ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВА СТРУКОВНИХ СТУДИЈА

**Нинковић Милица**

**ИМПЛЕМЕНТАЦИЈА ВЕБ АПЛИКАЦИЈЕ ЗА УЧЕЊЕ ,,FRONT-END” ЈЕЗИКА  
  
- завршни рад -**



Београд, септембар 2017.

Кандидат: **Нинковић Милица**

Број индекса: **РТ-27/14**

Студијски програм: **Рачунарска техника**

Тема: **Имплементација веб апликације за учење ,,front-end” језика**

Основни задаци:

**1. Имплементација ,,Back-end”-a**

**2. Комуникација између клијента и сервера**

**3. Кориснички интерфејс**

Ментор:

Београд, септембар 2017.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

др Перица Штрбац, проф. ВИШЕР

Резиме:

У завршном раду описане су технологије, процес, идеје и принципи израде веб апликације за учење фронтенд (front-end) језика. У теоретском делу рада, дат је опис технологија које су коришћене приликом израде апликације, а то су ,,HTML”, ,,CSS”, ,,JavaScript”, ,,JQuery”, на фронтенд страни, као и ,,PHP” и ,,MySQL” на бекенд (back-end) страни и њихова примена у самом раду. У практичном делу, дата је израђена апликација.

**Кључне речи:** : фронтенд, HTML, CSS, JavaScript, учење, развијање апликација.

ABSTRACT:

The basic principles, requirements, processes and ideas for developing web application for learning front-end are described in this paper. Technology used to develop this application, such as “HTML”, “CSS”, “JavaScript”, “JQuery”, “PHP” and “MySQL”, and their implementation are described in theory section. On the other hand, in practice section, there is an example of developed application.

**Key words**: front-end, HTML, CSS, JavaScript, learning, application development.

*Посвећено:  
  
 Мајци:  
  
 За сву храброст, снагу ,   
труд и одрицања која је уложила,  
 да бих данас била где јесам.   
  
Оцу:  
  
Бол због његовог одсуства  
тек је маргинално ублажен   
мојим (несавршено кодираним)   
сећањима на њега.[[1]](#footnote-0)\**

САДРЖАЈ:

[1. УВОД 1](#_Toc21252)

[2. Технологије коришћене за израду задатка 2](#_Toc13233)

[2.1. ,,Front-end” 2](#_Toc11294)

[2.1.1. HTML, CSS, BOOTSTRAP 2](#_Toc9259)

[2.1.2. JavaScript, JQuery, AJAX 4](#_Toc17914)

[2.2. ,,Back-end” 6](#_Toc13528)

[2.2.1. MySQL и база података 6](#_Toc27327)

[2.2.2. PHP 8](#_Toc5171)

[3. Делови апликације 13](#_Toc23441)

[3.1. Кориснички 13](#_Toc2745)

[3.1.1. Страница: ,,Почетна” 13](#_Toc15398)

[3.1.2. Страница: ,,О пројекту” 15](#_Toc16161)

[3.1.3. Страница: ,,Пошаљи нам поруку” 16](#_Toc13463)

[3.1.4. Страница курса 18](#_Toc19571)

[3.1.5. Страница лекције 20](#_Toc469)

[3.2. Администраторски 22](#_Toc31163)

[3.2.1. Пријављивање и приступ админстраторском делу апликације 23](#_Toc23831)

[3.2.2. Изглед администраторског дела и панели 24](#_Toc22599)

[4. Закључак 32](#_Toc9116)

[5. Захвалност 33](#_Toc17627)

[6. ИНДЕКС ПОЈМОВА 35](#_Toc27026)

[7. ЛИТЕРАТУРА 37](#_Toc14625)

# УВОД

Како је интернет постао наша свакодневница и не можемо да замислимо живот без њега, тако је почела експанзија развоја алата за развој веб апликација, самих апликација, па и људи који желе да се баве израдом истих.

Веб девелопмент (web development) и веб програмирање, као индустријска грана, последњих година узима све више маха и све више људи се интересује за ову струку. Интересовање се јавља не само међу средњошколцима и студентима, већ и међу децом и старијима.

У процесу образовања, управо за ову област, већина сајтова и материјала на које сам наилазила и из којих сам учила били у на еглеском језику. Млађа деца и старији, који, као што је већ речено, развијају интересовања у овој сфери индустрије, често не могу да почну са учењем, управо због језичке препреке. Због тога, основна идеја овог рада јесте да се људима који нису усавршили или не знају енглески језик, омогуће материјали на матерњем (српском/југословенском) језику.

# Технологије коришћене за израду задатка

Како функционисање ове врсте вебсајта неопходан динамички садржај, у процесу израде морамо користити и фронтенд и бекенд технологије (front-end, back-end). Одлучила сам се за следеће технологије:

* HTML
* CSS
* Bootstrap
* JavaScript
* JQuery
* AJAX
* PHP
* MySQL

У наставку биће описано коришћење сваке од технологија у зависности од ,,стране” апликације.

## ,,Front-end”

Презентацијски, илити, фронтенд део, односи се на оно што корисник приликом посете вебсајту види. Овај садржај мора бити кориснички настројен (user friendly), оносно, интуитиван за корисника који посећује нашу страницу. Приликом дизајнирања сајта водила сам се основним принципима израде корисничког интерфејса и настојала да сајт буде што интуитивнији и лакши за коришћење.

Од горе напоменутих технологија за израду презентацијског дела користе се HTML, CSS, JavaScript, JQuery A jax и Bootstrap library.

### HTML, CSS, BOOTSTRAP

Основу сваке странице, њен ,такорећи, костур чине управо ове технологије. HTML је ту да створи конструкцију, односно, оквир или темељ за нашу странциу који ће се касније надоградити CSS-ом. Он користи различите тагове (tag) за различите елементе на страници, на пример, за наслове користимо <h1> - <h6> тагове, док за ликове користимо <a> таг. Сваки таг има своје опште и посебне атрибуте који га описују и који су ту да би прецизирали начин рада сваког тага. Општи атрибути су типа класа и идентификатор (class, id). Морамо напоменути да класа нема исто значење као код објектно оријентисаних програмских језика, већ представља неку скупину обележја карактеристичних за таг или тагове којима је додељена. Више различитих елемената-тагова може имати исту класу.

Са друге стране, идентификатор је јединствен за сваки елемент и једнозначно га одређује. На страници не смемо имати више елемената са истим идентификатором јер може довести до колизије. Сам парсер на претраживачу неће ово препознати као грешку, али при првом позиву идентификатора, на пример у некој јаваскрипт (JavaScript) функцији, доћи ће до неслагања.

Посебни атрибути су атрибути који су карактеристични само за неколицину тагова. На пример, атрибут ,,method” карактеристичан је за форме и он нам говори којим методом ћемо слати податке на обраду.

CSS, каскадни стил је ту да нашој горе поменутој основи да финији изглед. Појам ,,каскадни” у његовом називу односни се на принцип који важи приликом додељивања обележја неком елементу. Према томе такозвано, дете елемент, као и у стварном животу, наслеђује особине свог родитељског елемента. Међутим, за разлику од стварности, где ми наше наслеђене особине не можемо променити, у CSS-у је могуће то урадити уз помоћ селектора, псеудо-класа и псеудо-елемената.

Селектори су горе поменути идентификатор и класа, као и име самог тага, док псеудо-класе чине одређена својства елемента. Та својства се односе на позицију елемента у односу на целу страницу, или свој родитељски елемент, као и на то како корисник поступа са њим, да ли је у фокусу, да ли је кликнут и слично. Свакој псеудо-класи претходи знак ,, : “.

Псеудо-елемент је сличан псеудо-класи, чак је и обележавање слично, почиње са ,, :: “, а користимо их када желимо да изменимо неку специфичност, као што су прво слово или први ред у некој реченици или чак да убацимо садржај пре или после елемента над којим смо позвали псеудо-елемент.

Када желимо да стилизујемо неки елемент прво морамо навести његов селектор, односно, путању до тог елемента у документу. Након тога у витичастим заградама ( {} ) наводимо својства која ће елемент имати. На пример, уколико желимо да елемент има оквир црне боје, ширине 1px и има пуну линију, па ћемо то навести као:   
,, *identifikacija\_elementa{ border: 1px solid black;}” .*

Bootstrap представља библиотеку готових HTML структура и CSS обележја, као и JavaScript функција. Творци ове библиотеке у многоме су олакшали посао свим програмерима тиме што су најчешће структуре и обележја формулисали и готове елементе и дали им имена која се често користе за њих. Поред тога, развили су свој, решеткасти (grid) систем који се користи за распоређивање елемената на страни и фонт иконице.

Још јенда сјајна ствар код ове библиотеке је та да је сав садржај респонзиван (responsive), oдносно, прилагођава се уређају који корисник користи за прегледање сајта, тако да на програмеру остаје само да се бави финесама како би добио жељени изглед сајта.

За коришћење елемената које нам ова библиотека нуди, неопходно је да је укључимо у наш пројекат. То можемо урадити на више начина.

Први, можемо да скинемо пакет који садржи фајлове у којима се налази Bootstrap код, са њиховог званичног сајта, а затим их укључити у наш пројекат.

Други начин је коришћењем такозваних ,,CDN” (Си-Ди-Ен) сервиса, који на својим серверима држе сав код који бисмо ми иначе држали у нашем пројекту. Једино што треба да урадимо јесте да у пројекту наведемо путању до сервиса и тиме смо укључили фајлове. Када користимо овај начин укључивања библиотека, морамо водити рачуна о томе да код укључимо са више извора, у случају да неки од сервера ,,падне”, тада нећемо имати приступ датотекама библиотеке. Због тога се овај начин укључивања не препоручује.

### JavaScript, JQuery, AJAX

### Иако се ове технологије често сврставају у презентацијски део прављења апликације, можемо слободно рећи да су оне средишњи део, оносно, спона између функционалног и презентацијског слоја једне апликације. Подједнако их користимо да бисмо од статичког дела неке странице направили динамички, као и за комуницирање са функционалностима које нам бекенд омогућује.

Јаваскрипт представља програмски језик у правом смислу те речи, за разлику од до сада поменутих технологија које се више односе на кодовање него на програмирање. Сам по себи, јаваскрипт је слабо типизиран језик, што значи да не морамо да дефинишемо тип променљиве која се користи већ ће она добити тип приликом доделе вредности.

Креирамо их писањем резервисане речи ,,var” (скраћено од ,,variable”), па затим име променљиве. Вредност променљивој се не мора доделити одмах, то се може урадити и касније у току писања кода. Као што је напоменуто, овај језик је слабо типизиран, што значи да једна променљива може прво бити једног типа, а касније јој се може доделити неки други тип без икаквог превођења већ једноставном доделом вредности.

Променљиве могу бити локалне и глобалне. Као и код сваког језика, локалне променљиве су видљиве само унутар одређеног простора где је она креирана, док је глобална видљива на нивоу целог фајла.

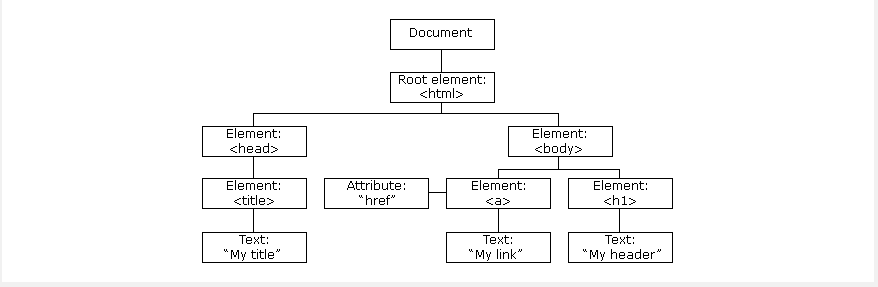
Поред променљивих, наравно, постоје и функције. За писање фунцкија, пре самог имена морамо користити резервисану реч ,,function”, а затим доделити назив функцији и написати њену дефиницију. Унутар њих можемо позивати друге функције, како оне које су програмски уграђене тако и наше.

Разлог због ког се овај програмски језик сврстава у презентацијске, илити, фронтенд језике је тај што се користи за манипулисање ДОМ-ом (DOM document-object model). ДОМ представља стандард на вебу који служи за организацију приступа документу, и језички је неутралан, односно, не зависи од технологије. У овом контексту мисли се на HTML ДОМ, односно, структуру једне веб странице и дефинише:

* HTML елементе као објекте
* Својства HTML елемената
* Методе за приступ HTML елементима
* Догађаје за HTML елементе

Другим речима, HTML ДОМ представља стандард за добављање, измену, додавање или брисање HTML елемената.

За приступ овим елементима користи се ,,document” објекат овог стандарда који је ,,власник” свих елемената на нашој страници. Уз помоћ њега можемо доћи до елемената преко тага, идентификатора, класе или атрибута за име, а затим им мењати неко својство, без обзира да ли је то изглед, садржај, приказивање или понашање елемента.



*Слика 1-ДОМ стабло објеката*

У структуру наше странице, јаваскрипт можемо укључити пре него што је садржај учитан или након тога. Лично, сматрам да је други начин бољи јер тако страница неће пријавити грешку уколико неки део кода захтева елемент који још није учитан. Фајл са писаним кодом за јаваскрипт укључујемо тако што ћемо у атрибуту ,,src” <script> тага навести његову путању у нашем пројектном директоријуму.

Како што Bootstap представља библиотеку за HTML и CSS, тако и JQuery представља библиотеку јаваскрипта. Код ове, као и код сваке библиотеке, неопходно је укључити је, и то опет можемо урадити на горе поменуте начине.

Ова библиотека представља скуп функција писаних у јаваскрипту које олакшавају писање кода већ готовим решењима. За приступ елементима не мора се користити објекат ,,document” већ се користи директно тај елемент навођењем његовог селектора. Као и код његовог претходника, сврха је за динамичко мењање садржаја презентацијске стране у било ком од четири контекста.

Постоји и такозвана минификована верзија коју можемо укључити у документ. Ова верзија је очишћена од непотребних размака, и код је форматиран тако да заузима што мање места, а самим тим апликација је бржа, тако да преферирам њено коришћење.

Оно што је важно напоменути је да без укључивања JQuery библиотеке неће радити све функције Bootstrap библиотеке.

Ајакс (AJAX) је, како кажу, сан сваког програмера, јер може да примени измене на страници без њеног освежавања. Његово име представља акроним од ,,Aynchronous Javascript And Xml”. Управо та асинхроност му омогућава да засебно комуницира са сервером, ондосно, бекендом, добавља информације неопходне за страницу и врати их на презентацијску страну без њеног освежавања. Под термином ,,без освежавања” мисли се без освежавања комплетне странице, већ се само један њен део освежава.

Када шаљемо захтев ка серверу можемо то радити гађањем одређене крајње тачке, односно у стручној терминологији ендпоинта (endpoint). Приликом слања захтева наводимо неколико параметара, ја ћу навести оне које сам најчећше користила:

* Тип захтева
* Начин синхронизације: синхроно/асинхроно
* Крајња тачка, односно, место обраде захтева
* Подаци неопходни за обраду

По завршетку навођења параметара наводимо шта се дешава након што захтев врати одговор са сервера. Одговор може бити успешан и неуспешан, а у зависности од тога предузимамо следеће кораке.

## ,,Back-end”

Термин бекенд (back-end) се користи за функционални део апликације, односно, део који корисник не види. У њему се дефинише база података коју апликација користи, комуникација са истом, права приступа и сигурност апликације.

У овом раду, за имплементацију функционалног дела коршћене су технологије MySQL за израду базе и PHP за читаву функционалност.

### MySQL и база података

Ова технологија је најраспрострањенија у потреби приликом израде базе података. Најчешће се комбинује управо са PHP-ом за имплементацију веб апликација, уз помоћ.

База података може бити релациона и нерелациона. Код релационих база, организација података се заснива на релационом моделу. Овај модел представља скуп релација између којих се дефинишу неке везе. Свака релација мора да има дефинисан примарни кључ који служи за једиствено одређивање н-торки. Поред тога, једна релација може да има и страни/спољни кључ који означава његову везу са неком другом релацијом.

Примарни кључ једне релације није нужно један податак, некада то може бити комбинација два податка. Тако, на пример, када желимо да повежемо неког радника са неким одељењем у фирми то ћемо урадити тако што наведемо примарни кључ фирме и примарни кључ одељења. Уколико наведемо само један од ова два податка нећемо имати потпуну информацију о томе где радник ради.

Конкретно, у MySQL-у, релације су представљене табелама, а н-торке редовима у табели. Свака табела има свој примарни кључ којим се јединствено идентификује један ред. У једном реду се налазе подаци који, односно, атрибути, који су дефинисани над истим доменом и садрже неке вредности, које су ту да би детаљно описале ту релацију.

Током пројектовања базе података, након дефинисања релација и атрибута, морамо извршити нормализацију. Нормализација представља процес који оптимизује базу и уклања могуће аномалије. Правила нормализације дефинисана су у такозваним ,,правилима нормалне форме”.

Код већине база, довољно је ићи до Треће нормалне форме да би табела била нормализована, односно, оптимизована. У ретким случајевима се иде и до виших степена нормализације, као што је Бојс-Кодова нормална форма, која се још назива и Нормална форма 3.5, јер има мало прецизнији ниво нормализације, па чак и до Пете и Шесте нормалне форме.

Као што сам већ напоменула, Трећа нормална форма је сасвим довољна за оптимизацију и нормализацију табеле, па ћу се посветити опису нормализације до трећег степена.

#### Прва нормална форма

У Првој нормалној форми је битно да атрибуте разбијемо на атомарне вредности. Тако, на пример, уколико смо имали поље у табели које се зове ,,Име и презиме”, разбићемо га на две одвојене вредности, посебно за име а посебно за презиме. Некада ова декомпозиција може довести до прављења нове релације, како бисмо растеретили стару. Тада, нову и стару релацију везујемо уз помоћ спољног кључа.

#### Друга нормална форма

Да би релација била у Другој нормалној форми, мора бити у Првој и када атрибут, који није примарни кључ, зависи од целог примарног кључа а не само од једног његовог дела. Слично горе поменутом примеру за примарни кључ код радника у одређеној фирми, у другој нормалној форми се бира кључ или комбинација кључева која ће једнозначно одредити један ред, а избацују се непотребни делови кључа који не представљају битну информацију за једнозначно одређивање једне н-торке.

#### Трећа нормална форма

Када кажемо да је табела у Трећој нормалној форми, то значи да је у Првој и у Другој и да ниједна колона без кључа не зависи од друге колоне без кључа. На конкретном примеру то значи да уколико имамо име и презиме неког радника они не смеју да зависе један од другог, односно, одређено име не сме да указује на одређено презиме, једино кључ сме да функционише на тај начин.

#### Приступ пољима табеле

Да би наша апликација фунцкионисала требају нам подаци из табела у бази података. Како се база налази на серверу, ми те податке немамо код себе, па их морамо тражити. За ово се користе упити у бази података који врше не само добављање података него и манипулацију над њима. Упитима можемо добавити, изменити, уписати и обрисати податке или релације.

Сваки упит треба има обавезне и необавезне делове. Под обавезним деловима се сматрају они делови без којих упит не може да функционише, а то су:

* Тип акције коју упит извршава (Select, Insert, Update, Delete,Create…)
* Поља која желимо да дохватимо, изменимо или креирамо у упитима селекције и измене и стварања
* Табела над којом се упит извршава
* Параметри за унос у упитима измене, креирања и уписивања

У необавезне делове спадају услови, који су ту да нам конкретизују захтеве за податке, на пример, да ли особа има одређено име, презиме, да ли ради у одређеном одељењу неке фирме итд..

Сваки упит враћа ниједан, један или више редова, уколико су упити селекције, број редова на које је упит деловао, уколико су упити измене или потврду о успешности креирања и брисања табеле.

### PHP

PHP је скрипт језик намењен за израду динамичког и функционалног дела апликације и извршава се на серверу. Захвалност за велику популарност коју има дугује програмском језику Цe (C), из ког је изведена синтакса. Поред класичног процедуралног програмирања подржава и концепте објектно оријентисаног програмирања као што су наслеђивање, апстрактне класе и интерфејси. Последња верзија овог језика је изашла у децембру 2015. године и то је верзија 7.0 .

#### Променљиве, типови података, операције и функције

Као и код јаваскрипта не захтева се навођење типа променљиве код њене декларације, осим ако је у питању низ, већ се тип добија након доделе вредности. Пре имена променљиве мора се ставити долар знак ($) а затим се додељује име променљивој, и у том формату се променљива позива сваки пут. Константе се дефнишу коришћењем функције define() у којој наводимо парове кључа и вредности. Кључ представља име константе и према конвенцији програмирања пише се великим словима под наводним знацима. Приликом позивања константи не морамо стављати доларски знак нити наводнике већ је једноставно позивамо навођењем имена.

#### Типови података

С обзиром на то да је изведен из језика Цe типови података које има су слични. То су стрингови, бројеви, бројеви са покретним зарезом, низови, објекти, логички (boolean) као и ресурси. Ресурси нису типови података у правом смислу те речи већ су референце ка функцијама и ресурсима изван PHP-a, као што је база података.

Како је већина података и правила везана за њих наслеђена из друигх језика, нећу се превише бавити описивањем сваког типа, изабрала мени најзанимљивији тип, низове

Декларација низова у PHP-у се врши уз помоћ фунцике array() која прима ниједан, један или више параметара. Када не проследимо ниједан параметар, то значи да је низ празан и да смо промељиву само декларисали као низ. Са друге стране, прослеђени параметри чине елементе низа.

Као и у другим језицима, низови су индексирани. Сјајна ствар коју PHP нуди је да приликом доделе вредности низу не морамо навести индекс, већ их он сам смешта један иза другог. Наравно, уколико желимо, можемо поставити на одређени индекс неку вредност, па чак и када тај индекс није следећи на реду. У сваком случају, морамо користити угласте заграде ( [ ] ) приликом додељивања елемента, у супротом, нова вредност ће пребрисати цео низ и постати обична променљива.

Други вид индексирања низова јесте асоцијативно индексирање, индексирање по кључу. Овај начин индексирања низа садржи парове кључ и вредност, где кључ представља индекс за вредност. Приликом спаривања кључа и вредности користи се синтаксa *,,$mojNiz=array(‘ime’=>’Moje ime’, ‘prezime’=>’Moje prezime’); “.*

Мултидимензионални низови, или матрице су још једна могућност коју PHP нуди. У терминологији се користи још и израз ,,низ низова” и првенствено се користи за складиштење некe JSON структуре података.

#### Операције

Операције, као и у сваком језику, могу бити логичке, математичке и операције спајања. Поред тога, постоји подела на унарне и бинарне. Унарне операције се врше само над једним елементом а то су:

* Инкрементација (повећање вредности за 1)
* Декрементација (смањење вредности за 1)
* Негација

Бинарне су оне операције које захтевају најмање два елемента за њихово извршавање, а ту спадају:

Сабирање

Одузимање

Дељење

Целобројно дељење (модуо)

Степеновање

Операција доделе

Конкатенација

Операције поређења:

* Једнакост
* Идентичка једнакост
* Неједнакост
* Логичка компарација
* Однос (веће, мање, веће-једнако, мање-једнако)

#### Функције

Функције су у свим језицима исте и имају исти принцип. Представљају рутину која се извршава изнова при сваком позиву. Приликом дефинисања функције наводимо параметре које прима, као променљиве, затим у телу функције код који се извршава и повратне вредности. Функција може да добије ниједан, један или више параметара, такође повратна вредност није обавезна, све зависи од тога шта се у функцији обрађује.

#### Функције у класама

Према принципима објектног програмирања функције у класама се називају методе и могу се само позвати на објектима те класе. Свака класа има свој конструктор и деструктор. Како им сама имена говоре користе се за стварање и уништавање објекта те класе. Конструктор и деструктор спадају у посебну врсту метода које се називају магичне методе. Најбитније међу њима су гетери и сетери (getters, setters), функције за испис објекта у стрингу, функција која проверава да ли је постављена нека вредност. Магичне методе карактерише специфична сингакса, односно, именовање. Испред свог имена имају двоструку доњу црту, ( \_\_ ) а затим име (consturct, destruct, set, get, toString).

Методе могу бити јавне, приватне и заштићене, као и код свих објектно оријентисаних језика и важе иста права приступа. Постоје и статичке методе којима се може приступити без прављења објекта класе, већ само преко имена класе. За приступ статичкој методи користе се две двотачке ( :: ).

#### Комуникација PHP-a са базом

У документацији PHP-a ресурси су наведени као један од типова података, иако они суштински нису тип података. Један од ресурса јесте и база података из које наша страница добија податке за приказ и имплементацију.

Да бисмо комуницирали са базом података, неопходно је да направимо класу конекције и конфигурациони фајл. У конфигурационом фајлу ћемо навести параметре који нам требају за конекцију са базом : име хоста, име корисника, лозинка за приступ и име базе у којој тражимо податке.

У класи конекције уз помоћ уграђених функција које PHP поседује правимо конекцију са базом коришћењем . Конекцију треба сместити у атрибут класе, а само прављење конеције се врши у конструктору класе ком се не прослеђују параметри, већ се укључују из конфигурационог фајла. Уграђена функција која се конектује са базом је *mysqli\_connect(),* којој се прослеђују име хоста, корисника, лозинка и име базе, респективно. Из функције се враћа веза са базом, уколико је конекција била успешна, или порука о грешци. Приликом конектовања јако је битно навести да уколико се конекција не изврши успешно програм ,,убије” проце и избаци поруку о грешци, да не би дошло до компромитовања базе. Након тога поставља се кодни распоред у бази коришћењем *mysqli\_set\_charset()* функције којој прослеђујемо атрибут са конекцијом и тип кодног распореда, за наше говорно и писано подручје UTF8.

Читав сет *mysqli* функција постоји за комуницирање са базом. Осим њега, постоји и стара верзија *mysql* која се више не користи, док се за упите користи *query()* функција која као параметар прима стринг упит.

Следећи код представља начин на који сам конектовала базу са PHP фајлом коришћењем Синглтон (Singleton) обрасца и приказ класе за конекцију. Разлог због којег сам се одлучила за Синглтон је да не бих оптерећивала и успоравала апликацију константним отварањем конекција ка бази података.

*include\_once 'config.php';  
  
class Connection  
{  
  
 private $con;   
 private static $point;  
  
 private function \_\_construct()  
 {  
 $host= DB\_HOST;  
 $user= DB\_USER;  
 $pass= DB\_PASS;  
 $name= DB\_NAME;  
  
  
 $this->con= @mysqli\_connect($host,$user,$pass,$name)  
 OR  
 die('Database error'.mysqli\_connect\_error());  
  
 mysqli\_set\_charset($this->con, 'utf8');  
 }  
  
 public static function pointerConnection(){  
  
 if(self::$point==null){  
 self::$point= new Connection();  
 }  
  
 return self::$point;  
 }  
  
 public function getConnection(){  
  
 if($this->con==null){  
 self::pointerConnection();  
 }  
 return $this->con;  
 }  
  
 public static function queryRequest($query){  
 return Connection::pointerConnection()->getConnection()->query($query);  
 }  
  
 public static function emptyQueryResults($result){  
 if($result->num\_rows===0)  
 return 0;  
 else  
 return 1;  
 }  
 public function \_\_destruct()  
 {  
 @mysqli\_close($this->con) OR die('Disconnect problem');  
 }  
  
 public static function redirectUser($page){  
  
 $url='http://'.$\_SERVER['HTTP\_HOST'].dirname($\_SERVER['PHP\_SELF']);  
 /\*Dodeljuje se ime hosta(domaćina) a zatim postavlja u tekući direktorijum, a to je direktorijum iz kog se poziva funkcija\*/  
  
 $url=trim($url,'/\\'); /\*uklanja kose crtne na kraju\*/  
  
 $url.='/'.$page; /\*dodaje ime stranice na koju preusmeravamo\*/  
  
 header('Location:'.$url);/\*Preusmeravanje\*/  
 exit(); /\*Prekida izvršavanje skripta\*/  
  
 }  
}*

# Делови апликације

Апликација се састоји из два дела: Корисничког и Администраторског. Кориснички део може да види свако ко дође на веб страницу, док је администраторски део резервисан само за регистроване кориснике. Оба дела користе исту идеју дизајна, али им се садржај разликује. На корисничкој страни се налазе прегледи курсева и лекција, као и контакт страна и једна страна са информацијама. Са друге стране, администраторски део садржи контролну таблу за управљање информацијама везаним за приказ на корисничкој страни.

## Кориснички

Како је већ наведено, кориснички део видљив је свима који приступе страници. На почетној страни налази се навигациони мени, у самом врху, који садржи странице сајта којима корисник приступа. На средини, налази се лого са именом сајта, а одмах испод њега мени са курсевима које корисник може да похађа. У самом дну, налази се подножје (footer) у ком се налазе подаци о аутору. Читав изглед и назив инспирисан је старим ,,осмобитним” играма које смо сви играли када су се појавиле прве конзоле.



*Слика 2- почетна страна апликације*

Поред почетне стране, у навигационом менију постоји и страница која садржи информације о пројекту и страница на којој корисник може да пошаље поруке администраторима сајта.

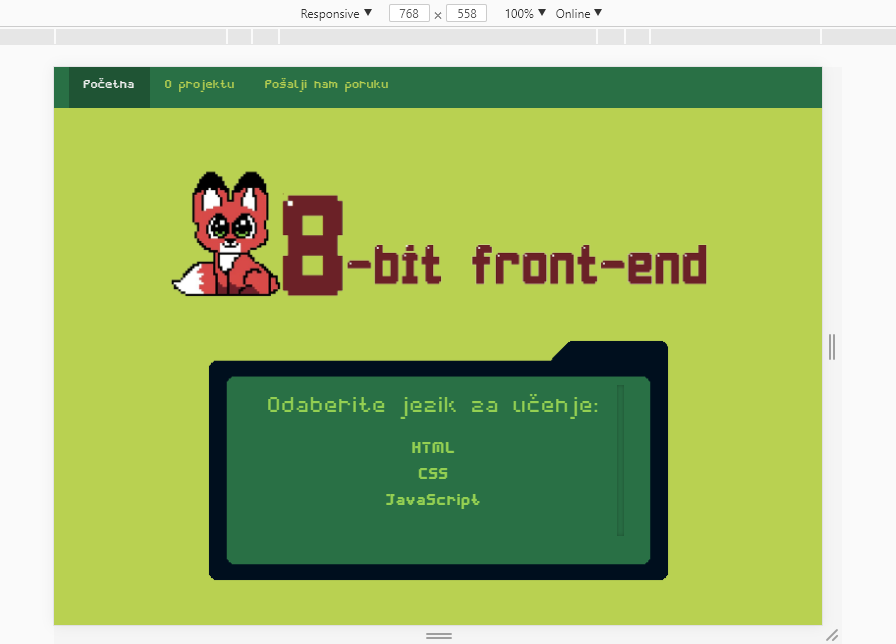
У менију који садржи курсеве, приказују се они курсеви који су тренутно доступни кориснику и које може да похађа. Одабиром било ког курса, одлази се на страницу која приказује његов опис и понуђене лекције.

### Страница: ,,Почетна”

Како је опис изгледа странице дат у претходном пасусу, овде ћу се бавити искључиво технологијом која је имплементирана у процесу израде странице.

За израду фронтенда коришћен је ,,Бутстреп” библиотека (Bootstrap library). Одлучила сам се за коришћење ове технологије јер омогућава нека готова решења која се могу имплементирати у свакој страници. Готова решења која су имплементирана на овој, а и на свакој страни, су решеткасти/мрежни систем и навигациони мени.

Мрежни систем се састоји из редова и колона, тако да практично приликом развоја апликације правимо једну табелу у коју ређамо елементе. Како је ширина елемента затата у процентима, она се потпуно прилагођава екрану, тако да је изглед респонзиван.



*Слика 3- Приказ почетне стране за резолуцију 768х558px*

Остали елементи на овој страни рађени су коришћењем чистог HTML и CSS кода уз малу помоћ класа из бутстреп библиотеке.

Оно што се не може видети јесте функционални PHP и средишњи јаваскрипт део. Улогу коју јаваскрипт у на овој страници има је да распоређује податке које добије са сервера. Тако добијене податке о активним курсевима смешта у мени. Следеће линије текста приказују позив ове функције у јаваскрипту и њену комуникацију са базом.

Јаваскрипт:

*$('document').ready(  
 courses = getAll('get-courses'),  
 listAllCourses(courses),  
 dataLessons = getAll('get-all-lessons')  
);*

Обрада ајакс захтева:

*case 'get-courses':{  
 $courses=Cours::getAllCourses();  
 echo (json\_encode( $courses));  
 break;  
}*

PHP метода:

*public static function getAllCourses(){  
 $qry="SELECT \* FROM kursevi WHERE 1";  
  
 $results=Connection::queryRequest($qry);  
 $response=array();  
 if(Connection::emptyQueryResults($results)){  
  
 while($row=$results->fetch\_assoc()){  
 $response[]=$row;  
 }*

*}  
 return ($response);  
 }*

### Страница: ,,О пројекту”

Ова страница је информативног типа и на њој не постоји никаква посебна имплементација функционалног и средњег дела апликације. За изглед је коршћена jumbatron класа из Бутстреп библиотеке.

*#courseDescriptionJumbatron, #courseLessonsJumbatron, #aboutJumbatron, #messageJumbatron{  
 min-height: 31.25rem;  
 height: auto;  
 margin: 3.125rem 0;  
 border-radius: 0.625rem;  
 background-color: #e0f098;  
 border-top: 0.25rem solid #a5bb48;  
 border-bottom: 0.25rem solid #a5bb48;  
 border-right: 0.125rem solid #a5bb48;  
  
}*

Преласком на са почетне, у навигационом менију се користити класа active из горе поменуте библиотеке која стилизује изглед поља.

*.navbar-default .navbar-nav>.active>a, .navbar-default .navbar-nav>.active>a:focus, .navbar-default .navbar-nav>.active>a:hover {  
 color: #f2f2f2;  
 background-color: #1f5434;  
}*



*Слика 4 - изглед странице ,,О пројекту"*

### Страница: ,,Пошаљи нам поруку”

Ова страница представља имплементацију једног од задатака овог рада, комуникацију клијента и сервера. Принцип функционисања комуникације је да корисник путем ове контакт форме пошаље своје податке и поруку за администраторе апликације. Порука се чува у бази на серверу и касније се излистава на администраторском панелу у секцији за приказ порука. Администратор, када види поруку, може са панела да одговори кориснику и тада му директно шаље меј на адресу коју је оставио у форми.



*Слика 5-форма за слање порука*

Форма мора да садржи CAPTCHA код који врши превенцију слања такозваних спам порука и самим тим претрпавање и блокирање базе. Имплементација кода за ову функционалност у току израде практичног дела преузета је oд Друа Филипса (Drew Phillips) и није намењена за комерцијалне сврхе. За проверу CAPTCHA кода користи се серверска страна.

Валидација ове форме има два дела, клијентски и серверски. На клијентском делу се проверавају валидности поља, да ли су сва попуњена и да ли садрже валидне податке. На серверском, као што је већ речено, проверава се унос CAPTCHA кода. Уколико је код регуларан, подаци се прослеђују даље на бекенд где се порука уписује у базу.

Серверска страна:

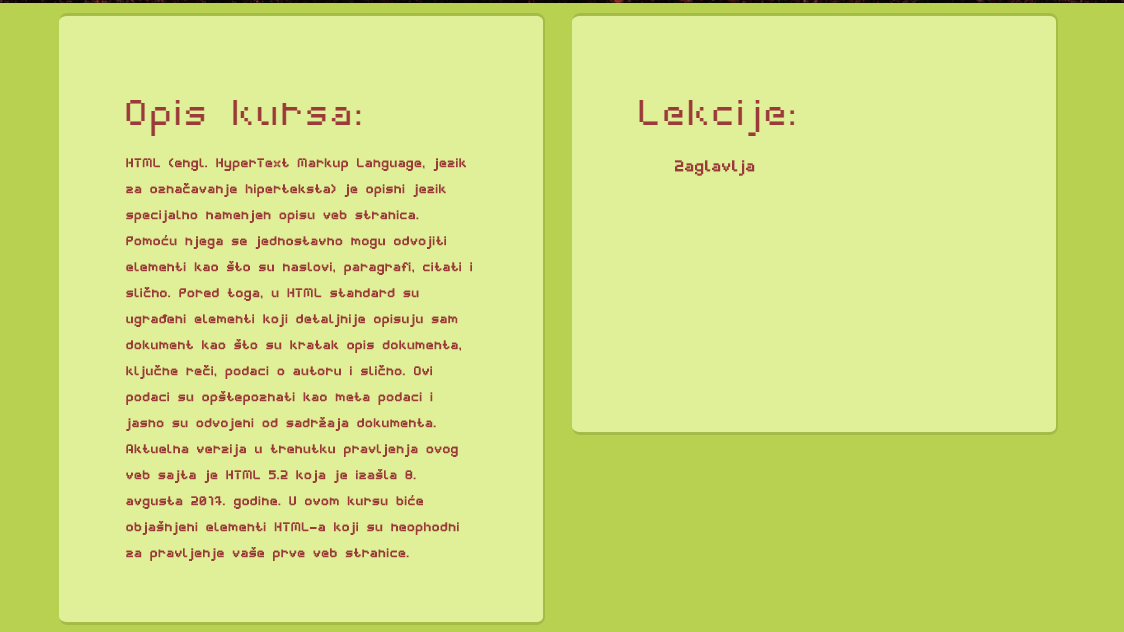
*public static function sendMessage($subject,$sender,$senderFirstName,$senderLastName,$message){  
   
 $qry="INSERT INTO posta(naslov , posiljalac , posiljalac\_ime, posiljalac\_prezime, procitana , poruka , odgovoreno)  
 VALUES ('".$subject."','".$sender."','".$senderFirstName."','".$senderLastName."',0,'".$message."',0)";  
  
 $responseStatus=Connection::queryRequest($qry);  
 if($responseStatus){  
 return 'Uspešno ste poslali poruku';  
 }  
 else{  
 return 'Došlo je do greške, pokušajte kasnije';  
 }  
  
}*

Обрада захтева:

*$firstName=$\_POST['name'];  
$lastName=$\_POST['surname'];  
$email=$\_POST['email'];  
$subject=$\_POST['subject'];  
$message=$\_POST['message'];  
$code=$\_POST['captcha\_code'];  
  
  
$image = new Securimage();  
if ($image->check($\_POST['captcha\_code']) == true) {  
  
 if(!empty($firstName)&& !empty($lastName) && !empty($email) && !empty($subject) && !empty($message)){  
  
 $result=Messages::sendMessage($subject,$email,$firstName,$lastName,$message);  
 if($result){  
 echo 1;  
 }  
 else{  
 echo 0;  
 }  
 }  
  
  
} else {  
 echo 2;  
}*

### Страница курса

До странице курса долази се одабиром и кликом на курс из менија. Страница садржи опис курса, који се добија из базе и списак активних лекција.



*Слика 6 -Страница курса*

И овде, као и на страници са информацијама о пројекту коршићен је стил jumbatron класе и класе за мрежни систем. Принцип попуњавања података је исти. Ајакс шаље захтев ка бекенду са идентификатором курса, који се добија из $\_GET захтева, чека одговор, а затим попуњава елементе са добијеним подацима.

Исписивање описа курса:

*<?php  
$id = trim($\_GET['id']);  
$results = Cours::getOneCourse($id);  
if (is\_array($results)) {  
 echo $results['opis'];  
} else {  
 echo $results;  
}  
  
  
?>*

Исписивање активних лекција курса:

*<?php  
$results = Lesson::getLessonsOfCours($id);  
  
if (is\_array($results) && !empty($results)) {  
 echo ' <ul class="lead" id="courseLessons">';  
 foreach ($results as $r) {  
 if ($r['status'] == 1) {  
 echo '<li class="listLesson"><a href="lessonPage.php?id=' . $r['id'] . '">' . $r['naziv'] . '</a></li>';  
 }  
 }  
 echo '</ul>';  
} else if(empty($results)) {  
 echo "<h3 class='display-3'>Trenutno ne postoji nijedna lekcija za ovaj kurs</h3>";  
}  
  
?>*

### Страница лекције

До странице лекције се долази одабиром и кликом на име лекције са списка на страници курса. Када се отвори страница, приказује се опис лекције и два поља. Прво лево поље је текстуално и у њему се приказује код из базе који је администратор унео приликом креирања лекције. У дну овог поља налази се дугме за извршавање промењеног кода.

Са десне стране се налази елемент <iframe> који је, у суштини, прозор ка некој другој страници. У њему је приказан ефекат унетог кода.



*Слика 7-Приказ унетог кода и његовог излаза на страници лекције*

Добијање описа лекције

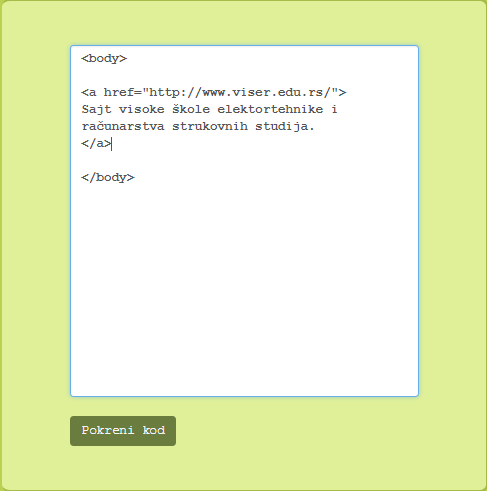
*<?php  
  
 $id=$\_GET['id'];  
 $result=Lesson::getOneLesson($id);  
  
if(is\_array($result)){  
  
echo $result[0]['opis'];  
  
?>*

Добијање кода лекције:

*<textarea id="lessonCode" class="form-control" rows="20" style="resize: none"><?php  
 echo $result[0]['primer\_koda'];  
 }  
 ?>  
</textarea>  
<br>  
<button type="button" class="btn" id="startCode"onclick="setIframe()" value="Pokreni kod">Pokreni  
 kod</button>*

Имплементирање кода у <iframe> таг:

*function setIframe(){  
  
 var code=document.getElementById('lessonCode').value;  
 var iframe=document.getElementById('iframeCode');  
 document.getElementById('iframeCode').src = "data:text/html;charset=utf-8," +(code);  
  
}  
  
$('document').ready(  
 setIframe()  
);*



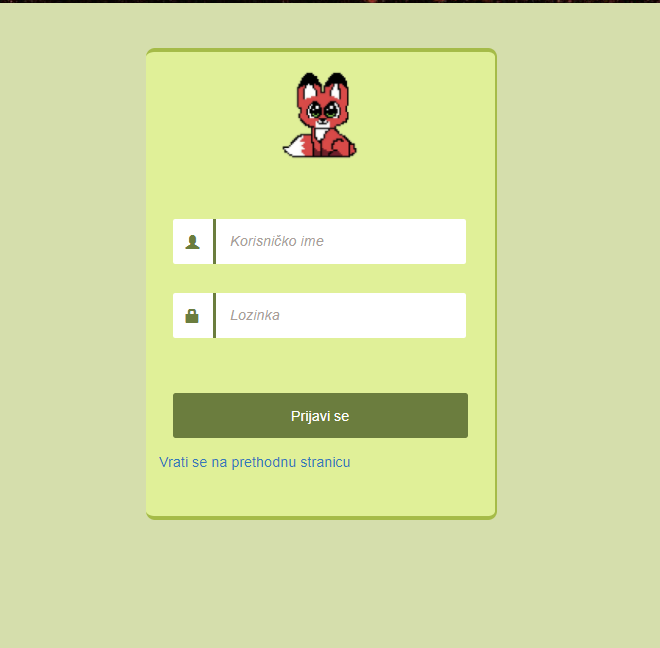
*Слика 8-Измењен код лекције*



*Слика 9- Излаз унетог кода након клика на дугме,,покрени код"*

## Администраторски

Администраторском делу апликације може да приступи само регистрован корисник. Обичан корисник, посетилац сајта, не може доћи до форме за логовање сурфовањем кроз сајт. Тек када се унесе путања у URL-у претраживача може се приступити форми за логовање.



*Слика 10-Форма за логовање и приступ администраторском делу сајта*

### Пријављивање и приступ админстраторском делу апликације

Сав садржај странице коју обичан корисник види се генерише из администраторског дела апликације. Овом делу приступ имају администратори и модератори, а да би дошли до њега прво морају да унесу креденцијале у форму.

Креденцијали који се уносе су корисничко име и лозинка, који се затим валидирају. Валидација се врши двострано, клијентски и серверски. Клијентска валидација проверава да ли су сва поља попуњена, ако јесу, процес валидације се наставља на серверској страни.

Први упит који се шаље бази података је добављање хеш вредности лозинке која одговара корисничком имену које је унето. Ако се врати празан резултат, не постоји корисник са тим креденцијалима и поставља се порука о грешци. У случају да је хеш лозинке нађен за унето корисничко име, приступа се поређењу две хеш вредности. Ово радимо уз помоћ функције *password\_verify()* која за параметре узима сринг унете лозинке и хеш вредност лозинке из базе, респективно. Излаз из ове функције је логички (boolean), и даје нам информацију о успешности поређења.

Метода за проверу једнакости лозинки:

*private function checkPasswordMatch($id,$password){  
  
  
 $qry="SELECT pwd\_hash FROM korisnici WHERE id=$id";  
 $result=Connection::queryRequest($qry);  
  
 if($result){  
 $result=$result->fetch\_assoc();  
 if(password\_verify($password,$result['pwd\_hash'])){  
 return 1;  
 }  
 }else{  
 return 0;  
 }  
}*

Метода за пријављивање корисника

*public static function login($username,$password){  
   
 $qry="SELECT \* FROM korisnici WHERE korisnicko\_ime='".$username."'";  
  
 $results = Connection::queryRequest($qry);  
 $error='';  
  
 if(Connection::emptyQueryResults($results)){  
 $results=$results->fetch\_assoc();  
  
 if(password\_verify($password,$results['pwd\_hash'])){  
 return $results;  
 }  
 else{  
 return $error;  
 }  
 }  
 return $error;  
  
 }*

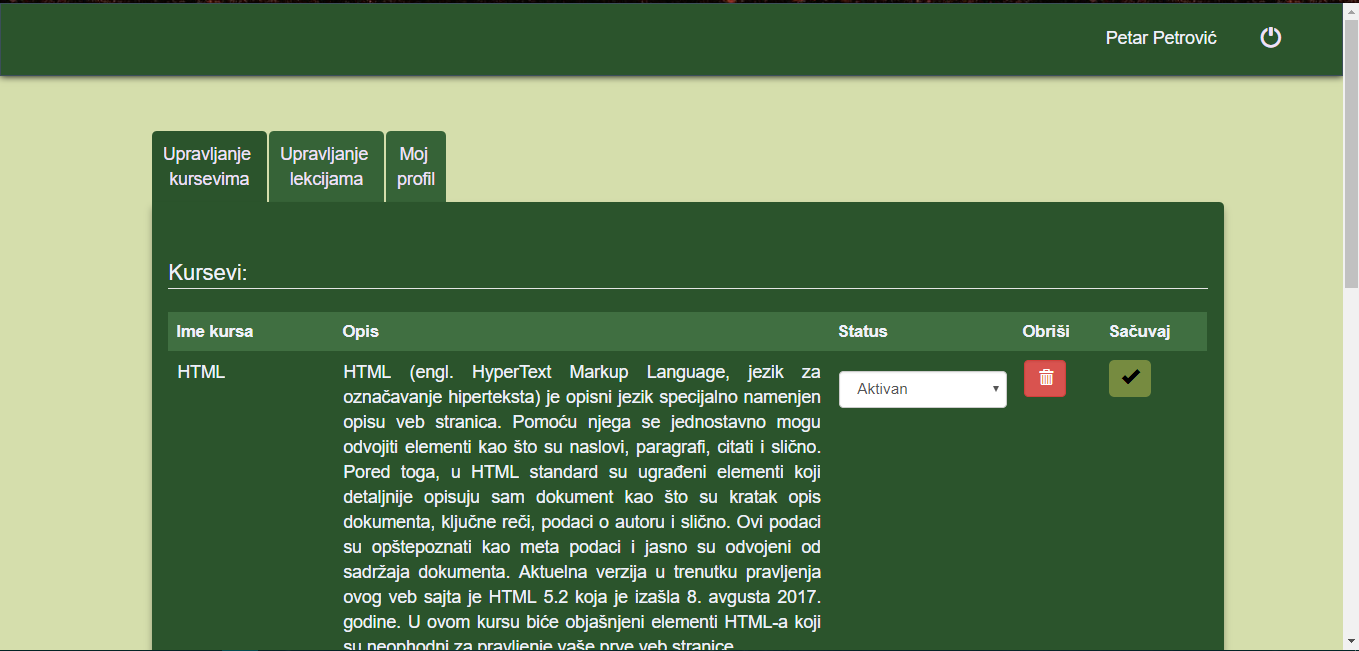
Након успешног логовања постављају се сесије са подацима о улогованом кориснику и прелази се на контролни део апликације.

### Изглед администраторског дела и панели

Након што се корисник улогује из страница из сесије добиај информације о томе ко је улогован и која је његова улога. На основу тога детерминише се приступ различитим секцијама и изменама. На пример, модератори могу само да управљају курсевима и лекцијама, као и својим информацијама, док администратори имају пун приступ апликацији и могу да управљају корисницима и да одговарају на поруке које су пристигле од посетилаца сајта.



*Слика 11- Изглед контролног панела за администраторе*



*Слика 12- Изглед контролног панела за модераторе*

#### Картица са курсевима

На овој картици приказани су сви курсеви који постоје у бази података без обзира на то да ли су активни или не. Подаци су организовани у табелу која приказује име курса, опис курса, његов статус и дугмад за манипулацију. Курс се може обрисати кликом на дугме са кантицом, а кликом на дугме сачувај чувају се промене статуса курса. Подаци за упис у табелу се, наравно, добијају из базе коришћењем ајакса. Када се страница учита, шаље се захтев бази података, а затим се добијени резултати шаљу у јаваскрипт функцију за попуњавање података на страници.

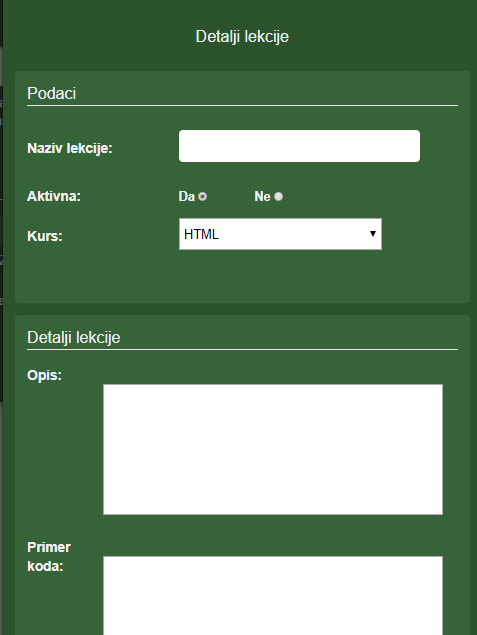
Код за попуњавање администраторског дела апликације:

*$('document').ready(  
 getRoles(),  
 dataUsers = getAll('get-all-users'),  
 fillWithUsers(dataUsers),  
 courses = getAll('get-courses'),  
 fillWithCurses(courses),  
 dataLessons = getAll('get-all-lessons'),  
 fillWithLessons(dataLessons),  
 getMyData(),  
 messages=getAll('get-all-messages'),  
 fillWithMessages(messages)*

У дну картице, испод табеле налази се дугме чијим кликом се отвара форма за додавање новог курса. У форми се налазе поља за име курса и за његов опис, као и радио дугмад и дугме за чување унетих података. Након клика на дугме, шаље се ајајкс захтев за обраду података из форме и након одговора се део странице са приказаним курсевима освежава.

#### Картица са лекцијама

Дизајн ове картице је исти као и дизајн претходне. На почетку се налази табела са подацима о лекцији и дугмадима за њеноу модификацију, као и дугметом за детаље о лекцији. Испод табеле, налази се дугме чијим се кликом отвара модал за додавање нове лекције. Исти модал коришћен је и за измену детаља о лекцији.



*Слика 13-Модал за додавање нове лекције*

#### Картица за управљање корисницима

Још једна картица која има исти дизајн, а то су табела са подацима и дугме за додавање новог корисника. Кликом на дугме отвара се модал за прављење новог корисника.

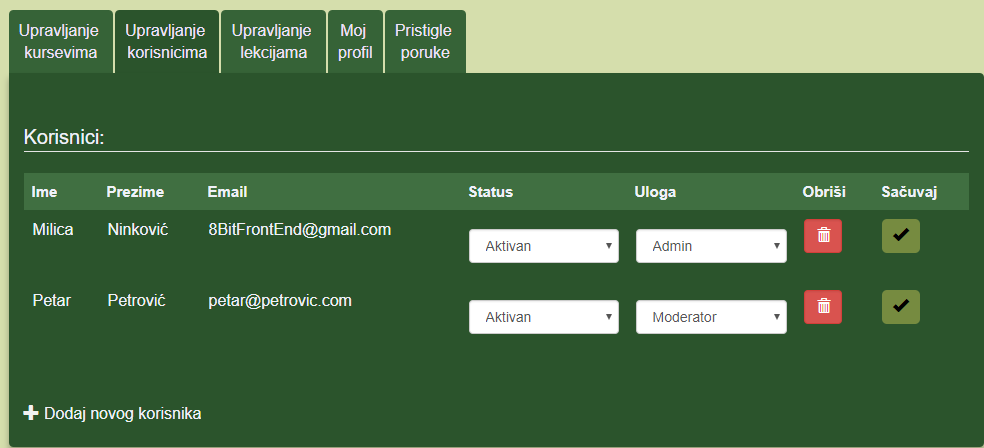
Привилегију управљања корисницима имају само администратори. Провера да ли је корисник администратор врши се преко података из сесије. У сесији су, као што је речено, смештени подаци о кориснику након логовања. Једно од поља у табели корисници чува податке о улози корисника. На основу тога испитујемо права приступа.

Детерминација права приступа корисника:

<?php  
if(isset($\_SESSION['role']) && $\_SESSION['role']==1){  
echo '  
 <li class="cardMenu" href="#" data-panel="manageUsers" onclick="*switchCards*(this)">  
 <p class="navText">  
 Upravljanje  
 <br>  
 korisnicima  
 </p>  
</li>';  
}  
?>

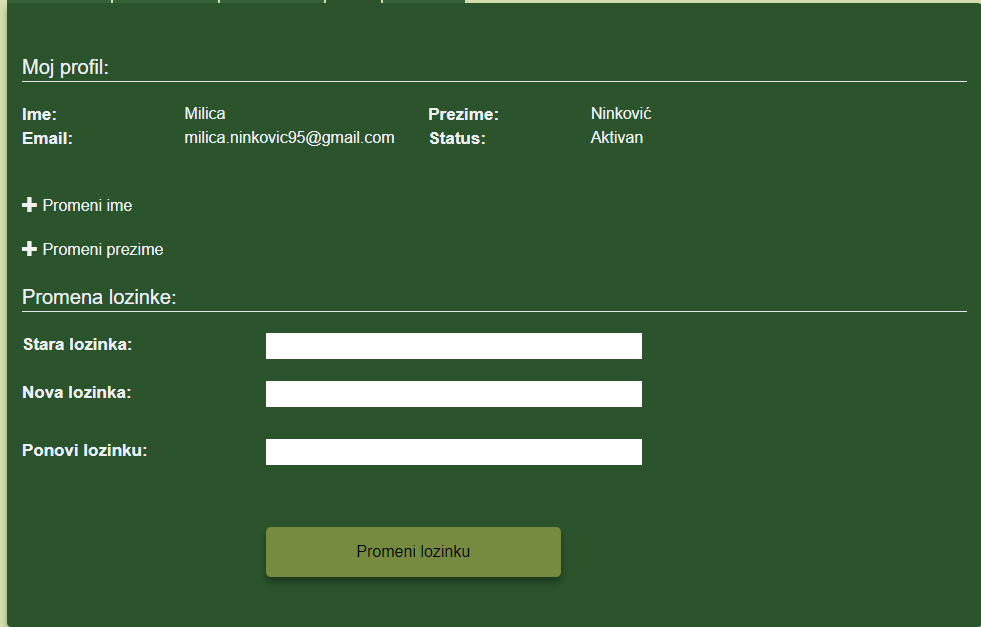
У овом кратком делу кода приказано је како се решава проблем ауторизације корисника. Ово исто смо могли урадити и путем јаваскритпа. Тако што бисмо прво сместити податак о роли у неку глобалну променљиву, а затим испитати њену вредност. Уколико вредност одговара оној која је дефинисана за администратора, приказаћемо картицу мењањем својства CSS-a, односно, сакрити уколико вредност не одговара администраторској.

Проблем који се јавља код овог вида ауторизације је то што се мењањем CSS-a не мења структура елемента и свако ко би отворио ,,Developer Tools “ (Ctrl+Shift+i) прозор у свом прегледачу могао ово својство да измени и да види шта се на сакривеној картици налази.



*Слика 14-Изглед картице за управљање корисницима*

#### Картица мој профил



*Слика 15-Изглед картице мој профил*

Ова картица садржи информације о тренутно улогованом кориснику. На њој корисник може да промени информације о себи као што су име и презима, а може и да промени лозинку. Све промене покривене су ајакс захтевима који након извршења освежавају део странице са изменом. Име, презиме и лозинка се не могу поставити на празне вредности, тај део валидације проверен је са клијентске стране.

Измена лозинке је нешто комплекснија од других измена на страници. Њу покрива и код за проверу критеријума које лозинка мора да задовољи да би била сигурна. То радимо на клијентској страни уз помоћ јаваскрипта на догађај притиска дугмета.

HTML код једног поља за унос лозинке:

*<div class="col-md-5">  
 <input type="password" class="whiteInput important passOrg" name="superuserSettings\_new\_password"  
 onkeyup="checkPwd(event, 1)">  
 <div class="originalPasswordError"></div>  
 <p class="capsWarning"></p>  
</div>*

Провера критеријума лозинке:

*function checkCriteria(str) {  
 var responseArray = [];  
 if (str.length < 8) {  
 responseArray.push(errorMessages.passTooShort, false);  
 } else if (str.length > 20) {  
 responseArray.push(errorMessages.passTooLong, false);  
 } else if (str.search(/\d/) == -1) {  
 responseArray.push(errorMessages.passNoNum, false);  
 } else if (str.search(/[a-zA-Z]/) == -1) {  
 responseArray.push(errorMessages.passNoChar, false);  
 } else if (str.search(/[^a-zA-Z0-9\!\@\#\$\%\&\\*\\_]/) != -1) {  
 responseArray.push(errorMessages.passBadChar, false);  
 }  
 else {  
 responseArray.push(successMessages.passAlright, true);  
 }  
 return responseArray;  
}*

Провера једнакости лозинки:

*function checkMatch(original, repeated) {  
  
 var responseArray = [];  
 if (repeated === original) {  
 responseArray.push(successMessages.matchingPassword, true);  
 }  
 else {  
 responseArray.push(errorMessages.passNotMatch, false);*

*}  
 return responseArray;  
}*

Ове две фунцкије су обједињене у функцији *checkPwd(event, 1)* која се позива на пољу у ком уносимо лозинку, а њени параметри су догађај и број који идентификује поље у форми над којим се догађај позива, да ли је оригинална или поновљена лозинка.

#### Картица пристигле поруке

На овој картици налази се приказ поштанског сандучета са порукама које су пристигле од стране посетилаца веб сајта. Када се отвори, свака порука има приказ о детаљима када је послата, ко је послао поруку, са које адресе, њен наслов и садржај.

Поред овога постоји могућност одговора на поруку у окиру модала где је порука приказана. Такође, постоји и преглед сви послатих порука са серверске стране.

Следећи пример кода приказује како се обрађује слање поруке клијенту:

Aјакс захтев за слање поруке:

*$("body").on("click","#sendResponse",function(){  
  
 var thisButton=this;  
 var data={  
 to:document.getElementById('mailDisplay').innerText,  
 subject:document.getElementById('messageSubject').innerText,  
 message:$("textarea[name=responseMessage]").val(),  
 id:thisButton.getAttribute('data-id')  
 };  
 $.AJAX({  
 type:"POST",  
 async:false,  
 url:"testfile.php?action=send-message",  
 data:data,  
 success:function(response){  
 $("textarea[name=responseMessage]").val('');  
 }  
 })  
  
  
});*

Обрада захтева за слање поруке:

*case 'send-message':{  
 $to=trim($\_POST['to']);  
 $subject=trim($\_POST['subject']);  
 $message=trim($\_POST['message']);  
 $id=trim($\_POST['id']);  
 $writeResult=ResponseMessages::writeMessage($id,$to,0,1,$message);  
 $result=ResponseMessages::sendMessage($to,$subject,$message,$id);  
 echo 'Poruka poslata;  
 break;  
}*

Статичке функције које се позивају приликом слања поруке:

public static function writeMessage(*$idReceived*, *$receiver*, *$sentStatus*, *$finnished*, *$message*)  
{  
 $qry = "INSERT INTO moje\_poruke( id\_primljene,primalac, poslata, dovrsena, poruka)  
 VALUES (*$*idReceived,'" . *$receiver* . "',*$*sentStatus,*$*finnished,'" . *$message* . "')";  
  
 $result = Connection::queryRequest($qry);  
  
 if ($result) {  
 return true;  
 } else {  
 return false;  
 }  
}  
  
public static function sendMessage(*$to*, *$subject*, *$message*, *$id*)  
{  
  
 $header = "From:8BitFrontEnd@gmail.com \r\n";  
 $header .= "MIME-Version: 1.0\r\n";  
 $header .= "Content-type: text/html\r\n";  
  
 $retval = *mail*(*$to*, *$subject*, *$message*, $header);  
  
 $qry = "UPDATE moje\_poruke SET poslata=1, dovrsena=1 WHERE id\_primljene=*$*id";  
 if ($retval == true) {  
 $result = Connection::queryRequest($qry);  
  
 if ($result) {  
 return true;  
 } else {  
 return false;  
 }  
  
 } else {  
 $result = ResponseMessages::makeDraftMessage(*$id*);  
  
 return $result;  
 }  
}

# Закључак

Развој веб апликација представља комплексан, континуалан и креативан процес, који захтева доста стрпљења и посвећености. Такође, захтева познавање различитих технологија, а у овом раду обрађен је један пример такве апликације.

## Технологије на клијентској страни

Технологије коришћене на клијентској страни:

* HTML 5
* CSS 3
* Bootstrap 3
* JavaScript
* JQuery
* AJAX
* ЈSON

HTML, CSS, Bootstrap коришћени су за реализацију изгледа сајта, односно, корисничког интерфејса. Делови странице, као што су заглавље и подножје, који су исти за сваку страницу издвојени су у посебне датотеке које се касније укључују по потреби. Такође, модуларни прозори и сандуче за пристиглу електронску пошту су издвојени у посебној датотеци (за сваки), док су JSON објекти коришћени за складиштење података добијених са серверске стране и користе се привремено, без чувања у самој структури пројекта.

## Технологије на Серверској страни

Технологије коришћене на серверској страни:

* PHP
* MySQL

Ове две технологије, иако засебне, на функционалном делу апликације раде заједно. PHP датотеке класа направљене су тако да свака одговара једној табели у бази, а свака метода одговара једном упиту за базу. Поред класа за табеле, постоји и класа за конекцију са базом података, конфигурациона датотека за њу и .sql датотека која се убацује на сервер и у којој је сачувано стање базе података. Такође, сви AJAX захтеви се обрађују на истој страници, сем захтева за слање порука од стране корисника апликације, пријављивања и одјављивања из контролног дела апликације.

База се састоји од шест табела, од којих је пет ентитетских и једна је, такозвана, пивот табела која регулише релације типа више на више, у којој се налази веза између назива роле и њеног примарног кључа. Свака табела има асоцијативно име, као и сваки њен атрибут.

## Побољшања

Како је израда ове апликације била у академске и демонстративне сврхе, она представља само основу која би се могла надоградити и побољшати. Побољшање се захтева, пре свега, на нивоу сигурности апликације и њених података, затим на квалитету материјала и детаљима лекција.

Што се проширења функционалности тиче, апликација би се могла надоградити да обухвата курсеве не само за веб програмирање већ и друга програмирања која су заступљена.

# Захвалност

На крају излагања желела бих да се захвалим породици која ме је подржала у току студирања и на великим одрицањима која је због тога морала да направи. Својим рођацима : Драгославу, Тањи, Мартини, Игору и Аци у чијем дому сам боравила за време студирања. Пријатељима и колегама: Лазару, Филипу, Кубури, Ненаду, Павлици, Јошки, Ракићу, Миши, Маринковићу, Горану, Жилету, Који и Декију, за то што су ме прихватили у овом ,,мушком” свету, за дивне успомене на студентске дане. Катарини, најбољем пријатељу и највећој подршци. Невени, која је сво моје теоретско знање о кваровима на рачунарском хардверу и софтверу употпунила праксом. И на крају, професорима и асистентима на Високој школи електротехнике и рачунарства који су предано радили свој посао у преношењу знања студентима: др Перици Штрпцу, др Немањи Мачеку, др Милану Мијалковићу, мр Марку Царићу, Бориславу Крнети и Банету Павићу.

# ИНДЕКС ПОЈМОВА

# ЛИТЕРАТУРА

[1] David Flanagan, ,,JavaScript sveobuhvatni vodič”, Beograd: Mikro knjiga, 2008.

[2] Larry Ulman, “PHP i MySQL za dinamičke veb sajtove”, četvrto izdanje, Beograd: CET,2012.

[3] PHP manual: http://php.net/

[4] W3 schools: https://www.w3schools.com/

[5] Stack Overflow: <https://stackoverflow.com/>

[6] Wikipedia: https://en.wikipedia.org/wiki/Database\_normalization

1. \* Парафразирано: Влатко Ведрал, ,,Декодирање стварности: универзум као квантна информација” у посвети својој мајци. [↑](#footnote-ref-0)