Универзитет у Београду

Електротехнички факултет



Container-изација MERN web апликације

Дипломски рад

|  |  |
| --- | --- |
| Ментор: | Кандидат: |
| Др Зоран Чича, доктор електротехничких наука | Вук Ивановић 2018/0367 |

Београд, Јул 2022.

Садржај

[Садржај i](#_Toc461025970)

[1. Увод 1](#_Toc461025971)

[2. priprema okruženja 2](#_Toc461025972)

[3. bACKEND 4](#_Toc461025973)

[3.1. 4](#_Toc461025974)

[3.2. Node 4](#_Toc461025975)

[3.3. Express 5](#_Toc461025976)

[3.4. MongoDB i MongoDB Atlas 7](#_Toc461025977)

[3.5. Postman 7](#_Toc461025978)

[4. FRONTEND 9](#_Toc461025982)

[4.1. Организација окружења 9](#_Toc461025983)

[4.2. React 10](#_Toc461025984)

[4.3. API 11](#_Toc461025985)

5[. docker 4](#_Toc461025973)

[3.1. Увод у контаинере 4](#_Toc461025974)

[3.2. Docker 4](#_Toc461025975)

[3.3. Docker Desktop 5](#_Toc461025976)

[3.4. DockerFile i Docker compose orkestracija 7](#_Toc461025977)

[Литература 15](#_Toc461025996)

[Списак скраћеница 16](#_Toc461025997)

[Списак слика 17](#_Toc461025998)

[Списак табела 18](#_Toc461025999)

1. Увод

У овом раду обрађена је идеја контаинеризације једне “fullstack” веб апликације, путем Docker платформе. Главна идеја контаинеризације неке апликације било да је она веб типа или неког другог типа је да се репетитивни послови нпр. брига о оперативном систему на ком се апликације покреће или да ли су одређене библиотеке дефинисане и правилно инсталиране, аутоматизују и да се инжењери њима не баве директно већ ће све ове послове обављати сам Docker.

Појам контаинеризације је форма визуелизације где апликација ради у изолираном простору, где користи дељени оперативни систем. Дакле контаинери практично постају потпуно упаковано и скалабилно окружење.

Узмимо за пример једну једноставну апликацију коју зависи од одређених библиотека које се често унапређују и побољшавају. У оваквим апликацијама се управо и види прва примена конатинер архитектуре. Када се подигне контаинер са одређеном сликом(о којој ће бити речи даље у раду), Docker engine брине о свим потребним библиотекама и инсталацијама.

Са технологијом контаинеризације постоји велика исплативост у коришћењу микросервиса, наиме сваки миркосервис се спакује у контаинер и након тога цела апликација постаје знатно ефикаснија и као таква и користи много мање ресурса на вируталној машини, серверу или облаку на коме живи.

Главна разлика између контаинеризације и виртуализације је та да се контаинеризација дешава на нивоу оперативног система, док у виртуализацији то није случај. Дакле сви контаинери који су подигнути у сврси једне апликације деле један оперативни систем, где са друге стране све вируталне машине које се дижу у сврху једне апликације захтевају одрећени део ресурса у систем, серверу или слично.

Навео бих неколика сервиса и апликација у којима је коришћење контаинера практично постало стандард, иако практично сви типови апликација се могу контаинеризовати. Реч је за почетак о базама података. Свака база података коју користи одређена апликација може се контаинеризовати и тако може имати своју сопствену базу података коју ће користити за своје потребе.

Након тога следећи пример употребе контаинера је у подизању веб сервера. Наиме конфигурисање и подизање веб сервера захтева велики број корака и подешавања који су само по себи напорни и пре свега изводи се из доста корака. Коришћењем контаинера потребно је искуцати свега пар линија кода и веб сервер се може подићи. Још један пример су и претходно наведени микросервиси. Микросервисна архитектура у великој већини се данас реализује и само преко контаинера. Сваки микросервис постаје сопствени контаинер који ће даље функцтионисати између себе и заједно формирати целу апликацију.

1. Припрема окружења

У овом поглављу биће речи о постављању самог окружења за развој апликације као и инсталације Docker-а и Docker Desktop-a који ће служити у контаинеризацији апликације. На изабрано је развојно окружење у коме ће се развијати програмски код апликације. У питању је Microsoft Visual Studio Code. Конкретно то је једноставно и меморијски мало решење које пружа кориснику лакшу обраду кода у смислу додатака који олакшвају рад и развијање програмског кода.

Као што је претходно речено у овом раду биће развијена “fullstack” апликација. У питању је апликација која представља веб продавницу. Дакле биће развијен део апликације коју ће корисник користити и преко ње комуницирати са сервером, као и сам сервер преко ког ће се одвијати комуникација са базом. Што се тиче технологија које су изабране за саму апликацију, на страни клиента(фронтенд) коришћена је библиотека тј “framework” React.js. На страни сервера коришћен је Node.јѕ као и Express.јѕ. Изабрана база је МоngoDB. Више детаља о томе зашто су изабране баш ове технологије биће дато током рада у секцијама које се баве конкретном технологијом.

Такође, за потребе контаинеризације апликације користе се Docker и Docker Desktop, који дају графички кориснички интерфејс преко кога је знатно лакше визуализирати како тачно функционишу и раде контаинери које подижемо. Такође битно је навести да ће се у овом раду користити локална машина, рачунар и да ће се на њему дизати само контаинери. Наравно могуће је подизати конатинери на неки сервервима и то је у већини случајева заправо и случај. У великим компанијама управо је то и приступ којим се иде. Дакле у оквиру приватног окружења подигне се сервер дефинисан тачно за потребу контаинера и он ће се искључиво користити у те сврхе.

За потребе Docker-a потребно је инсталирати и успоставити и Linux виртуалну машину. Битно је навести да Docker сам по себи ради на Linux Kernel-у те је из тог разлога потребно подићи вируталну машину да би он могао уопште да фунцкционише.

На крају инсталиран је и Postman који ће користити за теситрање функционалности серверове комуникације са базом података.

1. BACKEND

У овоме поглављу биће дато решење сервера и базе података и како се реализовала комуникација сервера са самом базом.

* 1. Увод у Node.js

Историјски гледано ЈаваСкрипт је примарно коришћен на клијентској страни, где су скрипте написане у ЈаваСкрипту биле уграђене у HTML странице, како би се извршиле на клијентској страни у веб прегледачу. Node.js предстаља вишеплатформско ЈаваСкрипт окружење отвореног кода за извршавање JavaScript-а на серверској страни

Node.js омогућава да се JavaScript користи за скрипте на серверској страни које омогућавају да се садржај динамичних веб страница генерише на серверу пре него што се пошаље до веб прегледача корисника. Због тога је Node.js постао један од основних елемената парадигме " JavaScript свуда" јер омогућава униформисање развој веб апликација у једном програмском језику, без потребе да се за скрипте на серверској страни користи различит програмски језик.једном реченицом најавити све делове те целине.

Node.js поседује архитектуру базирану на догађајима која је способна да обавља асинхроне улазе/излазе. Овакав избор архитектуре омогућава оптимизацију пропусности и скалабилности у веб апликацијама са много улазно/излазних операција као и за веб апликације у реалном времену (нпр. програми за комуникацију у реалном времену и игрице у веб прегледачу).

* 1. Увод у АPI

API (*Application Programming Interface* — програмски интерфејс апликације) је програмијабилни интерфејс који дефинише начине на које апликације могу да захтевају услуге од библиотека или оперативних система. API одређује речник и конвенције позивања које програмер треба да примени како би користио сервисе. То може да укључује спецификације за функције, структуре података, класе као и протоколе који се користе за комуникацију између софтвера који тражи услугу и библиотеке.

API може да буде:

* Језички завистан, тј. доступан само за дати програмски језик, коришћењем синтаксе и елемената тог језика
* Језички независтан, тј. писан на начин на који омогућава да API буде позван од стране различитих програмских језика, ово представља пожељну особину API-а

Сам API је у великој мери апстрактан у смислу спецификације и контролише понашање објеката спецификованих интерфејсом. Софтвер који пружа функционалност коју API опсије назива се имплементацијом API-а.

Акроним API понекад може и да се користи као референца не само на цео интерфејс, већ и на појединачну фунцкију, или чак на скуп више API-а које нека организације испоручује. Стога обим значења обично одређује фирма или документ који преноси информацију.

* 1. Увод у Express.js

Express.js је веб апликација у оквиру Node.js сервера, дизајнирана је за израду једностраничних, вишестраничних и хибридних веб апликација. Node.js представља развојно окружење а не библиотеку за развијање апликације те је управо из тог разлога потребна једна таква бибилиотека која помаже у развијању саме апликације. То је дефакто стандар за серверски оквир за Node.js.

**3.4 Увод у МongoDB**

MongoDB је водећа NoSQL база података. MongoDB чини интеграцију података у многим апликацијама једноставнијом и брзом. У складу са растућим интересовањем за алтернативне системе за управљање базама података, који се разликују од традиционалних релационих база података, појавио се концепт такозваних NoSQL база података, које не користе SQL за повезивање, нерелационе су, дистрибуиране, отвореног кода и хоризонтално скалабилне.

Написана је у C++ језику и отвореног је кода. MongoDB чува податке као JSON документе са динамичким шемама. JSON (JavaScript Object Notation) је отворени стандард заснован на тексту, осмишљен за размену података који су погодни за читање људима. MongoDB је прихваћен као бацкенд софтвер бројних значајних веб-сајтова и сервиса.

Њена динамичка шема и објектно-оријентисана структура, чине је правим избором за аналитику у реалном времену, као и за е-комерц, мобилне апликације, архивирање и слично. Познати случајеви коришћења MongoDB обухватају “big data” податке, управљање садржајем, мобилну и друштвену инфраструктуру и многе друге. Такодје, појављују се изазови за коришћење MongoDB за Bussines Inteligence modele.

MongoDB има званичне драјвере за разне популарне програмске језике и развојна окружења. Постоје и многи незванични драјвери, подржани од стране отворених заједница, за готово све програмске језике и различита окружења.

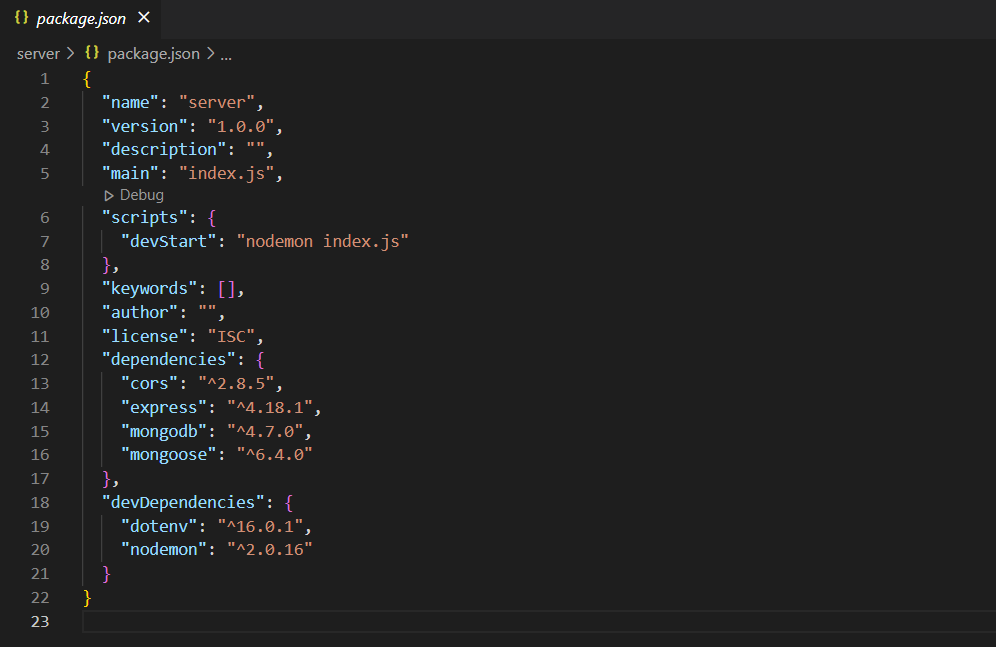
MongoDB Atlas је потпуно функционала веб апликација која олакшава корисницима да комуницирају са базом и даје једноставан графички кориснички интерфејс за посматрање промена у бази. Такође, када корисник направи налог на њему он подиже кластер на коме ће живети сама база. У овом пројекту база је реализована управо овим начином.

**3.5 Развијање backend-a**

Структура пројекта је једноставна. Направљена су два фолдера, један у коме ће се налазити клиентска апликација (frontend) и један у коме ће се налазити серверска апликација (backend). Након што се у Microsoft Visual Studio Code-у отвори дати директоријум и позиционира се у фолдер за серверску апликацију отвара се интегрисани терминал у Microsoft Visual Studio Code-у и иницијализује се Node пројекат командом: “npm init -y”.

Овом командом у датом директоријуму се појавлјује package.json фајл у коме се налазе основне информације о самој апликацији. У питању су име апликације, верзија и као најбитнија ставка све библиотеке које се користе у апликацији. Дате библиотеке ће се касније инсталирати командом “npm install”. Сваки пут када се ова команда извршава она покупи све билиотеке које се налазе у package.json фајл-у и инсталира их у апликацији.

Затим су инсталиране и све друге библиотеке потребне за развијање апликације. Инсталирани су mongodb, express као и mongoose пакет који помаже у комуницирању са базом податка. Дат је изглед package.json фајла у коме је претходно речено да ће се налазити све потребне библиотеке под секцијом “dependencies”.



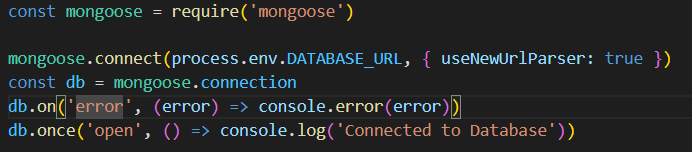
Слика 3.5.1 изглед package.json фајла

Затим креиран је index.js фајл који ће представљати главни фајл у самој апликацији. У оквиру њега се обавља конекција са базом као и покретање самог веб сервера уз помоћ Express-а.

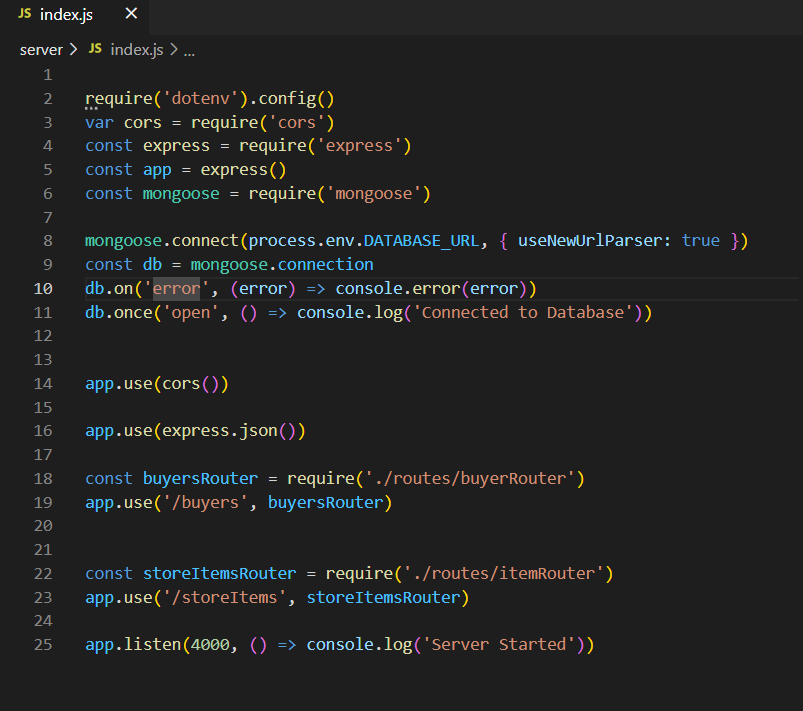
За почетак креиран је сам веб сервер и дефинисан је порт на коме ће се сервер вртети. А након тога успоставља се конекција са самом базом. Након што је креиран налог на MongoDB Atlas-у креирана је колекција. У самој колекцији ће се налазити подаци, у виду докумената. Сама база чува податке само у типу документа и сваки ентитет у бази се сматра документом.

Веза се успоставља након што се у оквиру базе креира корсник са корисничким именом и шифром који ће имати могућност повезивања и управљања њом.

Након тога, база даје “database url” , који представља јединствени токен преко кога ће се наша апликација повезати са њом. Препорука је да се овај токен не доставља онлајн те је из тог разлога, коришћен и додатни пакет “dotenv” у коме ће се чувати сам токен, а онда апликација уз помоћ задате библиотеке лако долази до њега.



Слика 3.5.2 конекција са базом



Слика 3.5.3 излгед index.js фајла

За потребе базе морамо дефинисати шеме модела података који ће се слати и дохватати као и чувати у самој бази. У овој апликацији дефинисана су два модела. Један за потребе чувања информација о купцима и куповини коју су направили. А друга за потребе намирница и продуката који ће се из базе дохватати и приказивати на самом сајту.

Што се тиче купаца, дефинисано је неколика поља која су потребна. Ради се о имену, презимену, маилу, адреси као и низу продуката који су они наручили у једној сесији. У овом пројекту није било рада са колачићима и чувања информација о корисницима након што они заврше коришћење апликације.



Слика 3.5.4 излед шеме о купцима

Са друге стране такоће је дефинисана и шема модела продуката који се продају. Од датих инфромација чувају се име продукта, цена и слика самог продукта.

Битно је напоменути да примарни кључеви докумената у бази не морају имплицитно да се чувају. База сама по себи сваком новододатом објекту додају јединствени идентификациони број.

Text

Description automatically generated

Слика 3.5.5 изглед шеме о продуктима у продавници

Након дефинисања модела прелазимо на развијање API-а преко ког ће сервер комуницирати са базом података. Креиране су све CRUD (create, read, update, delete) методе. Дакле ради се о методи за дохватање података из базе, мењања података о већ постојећем документу у бази, брисање одређеног документа у бази и додавање новог документа у базу. Свака метода гађа одређени урл преко којег прави захтев самој бази.

* Дохватање свих докумената:
* у овој методи након што се дохвати већ дефинисана шема модела тај тип модела се преко методе .find() доставља бази и она враћа све документе који одговарају датом моделу
* Дохватање једног документа:
* ова метода функционше исто као и претходна само се задаје идентификациони број како би се дохватио само тај један документ из базе, у одговору од странје базе се налази документ

A screenshot of a computer

Description automatically generated with medium confidence

Слика 3.5.6 дохватање докумената из базе

* Постављање документа у базу:
* ова метода захтева да се у телу захтева задају параметри који требају да се пошаљу бази а након тога се командом .save() они чувају у самој бази података



Слика 3.5.7 постављање документа у базу

* Брисање и модификивање докумената:
* што се тиче брисања докумената такође је потребно проследити идентификациони број документа који желимо обрисати и командом .remove() се брише сам документ

Text

Description automatically generated

Слика 3.5.8 брисање документа из базе

* Модификовање документа:
* модификовање документа исто захтеба идентификациони број али мали додатак који треба додати је да се прође кроз базу и провери да ли тај документ постоји и онда порменити дате парамтере

Text

Description automatically generated

Слика 3.5.9 модификовање документа из базе

Свака од ових метода заснована је на једној помоћној функцији под називом “getBuyer()” која се извршава сваки пут када се обавља нека операција. Реч је о асинхроној методи јер је потребно сачекати да се подаци из базе дохвате.

A screenshot of a computer

Description automatically generated with medium confidence

Слика 3.5.10 дохватање купца по идентификационом броју

**3.6 Postman и провера серверске апликације, MongoDB Atlas**

Postman је платформа дизајнирана да помогне инжењерима да дизајнирају, праве и тестирају њихове API-је. Има преко 20 милиона корисника и један је од најбољих и најбржих у својој области. Преко њега могуће је тестирати и слати захтеве са сервера на базу и проверавати да ли сервер добија добре или лоше одговоре од стране базе.

У овој апликацији коришћен је прво ради убацивања докумената о продуктима у базу а након тога тестирања дохватања истих који ће бити дохватани од стране клиентске апликације и бити приказани на екрани где ће купац моћи да их наручи.

Када се у Postman-у прави захтев за постављање података у базу мора се ручно задати тело захтева у коме ће се навести подаци који се шаљу бази од којих ће она направити документ. У питању је JSON тип објекта. Овај поступак се у овом случају урадио три пута за три различита продукта. База ће у одговору на захтев вратити исти тај објекат потрвђујући да је захтев довро прошао.

Graphical user interface, text, application, email

Description automatically generated

Слика 3.6.1 прављење POST захтева бази за додавање продуката преко Postman-а

Такође одрађено је тестирање GET захтева да би се касније могао исти захтев искористити на клиентској апликацији.

Са друге стране одрађен је и тест додавања наруџбине купаца и њихових информација у базу, такође у Postman-у.

Graphical user interface, text, application, email

Description automatically generated

Слика 3.6.2 прављење GET захтева бази за дохватање продуката преко Postman-а

Graphical user interface, text, application, email

Description automatically generated

Слика 3.6.3 GET захтев за дохватање купаца преко Postman-а, приказан је наруџбина једног купца

Са друге стране осрвнимо се и на проверу података у самој бази која ће се одбвијати преко MongoDB Atlasa-a. Након што се у њему направи налог и конектује се база и колекција као и сервер на коме ће стајати, у питању су неки од MongoDB сервера који се налазе у Европи, можемо посматрати шта се дешава са нашом базом.

Напоменућу да пошто су претходно дефинисане две шеме модела за базу у датој колекцији налазиће се једна база која је овај пут дефинисана именом тест а унутар ње две колекције података са датим именима “buyers” и “storeitems”.

У оквиру њих можемо видети све документе које смо предходни убацили у базу путем Postman-а.

Graphical user interface, application, table

Description automatically generated

Слика 3.6.4 изгледа кластера и структура базе у MongoDB Atlasa-у

Graphical user interface, text, application, email

Description automatically generated

Слика 3.6.5 изглед колекције “storeitems” у MongoDB Atlas-у

1. FRONTEND
   1. Увод у React.js

React је JavaScript библиотека отвореног кода која обезбеђује преглед података записаних преко  HTMLa.

React апликације су обично обезбеђенe коришћењем компоненти које садрже додатне компоненте дефинисане као прилагођене HTML ознаке. React обезбеђује програмеру модел у којем подкомпоненте не могу директно да утичу на спољашње компоненте, ефикасно ажурирање HTML документа при промени података и јасно раздвајање компоненти на данашњим једностраничним апликацијама.

Он је подржан од стране Facebook, Instragram и заједнице појединачних програмера и корпорација.

* **Једносмерни ток података:**

Својства, скуп непроменљива вредности, прослеђени су приказивачу компоненти као својство у његовим HTML ознакама. Компоненте не могу директно да промене било које својство које им је прослеђено али може бити прослеђена опозивајућа функција да промени вредности. Овај механизам је изражен као „својства иду доле; акције горе”.

На пример, компонента колица за куповину може да укључује више компоненти производа. Визуелизација производа користи само својства која су им прослеђена и то не може да утиче на укупну вредност колица за куповину. Међутим, производ може бити прослеђен опозивајућој функцији као својство које би било позвано када активирамо дугме „обриши овај производ” и тада та функција утиче на укупан рачун.

* **Виртуелни DOM**

Још једна специфичност је да се користи виртуелни DOМ. React одржава у меморији кеш података, израчунава актуелне разлике, а затим ефикасно ажурира прегледачев DOM. Ово омогућава програмерима да пишу код као да се цела страна освежава при свакој промени, док React библиотека приказује само подкомпоненте које су заиста промењене.

На пример, компоненте колица за куповину могу бити записане да прикажу целокупан садржај колица при било којој промени података. Уколико подкомпоненте производа немају промене у својству, биће употребљен кеширани приказ. Ово значи да ће релативно споре пуне измене у прегледачевом DOM-у бити избегнуте. Поред тога, уколико се промени број производа, подкомпоненте производа ће бити приказане; крајњи HTML се може разликовати у само једном члану и само тај члан ће бити измењен у DOM-у.

* **JSX**

Компоненете React-а су обично записане у JSX-у, а Јаваскрипт синтаксно проширење омогућава лако навођење HTML-а и коришћење синтаксе HTML ознака приказаним подкомпонентама.

HTML синтакса обрађена у Јаваскрпиту позива React библиотеку. Програмери такође могу да пишу и у чистом Јаваскрпиту. JSX је сличан фејсбуковом синтаксном проширењу за PHP, XHP.

* **Архитектура из HTML-а**

Основне архитектуре React примењују се иза приказаног HTML у претраживачу. На пример, Facebook има динамичке графиконе, који се приказује у  <canvas> ознакама, и Netflix и PayPal користе изоморфна учитавања да прикажу исти HTML на серверу и клијенту.

**4.2 Развијање клиентске апликације**

Литература

1. D. Gookin, *Word 2003 for Dummies*, Wiley, 2003.
2. M. Kodialam, T. V. Lakshman, and S. Sengupta, "Traffic-Oblivious Routing for Guaranteed Bandwidth Performance," *Communications Magazine*, 45(4), pp. 46-51, Apr. 2007.
3. M. Antić, A. Smiljanić, "Oblivious Routing Scheme Using Load Balancing Over Shortest Paths", *Proc. of IEEE International Conference on Communications*, 2008.
4. *Understanding Styles in Microsoft Word* [Online]. Available: <http://www.addbalance.com/usersguide/styles.htm> (09.02.2016.)
5. М. Гојгић, "Прописи из области метрологије," *Електропривреда*, 58(2): 84-93, Април-Јун 2006. (<http://www.eps.rs/Elektroprivreda/2-2006.pdf>, 09.02.2016.)

Списак слика

[Слика 3.3.1. Неправилан приказ графика 3](#_Toc445721835)

[Слика 3.3.2. Правилан приказ графика 3](#_Toc445721836)

[Слика 4.1.1. Одабирање стила текста у алату *MS Word 2003*. 3](#_Toc445721837)

[Слика 4.1.2. Одабирање стила текста у алату *MS Word 2007*. 3](#_Toc445721838)

[Слика 4.3.1. Хијерархија наслова текстуалних целина. 3](#_Toc445721839)

[Слика А.1.1. Подешавање маргина на страници. 3](#_Toc445721840)

[Слика А.2.1. Прозор са листом стилова текста. 3](#_Toc445721841)