelës përdoret më së shumti për të rritur performancen e SQL Server-it, veçanërisht në rastet kur një pyetësor tërheq të gjitha kolonat nga një tabelë që përmban një numër të kolonave të tekstit shume të gjerë ose BLOB. Në këtë rast, për të zvogëluar kohën e aksesit, kolonat BLOB mund të ndahen në tabelën e vet. Një shembull tjetër është të kufizosh hyrjen në të dhëna të ndjeshme p.sh. fjalëkalimet, informacioni për pagat, etj.

Ndarja vertikale ndan një tabelë në dy ose më shumë tabela që përmbajnë kolona të ndryshme.

Ndarja horizontale ndan një tabelë në tabela të shumta që përmbajnë të njëjtin numër kolonash, por më pak rreshta. Për shembull, nëse një tabelë përmban një numër të madh të rreshtave që përfaqësojnë raportet mujore, ajo mund të ndahet horizontalisht në tabela sipas viteve, me secilen tabelë që përfaqëson të gjitha raportet mujore për një vit specifik. Në këtë mënyrë pyetësorët që kërkojnë të dhëna për një vit specifik do të referojnë vetëm në tabelën e duhur. Tabelat duhet të ndahen në një mënyrë që pyetësorët të referojnë sa më pak tabela të jetë e mundur. Në figurën e mëposhtme është ilustruar ndarja vertikale dhe horizontale e tabelave.

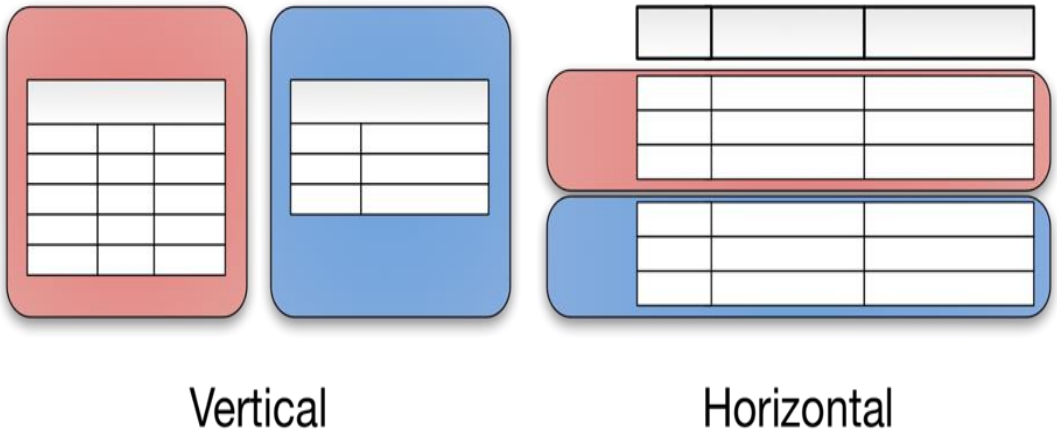

Figure 5. Ndarja vertikale dhe horizontale e tabelave [25].

Tabelat ndahen horizontalisht bazuar në një kolonë e cila do të përdoret për ndarjen dhe diapazonet që shoqërojnë secilën ndarje. Kolona e ndarjes është zakonisht një kolonë e datës, por të gjitha llojet e të dhënave që janë të vlefshme për t'u përdorur si kolona indeksi mund të përdoren si një kolonë e ndarjes, përveç një kolone me vlerën e kohës.

Ekzistojnë dy qasje të ndryshme që mund të përdorim për të arritur ndarjen e tabelës. E para është të krijoni një tabelë të re të ndarë dhe pastaj thjesht kopjoni të dhënat nga tabela juaj ekzistuese në tabelën e rë dhe të bëni një riemërtim të tabelës. Qasja e dytë është ndarja e një tabele ekzistuese duke rindertuar ose krijuar një indeks të grumbulluar në tabelë.

3.1.1 Implementimi i bazës së të dhënave në SQL Server

Meqenëse aplikacioni i zhvilluar për këtë punim shkencor kërkon ruajtjen e të dhënave në bazë të të dhënave, atëherë në këtë kapitull do të shpjegohet implementimi i bazës së të dhënave në SQL, përkatësisht Microsoft SQL Server.

Procesi i krijimit të një baze të të dhënave fillon me krijimin e një databaze të re. Një databazë mund të krijohet duke shfrytëzuar panelin e majtë apo Object Explorer, duke klikuar me tastin e djathtë të mausit mbi opsionin "Databases" dhe pastaj klikohet "New Database" që hap një dritare ku mund të vendoset emri i databazës që dëshironi t'a krijoni.

Pas krijimit të databazës mund të vazhdohet me krijimin e tabelave apo objekteve të caktuara të databazës.

Duke pasur parasysh që për bazat e të dhënave relacionale është e nevojshme që të dihet skema paraprakisht atëherë si fillim është e nevojshme që të detajohet skema mbi të cilën përcaktohen tabelat e bazës së të dhënave si dhe fushat apo kolonat e tabelave. Në skemën tonë kemi definuar 4 tabela të cilat do të ruajnë të dhëna të ndryshme. Tabela e parë që është krijuar është tabela **Patient** e cila ruan të dhënat e pacientëve. Krijimi i një tabele të re mundësohet duke naviguar tek nën-opsionet e databazës së krijuar tek opsioni Tables → New → Table. Pas kësaj, hapet dritarja për krijimin e tabelës ku duhet te vendosen emrat e kolonave, tipet e të dhënave të cilat do të ruhen në ato kolona, si dhe përcaktimi i çelësave primar ose të huaj. Në figurën e mëposhtme është paraqitur një shembull.

	Column Name	Data Type	Allow Nulls
🔑	Id	int	☐
	Name	nvarchar(30)	☑
	Surname	nvarchar(30)	☑
	PersonalNumber	bigint	☑
	Birthday	date	☑
	Gender	nvarchar(1)	☑
	BloodType	nvarchar(3)	☑
			☐

Figure 6. Krijimi i tabelës "Patient"

Procesi i njejtë vijon edhe për krijimin e tabelave tjera, prandaj nuk do të përsëritet procesi i njejtë mirëpo do të paraqitet vetëm skema e databazës në të cilën shfaqen të gjitha tabelat si dhe relacionet në mes tyre. Në figurën e mëposhtme është shfaqur skema në të cilën përveq tabelave dhe kolonave të tyre, shihen edhe relacionet e tabelave. Në skemë shihet se kemi predefinuar 4 tabela të cilat do të ruajnë të dhëna te ndryshme. Tabela "Doctor" e cila ruan të dhënat për stafin mjekësor, pastaj tabela "Treatments" e cila ruan të dhënat për trajtimet e marrura të pacientëve nga doktorët, "Patient" ruan të dhënat e pacientëve, si dhe tabela Result shërben si tabelë për ruajten e rezultateve të marrura nga aplikacioni për vlerësimin e kohës së veprimeve të realizuara në databazë.

Relacionet e përdorura janë relacionet një-me-shumë, që nënkupton se tabela e çelësit primar përmban vetëm një record që lidhet me një ose shumë rekorde në tabelën përkatëse. Në rastin tonë, tabela Doctor referencohet në tabelën Treatment përmes çelësit të huaj, si dhe tabela Patient referencohet me tabelën Treatment.

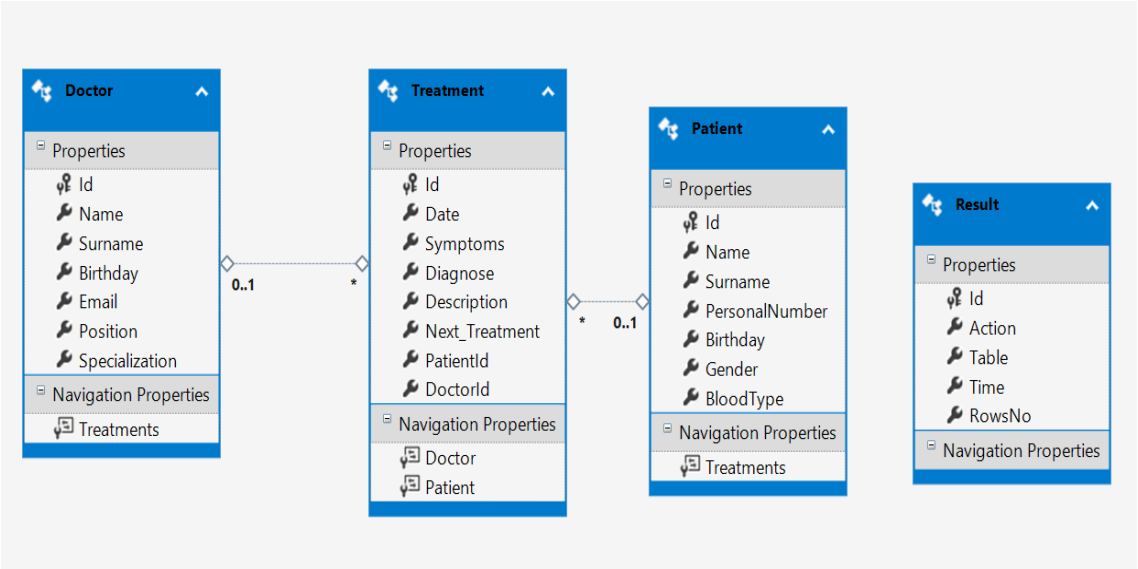

Figure 7. Skema e bazës të të dhënave

3.2 Bazat e të dhënave NoSQL

Një bazë e të dhënave NoSQL (fillimisht duke iu referuar "jo-SQL" ose "jo-relacionale") [16] ofron një mekanizëm për ruajtjen e të dhënave që modelohet ndryshe nga relacionet tabelare të përdorura ne bazat e të dhënave relacionale. Bazat e të dhënave të tilla kanë ekzistuar që nga fundi i viteve 1960, por emri "NoSQL" u krijua në fillim të shekullit 21 [17]. Bazat e të dhënave NoSQL përdoren gjithnjë e më shumë në te dhëna të mëdha dhe ueb aplikacione. Keto sisteme nganjëherë quhen "jo vetëm SQL" për të theksuar se ato mund të mbështesin gjuhë pyetësore të ngjajshme me SQL.

Motivimet për këtë qasje perfshijnë: thjeshtësine e dizajnit, shkallëzimin e thjeshtë horizontal (që është një problem për bazat e të dhënave relacionale), [16] kontroll më të mirë mbi disponueshmërinë dhe kufizimin e mospërputhjes së rezistences se databazave relacionale te bazuara ne objekte. Strukturat e të dhënave të përdorura nga NoSQL (psh. çifti çelës-vlerë, grafi ose dokumenti) janë të ndryshme nga ato të perdorura në bazat e të dhënave relacionale, duke i bërë disa operacione më të shpejta në NoSQL. Ndonjëherë strukturat e të dhënave të përdorura nga NoSQL shihen gjithashtu si "më fleksibile" sesa tabelat e bazave të të dhënave relacionale [18]. Ato janë një përshtatje e shkëlqyeshme për shumë aplikacione si aplikacionet mobile, ueb aplikacionet dhe lojëra që kërkojnë baza të të dhënave fleksibile, të shkallëzuara, me performancë të lartë dhe shumë funksionale për të siguruar përvoja të shkëlqyera të përdoruesit duke ofruar:

- **Fleksibilitet**: Bazat e të dhënave NoSQL zakonisht ofrojnë skema fleksibile që mundësojnë zhvillim të shpejtë. Modeli fleksibël i të dhënave i bën bazat e të dhënave NoSQL ideale për të dhëna gjysëm te strukturuara dhe të pastrukturuara.
- **Shkallezim**: NoSQL jane krijuar për të shkallëzuar duke përdorur grupe të shpërndara të pajisjeve në vend që të shkallëzohen duke shtuar servera të shtrenjte.
- **Performancë të lartë**: Ato janë të optimizuara për modele specifike të të dhënave dhe modele aksesi që mundësojnë përformancë më të lartë sesa perpjekja për të kryer funksionalitet të njejtë me bazat e të dhënave relacionale

- **Funksionalitet të lartë**: Ato ofrojnë API dhe lloje të dhënash shumë funksionale që janë krijuar për secilin nga modelet përkatëse të të dhënave.

Bazat e të dhënave jo-relacionale nuk perdorin principet e bazave të të dhënave relacionale dhe nuk ruajnë të dhënat në tabela, nuk kanë skemë fikse dhe kanë model të thjeshtë të të dhënave. Ato përdorin çelësa identifikues dhe të dhënat mund të gjinden në bazë të çelesit.

Janë gjithsej 4 modele për ruajtjen e të dhënave në një bazë të të dhënave jo-relacionale, qe jane:

1. **Modeli celës-vlerë** (ang. key-value) – përdor grupin asociativ (i quajtur ndryshe si fjalor ose hartë) si modelin e tyre themelor të të dhënave. Në këtë model, të dhënat përfaqësohen si një koleksion i çifteve çelës-vlerë, ashtu qe secili çelës i mundshëm shfaqet më së shumti një herë ne koleksion [19].
   Modeli çeles-vlerë është një nga modelet më të thjeshta të të dhenave jo-triviale, dhe modelet më të mira të të dhënave shpesh zbatohen si vazhdim i tij. Ky model mund të zgjerohet në një model të renditur që mban çelësat në renditje leksikografike.
   Disa baza të të dhënave mbështesin renditjen e çelësave. Modeli çeles-vlere perdor modele të konsistencës duke filluar nga qëndrushmëria eventuale deri në serializueshmërinë. Ekzistojnë implementime të ndryshme harduerike, dhe disa përdorues i ruajnë të dhënat në memorie (RAM), ndërsa të tjerët në disqe (SSD ose HDD).
2. **Dokumenti** – koncepti kryesor i këtij modeli është dokumenti. Përderisa detajet e këtij përkufizimi ndryshojnë prej bazave të të dhënave të orientuara në dokument, të gjithë supozojnë se dokumentet enkapsulojnë dhe kodifikojnë të dhëna (ose informacione) në disa formate standarte ose kodifikime. Kodifikimet në përdorim përfshijnë XML, YAML dhe JSON si dhe forma binare si BSON. Dokumentet adresohen në bazën e të dhënave përmes një çelësi unik që përfaqëson atë dokument. Karakteristikë tjetër përcaktuese e një baze të të dhënave të orientuar në dokumente është një API ose gjuhë pyetëse për të marrë dokumentet bazuar në përmbajtjen e tyre.
   Implementime të ndryshme ofrojnë mënyra të ndryshme të organizimit dhe grupimit të dokumenteve:
   - Koleksione
   - Etiketa
   - Meta-data jo të dukshme
   - Hierarki drejtuese

   Krahasuar me bazat e të dhënave relacionale, koleksionet mund të konsiderohen të ngjajshme me tabelat dhe dokumentet të ngjajshme me rekordet (ose rreshtat). Por ato janë të ndryshme: çdo rekord në një tabelë ka sekuencë të njejtë të fushave, perderisa dokumentet në një koleksion mund të kenë fusha që janë krejtësisht të ndryshme.
3. **Grafi** – është i dizajnuar për të dhëna në të cilat relacionet përfaqësohen si një graf i përbërë nga elemente të lidhura nga një numër i caktuar i relacioneve. Baza e të dhënave për grafe është një lloj i bazës së të dhënave që është I aftë të integrojë të dhëna heterogjene nga shumë burime dhe të bëjë lidhje midis grupeve

të të dhënave. Ai përqëndrohet në marredhëniet midis entiteteve dhe është në gjendje të nxjerrë njohuri të reja nga informacioni ekzistues. Shembuj të të dhënave përfshijnë relacionet shoqërore, lidhjet e transportit publik, hartat rrugore, topologjitë e rrjetit, etj.

4. **Kolona –** Në paraqitje të pare janë të ngjashme me DBMS tradicionale meqenëse duhet tër përcaktohet tipet e kolonave para se të ngarkohen të dhënat në bazë të të dhënave, që do të thotë se duhet të dihet struktura e të dhënave paraprakisht. Sidoqoftë, këto baza të të dhënave organizojnë të dhënat ndryshe nga DBMS tradicionale. Në vend të ruajtjes së të dhënave në një rresht për qasje të shpejtë, të dhënat janë të organizuara për operacione të kolonave të shpejta, Kjo pamje kolonë-centrike i bën ato ideale për ekzekutimin e funksioneve agregate ose për kërkimin e rekordeve që përputhen me shumë kolona. Funksionet agregate janë kombinime të të dhënave ose funksione analizuese. Ato mund të jenë aq të thjeshta sa llogaritja e numrit të rezultateve, mbledhja e tyre ose llogaritja e vlerës mesatare. Mirëpo, ato mund të jenë më komplekse – për shembull, duke kthyer një vlerë komplekse që përshkruan një gamë gjithëpërfshirëse të kohës.

Në figurën e mëposhtme janë paraqitur lloje të ndryshme të bazave të të dhënave jo-relacionale, bazuar sipas modeleve të tyre:

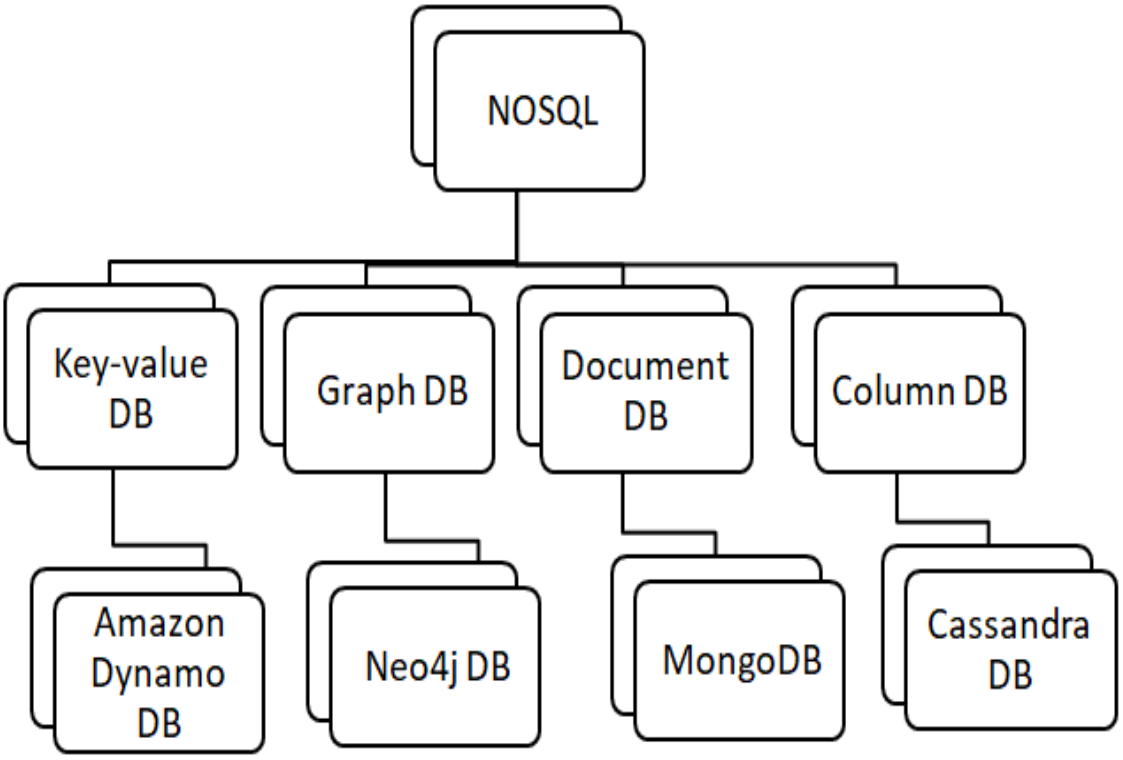

Figure 8. Ndarja e bazave të të dhënave jo-relacionale bazuar në modelet e tyre [20].

Eshtë shumë e rëndësishme të kuptohen kufizimet e bazave të të dhënave NoSQL. NoSQL nuk mund të sigurojë qëndrueshmëri dhe disponueshmëri të lartë së bashku. Kjo u paraqit si teoremë për herë të parë nga Eric Breëer në teoremën e "CAP".
Teorema e CAP përdoret për të bërë të vetëdijshëm dizajnerët e sistemit për shkëmbimet gjatë hartimit të sistemeve të rrjetit të të dhënave të ndara. CAP ka ndikuar në hartimin e shumë sistemeve të shpërndara të të dhënave. Eshtë shumë e rëndësishme të kuptohet kjo

teoremë pasi që përbën bazat për zgjedhjen e llojit të bazës së të dhënave bazuar në kërkesat.
Teorema e CAP shprehet se në sistemet e rrjetit të të dhënave të ndara ose sistemeve të shpërndara, ne mund të arrijmë më së shumti dy nga tre garanci për një bazë të dhënash që janë: konsistenca, disponueshmëria dhe toleranca e ndarjes [21].
Një sistem i shpërndarë është një rrjet që ruan të dhëna në më shumë se një nyje (makina fizike ose virtuale) në të njejtën kohë.

Le të kuptojmë së pari çka do të thotë CAP me fjalë të thjeshta:

Konsistenca: do të thotë që të gjithë klientët shohin të njëjtat të dhëna në të njejtën kohë, pa marrë parasysh se në cilen nyje (ang. node) lidhen në një sistem të shpërndarë. Për të arritur qëndrueshmëri, sa herë që të dhënat shkruhen në një nyje, ato duhet menjëherë të përcjellen ose përsëriten në të gjitha nyjet tjera në sistem përpara se shkrimi të konsiderohet i suksesshëm.

Disponueshmëria: do të thotë që çdo nyje që nuk dështon kthen një përgjigje për të gjitha kërkesat e leximit dhe shkrimit në një kohë të arsyeshme, edhe nëse një ose më shumë nyje kanë rënë nga sistemi. Një mënyrë tjetër për ta deklaruar këtë – të gjitha nyjet në sistemin e shpërndarë kthejnë një përgjigje të vlefshme për çdo kërkesë, pa dështuar ose kundërshtuar.

Toleranca e ndarjes: do të thotë që sistemi vazhdon të funksionojë pavarësisht humbjes arbitrare të mësazhit ose dështimit të një pjese të sistemit. Me fjalë të tjera, edhe nëse ka një ndërprerje rrjeti në qendrën e të dhënave edhe disa prej kompjuterëve janë të paarritshëm, sistemi vazhdon të përformojë. Sistemet e shpërndara që garantojnë tolerancën e ndarjës mund të rikuperohen me lehtësi nga ndarjet sapo ndarja të realizohet.
Teorema e CAP i kategorizon sistemet në tri kategori:

Baza e të dhënave CP: që ofron qëndrueshmëri dhe tolerancë në ndarje në saje të disponueshmërisë. Kur ndodh një ndarje midis dy nyjeve, sistemi duhet të mbyllë nyjen jo-konsistente (dmth., ta bëjë atë të padisponueshme) derisa të zgjidhet ndarja.
Ndarja i referohet një ndërprerje komunikimi midis nyjeve brenda një sistemi të shpërndarë. Do të thotë, nëse një nyje nuk mund të marrë ndonjë mesazh nga një nyje tjetër në sistem, ekziston një ndarje midis dy nyjeve. Ndarja mund të ketë qenë për shkak të dështimit të rrjetit, prishjes së serverit ose ndonjë arsye tjetër.
Baza e të dhënave AP: që sjell disponueshmërinë dhe tolerancën e ndarjes në sajë të qëndrueshmërisë. Kur ndodh një ndarje, të gjitha nyjet mbeten të disponueshme, por ato në fundin e gabuar të një ndarjeje mund të kthejnë një version më të vjetër të të dhënave se të tjerët. Kur zgjidhet ndarja, bazat ë të dhënave AP zakonisht sinkronizojnë nyjet për të riparuar të gjitha mospërputhjet në sistem.
Baza e të dhënave CA: që ofron qëndrueshmëri dhe disponueshmëri në mungesë të ndonjë ndarjeje rrjeti. Shpesh serverat të nyje të vetme kategorizohen si sisteme CA për shkak se, serverat me një nyje të vetme nuk kanë nevoje të merren me tolerancen e ndarjes.
Në çdo sistem të rrjetit të të dhënave të ndara ose në sistemin e shpërndarë toleranca e ndarjes është e domosdoshme. Ndarjet e rrjetit dhe mesazhet e lëshuara janë një fakt i jetës dhe duhet të trajtohen në mënyrë të përshtatshme. Si pasojë, dizajnerët e sistemit

duhet të zgjedhin midis qëndrueshmërisë dhe disponueshmërisë. Diagrami i mëposhtëm tregon klasifikimin e bazave të të dhënave bazuar në teoremën e CAP.

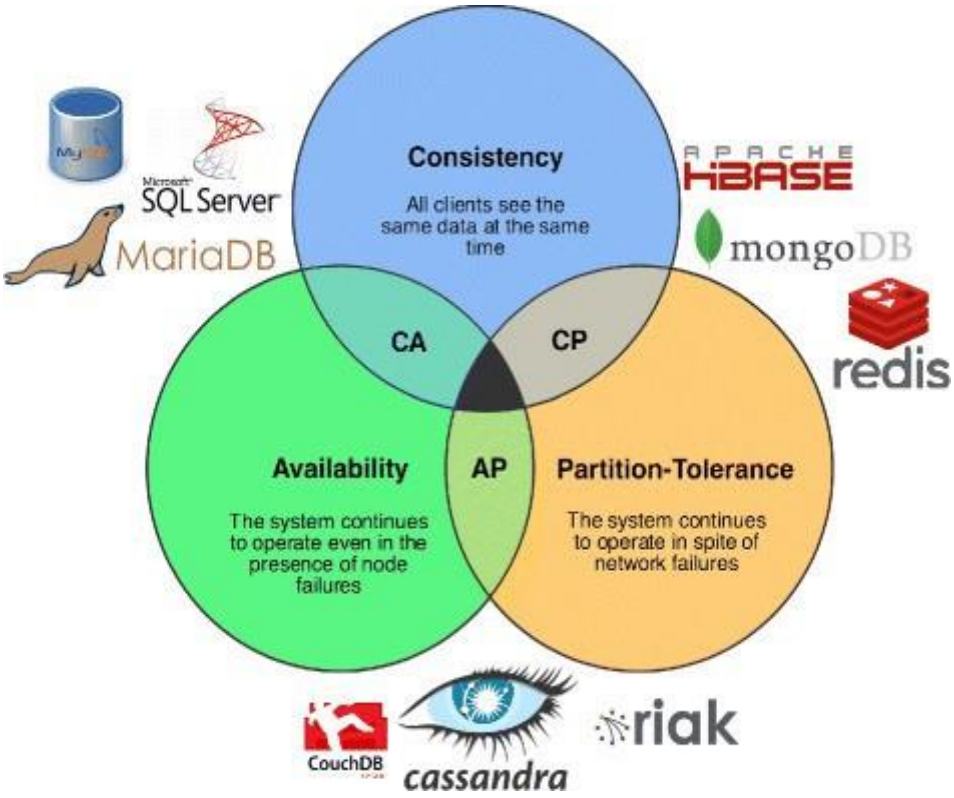

Figure 9. Klasifikimi i bazave të të dhënave bazuar në teoremën e CAP-it [24].

Në mënyrë që të ruhen të dhënat në shumë makina, të dhënat duhet të ndahen. Bazat e të dhënave NoSQL zakonisht përdorin konceptin e quajtur "shkallëzimi" (ang. sharding) për të ndarë të dhënat e tabelës dhe për t'i shpërndarë në shumë makina.

Shkallëzimi është një koncept i ndarjes së një grupi të madh rekordesh të vendosur në grupe relativisht të vogla rekordesh. Grupet e rekordeve zakonisht ndahen duke përdorur një çelës copëtues i cili është një nga kolonat në tabelë.
Një varg specifik i çelësave përcaktohet si një copëtim dhe çdo çelës që bie brenda këtij vargu përcaktohet për atë pjesë. Për shembull, le të themi se kemi tabelën e klientit dhe fusha "Emri" është çelësi i copëtimit, atëherë rekordet mund të ndahen duke përdorur vargun e karakterit të parë në emër duke filluar nga shkronja 'A' ne 'F' dhe 'G' në 'L' dhe kështu me rradhë. Një copëtim mund të caktohet në një nyje ose një grup nyjesh në rast te përsëritjes. Në përgjithësi, një nyje mban një skemë të rangut të çelësit copëtues dhe nyjeve që egziston copëtim. Pra bazuar në çelësin e rekordit i cili do të modifikohet ose ruhet, kërkesa dërgohet në nyjen përkatëse.

Shkallëzimi mund të shfrytëzohet duke zgjedhur një çelës të rastit dhe jorepetitiv, sepse jep shpërndarje më të mirë. Në vend që të shpërndahet rekordi me algoritmin "hash", ku rreshtat e njëpasnjëshëm mund të jenë në makina të ndryshme, të kesh një sërë rekordesh radhazi në të njëjtën makinë mund të ndihmojë në rritjen e leximit, shkrimit dhe skanimit.

Në të njëjtën kohë leximi dhe shkrimi i rastësishëm gjithashtu mund të arrijë një përformancë të lartë duke përdorur makina të shumta.
Shumica e bazave të të dhënave NoSQL mbështësin shkallëzimin automatik i cili kërkon ndarjen dhe shpërndarjen në nyje të të dhënave.

3.2.1 Implementimi i bazës së të dhënave në MongoDB

Për ruajtjen e të dhënave në MongoDB, hapi i parë themelor është të kesh një bazë të dhënash dhe koleksion. Baza e të dhënave përdoret për të ruajtur të gjitha koleksionet, dhe koleksioni nga ana tjetër përdoret për të ruajtur të gjitha dokumentet. Dokumentet nga ana e tyre do të përmbajnë emrin përkatës të fushës dhe vlerat e fushës. Egzistojnë disa mënyra të ndryshme për krijimin e bazës së të dhënave apo koleksioneve në MongoDB, mirëpo për këtë projekt është shfrytëzuar aplikacioni MongoDB Compass, andaj në vazhdim do të tregojmë hapat e nevojshëm për krijimin dhe implementimin e bazës së të dhënave në MongoDB.

Pas startimit të aplikacionit, klikohet opsioni **"Databases"** në anën e majtë, me të cilin hapet dritarja sic tregohet në figurën e mëposhtme.

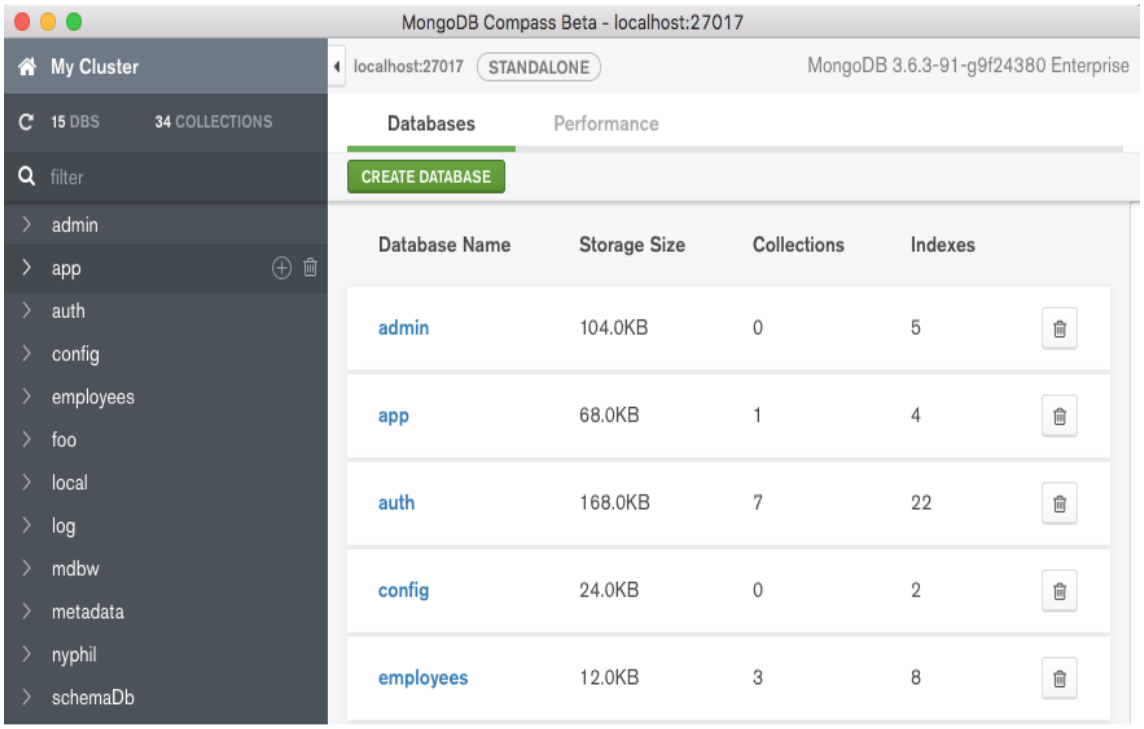

Figure 10. Dritarja për krijimin e bazës së të dhënave në MongoDB Compass

Nga kjo pamje, mund të klikoni një emër të bazës së të dhënave në listë për të parë koleksionet e saj. Përndryshe, ju mund të shihni koleksionet e bazës së të dhënave duke klikuar në bazën e të dhënave të dëshiruar në navigimin në të majtë.

Për krijimin e një baze të re të të dhënave, shfrytëzohet butoni "Create Database" i cili hap dritaren për krijimin e një baze të të dhënave. Në atë dritare kemi vendosur emrin e

bazës së të dhënave, si dhe emrin e koleksionit që do të krijohet për ruajtjen e dokumenteve. Të dyja fushat e përmendura janë të obligueshme për plotësim. Nëse dëshironi të krijoni një koleksion të mbyllur, zgjidhni opsionin “Capped Collection” dhe futni bajtet maksimale. Nëse dëshironi të përdorni system të përcaktuar në koleksionin e krijuar rishtas, zgjidhni opsionin “Use Custom Collection” dhe zgjidhni opsionet e dëshiruara. Shihni figurën e mëposhtme për më shumë detaje.

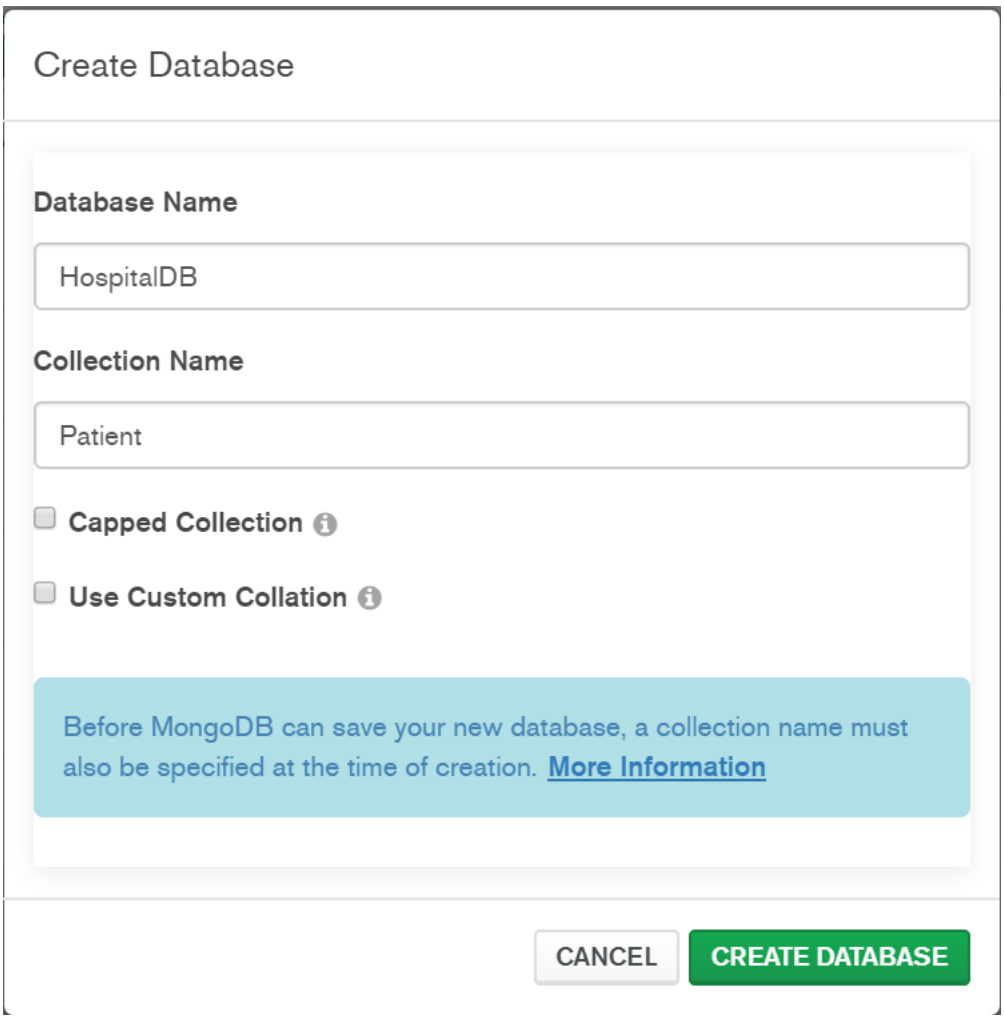

Figure 11. Krijimi i bazës së të dhënave në MongoDB Compass

Në figurë shihet se është përcaktuar emri i bazës së të dhënave si **HospitalDB**, si dhe emri i koleksionit si **“Patient”**. Pas klikimit të opsionit “Create database” do të krijohet baza e të dhënave bashkë me koleksionin e përcaktuar. Ndërsa për krijimin e koleksioneve tjera shfrytëzohet butoni “Create Collection” i cili hap dritaren për krijimin e koleksionit me detaje si në figurën e mësipërme.

Koleksionet e krijuara për këtë bazë të të dhënave janë paraqitur më poshtë:

```
_id: ObjectId("60053515616b12df32330e25")
Name: "Egzon"
Surname: "Murati"
PersonalNumber: 1111111
Birthday: "1999-01-01"
Gender: "Female"
BloodType: "A-"
ToUpdate: false
```


Figure 12. Koleksioni "Patient" në MongoDB


```
_id: ObjectId("603c0eb1527e63e938ef0fc6")
Name: "Dardan"
Surname: "Hasani"
Birthday: 1976-02-15T23:00:00.000+00:00
Email: "dardanhasani@gmail.com"
Specialization: "Dermatology"
ToUpdate: true
```


Figure 13. Koleksioni "Doctor" në MongoDB


```
_id: ObjectId("603c0ed9527e63e938ef0fca")
Date : 2021-01-27T23:00:00.000+00:00
Symptoms : " Symptoms "
Diagnose : " Diagnose "
Description : " Description "
NextTreatment : null
PatientId : " 171502 "
DoctorId : null
ToUpdate : false
```


Figure 14. Koleksioni "Treatment" në MongoDB

Për shtimin e të dhënave në një koleksion të caktuar ofrohen dy opsione që janë:

- Importimi i një fajlli
- Insertimi i një dokumenti

Me klikimin e opsionit të parë, ofrohet mundësia e shtimit të të dhënave duke ngarkuar një json file në të cilin janë të ruajtura të dhënat, ndërsa nëse klikohet opsioni i dytë hapet dritarja nga e cila mund të insertojmë dokumente në formatin e caktuar.

3.3 Krahasimi i bazave të të dhënave SQL dhe NoSQL

Kur bëhet fjalë për zgjedhjen e një baze të të dhënave moderne, një nga vendimet më të mëdha është zgjedhja e një databaze relacionale (SQL) ose jo-relacionale (NoSQL). Përderisa të dyja janë opsione të vlefshme, ekzistojnë disa dallime kryesore midis të dyjave që përdoruesit duhet të kenë parasysh kur marrin një vendim. Në këtë kapitull do të diskutohen dallimet kryesore ndërmjet tyre dhe do të diskutohen sistemet më të mira të bazave të të dhënave SQL dhe NoSQL.

Bazat e të dhënave SQL përdorin gjuhë të strukturuar të pyetësorëve për definimin dhe manipulimin e të dhënave. Nga njëra anë, kjo është jashtëzakonisht e fuqishme, duke e bërë atë një zgjedhje të sigurt dhe veçanërisht të shkëlqyeshme për pyetësorë kompleks. Nga ana tjetër, mund të jetë kufizuese. SQL kërkon që të përdorni skema të paracaktuara për të përcaktuar strukturën e të dhënave tuaja para se të punoni me të. Për më tepër, të gjitha të dhënat tuaja duhet të ndjekin të njejtën strukturë. Kjo mund të kërkojë përgatitje të konsiderueshme paraprake dhe kjo mund të nënkuptojë që një ndryshim në strukturë do të ishte i vështirë për sistemin tuaj.

Bazat e të dhënave jo-relacionale, në anën tjetër, kanë skema dinamike për të dhëna të pastrukturuara, dhe të dhënat ruhen në mënyra të ndryshme. P.sh, ato mund të jenë të orientuara në kolona ose dokumente, të bazuara në grafikë ose të organizuara si çelës-vlerë. Ky fleksibilitet nënkupton që ju mund të krijoni dokumente para se të përcaktoni më parë strukturën e tyre. Përveq kësaj, secili dokument mund të ketë strukturën e vet unike, duke e bërë të ndryshëm prej dokumenteve tjera, si dhe mund të shtoni fusha të reja më vonë.

Në shumicen e rasteve, bazat e të dhënave SQL janë vertikalisht të shkallëzuara, që do të thotë që ju mund të rrisni ngarkesën në një server të vetë duke rritur gjëra të tilla si CPU, RAM ose SSD. Nga ana tjetër, bazat e të dhënave jo-relacionale janë horizontalisht të shkallëzueshme. Kjo do të thotë që ju mund t'a trajtoni ngarkesën duke shtuar më shumë servera në bazën tuaj të të dhënave. Kjo e bën këtë sistem zgjedhjen e preferuar për grupe të mëdha.

Bazat e të dhënave relacionale janë të bazuara në tabela, ndërsa ato jo-relacionale janë ose të bazuara në dokumente, çifte me vlerë kyçe, grafe ose kolona. Kjo i bën bazat e të dhënave relacionale SQL një mundësi më të mirë për aplikacionet që kërkojnë transaksione me shumë rreshta – të tilla si një sistem kontabiliteti – ose për sistemet e trashëguara që janë ndërtuar për një strukturë relacionale.

Shembuj të bazave të të dhënave relacionale përfshijnë MySQL, Oracle, PostgreSQL dhe Microsoft SQL Server. Ndërsa llojet e bazave të të dhënave jo-relacionale janë: MongoDB, BigTable, Redis, RavenDB Cassandra, Hbase, Neo4j dhe CouchDB. Egzistojnë disa dallime midis termeve të përdorura në SQL dhe NoSQL të cilat janë paraqitur në tabelën e mëposhtme:

Koncepti në SQL	Koncepti në MongoDB
Bazë e të dhënave	Bazë e të dhënave
Tabelë	Koleksion
Rresht	Dokument
Kolonë	Fushë
Index	Index
Lidhje mes tabelave	Dokumente të ngjitura
Çelës primar	Çelës primar
Specifikon cilëndo prej kolonave unike si çelës primar	Mban një çelës primar të dokumentit i cili është objekt BSON

Table 2. Dallimet e koncepteve mes SQL dhe MongoDB

Sa i përket përformancës, në debatin e shpejtësisë së MongoDB apo SQL Server, MongoDB zakonisht del si fitues. MongoDB mund të pranojë sasi të mëdha të të dhënave të pastrukturuara shumë më shpejt se SQL. Eshtë e vështirë të matësh saktësisht se sa më i shpejtë është MongoDB sesa SQL Server-i kur merret me projekte të mëdha. Shpejtësia e saktë që përjetoni mund të varet nga një sërë faktorësh, të tilla si gjerësia e lidhjes suaj të internetit, distanca midis vendndodhjes suaj dhe serverit të bazës së të dhënave dhe sa mirë i organizoni të dhënat tuaja.

Nëse bazohemi në njërin nga punimet shkencore të ngjajshme, si p.sh. punimi shkencor "A performance comparison of SQL and NoSQL databases" i cili nxjerr në pah vlerësimin e përformancës së bazave të të dhënave SQL dhe NoSQL duke u bazuar në 4 pika kryesore të cilat janë: leximi, shkrimi, editimi dhe fshirja e të dhënave nga një objekt në databazë, shihet se NoSQL ka një përformancë shumë më të mirë.

Për secilin nga operacionet e përmendura më lartë, janë bërë disa testime për numer të caktuar të rekordeve/dokumenteve, duke filluar nga 10 deri 100000 rekorde. Rezultatet e nxjerra për operacionin e leximit tregojnë për një përformancë më të mirë të NoSQL-it, mirëpo nuk qëndron prapa as SQL për nga shpejtësia. P.sh., sipas statistikave të paraqitura në atë punim, për leximin e **10000** rekordeve apo dokumenteve nga MongoDB merr kohë **1085ms**, ndërsa nga SQL Server **1968ms** [22].

Përformancë dukshëm më të mirë ka MongoDB kundrejt SQL Server-it në operacionin e shkrimit dhe editimit të të dhënave. P.sh. për insertimin apo editimin e **10000** dokumenteve në MongoDB, statistikat tregojnë se MongoDB merr kohë prej **2693ms** ndërsa SQL Server-i merr kohë **15588ms**, që do të thotë se MongoDB ka përafersisht 5 ose 6 herë përformancë më të mirë sesa SQL Server-i në operacionet e leximit ose editimit [22].

Renditur sipas përformancës së fshirjes së të dhënave, ato nuk qëndrojnë shumë larg njëra-tjetrës. Statistikat tregojnë se për fshirjen e **10000** rekordeve ose dokumenteve nga MongoDB, shpenzohet kohë prej **2115ms**, ndërsa nga SQL Server-i shpenzohet kohë prej **3571ms** [22].

4 Rast studimi – zhvillimi i aplikacionit

Në njërin nga kapitujt e mëparshem kemi treguar për teknologjitë e përdorura për zhvillimin e aplikacionit ndërsa në këtë kapitull do të zgjerohemi më shumë duke treguar se si janë shfrytëzuar ato teknologji, për arkitekturën e aplikacionit si dhe për aplikacionin në përgjithësi.

Aplikacioni që është zhvilluar quhet **PMS** (Patient management system) ose sistemi i menaxhimit të pacientëve. Qëllimi i aplikacionit është vlerësimi i përformancës së bazave të të dhënave relacionale dhe jo-relacionale në bazë të operacioneve “crud”. Shpjegimin e zhvillimit të aplikacionit do ta bëjmë duke filluar prej pjesës së back-end-it apo pjesës së komunikimit me bazat e të dhënave e pastaj në funksionalitetet dhe karakteristikat e aplikacionit në front-end.

Për ruajtjen e të dhënave të aplikacionit, kemi përdorur dy baza të të dhënave, SQL Server dhe MongoDB. Duke pasur parasysh që për bazat e të dhënave jo-relacionale është e nevojshme që të krijohet skema paraprakisht atëhere si fillim kemi definuar skemën mbi të cilën përcaktohen tabelat e bazës së të dhënave, si dhe fushat apo të dhënat të cilat do të ruhen në atë tabelë. Në figurën e mëposhtme është paraqitur skema që përshkruan skëmën e të dhënave dhe relacionet e tabelave ndermjet vete.

4.1 Komunikimi server-client

Për të vendosur komunikimin e ueb aplikacionit me server kemi përdorur Web API. Një API për një ueb-faqe në internet është kod që lejon dy programe softuerësh të komunikojnë me njëri-tjetrin. API tregon mënyrën e duhur për një zhvillues për të shkruar një program që kërkon shërbime nga një sistem operativ ose një aplikacion tjetër. Egzistojnë qasje të ndryshme për komunikimin klient-server sic janë: REST, SOAP, AJAX, mirepo në ketë aplikacion shfrytëzohet qasja e parë, dmth arkitektura REST.

Shkurtesa REST qëndron për "Transferimi i gjendjës së përfaqësimit" dhe i referohet një stili arkitektonik të softuerit. Bazohet në gjashtë parime që përshkruajnë mënyrën se si përcaktohen dhe adresohen burimet e lidhura në rrjet në internet. Këto parime u përshkruan nga Roy Fielding në punimin e tij shkencor "Stilet arkitektonike dhe dizenjimi i arkitekturave softuerike të bazuara në rrjetë" në 2000. Parimi i pashtetësisë (stateless) është thelbësor për një REST API. Aty thuhet se çdo mesazh REST përmban të gjithë informacionin e nevojshëm për të kuptuar atë mesazh. REST nuk është një gjuhë programimi ose një strukturë bazë dhe nuk është një pjesë e softuerit që mund të ekzekutohet. RESTful API-të përdorin metodat e kërkesës HTTP GET, POST, PUT dhe DELETE të përcaktuara në RFC 2616. Me GET, burimet kërkohen nga një API Pushues. POST përdoret për të azhurnuar ose ndryshuar gjendjen e një burimi. Me PUT, burimet e reja mund të krijohen ose përmbajtja e burimeve ekzistuese mund të zëvendësohet. DELETE përdoret për të fshirë burimet. Këto katër metoda HTTP zakonisht janë të mjaftueshme për të mbuluar shumicën e rasteve të përdorimit. Në figurën e mëposhtme është ilustruar se si REST API vendos komunikimin ndërmjet klientit dhe serverit. Pra, klienti nuk ka komunikim direkt me server, atë ia mundëson REST API si pjesë ndërlidhëse ndërmjet tyre.

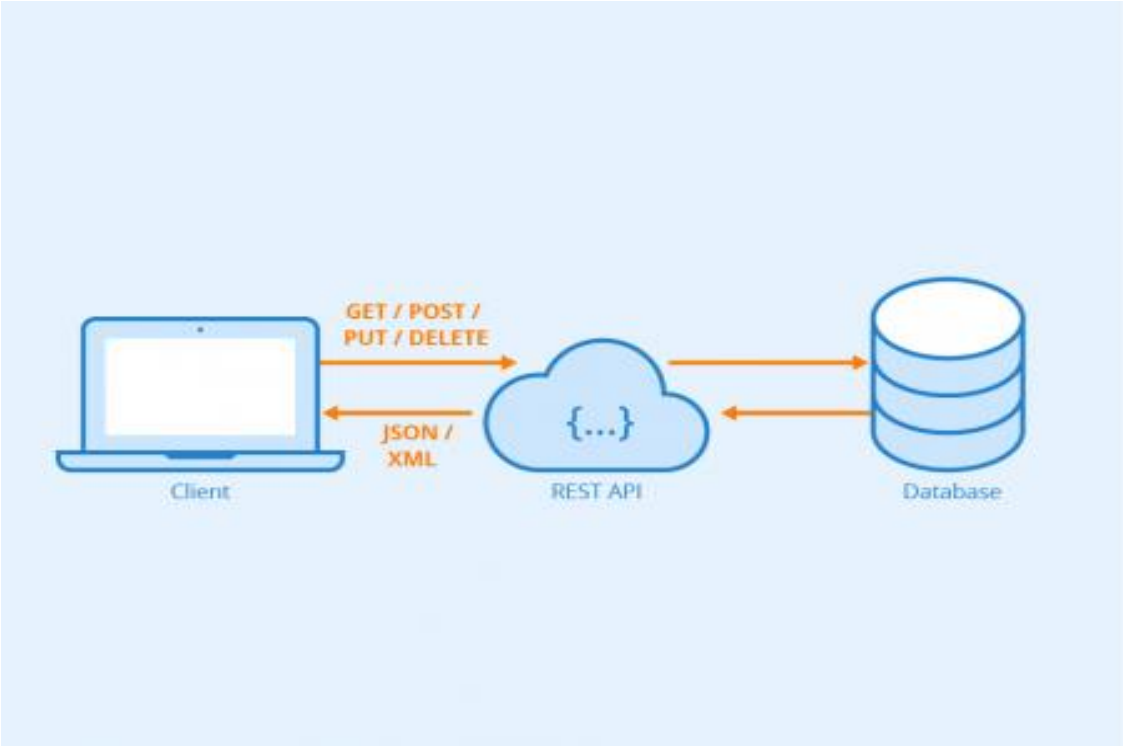

Figure 15. Komunikimi klient-server përmes Rest API [23].

Sic shihet në figure, komunikimi bazohet në atë mënyrë që i gjithë komunikimi të kryhet përmes API-it. Psh, klienti dërgon kërkesa të ndryshme si: get, post, put ose delete dhe atë kerkëse e pranon dhe e dërgon në server. Përgjigja kthehet në API dhe pastaj e kthen tek klienti si JSON ose XML.

4.2 Operacionet CRUD

Dokumentimi i referencës për pikat përfundimtare të API-së përbëhet nga pesë seksione të përgjithshme: përshkrimet e burimeve, pikat përfundimtare dhe metodat, parametrat, kërkesat e mostrave dhe përgjigjet dhe skemat e mostrave. Për të dokumentuar pikat përfundimtare të referencës të një API, në këtë kapitull do të japim informacione të detajuara për secilën pikë. Në vazhdim do të paraqesim tabelat me informacionet e nevojshme për secilën metodë.

Patient	Doctor	Treatment
GET api/patient/nrOfRows	GET api/doctor/nrOfRows	GET api/treatment/nrOfRows
POST api/patient/nrOfRows	POST api/doctor/nrOfRows	POST api/treatment/nrOfRows
PUT api/patient/nrOfRows	PUT api/doctor/nrOfRows	PUT api/treatment/nrOfRows
DELETE api/patient/nrOfRows	DELETE api/doctor/nrOfRows	DELETE api/treatment/nrOfRows

Table 3. Kanalet e komunikimit në API (endpoints)

Në tabelën e mësipërme janë paraqitur kanalet e komunikimit (endpoints) për secilën prëj 4 operacioneve themelore CRUD, të ndara në bazë të tabelave ose koleksioneve. Këto kanale të komunikimit të paraqitura në tabelë shihet se pranojnë si parametër numrin e rreshtave në të cilët kërkohet veprimi i caktuar në bazën e të dhënave. Më poshtë tregohet se çfarë ndodh nëse njëra nga metodat dërgohet nga klienti.

1. **GET** – Kjo metodë përdoret për të marrë listën e të dhënave të ruajtura në atë resurs në bazë të numrit të të dhënave. P.sh., GET api/patient/10000 kthen si përgjigje të dhënat e 10000 pacientëve të rujtura në bazën e të dhënave.
2. **POST -** Kjo metodë përdoret për të krijuar një numër të caktuar të të dhënave. P.sh., POST api/patient/10000 ruan të dhëna në databaze për 10000 pacientë.
3. **PUT** – Kjo metodë përdoret për të azhurnuar një numër të caktuar të të dhënave në databazë.
4. **DELETE** – Kjo metodë përdoret për të fshirë një numër të caktuar të të dhënave në databazë.

4.3 Zhvillimi i Ueb aplikacionit

Për realizimin e ndërfaqes së aplikacionit (UI) është përdorur gjuha programuese ReactJS. Qëllimi i ueb aplikacionit është ruajtja dhe menaxhimi i të dhënave të pacientëve. Përveq ruajtjes, ofrohet mundsia e leximit, azhurnimit, si dhe fshirjes së të dhënave. Në vazhdim do të shfaqim pamje të aplikacionit si dhe do të shpjegojmë mënyrën e përdorimit të aplikacionit.

Në momentin e hapjes së aplikacionit, përdoruesi i qaset faqes kryesore e cila shfaq statistikat e kohës së marrë për realizimin e kërkesave (metodave) crud në bazat e të dhënave “SQL” dhe “MongoDB”. Në vazhdim do të paraqesim detaje të këtyre statistikave, duke filluar nga statistikat e kohës mesatare për rreshta/dokumente.

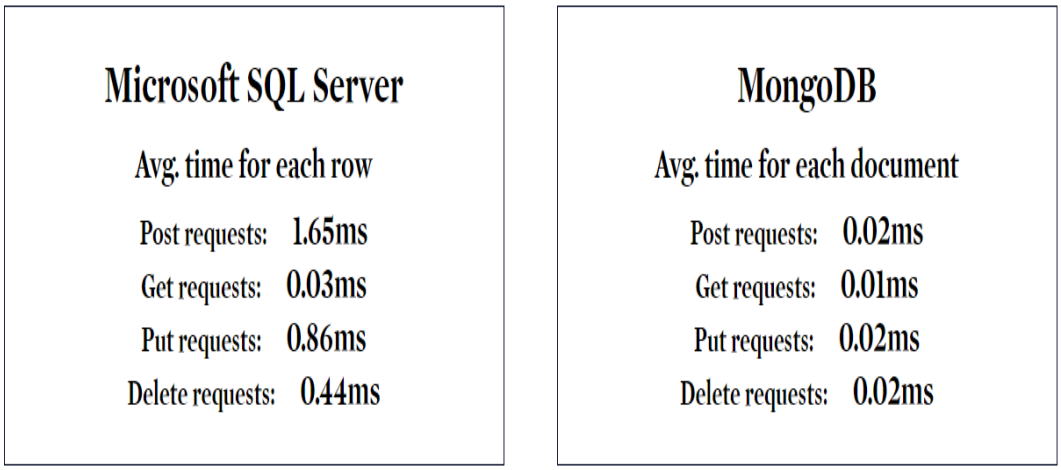

Figure 16. Statistikat e kohës mesatare së realizimit të operacioneve CRUD për cdo rresht/dokument

Në figurën e mësiperme shihet se ka rezultate më të mira për MongoDB, duke pasur parasysh kohën më të mirë të realizimit të operacioneve në bazën e të dhënave. Statistikat

tregojnë që për realizimin e kërkesës “post” apo krijimit të një rreshti/dokumenti në bazën e të dhënave SQL Server kërkohen **1.65ms**, ndërsa në MongoDB kërkohen **0.02ms**.
Realizimi i kërkesës për lexim të një rreshti apo dokumenti kërkon pak a shumë kohë të njejtë, pasi që sic shihet nuk është shfaqur diferencë e madhe në mes të dy rezultateve, pra **0.03ms** për SQL Server ndërsa **0.01ms** për MongoDB.
Për editimin ose azhurnimin e një rreshti apo dokumenti në bazën e të dhënave SQL Server kërkohen përafërsisht **0.86ms**, ndërsa në MongoDB kërkohen **0.02ms**.
Rezultatet për fshirjen e një rreshti ose dokumenti në bazë të të dhënave tregojnë se në SQL Server kërkohen **0.44ms** ndërsa në MongoDB kërkohen **0.02ms**.

Përvec statistikave për kohën mesatare të realizimit të operacioneve CRUD, janë bërë edhe testime tjera për matjen e përformancës. Testimet që janë bërë tregojnë kohën e marrë për realizimin e operacioneve CRUD për 10000, 25000 dhe 50000 rekorde apo dokumente në tërësi. Në vazhdim do t’i paraqesim rezultatet e këtyre statistikave. Në anën e majtë të këtyre statistikave është paraqitur koha në milisekonda, ndërsa në anen e poshtme është paraqitur numri i rekordeve apo dokumenteve.

Sipas statistikave të shfaqura në figurën e mëposhtme shihet se për operacionin e insertimit të një numri më të madh të të dhënave, MongoDB ka një përformancë dukshëm më të mirë.

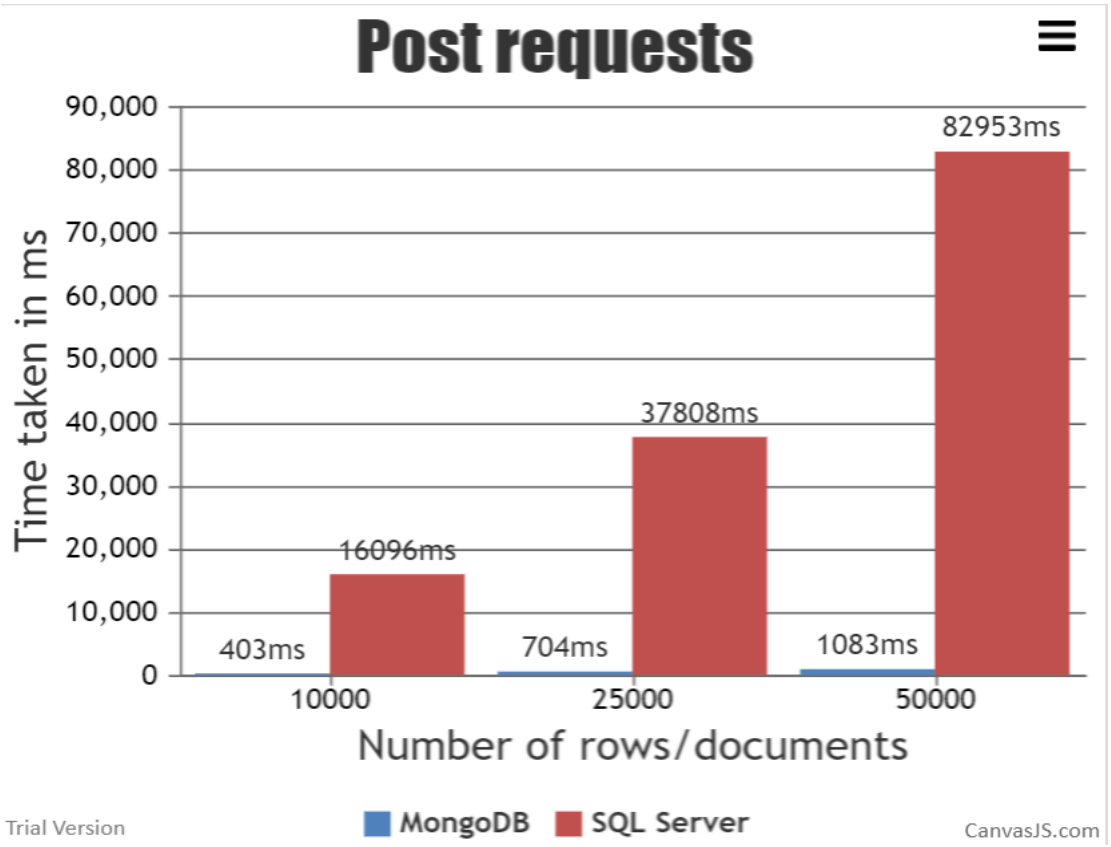

Figure 17. Statistikat e përformancës për krijimin e rekordeve/dokumenteve

Sa i përket operacionit të leximit nuk është shfaqur ndonjë dallim shumë i madh në shpejtësi, mirëpo prapë se prapë prin MongoDB. Statistikat për këtë operacion janë shfaqur në figurën e mëposhtme.

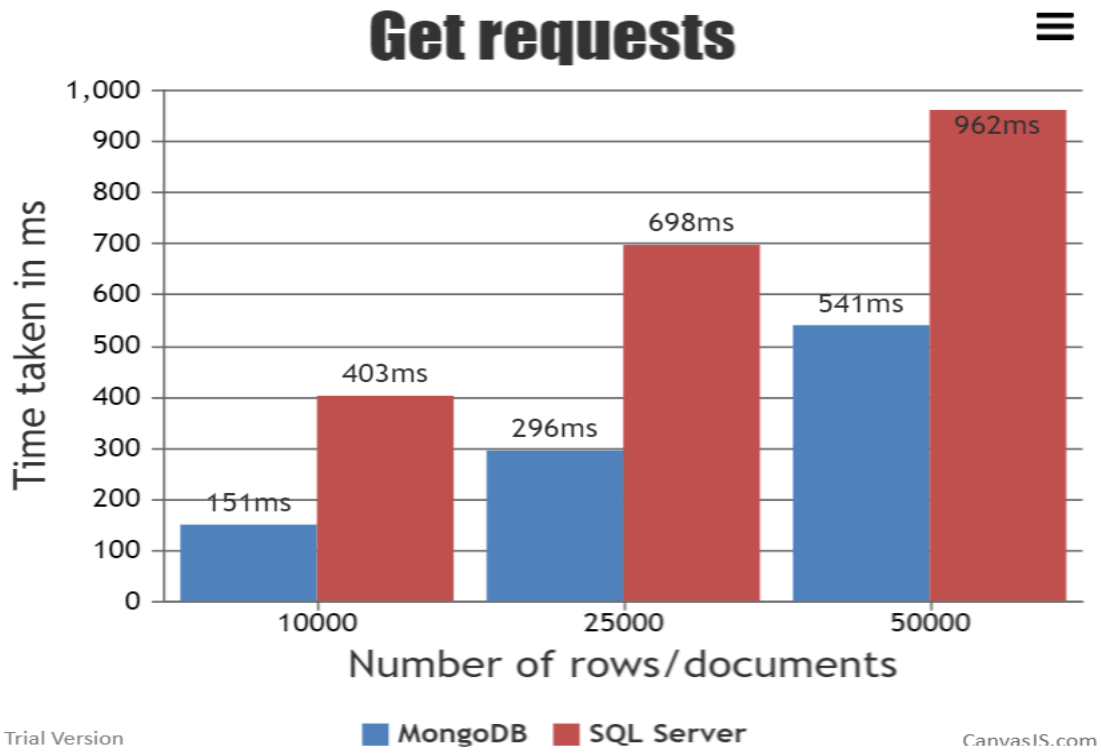

Figure 18. Statistikat e përformancës për leximin e rekordeve/dokumenteve

Shpejtësi më të madhe ka MongoDB kundrejt SQL Server-it edhe për nga përformanca e operacionit të editimit të të dhënave. Statistikat janë shfaqur në figurën e mëposhtme.

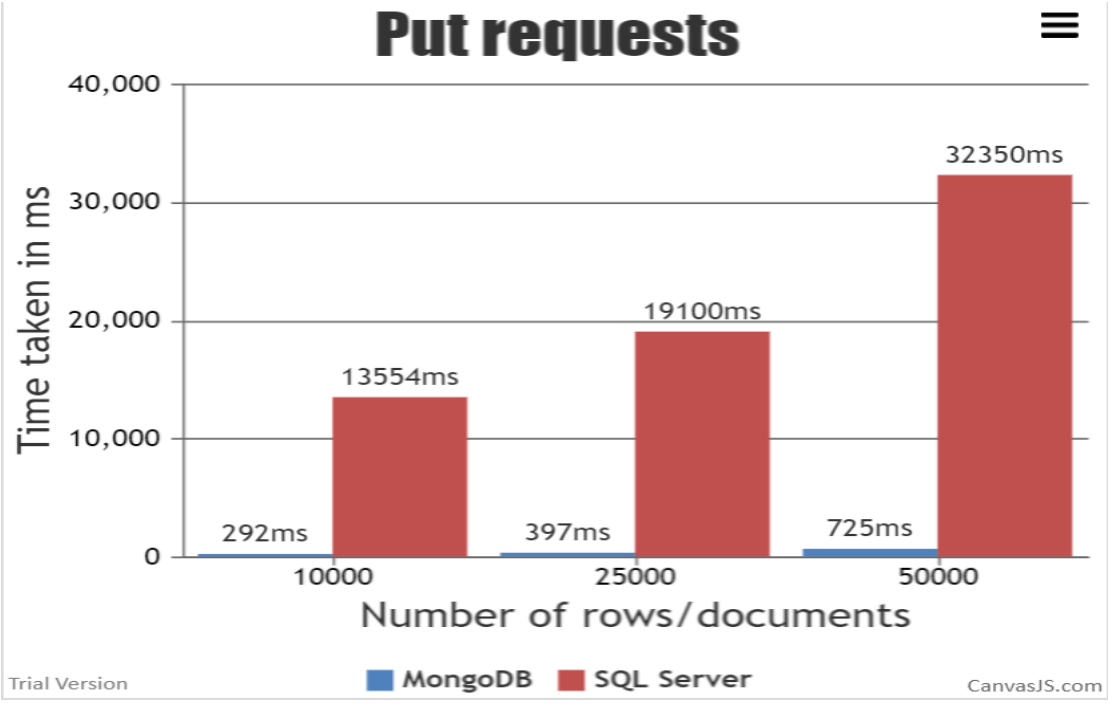

Figure 19. Statistikat e përformancës për editimin e rekordeve/dokumenteve

Operacioni i fundit për të cilin kemi vlerësuar përformancën e bazave të të dhënave, është operacioni i fshirjes. Edhe ky operacion tregon për përformancë më të mirë të MongoDB, sic shihet në figurën e mëposhtme.

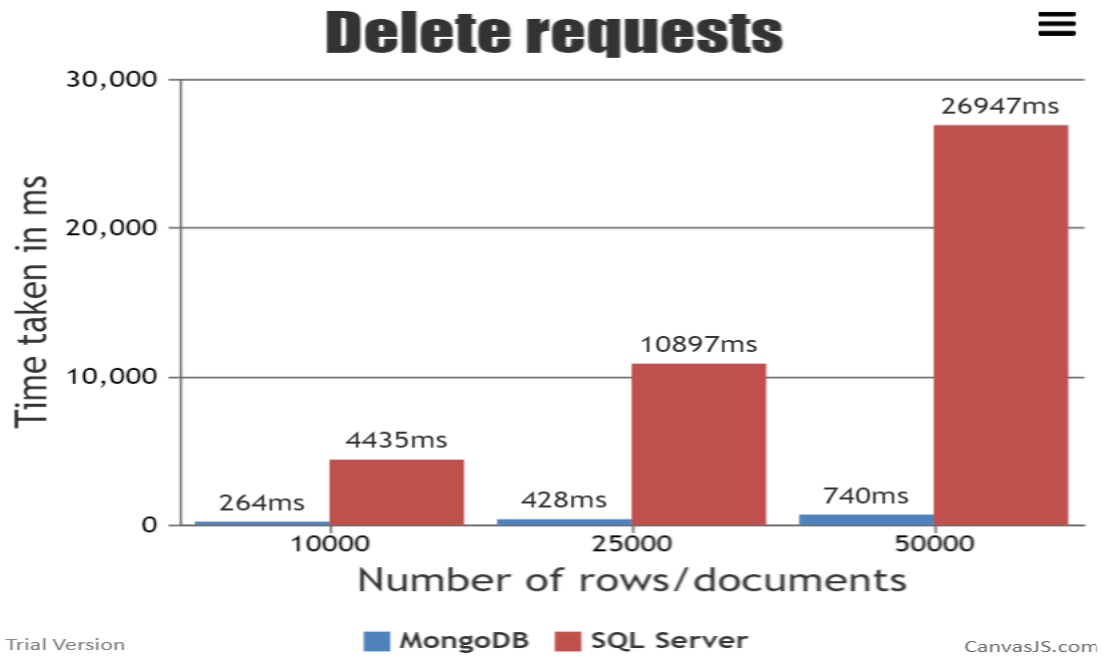

Figure 20. Statistikat e përformancës për fshirjen e rekordeve/dokumenteve

Për navigim nëpër faqet tjera të aplikacionit, shfrytëzohet shiriti anësor në anën e majtë (sidebar), në të cilin shfaqet menyja e navigimit që përmban gjithsej 5 faqe.
Për nga struktura, faqet Patients, Doctors dhe Treatments, janë të njëjta. Ato shfaqin të dhënat e ruajtura në formë të tabelës së të dhënave, si dhe ofrojnë mundësinë e shtimit, leximit, azhurnimit dhe fshirjes së të dhënave. Në vazhdim do të shfaqim pamje për secilën prej tyre.

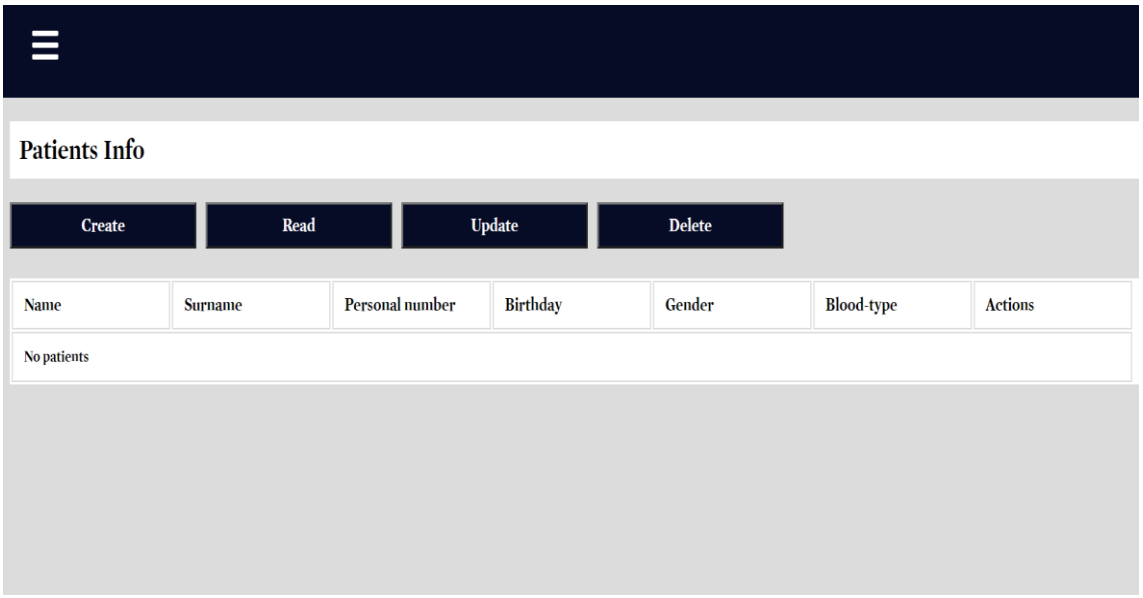

Figure 21. Pamja e faqes "patients" në ueb aplikacion

Nëse klikojmë në butonin Create, hapet modali (forma) për krijimin apo futjen e të dhënave për pacientë, sic shihet në figurën e mëposhtme.

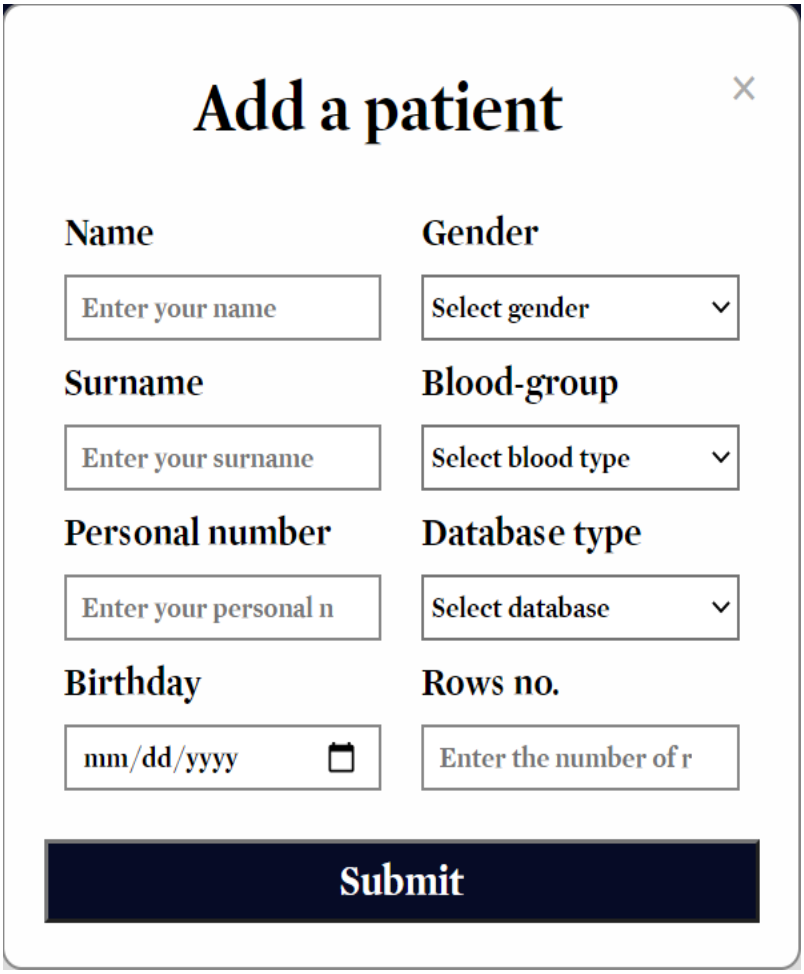

Figure 22. Modali për krijimin e pacientëve

Në këtë modal duhet të plotësohen fushat me të dhënat e caktuara, përzgjedhet baza e të dhënave në të cilën përdoruesi dëshiron të ruajë ato të dhëna, si dhe numrin e rreshtave. Kur plotësohet forma, klikohet butoni Submit dhe dergohet kërkesa në server. Serveri pranon kërkesën dhe kthen si përgjigje kohën e marrur për realizimin e atij veprimi në bazë të të dhënave.

Nëse klikohet butoni Read, atëherë hapet forma tjetër për leximin e një numri të caktuar të të dhënave të cilat përcaktohen përmes fushës së parë. Fusha e parë përcakton numrin e të dhënave të cilat do të shfaqen në tabelën e të dhënave të pacientëve, ndërsa fusha e dytë përcakton bazën e të dhënave në të cilën dërgohet ajo kërkesë.

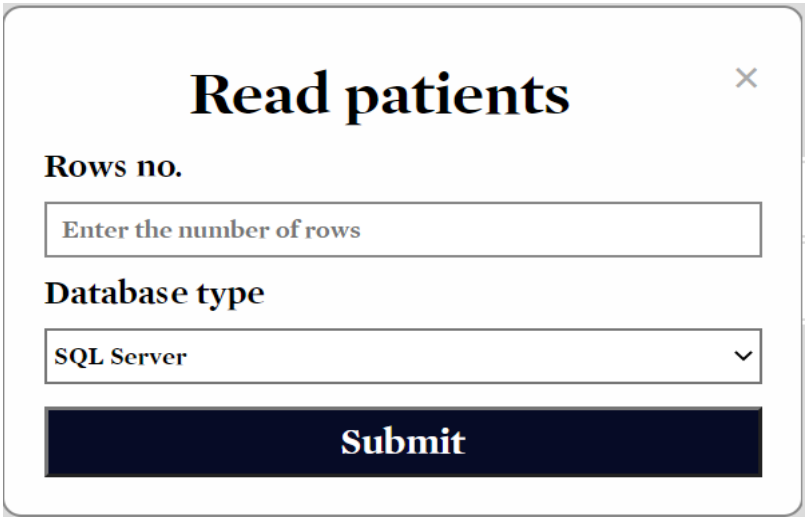

Figure 23. Modali për leximin e të dhënave të pacientëve

Modali (forma) për azhurnimin e të dhënave të pacientëve hapet kur klikohet butoni Update. Në këtë formë kërkohet plotësimi i të dhënave të cilat duhet të editohen në bazë të të dhënave.

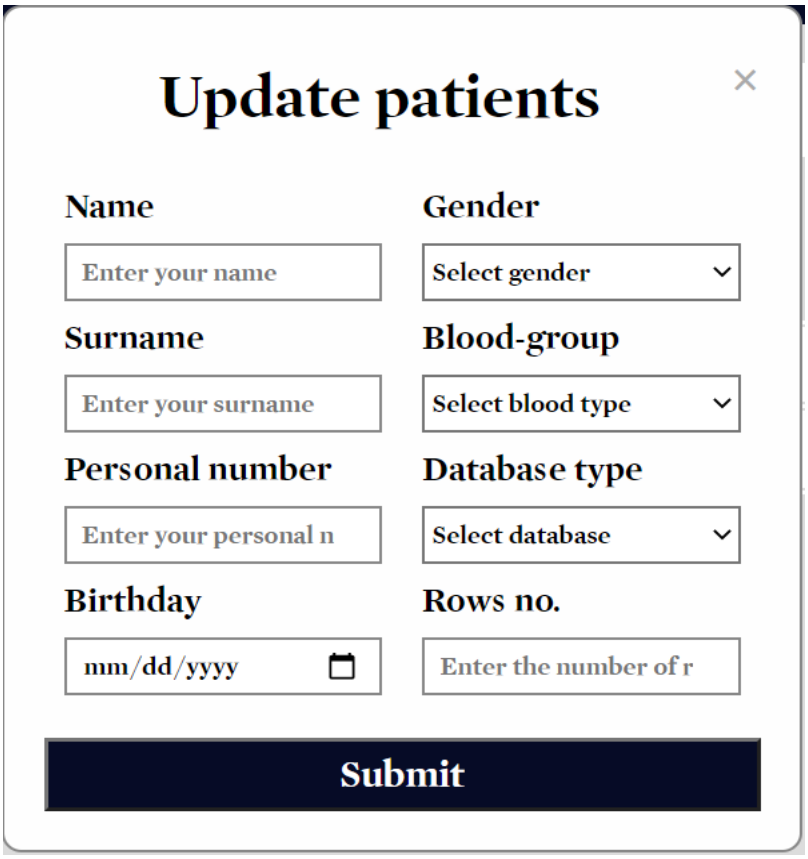

Figure 24. Modali për editimin e të dhënave të pacientëve

Për fshirjen e të dhënave të pacientëve nga baza e të dhënave duhet të klikohet në butonin "delete". Me klikimin e butonit hapet forma e njëjtë sikur forma për leximin e të dhënave.

Pra, kërkohet plotësimi i fushave të numrit të rekordeve të cilat shfrytëzuesi dëshiron t'i fshijë dhe përcaktimi i bazës së të dhënave në të cilën dërgohet ajo kërkesë, sic shihet në figurën e mëposhtme.

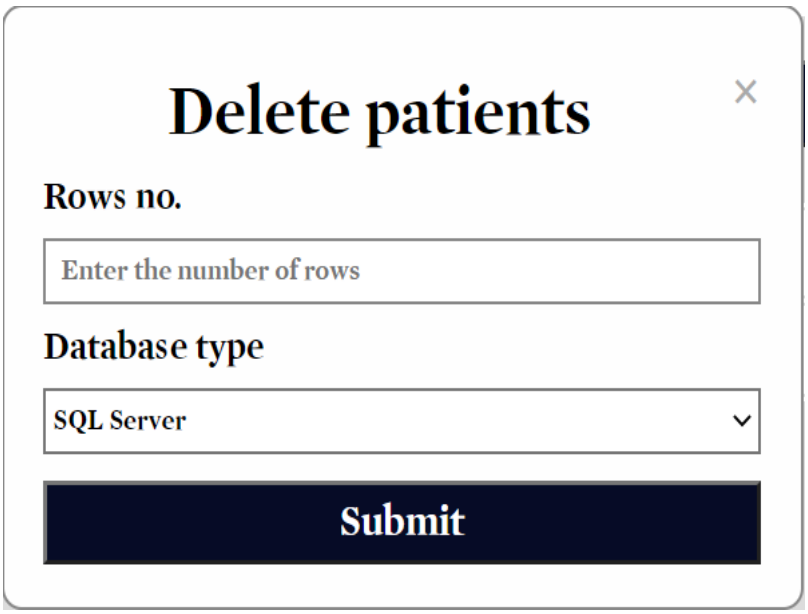

Figure 25. Modali për fshirje të të dhënave të pacientëve

Pas klikimit të butonit Submit, me rastin e dërgimit të suksesshëm të të dhënave në bazën e të dhënave të caktuar, hapet një dritare tjetër e cila informon shfrytëzuesin për përfundimin e veprimit të caktuar me sukses, si dhe shfaq kohën e realizimit të atij operacioni ose veprimi. Shihni figurën e mëposhtme për më shume detaje.

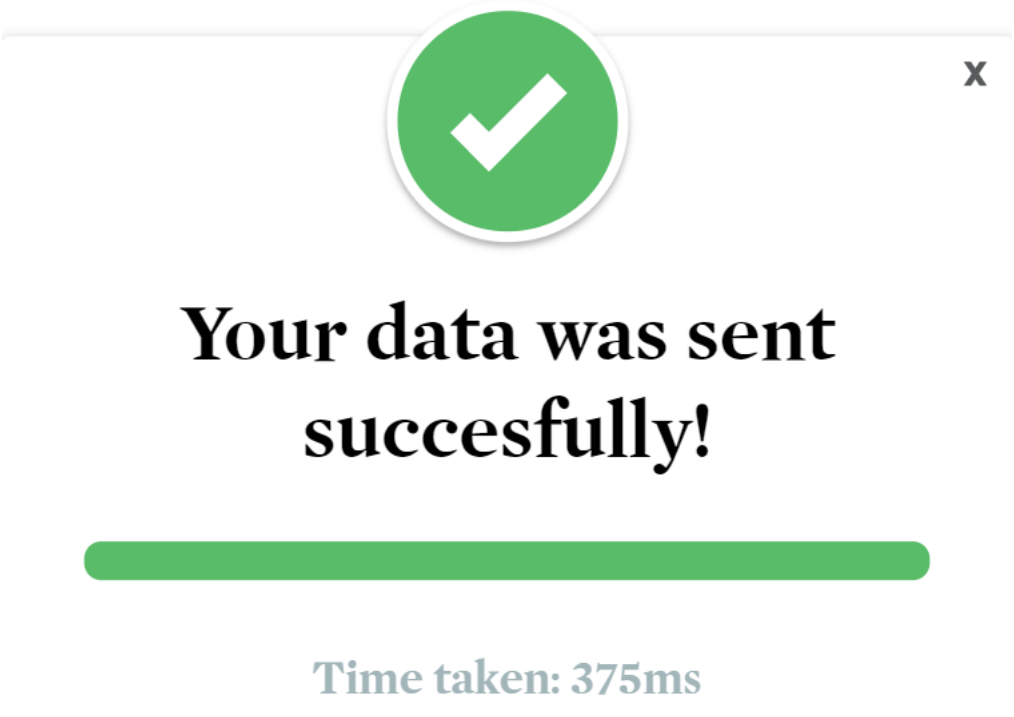

Figure 26. Dritarja për realizimin e suksesshëm të operacionit në bazën e të dhënave të përcaktuar

5 Përfundimi

Qëllimi i këtij punimi ka qenë që të analizohet përformanca e bazave të të dhënave relacionale dhe jo-relacionale. Në punim është treguar në detaje për specifikat e këtyre dy llojeve të bazave të të dhënave, sistemeve të tyre si dhe përparësive apo mangësive të tyre.

Krahasimi i performancës është bërë përmes zhvillimit të një aplikacioni i cili matë kohën e realizimit të katër operacioneve themelore të bazave të të dhënave të cilat janë:

- Leximi
- Shkrimi
- Editimi
- Fshirja

Realizimi i këtyre operacioneve është mundësuar përmes zhvillimit të API-ve duke shfrytëzuar sistemin ASP.NET Web API. Rezultatet e fituara tregojnë për një përformancë dukshëm më të mirë të MongoDB-së në secilin prej operacioneve të lartë-permendura, që do të thotë se MongoDB ka shpejtësi më të madhe për trajtimin e të dhënave të mëdha (big data).

Në të ardhmen është menduar që të bëhet krahasimi i përformancës së bazave të të dhënave relacionale dhe jo-relacionale në sisteme me të dhëna reale.

6 Literatura

1. A brief history of NoSQL, https://www.dataversity.net/a-brief-history-of-non-relational-databases/, qasur më 25.12.2020.
2. Dere, Mohan (2018-02-19). "How to integrate create-react-app with all the libraries you need to make a great app", qasur më 26.12.2020.
3. Samp, Jon (2018-01-13). "React Router to Redux First Router", qasur më 26.12.2020.
4. React, Facebook. "Component and Props", qasur më 26.12.2020.
5. React vs Angular vs Vue-js, https://www.indeed.com/, qasur më 26.12.2020.
6. Mozilla Developer Network, "CSS developer guide", qasur më 28.12.2020.
7. Fielding, Roy (June 2014). "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, Section 4". Internet Engineering Task Force (IETF), qasur më 05.01.2021
8. Lardinois, Frederic (April 29, 2015)."Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor For OS X, Linux And Windows" .
9. StackOverflow Insights. "Developer Survey Results 2019 - Most Popular Development Environments", qasur më 07.02.2021
10. MSDN: "Introducing SQL Server Management Studio", qasur më 11.02.2021.
11. Codd, E.F (1970). A relational Model of Data for Large Shared Data Banks". Faqe 377-387.
12. Ambler, Scott. "Relational Databases 101: Looking at the Whole Picture", qasur më 17.01.2021
13. Carey Wodehouse, SQL vs NoSQL databases – what's the difference, qasur më 18.01.2021
14. Drake, Mark (2020). "Understanding Relational Databases", qasur më 17.01.2021
15. Geeks for Geeks, "Acid properties in DBMS", qasur më 23.01.2021
16. NoSQL Definition, http://nosql-database.org/, qasur më 23.01.2021
17. Leavitt, Neal (2010). "Will NoSQL Databases live up to their Promise?", qasur më 25.01.2021
18. Vogels, Werner (18 Janar 2012). "Amazon DynamoDB – a Fast and Scalable NoSQL Database Service Designed for Internet Scale Applications", qasur më 18.01.2021
19. Sandy (14 Janar 2011). "Key value stores and the NoSQL movement" , qasur më 25.01.2021.
20. Rasheed, Yasmin (Prill 2019). "Overview of the current status of NoSQL databases". Faqe 48-50.
21. Facebook, React. "React – A javascript library for building user interfaces", qasur më 27.01.2021.
22. Li, Yishan (Gusht 2013). "A performance comparison of SQL and NoSQL Databases". Faqe 17-18.
23. Seobility Wiki, "REST API" , qasur më 15.02.2021
24. Cabral, Bruno (2015). "Choosing the right NoSQL database for thejob: a quality attribute evaluation", faqe 6.
25. Choudhury, Sid (Qershor 2019). "How data sharding works in a distributed SQL Database", qasur më 27.02.2021.

Lista e figurave

Lista e tabelave