Зборник радова Факултет техничких наука, Нови Сад

УДК: 004
ДОИ: https://doi.org/10.24867/28BE05Matkovic

ТРОСЛОЈНА АРХИТЕКТУРА КАО СОФТВЕРСКО РЕШЕЊЕ СИСТЕМА ЗА ЕВИДЕНЦИЈУ ТРЕНИНГА

THREE-TIER ARCHITECTURE AS A SOFTWARE SOLUTION FOR TRAINING TRACKING SYSTEMS

Јована Матковић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду описано је развијање целокупног система за евиденцију тренинга у спортским центрима и теретанама. Као софтверско решење имплементиране су веб и апликација за мобилне уређаје за тренере где се могу клијентима креирати налози и састављати планови тренинга. За ово решење коришћен је JavaScript програмски језик. Као хардверско решење имплементиран је мали ембедед уређај за клијенте који се може користити током тренинга за праћење вежби и здравствених параметара.*

Кључне речи: *Микроконтролер, веб апликација, Android апликација*

Abstract – *This project describes the development of the tracking training system for sports centers and gyms. Software solutions include a web platform and a mobile application for trainers, enabling them to create client accounts and design training plans. JavaScript programming language was used for this software solution. As a hardware solution, a small embedded device for clients was implemented, enabling them to track exercises and health parameters during training sessions.*

Keywords: *microcontroller, web application, Android application*

1. УВОД

Веб апликације у последње време стичу велику популарност развојем Интернет технологије и веома се често интегришу у разне пројекте због своје приступачности и једноставности. Све што је кориснику у улози клијента потребно за коришћење овог типа апликација јесте веб прегледач. На овај начин се избегава ажурирање и одржавање апликације без потребе за инсталацијом и дистрибуцијом софтвера на различите рачунаре тих корисника што значајно олакшава коришћење без потребе за компликованим процесима инсталације софтвера на њиховим уређајима. Успостављање веб апликација донело је брзину развоја и лако ажурирање у системе који захтевају мало времена за имплементацију измена и исправки.

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

Важно је поменути и да, за разлику од старијих метода које се ослањају на инсталацију софтвера на уређајима, код веб апликација за ажурирање на целокупном скупу уређаја је довољно извршити измене само на серверу.

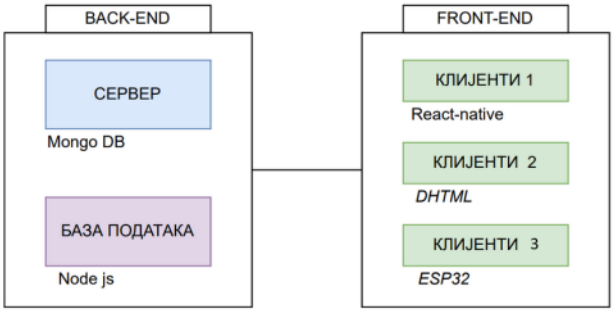
Са друге стране у свакодневној употреби данашњег човека су и ембедед системи чија је имплементација на доста нижем нивоу апстракције због употребе микроконтролера и њихове могућности су специјализоване за одређену примену што доводи до веће ефикасности приликом извршавања жељених функција. Међутим такав дизајн је за крајњег корисника доста ограничавајући јер не оставља превише могућности за персонализацију и интеракцију уопште. Из тог разлога све више се развијају техологије попут IoT (*Internet of Things*) које омогућавају повезивање ових уређаја са веб апликацијама за постизање једноставније контроле и надзора специјализованих ембедед уређаја путем Интернета што крајњем кориснику омогућава већу персонализацију и интеракцију [1].

Идеја овога пројекта јесте повезати поменуте две технологије у једну целину која налази примену у спортским центрима и теретанама где чланови добијају уређаје за евиденцију и праћење тренинга које им тренери задају преко мобилне web апликације.

2. КОНЦЕПТ РЕШЕЊА

За израду веб апликације користи се трослојна архитектура која се састоји из три слоја - презентационог (кориснички интерфејс), слоја пословне логике и слоја података (база за складиштење података). Овакав систем чија је шема приказана на слици 1. је реализован у два блока од којих један представља серверску (енг. *Back-end*), а други клијентску (енг. *Front-end*) страну. Серверска страна је реализована помоћу *Node.js* платформе где је имплементирана апликација коришћењем Javascript програмског језика и *Express.js back-end web* application *framework*-а за *node.js* платформу [2]. Што се тиче клијентске стране, ту постоје два типа клијената, од којих се први тип односи на мобилну апликацију реализовану уз помоћ *React Native framework*-а коришћењем Javascript програмског језика, док други тип клијента подразумева ембедед уређај базиран на *ESP32* модулу који је испрограмиран коришћењем C програмског језика. Оба клијента се повезују са сервером како би послали или прикупили релевантне податке о тренингу.

Сервер на почетку свог рада успоставља конекцију са *MongoDB Atlas cloud* базом података. Током даљег рада сервер чита и уписује податке у ту базу на захтеве клијената. Предвиђено је да сервер и клијенти буду повезани на исту локалну мрежу како би се комуникација са сервером успоставила помоћу његове локалне *ip* адресе, мада постоји могућност и да се web страница *host*-ује па да се онда она користи као приступна тачка.



Слика 1. *Блок шема целокупног система*

2.1. Серверска апликација

Серверска апликација се састоји од базе података, модела, рута и контролера и замишљено је да се покреће на рачунару спортског центра који жели да омогући коришћење овог система својим корисницима. Сврха ове апликације јесте извршавање логике целог система и манипулација са подацима које она прима од клијената или их шаље клијентима.

Клијентска апликација за мобилне уређаје користи *HTTP* комуникациони протокол за повезивање са сервером, док се ембедед апликација са серверм повезује преко тзв. *Websocket* протокола [3].

Услов за коришћење серверске апликације јесте да рачунар буде повезан на локалну мрежу и да има приступ интернету. Када је реч о *HTTP* захтевима, клијентски захтеви се даље обрађују помоћу рута креираних за одређену крајњу тачку како би се лакше разазнале врсте захтева и упита. За обраду овог типа захтева користи се *Express.js* који представља слој изграђен поврх *Node.js* и помаже у управљању сервиса и рута. Са друге стране, клијенти који се повезују преко *Websocket* протокола након конекције поруке размењују преко једног те истог канала, тако да нема нових захтева након једном успостављене конекције и за то се користи посебан *ws* пакет. Током комуникације са сервером клијенти могу да шаљу или примају податке. Ови подаци се складиште у *MongoDB Atlas cloud* бази података, а да би се сервер повезао са њом, потребан му је приступ Интернету.

За приступ бази и извршавање упита *Node js* поседује пакет *Mongoose* са једноставним интерфејсом за објектно моделирање података (*Object Data Modeling - ODM*) [4]. Треба имати у виду да *MongoDB* спада у нерелациону врсту базе података (*NoSQL*) што значи да се подаци не морају складиштити по утврђеним шемама. Главна предност у односу на релационе базе поодатака јесте могућност складиштења података у JSON формату где структура може да варира.

JSON представља отворени стандардни формат датотеке и формат за размену података који користи обичан текст за складиштење и пренос објеката који се састоје од парова атрибут-вредност и низова или других вредности које се могу серијализовати.

Типови података за складиштење у базу дефинишу се у моделима који представљају шеме са атрибутима које један модел поседује. Преко поменутих модела даље контролери шаљу захтеве ка бази или креирају упите са жељеним параметрима претраге.

Разлог зашто је за реализацију серверске апликације коришћена баш платформа *Node.js* јесте могућност коришћења *Javascript* програмског језика за програмирање који је раније био намењен искључиво клијентској страни с обзиром на своју употребу ограничену само на прегледач. На овај начин постиже се једноставност имплементације јер је довољно познавање само *Javascript* програмског језика и за креирање клијентске и за креирање серверске апликације и сви новији веб прегледачи га подржавају.

Треба поменути да је *Javascript* скриптни језик који садржи *API* са рад са текстом, низовима, датумима и регуларним изразима, али не и улазно/излазне функционалности, као што су повезивање, складиштење података или графичке функционалности, за шта се ослања на окружење у коме се извршава.

2.2. Клијентске апликације

У пројекту постоји три типа клијената:

1. Веб апликација
2. Мобилна апликација
3. Ембедед апликација

Клијентска мобилна апликација је реализована помоћу *React Native* који представља *open-source framework* за графички дизајн. Овај framework покреће позадински процес који интерпретира Javascript директно на уређају крајњег корисника и комуницира са *native* платформом путем серијализованих података коришћењем асинхроних мостова. Користи се за развој *cross-platform* апликација и не захтева познавање *HTML*-а и *CSS*-а. Мобилна апликација намењена је тренерима и самосталним вежбачима који састављају планове тренинга. Корисници ове апликације имају могућност да направе налог који ће се користити на ембедед уређају и да за њега убележе све потребне информације о распореду вежби и броју понављања и серија на једном тренингу. Корисник може да креира, ажурира или обрише тренинг тако што се преко *HTTP* протокола шаље захтев серверу са информацијама које је потребно модификовати у бази. Овај протокол је изворно дизајниран за преузимање *HTML* страница у клијент-сервер окружењу, широко се користи за веб и одликује га то да је сваки захтев од клијента до сервера независан. Клијент је увек тај који шаље захтев, а сервер одговара на тај захтев и размена података се врши у читљивом текстуалном формату.

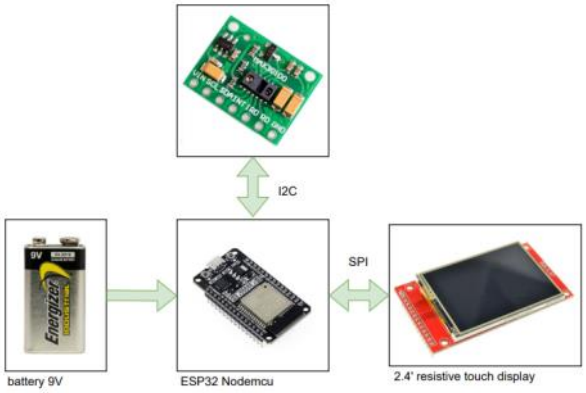
Свака трансакција је независна, што значи да се веза затвара након што је одговор послат. Захтева много заглавља и метаподатака у својим порукама и обично се користи за претраживање интернета помоћу *REST API* који укључује извршавање *GET, POST, PUT* и *DELETE* захтева. Треба поменути да је за овај тип клијената потребно реализовати и веб апликацију која

имплементира идентичну функционалност како би корисници могли и преко веб прегледача да приступе истим ресурсима у случају да не желе да користе *Android* ili *iOS* уређај.
За креирање веб страница коришћен је *HTML* (*HyperText Markup Language*) који представља описни језик намењен креирању веб садржаја. *HTML* користи тагове да означи различите елементе на веб страници попут наслова, параграфа, слика итд.
Поменути тагови имају структуру која описује како садржај треба да се прикаже и организује на страници тако да се може рећи да *HTML* представља складиште информација које прегледачи читају и преводе у видљиве веб странице за кориснике. Како *HTML* има веома ограничене могућности у погледу дефиниције изгледа, структура и садржај се форматирају помоћу *CSS* дескриптивног језика.
У комбинацији са Javascript-ом *HTML* и *CSS* чине *DHTML* (*Dynamic HTML*).

Клијентска апликација која се покреће на ембедед уређају од сервера тражи податке о тренингу како би их исписала кориснику на дисплеј. То омогућава кориснику да провери коју вежбу треба да ради на који начин и колико понављања. Дисплеј који се користи за интеракцију са корисником садржи резистивни панел осетљив на додир. На почетку коришћења апликације, потребно је да се вежбач улогује на свој налог како би могао видети свој план тренинга.
Након сваког сета могуће је означити да је вежба одрађена и након тога тајмер одбројава паузу пре него што на дисплеју прикаже наредну серију/вежбу. Поред тога овај ембедед уређај поседује *MAX30102* сензор откуцаја срца и кисеоника у крви, које је по потреби могуће мерити и у реалном времену послати на *cloud*. Као што је већ поменуто ова врста клијента се са сервером повезује преко *Websocket.*
Под појмом *Websocket* се подразумева клијент-сервер комуникациони протокол који се извршава преко *TCP* (*Transmission Control Protocol*). Главна разлика између *WebSocket* и *HTTP* протокола јесте да је *HTTP* конекцијски оријентисан, док WebSocket обезбеђује комуникацију преко једне дугометражне *TCP* везе.
Овај протокол, иако мало захтевнији од *HTTP* протокола, омогућава бидирекциону *full-duplex* размену порука између клијента и сервера. Веома је поуздан и ефикаснији је од *HTTP* протокола јер се отвара једна веза која се одржава током целог трајања једне сесије. Погодан је за апликације које захтевају стална ажурирања и размену порука у реалном времену.

3. ХАРДВЕРСКИ ДЕО СИСТЕМА

На слици 2 приказана је блок шема хардверског уређаја са свим деловима који га чине. Хардверски део базира се на ESP32 модулу који поседује *Xtensa LX6* микроконтролер. Поред тога контролер је повезан и са сензором *MAX30102* и комуникација се одвија преко *I2C* протокола.
Уређај се напаја 9V батеријом и поседује прекидач за укључење напајања.



Слика 2. Блок шема хардверског дела система

4. СОФТВЕРСКИ ДЕО СИСТЕМА

4.1. *C* апликација

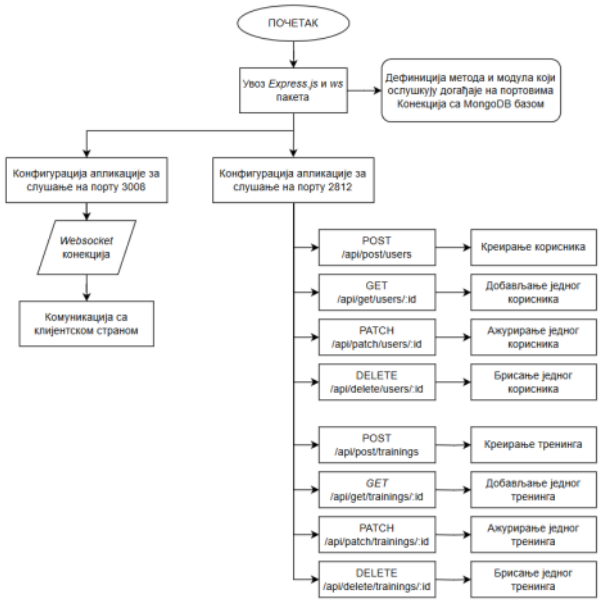
На слици 3 приказан је алгоритам рада *main* функције C апликације.

Слика 3. Алгоритам рада *main* функције *C* апликације

Функција почиње иницијализацијом потребних периферија која је дефинисана у увезеним библиотекама. Након тога улази се *loop* петљу која се стално понавља, и проверава кориснички унос на екрану осетљивом на додир како би се преузеле одређене акције.

Прво је потребно да корисник унесе корисничко име и шифру. Уређај се повезује са сервером како би проверио да ли је унос успешан и ако јесте, уређај од сервера тражи податке о тренингу. Током тренинга се подаци ажурирају у реалном времену, и могуће је помоћу сензора очитати податке о телесној температури, пулсу и нивоу кисеоника у крви.

4.2. Веб апликација - серверски део



Slika 4. Алгоритам рада серверске апликације

Програмски код главног модула серверске апликације изгледа тако што се прво се учитавају пакети *ws* и Express.js који омогућавају комуникацију путем *Websocket* и *HTTP* протокола. *HTTP* захтеви клијената се даље обрађују кроз *MVC* (*Module*, *View*, *Controller*) приступ на порту 2812 [5].
Захтеви се обрађују помоћу дефинисаних рута и организују помоћу модела док је логика за рад са базом података имплементирана у контролерима. На порту 3108 се успоставља *Websocket* конекција и одржава се док год је ембедед уређај активан.

5. РЕЗУЛТАТИ

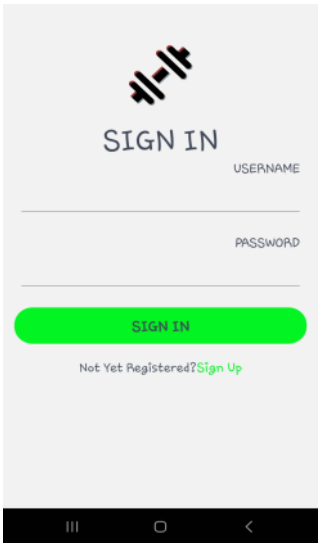
5.1. Ембедед апликација

На слици 5. приказан је хардвер из ког се састоји ембедед уређај за праћење тренинга. На дисплеју се види почетни приказ за пријаву корисника.

Слика 5. Хардверско решење

5.2. *Android* и веб апликација

Android и веб апликација имају исту намену, главна разлика је у платформи за коју су предвиђене. На слици 6. приказана је почетна страница *Android* апликације



Слика 6. Почетна страна *Android* апликације

6. LITERATURA

[1] Шта је *Internet of Things*, https://samoobrazovanje.rs/sta-je-internet-of-things-internet-inteligentnih-uredjaja/, последњи приступ децембар 2023.
[2] Шта је *Node*? https://www.skolaprogramiranja.rs/sta-je-to-node/, poslednji pristup oktobar 2023.
[3] *WebSocket* vs. *HTTP* https://sendbird.com/developer/tutorials/websocket-vs-http-communication-protocols последњи приступ децембар 2023.
[4] *MongoDB* и *Mongoose* https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/, последњи приступ децембар 2023.
[5] Шта је *Model-View-Controller* https://blog.logrocket.com/building-structuring-node-js-mvc-application/ последњи приступ децембар 2023.

Кратка биографија:

Јована Матковић рођена је 28.XII 1999. године. Дипломски рад на Факултету техничких наука из области Мехатронике – Импулсна електроника одбранила је у септембру 2022. године. Има годину дана радног искуства, најпре у компанији *Typhoon-HIL*, а потом у компанији *ARS-Embedded Systems LLC*.
Контакт: jovana.matkovic99@gmail.com