Зборник радова Факултета техничких наука, Нови Сад

UDK: 621.38
DOI: https://doi.org/10.24867/22BE13Ninkovic

СОФТВЕРСКО РЕШЕЊЕ ИНТЕРНЕТ ПРОДАВНИЦЕ

A SOFTWARE SOLUTION OF A WEB SHOP

Катарина Нинковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом чланку описана је имплементација и рад интернет продавнице. Реч је о тролојном систему који укључује имплементацију сервера и клијента, као и креирање и управљање базом података. Пројекат представља приступ развијања модерних web апликација уз последње технологије и програмска окружења као што су Node, React, Redux и MongoDB. Подржана је евиденција купаца и поруџбина, динамичко прикупљање производа и осталих информација, администраторски простор за управљање сајтом и друге функционалности које покрива савремена интернет продавница.*

Кључне речи: *Web, Node, React, Redux, MongoDB*

Abstract – *This article describes implementation and functionality of a web shop. It is about a three-layer system that includes the implementation of a server and a client, as well as the creation and management of a database. Project presents the approach to development of modern web application including latest technologies such as Node, React, Redux and MongoDB. It supports notation of customers and orders, dynamical loading of products and rest of the information, admin space for site management and other functionalities that web shops cover nowadays.*

Keywords: *Web, Node, React, Redux, MongoDB*

1. УВОД

У данашње време процес *online* куповине или куповине преко интернета постао је општеприхваћен и заступљен у модерном друштву. Динамика живота данашњег човека често оставља без могућности да пролази кроз многобројне радње у потрази за оним што му је потребно, што поред напорног обилажења, укључује и велик губитак слободног времена, те овакав вид куповине представља олакшавајућу околност. Путем интернет продавнице (енг. *WebShop*) корисник је у могућности да прегледа све производе, одабере оне одговарајуће и "пошаље их у корпу", одакле ће даље моћи да изврши куповину.

У оквиру овог пројекта реализован је систем који се састоји од два основна блока. Први блок представља, серверску страну, колоквијално названу *Back-end*, која укључује серверску апликацију базирану на *Node.js* [1] платформи и њој специфичном *JavaScript* програмском језику, и базу података, која је у овом случају *MongoDB* [2]. Други блок обухвата клијентску страну, илити *Front-end* који садржи *React.js* [3] кориснички интерфејс, који је такође базиран на *JavaScript* програмском језику, али и на *HTML* и *CSS* језицима за креирање и дизајнирање *web* страница. Поред тога на "фронту" се користи *Local Storage* за складиштење података у претраживачу, те *Redux* [4] који контролише стања чувана у локалној бази података и *PayPal* као *third party* платни систем.

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

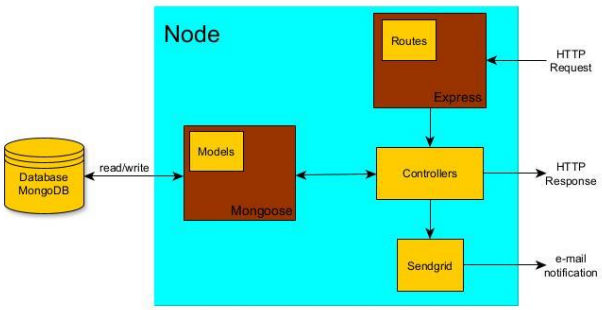
2. СИСТЕМ

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

2.1 Серверска страна

Сервер је сачињен од следећих ентитета:

- *Node.js* - за покретање сервера
- *MongoDB* - за управљање базом података
- *Express.js* - за рутирање
- *Sendgrid* - за слање *e-mail* порука
- Контролера - за обраду захтева



Слика 1. Блок шема серверске стране

Node.js

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

Уз *Node* долази и *Node Package Manager - npm*. У питању је алат који је задужен за добављање екстерних *Node.js* пакета, као и за складиштење и

проналажење истих. Инсталација нових пакета обавља се помоћу *npm* команди командне линије (терминала), чиме се пакети аутоматски преузимају и додају у листу зависности (енг. *Dependency*) у датотеку *package* са *JSON* екстензијом. Пакети који су коришћени на *back-end*-у су:

- *Mongoose* - за рад са *MongoDB*
- *Dotenv* - за учитавање променљивих из *env* датотеке у процесу
- *Express* - за рутирање сервера и обраду *HTTP* захтева
- *BCrypts* - за енкриптовање шифре
- *JSONWebToken* - за генерисање токена
- *Path* - за добијање *jpg* и *png*
- *Multer* - за учитавање датотека
- *Morgan* - за приказ *HTTP* zahteva ради анализе
- *Sendgrid* - за слање *e-mail* порука ка корисницима

MongoDB

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

Express.js

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

Након што сервер пошаље захтев или добије исти, потребно је проследити одговор. Да би било лакше разазнати врсте захтева и упита, креирају се путање или руте (енг. *Routes*) које су креиране за одређену крајњу тачку (енг. *Endpoint*). Изворна (енг. *Root*) тачка у фази развоја јесте http://localhost:5000/api. Оваква путања могућа је само за машину у локалној мрежи, док би за остале уређаје у мрежи на рути уместо *localhost* кључне речи била прослеђена конкретна *IP* адреса уређаја на којем се сервер заправо извршава. У продукцији би крајња тачка била назив *web* странице, која би била *host*-ована. Руте служе за прослеђивање захтева одговарајућим контролерима, који даље обављају сву потребну функционалност.

Контролери

Контролери функционишу тако што преко модела шаљу захтеве ка бази, креирају одређене упите у зависности од тога за шта су им потребни ти подаци и уколико испуњавају услове онда као одговор добијају жељене податке.

Уколико би се узео пример пријаве (енг. *Sign in*) корисника на *web* страници, први корак би био унос података (*e-mail* адреса и лозинка) на клијентској страни, након чега следи паковање унетих података у *HTTP* захтев ка рути на серверској страни http://localhost:5000/api/users/login који сервер преузима и потом прослеђује одговарајућем контролеру. Контролер даље за улогу има проверу да ли је корисник евидентиран у бази тако што шаље захтев ка бази, тражећи да прочита податке и изврши одређена испитивања као што је у овом случају постојање и поклапање унете *e-mail*_адресе и шифре. Уколико корисник постоји као одговор од базе добијамо податке везане за тог коринсика.

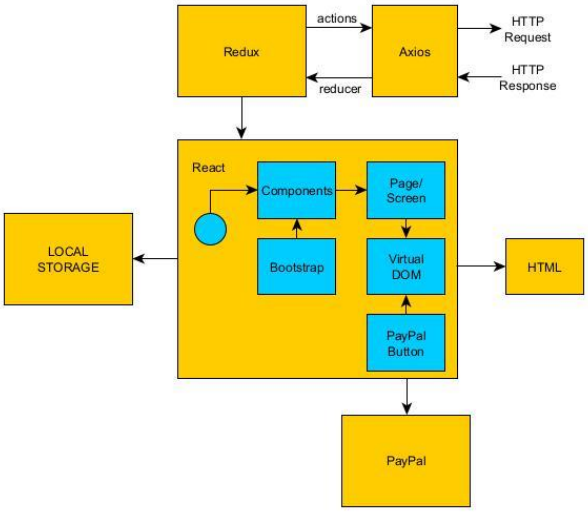
Sendgrid

У питању је екстерни (*third-party*) систем за реализацију слања *e-mail* порука ка корисницима од стране сервера. Ова платформа поседује подршку за *Node*, те се може увући и користити као остали пакети. Неопходно је креирати *developer* налог и *API* кључ како би слање порука било могуће. *Sendgrid* је једна од платформи која изузетно брине о безбедности, те је потребно испоштовати неке од њихових процедура и гарантовати доставу сигурног садржаја, а њихова техничка подршка редовно надгледа рад налога и сигурност *API* кључа. Поред примаоца, у заглављу поруке, потребно је проследити *e-mail* адресу пошиљаоца, коју је претходно потребно верификовати.

2.2 Клијентска страна

За развој корисничког интерфејса коришћена је *React* библиотека са својом *ES6 JavaScript* верзијом. Као и на *back-end* страни, за креирање пројекта и инсталацију пакета треће стране користи се *npm*. Кључни ентитети на *front-end* делу су:

- *React - JavaScript front-end framework*
- *Redux* - за контролу локалне базе података
- *Bootstrap* - за изглед корисничког интерфејса
- *PayPal* - систем за плаћање



Слика 2. Блок шема клијентске стране

React

Ова платформа функционише тако што се за сваки *HTML* елемент креира специјални *JavaScript* објекат

уз чију се помоћ врши контрола над *DOM* моделом сајта. Објекти могу бити елементи или компоненте (сложени елементи) и сваки од њих може укључити неке друге чиме се образује структура "стабла" у којем сваки чвор представља један објекат. Тако се креира виртуални *DOM* - идентична копија реалног модела, чија је улога упоређивање и ажурирање истог, који се користи за рендеровање странице. Уколико би дошло до промене стања неког од објеката виртуални *DOM* ће заправо ажурирати само њега, док остале неће дирати, за разлику од реалног модела који би ажурирао цело стабло чиме представља далеко спорију имплементацију.

React елементи се користе за описивање изгледа *HTML* елемената. За њихов опис користи се *JavaScript* синтаксно проширење *JSX* (енг. *JavaScript Syntax Extension*) који обезбеђује писање *HTML* структуре и *JavaScript* кода у истој датотеци. За управљање тим елементима користе се компоненте.

Redux

Redux је библиотека која се користи за управљање стањима. Стања из *Redux*-а могуће је само читати (енг. *Read-only*), док се за њихову промену и манипулацију користе акције и редуктори (енг. *Reducer*). Акција описује жељене измене, док редуктори те измене извршавају. Редукторске методе се не позивају директно него уз помоћ специјалне функције *dispatch* из *Redux* библиотеке.

Bootstrap

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

Pay Pal

Америчка компанија *Pay Pal* поседује дугу историју пословања у свету, један је од зачетника новчаних трансакција преко интернета, али и незаобилазни платни систем до данас. Уколико нека продавница жели да има велик домашај (енг. *Reach*) ка корисницима, *Pay Pal* је у том случају *must-have*. Њена интеграција се врши уз помоћ скрипте која се увлачи приликом учитавања странице, где је потребно проследити претходно креирани *developer*-ски *API* кључ. Увлачењем скрипте омогућено је убацивање неког од њених популарних дугмића који прослеђују корисника на њен прозор. Овим је сва одговорност у вези са сигурношћу трансакција пребачена на страну разрађеног платног система, те је ризик од малверзација сведен на статистичку грешку.

3. ФУНКЦИОНАЛНОСТИ

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

Почетна страна и странице производа

На почетној страници *web* апликације која је видљива свима, приказан је преглед свих производа. Наиме, приказани су основни подаци као што су цена производа, слика, назив и просечна оцена потрошача. Кликом на слику или назив продукта помоћу *React Link*-а отвара се нова страница са његовим детаљнијим описом и листом рецензија потрошача. Такође, на овој страници обезбеђено је и убацивање жељених производа у корпу, где су даље могућности блокиране све док се корисник не пријави. Сваки производ саржи поље количина (енг. *Quantity*), које купцу говори да ли га има на стању. Ажурирање стања омогућено је администратору о чему ће више речи бити у наставку.

Претрага производа

Како би кориснику било омогућено што лакше коришћење апликације, креирана је компонента *SearchBox*, која обезбеђује једноставну претрагу производа. Помоћу *Regex*-а обезбеђена је претрага где није неопходан унос идентичног назива већ сви производи који садрже тражене речи ће бити исфилтрирани и приказани на новој страници. Такође, обезбеђено је и игнорисање малих и великих слова, чиме је олакшана претрага тражених производа.

Регистрација и активација налога

Уколико би корисник желео да креира поруџбину неопходно је да буде пријављен. Као нови корисник први корак би била регистрација где би се унели тражени подаци као што су име, *e-mail* адреса и шифра. Након тога, на *e-mail* адресу стиже порука за активацију корисничког налога преко *Sendgrid*-а чијом потврдом, уласком на линк, се завршава процес регистрације.

Аутентикација

Како би се заштитили подаци, као што је, на пример, профил корисника креиран је *authMidleware* за аутентикацију. Заштита је обезбеђена помоћу *JWT* токена, чије се креирање врши уз помоћ *npm* пакета који се назива *JSONWebToken*.

Пријава и одјава корисника

Након што је корисник креирао налог омогућена је пријава. Приликом пријаве долази до провере унетих података са подацима из базе. Уколико се подаци не поклапају долази до приказа грешке која је добијена од сервера. У супротном корисник се прослеђује на почетну страну апликације, те се дешава ажурирање навигационог менија које сада садржи његов профил. Као што је горе већ речено, профил је заштићен помоћу токена. Након пријаве, на *web* претраживачу се чувају подаци о кориснику у *Local Storage*-у докле год је он пријављен. По одјави, подаци о кориснику се бришу као и друга чувана поља.

Ажурирање профила

Докле год је корисник пријављен може манипулисати подацима које је приложио. Та манипулација представља ажурирање профила и може се извршити на посебној страници преко навигационог менија. Ажурирање профила подразумева измену података

као што су корисничко име, *e-mail* адреса и шифра. За промену корисничког имена или *e-mail* адресе проверава се да ли исти већ постоје у бази, а приликом ажурирања лозинке потребно је нову лозинку унети два пута.

Рецензије

Могућност остављања и прегледа рецензије пружа кориснику додатно искуство приликом одабира производа, где може непосредно увидети искуства претходних купаца и видети да ли је овај производ стварно за њега. Сваки корисник може једном оставити оцену од 1 до 5 и опционо додати свој коментар, што се потом додаје у базу података и постаје видљиво осталим корисницима. Овим се просечна оцена аутоматски прерачунава за нову оцену. Ова опција омогућена је искључиво за оне кориснике који су пријављени.

Корпа

Корпа представља складиште производа које корисник одабере за своју наредну поруџбину. Уласком у корпу корисник може прегледати све производе које је одабрао. Уколико је одустао од куповине неког од артикала, исти може једноставно избацити кликом на икону кантице.

Поруџбине

Када је корисник задовољан са одабиром производа које жели да купи прелази на следећи корак а то је креирање поруџбине. Докле год поруџбина није плаћена дозвољено је отказивање исте. У било ком тренутку корисник може да провири статус своје поруџбине одласком на посебну секцију свог профила.

Плаћање

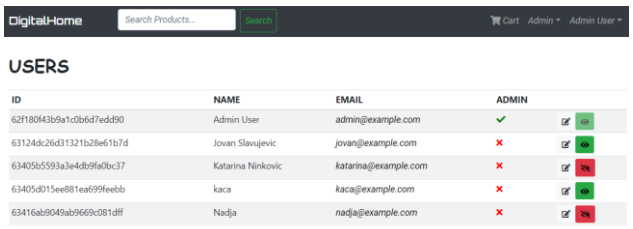
Уколико је креирао поруџбину, корисник може у било ком тренутку реализовати плаћање путем платног промета *Pay Pal* где се успешном транскацијом стање по аутоматизму мења у плаћено.

Администраторски мени

Поред корисника креиран је и администратор који управља сајтом. Пријављивањем као администратор отвара се ново поље у навигационом менију са називом админ. Ово поље садржи мени са секцијама корисници, производи и поруџбине. Администратор има могућност прегледа и манипулације подацима из тих секција.

Преглед и (де)активација корисника

Уласком у секцију корисника, отвара се преглед свих корисника и њихов тренутни статус.



Слика 3. Приказ листе корисника из админ менија

Администратор има могућности ажурирања профила корисника са могућношћу постављања неког од корисника за админа. Такође, администратор може деактивирати налог уколико то сматра потребним или га вратити у активно стање. Деактивираном кориснику је онемогућена пријава, а тиме и све могућности везане за пријаву.

Преглед и манипулација производима

Све креиране производе је могуће ажурирати, где су сва поља отворена за измену, али се могу и креирати потпуно нови продукти. Још једна од опција манипулацијом производа јесте њихово брисање.

Преглед, достављање и отказивање поруџбина

Администратор има на увид све поруџбине које су креиране и њихове статусе. Уколико неку од поруџбина није могуће извршити постоји могућност отказивања, али само у случају да је корисник није платио. Када се провери да ли је уплата прошла администратор може покренути процес доставе, чиме се ажурира стање поруџбине.

4. ЗАКЉУЧАК

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

Предлози за побољшање:

- Категоризација и филтрирање производа
- Подршка за *Android* и *iOS* кроз мобилне апликације
- *Chat bot* за добављање генеричких информација
- Додавање више слика на једном производу

5. ЛИТЕРАТУРА

[1] SHAH, DHRUTI NA. *Node .JS*. BPB PUBLICATIONS, 2018.
[2] Chodorow, Kristina. *MongoDB*. O'Reilly, 2013.
[3] Dunk, Jim. *React*. Van Duuren Media, 2019.
[4] Lee, Stan, and Jack Kirby. *Redux*. Tunbridge Wells, Kent: Panini Publishing, 2018.

Кратка биографија:

Катарина Нинковић рођена је у Руми 1996. год. Дипломски рад на Факултету техничких наука из области Електротехника и рачунарство – Примењена електроника одбранила је 2020.год. Области интересовања су јој развој софтвера у програмским језицима *Java* и *Javascript*.
контакт: *kaca.ninkovic.69@gmail.com*