Зборник радова Факултета Техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/23BE20Karadzic

ТРОСЛОЈНА АРХИТЕКТУРА КАО СОФТВЕРСКО РЕШЕЊЕ СИСТЕМА ЗА ЕВИДЕНЦИЈУ СТУДЕНАТА

THREE-TIER ARCHITECTURE AS A SOFTWARE SOLUTION FOR A STUDENT RECORDS SYSTEM

Игор Караџић, *Факултет Техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом чланку је описана имплементација и рад евиденције студената на одређеним предметима током школске године. Реч је о трослојном систему који укључује имплементацију сервера и клијента, као и креирање и управљање базом података. Пројекат представља приступ развијања web апликација уз технологије и програмска окружења као што су Node, Javascript i MongoDB. Подржано је додавање студената и предмета, додељивање студената да у задатој школској години слушају одређени предмет, дефинисање категорија за дати предмет, унос поена за датог студента на датом предмету у одговарајућим категоријама.*

Кључне речи: *Web, Node, Javascript, MongoDB*

Abstract – *This article describes implementation and functionality of student records on specific subjects during school year. It is about a three-layer system that includes the implementation of a server and a client, as well as the creation and management of a database. Project presents the approach to development of web application including technologies such as Node, Javascript and MongoDB. It supports creating students and subjects, assigning students to specific subjects in a given school year, defining categories for a given subject, assigning points to a given student on a given subject in specific categories.*

Keywords: *Web, Node, Javascript, MongoDB*

1. УВОД

Путем сајта за евиденцију студената корисник је у могућности да прегледа све студенте и предмете активне у систему, да провери за сваког студента које предмете слуша, да прегледа за сваки предмет које категорије има као и колико је поена студент освојио на одређеним категоријама на задатом предмету. У оквиру овог пројекта реализован је систем који се састоји од два основна блока. Први блок представља, серверску страну, колоквијално названу *Back-end*, која укључује серверску апликацију базирану на *Node.js* [1] платформи и њој специфичном *JavaScript* програмском језику, и базу података, која је у овом случају *MongoDB* [2]. Други блок обухвата клијентску страну, илити *Front-end* који садржи *HTML* и *CSS* за креирање и дизајнирање *web* страница као и *Javascript* језик за динамичко управљање садржајем *web* странице.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Предраг Теодоровић, доцент.

Такође је реализована *Android* апликација која је направљена коришћењем *React Native*.

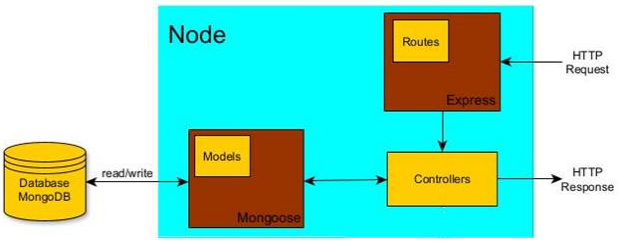
2. СИСТЕМ

У овом поглављу наведени су кључни аспекти и модули система и описани са теоријским осврtом.

2.1. Серверска страна

Сервер је сачињен од следећих ентитета:

- *Node.js* - за покретање сервера
- *MongoDB - за управљање базом података*
- *Express.js - за рутирање*
- *Контролера - за обраду захтева*



Слика 1. *Блок шема серверске стране*

Node.js

Node.js представља *JavaScript* платформу базирану на *Google V8 Engine*-у, која пружа једноставан и брз модел програмирања, а уз то подржава и асинхрони рад *web* апликације. Захтеве обрађује у само једном процесу, без потребе за креирањем додатне нити (енг. *Thread*) услед нових захтева. Захваљујући томе омогућен је бржи процес јер не постоји потреба за чекањем извршења захтева, већ апликација наставља даље са извршавањем наредне линије кода.

Уз *Node* долази и *Node Package Manager - npm*. У питању је алат који је задужен за добављање екстерних *Node.js* пакета, као и за складиштење и проналажење истих. Инсталација нових пакета обавља се помоћу *npm* команди командне линије (терминала), чиме се пакети аутоматски преузимају и додају у листу зависности (енг. *Dependency*) у датотеку *package* са *JSON* екстензијом. Пакети који су коришћени на *back-end*-у су:

- *Mongoose* - за рад са *MongoDB*
- *Dotenv* - за учитавање променљивих из *env* датотеке у процесу
- *Express* - за рутирање сервера и обраду *HTTP* захтева

MongoDB

Node поседује пакет (библиотеку) која омогућава приступање бази и извршавање различитих упита. Реч је о пакету који се назива *Mongoose* и пакету који поседује веома једноставан интерфејс за моделирање података објеката (енг. *Object Data Modeling - ODM*). Користи се као спрега између објеката у самом коду и њихову репрезентацију у бази. *MongoDB* представља нерелациону врсту базе података илити *NoSQL*, односно документациону базу, у оквиру које податке је могуће складиштити као *JSON* документе. Структура докумената може да варира, услед чега је смањена сложеност примене, а самим тим и бржи развој саме апликације.

Express.js

Express је *Node* пакет који представља уграђену библиотеку чије основно задужење јесте руковођење *HTTP* захтевима са клијентске стране. Такође, *Express* пружа комуникацију између сервера и базе података, где серверска страна тражи од модела извршавање операција од базе. Коришћене су *GET, POST, PUT* и *DELETE* методе.

Након што сервер пошаље захтев или добије исти, потребно је проследити одговор. Да би било лакше разазнати врсте захтева и упита, креирају се путање или руте (енг. *Routes*) које су креиране за одређену крајњу тачку (енг. *Endpoint*). Изворна (енг. *Root*) тачка у фази развоја јесте http://localhost:3000. Оваква путања могућа је само за машину у локалној мрежи, док би за остале уређаје у мрежи на рути уместо *localhost* кључне речи била прослеђена конкретна *IP* адреса уређаја на којем се сервер заправо извршава.

У продукцији би крајња тачка била назив *web* странице, која би била *host*-ована. Руте служе за прослеђивање захтева одговарајућим контролерима, који даље обављају сву потребну функционалност.

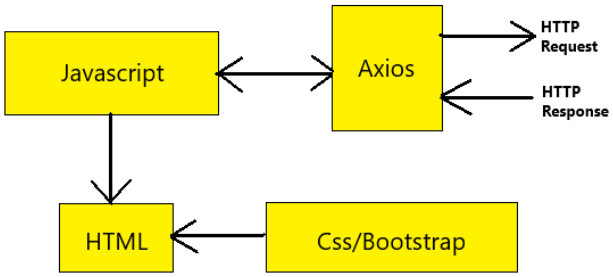
Kontroleri

Контролери функционишу тако што преко модела шаљу захтеве ка бази, креирају одређене упите у зависности од тога за шта су им потребни ти подаци и уколико испуњавају услове онда као одговор добијају жељене податке. Уколико би се узео пример додавања новог студента на *web* страници, први корак био би унос података (име, презиме и број индекса) на клијентској страни, након чега следи паковање унетих података у *HTTP* захтев ка рути на серверској страни http://localhost:3000/api/v1/students који сервер преузима и потом прослеђује одговарајућем контролеру.

2.2. Клијентска страна

За развој корисничког интерфејса коришћени су *HTML, CSS* и *Javascript*. Апликација је направљена тако да буде *single-page* што значи да ће се страница само једном учитати при иницијалном покретању а након тога се промене на страници динамички учитавају са сервера односно асинхроно без учитавања нове странице. Кључни ентитети на *front-end* делу су:

- *Javascript* – динамичко *понашање* садржаја *web* странице
- *HTML* – за креирање *web странице*
- *CSS* и *Bootstrap* – за изглед корисничког интерфејса
- *Axios* – за слање *HTTP захтева*



Слика 2. *Блок шема клијентске стране*

Javascript

Javascript је динамичан, слабо типизиран и интерпретиран програмски језик високог нивоа. Поред *HTML*-а и *CSS*-а, једна је од три водеће технологије за дефинисање садржаја на *web*-у; већина *web* сајтова користи *Javascript* а сви модерни *web* читачи га подржавају без потребе за инсталирањем додатака. Комбинован са *HTML*-ом и *CSS*-ом, *Javascript* чини *DHTML* (*Dynamic HTML*). Садржи *API* за рад са текстом, низовима, датумима и регуларним изразима, али не и улазно/излазне функционалности, као што су повезивање, складиштење података или графичке функционалности, за шта се ослања на окружење у коме се извршава.

Javascript се поред *web*-а користи у другим окружењима, као што су *PDF* документи, *web* читачи за специфичне *web* сајтове и десктоп виџети. Нове и знатно брже *Javascript* виртуелне машине и платформе засноване на њима, популаризовале су *Javascript* за израду *web* апликација на серверској страни. На клијентској страни, програмери најчешће имплементирају *Javascript* као интерпретирани језик, али све више новијих *web* читача обавља *just-in-time* компајлирање. *Javascript* се још користи и за развој видео игара, десктоп и мобилних апликација као и у мрежном програмирању на серверској страни са извршним окружењима као што је *Node.js*.

Javascript је најпопуларнији скриптни језик на Интернету којег подржавају сви познатији прегледачи (*Internet Explorer, Mozilla Firefox, Netscape, Opera, Safari*). Неке примене *JS*-а су:

- *Javascript* даје *HTML* дизајнерима алат за програмирање – *HTML аутори обично нису програмери, али Javascript је скрипти језик са веома једноставном синтаксом.*
- *Javascript може да динамички унесе код у HTML страну,*
- *Javascript може да реагује на догађаје – може да се подеси тако да се изврши кад се нешто деси, нпр. кад се страна учита или кад корисник кликне на HTML елемент,*
- *Javascript може да прочита или испише елементе – може да прочита и да промени садржај HTML елемената,*

- *Javascript може да се користи и за проверу исправности унетих података – може* да се користи за проверу исправности података унетих у форму, да провери исправност података пре него што се пошаљу серверу,
- *Javascript* може да се користи за детектовање *browser*-а корисника – у *зависности од browser-а, учитава се страна специјално дизајнирана за тај browser,*
- *Javascript може да се користи за креирање cookie*-ја – може да се користи за чување и враћање информација о рачунару посетиоца, итд.

HTML

HTML је описни језик специјално намењен опису *web* страница. Помоћу њега се једноставно могу одвојити елементи као што су наслови, параграфи, цитати и слично. Поред тога, у *HTML* стандард су уграђени елементи који детаљније описују сам документ као што су кратак опис документа, кључне речи, подаци о аутору и слично. Ови подаци су општепознати као мета подаци и јасно су одвојени од садржаја документа.

CSS

CSS је језик форматирања помоћу ког се дефинише изглед елемената *web* странице. Првобитно, *HTML* је служио да дефинише комплетни изглед, структуру и садржај *web* странице, али је од верзије 4.0 *HTML*-а уведен *CSS* који би дефинисао конкретан изглед, док је *HTML* остао у функцији дефинисања структуре и садржаја.

CSS синтакса се састоји од описа изгледа елемената у документу. Опис може да дефинише изглед више елемената, и више описа може да дефинише један елемент. На тај начин се описи слажу један преко другог да би дефинисали коначни изглед одређеног елемента (отуда назив *cascading* (енгл. *cascade* - цреп) да би се дочарало слагање једног стила преко другог у дефинисању коначног изгледа елемента).

Bootstrap

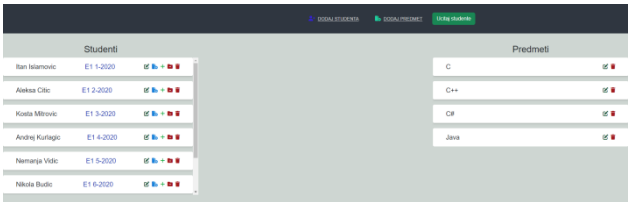
Bootstrap је *framework* који је задужен за естетику, тј. за изглед корисничког интерфејса. Представља колекцију *CSS* класа које су задужене за унапред дефинисан и стандардизован изглед *HTML* елемената. Поред лепог изгледа, пружа и додатне функционалности кроз уграђени *jQuery*, али једна од најбитнијих одлика јесте одзивност страница, односно скалиран и контролисан приказ елемената на различитим резолуцијама и величинама екрана.

3. ФУНКЦИОНАЛНОСТИ

У овом поглављу, урађен је осврт на неке од кључних функционалности *web* апликације.

Почетна страна

На почетној страници *web* апликације приказан је преглед свих студената и предмета. Наиме, приказани су основни подаци као што су име, презиме и број индекса студента за студенте и назив предмета за предмете. Кликом на име и презиме или број индекса студента отвара се прозор (модал) са детаљнијим информацијама о студенту: уколико нема предмета да слуша само су приказани име, презиме и број индекса студента а уколико има неки предмет да слуша онда је приказана школска година у којој слуша дати предмет као и освојени поени за одређене категорије за дати предмет. Кликом на назив предмета отвара се прозор (модал) са детаљнијим информацијама о предмету: уколико нема дефинисане категорије само је приказан назив предмета а уколико има дефинисане категорије онда су и оне приказане заједно са називом предмета. Такође, на овој страници обезбеђено је и додавање нових студената и предмета након чега је могуће њихово даље ажурирање, брисање, додељивање предмета студенту, додељивање категорија предмету као и додељивање поена студенту за категорије датог предмета.



Слика 3. *Изглед почетне странице*

Додавање новог студента

За додавање студента, мора се попунити поља за име, презиме и број индекса студента и тек тада се може додати. Уколико се то не уради од сервера се добија одговор о грешци да одговарајуће поље није попуњено или у случају броја индекса може добити и грешку да није добро унет формат за број индекса. Након додавања новог студента исти ће се појавити у колони студенти и над њим се онда могу радити следеће функционалности: ажурирање, брисање, додељивање предмета студенту, уклањање додељених предмета као и унос за дате предмете.

Додавање новог предмета

Уколико би корисник желео да дода нови предмет, мора попунити поље за назив предмета и уколико жели може попунити поље за категорије које ће дати предмет имати. Уколико поље за назив предмета остави празно, од сервера ће добити одговор о грешци да поље није попуњено. Након додавања новог предмета исти ће се појавити у колони предмети и над њим се онда могу радити функционалности ажурирања и брисања.

Ажурирање студента

Ажурирање информација о студенту постиже се притиском дугмета за ажурирање које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити потребне измене. Ажурирање студента подразумева измену података као што су име, презиме и број индекса. Приликом ажурирања било ког од датих поља важи исто правило као и код додавања новог студента, да сва поља морају бити попуњена. За ажурирање броја индекса важи још једно правило а то је да то поље мора бити написано у правилном формату, у супротном ће сервер вратити грешку о неправилном формату и неће ажурирати студента.

Ажурирање предмета

Уколико би корисник желео да ажурира неки предмет то може урадити тако што притисне дугме за ажурирање које стоји поред сваког предмета. Након што то

уради отвориће се прозор (модал) преко кога ће обавити потребне измене. Ажурирање предмета подразумева измену података као што су име предмета и категорије које ће дати предмет садржати. Приликом ажурирања поља за име предмета важи исто правило као и код додавања новог предмета, да то поље мора бити попуњено док поље за категорије може остати празно.

Брисање студента

Брисање студента ради се тако што се притисне дугме за брисање које стоји поред сваког студента. Након што то уради дати студент биће уклоњен из листе студената и листа ће бити ажурирана.

Брисање предмета

Уколико би корисник желео да обрише неки предмет то може урадити тако што притисне дугме за брисање које стоји поред сваког предмета. Након што то уради дати предмет биће уклоњен из листе предмета и листа ће бити ажурирана.

Додељивање предмета студенту у одређеној школској години

Додела предмета студенту извршава се тако што се притисне дугме за додељивање предмета студенту које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Додељивање предмета студенту подразумева следеће: студенту се мора доделити предмет као и школска година у којој би слушао дати предмет. Уколико се не попуни неко од поља, сервер ће вратити поруку о грешци да дато поље не сме бити празно. Додатно, уколико је попуњено поље за школску годину, сервер може вратити поруку о грешци да није добро унет формат за школску годину.

Уклањање додељеног предмета студенту

Уколико би корисник желео да уклони неки предмет који је додељен одређеном студенту то може урадити тако што притисне дугме за уклањање додељеног предмета које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Уклањање додељеног предмета студенту подразумева да се изабере неки предмет који би желели да уклонимо тако да га студент више не слуша.

Унос поена за датог студента на датом предмету у одговарајућим категоријама

Уколико би корисник желео да унесе поене за датог студента то може урадити тако што притисне дугме за унос поена које стоји поред сваког студента. Након што то уради отвориће се прозор (модал) преко кога ће обавити дату функционалност. Унос поена за датог студента подразумева следеће: прво се мора одабрати предмет за који би се желели унети поени, затим се могу унети поени за одређене категорије изабраног предмета.

Уколико се не одабере предмет, сервер ће вратити поруку о грешци да то поље не сме да буде празно. Поени нису обавезни да се унесу тј. у случају да се поље поени остави празно, то ће се протумачити као да је за сваку категорију датог предмета унето нула поена. У супротном се могу унети конкретни поени и потребно их је унети онолико колико има категорија изабраног предмета и ти поени треба да буду раздвојени зарезом.

Уношење више студената одједном путем JSON датотеке

Ова функционалност више служи као тест функционалност у случају да неко жели брже да тестира функционалност *web* сајта. Притиском на дугме учитај студенте, биће додато 8 студената уместо да се они ручно уносе.

4. ЗАКЉУЧАК

Може се закључити да је систем чији је циљ био примена клијент-сервер комуникације са практичном применом успешно реализован. Пројекат је укључио развијање *web* сајта са технологијама за развијање *single-page* апликације.

Предлози за побољшање:

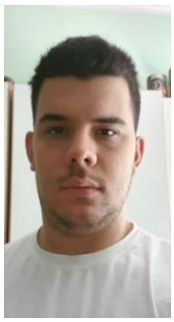
- Додавање опције за *претрагу студената и предмета у циљу њиховог лакшег налажења*
- *Додавање опције за сортирање студената и предмета*
- *Боља подршка за IOS кроз мобилну апликацију*
- *Прелазак са HTTP на HTTPS* трансфер протокол као стандардизовани гарант сигурности и безбедности интернет странице

5. ЛИТЕРАТУРА

[1] Shah, Dhruti Na. *Node .Js*. Bpb Publications, 2018.

[2] Chodorow, Kristina. *MongoDB*. O'Reilly, 2013.

Кратка биографија:



Игор Караџић рођен је Ужицу 1996. год. Дипломски рад на Факултету техничких наука из области Елктротехника и рачунарство – Примењено софтверско инжењерство одбранио је 2020. год. Област интересовања су му развој софтвера у програмским језицима *Java, Javascript* i *C#.*

контакт: igigotika@gmail.com