Факултет инжењерских наука
Универзитета у Крагујевцу

Назив студиског програма: Индустријски инжењеринг - ПИС

Ниво студија: Мастер академске студије

Предмет: Менаџмент комуникација

Број индекса: 324/2019

Александар Гламочек

ВИРТУАЛНА ПРОДАВНИЦА

Мастер рад

Комисија за преглед и одбрану:

1. Проф. др Фатима Живић - ментор *Датум одбране:*
2. *Оцена:*
3.

Крагујевац, 2021.

У оквиру овог мастер рада кандидат треба да:

Креира виртуалну продавницу. Опише све фазе развоја и постављања виртуалне продавнице, као и све утицајне аспекте онлајн продаје.

Препоручена литература:

[1] Јеремић, Б., Тодоровић, П., Мачужић, И., Ђапан, М., Милошевић, М., Раденковић, М., Вучковић, Ђ. (2013), *Основи Lean производње*, Факултет инжењерских наука, Универзитет у Крагујевцу, Крагујевац.

[2] https://neilpatel.com/blog//how-to-start-an-online-store/

Крагујевац, 2021

Ментор:

_____

Проф. др Фатима Живић

Резиме: *Виртуална продавница је веб сајт који има одређене производе и услуге са јасно приказаном ценом и као главни циљ има продају путем интернета. Било да поседујете малопродајну радњу или не, виртуална продавница може да функционише и не зависи од тога да ли имате малопродају у неком локалу. Готово све интернет продавнице се заснивају на једном истом принципу и 4 фазе. Онлине шоп је метод куповине путем интернета, што значи да корисник очекује испоруку на кућну адресу. Да би се ово омогућило потребно је да продавница има 4 основна фазе.*

1) *Додавање у корпу*
2) *Одабир плаћања и плаћање картицама*
3) *Организација испоруке*
4) *Достава*

***Кључне речи**: Виртуална продавница, корпа, достава*

***Abstract:** A virtual store is a website that has certain products and services with a clearly displayed price and its main goal is to sell online. Whether you own a retail store or not, a virtual store can work and does not depend on whether you have a retail in a local. Almost all online stores are based on the same principle and 4 phases. Online shop is a method of shopping online, which means that the user expects delivery to a home address. In order to enable this, it is necessary for the store to have 4 basic phases.*

***Kay words**: Virtual store, basket, delivery*

С А Д Р Ж А Ј

1. УВОД

Са напредовањем технологија повећава се употреба интернета и он постаје један од главних видова комуникације, али и система продаје. Он-лине продаја је све заступљенија у свету, тако да је у Америци постала и најпопуларнији начин куповине и самим тим је потиснула многе традиционалне продавнице. У Србији до пре неколико година није био популаризован овај начин продаје, јер је мали број људи имао храбрости да се одлучи за куповину преко интернета, али из године у годину расте број сајтова за куповину и код нас. Главна предност овог начина куповине је могућност куповине од куће, без одласка до продавница које су километрима удаљене, а то значи да се слободно време намењено куповини може другачије искористити.

Он-лине куповина захтева сигурност и поузданост од стране продаваца, али и одговорност од стране купаца да ће поручену пошиљку преузети. Тако да приликом куповине морамо прочитати услове и потврдити сагласност са унетим личним подацима да бисмо гарантовали њихову исправност. Сајт попут „Купујем Продајем“ захтева уношење података са личне карте и у случају неког прекршаја или преваре можете одговарати.

Живот без мобилних телефона постао је незамислив, тако да данас ретко коју особу можемо срести да не поседује или она или неко од укућана бар један мобилни уређај или таблет. Због ових података Веб апликације су се временом прилагођавале уређајима, тако да сада на својим телефонима можемо приступити готово свим Веб апликацијама и оне ће бити прилагођене мобилном уређају или таблету односно њиховим екранима. Систем куповине преко интернета самим тим је омогућен свима, односно особама које поседују бар неки од уређаја преко којих се могу повезати на интернет.

У основи креиране Веб апликације за продају цвећа налази се љубав према цвећу која је испољена кроз ненаменску апликацију за продају. Апликација која је прављена у раду је презентационог типа, односно није прављена наменски за неку фирму. Веб продавница за продају цвећа из рада је илустративна и не подржава све опције он-лине куповине али постоји могућност за надоградњу и усавршавање.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Он-лине продаја

Он-лине продаја је вид електронске куповине која омогућава купцу да преко интернета одабере и наручи производ који жели да купи и да му производ буде достављен на кућну адресу. Половином 80-их година се први пут спомиње појам интернет куповине и интернет пословања. Први предмет који је продат преко интернета био је CD Stinga„ Ten Summoner's Tales “, који је продала група студената Ten Summoner's Tales из Америке. Након тога настао је први Веб сајт за онлине продају ког је направио Пјер Омидар 1996., „АуцтионВеб“ и он је на свој сајт поставио покварени ласер који је желео да прода, чувени Пјеров ласер тада је постигао цену од 14.83$ и први он-лине сајт за продају је заживео.

Пјер је свој сајт преименовао у „Ецхо Бау“ познатији као „еБау“ и временом постао милионер, популаризацијом и развојем интернет продаје. Годинама се трговина преко интернета усавршавала, да би већ данас постала једна од водећих начина трговине. Као зачетак електронског пословања у Србији узима се 29. јун 1993. године када је основана Југословенска асоцијација за електронску размену података (Yugoslav Association for Electronic Data Interchange – YUEDI) са задатком да популаризује примену ЕДИ система у тадашњој Југославији. До појаве првих рачунарских добављача Интернет услуга, односно Интернет провајдера долази 1995. године.У првој фази електронске трговине у Србији имплементиран је само процес наручивања робе преко Интернета, док се плаћање за робу обавља поузећем уз физичку испоруку робе.

До масовније појаве продавница у СРЈ долази у јулу 1998. када Еунет у сарадњи са Фирмом “YUGate” отвара први електронски виртуелни трговински центар. До појаве већег броја функционалних е-продавница долази са појавом електронских картица које омогућавају онлајн плаћање, што представља другу фазу у развоју е-трговине у Србији. 1999. оснива се Е-Банк.цо.уу први и до данас једини Интернет паумент провајдер, намењен процесирању платних картица. Као трећа фаза развоја српске е-трговине узима се период од 2002. године до данас. Из године у годину расте број сајтова у Србији који се баве овим видом продаје.

Не постоји ни један већи и популарнији бренд који нема свој Веб схоп. Такође се популаризују и мобилне апликације као вид куповине и људи се полако навикавају на трговину преко интернета. Свака он-лине куповина захтева одређене кораке које корисник мора испунити да би поручио жељени производ. Првенствено се мора фокусирати на избор плаћања, начин испоруке, жељени производ врсту и сигурност у избору врсте производа за доставу.

Неке он-лине продавнице дозвољавају и замену робе уз плаћање трошкова транспорта које сноси купац. Саветује се увек провера података пре поручивања, због тога већина он-лине продавница при сваком кораку нуди опцију „Да ли сте сигурни“ да би се корисник вратио на претходни корак ако није сагласан са наруџбином или ако је дошло до грешке при избору производа.

***Слика 1**. Илустрација он-лине куповине* [1]

Куповина путем интернета еволуира из дана у дан. Све више фирми жели да прошири своју продају и све су веће потребе за програмерима Веб сајтова.

Постоје различите врсте он-лине продаје, а међу најпознатијим су:

- B2C (Business-to-Consumer)
- B2B (Business-to-Business)
- C2C (Consumer-to-Consumer)

B2C вид продаје подразумева да компаније продају робу крајњим потрошачима преко софтвера за продају аутоматски без потребе за додатном интеракцијом између људи. Најпознатији сајт који функционише по овом принципу је Амазон.

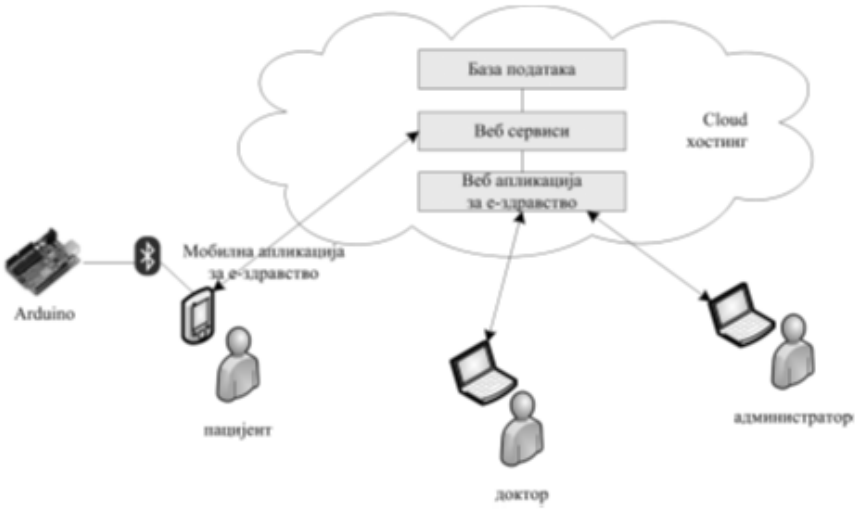
B2B продаја се односи на продају између компанија, када примера ради велике компаније продају робу продавцима на мало.

C2C је врста куповине где купац продаје купцу. Најверодостојнији примера оваквог начина он-лине продаје је eBay.

[1] https://www.web-ideas.com.au/online-shopping-system

2.2. Веб апликације

Веб апликације су настале развојем Веб сајтова који у својој основи представљају скупове веб страница које су смештене на одређени Веб сервер. Веб апликација представља софтверски производ који се извршава на Веб серверу, а приказује кориснику помоћу веб читача. Да би Веб апликација функционисала потребно је да се реализује у програмском језику који подржава Веб сервер као софтверски „engine“ (односно хостинг Веб апликације коју технологију подржава), да би он могао дату апликацију да преведе и изврши. веб апликација омогућава корисницима лакши приступ интернету и интерактивност између Интернета и корисника. Десктоп апликације морају бити инсталиране на клијент рачунару да би се уопште могле користити. Код Веб апликација то није случај јер се оне налазе на једном месту (серверу или групи сервера). Овим се минимизују многи проблеми, пре свега проблеми са дистрибуцијом, надоградњом апликација и генерално у случајевима када код десктоп 6 апликација корисник мора да преузме и инсталира/ажурира апликацију. Битна ствар је да код Веб апликација корисник никада не добија саму апликацију, већ само интерфејс ка њој што је у суштини све што му и треба.



***Слика 2**. Функционисање типичне Веб апликације*[2]

Принцип по коме функционишу Веб апликације је прилично једноставан. Корисников Веб претраживач (читач) шаље захтев Веб серверу, који податке прослеђује Веб апликацији, она их обради, а резултат враћа серверу, па напокон и претраживачу. У тој ситуацији бровсер уопште не зна да ли сте услужени статичком страницом или је код који ваш претраживач пиказује генерисан од стране Веб апликације. Није му ни битно, докле год прослеђени код разуме и може да прикаже.

[2] https://elab.fon.bg.ac.rs/praktikum-iz-interneta-inteligentnih-uredaja/primer-18-razvoj-veb-i-mobilne-aplikacije-za-pracenje-zdravstvenog-stanja-pacijenata/

Увођењем програмских елемената у приказ веб страница настала су два типа веб страница, а то су:

- Статичке – изглед им је дефинисан већ приликом прве израде
- Динамичке – изглед им се дефинише кроз интеракцију са сервером

Најпознатије технологије – програмски језици за развој Веб апликација су:

- ASP/ASP.NET (C#)
- Java
- PHP

Битна квалитативна својства самих Веб апликација и процеса развоја могу се повезати са општим стандардом евалуације квалитета софтвера ISO/IEC 9126-1.

Једна од основних мера квалитета Веб апликација су перформансе. Главна перформанса која утиче на успешност једне Веб апликације је време одзива, која првенствено утиче на квалитет и употребну вредност апликације. Ако је некој Веб апликацији потребно много времена да би се учитала, корисник ће одустати од датог сајта и потражити следећи.

Због тога је веома битно обратити пажњу на време одзива приликом креирања Веб апликације да би наша апликација била квалитетна и посећена. Време одзива представља период који прође од упућеног захтева 7 корисника до повратне информације коју корисник добија и ако прође дуже од 8 секунди, корисници најчешће одустају од датог сајта.

Процес развоја софтверских производа, па тако и Веб апликација је скуп метода, активности, праксе и аутоматизованих алата које програмери користе за развој и одржавање система и софтвера. Постоје две основне врсте методологија развоја софтвера, а то су:

- Секвенцијална метода (метода водопада)
- Итеративна метода

Секвенцијална метода је старији приступ развоја софтвера, и употребљив је искључиво на мање захтевним апликацијама. Ова метода даје резултате високог квалитета али један од главних недостатака јој је веома велико трајање развоја оваквог система, али и немогућност накнадних отклањања недостатака. Из овог разлога су развијене различите варијације модела водопада у којима се фазе дизајна и имплементације у извесној мери преклапају. Све ове методологије деле неколико заједничких атрибута: број фаза кроз које пролазе, број итерација који је потребан да би се креирао софтвер, као и типично трајање једне итерације. Све фазе се извршавају секвенцијално, и увек постоји бар једна итерација која завршава са креирањем софтвера. Разлике између методологија су у редоследу по којем се улази у одређене фазе, броју захтеваних итерација и трајању сваке итерације.

Итеративна методологија настоји да се животни циклус софтвера састоји од неколико итерација које могу да се преклапају у времену. Овакав тип метедологије омогућава лакше прелажење са процеса на процес али и враћање на претходне процесе уколико дође до грешке што омогућава исправљење грешака, за разлику од секвенцијалне методе код које је овај поступак доста компликованији и не постоји враћање када се једном прође кроз дати корак док се читав процес не заврши. Ова метода је доста практичнија за употребу од методе водопада и примењује се у захтевнијим Веб апликацијама и софтверским производима.

2.3. Вишеслојна архитектура софтвера

[3]Софтверска архитектура једног рачунарског система представља скуп објеката који су потребни за функционисање неког система, а садрже софтверске елементе, односе између елемената и њихове особине. Софтверска архитектура свој фокус усмерава на правилно коришћење елемената и компоненти у оквиру апликације. Неизоставни појам уз софтверску архитектуру је софтверски дизајн. Да би софтвер био што бољег квалитета мора се усавршити и дизајн али и архитектура и ове две особине се морају комбиновати.

Архитектуру софтвера можемо поделити према општим обрасцима, а то су:

- Монолитна архитектура
- Двослојна (клијент-сервер) архитектура
- Трослојна и вишеслојна архитектура (тхрее – тиер /мулти – тиер)
- Сервисно орјентисана архитектура софтвера

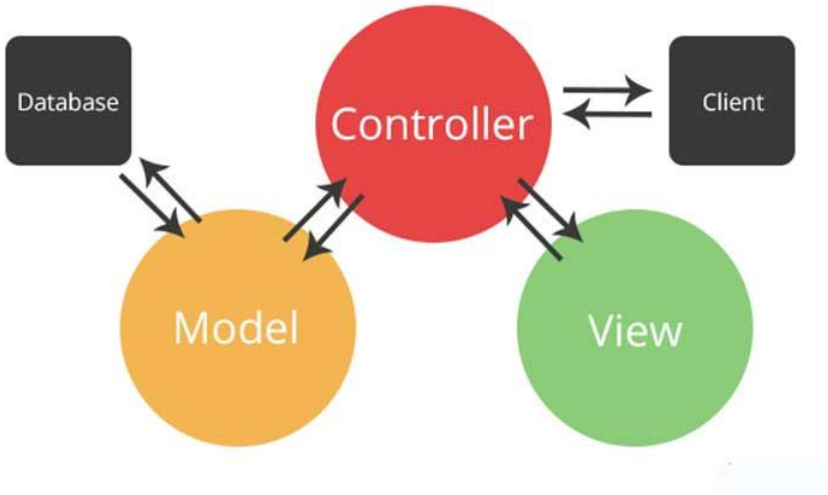
Вишеслојна архитектура је појам који се односи на архитектуру која се састоји из три главна слоја. Трослојна архитектура је таква архитектура код које је потребно увести још један слој тзв. средњи слој у коме се налази апликативни сервер. Почетак развоја оваквих архитектура датира од 1990 године и за развој овакве архитектуре могуће је применити више различитих технологија. Трослојна архитектура је генеричка за вишеслојне архитектуре. Овде се врши даља подела на компоненте у оквиру средњег слоја са циљем још већег повећања скалабилности, односно перформанси. Средњи слој се дели на два слоја: један је намењен за опслуживање Веб клијената, а други врши implementaciju пословне логике система. Некада се средњи слој делио на два или више слоја у зависности од апликације па добијамо вишеслојну (”multitier”) архитектуру.

МВЦ[4] архитектурни образац дели објектно орјентисану апликацију на три целине:

1. Model – задужен за управљање подацима
2. View – има задатак да управља приказом информација
3. Controller – представља логику апликације (преноси захтеве и усмерава улазе)

[3] http://www.tfzr.uns.ac.rs/Content/files/1/UDZBENIK%20softversko%20inzenjerstvo%202.pdf
[4] https://www.ucim-programiranje.com/2013/02/mvc-model-view-controller/

На следећој слици је ислустрован приказ МВЦ архитектуре:



Слика 3. MVC образац[5]

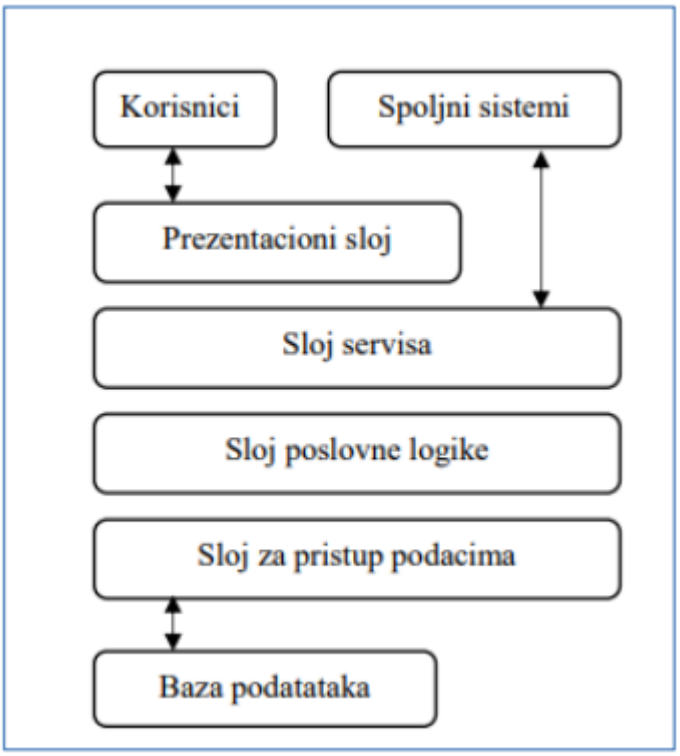
Модел чува податке, Controller задаје команде моделу за преузимање података, а затим контролер задаје команде којима се подаци приказују у View . View генерише нови излаз (output) који је представљен кориснику, на основу промена које су се десиле у моделу. Controller може да шаље команде Модел-у да промени стање модела.

Controller може да шаље команде на View да се измени приказ података на основу измена модела или да се прикаже друга врста или начин понашања View. Сваки View има компоненте које су повезане са Controller-ом тако да су међусобно уско повезани и не би могли засебно да функционишу. MVC захтева раздвајање одговорности (SoC) јер су модел и логика контролера раздвојени од корисничког интерфејса. У оквиру Веб апликација, ово значи да је HTML код издвојен од остатка апликације, што чини одржавање и тестирање једноставнијим и лакшим.

Објектно орјентисана имплементација MVC модела дефинише посебне класе за сваку компоненту. Не постоје било каква ограничења када је у питању имплементација домен модела. Можемо креирати модел помоћу стандардних C# објеката и имплементирати складиштење помоћу било које базе података, објектно-релационог мапирања, или других алата који су подржани од стране . NET-а.

MVC се такође може користити за изградњу фрамењорк-а интерактивних апликација. Са слике 7. можемо видети да корисници приступају презентационом слоју , а тај слој представља сам изглед апликације и он се може имплементирати коришћењем MVC патерна.Такође додат је слој сервиса који комуницира са спољним системом у циљу побољшања захтева корисника и реаговања на грешке апликације.

[5] https://www.manuelradovanovic.com/2017/09/uvod-u-asp-net-mvc.html



Слика 4. Вишеслојна архитектура (ASP.NET апликација) [6]

Путем презентационог слоја се приступа слоју сервиса, а он даље обрађује податке и шаље повратне информације, али и комуницира са осталим деловима апликације. Ако је потребно да неки систем комуницира са апликацијом, он то ради преко слоја сервиса користећи сопствени презентациони слој.

Предност која се добија коришћењем слојева у креирању неке апликације је могућност за раздвајање надлежности делова апликације, али и лакша измена и поновна употреба делова кода. Креирање апликације са слојевима се предлаже код комплекснијих апликација, али је све заступљеније и код мање захтевних апликација.

Презентациони слој представља кориснички интерфејс и ниједна апликација не би била употребљива без њега, због тога има изузетну важност у слојевитим апликацијама. Презентациони слој се додаје на постојећи средњи слој апликације.

Основне компоненте презентационог слоја су:

- Кориснички интерфејс
- Презентациона логика

Кориснички интерфејс, иако неки сматрају мање битном компонентом апликације, представља неопходан део слојевитог система јер он даје кориснику алате помоћу којих ће програм користити, тако да би без корисничког интерфејса апликација била неупотребљива. Свака опција програма коју корисник одабере приказиваће му се помоћу корисничког интерфејса.

[6] http://tfzr.rs/Predmet/softversko-inzenjerstvo-2/download

Презентациона логика је повезана са приказом података на екрану. Овај тип логике одговара на корисникове захтеве, шаље повратне информације и дате захтеве даљим слојевима апликације. Презентациона логика има задатак да обрађује информације које проследи кориснички интерфејс и да их шаље ка средњем слоју и исто тако повратне информације од средњег слоја ка корисничком интерфејсу.

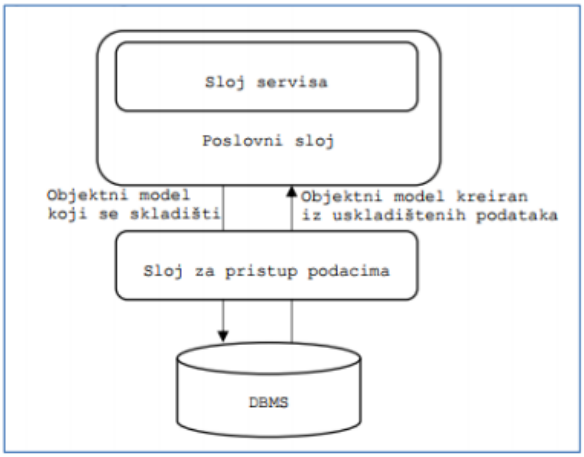
Слој сервиса се најчешће посматра као део пословног слоја. Могло би се рећи да слој сервиса представља границу између презентационог слоја и слоја пословне логике. Задужења која има сервисни слој су следећа:

- Обрада захтева
- Одржавање сигурности система
- Комуникација и раздвајање захтева пословног и презентационог слоја
- Обавештавање о грешкама

Пословни слој је задужен за обраду свих захтева система и садржи сву логику и правила која одржавају у целини неку Веб апликацију. Овај слој обрађује процесе, садржи алгоритме који управљају системом, али је задужен и за валидацију захтева. Када бисмо разложили пословни слој, подела би била следећа:

- Објектни модел који моделује учеснике у пословном процесу
- Пословна правила која представљају клијентску политику и захтеве
- Сервисе који имплементирају аутономне функционалности
- Процес рада – дефинише како се врши пренос података из једног модула у други

Слој за приступ подацима комуницира са базом и шаље податке апликацији на основу захтева које прими. Слој за приступ подацима представља једино место у систему где је позната веза са базом података и именима табела. Ту је садржан сав код који врши упис, измену и брисање у бази. Слој за приступ подацима шаље податке пословном слоју који не садржи никакве информације о бази, он их само прослеђује до одређених делова апликације. Слика 7. приказује однос слоја за приступ подацима са осталим слојевима и базом података.



***Слика 5**. Слој за приступ подацима* [7]

[7] http://www.tfzr.uns.ac.rs/Content/files/0/SOFTVERSKO%20INZENJERSTVO%202%20tema%2034%20viseslojna%20i%20mvc.pdf

3. РАД СА БАЗОМ ПОДАТАКА

3.1. Опис базе података

База података представља колекцију међусобно повезаних података који су организовани у табеле и друге структуре података, а користе једну или више апликација. Основна намена базе података је да буде репозиторијум (складиште) за податке. Подаци могу бити различитог типа, текстуални, нумерички, слике, аудио и видео записи и сл. Подаци у базама података су организовани у дводимензионалне табеле. Табела може да има више колона, где свака колона представља неку особину или атрибут. Врсте табеле чине конкретни подаци, односно конкретне вредности особина/атрибута неког објекта. На пример, једна табела може да садржи информације о ученицима. Колоне табеле могу да дефинишу име, презиме, годину рођења ученика, и сл. Врсте у таквој табели су ученици, тако да се свака врста односи на једног ученика. Које ће табеле да садржи база података зависи од проблема за који треба реализовати базу података.

На пример, база података се може одности на школу, па ће у том случају табеле бити о ученицима, наставницима, одељењима, и сл. Поступак избора и дефинисања табела за базу података је део процеса моделирања односно изградње модела података. Међусобна повезаност података је оно по чему се база података разликује у односу на фајл системе (датотеке) и програме за унакрсна израчунавања као што је *Excel*. Повезаност података обезбеђује значајне предности код претраживања када корисник може да на основу веза извуче много више података. На пример, ако постоји табела која чува податке о ученицима и табела са подацима о одељењима, веза између ученика и одељења може да обезбеди да одговарајућим захтевом (*SQL* упитом) прикажете све ученике жељеног одељења. База података садржи и тзв. метаподатке, односно податке о самој структури базе података. Метаподаци могу да се односе на имена табела, имена колона у свакој табели, на податке о корисницима података, као и разним помоћним структурама које обезбеђују брз приступа подацима (индекси).

3.2. SQL језик

За рад с релацијском базом података користе се упитни језици. Данашњи упитни језици који се користе су најчешће процедурални и непроцедурални. За релацијску базу података се користе непроцедурални језици. Постоји више непроцедуралних језика, а највише се користи *SQL* језик. Иако се назива упитним језиком, *SQL* има наредбе за свеобухватан рад с релацијском базом података, а то су:

- *креирање таблице*
- *унос података у таблице*
- *брисање података из таблица*
- *наредбе за дефинисање базе података (DDL-Data Definition Language)*
- *наредбе за манипулацију подацима (DML-Data Manipulation Language)*
- *управљачке наредбе.*

SQL (***Structured Query Language***) , тј, његов развој је започео 1974. године, када је објављен чланак *D.D.Chamberlin* и *R.F.Boyce* у којем они описују упитни језик за претраживање под називом *SEQUL*[8]. Након тога 1975. им се придружује и *M.Hammer*, те они заједничко представљају концепт језика *SQUARE*. Убрзо након тога *SQUARE* мења назив у *SEQUEL2*, и тај језик је коришћен у развоју првог прототипа релацијског састава за управљање базама података који је имао назив *«System P»,* а касније тај језик мења име у *SQL*. *SQL* је релацијски потпун јер за свих 5 основних релацијских оператора постављао је семантички еквивалентне *SQL* наредбе. *SQL* језик је стандардизован од *ISO* (1986.г.) и *ANZI* (1986.године) института и данас се ради у већини релацијских система базе података. *SQL* омогућава прављење упита над више релација истовремено. Релацијска база података спада у савремене базе података уз објектни или димензијски модел. *SQL* је декларативан језик у којем корисник бира само жељени резултат. *SQL* 1992 подупире само функције које крајњи корисник жели, али такође подупире и оне функције које обрађују апстрактне типове података.

Апстрактни типови података се данас могу додавати или брисати из система, без да то има икаквог утицаја на сам састав. Иако такав приступ повећава целокупну функционалност састава, сами састави нуде веома ограничену потпору за оптимисање операција *SQL*-стандардни упитни језик.

3.3. Основе система *RDBMS*

Савремени напредни системи *RDBMS* (Relational Database Management System – RDBMS) не само да складиште податке, већ и управљају њима тако што ограничавају податке који могу ући у систем, а такође омогућују узимање података из система. Ако вам је потребно само да сместите податке негде на сигурно, можете да користите било какав систем за складиштење података. Системи *RDBMS* вам омогућују да одете изнад самог складиштења података у домен дефинисања начина на који би ти подаци требало да изгледају или пословних правила за податке. Не треба мешати оно што се назива „пословним правилима за податке" са генерализованим пословним правилима која управљају целокупним системом (на пример, неко не може ништа да види док се не пријави на систем или аутоматско прилагођавање текућег периода у рачуноводственом систему првог у месецу). Ти типови правила у ствари се могу применити на било ком нивоу система (у данашње време то се обично примењује у средњем или клијентском слоју вишеслојног система). Уместо тога, оно по чему се овде говори су пословна правила која се посебно односе на податке. На пример, не може постојати поруџбина са негативном количином. Код система *RDBMS* та правила могу да се угради директно у интегритет саме базе података[9].

[8] Лазаревић Б.,2003, Базе података, ФОН Београд.

[9] https://www.codecademy.com/articles/what-is-rdbms-sql

3.4. Преглед објеката базе података

Систем *RDBMS* као што је *SQL* сервер садржи велики број објеката. Код *SQL* сервера списак неких важнијих објеката базе података садржи ствари као што су:

- *сама база података*
- *дневник трансакција*
- *склопови*
- *табеле*
- *групе датотека*
- *ускладиштене процедуре*
- *приказ*
- *извештаји*
- *функције које дефинише корисник*
- *корисници*
- *роле*

3.4.1. Базе података као објекат

База података је у ствари објекат највишег нивоа који може да се референцира у оквиру датог *SQL* сервера. (Технички говорећи, и сам сервер се може сматрати објектом, али не из било какве реалне „програмерске" перспективе.) Већина других објеката *SQL* сервера, али не сви, изведени су од објекта базе података.

База података обично представља групу која садржи барем скуп објеката табела и најчешће друге објекте као што су ускладиштене процедуре и прикази који се односе на одређено груписање података сачуваних у табелама базе података.

Које типове табела чувамо у само једној бази података, а шта се налази у засебној бази података? Посматрајући једноставно рећи ћемо да се сви подаци за које се сматра да припадају само једном систему или да су значајно повезани, чувају у једној бази података. Систем *RDBMS* као што је *SQL* сервер може да има више корисничких база података на само једном серверу, а може да има и само једну. Број база података који може бити смештен на једном *SQL* серверу зависи од фактора као што су капацитет (снага *CPU, I/O* ограничења диска, меморија итд.), аутономија (желите да једна особа има право управљања сервером на којем се налази тај систем, а да неко други има администраторска права на другом серверу) или само број база података које ваша компанија или клијент има. Велики број сервера има само једну производну базу података, док их други могу имати и више. Треба имати на уму да код сваке верзије *SQL* сервера коју налазимо у употреби данас (*SQL* сервер 2000 је био већ пет година стар када је замењен), има могућност постојања већег броја примерака *SQL* сервера - укључујући и посебно пријављивање и посебна права управљања и то све на физички једном серверу.

Када се први пут покрене *SQL* сервер, почиње се са системским базама података:

- *мастер*
- *модел*
- *msdb*
- *tempdb*

3.4.2. Дневник трансакције

Сама датотека базе података није место на коме се дешава већина ствари. Иако се подаци дефинитивно читају одатле, свака измена која се прави не иде иницијално у саму базу података. Уместо тамо, они се серијски уписују у дневник трансакција. У неком каснијем тренутку база података задаје тренутак провере контролне тачке - у том тренутку времена све измене из дневника се преносе на саму датотеку базе података.

База података има уређење које се заснива на случајном приступу, али је сам дневник по својој природи серијски. Док случајан приступ датотеци базе података омогућује брз приступ, серијски приступ дневнику омогућује праћење ствари одговарајућим редоследом. Дневник акумулира измене за које се сматра да су урађене, а затим неколико измена одједном уписује у саму датотеку базе података. Да би имали функционалну базу података потребни су и датотека базе података и дневник трансакција.

3.4.3. Основни објекат базе података – табела

Базе података сачињавају многе ствари, али ни једна од њих није толико централна за саму садржину базе података као што је табела. Табела се може сматрати еквивалентом главној књизи рачуновође или радном листу у *Excelu*. Састоји се од такозваних података домена (колоне) и података ентитета (редови). Сами подаци из базе података чувају се у табелама.

Свака дефиниција табела садржи и метаподатке (описне информације о подацима) који описују природу података које би табела требало да садржи. Свака колона има свој скуп правила која одређују шта се може чувати у тој колони. Кршење правила у било којој колони може да доведе до тога да систем одбаци ред који желите да унесете или да одбије да ажурира постојећи ред или да спречи брисање реда.

```
CREATE TABLE `products` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(255) NOT NULL,
  `price` float NOT NULL,
  `quantity` float NOT NULL,
  `uom` varchar(255) NOT NULL,
  `description` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

***Слика 6**. SQL код за креирање табеле*

3.4.4. Индекси

Индекс представља објекат који постоји само унутар радног окружења одређене табеле или приказа. Индекс функционише веома слично као и индекс на крају енциклопедије; постоји некаква вредност по којој се претражује (или кључна вредност), која је уређена на одређени начин и када се пронађе, добије се други кључ помоћу којег се могу пронаћи саме информације које су вам потребне.

Индекси обезбеђују начине који омогућавају да се убрза претраживање информација. Индекси спадају у две категорије:

Груписане - Само један овакав индекс може да постоји по табели. Ако је индекс груписани, то значи да је табела у којој се налази физички уређена у складу са тим индексом. Када би индексирали енциклопедију, кластерски индекс би био број стране; информације у енциклопедији су сачуване према редоследу бројева страна.

Негруписане – Може их бити више у свакој табели. Он се више уклапа у оно што се обично подразумева када се чује реч „индекс". Овај тип индекса указује на неку другу вредност која ће омогућити да се пронађу подаци. У енциклопедији, то би био индекс кључних речи на крају књиге.

Треба обратити пажњу на то да прикази који имају индексе или индексирани прикази - морају имати бар један груписани индекс да би уопште могли имати негруписане индексе.

```
-- Indexes for table `products`
--
ALTER TABLE `products`
  ADD PRIMARY KEY (`id`);
```

***Слика 7**. SQL код за креирање индекса*

3.4.5. Окидачи

Ограничење је још један објекат који постоји само унутар граница табеле. Ограничења су баш то што и њихово име говори, објекти који ограничавају податке у табели тако да испуњавају одређене критеријуме. Ограничења се на неки начин такмиче са окидачима као могућа решења за захтеве у вези са интегритетом података. Међутим, они не представљају исту ствар јер имају различите предности.

3.4.6. Групне датотеке

Према подразумеваној опцији, све табеле и све остало у вези са базом података (изузев дневника) чува се у једној датотеци. Та датотека је део нечега што се назива примарна група датотека. Међутим, то није све што се тиче уређења.

SQL сервер дозвољава дефинисање нешто више од 32000 секундарних датотека. Те секундарне датотеке могу се придружити примарној групи датотека или се могу направити као део једне секундарне групе датотека или више таквих група. Док постоји само једна примарна група датотека (и она се у свари назива „*Primary*"), могу постојати до 255 секундарних група датотека. Секундарна група датотека се прави као опција у оквиру команде *CREATE DATABASE* или *ALTER DATABASE*.

3.4.7. Дијаграми

Дијаграм базе података визуелно представља дизајн базе података укључујући различите табеле, имена колона у свакој табели и везе између табела. Постоји термин дијаграм ентитет-веза или *ERD*. На *ERD* дијаграму база података је подељена на два дела: ентитета (као што су „снабдевач" и „производ") и везе (као што су „набавка" и „куповина").

3.4.8. Прикази

Приказ је нешто налик на виртуелну табелу. У већини случајева, приказ се и користи као табела, али се од табеле разликује по томе што не садржи своје податке. Приказ представља унапред планирано мапирање и представљање података који се чувају у табелама. Тај план се чува у бази података у облику упита. Упит захтева податке из неколико колона (није неопходно да буду из свих) из једне или више табела. Враћени подаци можда морају, или не (зависно од дефиниције приказа) да испуњавају одређене критеријуме да би се појавили у том приказу. До *SQL* сервера 2000 основна сврха приказа била је контрола онога што корисник може да види. На то утичу две основне ствари: безбедност и лакоћа коришћења. Са приказима можете да контролишете оно што корисници виде, тако да ако постоји део табеле којем може да приступи само неколико корисника (на пример, детаљи у вези са платама), можете да направите приказ који садржи само колоне којима сви могу да приступе. Поред тога, приказ се може направити тако да корисник не мора да претражује скроз непотребне информације. Поред ових најосновнијих примена приказа, постоји могућност да се направи и такозвани индексирани приказ. Он је исти као и било који други приказ, изузев што у односу на тај приказ можемо да направимо индекс.

То има неколико утицаја на перформансе (неки су позитивни, а неки негативни):

- *Прикази који референцирају више табела извршавају се много брже ако је приказ индексиран зато што је спајање табела унапред направљено.*
- *Агрегирања која се обављају у приказу унапред се рачунају и чувају се као део индекса, то опет значи да се агрегација обавља у једном тренутку (када се унесе или ажурира неки ред), а затим се може читати из информација које садржи индекс.*
- *Уношење или брисање реда је спорије зато што и индекс у приказу мора одмах да се ажурира, ажурирања су такође спорија ако ажурирање утиче на колону која представља кључ у индексу.*

3.4.9. Ускладиштене процедуре

Ускладиштене процедуре су историјски, па чак и у ери *.NET*-а, основа програмске функционалности код *SQL* сервера. Ускладиштене процедуре у ствари представљају низове исказа *Transact-SQL* (језик који се користи за задавање упита код *Microsoft SQL* сервера) спојених у једну логичку целину. Дозвољавају коришћење променљивих и параметара као и конструкција које омогућују избор и пролазак у циклусима. Ускладиштене процедуре нуде неколико предности у односу на слање појединачних исказа серверу због следећих разлога: Референцирају се коришћењем кратких имена, уместо дугачких низова текста, због чега је потребно мање мрежног саобраћаја како би се извршио код унутар ускладиштене процедуре. Оне су преоптимизоване и прекомпајлиране, чиме се штеди мала количина времена сваки пут када се ускладиштена процедура извршава. Учаурују процес, обично из безбедносних разлога или да би се сакрила сложеност базе података. Могу се позивати из других ускладиштених процедура, због чега се могу сматрати поново употребљивим у неком ужем смислу. Поред тога, можете да користите било који *.NET* језик ако у ускладиштене процедуре желите да убаците програмске конструкције које се не налазе у матерњем језику *T-SQL*.

3.4.10. Функције које дефинише корисник

Функције које дефинише корисник (*UDF*-ови) имају веома много сличности са ускладиштеним процедурама, а разликују се по томе што:

Могу да врате вредност чији тип може бити већина типова података који постоје код *SQL* сервера. Повратна вредност не може бити типа *text, image, cursor, timestamp*. Не могу имати „споредне ефекте". У основи, оне не могу да ураде ништа изван домена функције, као што је мењање табела, слање електронских порука или мењање параметара система или параметара базе података.

Функције које дефинише корисник сличне су функцијама које бисте користили у стандардном програмском језику као што је *PHP*. Можете да проследите више од једне променљиве и да добијете резултат на основу њих. Функције које дефинише корисник код *SQL* сервера разликују се од функција које можете наћи у многим процедуралним језицима али по томе да се све променљиве које се прослеђују функцији увек прослеђују по вредности. Међутим, постоје неке добре вести, а то су да можете вратити посебан тип података који представља табелу.

3.4.11. Корисници и роле

Ово двоје увек иду заједно. Корисници су прилично еквивалентни налозима. Укратко, овај објекат представља идентификатор потребан да би се неко пријавио на *SQL* сервер. Свако ко се пријављује на *SQL* сервер мора да се преслика у корисника (директно или индиректно зависно од безбедносног модела који се користи). Корисници припадају једној роли или већем броју рола. Права да се обаве одређени поступци код *SQL* сервера могу се доделити директно кориснику или роли којој припада један или више корисника.

3.4.12. Правила

Правила и ограничења ограничавају информације које могу да уђу у табелу. Ако ажурирани или убачени запис крши правило, убацивање или ажурирање ће се одбацити. Поред тога, правило може да се користи за дефинисање ограничења код кориснички дефинисаних типова података. За разлику од правила, ограничења нису баш самостални објекти, већ делови метаподатака који описују одређену табелу.

Правила се морају сматрати објектом који ту постоји само због компатибилности уназад и требало би их избегавати у новим апликацијама.

3.4.13. Подразумеване вредности

Постоје два типа подразумеваних вредности. Постоји подразумевана вредност која представља самосталан објекат и подразумевана вредност која у ствари и није објекат већ представља метаподатке који описују одређену колону у табели (слично као што имамо правила, која представљају објекте, и ограничења, која нису објекти већ метаподаци). Имају исту намену. Ако приликом убацивања новог реда не обезбедите вредност за неку колону, а колона има дефинисану подразумевану вредност, аутоматски ће се убацити вредност која је дефинисана као подразумевана.

3.4.14. Типови података које дефинише корисник

Типови података које дефинише корисник представљају проширења за типове података које дефинише систем. Почевши од ове верзије *SQL* сервера, могућности у овој области су скоро бескрајне. Иако су *SQL* сервер 2000 и претходне верзије имали појам типова података које дефинише корисник, ти типови података били су ограничени на различито филтрирање постојећих типова података. Код *SQL* сервера 2005 имате могућност да везујете *PHP* склопове за ваше типове података, што значи да можете имати тип података који чува (са разлогом) све што можете сачувати у PHP објекту.

3.4.15. Подаци типа *NULL*

Шта ако имате ред који не садржи никакве податке у одређеној колони, то јест, шта ако једноставно не знате ту вредност? На пример, рецимо да постоји запис који покушава да чува информације о раду компаније током дате године. Сада замислите да је једно од поља проценат пораста у односу на претходну годину, али немате записе за годину пре првог записа у бази података. Можете доћи у искушење да унесете нуле у колону. Међутим, да ли ће то обезбедити праве информације? Они који не знају детаље могу помислити да та нула означава да сте имали проценат раста од нула посто, а не чињеницу да једноставно не знате вредност за посматрану годину.

Вредности које су неодређене означавају се као вредности *NULL*. Веома је тешко дефинисати вредност *NULL* јер значи да не знате која је вредност. Могла би бити 1; могла би бити 347; могла би бити -294 и то је све што знамо. Укратко, значи да је вредност недефинисана или можда неприменљива.

4. ОСНОВНЕ НАРЕДБЕ SQL-а

Основна наредба у *SQL*-у је *SELECT* израз *FROM* име таблице, у преводу ИЗАБЕРИ израз ИЗ име таблице. У наредби реч ИЗРАЗ замењује се са именима редова које је потребно приказати или у случају да је потребно видети све са *. Поред имена реда може се написати ново име реда под којим ће се видети у извештају, али у таблици остаје све по старом.

Следећа *SELECT* наредба се разликује код *SQL*-а и *SAP*-а. Као што је приказано у претходној наредби да сваки ред има своје име, али некад може бити потребно да се у неком извештају споје два реда таблице у један ред извештаја. Спајање се врши у *SQL* -у са знаком +, док се иста радња у *SAP*-у ради са знаком назива пипе (*Alt Gr + W*). Можемо још споменути и кључну реч *TOP x*, која нам даје само првих *x* редова, што је корисно при раду са великим таблицама када често морамо проверавати резултате наших израза. Постоји још једна могућност коришћења наредбе *SELECT* без додатка *FROM* јер тај податак не мора бити унешен у таблицу. Следећи користан додатак наредби *SELECT* је захтев *WHERE* који показује који се редови могу видети јер у већини претходних примера имамо приказане све редове.

Наредни корак уношења података би био употреба логичких оператора тако да је могуће још више проширити захтеве за претраживање. Логички оператори су *AND*, *OR* и *NOT*. Код сложенијих израза би требало водити рачуна о приоритетима. Хијерархија примене израза гласи:

- *Заграда ( ),*
- *Дељење / и множење **,*
- *Сабирање + и одузимање -,*
- *NOT (не),*
- *AND (и),*
- *OR (или).*

Ако је потребно дефинисати у захтеву распон имамо две могућности. Једна је да радимо са операторима упоређивања (<, =, >, ...), а друга је да користимо кључну реч *BETWEEN*. Међутим, уколико је потребно наћи податак којем је захтев да се поклапа са једном од вредности у листи користимо кључну реч *IN*. Некад се може догодити да тражимо, на пример, неко име које почиње као ... или телефонски број. Тако задате захтеве је мало теже добити кроз прошле наредбе или операторе, па за то служи кључна реч *LIKE*. Оператор *IS NULL* нам даје информацију, на пример, за коју особу из таблице немамо податке. Уз коришћење ове кључне речи може се добити резултат који је читљивији, односно можемо резултат сортирати по неком редоследу.Међутим, кад постоје само бројеви или само слова јасно је како ће бити сортирани ти низови, али кад је све то измешано поставља се питање, како ће сад то бити сортирано? Ту можемо позвати у

помоћ једну процедуру у *MS SQL*-у која ће нам дати редослед низа по којем се врши сортирање, а то је *EXEC SP_HELPSORT* [10].

5. КОРИШЋЕНА РАДНА ОКРУЖЕЊА

5.1 *PHPMyADMIN*

MySQL језик је свеобухватни приручник који покрива све аспекте језика упита који програмери користе за комуникацију и интеракцију са базом, путем својих апликација, са *PHPMyADMIN*-овим системом управљања релационим базама података. Има дугу историју. Предмет овог дела је у потпуности *Firebird*-ова имплементација *SQL* релације језика базе података. *Firebird* се блиско усклађује са међународним стандардима за *SQL*, од типа података подршка, структуре за складиштење података, референтни механизми интегритета, за манипулацију подацима могућности и привилегије приступа. Такође примењује робустан процедурални поступак, језик процедурални *SQL(PSQL)* за ускладиштене процедуре, окидаче и динамички извршну датотеку кодних блокови[11].

Изразити подскупови *SQL*-а примењују се на различите секторе активности. Подскупови *SQL* на језику *MySQL* имплементације су:

• Динамички *SQL*

• Процедурални *SQL*

• Уграђени *SQL*

• Интерактивни *SQL*

Динамички *SQL* је главни део језика и представља изразе које су клијентске апликације проследиле путем јавног *MySQL API*-а и обрађује га механизам базе података.

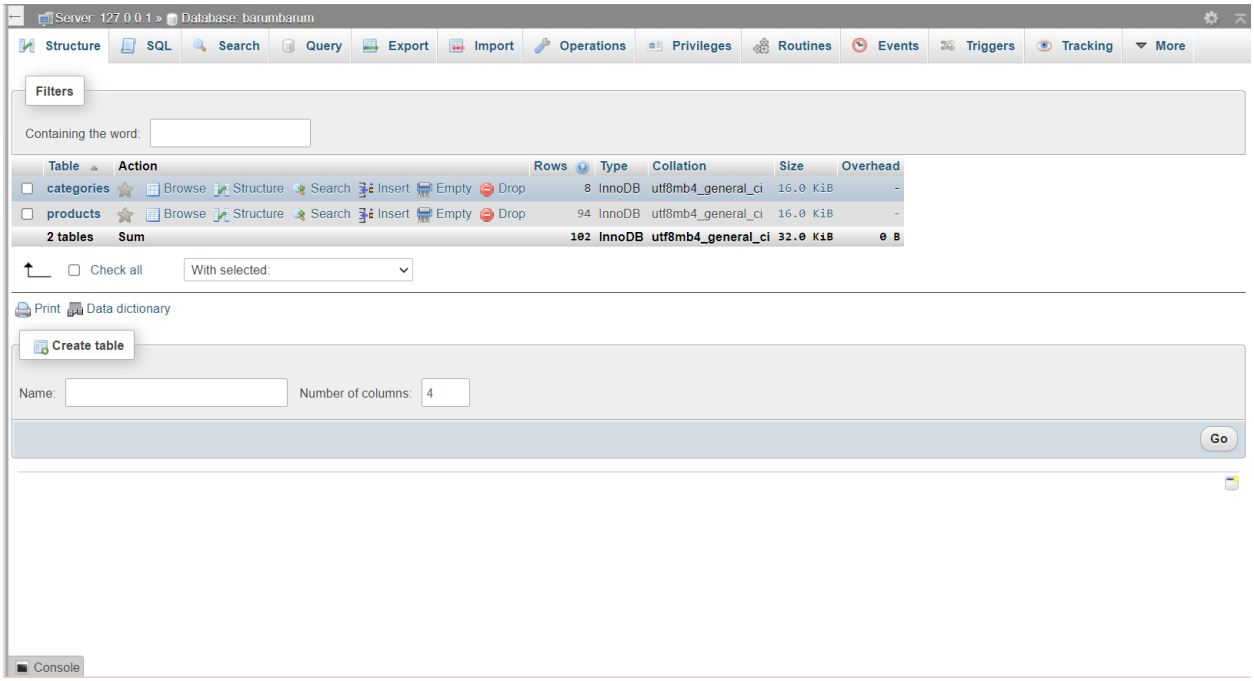
Процедурални *SQL* увећава динамички *SQL* како би омогућио сложене изразе који садрже локалне променљиве, задаци, услови, петље и друге процедуралне конструкције.

Уграђени *SQL* дефинише динамички *SQL* подскуп који подржава *MySQL gpre*, апликација која омогућава уградњу *SQL* конструкције у програмски језик (*C, C ++, Pascal*) унапред обрађује те уграђене конструкције у одговарајуће *MySQL API* позиве.

Интерактивни *SQL* односи се на језик који се може извршити помоћу *sql*, командне линије, апликација за интерактивни приступ базама података. Као уобичајена клијентска апликација, њен матерњи језик је *DSQL*. Такође нуди неколико додатних команди које нису доступне изван његове специфичности.

[10] https://docs.microsoft.com/en-us/sql/relational-databases/system-stored-procedures/sp-helpsort-transact-sql?view=sql-server-ver15
[11] https://www.phpmyadmin.net/

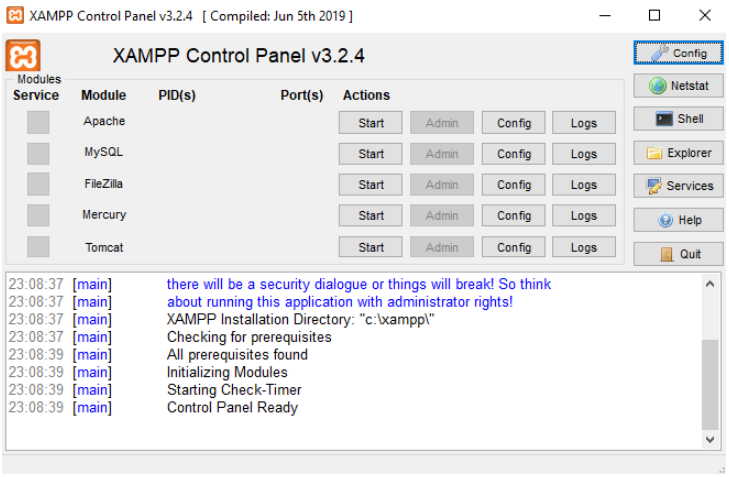
PHPMYADMIN (Слика 2) је професионално интегрисано развојно окружење (*IDE*) за развој и управљање базама података MySQL.



Слика 8. *Изглед PHPMYADMIN*

Потребно је инсталирати *PHPMYADMIN* (исталациони фајл пронаћи на интернету)

Приликом инсталације потребно је да се преузме Xmp. Код инсталације овог програма потребно је да се дефинише место инсталирања програма. Под фолдер „htdoc“ је фолдер у коме се смештају пројекти на коме се ради. Под фолдер „phpmyadmin“ је фолддер који садржи подфолдер који сте креирали. У phpmyadmin се приступа тако што омогућите у контролном центу приступ.



Слика 9. *Контролни панел локалног сервера*

Потребно је да се омогући *mysql* і *apach*, тиме смо покренули локални сервер.

Да би се наша апликација повезала на нашу базу потребно је следећи код:

database.php

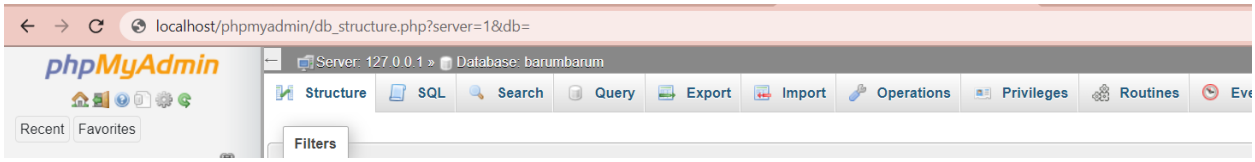
Application > config > database.php

```php
<?php

return [
    'user'            => 'root',
    'password'        => '',
    'host'            => '127.0.0.1',
    'database'        => 'diplomski',
    'charset'         => 'utf8',
    'emulate_prepares' => false,
    'error_mode'      => PDO::ERRMODE_EXCEPTION,
    'fetch_mode'      => PDO::FETCH_ASSOC,
];
```

Слика 10. *PHP функција за повезивање на базу*

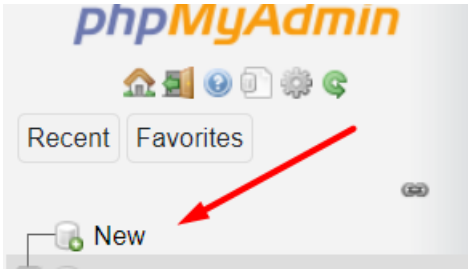
Следећи корак је да у php-у направимо базу. Прва је да креирамо помоћу sql-а, а друга је преко дизајна.

Када у претживачу претражимо адресу „localhost/phpmyadmin“ отвара нам се следећа страница:

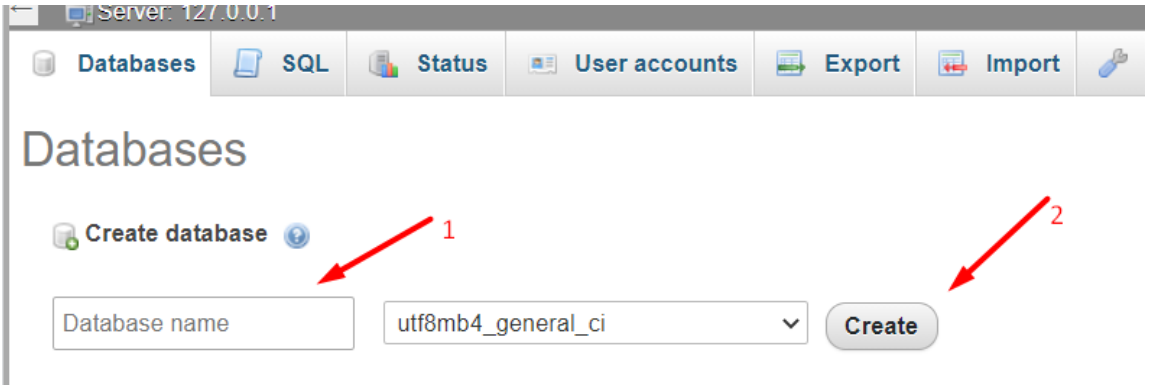


Слика 11. *Приказ наше базе*

Да би смо креирали нашу базу потребно је да се кликне на дугме „new“, и отвара нам се следећи прозор:



Слика 12. *Функција за креирање базе*



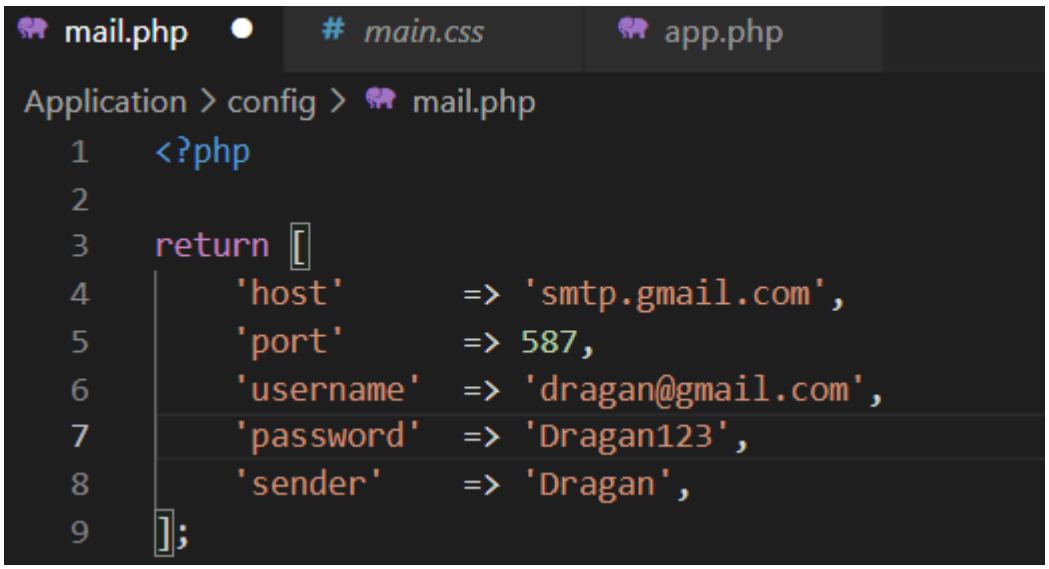
Слика 13. *Приказ за унос имена базе*

Потребно је да се у поље „Datebase name“ упише наше име. Следеће поље „UTF8“ служи за читање наших слова, односно слова са квачицом (ч,ћ,ж,ђ,ш,љ.њ).

Када се то одради, следећи корак је дугме „create“. Овим дугметом креирамо базу података.

5.2 PHP

PHP се највше користи међу програмерима, погодан је за израду веб сајтова. PHP има готове контроле које је лако имплементирати у програм. Овај програмски језик је изузетно лак за учење и синтакса је веома слична осталим програмским језицима. У тренутку када корисник одради неку акцију програмски језик у себи има такозване руте које корисника дистанцирају на одређену адресу. Све што се дешава на страници заслузан је позадински код тј. операције, варијабле и неке промене. Те операције корисник не види зато што је то одрађено у позадини, а кориснику се приказује HTML, CSS код. PHP је извучен из програмског језика C-а који је осмишљен за брзе и лаке динамичке операције. Да бисмо на нашем серверу користили PHP скрипту потребно је да наш сервер има омогућен PHP. Постоји програмски језик Larvel, он је изведен из PHP и надограђен као помоћ програмерима за лакше куцање кода. Ако корисник жели да користи PHP постоји доста комјунити верзија које корисник може да скине као што су WAMP, XAMP[12].



***Слика 14**. Функција за слање мејла*

Помћу ове функције прослеђујемо параметре за слање меила. У параметар host стављамо адресу нашег сервера, у параметар порт стављамо параметар нашег porta.

Параметар username/ password нам служе за нашу аутентификацију на наш мејл. Кад се улогујемо на наш мејл тада је могуће слање мејл sender- u преко интерфејса.

[12] https://leanpub.com/php-pandas/read

5.3 JavaScript(JS)

Овај програмски језик помаже функционалностима нашег сајта. Да бисмо користили JavaScript функције потребно је да у нашем HTML коду убацимо следећи код[13]:

```
<script src="myscripts.js"></script>
```

***Слика 15**. Повезивање JavaScript-а са HTML-ом*

У овом коду видимо да смо фајл назвали myscript.js и да стоји одмах поред наше странице. Постоји могућност да у JavaScript убацимо JQuery који нам олакшава брзи унос кода.

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

***Слика 16**. Повезивање JQuery-а са JavaScript-ом*

JavaScript програмски језик има налик објекту оријентисаном програмирању и ако нема никакве везе са њим и има налик на Java програмски језик, он ипак потиче од програмског језика C. Brendi Ajax је развио JavaScript програмски језик. У овом програмском језику постоји функција која прослеђује податке са корисничког интерфејса на PHP програмског језика.

[13] https://www.w3schools.com/js/DEFAULT.asp

```
<html>
<head>
<script>
function showHint(str) {
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  } else {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("txtHint").innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET", "gethint.php?q=" + str, true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>
```

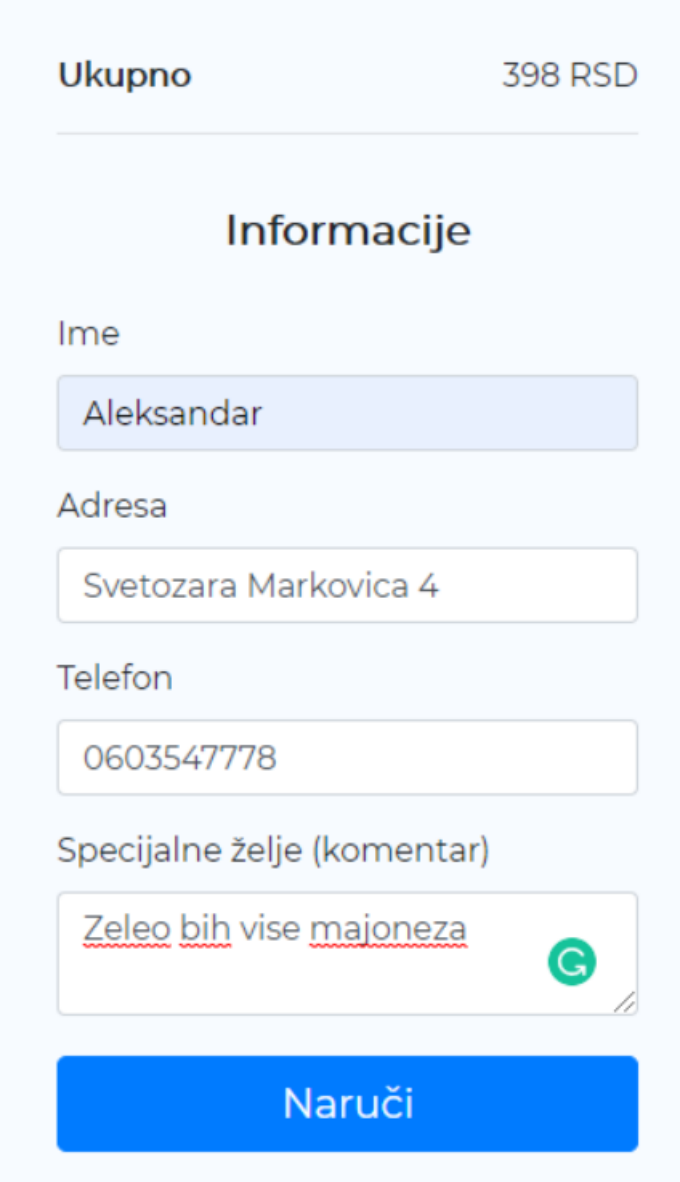
***Слика 17**. Ajax функција*

JavaScript је Microsoft усвијио 1996 године и убацио у свој претраживач исте године је на серверској страни убачен овај језик. JavaScript језик је могуће убацити у HTML код и тако динамички обрадити интерфејс.

```
var scrolltoOffset = $('#header').outerHeight() - 1;
$(document).on('click', '.nav-menu a, .mobile-nav a, .scrollto', function(e) {
  if (location.pathname.replace(/^\//, '') == this.pathname.replace(/^\//, '') && location.hostname == this.hostname) {
    var target = $(this.hash);
    if (target.length) {
      e.preventDefault();

      var scrollto = target.offset().top - scrolltoOffset;

      if ($(this).attr("href") == '#header') {
        scrollto = 0;
      }

      $('html, body').animate({
        scrollTop: scrollto
      }, 1500, 'easeInOutExpo');

      if ($(this).parents('.nav-menu, .mobile-nav').length) {
        $('.nav-menu .active, .mobile-nav .active').removeClass('active');
        $(this).closest('li').addClass('active');
      }

      if ($('body').hasClass('mobile-nav-active')) {
        $('body').removeClass('mobile-nav-active');
        $('.mobile-nav-overly').fadeOut();
      }
      return false;
    }
  }
});
```

***Слика 18**. JavaScript код*

Овај код писан у JS- у нам омугућава да када корисник користи мањи монитор(таблет, телефон) наш приказ буде „respon“.



Слика 19. *Приказ програма на мобилном уређају*

5.4.CSS

CSS је скраћеница од „ Cascading Style Sheets “, односно у нашем преводу стилски језик. Он дефинише стилове који дизајнирају изглед HTML елемената, односно фонта, боје, позадине, размаке[14].

Данас не можемо да замислимо веб сајтове и апликације без CSS. Без њега не би имали лепо дизајниране веб сајтове.

Такође данас CSS нам омогућава да можемо на свим уређајима да прегледамо лепо веб сајтове, прилагођени су свим резолуцијама екрана.

Сви ти стилови се налазе у “Table style”, односно пишу се у заглављу HTML документа или се већ само стављају на самим елементима. CSS документи се спремају под екстензијом .css

Овакви екстерни стилови не само да вам штеде време при раду, већ и при редизајнирању самог HTML документа.

Елементи у CSS- у се састоје од селектора и описног блока.

Селектор нам служи да помоћу ID- а који смо сетовали у HTML-u у CSS- у имамо могућност измене тог елемента.

Описни блок нам служи да када се у CSS-у мења одређен елемент селектујемо га ID -ем и витичастим заградама

```
/*-------------------------------------------------------------
  # General
  -------------------------------------------------------------*/
body {
  font-family: "Open Sans", sans-serif;
  background: #0c0b09;
  color: #fff;
}

a {
  color: #cda45e;
}

a:hover {
  color: #d9ba85;
  text-decoration: none;
}

h1, h2, h3, h4, h5, h6 {
  font-family: "Playfair Display", serif;
}
```

***Слика 20**. CSS код*

Овај код нам показује како смо “body” тело сајта дизајнирали..

[14] https://sr.wikipedia.org/sr-ec/ Каскадни_стилски_листови

Параметар „Font-family“ даје могућност целом фонту да користи одређен фонт.

Параметар Backgound даје могућност да цео сајт користи одређену позадину.

Параметар Color нам даје могућност да цео сајт користи одређену боју.

5.5 Bootstrap

Bootstrap је најпопуларнији фрејмворк који служи за праљење сајтова и апликација. Он стоји између серверске стране и корисника, односно он је интерфејс између њих[15].

Bootstrap је направљен од стране два програмера Twitter-а Марк Отоа и Џејкоба Тронтона.

Њега можете да скинете бесплатно и у преузетом пакету наћи ћете три посебна фолдера, сваки од њих има посебну намену. Један од фолдера служи за CSS датотеке, други за JavaScript датотеке, док трећи служи за фонтове. Унутар сваког од ових фолдера налазе се различите варијанте које пружа Bootstrap. У CSS датотеци се налазе три различита фајла, од којих је пожељно да изаберете један који има екстензију .css. Док у фолдеру JavaScript можемо наћи прикључке компоненте у форми JQuery. У трећем фолдеру имамо фонтофе у којима можемо пронаћи бесплатне иконице које можете користити у изради свог пројекта.

Bootstrap пружа широк спектар компоненти које корисник може да користи у изради свог веб сајта, од дугмета све док табела и колона.

Bootstrap је од своје верзије 2.0 подржао све величине екрана, што значи да је прилагодив свим уређајима и да је потпуно динамичан. А у верзији 3.0 Bootstrap је усвојио да је прилагодив дизајн на мобилним уређајима постао стандардан.

***Слика 21**. BootStrap*

[15] https://getbootstrap.com/

5.6.Visual Studio Code

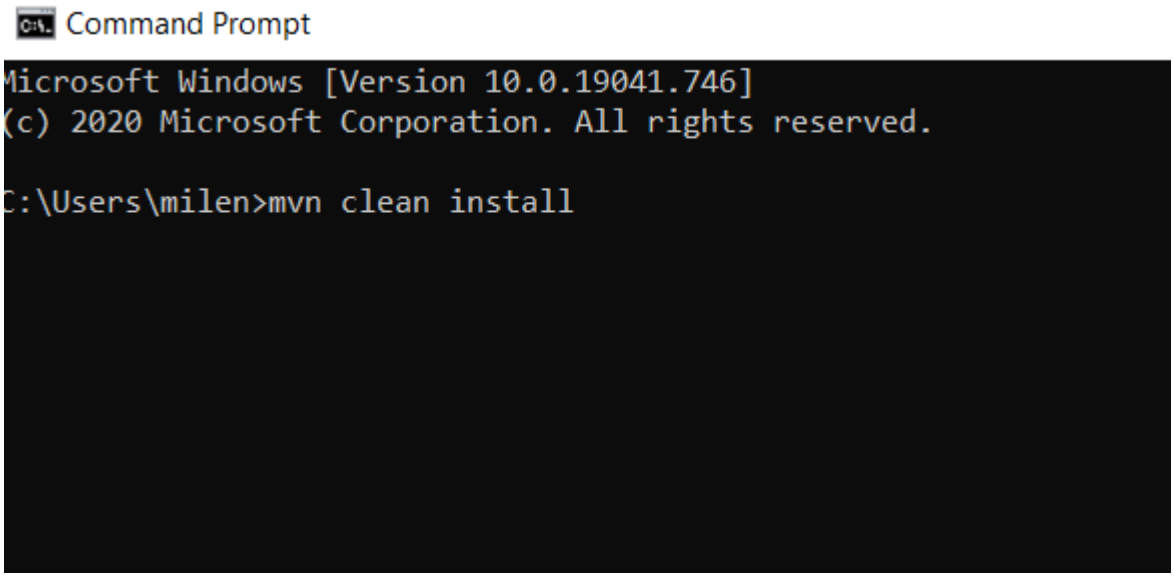
Visual Studio Code је призвео Microsoft и он служи сза сређивање текста. Нуди нам уређивање тесктова и кода на различите начине, односно Visual Studio Code је интегрисано развојно окружење. Веома је лак за употребу, поред тога бесплатан је и отишао је далеко испред својих предходника. По урађеним анкетама Visual Studio Code је дефинитивно најпопуларнији алат за уређивање[16].

Visual Studio Code подржава доста различитих програмских језика и који год програмски језик ви користите основне функције су идентичне. Постоје разне пречице на самој тастатури корисника у Visual Studio Code- у. Неки од програмских језика које он подржава су Java, Visual Basic, C#, Jason, C++…

Екстензије у Visual Studio Code- у су бесплатне и већи део екстензија коју нађете су "open sorce". Екстензије служе за управљање базе података и уклањање грешака.

Постоји могућност замене речи или више њих уз функцију „замени", као и функцију „пронађи" која вам омогућава да пронађете кључне речи у самом коду Visual Studio Code.

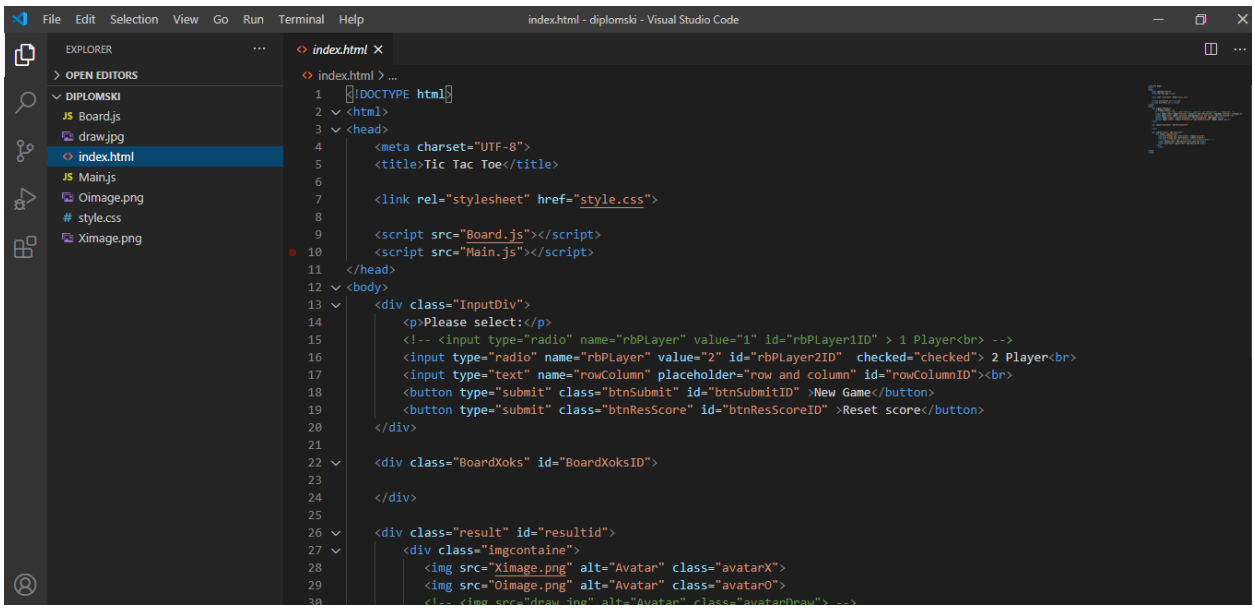
Visual Studio Code вам омогућава повезивање са „maven" пројектима. Потребно је инсталирати „maven" на рачунар како бисмо спојили пројекта са „maven-ом". Позивамо следећу функцију да би се „maven" инсталирао, у командној линији позивамо следећи код:



***Слика 22**. Покретање "maven"пројекта*

Добра ствар код овог програма је да иако сте у сред рада, а рачунар вам се угаси ваш код ће остати сачуван. Када је у питању изглед програма постоји доста тема и избор иконица за одређене екстензије фајлова.

[16] https://code.visualstudio.com/



Слика 23. *Приказ HTML кода*

Предности:

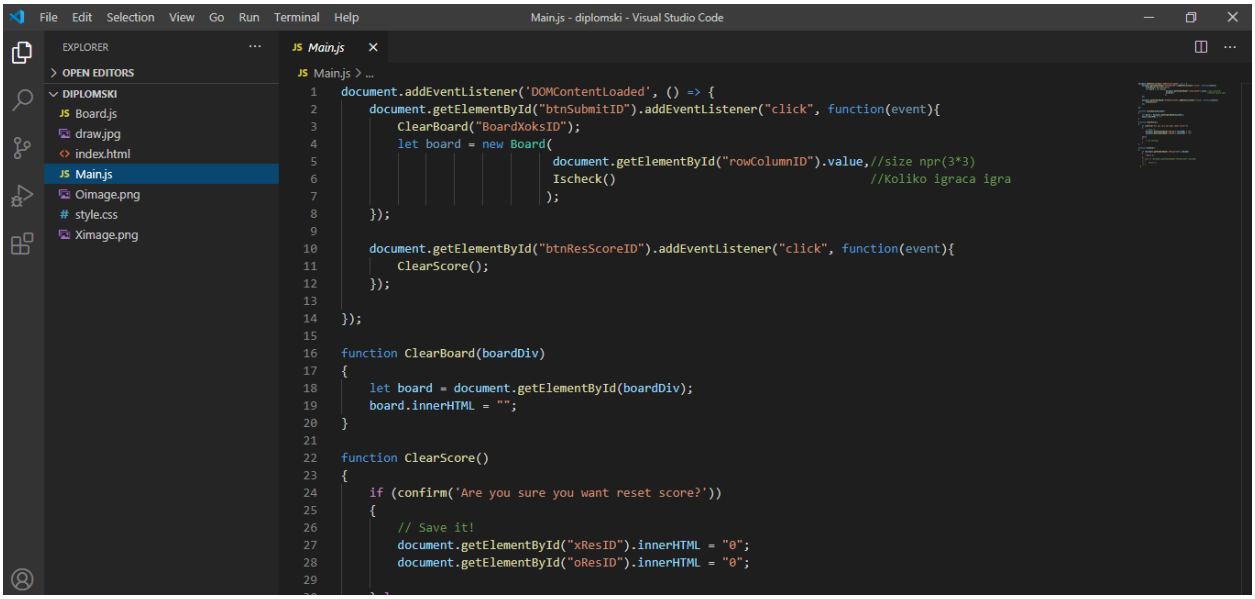
- *Брза и једноставна инсталација на лични рачунар*
- *Општеприхваћени стандард за програмирање у оперативном систему Windows*
- *Подршка за велики број језика и формата*
- *Брза интерпретација кода*
- *Лагана имплементација веб апликација*
- *Интегрирана подршка за надоградњу базе кода*

Недостатци:

- *Уређивач можда није погодан за почетнике*
- *Учитава рачунарски систем*

Кључне карактеристике :

- *Стварање апликација за разне платформе и оперативне системе*
- *Прикладан уређивач кода опремљен додатним "помагачима"*
- *Доступност алата за преуређивање кода*
- *Подршка за популарне програмске језике Python, C++ и друге*
- *Приступ 5000 проширења Visual Studio-а*
- *Програм је доступан потпуно бесплатно*
- *Могу се користити у комерцијалне и некомерцијалне сврхе*



***Слика 24**.Приказ JavaScript кода*

6. Виртуелна продавница – онлине наручивање из ресторана

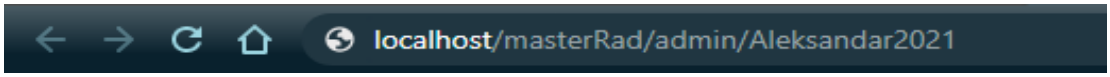
6.1. Админ панел

Проблем који постоји током ове ситуације са корон је проузроковао тешку ситуацију код у угоститељству. Самим тим сами угоститељи су морали да се преформулишу на онлине поручивање које је било доступно у ситуацију која задесила цео свет. Такође овај начин продаје помаже муштеријама да се лакше одлуче за који производ желе да поруче.

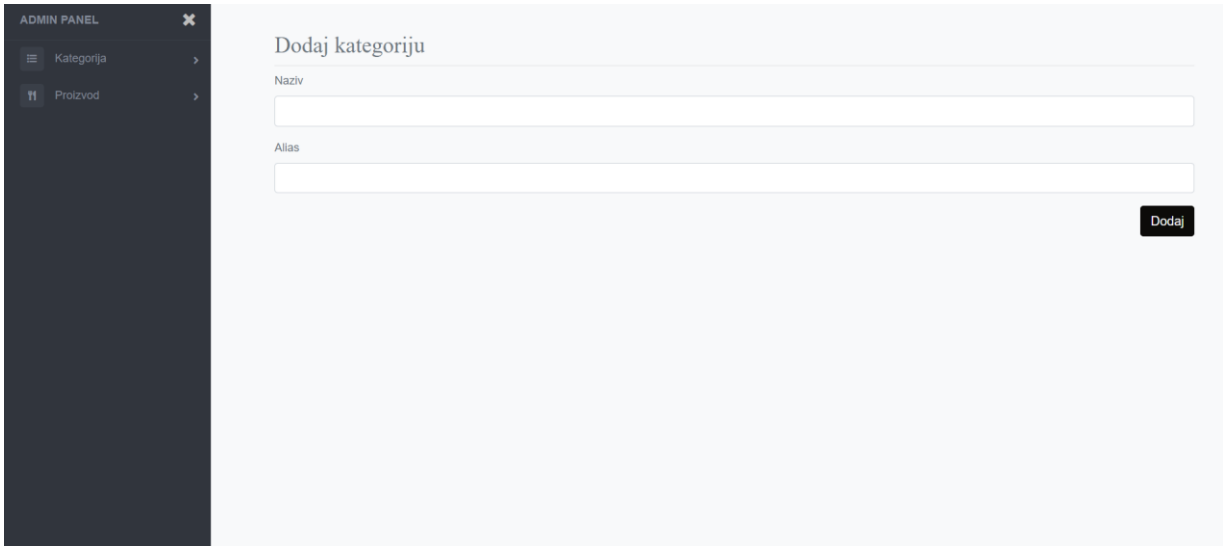
Пожељно је да веб продавница уз артикле које нуди има и своји рекламни сајт у једном, где може да обавести купце тј. клијенте да могу преко онлине менија који погледају дневно 200 људи да се рекламирају путем њега.

За почетак је потребно да се креирају база података и табеле „Category“ и табелу „Product“. То значи да се направи кориснички интерфејс где ће власник ресторана у свако доба дана моћи да унесе, обрише или промени артикал.

Админ има приступ тако што у назив странице унесе линк ресторана(у овом случају је то “localhost” зато што тренутно ресторан није онлине него на локалном серверу) плус „/admin/“ + шифру коју је власник одредио у овом слуцају је „Aleksandar2021“

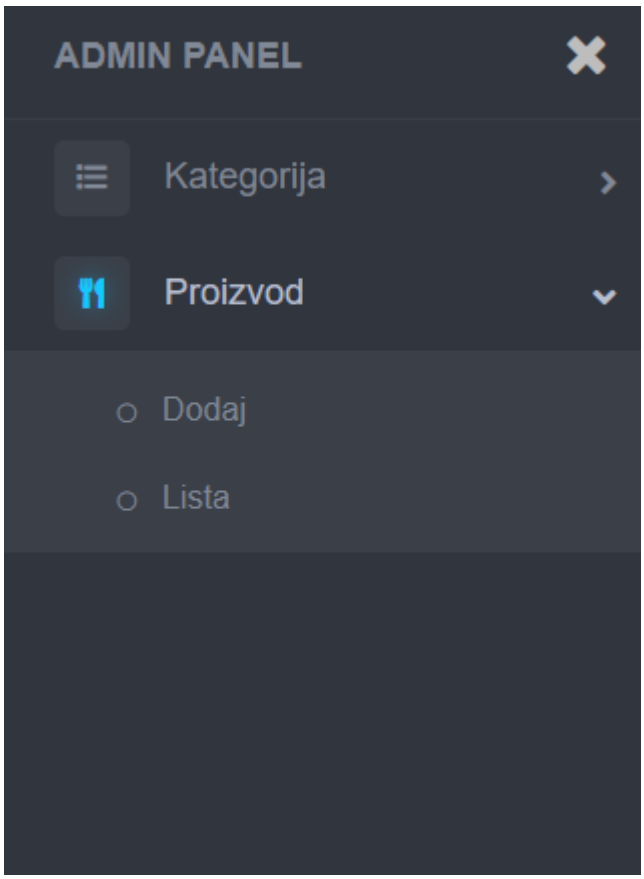


Ако је шифра тачна отвара нам се админ панел.



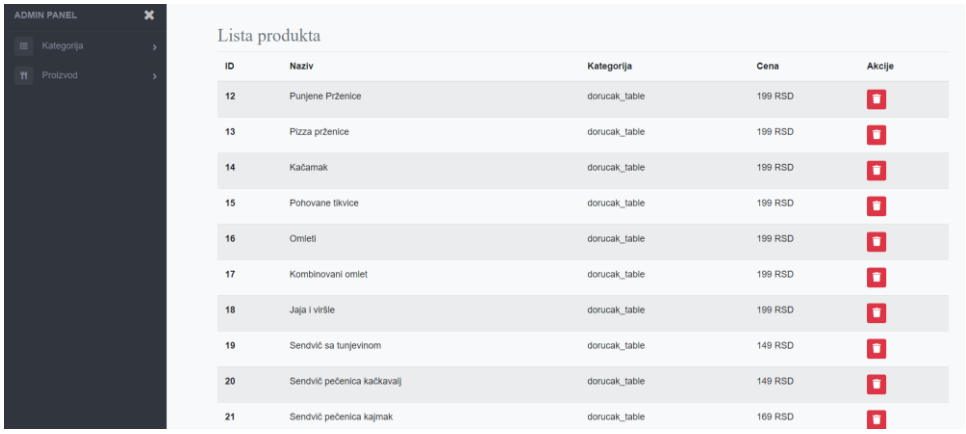
Слика 25*.Приказ админ панела*

У админском панелу можемо да додајемо категорију и артикле. У сваком тренутку можемо да прегледамо листу артикала.



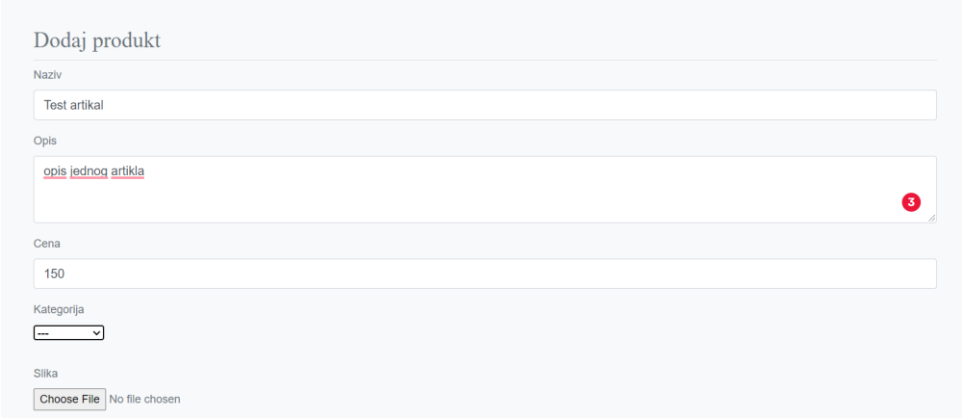
Слика 26.*Избор могућности у админ панелу*

Листа артикала једног ресторана.



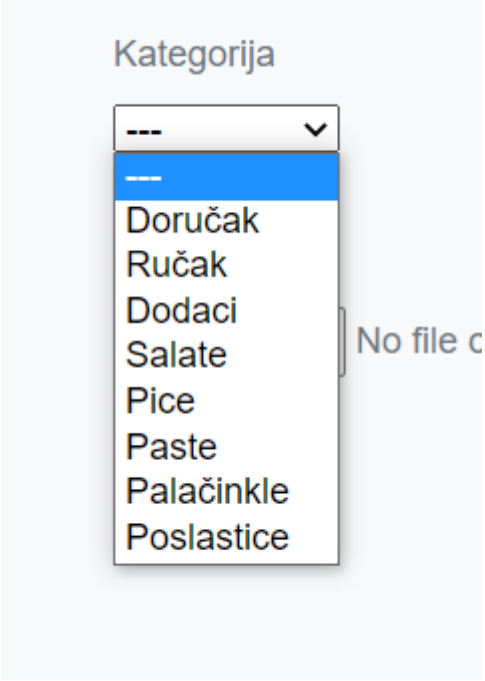
Слика 27.*Приказ артикла у админ панелу*

Приказ изгледа једног артикла.



***Слика 28**.Додавање артикла у админ панелу*

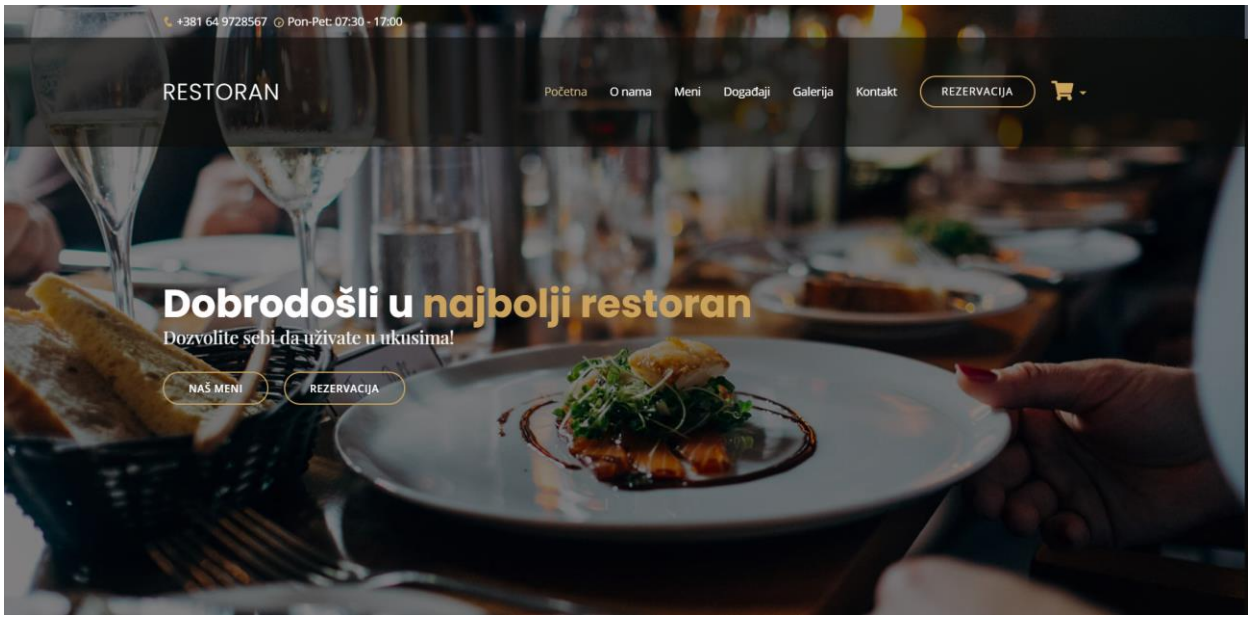
Кад кликнемо на категорију, преко php-а из базе повлачимо податке и приказујемо их у листи.



Слика 29.*Избор категорија у админ панелу*

И на крају имамо дугме које нам омогућава да додамо слику артикла.

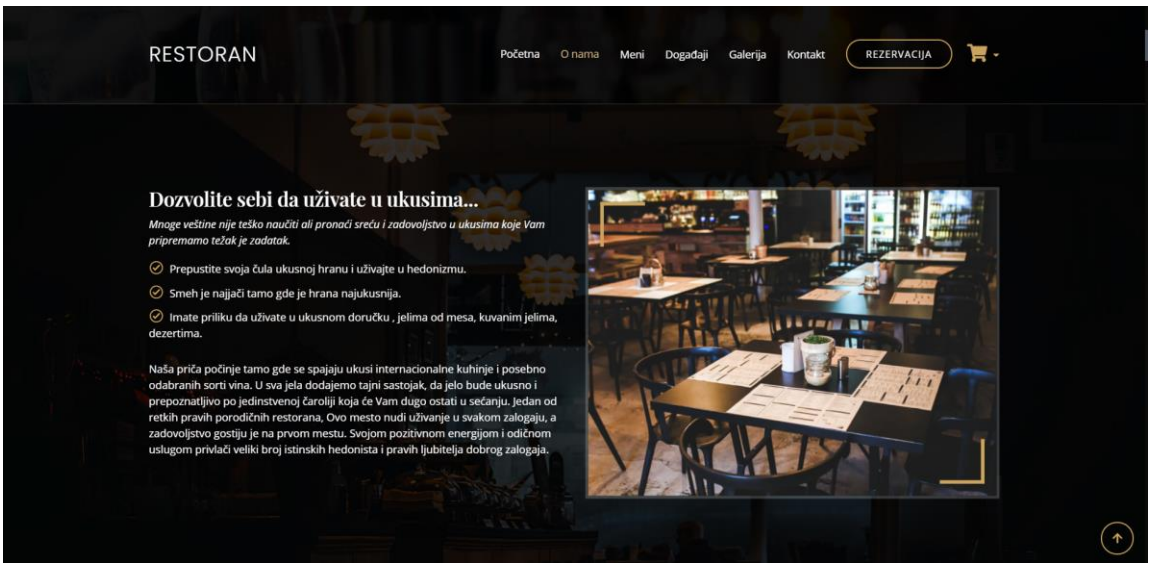
6.2. Веб продавница и рекламни сајт



***Слика 30**. Почетна страна веб продавнице*

Идеја ове веб продавнице је да у себи садржи рекламни део и део за одабир артикла и наручивање.

Продавница је осмишљена тако да буде све на једној страни (Landing страница). Кликом на нпр. “Мени” анимацијом се спушта на тај део сајта. Такође видимо и корпу. Подразумевано када уђемо на сајт корпа је празна. Имамо дугме “Резервација” која нас води на део где можемо да резервишемо сто.

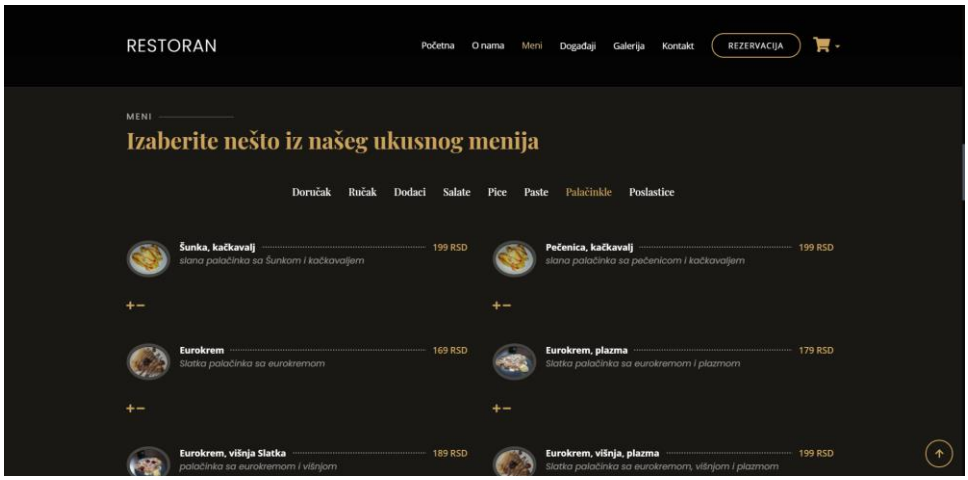


***Слика 31**. Део странице “О нама”*

Следећи део продавнице је “О нама”. Тај део говори о продавници ,потребне информације које корисник треба да зна.

Слика 32. *Део странице "Зашто изабрати наш ресторан"*

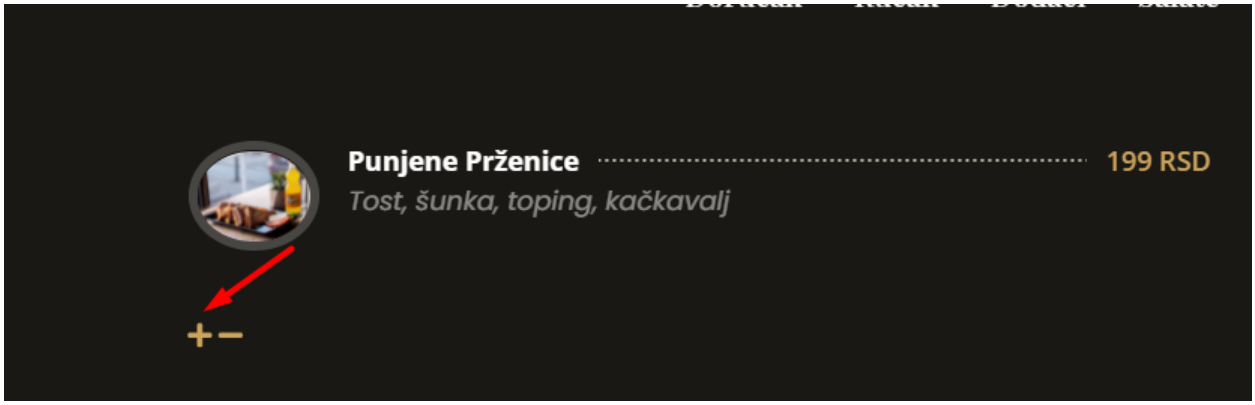
Део "Зашто изабрати наш ресторан" који је испод дела "О нама" нам говори по чему је веб продавница значајна и по чему се издвајамо од осталих.



Слика 33. *Део странице "Избор артикла"*

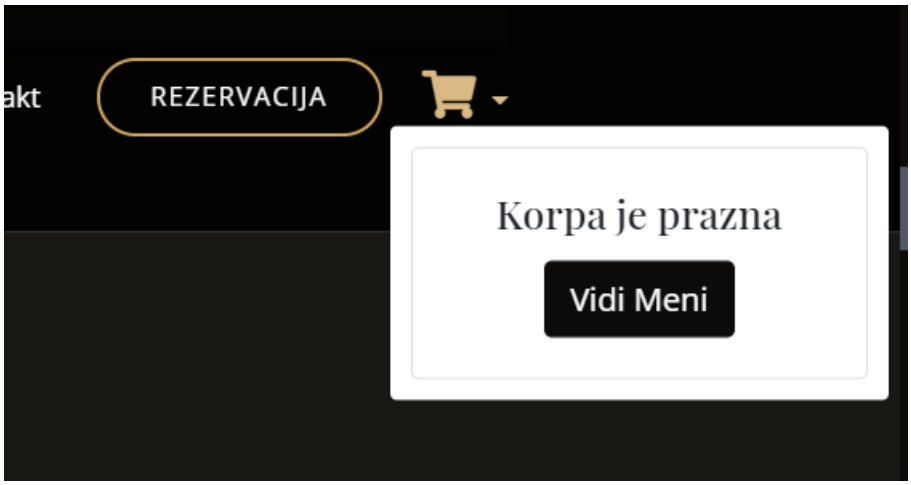
Слика 34. *Део странице "Избор артикла"*

Део где се приказује мени има више табова који су динамички и који могу да се уређују према потреби. У овом примеру је демонстриран јеловник и његове категорије. Админ има приступ као што је приказано на почетку да промени дода или обрише неки артикал или категорију.



Слика 35. *Део странице "Избор артикла(додавање у корпу)"*

Када желимо неки артикал да додамо у корпу испод сваког артикла имамо дугме „+ (додавање артикла у корпу)" И дугме "- (одузимање артикла из корпе)". Корпа је празна када се приступа сајту.

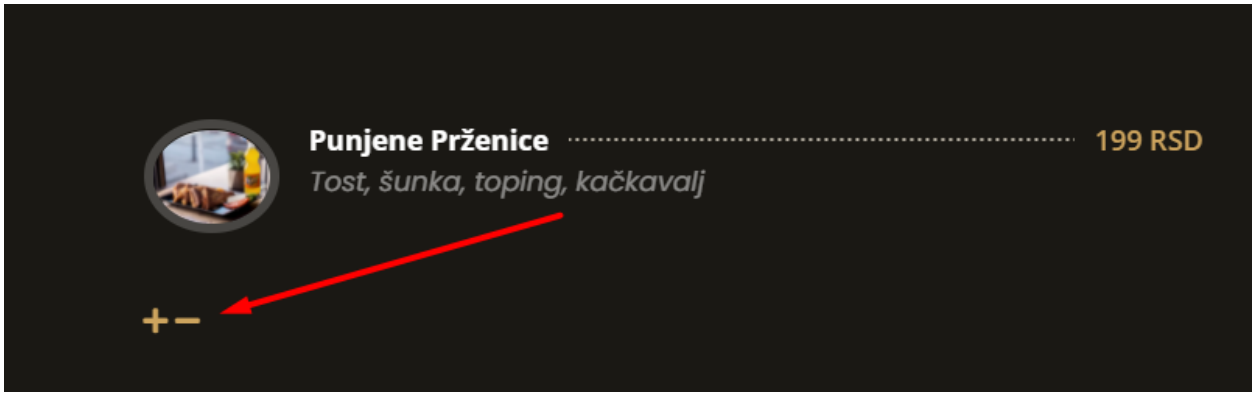


Слика 36. *Део странице "Приказ корпе"*

Кад додамо неки артикал у корпу то изгледа овако.

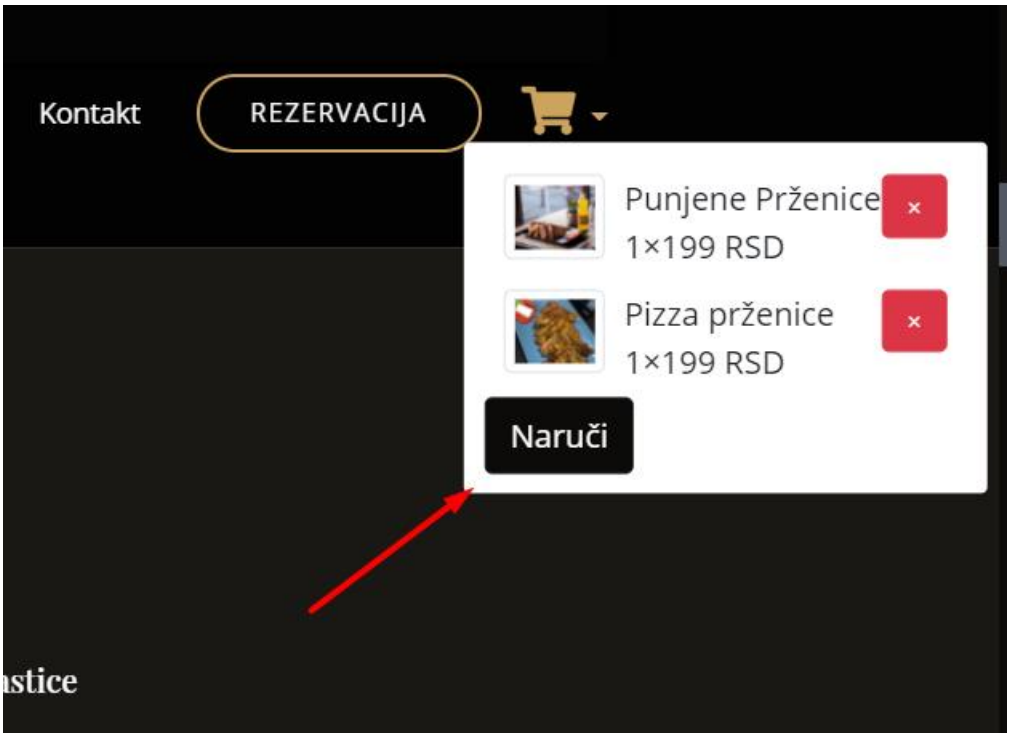
Слика 37. *Део странице "Приказ артикла у копри"*

Ако желимо да скинемо неки артикал из корпе можемо да дугме „x" а ако имамо више артикла у корпи нпр 3 пута пуњене прженице и хоћемо да скинемо на две прженице. Онда идемо на дугме „-".



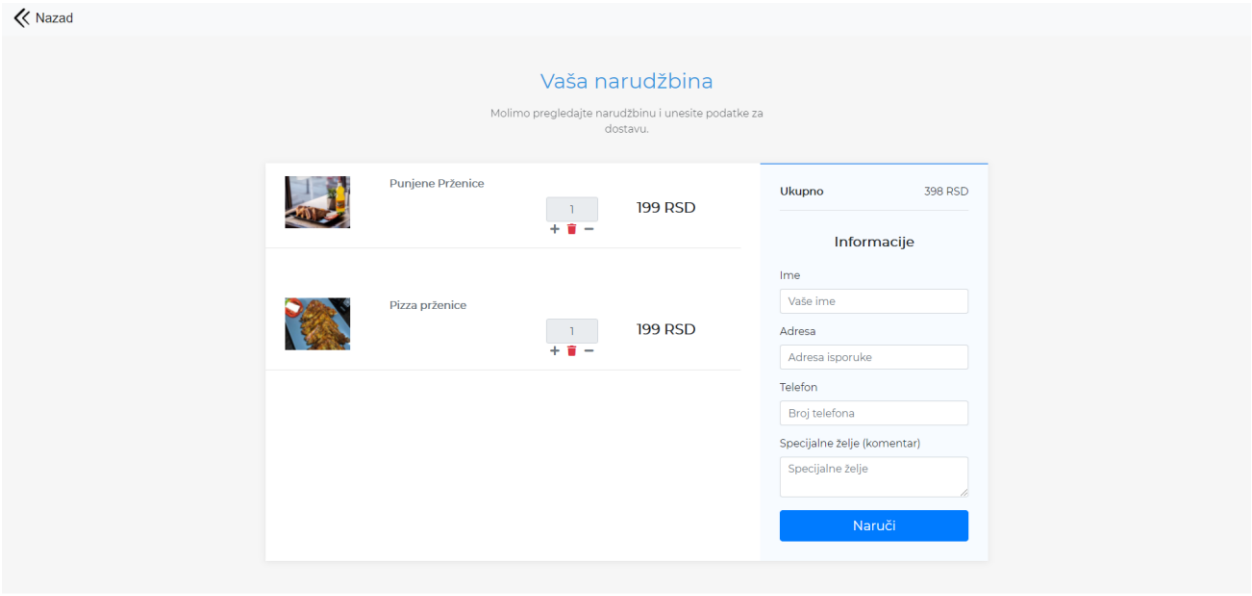
Слика 38. *Део странице "Одузимање артикла из корпе"*

Када завршимо додавање акртикла и желимо да наручимо идемо на дугме „наручи“.



Слика 39. *Део странице “Прелазак на страницу наручи”*

Притиском на дугме наручи корисника шаље на страницу где је потребно да унесе своје податке и да заврши поруџбину.



Слика 40. *Део странице “Ваша наруџбина”*

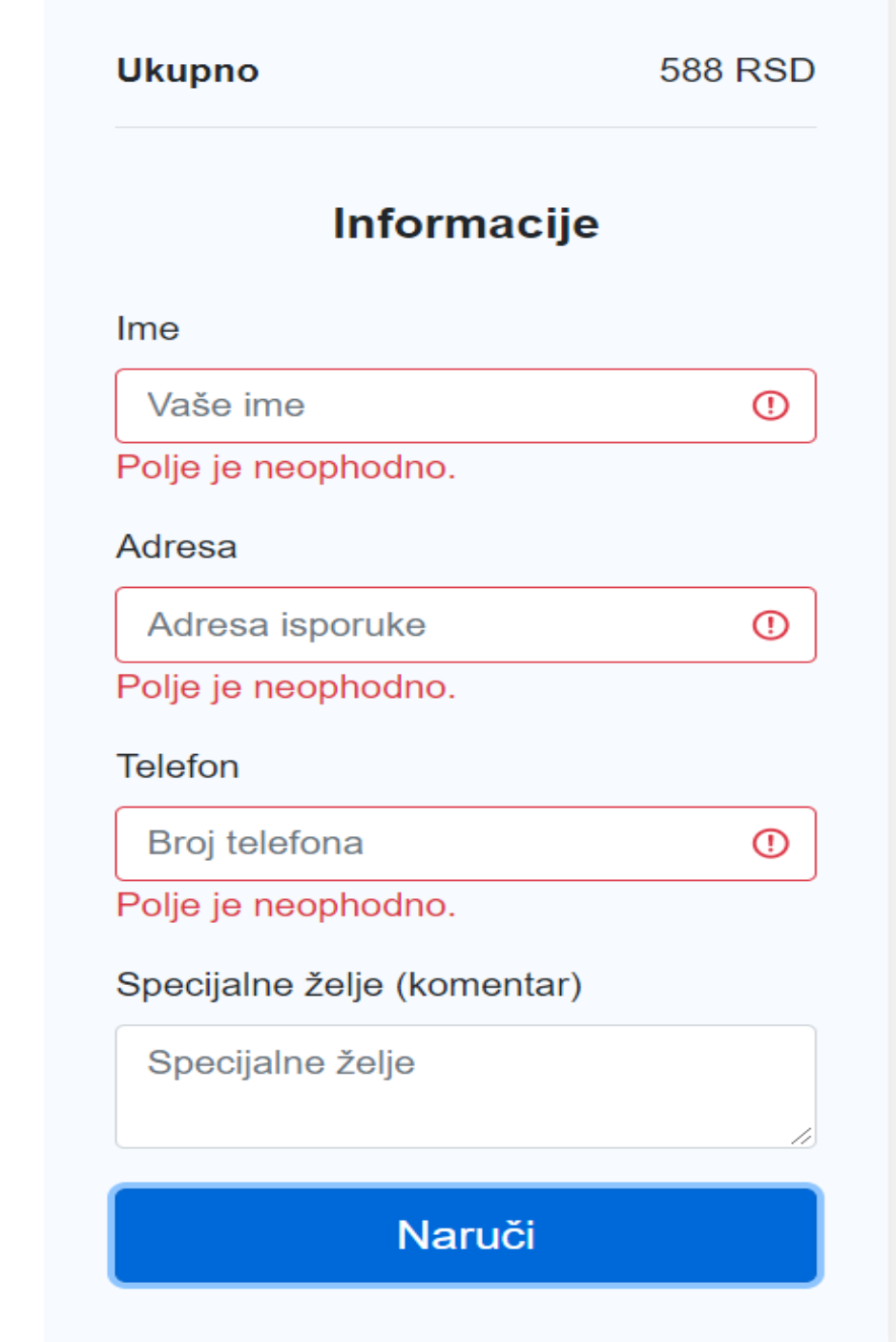
На страници „Ваша наруџбина“ имамо приказ артикла који су стављени у корпу имамо дугме назад које нас враћа на страницу да додамо још артикла ако желимо, и имамо приказ табеле са информацијама које морамо да упунимо. Такође имамо укупан збир колико ће нас коштати.

На листи где су артикли имамо три дугмета и једно поље које нам исписује број наручених артикла. Прво дугме нам омогућава да додамо још тог артикла следеће дугме нам омогућава да у потпуности избришемо артикал и последње дугме нам омогућава да смањимо за један артикал.

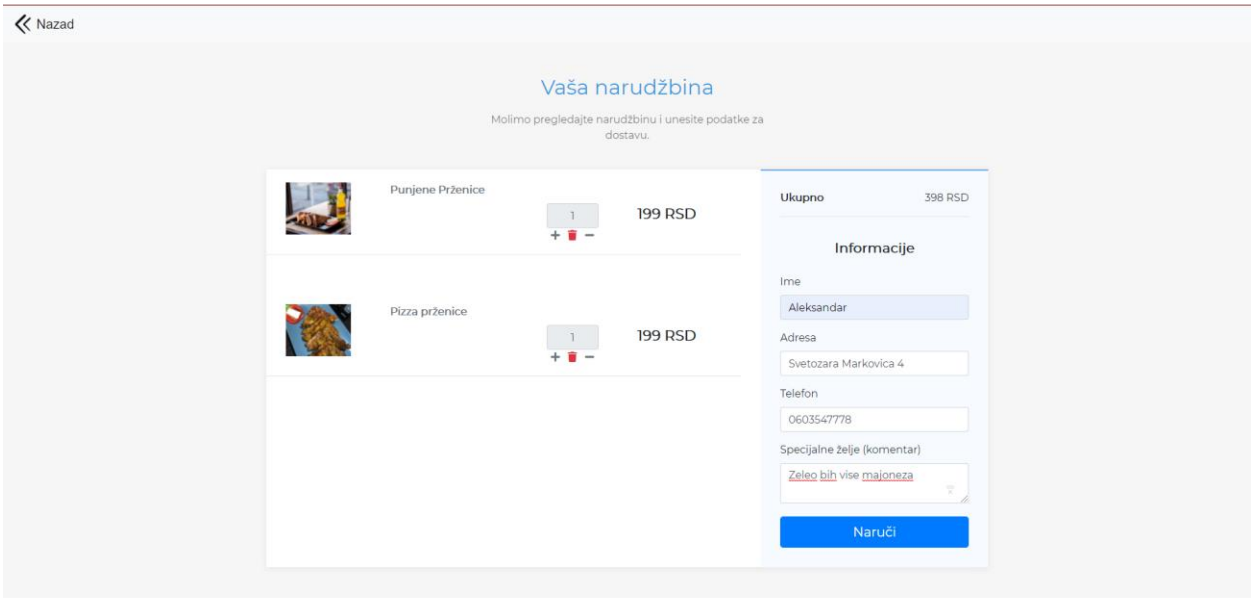
Потребни подаци за наручивање су:

- Име
- Презиме
- Телефон
- Специјална жеља(ово поље није обавезно)

Ако желимо да наручимо, а нисмо унели податке систем ће приказати следеће:

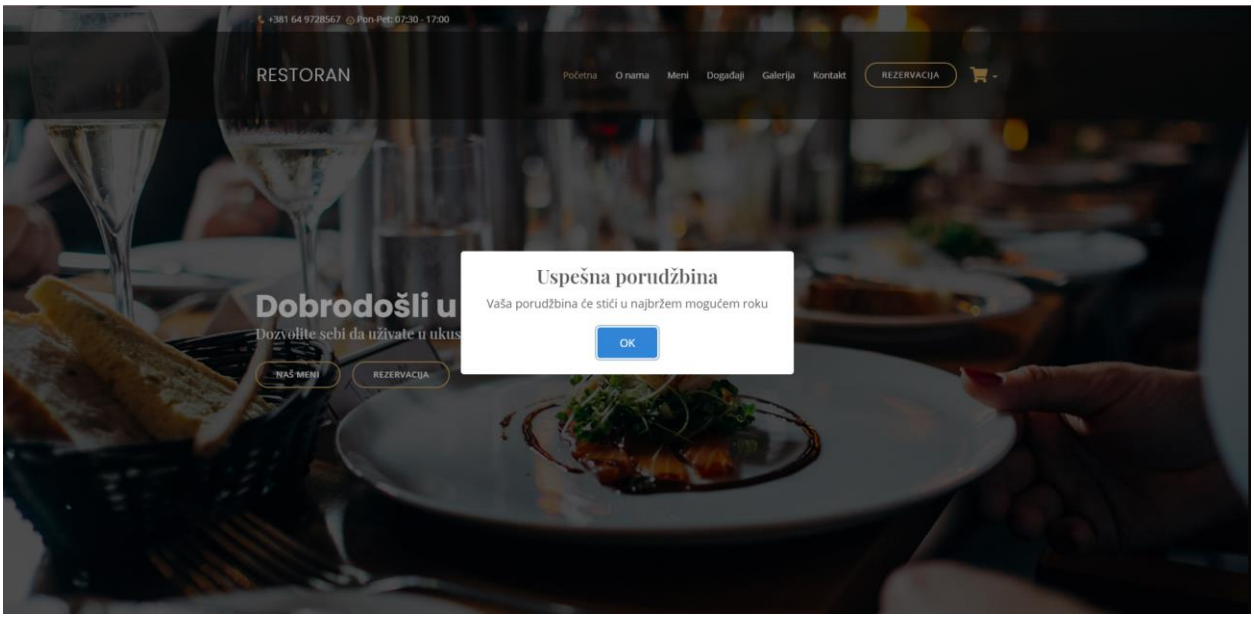


Слика 41. *Део странице "Неопходни подаци"*



Слика 42. *Део странице "Попуњена наруџбина"*

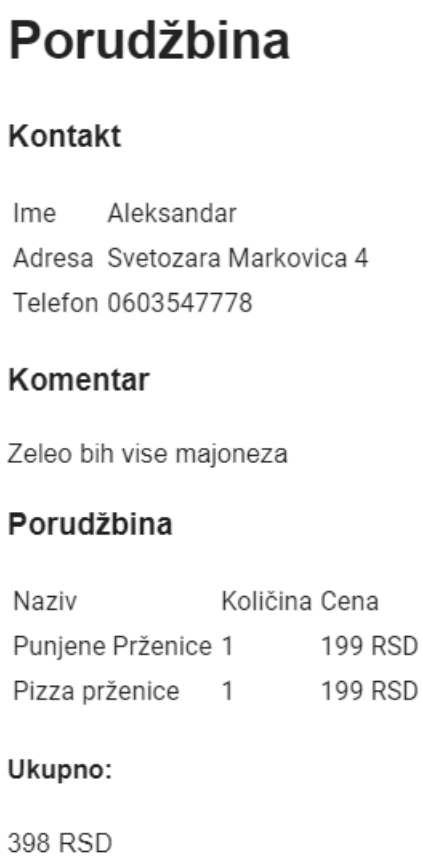
Ако је корисник унео добре податке као на слици И наручи систем ће проследити те податке на сервер и послати на маил људима који раде доставу.



Слика 43. *Део странице "Наруџбина успешно поручена"*

Ако је све прошло добро тј. поруџбина је успешно прослеђена систем ће избацити следећу поруку.

Порука која стиже запосленима који раде доставу изгледа овако.



Porudžbina

Kontakt

Ime	Aleksandar
Adresa	Svetozara Markovica 4
Telefon	0603547778

Komentar

Zeleo bih vise majoneza

Porudžbina

Naziv	Količina	Cena
Punjene Prženice	1	199 RSD
Pizza prženice	1	199 RSD

Ukupno:

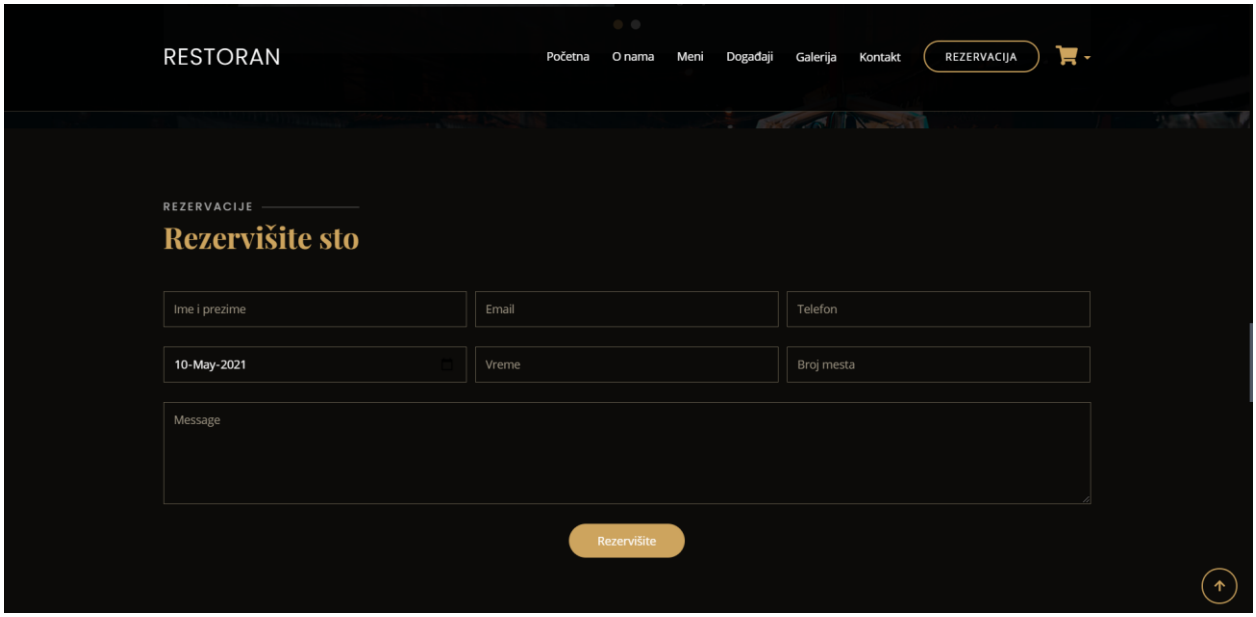
398 RSD

Слика 44. *Део странице "Наруџбина успешно поручена"*

Поруџбина коју је корисник креирао има све потребне информације и формата је HTML.

Достављач и људи који раде у кухињи знају тачно шта треба да припреме и колико, знају адресу и број телефона. Јако је битно да корисник у пар кликова може да наручи храну да то не буде компликован поступак.

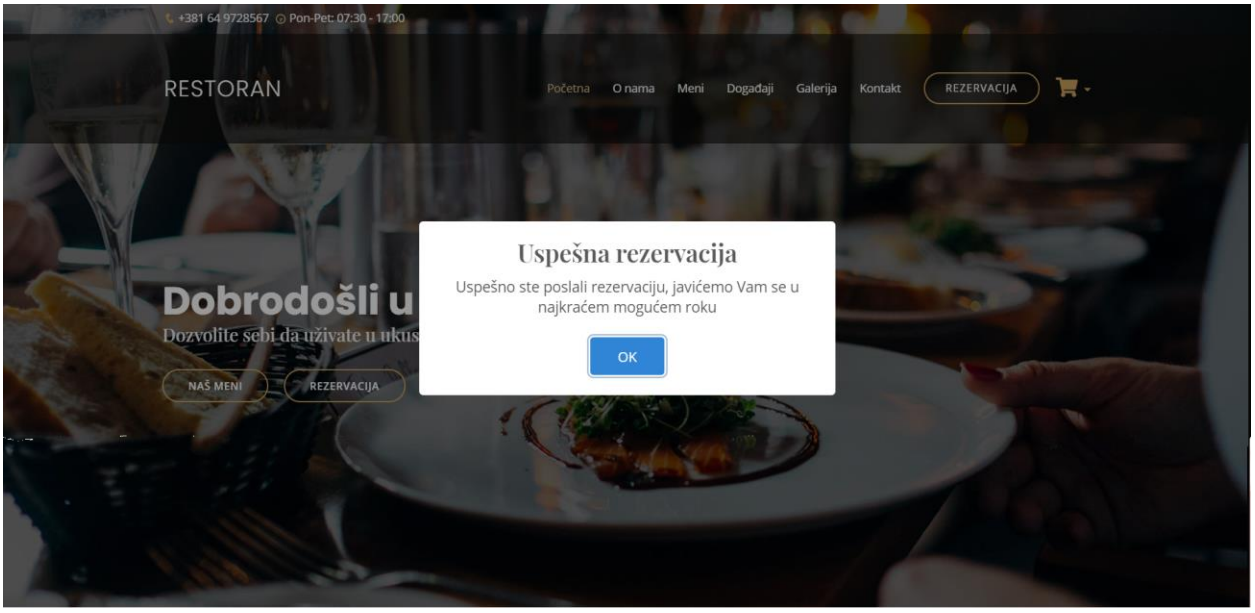
Корисник има могућност резервисања стола у одређено време и за одређен број особа.



Слика 45. *Део странице "Резервација стола"*

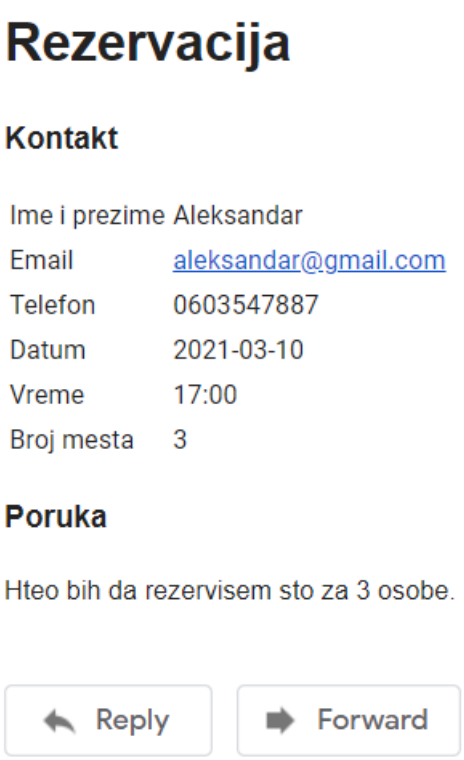
Подразумевано је датум буде стављен на данашњи дан али ми можемо да променимо кад нама одговара. Потребно је да се упише име,емаил, телефон,време и датум и за колико особа желимо да резервишемо и додатку поруку ако желимо.

Ако резервација прође софтвер ће исписати следећу поруку.



Слика 46. *Део странице "Успешна резервација стола"*

На маил тог ресторана ће стићи порука која изгледа овако.



Rezervacija

Kontakt

Ime i prezime	Aleksandar
Email	aleksandar@gmail.com
Telefon	0603547887
Datum	2021-03-10
Vreme	17:00
Broj mesta	3

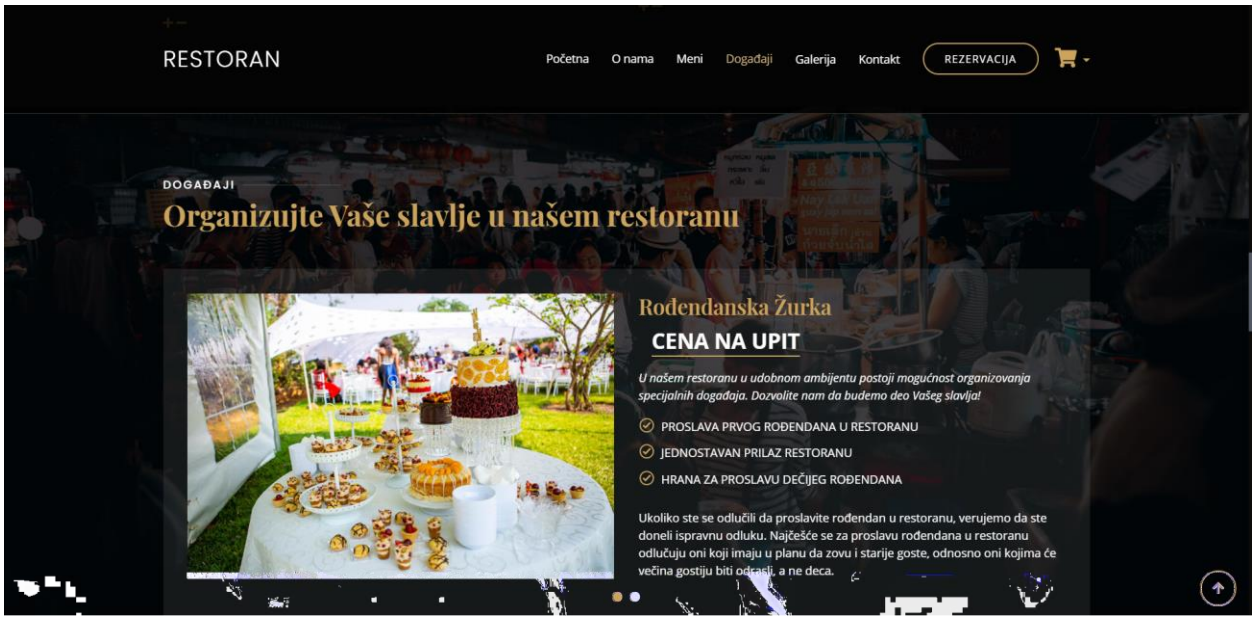
Poruka

Hteo bih da rezervisem sto za 3 osobe.

Reply Forward

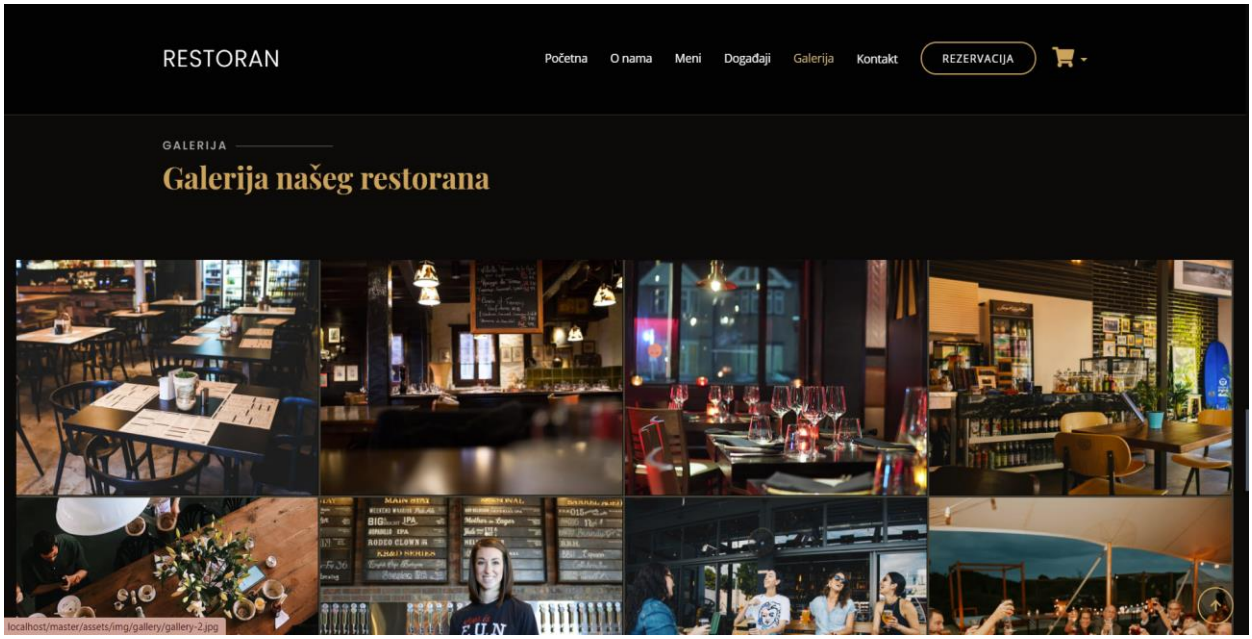
Слика 47. *Део странице "Порука на маил-у за резервацију стола"*

Део који је рекламни после резервација је део за промоцију догађаја. Промовишемо славља у ресторану где могу клијенти да прославе своје значајне догађаје са својим најмилијим.



Слика 48. *Део странице "Догађаји"*

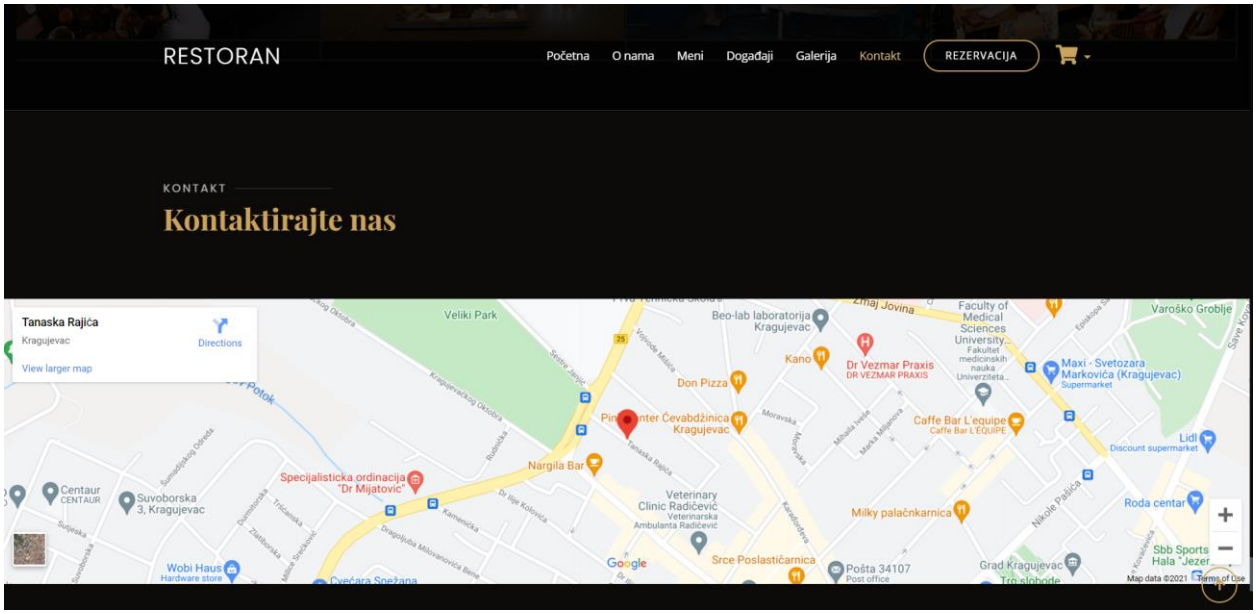
Секција за галерију слика је убачена испод секције за догађаје. Ова секција омогућава кориснику да погледа амбијент где ће јести или правити славље. Ове слике се убацују при прављењу веб продавнице и могуће је мењати само у коду. У некој наредној верзији биће да се мења у администском панелу.



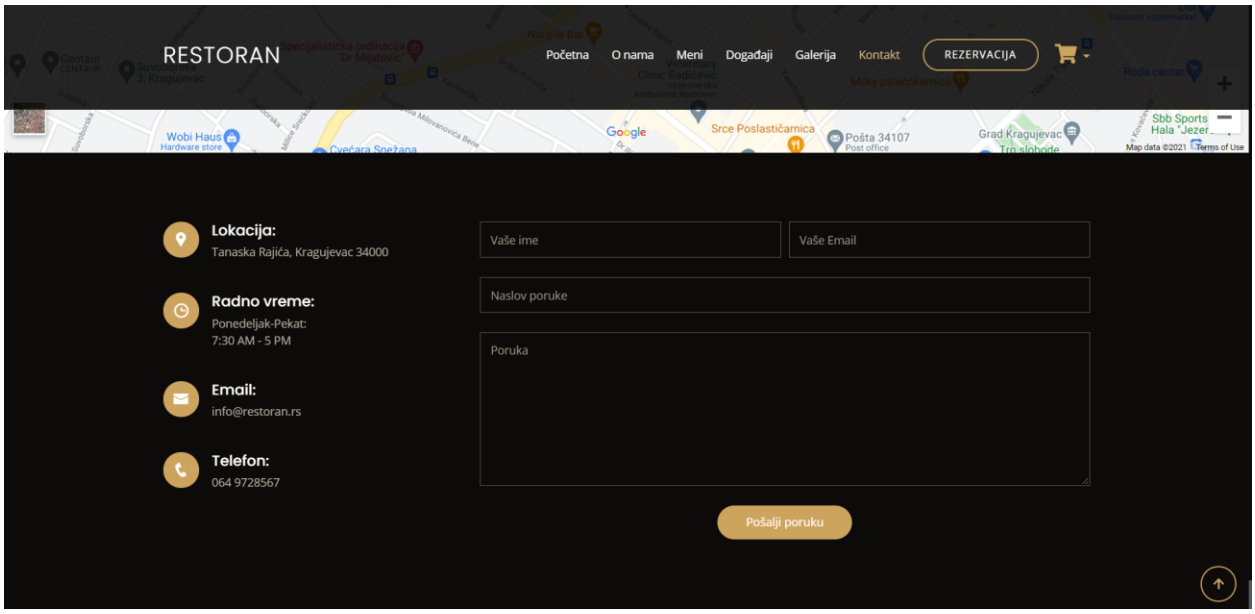
Слика 49. *Део странице "Галерија"*

Слика 50. *Део странице "Галерија, приказ једне слике"*

И последња секција је „Контакт“, ту корисник може да се информише о телефону, локацији, и маил-у.Такође има могућност да пошаље поруку за додатне информације путем маил-а.



Слика 51. *Део странице “Контакт”*



Слика 52. *Део странице “Питајте нас”*

7. ЗАКЉУЧАК

Развој интернета и нових технологија, тј познатих као развој апликација за брже и лакше пословање вашег бизниса, дизајн саме апликације представља развој софтверског производа. Сам назив развој софтвера значи у ширем смислу да изворни код се користи за рачунарско програмирање.

У раду је приказан процес развоја веб апликације за онлине наручивање хране. Обухваћен је развој апликације од саме спецификације захтева и анализе семантике рада, па све до креирања софтвера.

Софтвер има више корака, почевши од дизајна жељеног софтвера, па све до коначног производа. Софтвер може да укључује ново развијање, одржавање, мењање дизајна или било коју другу активност која доводи до бољег производа. У имплементацији самог софтвера спада не само одржавање, већ и упознавање проблема који има ваш клијент.

Једном када је представљен јавности, технологија отвореног кода постала је саставни део сваког алата за сваки развој. Јака мрежа знања гради моћну заједницу која заједнички користи снагу својих појединаца. Заједнице са отвореним изворима, функционишу по сопственим законима и знају предности у односу на конкуренцију, како би интеграцију подигли на виши ниво.

У овом завршном раду је коришћено доста програмских језика и помоћних алата како би добили једну веб продавницу. Програм је заменио папир и оловку за лакшу и бржу прегледност артикла.

Током израде овог софтвера дошло је до доста нових сазнања , као што је сама архитектура и конкретно програмирање. Стечено је и ново знање у раду са базом података, као и доста скраћене линије у самом коду, што нас доводи до оптималног решења у самој изради овог програма. PHP спада у објектно оријентисане програмирање, које омогућава креирање компоненти за вишеструко коришћење у разним корисничким програмима и самим тим нам је помогло да развијемо веб софтвер за онлине поручивање тј. веб продавницу.

Након свега наведеног може се закључити и видети директна примена и предности које једна веб продавница омогућава.

Долази се до закључка да су овакви системи данас доведени до савршенства уколико оно постоји, јер у само пар кликова је могуће наручити храну или резервисати сто.

8. ЛИТЕРАТУРА

Књиге и научни радови:

1. George Sepherd, Microsoft ASP.NET 4 Step by Step, Redmond – Washington 2010. Bartholomew, D., 2015, Getting Started with MariaDB, Packt Publishing, Birmingham.
2. Ivana Stojanović, MR - Elektronska trgovina i kupovina putem Interneta u Srbiji, 2010. Beograd
3. Лазаревић, Б.,2003, Базе података, ФОН Београд.
4. Павловић-Лажетић,Г.,2000,Основе релационих база података, Математички факултет, Београд.
5. Веиновић, М., Шимић, Г., Јевремовић, А., Франц, И., 2013, Базе података, Универзитет Сингидунум, Београд.
6. Веиновић, М., Шимић, Г., 2010, Увод у базе података, Универзитет Сингидунум, Београд.
7. Steve Prettyman(prevod-Slavica Prudkov), Naučite PHP 7, 2016. Beograd

Интернет извори:

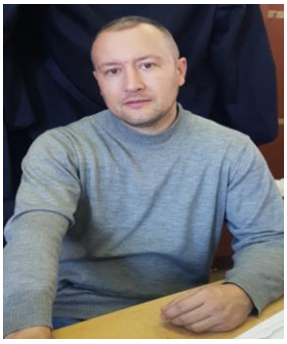
9. https://www.phpmyadmin.net/ (Прегледано: 18.03.2021).

10 https://docs.phpmyadmin.net/en/latest/ (Прегледано: 18.03.2021).

11.https://code.visualstudio.com/ (Прегледано: 22.03.2021).

12.https://download.tutoriali.org/Tutorials/Web_Aplikacije/web_aplikacije.pdf (Прегледано: 22.03.2021).

13.http://www.vtsnis.edu.rs/predmeti_2012/klijent_server_sistemi/predavanja_2016/KS%20Predavanje%209%202016.pdf (Прегледано:22.03.2021).



Europass Curriculum Vitae

Personal information

First name(s) / Surname(s)	Aleksandar Glamoček
Address(es)	Srete Mladenovica No 3/1, 34000 Kragujevac, Republic of Serbia
Mobile	00 381 64 / 16-19-448
E-mail	sasarca@gmail.com
Nationality	Serbian
Date of birth	June 11th, 1981
Gender	Male

Work experience

Dates	September 25th, 2006
Occupation or position held	Electrical technician
Main activities and responsibilities	Reading of electro projects and implementation, elaboration of electro drawings, making specifications of electrical materials, ordering electro materials, electromagnetic wiring diagrams (single-pole diagrams)
Name and address of employer	Zastava arms Ltd Kragujevac, Kosovska Street No 4, 34000 Kragujevac, Republic of Serbia

Education and training

Dates	2015 – 2019
Title of qualification awarded	Menager
Principal subjects/occupational skills covered	Organizational and managerial skills in all business segments
Level in national or international classification	VII

Personal skills and competences

Mother tongue(s): Serbian

Other language(s)

Self-assessment	Understanding		Speaking		Writing
European level (*)	Listening	Reading	Spoken interaction	Spoken production	
English	B1	A2	A2	A2	A1

(*) Common European Framework of Reference for Languages

Social skills and competences: Communicative person, good adaptation to new environments and work responsibilities, voluntary blood donor

Organisational skills and competences: Excellent organizational and leadership skills, working in pressures

Computer skills and competences: Office package: Word, Excel, PowerPoint; WEB

Driving licence: Category C